**CSCE 5585: Secure Network Design and Implementation project**

**Virtual Lab Setup**

***Tools Used:***

*GNS3:* In network emulation and topology design.

*VMware:* As the hypervisor to support virtual machines needed for the project execution

***Setup Details:***

*GNS3 Server Configuration:*The GNS3 server was then installed by importing it on VMware as an OVA file.Stating used the GNS3 server linked with the GNS3 client for topological construction and control.

It also has web management capability that can be accessed and manage through its Web User Interface .

*Virtual Machine (VM) Integration:*Several VMware-based virtual machines were imported and integrated into the GNS3 environment:

*Attack Machine:* It is applied for security testing and penetration simulation practices.

*Ubuntu Server*: Used to install software firewall and IDS and database hosting service provider

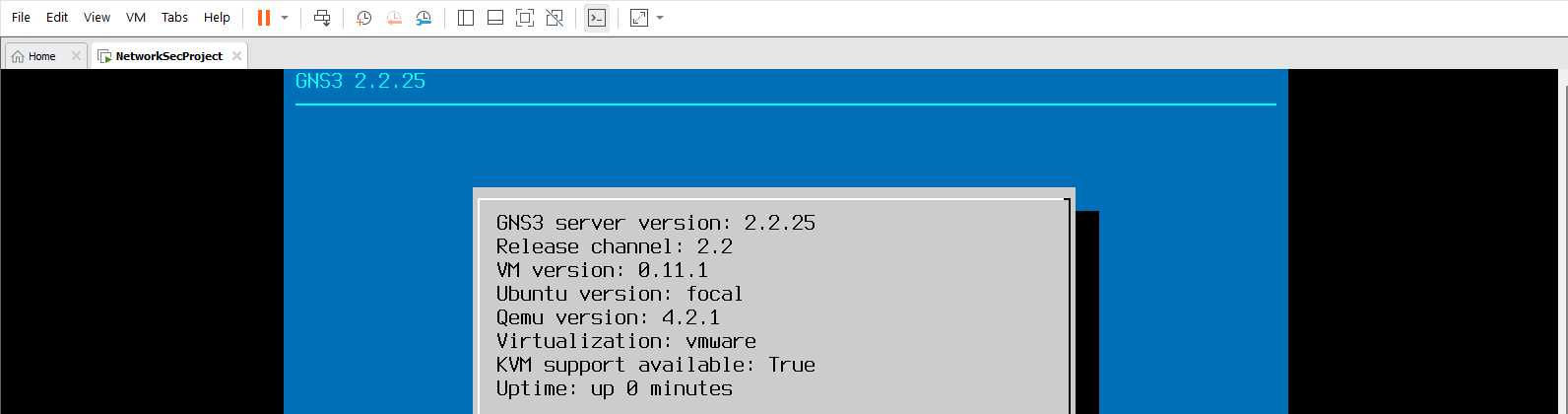
*Remote Machine*: Emulated an external entity that needs to connect to the network resources from outside.

***Network Emulation:***

The network topologies that were emulated in GNS3 were connected with the virtual machines that were being run in the VMware to make the results of the two interfaces compatible.

Each of the machines was connected to the correct network interface to be able to communicate with those VLANs, and the subnetworks.

*Verification:*Ensured that the GNS3 server and client where working by checking that they could both connect.Used the ping and traceroute command to make sure that VMware machines that were imported into the GNS3 topology work by connecting them.Opened GNS3 Web UI to check connection of the server and to follow the changes in the topology.



**Network Design and Segmentation**

The proposed architecture of the logical network was to have internet connectivity, internal departmental networks, the DMZ for public sectors, VPN for remote connections, and connection to the external world.

***Network Segmentation***

The network was divided into the following segments:

VLANs for Internal Departments and Guests:

* VLAN 10: IT Department
* VLAN 20: Finance Department
* VLAN 30: Guest Network

*DMZ (Demilitarized Zone):*responsible for hosting include Web and mail servers that require face interface with the outside public.

*Internal Network:*Special for the security of critical systems in enterprise such as the database server.

*Connectivity Details:*It was necessary to provide for a safe VLAN mechanism that allows for communication and isolates departments and the Guest network.

The DMZ was intended to be used to keep outside services away from the internal networks.

***Configuring VLANs***

Step by step VLAN was configured on the layer three switch for segmentation where each VLAN was to represent a given department or network.

*VLAN Configuration Steps*

Create VLANs on the Layer 3 Switch:

The VLANs were assigned proper IDs and were also given proper names.

*vlan 10*

*name IT*

*vlan 20*

*name Finance*

*vlan 30*

*name Guest*

*Assign VLANs to Devices:*

IT devices (desktops and servers) had topology access ports that were in VLAN 10.

Finance devices (desktops and laptops) were connected to access ports in VLAN 20.

The Guest network was created and placed in VLAN30 for those people who only have limited access on your network.

*interface ethernet 1/0*

*switchport mode access*

*switchport access vlan 10*

*description IT\_Desktop*

*interface ethernet 1/1*

*switchport mode access*

*switchport access vlan 20*

*description Finance\_Laptop*

*interface ethernet 1/2*

*switchport mode access*

*switchport access vlan 30*

*description Guest\_Device*

***Configure Inter-VLAN Routing:***

Sub-Interface were configured to enable inter-VLAN routing using Switched Virtual Interface (SVIs). They all have their own gateway IP address assigned for every VLAN.

*interface vlan 10*

*ip address 192.168.10.1 255.255.255.0*

*no shutdown*

*interface vlan 20*

*ip address 192.168.20.1 255.255.255.0*

*no shutdown*

*interface vlan 30*

*ip address 192.168.30.1 255.255.255.0*

*no shutdown*

***Control Inter-VLAN Communication:***

VLANs isolation was done through the implementation of the Access Control List (ACL). For instance, the Guest network (VLAN 30) was limited from accessing IT (VLAN 10) and Financial (VLAN 20) IT resources.

*ip access-list extended GUEST\_RESTRICTIONS*

*deny ip 192.168.30.0 0.0.0.255 192.168.10.0 0.0.0.255*

*deny ip 192.168.30.0 0.0.0.255 192.168.20.0 0.0.0.255*

*permit ip any any*

*interface vlan 30*

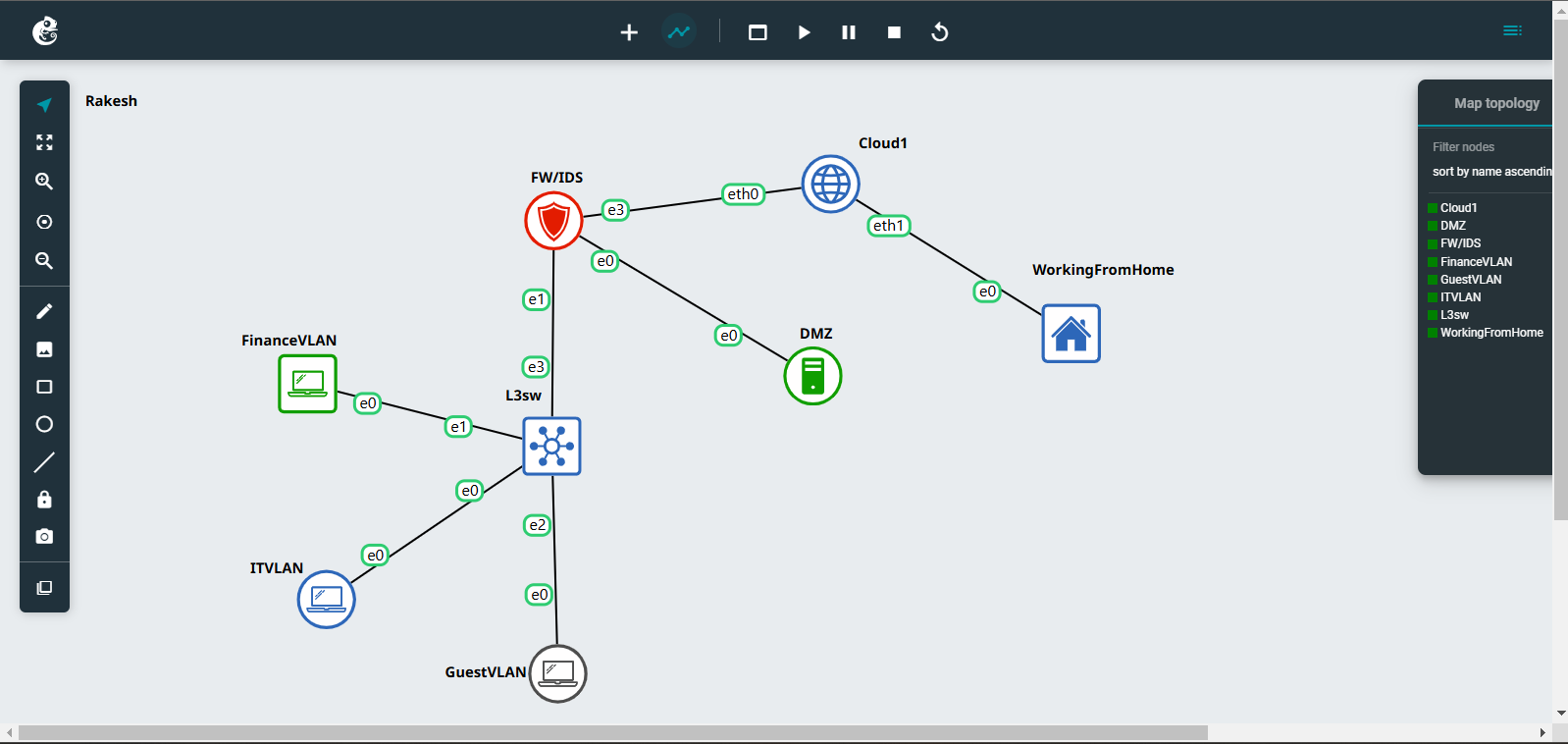
*ip access-group GUEST\_RESTRICTIONS in*

