CS-356 Computer Networks Lab
Lab Session -2

Application Layer

Capture HTTP Password

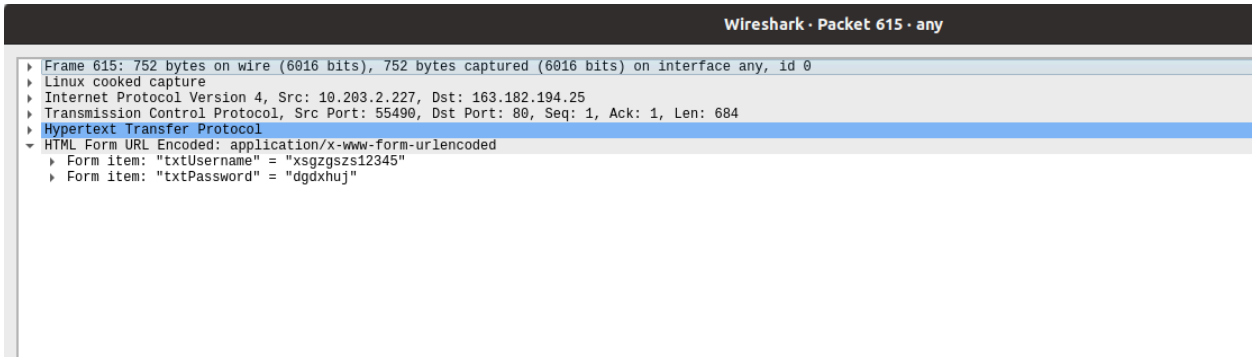
HTTP, which stands for HyperText Transfer Protocol, is a fundamental protocol used for communication on the World Wide Web. It is an application layer protocol that facilitates the transfer of data between a web server and a web browser.

HTTP transmits data in plain text, which means that any data exchanged between your browser and the web server can be intercepted and read by malicious actors.

Without encryption, sensitive information such as login credentials, personal details, or financial data can be easily captured by eavesdroppers.

website: http://www.vbsca.ca/login/login.asp

```
filter:
http.request.method == "POST"
```




```
filter packets with TCP ports
tcp.port == 80 or tcp.port==443
```

Wireshark cannot see application data because it is encrypted with TLS. That's why Wireshark uses the TLS and TLS version in the protocol column instead of HTTPS.

```
▸ Frame 480: 194 bytes on wire (1552 bits), 194 bytes captured (1552 bits) on interface enp0s31f6, id 0
▸ Ethernet II, Src: ExtremeN_9f:82:ec (00:04:96:9f:82:ec), Dst: 08:92:04:d5:bc:38 (08:92:04:d5:bc:38)
▸ Internet Protocol Version 4, Src: 142.250.192.78, Dst: 10.203.2.227
▸ Transmission Control Protocol, Src Port: 443, Dst Port: 37436, Seq: 802, Ack: 3077, Len: 128
▸ Transport Layer Security
```

Linux command related to DNS

Linux command to find out details of the local network:

```
nmcli dev show
```

```
cse@cse-OptiPlex-7000:~$ nmcli dev show
GENERAL.DEVICE:                         enp0s31f6
GENERAL.TYPE:                           ethernet
GENERAL.HWADDR:                         08:92:04:D5:BD:98
GENERAL.MTU:                            1500
GENERAL.STATE:                          100 (connected)
GENERAL.CONNECTION:                     Wired connection 1
GENERAL.CON-PATH:                       /org/freedesktop/NetworkManager/ActiveC>
WIRED-PROPERTIES.CARRIER:               on
IP4.ADDRESS[1]:                         10.203.4.240/20
IP4.GATEWAY:                            10.203.1.1
IP4.ROUTE[1]:                           dst = 0.0.0.0/0, nh = 10.203.1.1, mt = >
IP4.ROUTE[2]:                           dst = 10.203.0.0/20, nh = 0.0.0.0, mt =>
IP4.ROUTE[3]:                           dst = 169.254.0.0/16, nh = 0.0.0.0, mt >
IP4.DNS[1]:                             172.16.2.248
IP4.DNS[2]:                             172.16.2.249
IP4.DNS[3]:                             10.100.170.1
IP4.DOMAIN[1]:                          iiti.ac.in
IP6.ADDRESS[1]:                         fe80::a295:715d:5e2b:f4b7/64
IP6.GATEWAY:                            --
IP6.ROUTE[1]:                           dst = fe80::/64, nh = ::, mt = 100
```

To find the location of the DNS server:

```
cse@cse-OptiPlex-7000:~$ nmcli dev show | grep 'IP4.DNS'
IP4.DNS[1]:                             172.16.2.248
IP4.DNS[2]:                             172.16.2.249
IP4.DNS[3]:                             10.100.170.1
```

dig

The dig command in Linux is used to gather DNS information.

