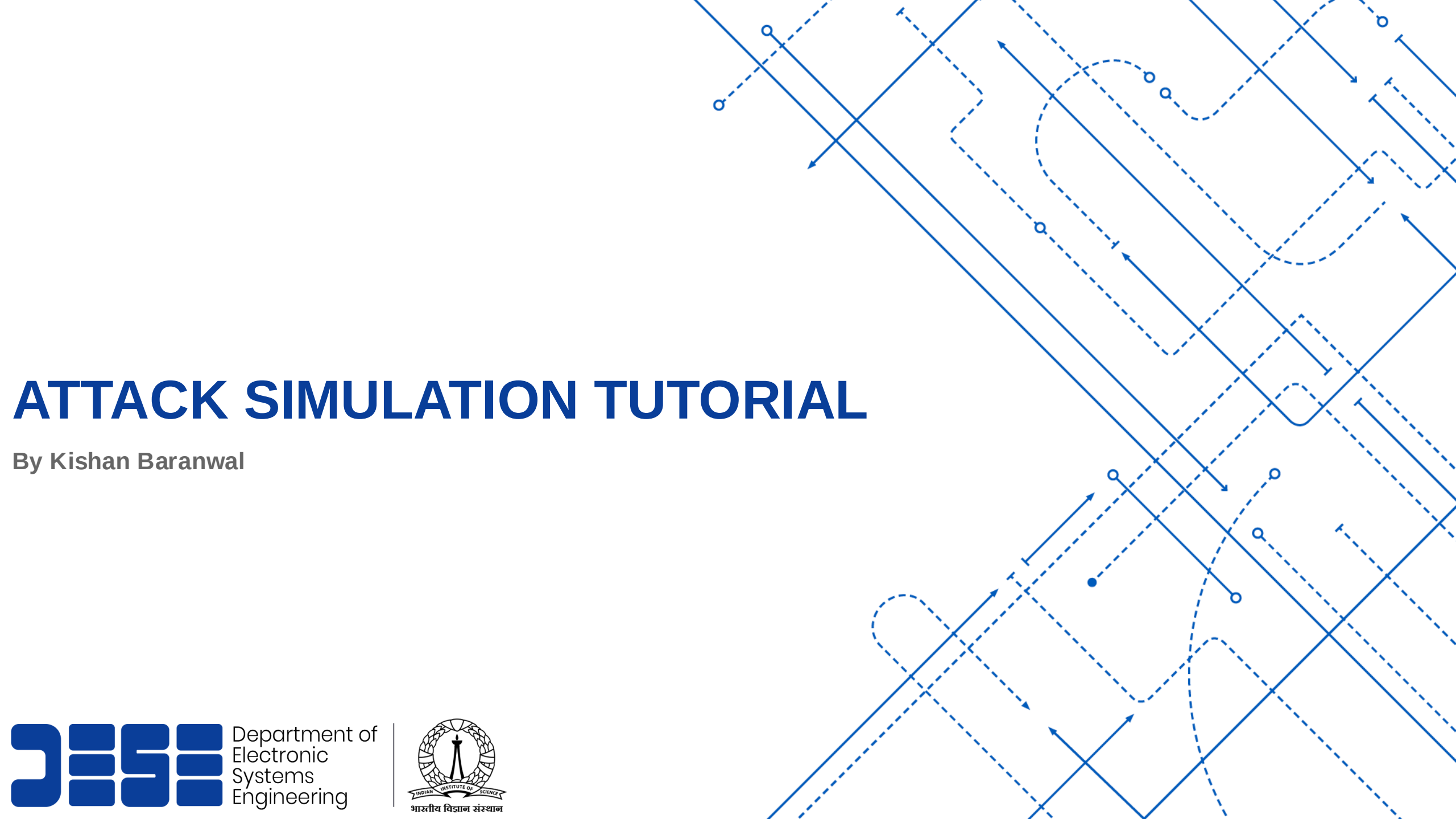


ATTACK SIMULATION TUTORIAL

By Kishan Baranwal

Department of Electronic Systems Engineering

Indian Institute of Science
भारतीय विज्ञान संस्थान

Pre-installation

- Wireshark -
  - Linux
    - https://www.youtube.com/watch?v=wuH0bGTusXU
  - Windows
    - https://www.youtube.com/watch?v=L7Q2ZSkgHF4
- Tshark - sudo apt-get install tshark
  - https://www.youtube.com/watch?v=21M1rnbd5xo
- Scapy – pip install scapy
- **Watch before coming to lab -**
  - https://www.youtube.com/watch?v=Lb-PJl9u3z8&t=135s
  - https://www.youtube.com/watch?v=yD8qrP8sCDs

- Better to install on ubuntu or kali Linux
  - VM or laptop with dual boot
- Download one drive link folder before coming to lab
  - https://indianinstituteofscience-my.sharepoint.com/:f:/g/personal/kishankumar_iisc_ac_in/Ej3ASob4VThBkhWNQ6q63XABILxKJwsXBVIUUXRcs6Gnbw?e=hjy6sc