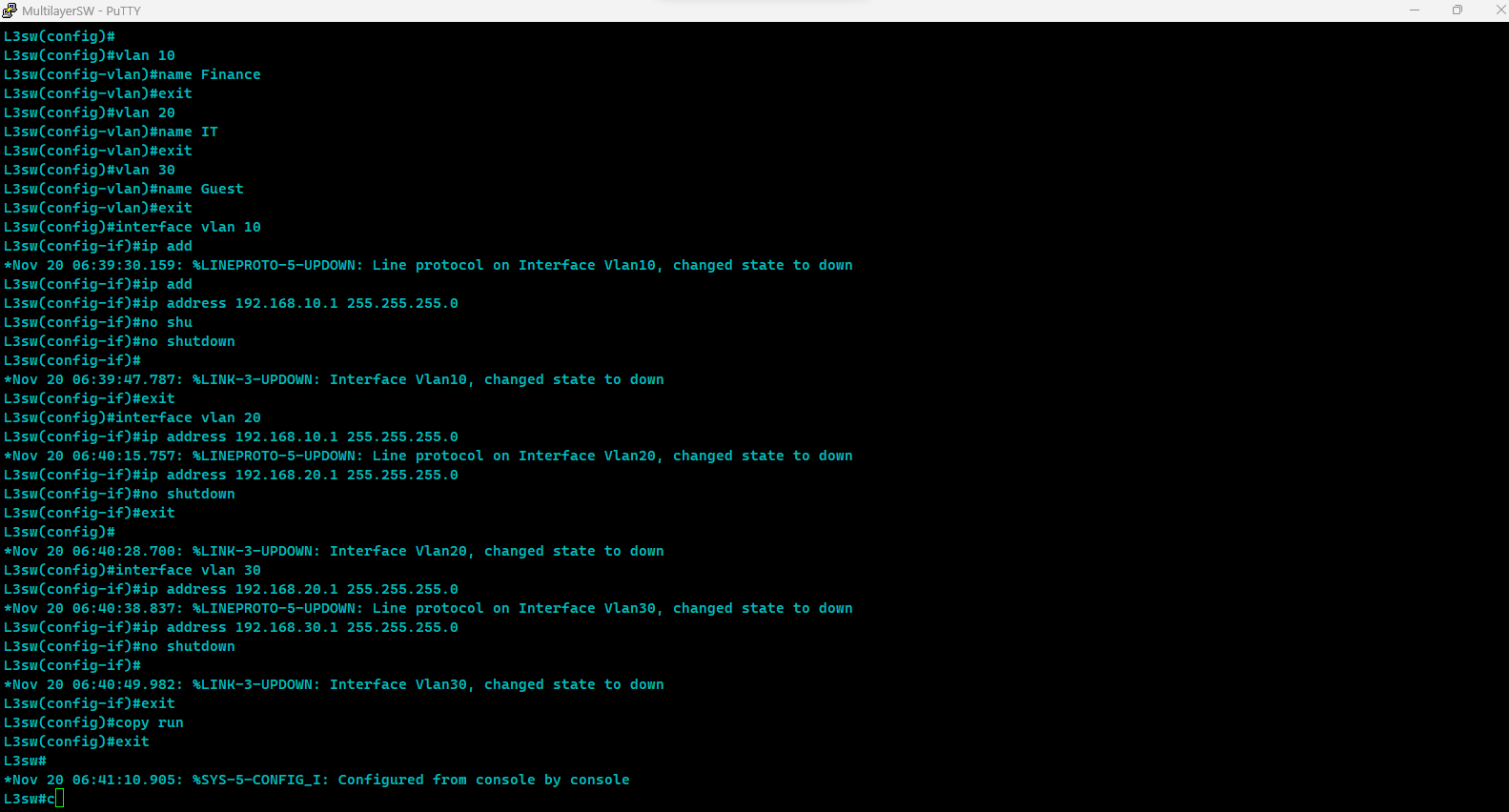
Three VLANs: There are four VLAN created namely, IT departments (VLAN 10), Finance (VLAN 20) and Guest (VLAN 30).

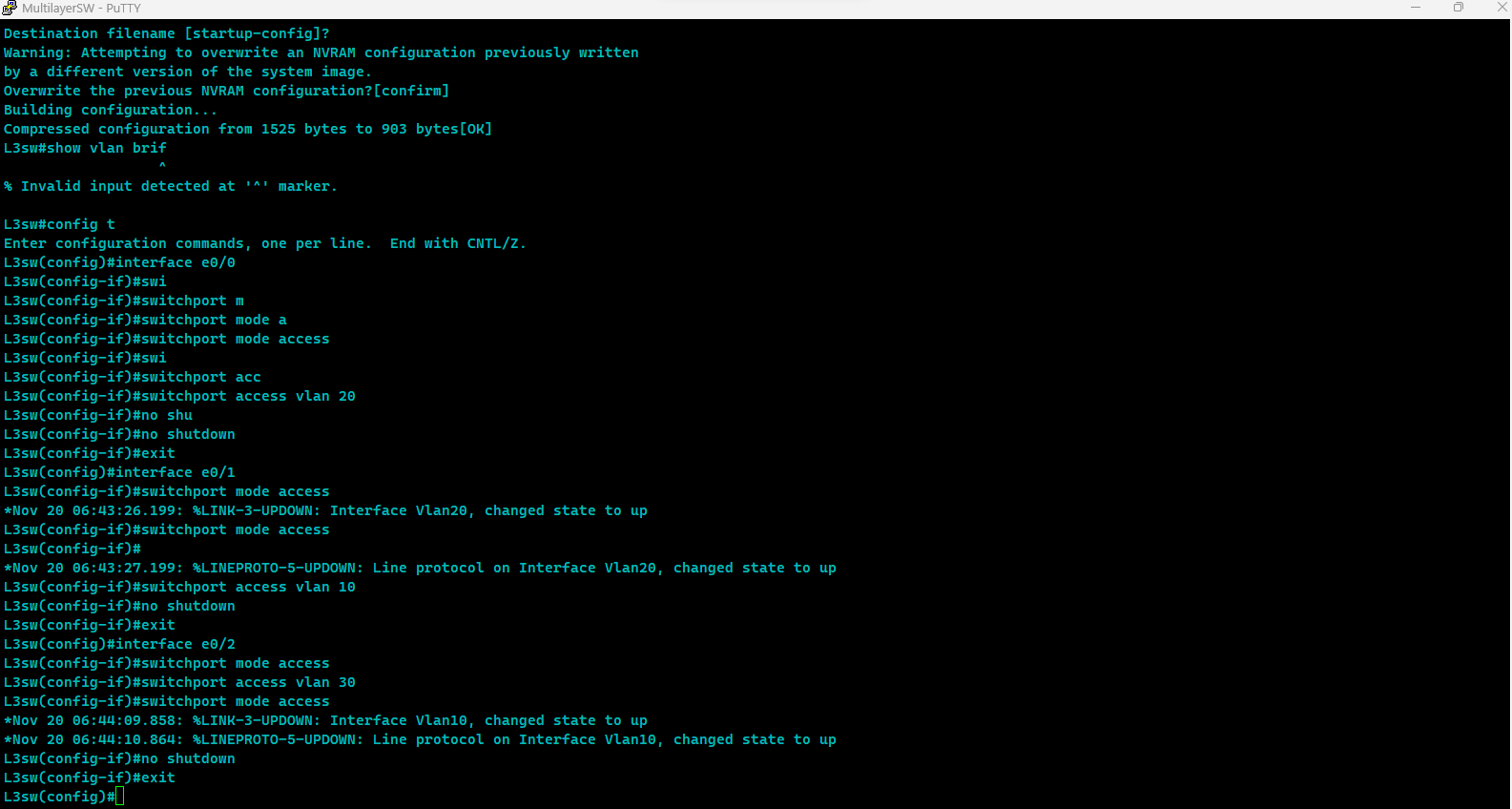
The DMZ is located at the familiar area where internal network and connections to external sources are made.

