CSCI 401

Lab -11

Prof. Kadri Brogi

May 01, 2020

Emranul Hakim

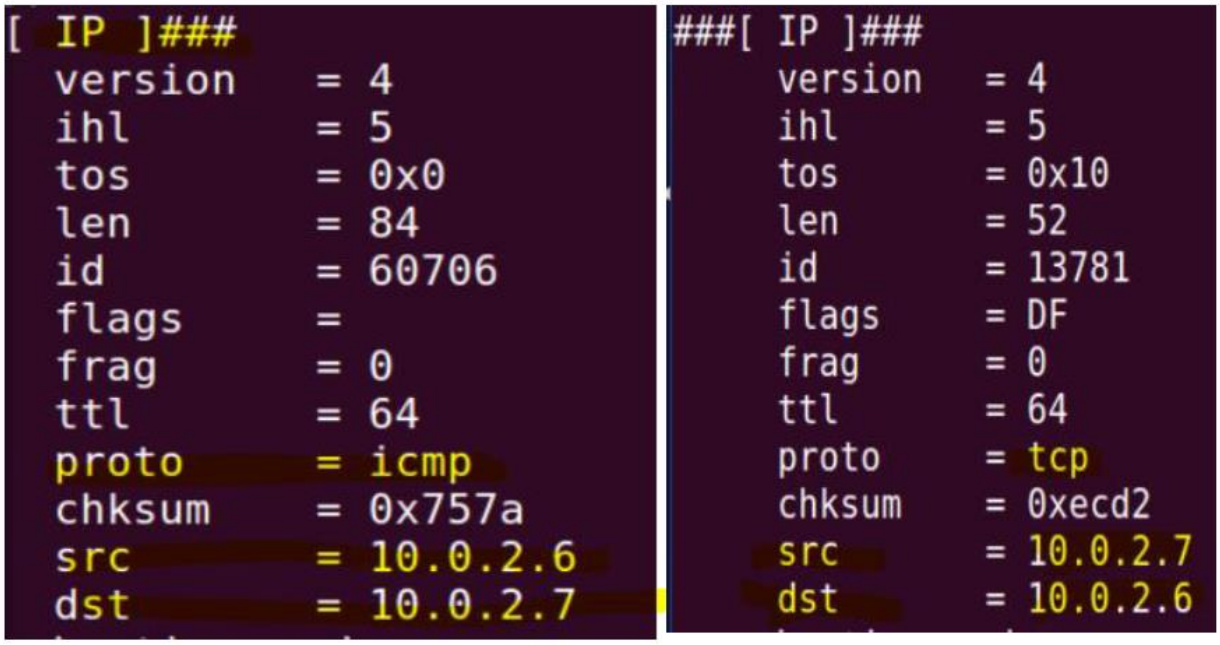
23467834

Packet Sniffing and Spoofing Lab

Packet sniffing in other word is listening to a conversation like capturing username or password or network traffics, and spoofing is actively pretending to be someone else like sending fake packets over the network/fake email/fake IP address. In this lab, we will learn to use how packet sniffing and spoofing are implemented in software.

Using Tools to Sniff and Spoof Packets

Server: 10.0.2.7, Attacker: 10.0.2.6



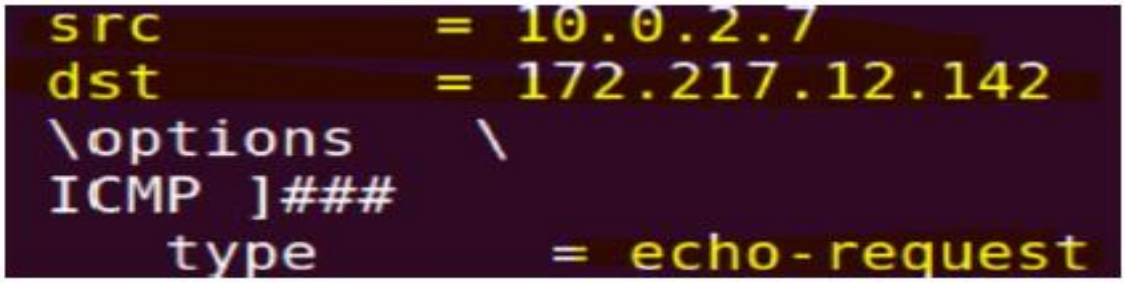
I used Scaly as a building block to construct other tools as well as to do packet sniffing in python programs. I have demonstrated the Ping Reply for ICMP packets as well as for TCP packets and packets from subnet **10.0.2.0/24**. So, I used Wireshark to capture the ping request sent by server and my machine replied back. This demonstrated that we can spoof an ICMP echo request packet with an arbitrary source IP address. **10.0.2.3**