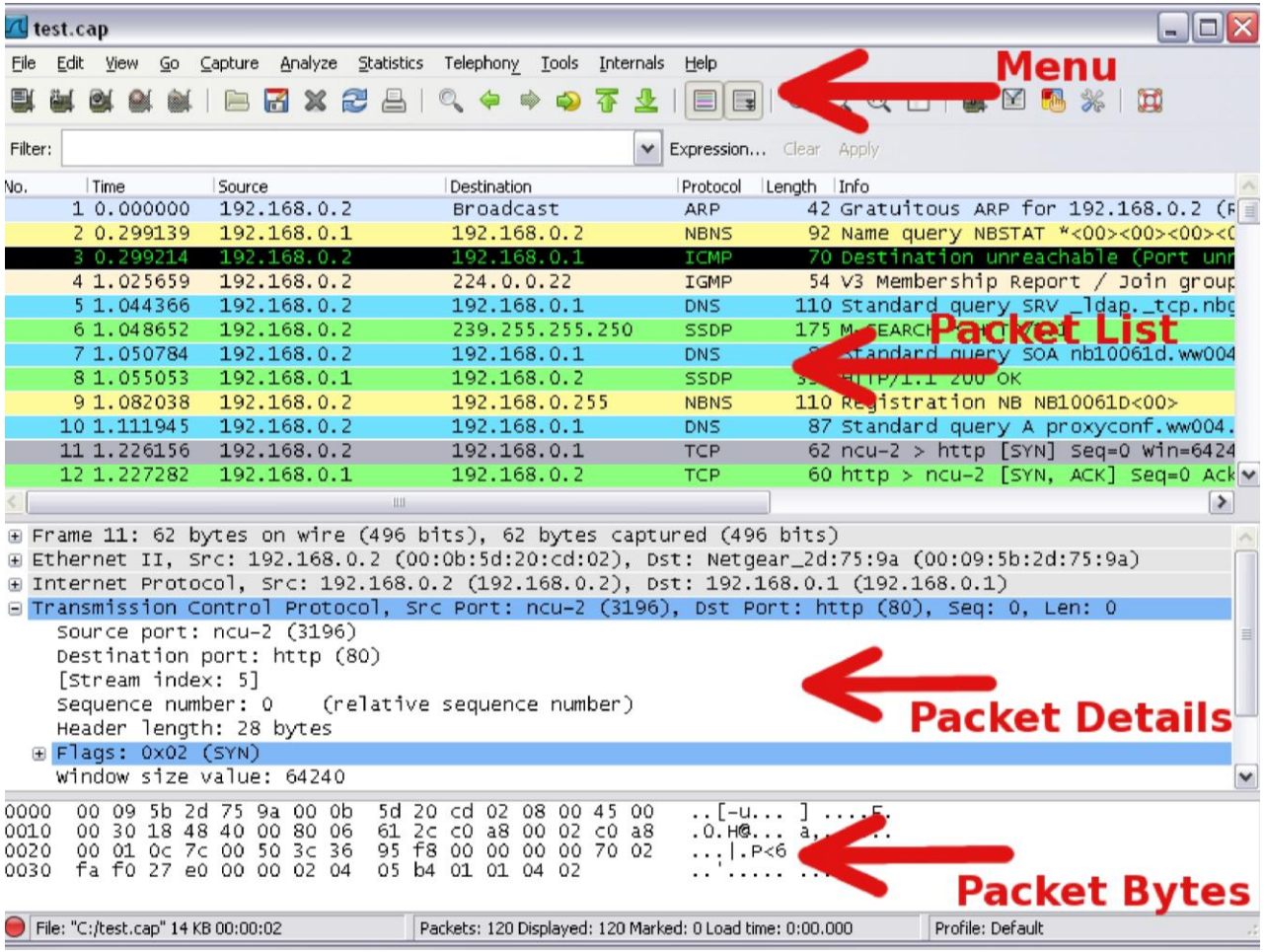
NETWORK ANALYSIS

Department of Electronic Systems Engineering

INDIAN INSTITUTE OF SCIENCE

भारतीय विज्ञान संस्थान

Introduction to Wireshark



Download link for pcap file-

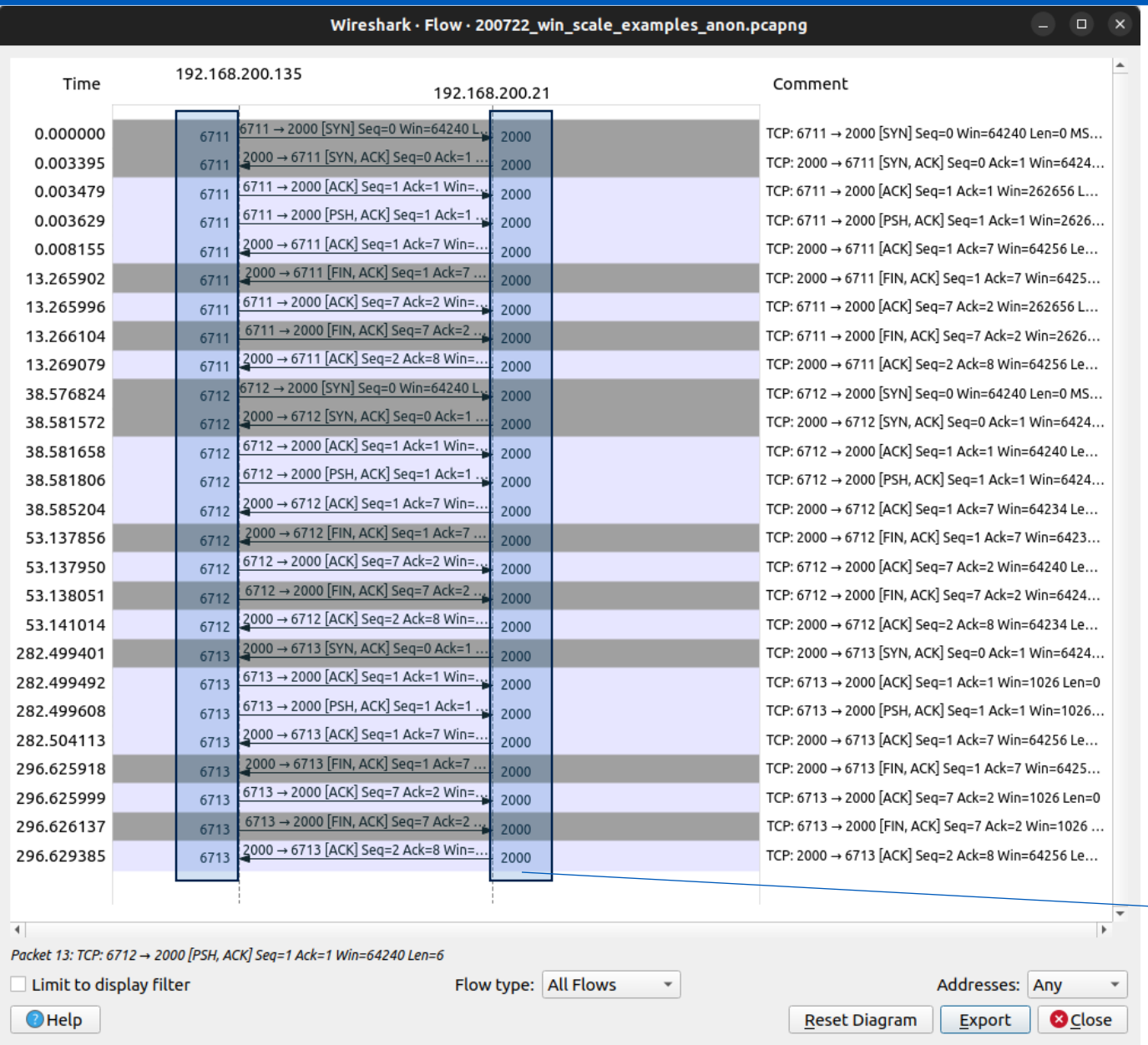
example.pcapng in drive folder

or

https://wiki.wireshark.org/uploads/__moin_import__/attachments/SampleCaptures/200722_win_scale_examples_anon.pcapng

