NFS Lab 6: Firewalling and Bypass

I Introduction

This lab session focuses on configuring simple firewall filtering rules to affect communication between different parts of the network, and find ways to bypass them

I.1 Prerequisites

For this lab, we will be using docker to emulate two networks seperated by a firewall. You can download them from here https://avalonswanderer.github.io/assets/zip/nfs/tp6_docker.tar.gz.

Docker compose cheatsheet:

```
~$ docker compose build
~$ docker compose up -d
~$ docker compose exec [hostname] bash
~$ docker compose down
```

Host Credentials:

```
login:  seed
pwd:    dees
```

II Topology and Setup

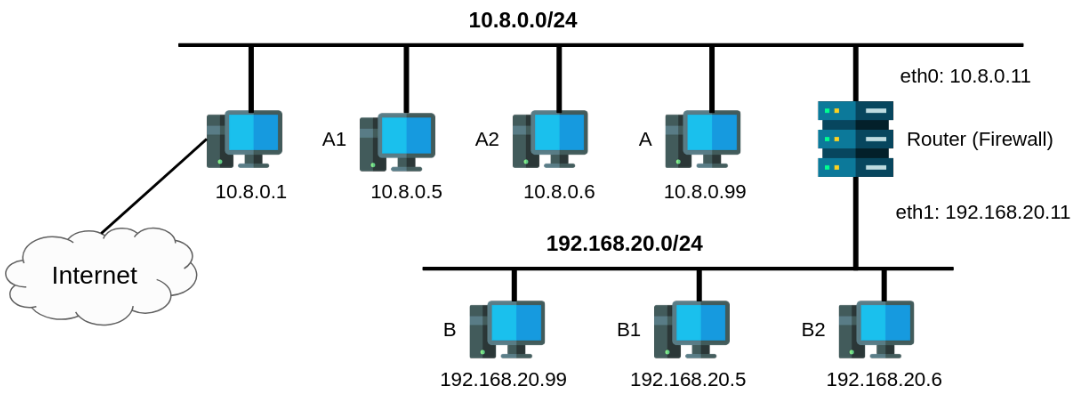

Figure 1: Network Topology.

The router is currently configured to perform a NAT operation from the 192.168.20.0/24 network to 10.8.0.0/24.

II.1 Setting up and bypassing Ingress Filtering

Ingress filtering: refers to the practice of enforcing strict control rules on traffic entering a network. In this section, you will setup an Ingress filtering policy to block any TCP connection that are not SSH.

II.1.1 SSH Only Rule

Question 1: On the firewall, write a set of rules in the forward chain of the filter table to only accept SSH connection coming to eth0 and drop all other tcp packets.

You will need to use the conntrack module of iptables to track TCP connection status.

Checkpoint: Contact your lab supervisor to show your rules and explain what you did.

Note: On your report, write and explain your rules.

II.1.2 Bypassing Ingress

The firewall currently blocks external hosts from connecting to any TCP servers on the internal network except SSH. Now, we want to reach the telnet service on B1 from A1, so we will use static port forwarding to bypass this restriction. Concretely, we will create an SSH tunnel between host A and host B such that any traffic arriving at A on port 9090 is forwarded through the tunnel to B, and B then delivers it to the target B1.

Question 2: Using the ssh command line tool on A, create a static port forwarding connection with B to redirect incoming traffic from the port 9090 to B1's IP on port 23 (telnet).

```
A:~$ ssh -4NT -L ... // Check -L in the man
// -4: IPv4 only
// -N: No remote command (for port forwarding purposes).
// -T: Disable pseudo-terminal allocation.
```

Once setup, you should be able to use, for instance, A1 to telnet to A's IP on port 9090 and will be redirected to a telnet connection on B1.

Checkpoint: Contact your lab supervisor and demonstrate that you can use telnet from hosts A, A1 or A2 to connect to B1.

Question 3: How many TCP connections are involved in this entire process. Run wireshark to capture the network traffic, and then point out all the involved TCP connections from the captured traffic.

Question 4: Explain why can this tunnel successfully help users evade the firewall rules.

II.2 Setting up and bypassing Egress Filtering

Egress filtering refers to the practice of enforcing strict control rules on traffic leaving a network. In this section, you will setup rules to block access to specific websites from the internal network 192.168.20/24.

II.2.1 Blocking Access

Question 5: On the firewall, add an iptables rule to block the univ-rennes.fr website and another website of your choosing.

- Find a way to recover the website IP address.
- Add a forwarding rule to the filter table on eth1 to drop all packet toward this IP.

Note: On your report, write and explain your rules and how you managed to recover the IP addresses.

II.2.2 Bypassing the filtering

In the static port forwarding solution used for ingress bypass, each port-forwarding tunnel forwards the data to a particular destination and port. If we want to forward data to multiple destinations, for instance due to multiple blocked IPs, we need to set up multiple tunnels. To avoid this, we can use dynamic port forwarding to solve the problem.

In the following, you will need to setup a ssh proxy to allow a device in the 192.168.20.0/24 network to access the blocked websites.

```
B:~$ ssh -4NT -D // Check -D in the man
```

Once properly setup, you should be able to contact blocked websites using the open connection as a proxy. To access a specific URL with curl and a proxy use the following command line:

```
B:~$ curl --proxy socks5h://<B’s IP>:<B’s port> <blocked URL>
```

Checkpoint: Contact your lab supervisor and demonstrate that you can visit all the blocked websites using curl from hosts B, B1, or B2 on the internal network.

Question 6: Which computer establishes the actual connection with the intended web server?

Question 7: How does this computer know which server it should connect to?

Bonus On your host, configure your browser to use B as your proxy and try to access the blocked websites when the tunnel is up and down. Docker networks always reserve an IP for your host to be able to contact containers, meaning your are also part of the private network 192.168.20.0/24.

Acknowledgements

This work is inspired by seedlabs and the book *Internet Security: A Hands-on Approach, 3rd Edition,* by Du Wenliang.