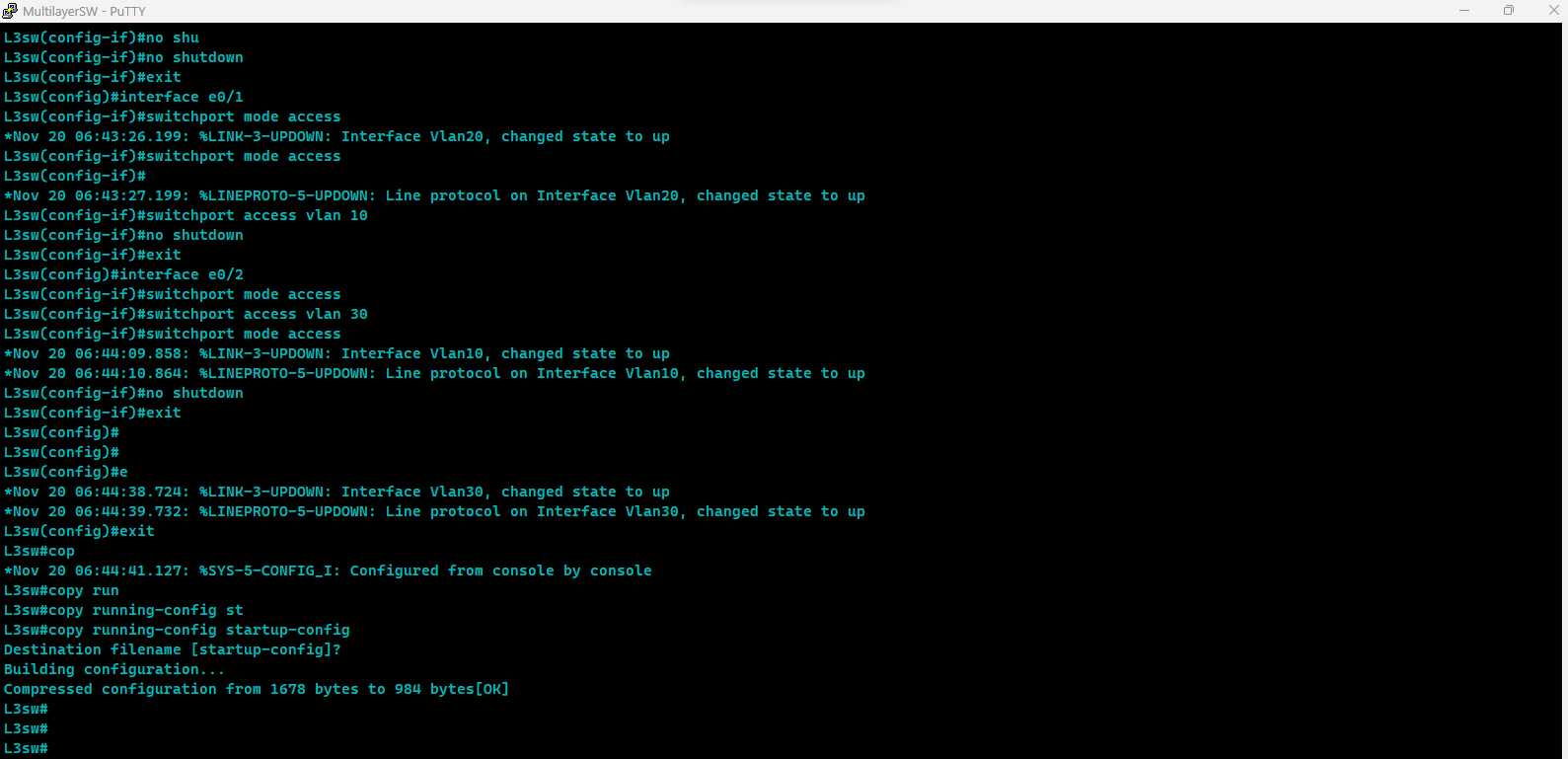
Fully identified as to the VLAN to which they belong each properly bipolar bit range.

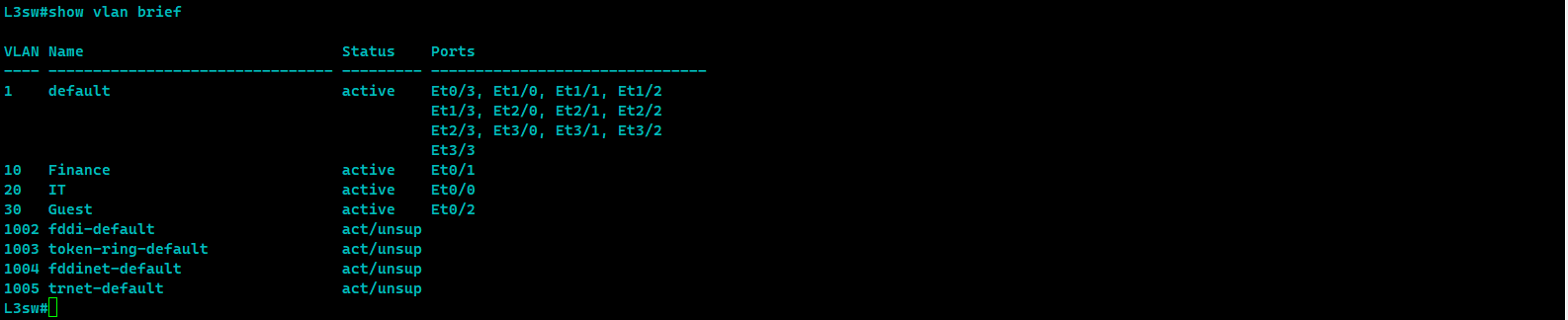
Configuration tools for inter VLAN communication routing paths and access controls.