Reference -
https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html

Understand TCP handshake in Wireshark

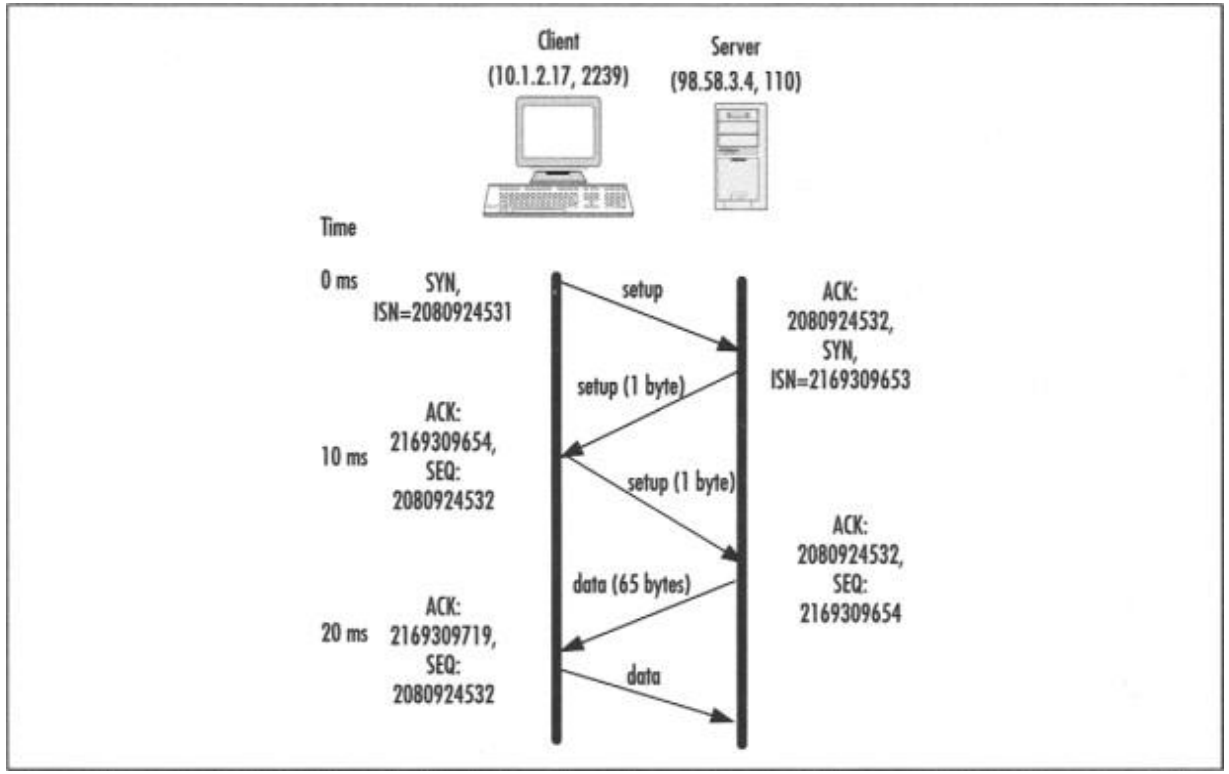


Statistics -> Flow graph

port

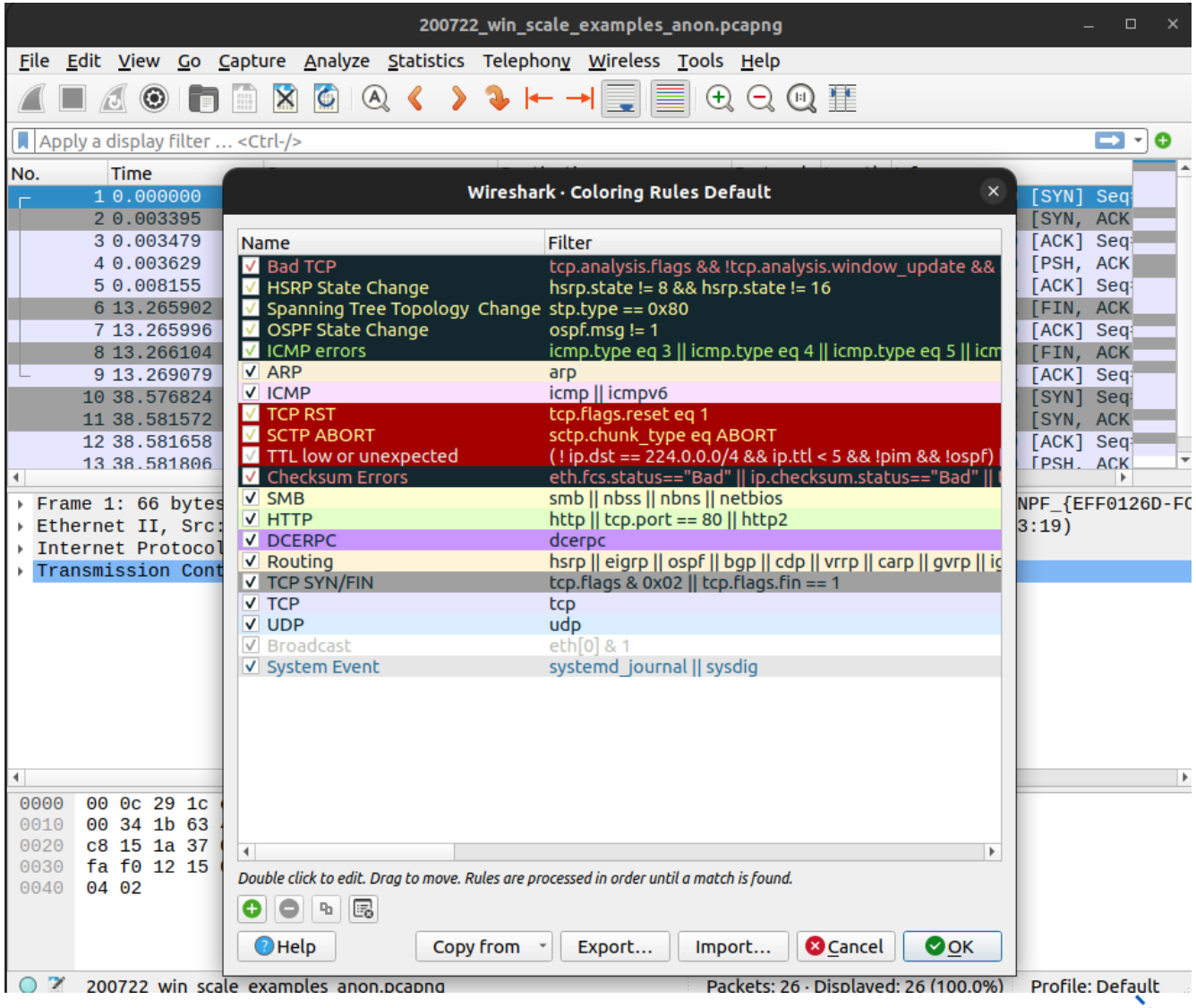
Understand TCP handshake in Wireshark

Time	192.168.200.135		192.168.200.21	Comment
0.000000	6711	SYN →	2000	Seq = 0
0.003395	6711	← SYN, ACK	2000	Seq = 0 Ack = 1
0.003479	6711	ACK →	2000	Seq = 1 Ack = 1
0.003629	6711	PSH, ACK - Len: 6 →	2000	Seq = 1 Ack = 1
0.008155	6711	← ACK	2000	Seq = 1 Ack = 7
13.265902	6711	← FIN, ACK	2000	Seq = 1 Ack = 7
13.265996	6711	ACK →	2000	Seq = 7 Ack = 2
13.266104	6711	FIN, ACK →	2000	Seq = 7 Ack = 2
13.269079	6711	← ACK	2000	Seq = 2 Ack = 8
38.576824	6712	SYN →	2000	Seq = 0
38.581572	6712	← SYN, ACK	2000	Seq = 0 Ack = 1
38.581658	6712	ACK →	2000	Seq = 1 Ack = 1
38.581806	6712	PSH, ACK - Len: 6 →	2000	Seq = 1 Ack = 1
38.585204	6712	← ACK	2000	Seq = 1 Ack = 7
53.137856	6712	← FIN, ACK	2000	Seq = 1 Ack = 7
53.137950	6712	ACK →	2000	Seq = 7 Ack = 2
53.138051	6712	FIN, ACK →	2000	Seq = 7 Ack = 2
53.141014	6712	← ACK	2000	Seq = 2 Ack = 8
282.499401	6713	← SYN, ACK	2000	Seq = 0 Ack = 1
282.499492	6713	ACK →	2000	Seq = 1 Ack = 1
282.499608	6713	PSH, ACK - Len: 6 →	2000	Seq = 1 Ack = 1
282.504113	6713	← ACK	2000	Seq = 1 Ack = 7
296.625918	6713	← FIN, ACK	2000	Seq = 1 Ack = 7
296.625999	6713	ACK →	2000	Seq = 7 Ack = 2
296.626137	6713	FIN, ACK →	2000	Seq = 7 Ack = 2
296.629385	6713	← ACK	2000	Seq = 2 Ack = 8



