# Firewall Configuration Project Report

## Project Overview

The idea was to build a firewall for a web server running on a Kali Linux VM (with the IP address 192.168.213.128). I wanted to make sure the server could handle web traffic (like people visiting a website) and allow me to connect remotely, but block everything else to keep it safe from attacks. I used my Windows laptop as the host machine, running the VM through VMware, and worked entirely from the VM’s console to avoid any connection issues. This project was part of my portfolio to show I can handle network security tasks, and I aimed to make it practical, well-tested, and something that would impress potential employers.

## Steps Taken

I broke the project into three main phases: getting everything ready, setting up the basic firewall, and adding advanced security features. Here’s how it went:

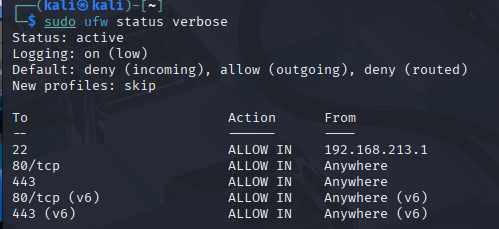
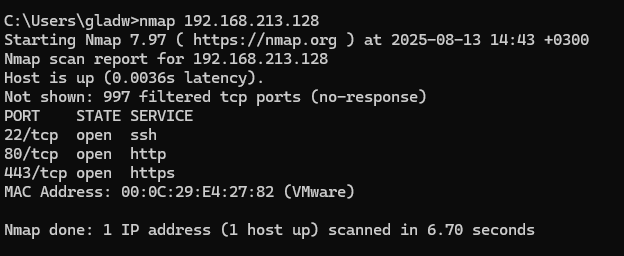
### Phase 1: Preparation

First, I set up my Kali Linux VM in VMware. I made sure the system was up to date by running commands to update and upgrade the software, which is like giving the VM a quick tune-up to avoid any glitches. I installed UFW, a tool that makes firewall setup straightforward, and Apache, which is the software that runs the web server. I also installed nmap, a tool to scan open ports, on my Windows laptop to test the firewall later. To be safe, I took a snapshot of the VM in VMware, so if anything went wrong, I could roll back and start over. Everything went smoothly, and I could see the Apache welcome page when I typed http://192.168.213.128 into my browser.

### Phase 2: Basic Firewall Setup

Next, I got to work on the firewall itself. My plan was to block all incoming traffic by default, only allowing specific connections for SSH (to log into the VM remotely), HTTP (for web browsing), and HTTPS (for secure web browsing). Here’s what I did:

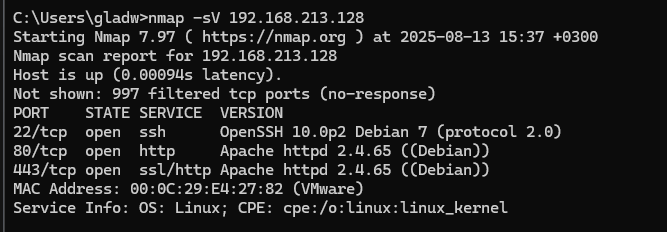
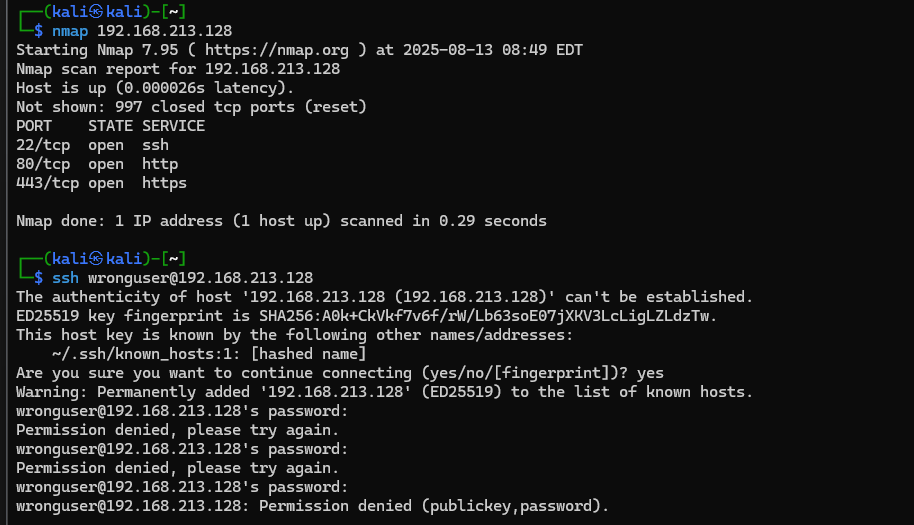
* **Set Default Rules**: I told UFW to block all incoming traffic and allow all outgoing traffic. This is like locking all the doors to the house but letting people inside send mail out.
* **Allowed Specific Connections**: I opened ports for SSH (port 22), HTTP (port 80), and HTTPS (port 443). Since I was working directly in the VM’s console, I didn’t have to worry about locking myself out, but I still restricted SSH to my laptop’s IP (192.168.213.1) for extra security.
* **Enabled the Firewall**: I turned on UFW and checked the rules with sudo ufw status verbose. The output showed exactly what I wanted: only ports 22, 80, and 443 were open.
* **Tested It**: I opened my browser and went to http://192.168.213.128—the Apache page loaded perfectly. I ran an nmap scan from my Windows laptop (nmap 192.168.213.128) to confirm only those three ports were open. At first, HTTPS (port 443) showed as closed, and SSH didn’t connect from my laptop, but I fixed those issues (more on that later).