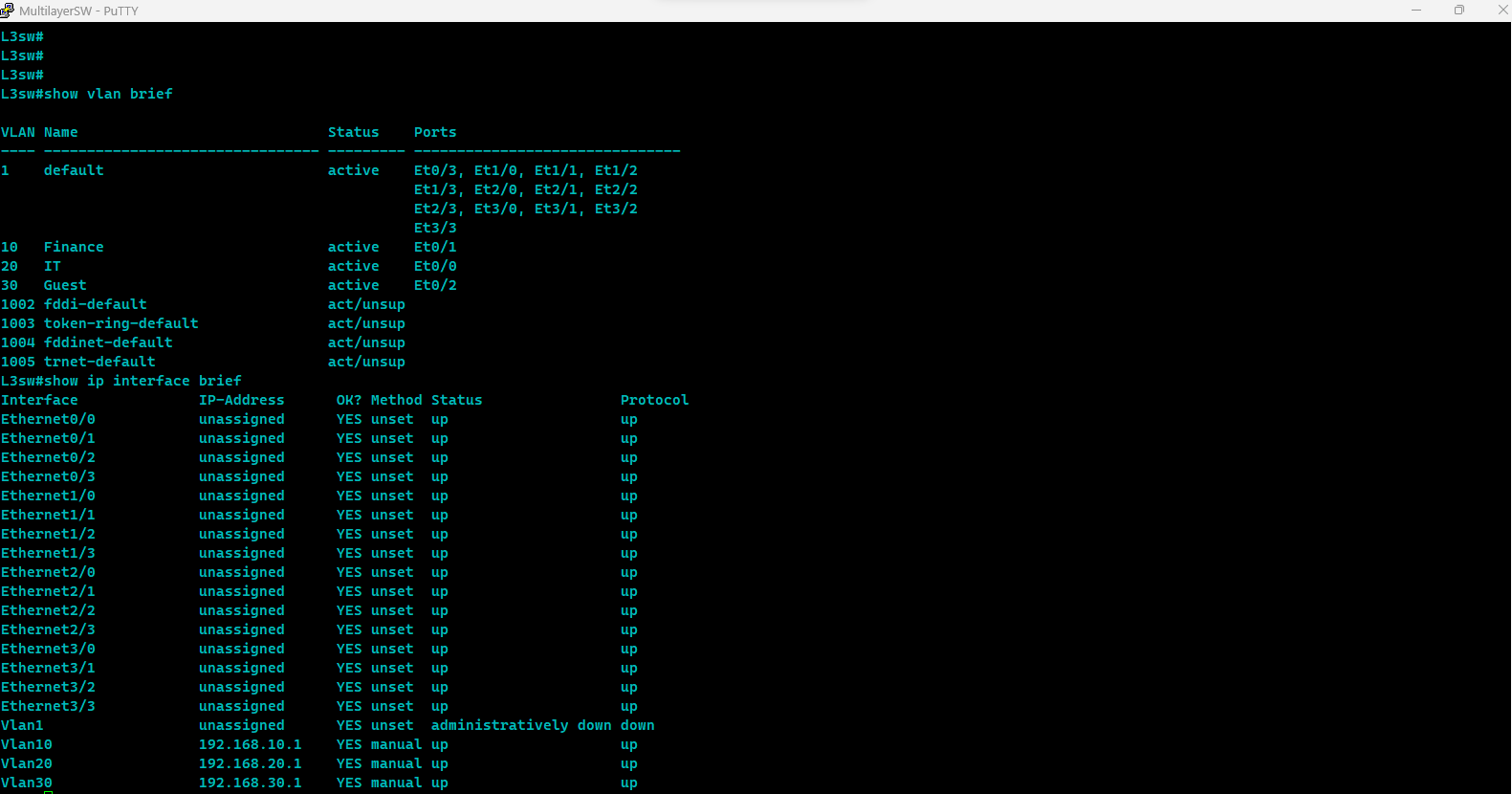


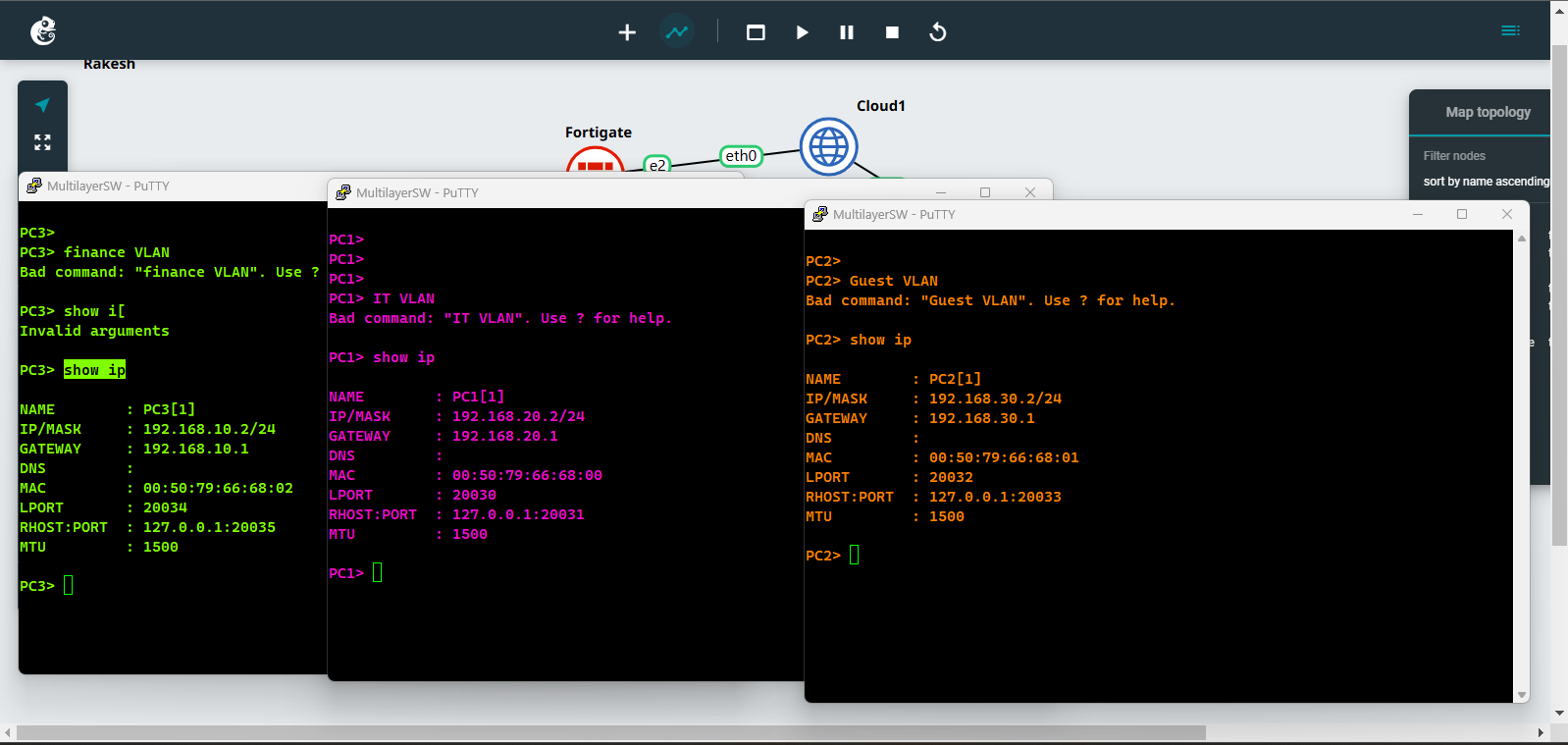
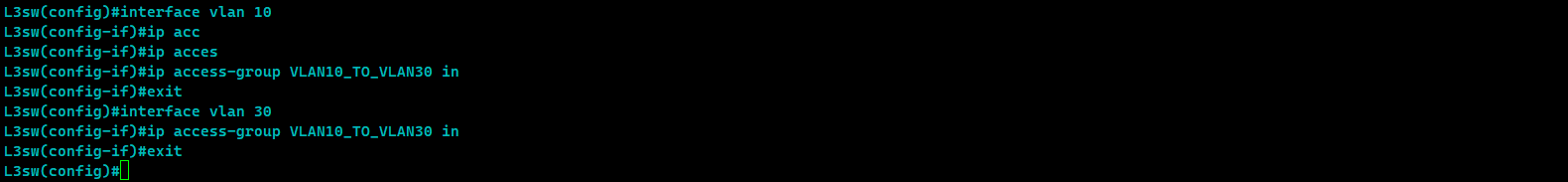
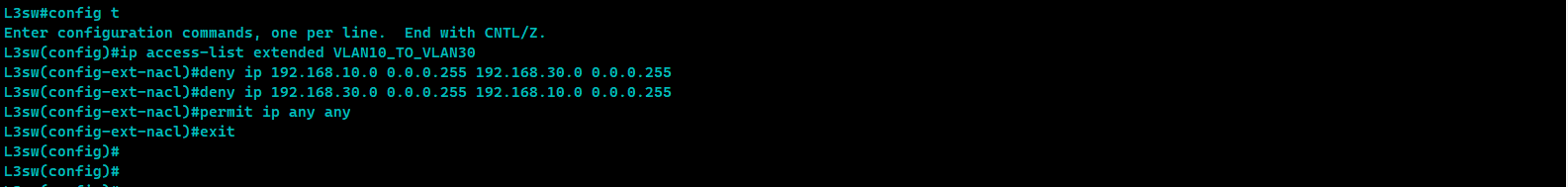
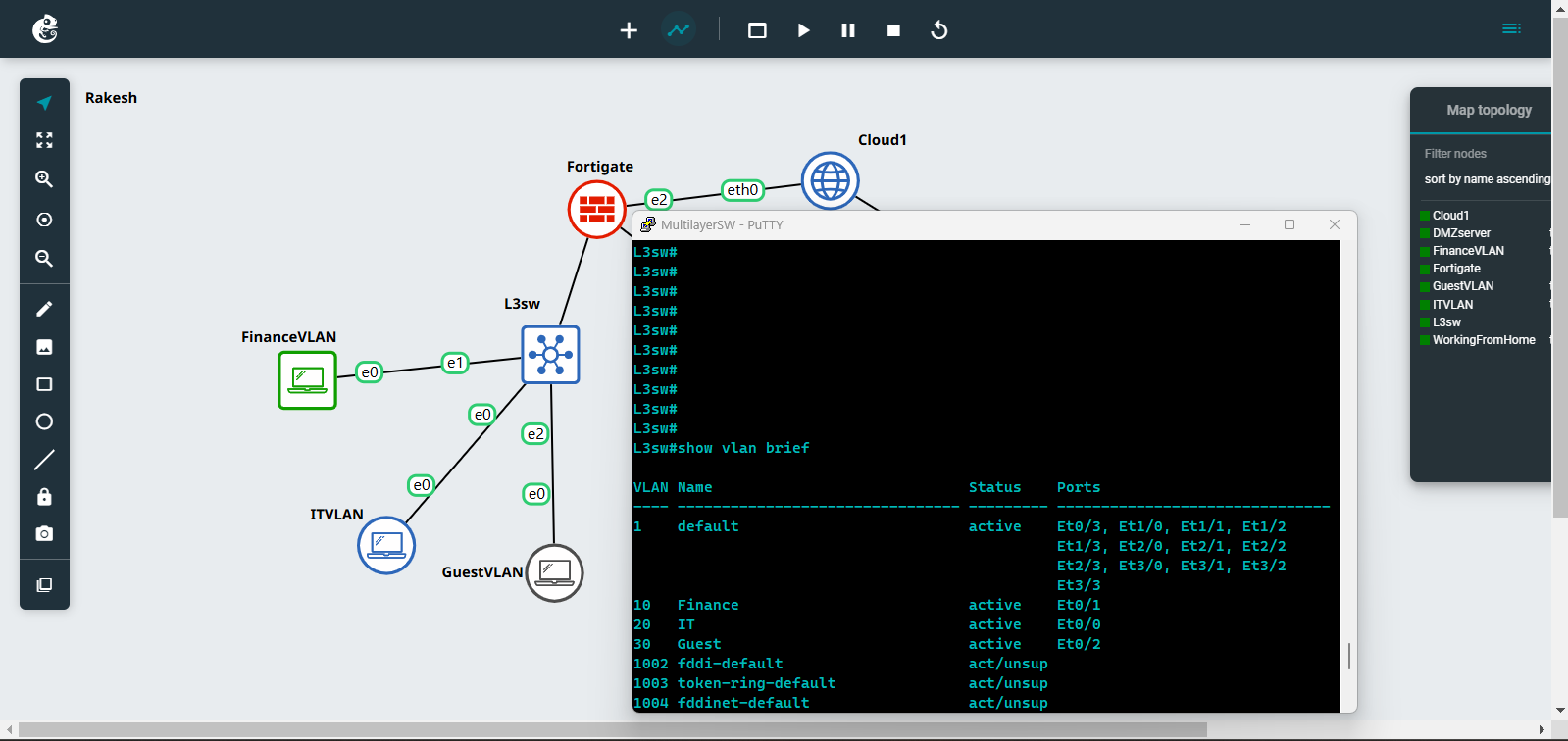
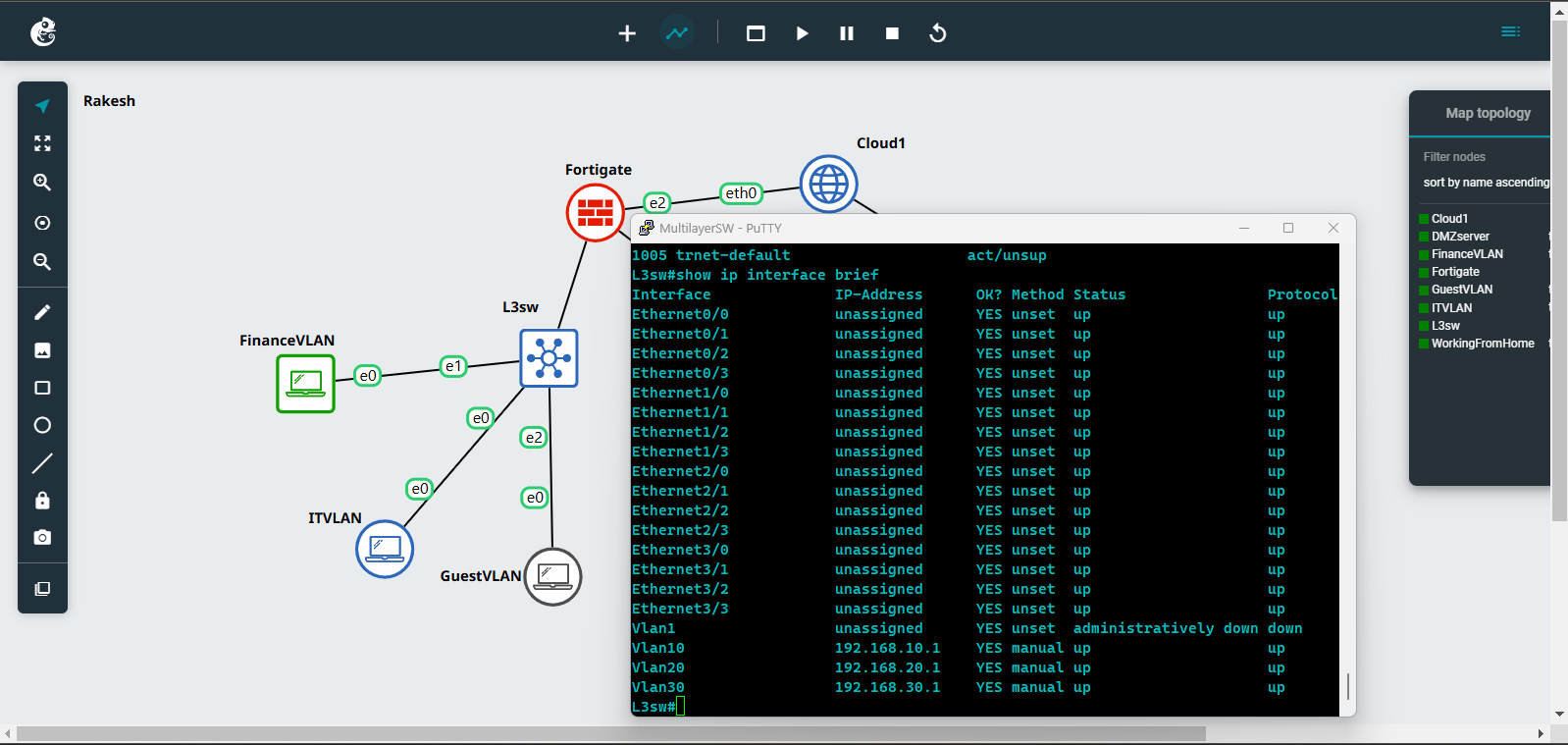