Filter

- Enter filter field
  - && - logical AND
  - || - logical OR
  - ! - logical NOT
- Filtered packet color (View -> coloring rules)
  - Create new color for "ip.dst == 127.0.1"



#installation of tshark

Superuser admin rights
Require password after command

sudo apt-get install tshark

#find your network interface
ifconfig

Ubuntu

```
enp158s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.114.56.157  netmask 255.255.255.0  broadcast 10.114.56.255
        inet6 fe80::5690:4886:d71c:e514  prefixlen 64  scopeid 0x20<link>
        ether 00:1b:21:39:34:81  txqueuelen 1000  (Ethernet)
        RX packets 81352826  bytes 97394095561 (97.3 GB)
        RX errors 0  dropped 2029  overruns 0  frame 0
        TX packets 31895678  bytes 11934623124 (11.9 GB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 100  memory 0xc5ec0000-c5ee0000
```

Open terminal in separate file location

#capture live traffic in csv

Wireshark terminal version tool

sudo tshark -i enp158s0 -a duration:60 -T fields -E header=y -E separator=, -e frame.number -e eth.src -e eth.dst -e eth.type -e frame.len > ethernet_traffic.csv

#capture live traffic in csv

Listen to network for 60 seconds

```
sudo tshark -i enp158s0 -a duration:60 -T fields -E header=y -E separator=, -e frame.number -e eth.src -e eth.dst -e eth.type -e frame.len > ethernet_traffic.csv
```

#capture live traffic in csv

```
sudo tshark -i enp158s0 -a duration:60 -T fields -E header=y -E separator=, -e frame.number -e eth.src -e eth.dst -e eth.type -e frame.len > ethernet_traffic.csv
```

Collect only special fields

Csv file separated by ","
With first line as header

Fields written in csv

Csv file name to store

#capture live traffic in csv

```
sudo tshark -i enp158s0 -a duration:60 -T fields -E header=y -E separator=, -e frame.number -e eth.src -e eth.dst -e eth.type -e frame.len > ethernet_traffic.csv
```

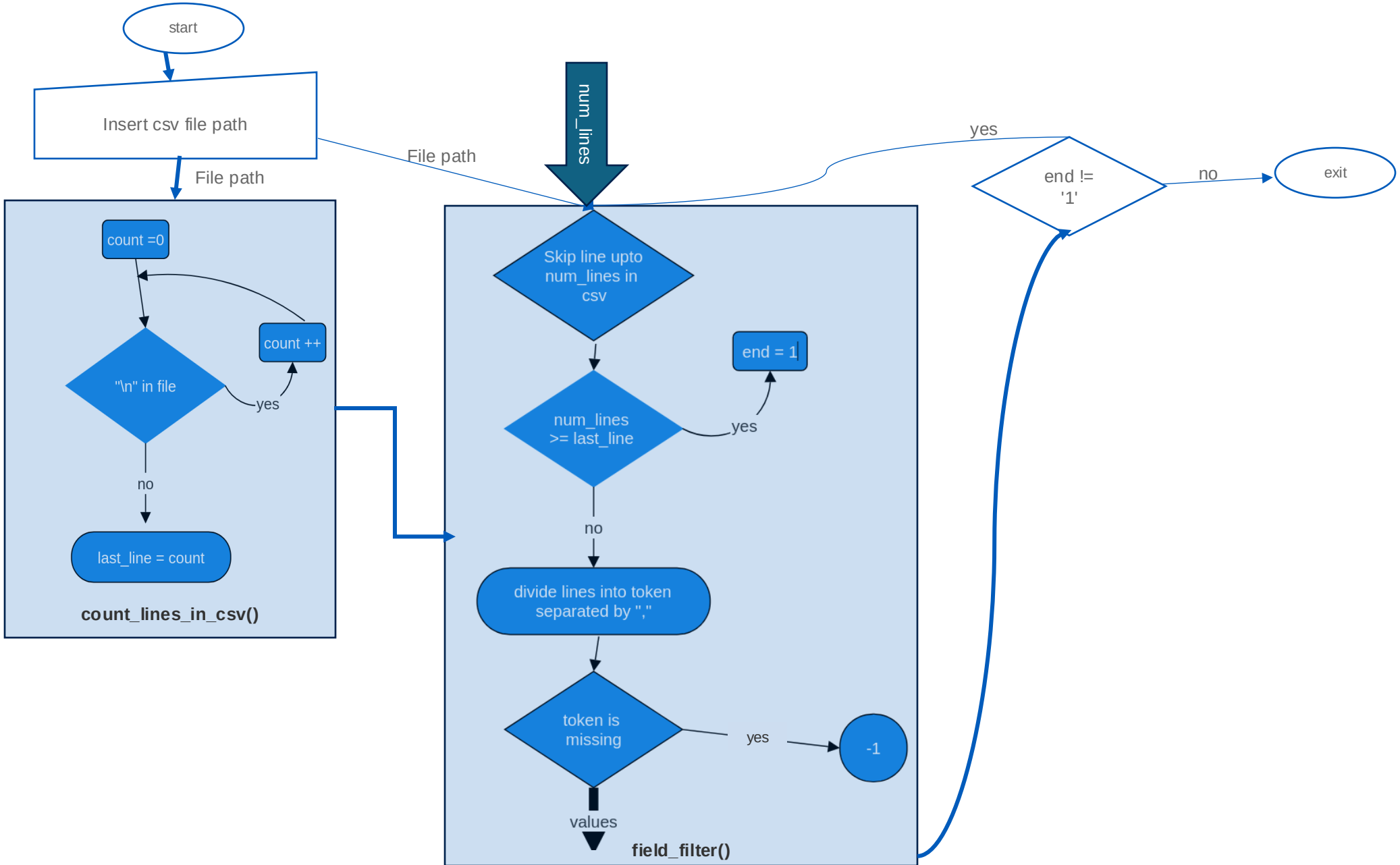
frame.number	eth.src	eth.dst	eth.type	frame.len
1	50:9a:4c:2d:c9:0b	01:00:5e:7f:ff:fa	0x0800	217
2	00:6c:bc:1a:26:04	01:00:0c:cc:cc:cd		64
3	00:1b:21:39:34:81	00:6c:bc:1a:26:48	0x0800	111

```
#capture live tcp traffic in csv
sudo tshark -i enp158s0 -f "tcp" -T fields -E header=y -E separator=, -e tcp.srcport -e tcp.dstport -e
tcp.seq -e tcp.ack -e tcp.flags -e tcp.len -e tcp.window_size > tcp_traffic.csv
```