Most modern Linux systems include the dig command.

```
sudo apt-get install dnsutils
```

The dig command enables searching for a domain name. To perform a DNS lookup, open the terminal and type:

```
dig google.com
```

```
(base) gaurav@gaurav-OptiPlex-7000:~$ dig google.com

; <<>> DiG 9.16.1-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5934
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             226     IN      A       142.250.192.110

;; Query time: 4 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Wed Jan 17 23:00:52 IST 2024
;; MSG SIZE  rcvd: 55
```

```
dig @8.8.8.8 google.com
```

DNS is the system used to look up addresses.

When you type an address into the bar in Chrome such as www.example.com then your browser contacts a DNS server to find out the IP address of that address. It is this function the 8.8.8.8 provides.

```
(base) gaurav@gaurav-OptiPlex-7000:~$ dig @8.8.8.8 google.com

; <<>> DiG 9.16.1-Ubuntu <<>> @8.8.8.8 google.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16418
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             125     IN      A       142.251.42.14

;; Query time: 356 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Wed Jan 17 23:00:57 IST 2024
;; MSG SIZE  rcvd: 55
```

nslookup

```
(base) gaurav@gaurav-OptiPlex-7000:~$ nslookup iiti.ac.in
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   iiti.ac.in
Address: 172.16.1.20
```

host

It is used to find the IP address of a particular domain name, or if you want to find out the domain name of a particular IP address the host command becomes handy.

```
(base) gaurav@gaurav-OptiPlex-7000:~$ host 172.16.1.20
20.1.16.172.in-addr.arpa domain name pointer adsa2020.iiti.ac.in.
20.1.16.172.in-addr.arpa domain name pointer iiti.ac.in.
20.1.16.172.in-addr.arpa domain name pointer threatweb.iiti.ac.in.
20.1.16.172.in-addr.arpa domain name pointer machineintelligence.iiti.ac.in.
20.1.16.172.in-addr.arpa domain name pointer fluxus.iiti.ac.in.
20.1.16.172.in-addr.arpa domain name pointer icc.iiti.ac.in.
20.1.16.172.in-addr.arpa domain name pointer metacryst.iiti.ac.in.
20.1.16.172.in-addr.arpa domain name pointer sociology.iiti.ac.in.
20.1.16.172.in-addr.arpa domain name pointer alumnicell.iiti.ac.in.
```

The virtual hosts serve multiple hostnames on a single machine with a single IP address. This is possible because when a web browser requests a resource from a web server using HTTP/1.1 it includes the requested hostname as part of the request. The server uses this information to determine which website to show the user.

Transport Layer

List of listening ports on a Linux system

netstat

It gives an overview of network activities and displays which ports are open or have established connections

```
(base) gaurav@gaurav-OptiPlex-7000:~$ netstat | grep tcp
tcp        0      0 gaurav-OptiPlex-7:35216 stackoverflow.com:https ESTABLISHED
tcp        0      0 gaurav-OptiPlex-7:37436 bom12s16-in-f14.1:https ESTABLISHED
tcp        0      0 gaurav-OptiPlex-7:56324 ec2-3-109-176-237:https ESTABLISHED
tcp        0      0 gaurav-OptiPlex-7:43786 150.179.244.35.bc:https ESTABLISHED
tcp        0      0 gaurav-OptiPlex-7:55588 pnbomb-aa-in-f14.:https ESTABLISHED
tcp        0      0 gaurav-OptiPlex-7:50488 server-18-164-237:https ESTABLISHED
tcp        0      0 gaurav-OptiPlex-7:38828 bom07s29-in-f4.1e:https CLOSE_WAIT
tcp        0      0 gaurav-OptiPlex-7:51326 bom12s13-in-f14.1:https ESTABLISHED
tcp        0      0 gaurav-OptiPlex-7:48428 172.67.40.29:https      ESTABLISHED
tcp        0      0 gaurav-OptiPlex-7:45016 si-in-f188.1e100.:https ESTABLISHED
tcp        0      0 gaurav-OptiPlex-7:54212 172.64.141.13:https     ESTABLISHED
tcp        0      0 gaurav-OptiPlex-7:37348 bom07s32-in-f5.1e:https ESTABLISHED
tcp        0      0 gaurav-OptiPlex-7:52848 181.214.120.34.bc:https ESTABLISHED
tcp        0      0 gaurav-OptiPlex-7:42050 whatsapp-cdn-shv-:https ESTABLISHED
tcp        0      0 gaurav-OptiPlex-7:35720 sl-in-f84.1e100.n:https ESTABLISHED
tcp        0      0 gaurav-OptiPlex-7:46444 server-3-160-196-:https ESTABLISHED
tcp        0      0 gaurav-OptiPlex-7:45012 si-in-f188.1e100.:https ESTABLISHED
tcp        0      0 gaurav-OptiPlex-7:59552 stackoverflow.com:https ESTABLISHED
tcp        0      0 gaurav-OptiPlex-7:51782 82.75.98.34.bc.go:https ESTABLISHED
tcp        0      0 gaurav-OptiPlex-7:33534 relay-7f353b5b.ne:https ESTABLISHED
tcp        0      0 gaurav-OptiPlex-7:34872 stackoverflow.com:https ESTABLISHED
tcp        0      0 gaurav-OptiPlex-7:57192 30.42.36.34.bc.go:https ESTABLISHED
tcp       64      0 gaurav-OptiPlex-7:46426 ec2-52-40-196-44.:https CLOSE_WAIT
tcp       64      0 gaurav-OptiPlex-7:40080 ec2-52-40-196-44.:https CLOSE_WAIT
tcp        0      0 gaurav-OptiPlex-7:55310 bom07s45-in-f14.1:https ESTABLISHED
tcp        0      0 gaurav-OptiPlex-7:46436 server-3-160-196-:https ESTABLISHED
```