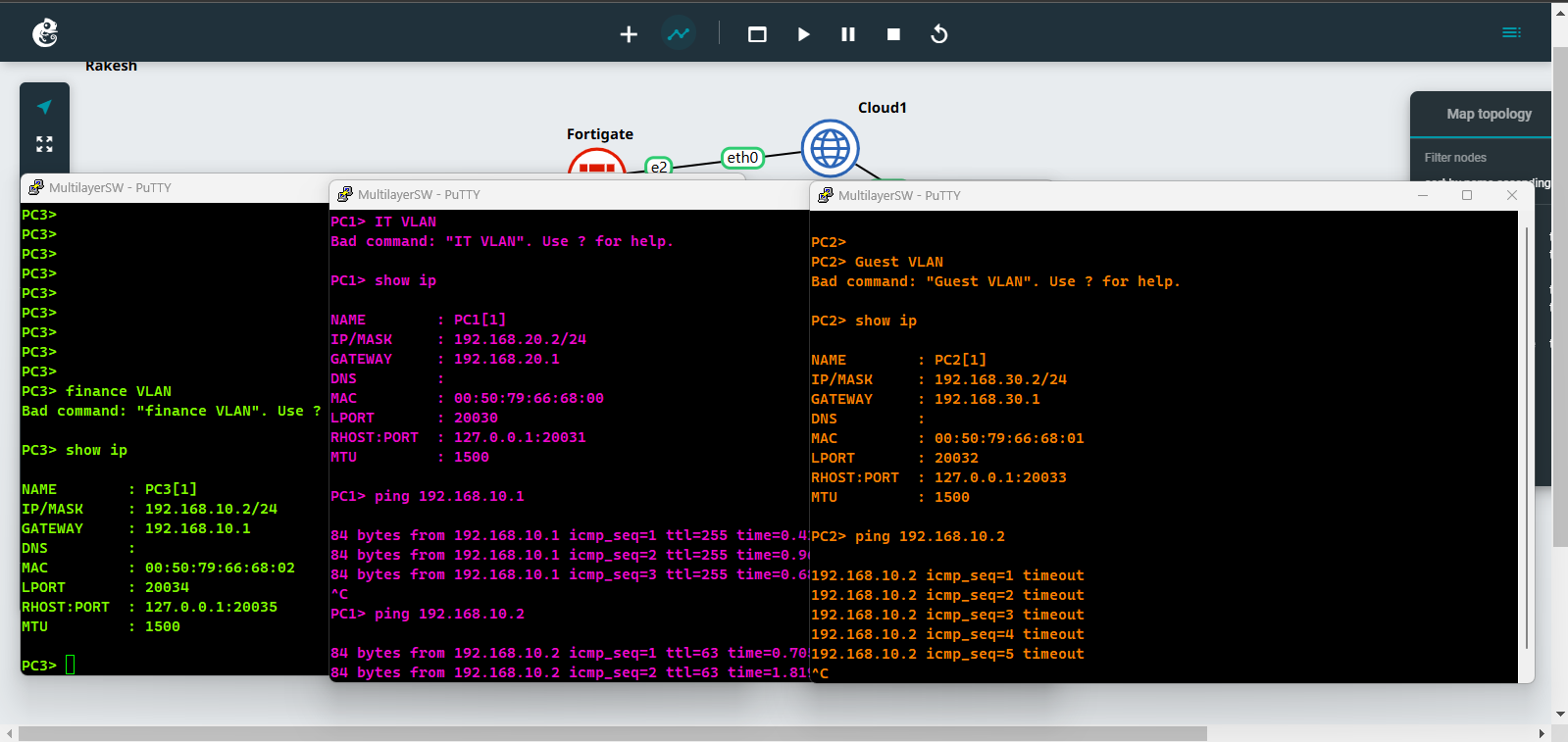












**Firewall Deployment**

To implement the processes, machines were configured such that an Ubuntu server served as the firewall and offered the DMZ for the honeypot and the web server. The network configuration specifically entails creation of WAN, LAN and DMZ interfaces, firewall policies and safe access to the LAN interface via SSH.

**Network Configuration:**

Managing Network Interfaces

The server has three network interfaces:

WAN of network connected to the internet is encompassed in the scenario as ens33.

ens34 – Local Area Network (internal organization network)

hrhhmrols35 – Proving grounds (for honeypot and web server)

It was used by Netplan, a tool for wilting up of Ubuntu network configuration.

***Netplan Configuration File***

The /etc/netplan/01-netcfg.yaml file was customized by changing the settings on the interface eth1 to have static IPs for both LAN and DMZ while the WAN interface would be set to DHCP.

The final network configuration is:

*network:*

*version: 2*

*ethernets:*

*ens33: # WAN interface*

*dhcp4: true*

*ens34: # LAN interface*

*addresses:*

*- 192.168.1.1/24*

*dhcp4: false*

*ens35: # DMZ interface*

*addresses:*

*- 192.168.2.1/24*

*dhcp4: false*

***Applying the Configuration***

After editing the file, we applied the new network configuration:

*sudo netplan apply*

This allowed for verification of the configuration of the interfaces (ens33, ens34, ens35) for the required static IPs and obtained through DHCP.

Installing and Configuring Firewall (iptables):

***Installing iptables***

We used another service called iptables to manage the firewall and made sure it would still load on reboot.

*sudo apt update*

Additionally, for changing the rules you need to run the following command to install iptables and iptables-persistent permanently: sudo apt install iptables iptables-persistent -y

This assures the users that the settings of the firewall are initialized for use as soon as the computer has booted.

***Basic iptables Configuration***

We created the following rules:

Default Policies: In the AD MX record, accept no traffic by default for any new connections or forwarded ones.

*sudo iptables -P INPUT DROP*

*sudo iptables -P FORWARD DROP*

*sudo iptables -P FORWARD ACCEPT*

*Allow loopback interface (localhost):*

Because existing as a root user, the cycle is performed as follows; sudo iptables -A INPUT -i lo -j ACCEPT According to me super user, sudo iptables -A OUTPUT -o lo -j ACCEPT

Allow established and related connections:

There is the following command to allow all incoming traffic which is coming from abroad through ESTABLISHED,RELATED state: sudo iptables -A INPUT -m conntrack —ctstate ESTABLISHED,RELATED -j ACCEPT

*sudo iptables -A FORWARD -m conntrack –ctstate ESTABLISHED,RELATED -j ACCEPT*

*Allow SSH on WAN (ens33):*

*sudo iptables -A INPUT -i ens33 -p tcp –dport 22 -j ACCEPT*

Allow LAN (ens34) to communicate with the DMZ (ens35):

New rule please type: sudo iptables -A FORWARD -i ens34 -o ens35 -j ACCEPT

The forwarding rule is as follows :

*sudo iptables -A FORWARD -i ens34 -o ens33 -j ACCEPT*

Allow DMZ (ens35) to access the internet via the WAN interface (ens33) for HTTP/HTTPS traffic:

*sudo iptables -A FORWARD -i ens35 -o ens33 -p tcp — dport 80 -j ACCEPT*

*sudo iptables -A FORWARD -i ens35 -o ens33 -p tcp — dport 443 -j ACCEPT*

Enable NAT (Network Address Translation) for LAN and DMZ to access the internet:

Next we need to change some parameters, in order to do that we need to have the root access as follows: *sudo iptables -t nat -A POSTROUTING -o ens33 -j MASQUERADE*

***Testing and Verification:***

Verifying iptables Rules

We used the following command to list the current firewall rules:

*sudo iptables -L -v -n*

This enabled us to confirm the $RULE$ where by we witnessed that traffic on port 22 was allowed to access the machine via SSH while other traffic was prohibited in as much as we implemented the other rules.

***Connectivity Testing***

Using ping, curl and wget we tested the internet connection from the LAN and confirm that it was live from the DMZ.

We also used Nmap to verify that the firewall rules were effectively blocking unwanted traffic and allowing the necessary services:

showing ports 22,80,443 scan them using

*nmap -sS -p 22,80,443 192.168.244.128*

***Logging Suspicious Activity:***

We added logging rules to monitor any dropped packets:

*sudo iptables -A INPUT -j LOG --log-prefix "IPTables-Dropped: " --log-level 4*

Logs are stored in /var/log/syslog and can be monitored using:

*sudo tail -f /var/log/syslog*

***Nmap Scan Testing***

As for the firewall configuration, we used Nmap port scanner in order to check if only the ports that should be opened are opened. All the testing was performed from a Kali Linux platform to probe the WAN interface, which is the IP address of the server (192.168.244.128).

Nmap Command Used

This can be done by using the following command;

*sudo nmap -p 22,443,80 192.168.244.128 –v -p 22,443,80*

The scan targets are defined to be SSH on port 22, HTTPS on port 443, HTTP on port 80.