Only tcp packets captured

URG ,ACK ,PSH ,RST ,SYN ,FIN ,ECE ,CWR ,NS
0xXXYY

Flow chart


start
Insert csv file path
File path
File path
num_lines
count =0
count ++
"\n" in file
yes
no
last_line = count
count_lines_in_csv()
Skip line upto num_lines in csv
end = 1
num_lines >= last_line
yes
no
divide lines into token separated by ","
token is missing
yes
-1
values
field_filter()
yes
end != '1'
no
exit

#Run c program

C programming

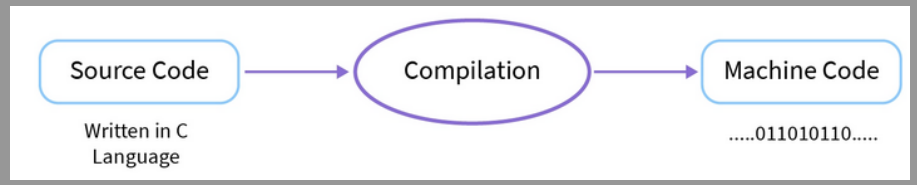
compiler

```
gcc -o run example.c file_handling.c
```

Executable file name

.c compiling files

```
./run
```



For getting field reference - https://www.wireshark.org/docs/dfref/t/tcp.html

```c
// Function to extract field values
void field_filter(const char *filePath, unsigned *num_lines, unsigned last_line, char *end, int *srcport, int
*dstport, int *seq, int *ack, int *flags, int *len, int *window_size);
```

file_handling.h

```c
void field_filter(const char *filePath, unsigned *num_lines, unsigned last_line, char *end, int *srcport,
int *dstport, int *seq, int *ack, int *flags, int *len, int *window_size) {
    FILE *file;
    char line[MAX_LINE_LENGTH];

    file = fopen(filePath, "r");
    if (file == NULL) {
        perror("Error opening file");
        exit(EXIT_FAILURE);
    }

    // Skip lines until reaching the line number stored in *num_lines
    for (unsigned i = 0; i < (*num_lines); i++) {
        if (fgets(line, MAX_LINE_LENGTH, file) == NULL) {
            fclose(file);
        }
    }


    // File field extraction
    if ((*num_lines) >= last_line) {
        fclose(file);
        *end = '1';
        return;
    }
        else {
        // Initialize variables as default missing value
        *srcport = -1;
        *dstport = -1;
        *seq = -1;
        *ack = -1;
        *flags = -1;
        *len = -1;
        *window_size = -1;

        // Extract fields from the current line
        if (fgets(line, MAX_LINE_LENGTH, file) != NULL) {
            char *delims = ",";
            char *token;

            // Use strtok_single to tokenize the line
            token = strtok_single(line, delims);
            if (token) {
                if (sscanf(token, "%d", srcport) != 1) {
                    *srcport = -1;
                }
            }
```

file_handling.c

example.c

```c
int srcport, dstport, seq, ack, flags, len, window_size;

int main(){

        char filePath[256];

//  USER INPUT
                printf("Enter csv file location\n");
                scanf("%s",filePath);

// Calculating last line of csv file
        unsigned last_line = count_lines_in_csv(filePath);

        char end = '0';
        for(unsigned int num_lines = 1; end!='1'; num_lines++){
            field_filter(filePath, &num_lines, last_line, &end, &srcport, &dstport, &seq, &ack, &flags,
&window_size);
            printf("Source Port: %d\t", srcport);
```

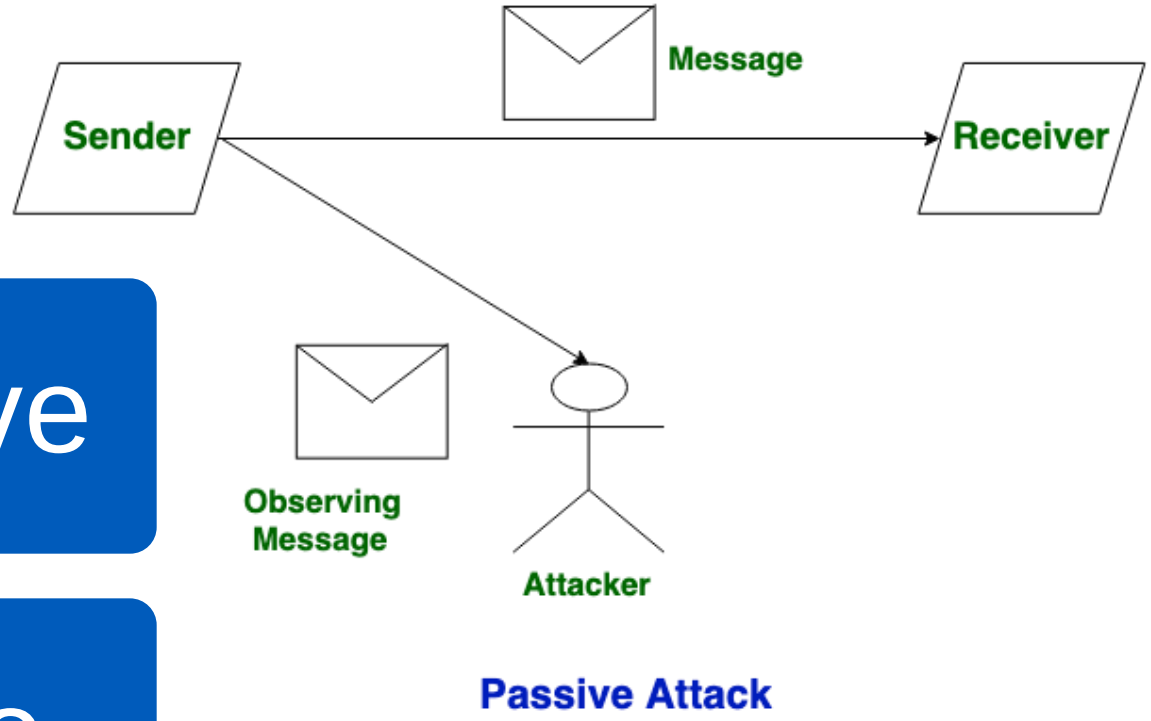
sudo tshark -i enp158s0 -f "tcp" -T fields -E header=y -E separator=, -e ~~**tcp.srcport**~~ **tcp.checksum_bad** -e tcp.dstport -e tcp.seq -e tcp.ack -e tcp.flags -e tcp.len -e tcp.window_size > tcp_traffic2.csv

NETWORK SECURITY

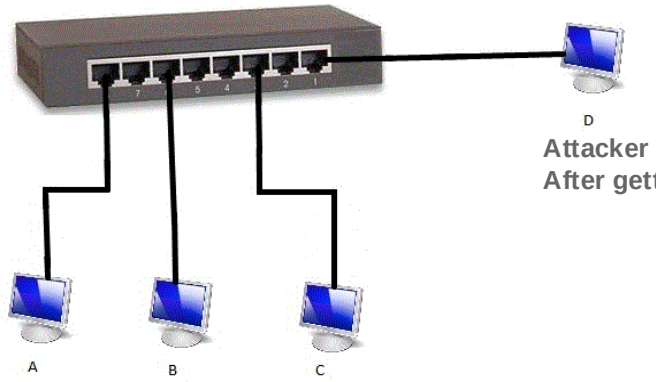
Department of Electronic Systems Engineering

