Documentation: Evaluating Firewall Effectiveness

---

1. Introduction

Project Overview: The "Building and Configuring Firewall" project involves setting up and testing a firewall using `ufw` (Uncomplicated Firewall) on a virtual Kali Linux machine. The goal is to configure specific firewall rules and verify their effectiveness using various testing tools like `ping` and `nmap`. The virtual Kali Linux machine is hosted on a Parrot OS environment.

Environment:

- **Host Machine OS:** Parrot OS
- **Virtual Machine OS:** Kali Linux
- **Firewall Tool:** UFW (Uncomplicated Firewall)
- **Testing Tools:** `ping`, `nmap`

---

2. Prerequisites

System Requirements:

- **Host Machine:**
  - **CPU:** Dual-core or higher
  - **RAM:** 4 GB or more
  - **Disk Space:** 20 GB available space
- **Virtual Machine:** Kali Linux image, configured with at least 2 GB RAM and 10 GB disk space.

Tools and Software:

- **UFW:** Installed by default on Kali Linux; if not, it can be installed using:

apt install ufw

Nmap: A network scanning tool, installed via:

apt install nmap

Network Configuration:

- **Host Machine IP:** E.g., `192.168.100.191`
- **VM IP:** Dynamic or Static IP within the same subnet as the host machine.

3. Firewall Setup with UFW

Installing UFW: If `ufw` is not installed on your Kali Linux VM, you can install it using

sudo apt install ufw

Enabling UFW: To enable ufw, run the following command:

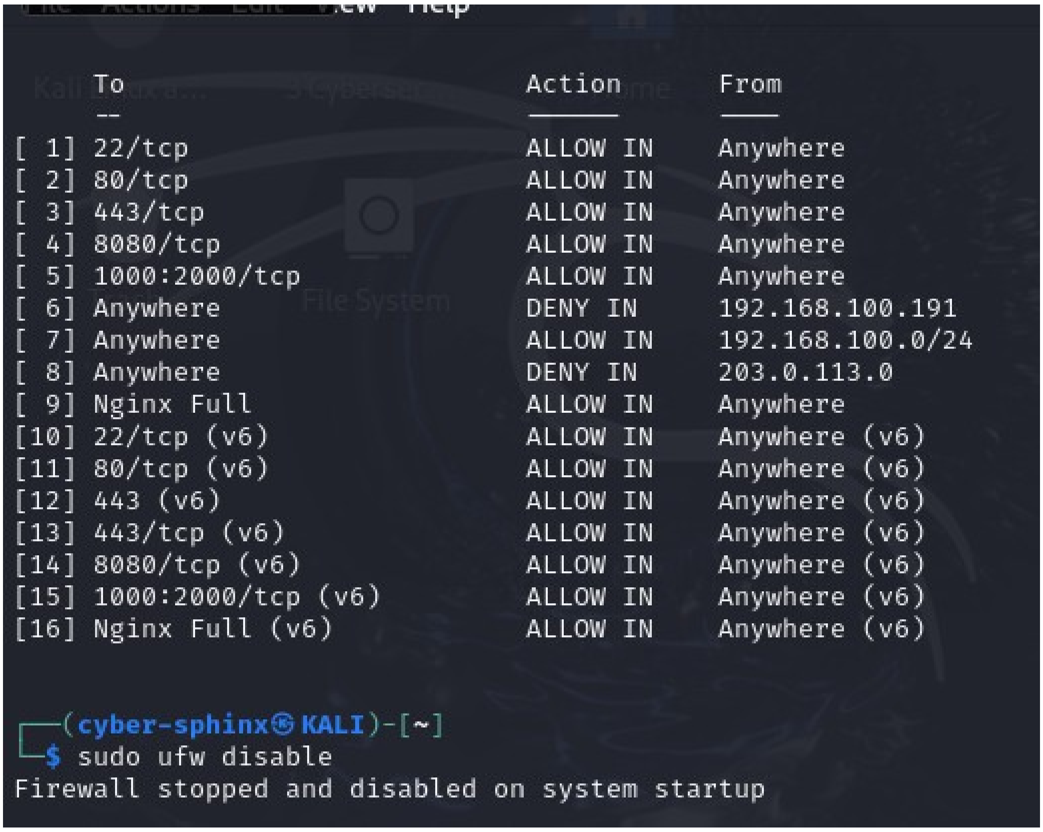
sudo ufw enable

Adding Rules: The following rules were set up on the Kali Linux VM:

```
sudo ufw allow 22/tcp
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw allow 8080/tcp
sudo ufw allow 1000:2000/tcp
sudo ufw allow from 192.168.100.191
sudo ufw allow from 192.168.100.0/24
sudo ufw deny from 203.0.113.0
sudo ufw allow 22/tcp (v6)
sudo ufw allow 80/tcp (v6)
sudo ufw allow 443/tcp (v6)
sudo ufw allow 8080/tcp (v6)
sudo ufw allow 1000:2000/tcp (v6)
```

Verifying Rules: To verify that the rules have been successfully added, run:

sudo ufw status numbered
This command will list all active rules in the firewall.



```
     To                         Action      From
     --                         ------      ----
[ 1] 22/tcp                     ALLOW IN    Anywhere
[ 2] 80/tcp                     ALLOW IN    Anywhere
[ 3] 443/tcp                    ALLOW IN    Anywhere
[ 4] 8080/tcp                   ALLOW IN    Anywhere
[ 5] 1000:2000/tcp              ALLOW IN    Anywhere
[ 6] Anywhere                   DENY IN     192.168.100.191
[ 7] Anywhere                   ALLOW IN    192.168.100.0/24
[ 8] Anywhere                   DENY IN     203.0.113.0
[ 9] Nginx Full                 ALLOW IN    Anywhere
[10] 22/tcp (v6)                ALLOW IN    Anywhere (v6)
[11] 80/tcp (v6)                ALLOW IN    Anywhere (v6)
[12] 443 (v6)                   ALLOW IN    Anywhere (v6)
[13] 443/tcp (v6)               ALLOW IN    Anywhere (v6)
[14] 8080/tcp (v6)              ALLOW IN    Anywhere (v6)
[15] 1000:2000/tcp (v6)         ALLOW IN    Anywhere (v6)
[16] Nginx Full (v6)            ALLOW IN    Anywhere (v6)


┌──(cyber-sphinx㉿KALI)-[~]
└─$ sudo ufw disable
Firewall stopped and disabled on system startup
```

