George, Levi

CS44500 – Computer Security

04/07/2023

Lab 9 – TCP Attacks

Env Setup.

I started the containers

Text

Description automatically generated

TASK 1: SYN FLOOD



I ran the command to check my syn backlog

I ran netstat -nat to see how many connections are open on my VM

Text

Description automatically generated

I opened a shell into my victim container

A screenshot of a computer

Description automatically generated with medium confidence

And checked the syncookies value on it, syncookies is turned off.

I wrote the code for the syn flood, with the chosen IP for the victim container and the port for telnet added.

Text

Description automatically generated

I ran the program

Text

Description automatically generated with medium confidence

I let it run for around a minute then tried telnet while it was running

Text

Description automatically generated

Text

Description automatically generated

The attack had failed.

I am using containers, so the VirtualBox issue is not relevant.

Checking the victim, it appears that one instance of synflood fulfills ¾ of the open communications. The other ¼ must be proven destinations.

I ran two instances of synflood and didn’t notice an appreciable change. However, once I ran three instances I was reliably kept from connecting. Some students may encounter fewer issues compared to me with getting one or even two instances to work, but since I am using a relatively low powered 3rd party VM (Digital Ocean) with only 2GB of ram and using python as my scripting language then I may be having a harder time.

Text

Description automatically generated

I began testing how one instance of my synflood could force an instance of telnet to reliably timeout with different backlog values.

Text

Description automatically generated

I set the syn backlog to ¾ it’s original value of 128, and tested.

However, even at ¾ we can still connect

Text

Description automatically generated

I set the backlog to half its original value

Text

Description automatically generated

Running one synflood succeeds at butting out telnet connections when our backlog is at 64

Text

Description automatically generated

Task 1.2 – Synflood in C

I set the backlog back to normal



And flushed tcp metrics



I compile synflood



I start synflood against my target



With only one instance of synflood running, I was blocked from making TCP connections

Text

Description automatically generated

I needed either a significantly reduced backlog or several instances of synflood running with the python version.

The reason for this is the fact that python is interpreted, so it is much slower, we would get similar results if we wrote this synflood in JavaScript. However, C is fast, faster than python since it is not interpreted. So, it is able to send out a spoofed packet much faster than python.

TASK 1.3

I enabled the syncookies

Text

Description automatically generated

Shortly after I attempted to telnet into the container and was able to get through fairly quickly.

Text

Description automatically generated

TASK 2: TCP RST Attack on Telnet

I opened a shell on 10.9.0.6 (our user1 container)

Text

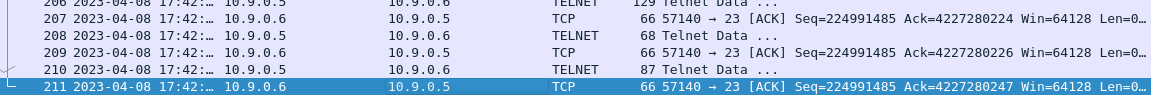
Description automatically generated

I telnet into the victim container

Text

Description automatically generated

These are some of my wireshark captures



Using the capture I modified the code provided in the lab handout.

Text

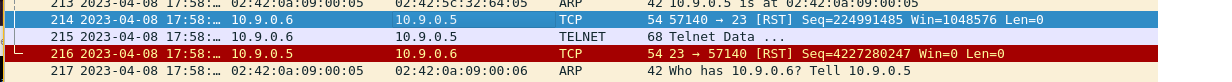
Description automatically generated

I launched the attack

Text

Description automatically generated

I got this back in Wireshark, I checked on my telnet connection and got this



Graphical user interface, text

Description automatically generated

TASK 3: TCP HIJACK

I reconnected to the victim system from the same container.

Text

Description automatically generated

I got some of the telnet info from wireshark



I wrote the python code for the hijacking (I also ran touch secret.txt on the victim container so that I would have something to grab)

Text

Description automatically generated

I ran nc -l 8080 on a separate ssh connection to my vm

Text

Description automatically generated

I ran the TCP hijack code and was able to get the secret text file from the victim system. (As seen above)

A screenshot of a computer

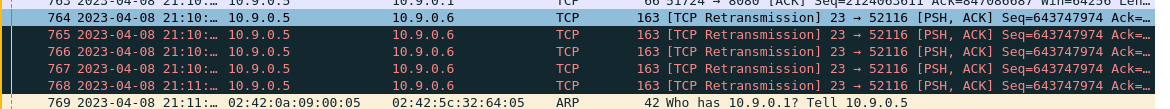
Description automatically generated with medium confidence

Here is the wireshark printout of my packet

A picture containing text

Description automatically generated

And we see that there was additional effects, as described in the book.



Since the main telnet connection was pre-empted by the spoofed packet the entire telnet communication was offset and now the telnet session I was running from 10.9.0.6 is frozen.

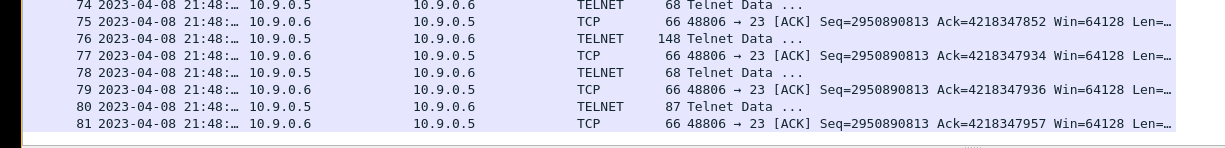
TASK 4: TCP HIJACK w/ REVERSE SHELL

I prepare my netcat

Text

Description automatically generated

I grabbed the telnet information from my 10.9.0.6 -> 10.9.0.5 connection.



I rewrote my hijack code to use a reverse shell command.

Text

Description automatically generated

Then I initiate my attack, I ran “sudo chmod a+x tcp-hijk.py”

Then I ran the command “sudo ./tcp-hijk.py” from one shell on my VM

With netcat open on another VM shell.

I was able to get into the victim server

Text

Description automatically generated

Text

Description automatically generated