Cyber attack

- Passive
- Active



Passive attack

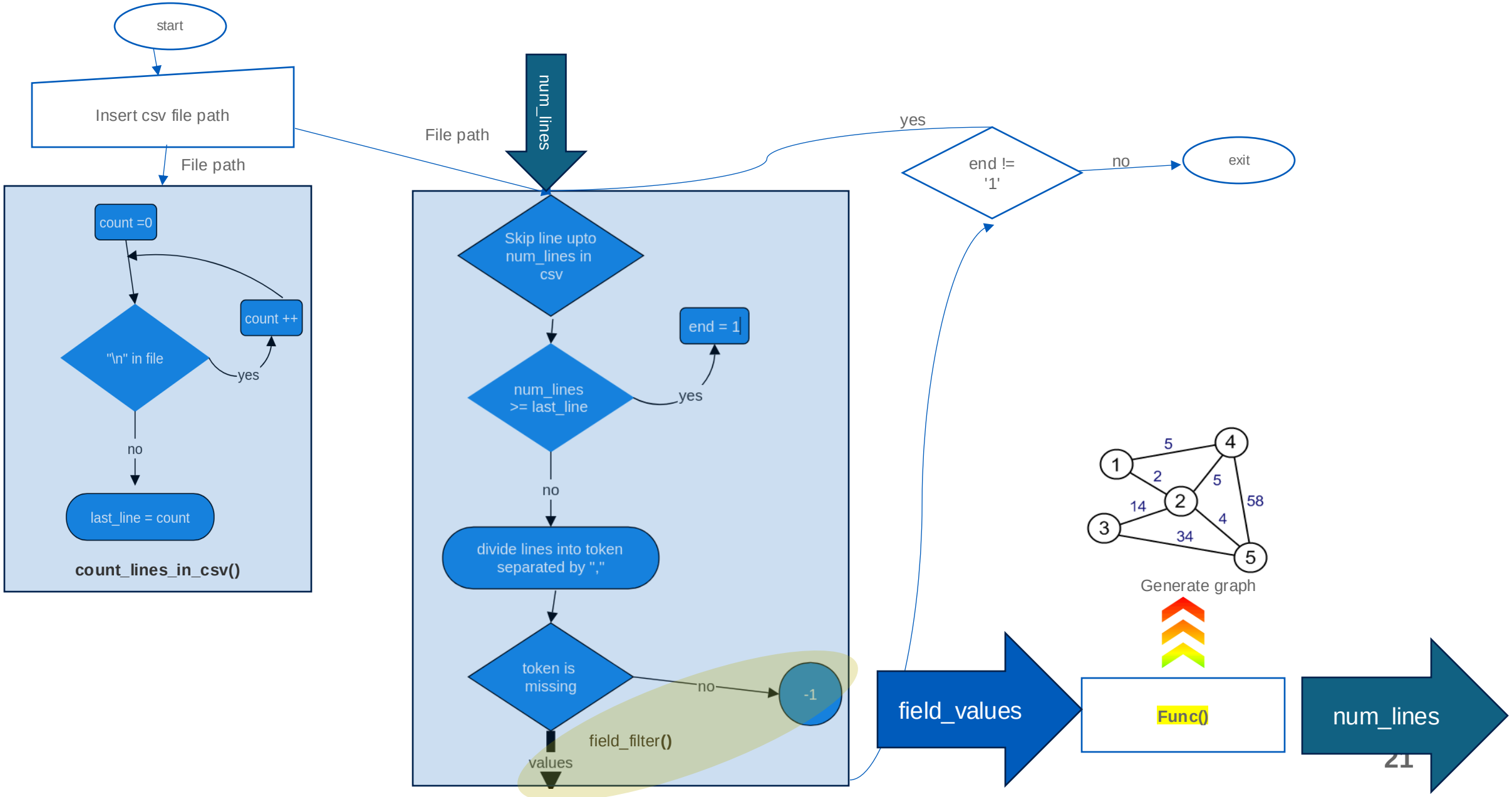


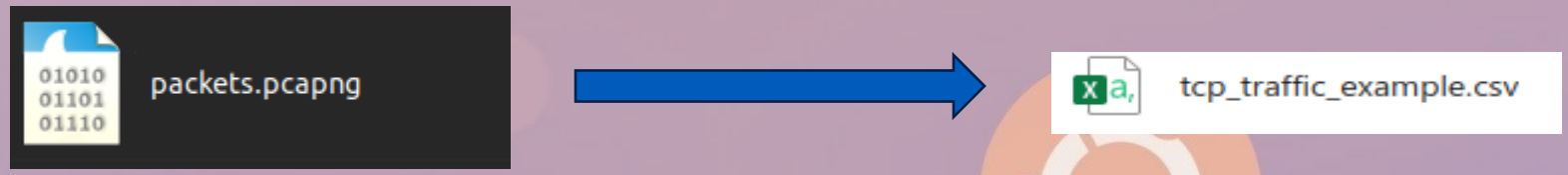
Attacker is somehow able to listen host traffic.
After getting user traffic attacker tries to get information out of traffic.

network_graph(int t) (Slight changes are allowed)

- Generate weighted graphs with Node representing MAC address
  - 1 node is connected to other node by different edges each representing {TCP port, IP address, average message in t time, total repeated message, total TCP retransmissions, total TCP duplicate ack}.
- **tshark -r input.pcap -T fields -E header=y -E separator=, -e frame.number -e ip.src -e ip.dst -e tcp.srcport -e tcp.dstport -e tcp.seq -e tcp.ack** ……other fields **> packet_data.csv**
- Create function to traverse edges and find information from graph.
- First test code on try.pcap
- 4SICS-GeekLounge-151022.pcap should used as dataset (there are practical network errors also)

Flow chart


start
Insert csv file path
File path
count =0
count ++
"\n" in file
yes
no
last_line = count
count_lines_in_csv()
File path
num_lines
Skip line upto num_lines in csv
end = 1
num_lines >= last_line
yes
no
divide lines into token separated by ","
token is missing
no
-1
field_filter()
values
yes
end != '1'
no
exit
field_values
Func()
num_lines
1
2
3
4
5
5
2
14
5
58
4
34
Generate graph

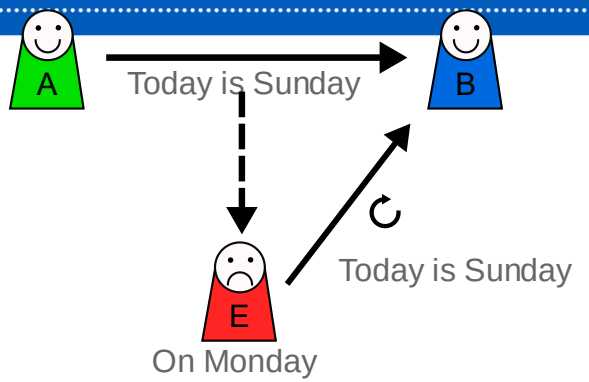


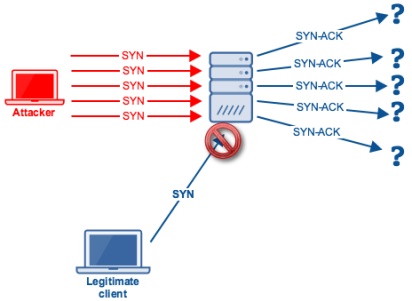
#reading pcap/pcapng file and writing as csv by running command in **folder** having pcap/pcapng