4. Testing Firewall Rules

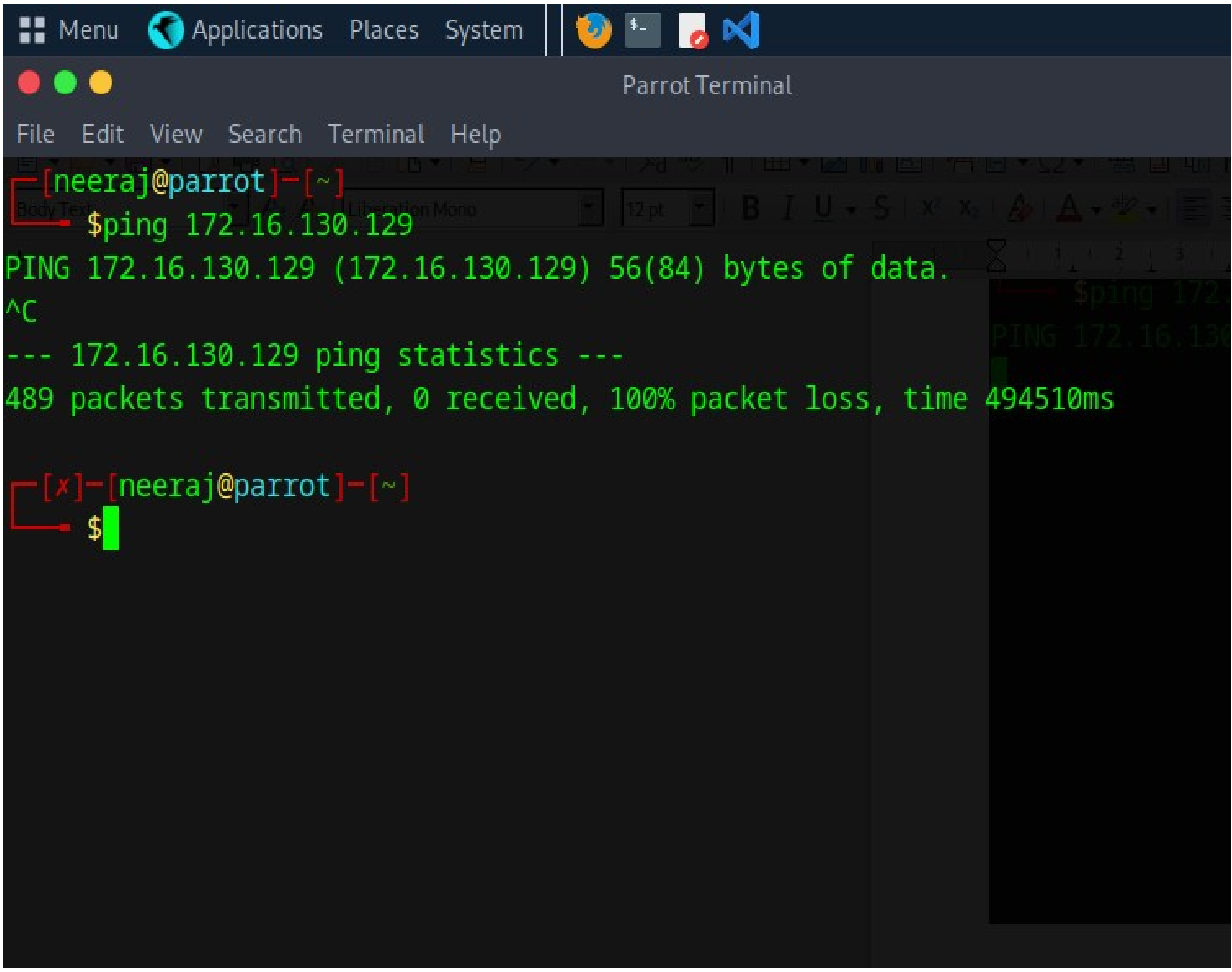
Ping Test: To test the deny rule for your own IP, attempt to ping the device's own IP from the other device:

Like Parrot os (host machine)

Kali os (firewall protected machine)

ping <your_virtual_machine_ip>

The ip of the machine which is protected with firewall

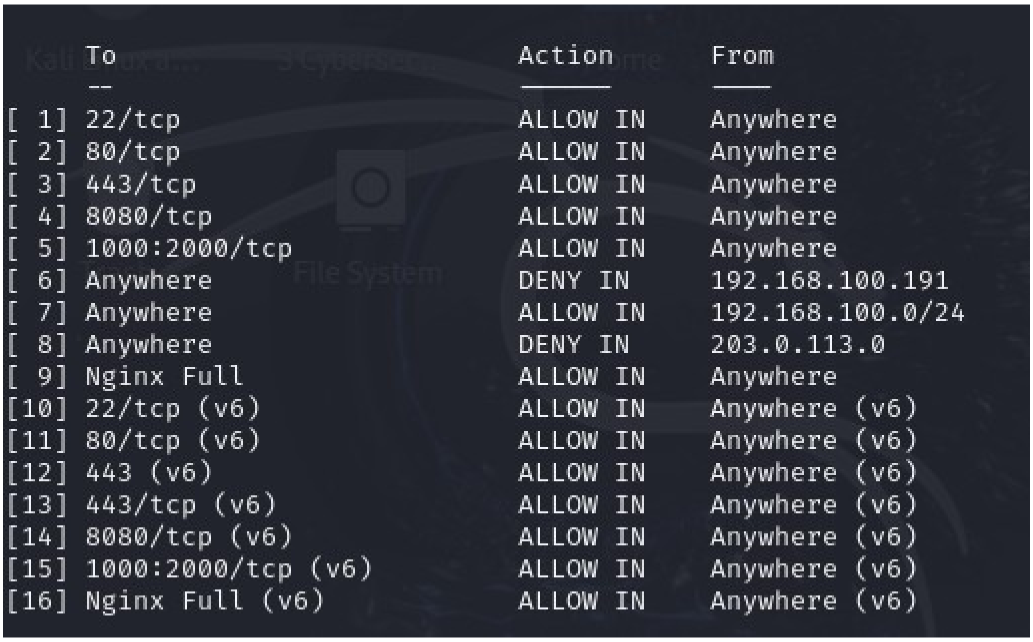


```
┌─[neeraj@parrot]─[~]
└──╼ $ping 172.16.130.129
PING 172.16.130.129 (172.16.130.129) 56(84) bytes of data.
^C
--- 172.16.130.129 ping statistics ---
489 packets transmitted, 0 received, 100% packet loss, time 494510ms

┌─[✗]─[neeraj@parrot]─[~]
└──╼ $
```