-v: Enables verbose mode which provide output in more detail.

***Key Observations:***

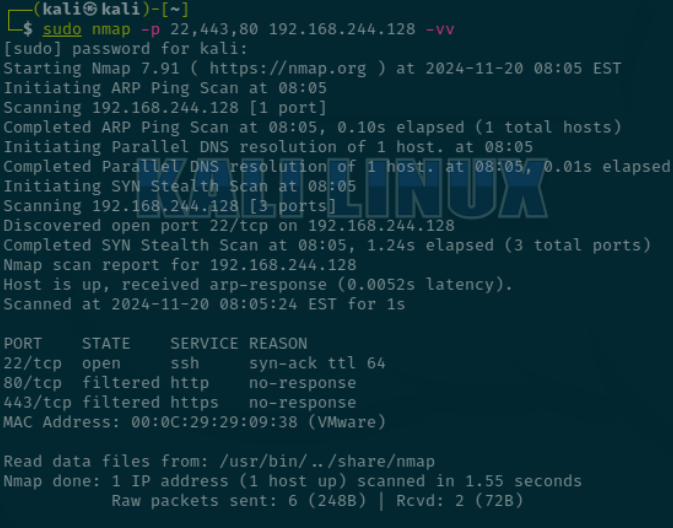
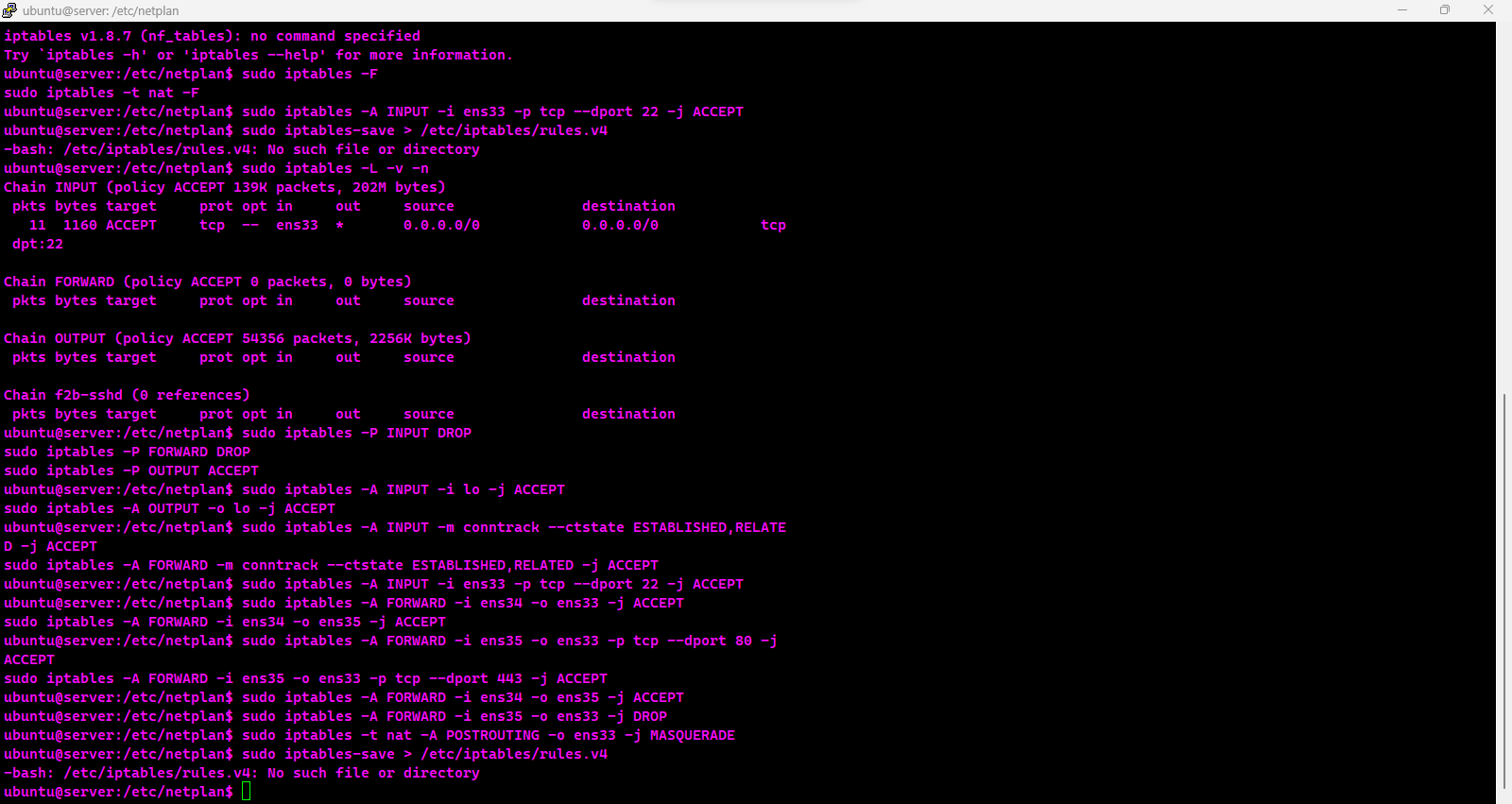
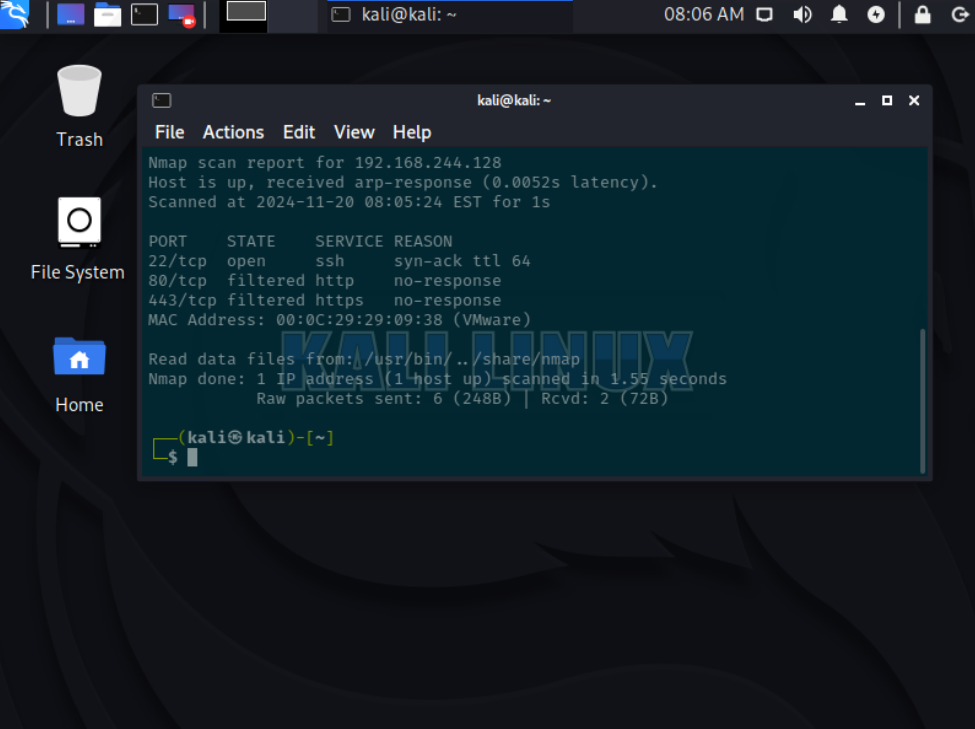
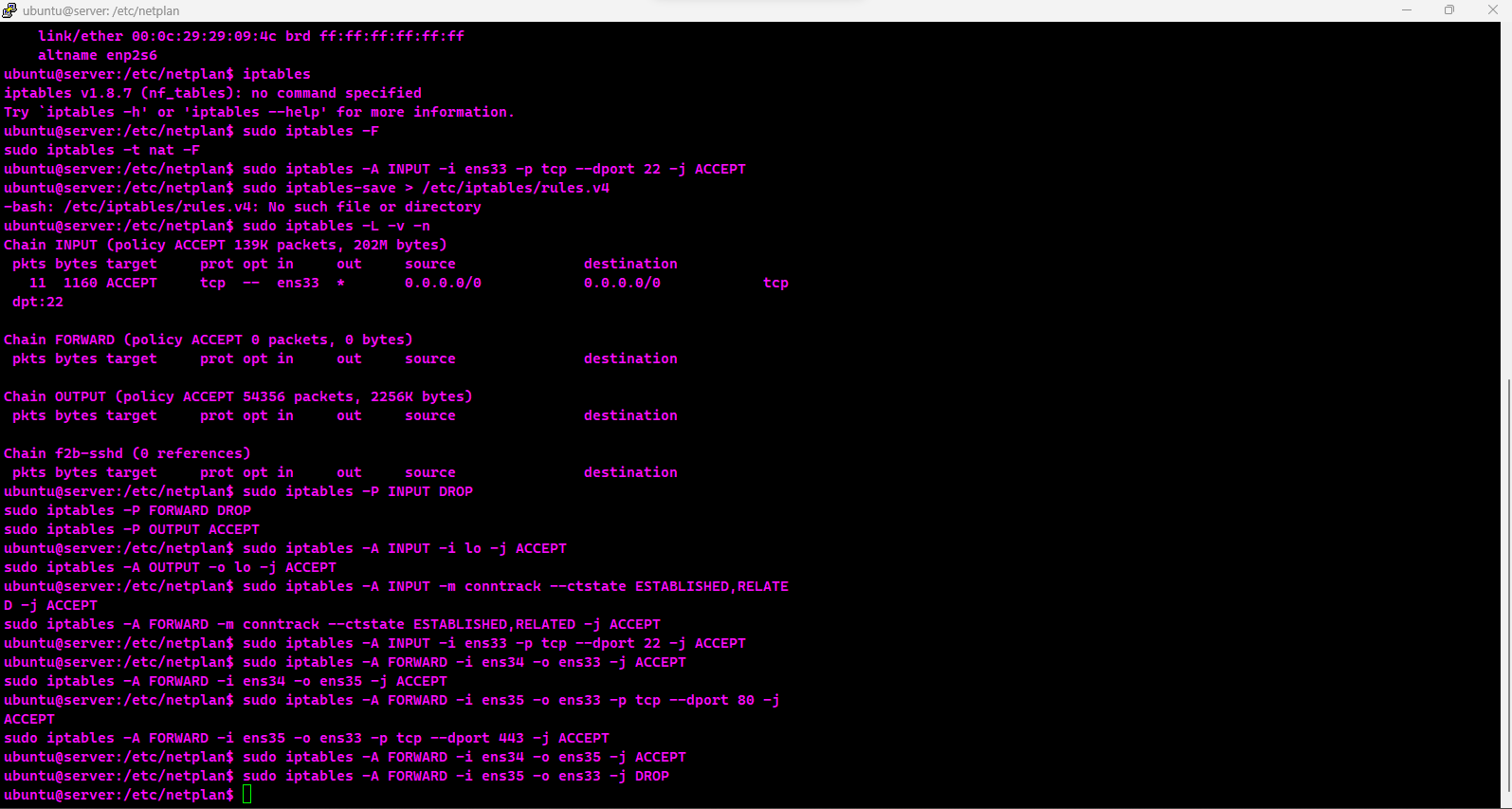
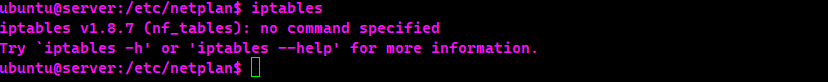
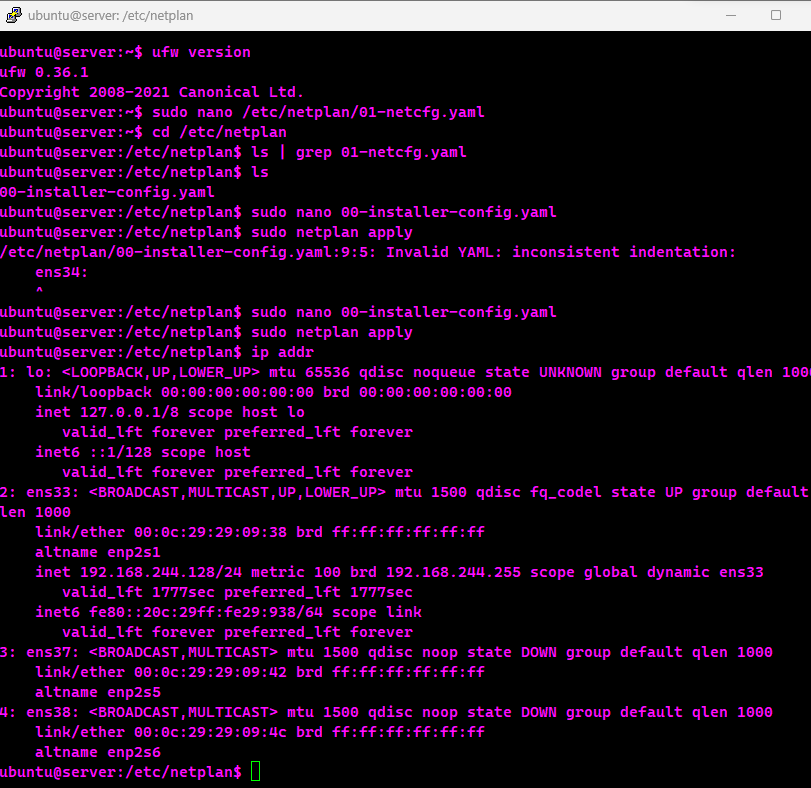
Port 22 (SSH): Utilizing the online scanner, the port scan probe found that port 22 is open, therefore SSH is permitted through the firewall on the WAN side. This is expected because when configuration the Linux server we only opened port number 22 for remote management.