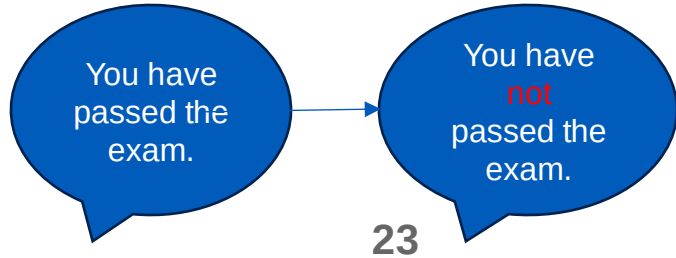
tshark -r packets.pcapng -Y "tcp" -T fields -E header=y -E separator=, -e tcp.srcport -e tcp.dstport -e tcp.seq -e tcp.ack -e tcp.flags -e tcp.len -e tcp.window_size > tcp_traffic_example.csv

Active

- Replay
- DoS
- Spoofing
- Message injection







#creating new tcp connection

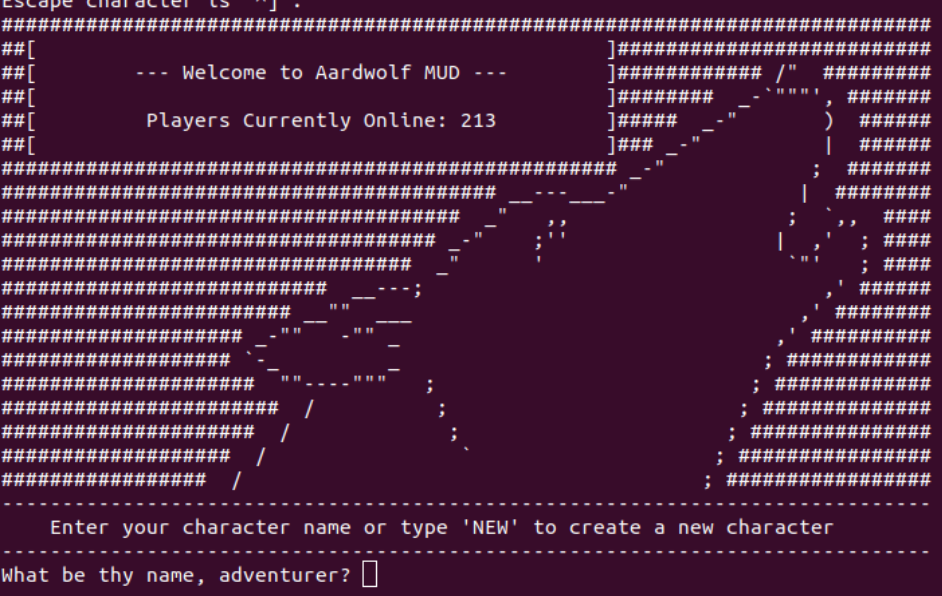
Tool to connect devices over internet or network

telnet aardmud.org 4000

Port number

komfuter@komfuter-Precision-7820-Tower: ~

```
komfuter@komfuter-Precision-7820-Tower:~$ telnet aardmud.org 4000
Trying 23.111.142.226...
Connected to aardmud.org.
Escape character is '^]'.
```




```
    Enter your character name or type 'NEW' to create a new character
What be thy name, adventurer?
```

komfuter@komfuter-Precision-7820-Tower: ~/Downloads/ilo...

```
komfuter@komfuter-Precision-7820-Tower:~/Downloads/ilovepdf_pages-to-jpg(1)$ sud
o python3 assign.py
.
Sent 1 packets.
komfuter@komfuter-Precision-7820-Tower:~/Downloads/ilovepdf_pages-to-jpg(1)$
```

*any

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

(ip.src == 23.111.142.226)

No.	Time	Source	Destination	Protocol	Length	Info
139	73.861590334	23.111.142.226	10.114.56.157	TCP	68	4000 → 50534 [ACK] Seq=1 Ack=2
140	73.972118644	23.111.142.226	10.114.56.157	TCP	68	4000 → 50534 [FIN, ACK] Seq=1 A
232	120.513393010	23.111.142.226	10.114.56.157	TCP	76	4000 → 48196 [SYN, ACK] Seq=0 A
234	120.854748408	23.111.142.226	10.114.56.157	TCP	1931	4000 → 48196 [PSH, ACK] Seq=1 A
237	121.138879386	23.111.142.226	10.114.56.157	TCP	68	4000 → 48196 [ACK] Seq=1864 Ack
247	129.650235351	23.111.142.226	10.114.56.157	TCP	56	20 → 4000 [SYN] Seq=0 Win=8192
248	129.650321216	23.111.142.226	10.114.56.157	TCP	62	[TCP Out-Of-Order] [TCP Port nu

```
    [Stream index: 15]
    [Conversation completeness: Incomplete (20)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 135377778
    [Next Sequence Number: 1    (relative sequence number)]
```

This shows the raw value of the sequence number (tcp.seq_raw), 4 bytes | Packets: 276 · Displayed: 7 (2.5%) | Profile: Default

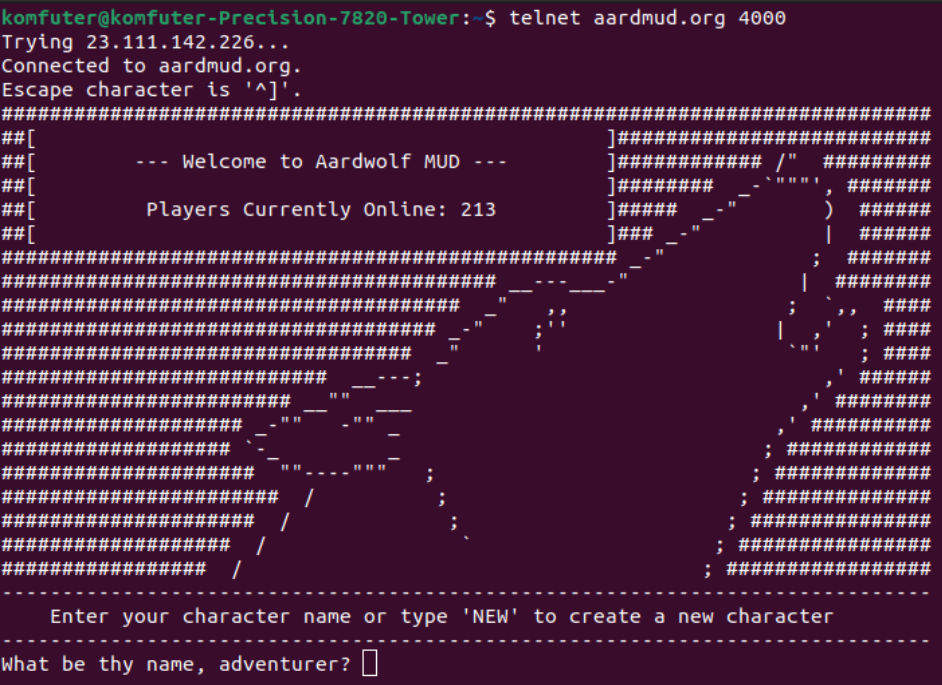
```
from scapy.all import *

# Use the Ethernet layer explicitly
ether = Ether()

# Create IP and TCP packets
ip = IP(src="23.111.142.226", dst="10.114.56.157")
tcp = TCP(dport=4000, flags='S', seq=275)

# Combine Ethernet, IP, and TCP layers
packet = ether/ip/tcp

# Send the packet on a specific interface, like eth0
sendp(packet, iface="enp158s0")
```