`netstat -tulpn`

- Options:
  - `-t`: Display TCP connections.
  - `-u`: Display UDP connections.
  - `-l`: Show only listening sockets.
  - `-p`: Display the process ID and name.
  - `-n`: Show numerical addresses instead of resolving hostnames.

```
(base) gaurav@gaurav-OptiPlex-7000:~$ sudo netstat -tulpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1195/sshd: /usr/sbi
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      921/systemd-resolve
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      9294/cupsd
tcp        0      0 127.0.0.1:42079         0.0.0.0:*               LISTEN      1190/containerd
tcp        0      0 0.0.0.0:7070            0.0.0.0:*               LISTEN      1186/anydesk
tcp6       0      0 :::22                   :::*                    LISTEN      1195/sshd: /usr/sbi
tcp6       0      0 ::1:631                 :::*                    LISTEN      9294/cupsd
udp        0      0 0.0.0.0:35392           0.0.0.0:*                           1018/avahi-daemon:
udp        0      0 127.0.0.53:53           0.0.0.0:*                           921/systemd-resolve
udp        0      0 0.0.0.0:631             0.0.0.0:*                           9295/cups-browsed
udp        0      0 0.0.0.0:50001           0.0.0.0:*                           1186/anydesk
udp        0      0 224.0.0.251:5353        0.0.0.0:*                           5423/chrome
udp        0      0 0.0.0.0:5353            0.0.0.0:*                           1018/avahi-daemon:
udp6       0      0 :::54253                :::*                                1018/avahi-daemon:
udp6       0      0 :::5353                 :::*                                1018/avahi-daemon:
(base) gaurav@gaurav-OptiPlex-7000:~$
```

Assignment for Lab Session 2:

1. Study the following documentation about Scapy: https://scapy.readthedocs.io/en/latest/
2. Find out the Mac addresses of your neighbouring friend's system (using the Linux commands discussed in the previous lab).
3. Use Scapy to create frames with the MAC address information and send it to your friend.

Installing Scapy:

```
pip install scapy
```

```
cse@cse-OptiPlex-7000:~$ pip install scapy
Defaulting to user installation because normal site-packages is not writeable
Collecting scapy
  Downloading scapy-2.5.0.tar.gz (1.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 11.1 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: scapy
  Building wheel for scapy (setup.py) ... done
  Created wheel for scapy: filename=scapy-2.5.0-py2.py3-none-any.whl size=1444325 sha256=2b79301764f86aad1f06f82804643ed34570cf368b3619db1734efbdbabf83d9
  Stored in directory: /home/cse/.cache/pip/wheels/8a/5e/7c/6f92df559dadf49d0e31cecdd5b104fc8c38a7c67fde16b029
Successfully built scapy
Installing collected packages: scapy
Successfully installed scapy-2.5.0
cse@cse-OptiPlex-7000:~$
```

Creating a packet/frame:

Create a frame

```
myFrame = Ether()/IP()
myFrame
```

```
myFrame.show()
```

Sniff packets

```
packets = sniff(count=2)
packets
packets[0]
packets[1]

## summary() provide minimal information regarding the packet
collection
packets.summary()
```

```
myFrame[IP]
myFrame[Ether].src
```

Function to print src mac addr

```
def printMac(frame):
     print(frame[Ether].src)


printMac(myFrame)
```

Sniff packets and show src mac addr

```
sniff(count=2, prn=printMac)
```

create a file sendP.py

```
from scapy.all import send, IP, ICMP

send(IP(src="10.203.2.227",dst="10.203.2.227")/ICMP()/"Hello World 123")
```

```
python sendP.py
```