I

have set the deny rule for my host ip(parrot os)



```
     To                         Action      From
     --                         ------      ----
[ 1] 22/tcp                     ALLOW IN    Anywhere
[ 2] 80/tcp                     ALLOW IN    Anywhere
[ 3] 443/tcp                    ALLOW IN    Anywhere
[ 4] 8080/tcp                   ALLOW IN    Anywhere
[ 5] 1000:2000/tcp              ALLOW IN    Anywhere
[ 6] Anywhere                   DENY IN     192.168.100.191
[ 7] Anywhere                   ALLOW IN    192.168.100.0/24
[ 8] Anywhere                   DENY IN     203.0.113.0
[ 9] Nginx Full                 ALLOW IN    Anywhere
[10] 22/tcp (v6)                ALLOW IN    Anywhere (v6)
[11] 80/tcp (v6)                ALLOW IN    Anywhere (v6)
[12] 443 (v6)                   ALLOW IN    Anywhere (v6)
[13] 443/tcp (v6)               ALLOW IN    Anywhere (v6)
[14] 8080/tcp (v6)              ALLOW IN    Anywhere (v6)
[15] 1000:2000/tcp (v6)         ALLOW IN    Anywhere (v6)
[16] Nginx Full (v6)            ALLOW IN    Anywhere (v6)
```

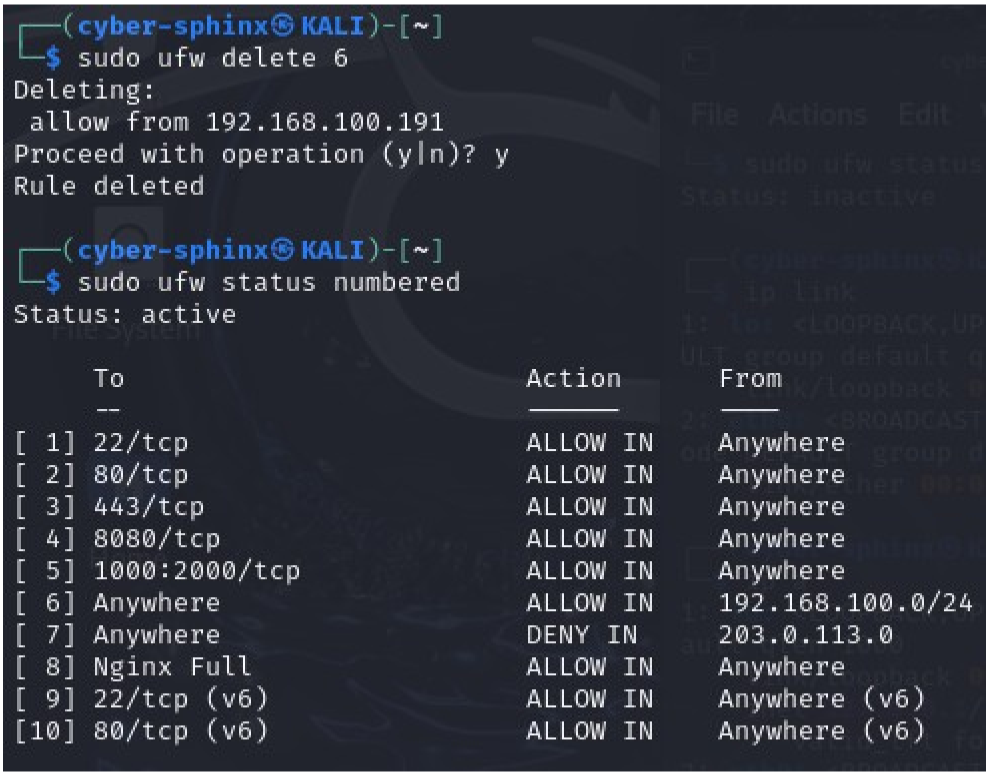
Nmap Scanning: To test the firewall rules using `nmap`, run from the other device:

