George, Levi

CS44500 – Computer Security

03/30/2023

Lab 8 – Sniffing and Spoofing

Env Setup.

I ran dcbuild and dcup to create the docker container’s but due to me being lazy the previous containers were still live.

Text

Description automatically generated

I ran through some of the networking and removal process of old containers in docker.

Text

Description automatically generated

After going through the removal of the old dockers, I built the new containers

Text

Description automatically generated

TASK 1 – Using Scapy to Sniff and Spoof

TASK 1.A

I wrote a basic scapy program

Graphical user interface, text

Description automatically generated

It built a basic IP packet,

Text

Description automatically generated

Nothing is filled out very interestingly, we even have a basic src and dst field on the loopback address.

In order to sniff and spoof packets from the containers I grab the network ids using ifconfig

Text

Description automatically generated

Here is the code for my packet sniffer

A screenshot of a computer

Description automatically generated with medium confidence

I pinged one of the host containers in the lab

Graphical user interface, text

Description automatically generated

When I checked my putty running the sniffer code I got this info back

Text

Description automatically generated

I attempted to run my packet sniffing code as seed, without root permissions.

Graphical user interface, text

Description automatically generated

I was not allowed as just seed.

TASK 1.B

I rewrote my code to include the filter

Text

Description automatically generated

I ran the code and on an adjacent putty I ran a ping command

Text

Description automatically generated

I got the following output from the first ping command.

Text

Description automatically generated

To show that I was only getting ICMP packets, I used Scapy to build an IP packet with a specific TTL (27)

Text

Description automatically generated

However the last packet I received was an ICMP packet

Text

Description automatically generated

TASK 1.B PART 2 - Capture any TCP packet ….

I rewrote the sniffer to use a new filter

A screenshot of a computer

Description automatically generated with medium confidence

I sent the tcp packet using netcat



This is what I received on my primary host

Text

Description automatically generated

TASK 1.B PART 3

I rewrote my sniffer code

Text

Description automatically generated

I pinged the shown IP through my VM by using one of the containers.

Graphical user interface, text

Description automatically generated with medium confidence

I got this from my sniffer, note: since I was on a cloud VM there were several adapters that I could ping through to reach different subnets.

Text

Description automatically generated

TASK 2 – SPOOFING ICMP

I opened VNC to use wireshark, then from one of my container’s I opened Scapy and ran the following commands

Text

Description automatically generated

Back in my wireshark instance, we see that my spoofed packet was successfully grabbed.

Graphical user interface, text

Description automatically generated

TASK 3 – TRACEROUTE

I wrote a basic script to carefully send and then receive each ICMP packet while incrementing their ttl until we got a hit. I used some of the class demo code as a guide.

Text

Description automatically generated

I ran the code on one of the containers

Text

Description automatically generated

Text

Description automatically generated

This was my wireshark output

A picture containing table

Description automatically generated

We can actually see from wireshark that ttl 10 is when we got the first reply, the same can be corroborated by the output from my trcrt program.

TASK 4 – Sniff and Spoof for ICMP

The code I used for this portion of the lab

Text

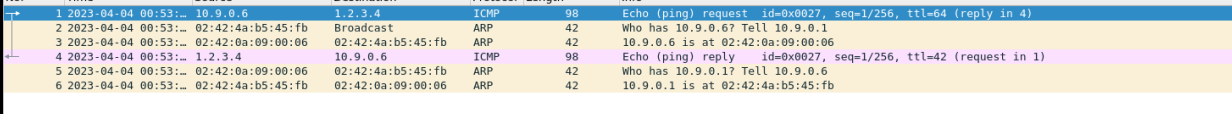
Description automatically generated

For pinging 1.2.3.4, I was able to send 1 packet, catch the packet and send it back.

Text

Description automatically generated

Wireshark capture (we know this is my packet since it has a ttl of 42.)



The original packet sent in the ping

Text

Description automatically generated

The spoofed packet

Text

Description automatically generated

FOR PINGING 10.9.0.99

I pinged 10.9.0.99

Text

Description automatically generated

Wireshark captures

Table

Description automatically generated

I captured no packets



The reason why this didn’t work for spoofing the packet is because we are on the same subnet, so instead of IP routing being the primary means of routing, it uses MAC addresses (and ARP) but we didn’t spoof the MAC Address, so when we try to find it, we fail to do so through ARP.

Text

Description automatically generated

PINGING 8.8.4.4

I pinged, 8.8.4.4



However, while pinging I noticed something interesting. I had gotten extra packets that were marked as duplicates. All of the packets that I had sent (TTL=42) were marked as duplicates.

Text

Description automatically generated

An example spoofed packet

Text

Description automatically generated

Wireshark entries

Table

Description automatically generated

I haven’t included the packets I received and sent back, since they weren’t successfully spoofed. However, I think a huge part of the cause is the processing time. We are using Python, a relatively easy and quick to use programming language that is interpreted. However, Google’s server is probably built on top of something that is faster (which is almost anything) like Java or C. If I wrote this in C, I could perhaps send these packets out much faster. This could also be a result of the resources I used to build my VM, due to the cost I tried to use the lowest amount of resources as possible. Meaning, my processor could only recreate the packets at a slower rate.