### Phase 3: Advanced Security Features

Once the basics were working, I wanted to make the firewall even stronger and show off some advanced skills. Here’s what I added:

* **Rate Limiting for SSH**: I set up a rule to limit how many times someone could try to log in via SSH. This helps stop hackers from guessing passwords over and over.
* **Detailed Logging**: I turned on high-level logging in UFW to keep track of blocked traffic. This way, I could see if anyone was trying to sneak in.
* **Fail2Ban**: I installed Fail2Ban, a tool that watches for suspicious activity (like too many failed login attempts) and bans the offending IP address. I set it up to protect SSH, and after testing with fake login attempts from my laptop, I saw it ban my IP, which was exciting proof it worked!
* **Automation Script**: I wrote a bash script called firewall\_setup.sh to apply all my rules automatically. This makes it easy to set up the firewall again or show others how I did it. The script includes all the rules, rate limiting, and logging settings.



### Challenges and How I Solved Them

This project wasn’t without hiccups, but solving them taught me a lot. Here are the main challenges:

* **SSH Connection Issues**: When I tried connecting to the VM from my Windows laptop (ssh kali@192.168.213.128), I got a “Connection refused” error. I checked the SSH service with sudo systemctl status ssh and found it wasn’t running. I installed openssh-server, started the service, and made sure port 22 was listening with sudo netstat -tuln | grep :22. I also realized my laptop’s IP might not always be 192.168.213.1 if VMware was in Bridged mode, so I temporarily allowed SSH from all IPs to test, then locked it back down.
* **HTTPS Not Working**: The nmap scan showed port 443 as closed, even though UFW allowed it. I discovered Apache wasn’t set up for HTTPS, so I enabled the SSL module (sudo a2enmod ssl), activated the default SSL site, and restarted Apache. After that, https://192.168.213.128 loaded, and nmap showed port 443 open.
* **IPv6 Confusion**: UFW showed rules for IPv6, but my VM wasn’t using it. I checked with ip addr show | grep inet6, saw no IPv6 addresses, and disabled IPv6 in UFW by editing /etc/default/ufw to set IPV6=no. This cleaned up the rules.
* **Script Error**: When I tried to make my script executable (chmod +x firewall\_setup.sh), it failed because I hadn’t saved the file yet. I laughed at myself, created the script properly, and it ran perfectly.

### Results and Testing

The final setup is secure and works like a charm. The UFW rules block all incoming traffic except SSH (from my laptop’s IP), HTTP, and HTTPS. Nmap scans from my Windows laptop confirm only ports 22, 80, and 443 are open, which means I’ve locked down the VM tightly. Fail2Ban is watching SSH, ready to ban anyone trying too many logins. I tested this by attempting fake logins and saw my IP get banned, which I unbanned afterward. The automation script makes it easy to recreate the setup, and I’ve saved all outputs (UFW status, nmap scans, Fail2Ban logs) for my portfolio.

Here’s a sample of my final UFW status:

Status: active

Logging: on (high)

Default: deny (incoming), allow (outgoing), deny (routed)

To Action From

22 LIMIT IN 192.168.213.1

22 ALLOW IN 192.168.213.1

80/tcp ALLOW IN Anywhere

443/tcp ALLOW IN Anywhere

And my nmap scan:

PORT STATE SERVICE

22/tcp open ssh

80/tcp open http

443/tcp open https

### What I Learned

This project was a big step forward for me. I learned how to:

* Set up a firewall to protect a web server, balancing security and functionality.
* Troubleshoot issues like SSH and HTTPS not working, which taught me to check services and configurations carefully.
* Use tools like nmap to test my work, which felt like putting on a hacker’s hat (in a good way!).
* Automate tasks with a script, which saves time and looks professional.
* Document everything clearly, which I know will help when I share this with future employers.