Regular Scan: nmap 172.16.130.129

- **Filtered Ports:** `784` ports are filtered, indicating that the firewall is blocking traffic and not allowing responses**.**
- **Closed Ports:** `216` ports are closed, rejecting connection attempts.

SYN Scan: sudo nmap -sS 172.16.130.129

- **Ignored States:** Similar to the regular scan, all 1000 scanned ports are in ignored states. This indicates that the firewall is effectively filtering or blocking traffic.
- **Filtered Ports:** `784` ports are filtered, meaning the firewall rules are blocking traffic and not allowing responses.
- **Closed Ports:** `216` ports are closed, meaning these ports are not open and connection attempts have been reset.
- **MAC Address:** The MAC address of the virtual machine's network adapter is also provided, which can be useful for network analysis.

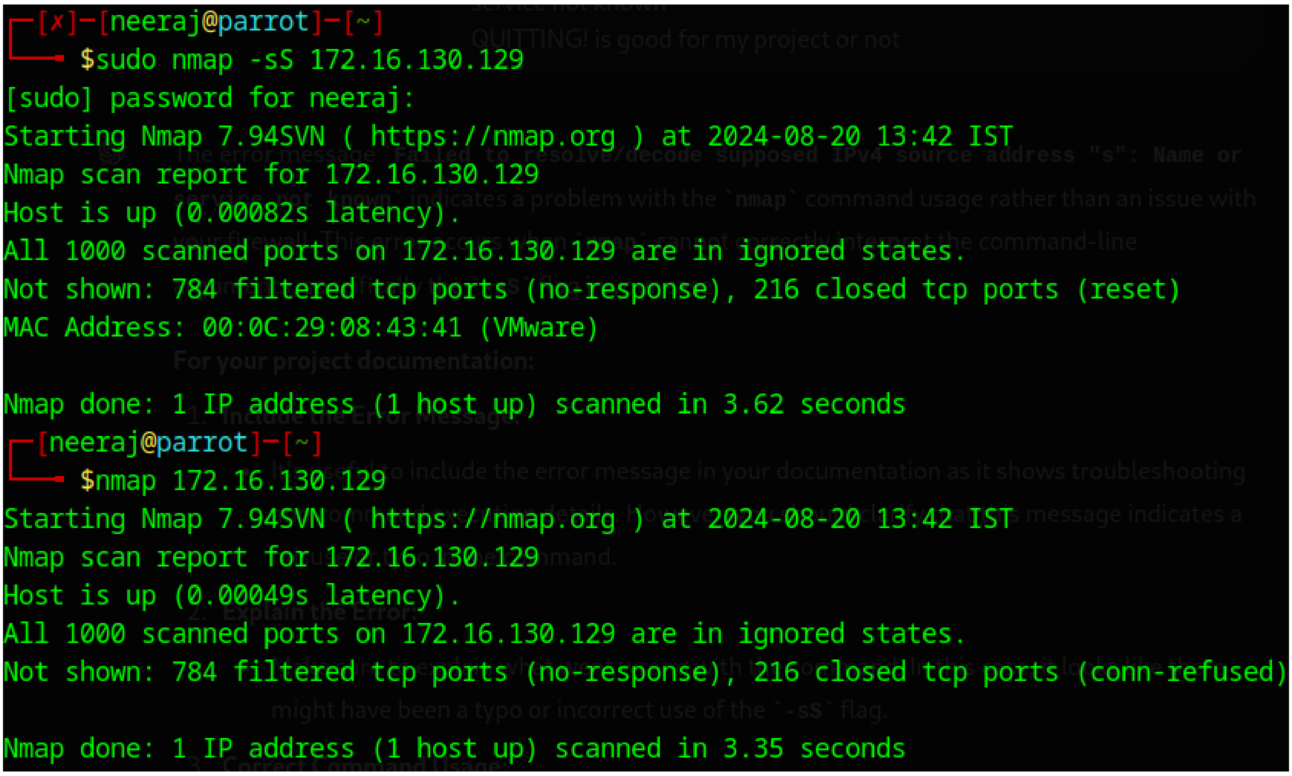


```
┌──(cyber-sphinx㉿KALI)-[~]
└─$ sudo ufw delete 6
Deleting:
 allow from 192.168.100.191
Proceed with operation (y|n)? y
Rule deleted

┌──(cyber-sphinx㉿KALI)-[~]
└─$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22/tcp                     ALLOW IN    Anywhere
[ 2] 80/tcp                     ALLOW IN    Anywhere
[ 3] 443/tcp                    ALLOW IN    Anywhere
[ 4] 8080/tcp                   ALLOW IN    Anywhere
[ 5] 1000:2000/tcp              ALLOW IN    Anywhere
[ 6] Anywhere                   ALLOW IN    192.168.100.0/24
[ 7] Anywhere                   DENY IN     203.0.113.0
[ 8] Nginx Full                 ALLOW IN    Anywhere
[ 9] 22/tcp (v6)                ALLOW IN    Anywhere (v6)
[10] 80/tcp (v6)                ALLOW IN    Anywhere (v6)
```