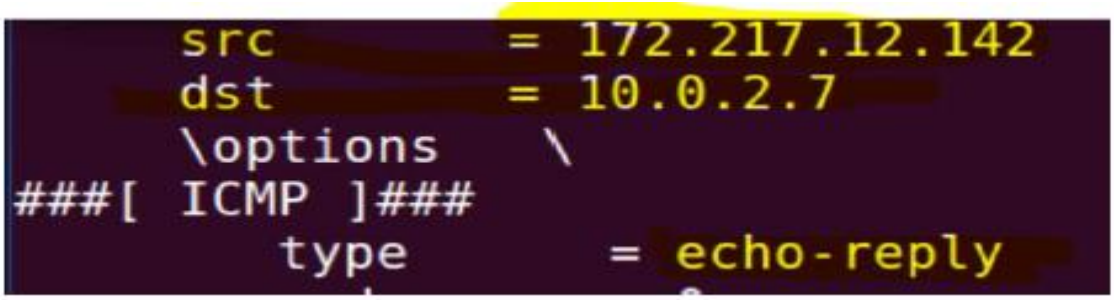
Source	Destination
10.0.2.7	10.0.2.6
10.0.2.6	10.0.2.7
10.0.2.7	10.0.2.6

Source	Destination
10.0.2.6	10.0.2.3
10.0.2.3	10.0.2.6

10.0.2.6	172.217.12.142
172.217.12.142	10.0.2.6

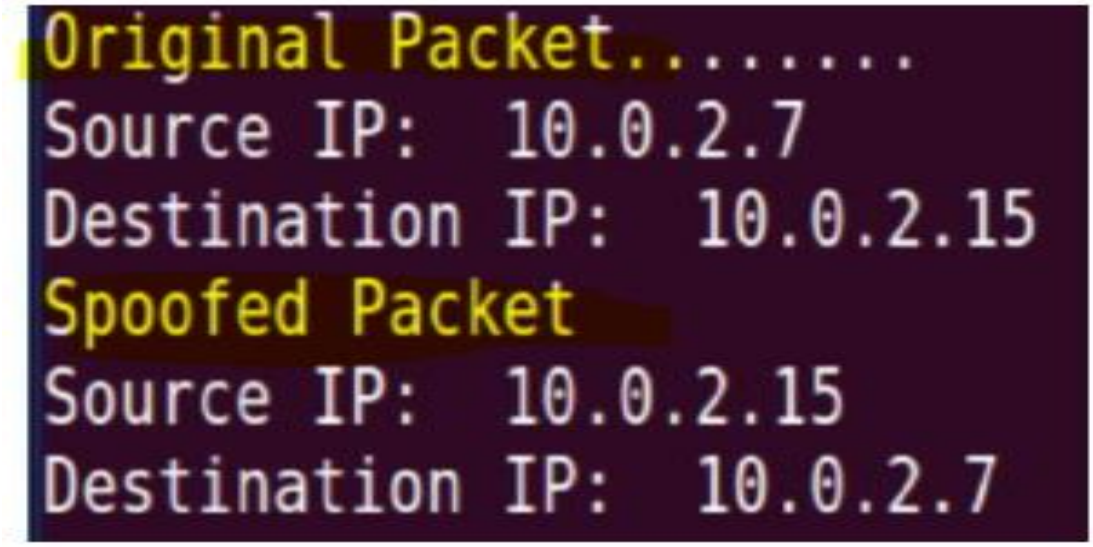
I did traceroute to estimate the distance, in term of no. of routers, between my VM and Google's destination, 172.217.12.142 with TTL = 30 hops.





This is another way of showing the packets are exchanging.

Sniffing and-then Spoofing:




```
#!/usr/bin/python3

from scapy.all import *

def spoof_pkt(pkt) :
        if ICMP in pkt and pkt[ICMP].type == 8:
                print ("Original Packet.....")
                print ("Source IP: ", pkt[IP].src)
                print ("Destination IP: ", pkt[IP].dst)

                ip = IP(src=pkt[IP].dst, dst=pkt[IP].src, ihl=pkt[IP].ihl)
                icmp = ICMP(type=0, id=pkt[ICMP].id, seq=pkt[ICMP].seq)
                data = pkt[Raw].load
                newpkt = ip/icmp/data

                print ("Spoofed Packet")
                print ("Source IP: ", newpkt[IP].src)
                print ("Destination IP: ", newpkt[IP].dst)
                send(newpkt, verbose=0)

pkt = sniff(filter='icmp and src host 10.0.2.7' ,prn=spoof_pkt)
```

I was able to combine sniffing and spoofing techniques. I had the client VM: 10.0.2.15, Attacker 10.0.2.6, Server: 10.0.2.7. So, even when my client machine was off, in my attacker machine I could get the original and spoofed packets.