Website IP

Your IP

komfuter@komfuter-Precision-7820-Tower: ~

```
komfuter@komfuter-Precision-7820-Tower:~$ telnet aardmud.org 4000
Trying 23.111.142.226...
Connected to aardmud.org.
Escape character is '^]'.
```




```
    Enter your character name or type 'NEW' to create a new character
What be thy name, adventurer?
```

komfuter@komfuter-Precision-7820-Tower: ~/Downloads/ilo...

```
komfuter@komfuter-Precision-7820-Tower:~/Downloads/ilovepdf_pages-to-jpg(1)$ sud
o python3 assign.py
.
Sent 1 packets.
komfuter@komfuter-Precision-7820-Tower:~/Downloads/ilovepdf_pages-to-jpg(1)$
```

*any

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

(ip.src == 23.111.142.226)

No.	Time	Source	Destination	Protocol	Length	Info
139	73.861590334	23.111.142.226	10.114.56.157	TCP	68	4000 → 50534 [ACK] Seq=1 Ack=2
140	73.972118644	23.111.142.226	10.114.56.157	TCP	68	4000 → 50534 [FIN, ACK] Seq=1 A
232	120.513393010	23.111.142.226	10.114.56.157	TCP	76	4000 → 48196 [SYN, ACK] Seq=0 A
234	120.854748408	23.111.142.226	10.114.56.157	TCP	1931	4000 → 48196 [PSH, ACK] Seq=1 A
237	121.138879386	23.111.142.226	10.114.56.157	TCP	68	4000 → 48196 [ACK] Seq=1864 Ack
247	129.650235351	23.111.142.226	10.114.56.157	TCP	56	20 → 4000 [SYN] Seq=0 Win=8192
248	129.650321216	23.111.142.226	10.114.56.157	TCP	62	[TCP Out-Of-Order] [TCP Port nu

Out of order sequence number message produced by scapy

Suppose you are able to predict next seq. no. You can manipulate host .

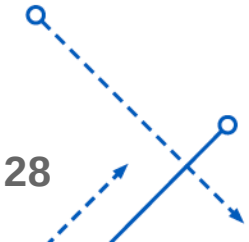
```
    [Stream index: 15]
    [Conversation completeness: Incomplete (20)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 135377778
    [Next Sequence Number: 1    (relative sequence number)]
```

This shows the raw value of the sequence number (tcp.seq_raw), 4 bytes | Packets: 276 · Displayed: 7 (2.5%) | Profile: Default

Active attack

tcp_attack(int attack_type,int msg_index,int msg_type)

- Attack on given message index destination address.
- Dependent on tcp_attack value
  - attack_type == 0 than **terminate tcp connection by sending FIN** flag ON with next sequence number
  - attack_type == 1 TCP **SYN flood attack**
  - attack_type == 2 **replay old random message** with next sequence number
  - etc. (use flags **SYN ,ACK ,PSH ,RST ,RST ,FIN ,URG**)
  - Should try all possible attacks
- Message (without header part) selection
  - If(msg_type == 1 && attack_type != 2)
    - Send same message (excluding header)
  - elseif(msg_type == 2)
    - Random previous message from same destination and same source
  - elseif(msg_type == 3)
    - Random previous message from random destination and same source
  - else
    - Random message



Passive attack

prediction_array(const char * MAC_address, unsigned int t, unsigned * prediction_array)

- Generate output predicted_array for all socket; all possible flags (like tcp flags = {SYN ,ACK ,PSH ,RST ,RST ,FIN ,URG ,etc}) and all important numbers (like {Presentation Layer Sequence Number (Session ID), Application Layer (InvokeID), Transport Layer Sequence Numbers}) and message length
- Should produce time and frequency based prediction function output for same connection after t time duration
- Write as clear assumptions
- Optional – use of ML.
- First test code on try.pcap
- 4SICS-GeekLounge-151022.pcap should used as dataset

Active attack -> Replay attack

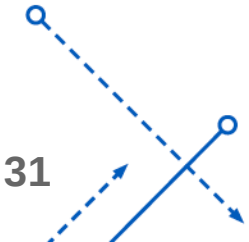
prediction_msg(char * dst, int dst_type, same_dst, unsigned predicted_array)

- If same_dst is 1 than all below search should be of same_dst else any destination
- Get **best message (without any header)** of near to predicted flags and predicted length of message.
- dst_type selection
  - If(dst_type == 1)
    - dst is "ip_address"
  - elseif(dst_type == 2)
    - dst is "MAC_address"
  - Elseif(dst_type == 3)
    - dst is "dst_port-dst_ip"
- Generate **complete message upto required layer** for given address (if dst is "ip_address" message header according to ip layer communication)
- If IP fragment MF = 1 , it should send all message fragments

Active attack -> DoS

DoS(ip_address)

- DoS using latest IP message.
- **Low TTL attack -** Set the TTL field to a low value (e.g., 1 or 2) in packets sent to the target. Packets with a low TTL will expire quickly, resulting in a large number of ICMP "Time Exceeded" messages being generated and sent to the sender.
- **Teardrop attack** - If you send a lot of IP packets with the "More Fragments" (MF) flag set but do not send the last fragment (the one without the MF flag set), and you do this from different IP addresses, it can cause host buffer consumption.
- **Large payload attack –** Send frequent large payloads with different TCP/IP/MAC spoofing.



Assignment

- Output - packet
  - tcp_attack(int attack_type,int msg_index,int msg_type)
  - prediction_msg(char * dst, int dst_type, same_dst, unsigned predicted_array)
  - DoS(ip_address)
- Output - array/graph
  - network_graph(int t)
  - prediction_array(char * MAC_address, unsigned int t, unsigned prediction_array)

Packets

- Normal/Industry packets
  - try
  - 4SICS-GeekLounge-151022
  - Packets
  - packet2wireshark
- Attacked packets
  - handshake_lost_hijack_netcat_loopback1
  - sloppy_spray_injection1
  - ordered_coalesce_netcat1
  - ordered_coalesce_netcat2
  - adhose-trickle-riseupvpn
- Ethical hacking competition data
  - hackeire-master (See in free time)

ALLOWED

- Python/C
- Use of research paper (**with mention**)
- Use of online available codes (github repository code (**with mention**))
- Use of chat GPT or copilot
- New packet dataset (which gives better testing capabilities) sharing
- Allowed messaging on **own** MAC/IP/port with spoofing different MAC/IP
- You're encouraged to read the pcapng file or real traffic, convert it to CSV format for easier analysis, and then write function code based on the created CSV.
- **1 Group of 5 will do all 5 functions**

NOT ALLOWED

- Do not run provided pcapng/pcap file in IISc network
- You're encouraged to make a fresh attempt at the assignment with
  available online resources.
- No group code (Copy of code or variable change will graded 0)
- It should be independent of dataset and checked on new dataset of similar network.
- Write all code as either two functions or an OOPs class: one for reading pcapng/tshark data (provided in c) and another for implementing the logic.