I have deleted here my host ip from ufw rule so that host and virtual machine can communicate.



```
┌─[✗]─[neeraj@parrot]─[~]
└──╼ $sudo nmap -sS 172.16.130.129
[sudo] password for neeraj:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-20 13:42 IST
Nmap scan report for 172.16.130.129
Host is up (0.00082s latency).
All 1000 scanned ports on 172.16.130.129 are in ignored states.
Not shown: 784 filtered tcp ports (no-response), 216 closed tcp ports (reset)
MAC Address: 00:0C:29:08:43:41 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 3.62 seconds
┌─[neeraj@parrot]─[~]
└──╼ $nmap 172.16.130.129
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-20 13:42 IST
Nmap scan report for 172.16.130.129
Host is up (0.00049s latency).
All 1000 scanned ports on 172.16.130.129 are in ignored states.
Not shown: 784 filtered tcp ports (no-response), 216 closed tcp ports (conn-refused)

Nmap done: 1 IP address (1 host up) scanned in 3.35 seconds
```

5. Troubleshooting

Common Issues:

- **UFW Rules Not Applying:** Ensure that UFW is enabled and the rules are added correctly. Restarting the UFW service might resolve some issues.

  sudo ufw reload

- **Unexpected Nmap Results:** If `nmap` shows unexpected results, verify the IP address and subnet configurations.

Debugging Tips:

- **Log Analysis:** Check UFW logs to see which rules are being triggered.

  sudo ufw status verbose

6. Conclusion

Project Summary: In this project, a firewall was successfully configured on a Kali Linux virtual machine using UFW. The setup involved allowing and denying specific traffic and was validated using `ping` and `nmap` tests. The firewall performed as expected, blocking or allowing traffic according to the configured rules.

Next Steps:

- **Advanced Firewall Configurations:** Experiment with more complex firewall rules.
- **Integration:** Combine UFW with other security tools for enhanced protection.

7. Appendix

References:

- UFW Documentation: UFW Official Documentation
- Nmap Documentation: Nmap Official Documentation

Command Reference:

- `sudo ufw enable`: Enables the firewall.
- `sudo ufw allow <port>/tcp`: Allows TCP traffic on a specific port.
- `sudo ufw deny from <IP>`: Denies all traffic from a specific IP address.
- `nmap -sS <target_ip>`: Performs a stealth scan on the target IP.