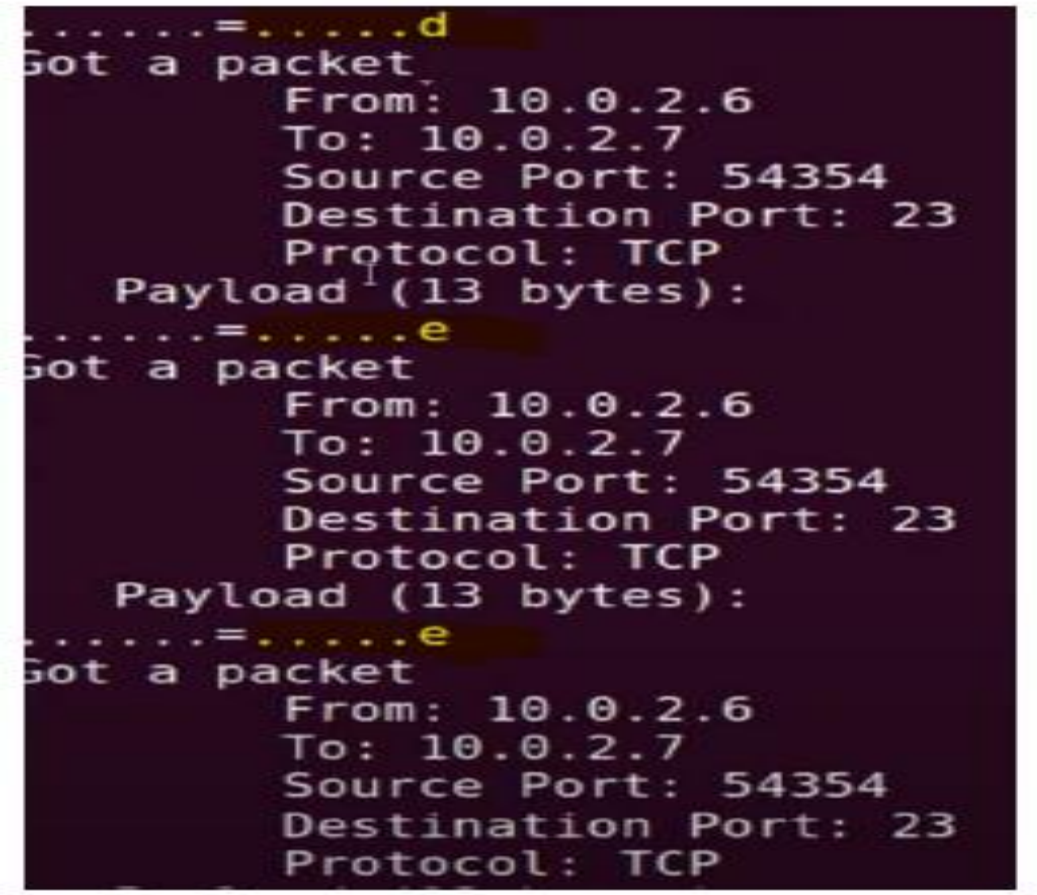
Writing Programs to Sniff and Spoof Packets:

Source	Destination	Protocol
10.0.2.7	10.0.2.6	ICMP
10.0.2.6	10.0.2.7	ICMP

Source	Destination	Protocol
10.0.2.7	10.0.2.6	TELNET
10.0.2.6	10.0.2.7	TCP

```
char filter_exp[] = "proto ICMP and (host 10.0.2.6 and 10.0.2.7)";
char filter_exp[] = "proto TCP and dst port range 10-100";
```

I wrote a sniffer program to print out the source and destination IP addresses of each captured packet along with protocol. We need the root privilege to run a sniffer program because the program fails if it is executed without the root privilege as it is needed for pcap, the functions will be invoked by pcap for each captured packet. pcap_t *handle; If we turn off the promiscuous mode (NIC card within our computer) in our sniffer program: handle = pcap_open_live ("enp0s3", BUFSIZ, 1, 1000, errbuf); we will get segmentation fault.



So, I wrote the program that sniffs the password when someone telnet on the server that I was monitoring. After that, I had to try 2-3 times to get the password, "dees**".**

Spoofing:

Raw sockets give programmers the absolute control over the packet construction, allowing programmers to construct any arbitrary packet, including setting the header fields and the payload.

```
Sending spoofed IP packet...

---------------------------------

   From: 172.217.10.4
   To: 10.0.2.7
```

Source	Destination
172.217.10.4	10.0.2.7
172.217.10.4	10.0.2.7
10.0.2.7	172.217.10.4

In a nutshell, I spoofed an ICMP echo request packet on behalf of some random IP address that's not even in my network. From the Wireshark snapshot, spoofing is successful, I can see the echo reply coming back from that remote machine.