Port 80 (HTTP): The scan also revealed that port 80 was filtered which meant that the firewall is preventing HTTP traffic, which accredits the setting of only allowing some traffic.

Port 443 (HTTPS): In the same way, port 443 was also filtered, therefore HTTPS is also banned, in compliance to DMZ and LAN firewall settings.

MAC Address: Also, the MAC address linked with the server is provided, which also shows that the server is being run on a VMware virtual machine.





**Open VPN Deployment in Ubuntu server**

The establishment of a secure OpenVPN for the purpose of allowing remote connection on an Ubuntu server is described. The concern was to establish and provide secure connection from client systems to the organization’s internal server, where valuable assets reside.Installation and configuration remain the most critical processes since they enable the software to meet the intended users’ needs fully.

To create an effectively protected VPN answer, OpenVPN was integrated into Easy-RSA software – for certificate and encryption management. Originally, the process involved upgrading the server so that it would work well with the current version and be secure.

Thus, there was established a Certificate Authority (CA) which would sign both server and client certificates. After that it generated the certificates and keys for the server, Diffie-Hellman parameters for encryption and an HMAC to add an increased layer of security. These elements made make sure that there was a secure method of authentication or the encryption of the VPN.

In the server configuration file (server.conf), the port (udp) was set as the protocol, the encryption system (AES-256-CBC) and the SHA256 authentication was used and the DNS forwarding for the VPN clients. Network forwarding was set to 1 for traffic routing and firewall rules were modified with UFW to add VPN (port 1194/UDP) and SSH. Finally the OpenVPN service was started and meant to start at boot time.

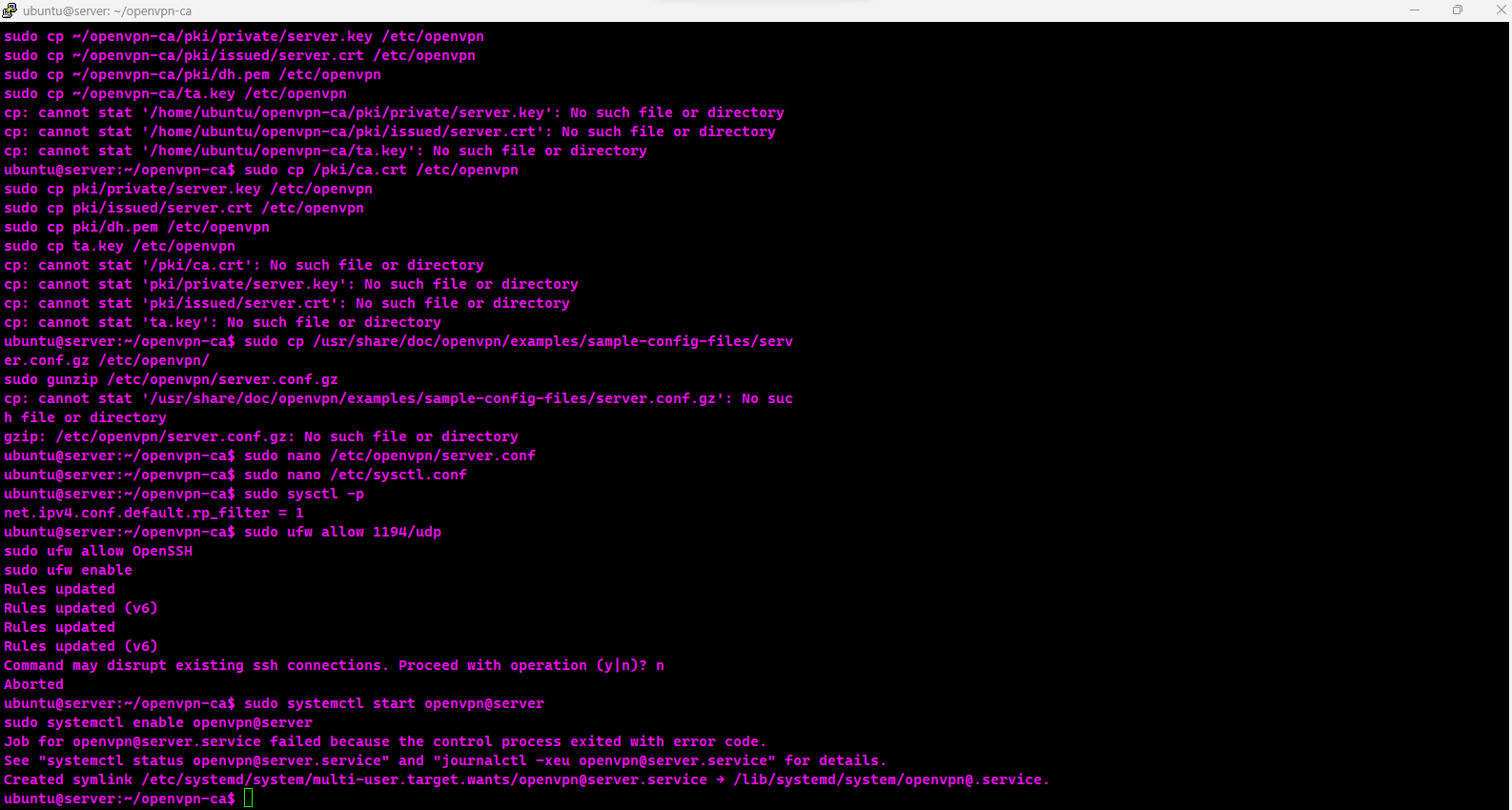
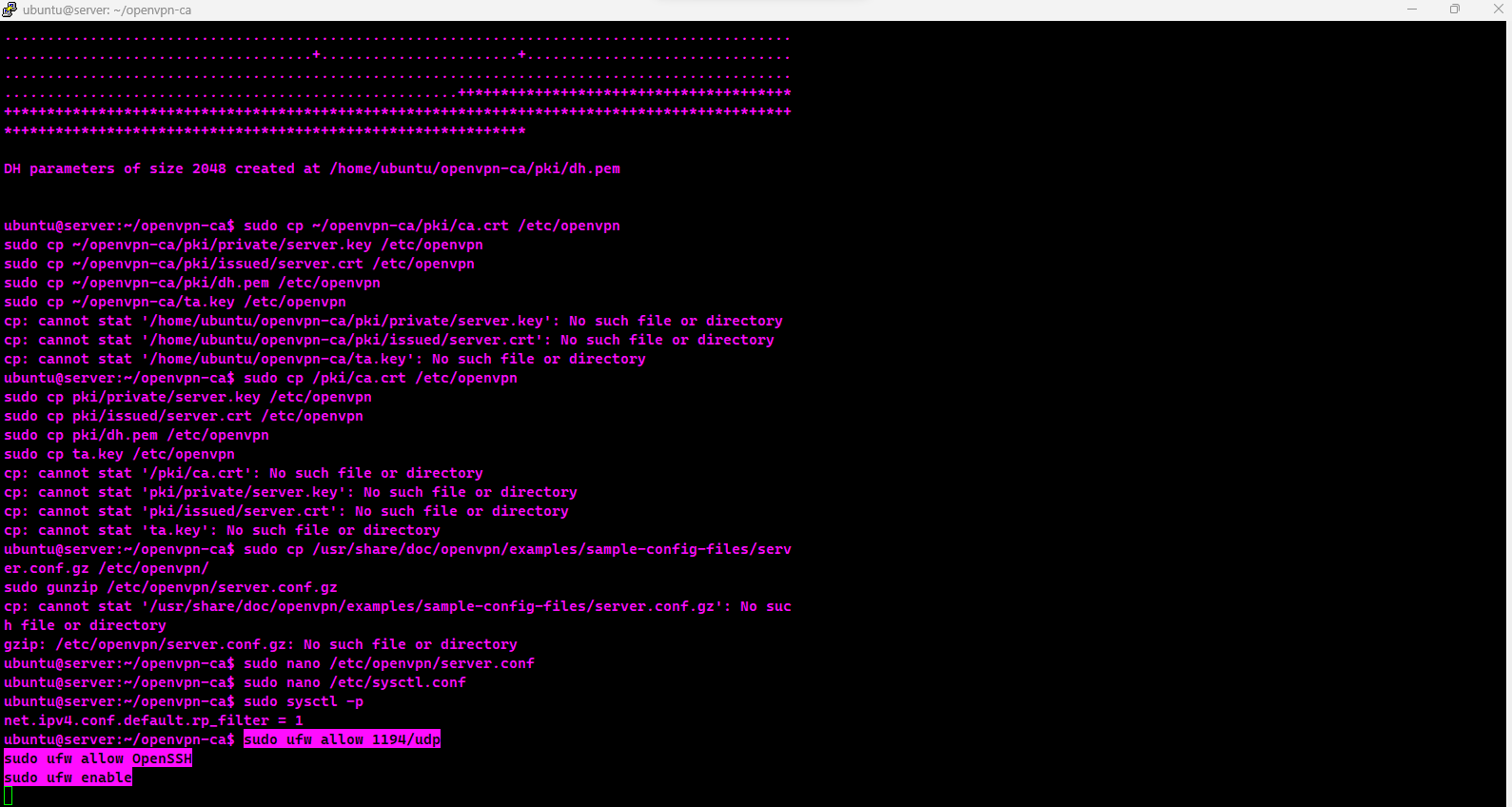
***Client Access Setup***

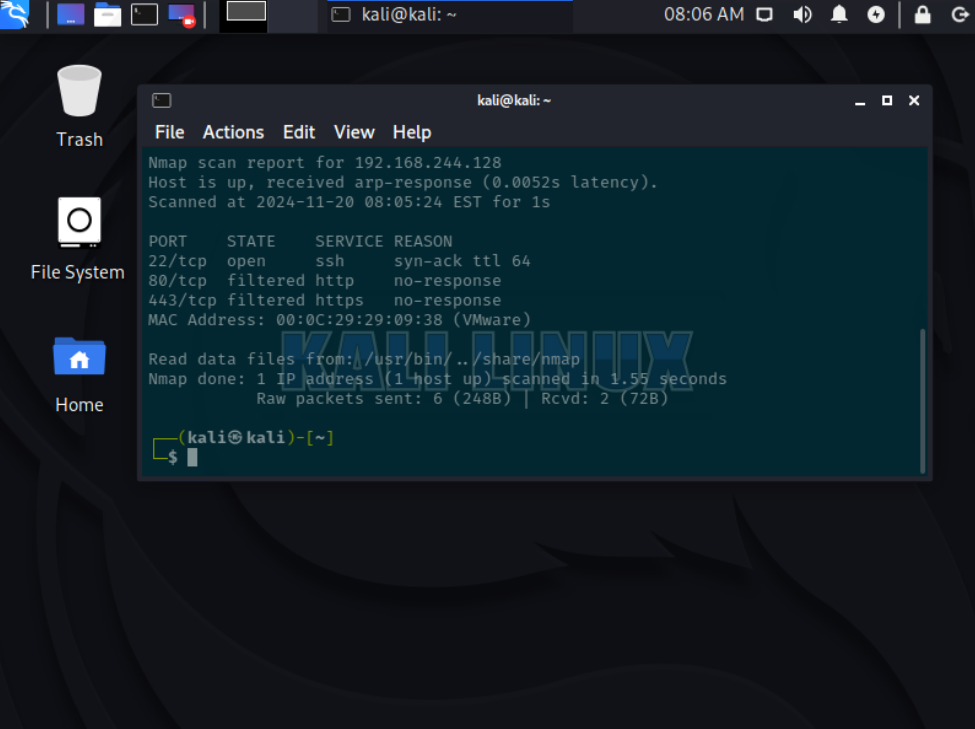
Client configuration was to create .ovpn files that include the appropriate certificate and key, as well as the connection profile. As for the other files, these files are to be copied onto the clients’ computers for their easy use in program design. It also needed settings capable of providing encryption, secure keys retention, as well as server connection parameters.

***Testing and Verification***

The employed VPN was checked for its functionality and its protection capability. For OpenVPN client configuration from a client device installation was conducted and then use the configuration file in .ovpn format to connect. Finally, the connection was successful by browsing the file server and other private web sites within the network.

Encryption was confirmed by auditing the client logs and it revealed that secure connection was initiated properly. Furthermore, tool for monitoring the traffic (tcpdump) was used to ensure that all the transmitted data was encrypted so that it could not be intercepted by a third party.



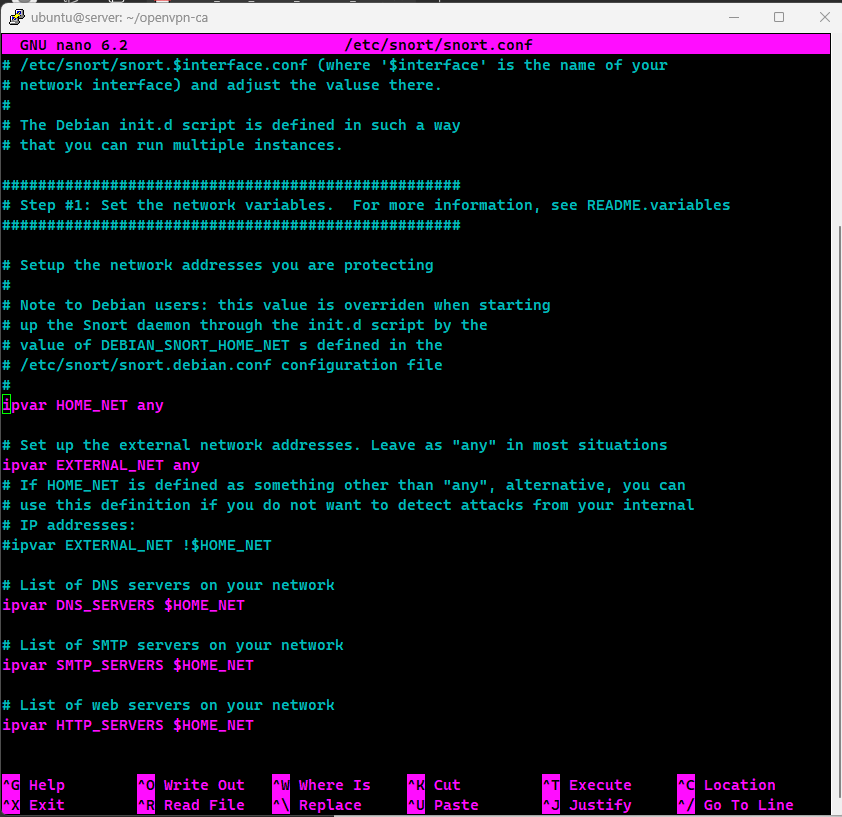
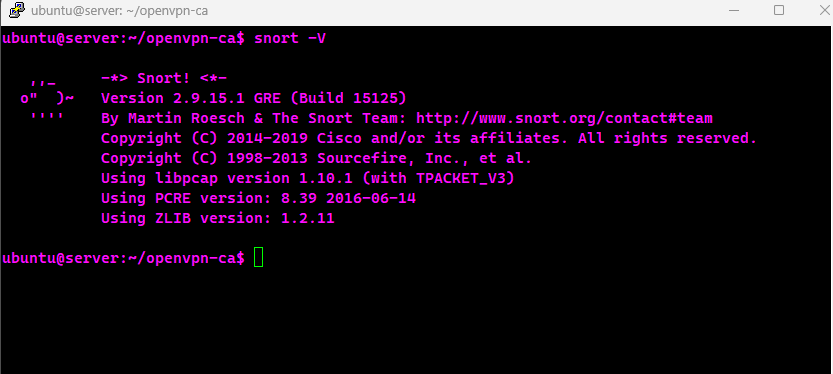
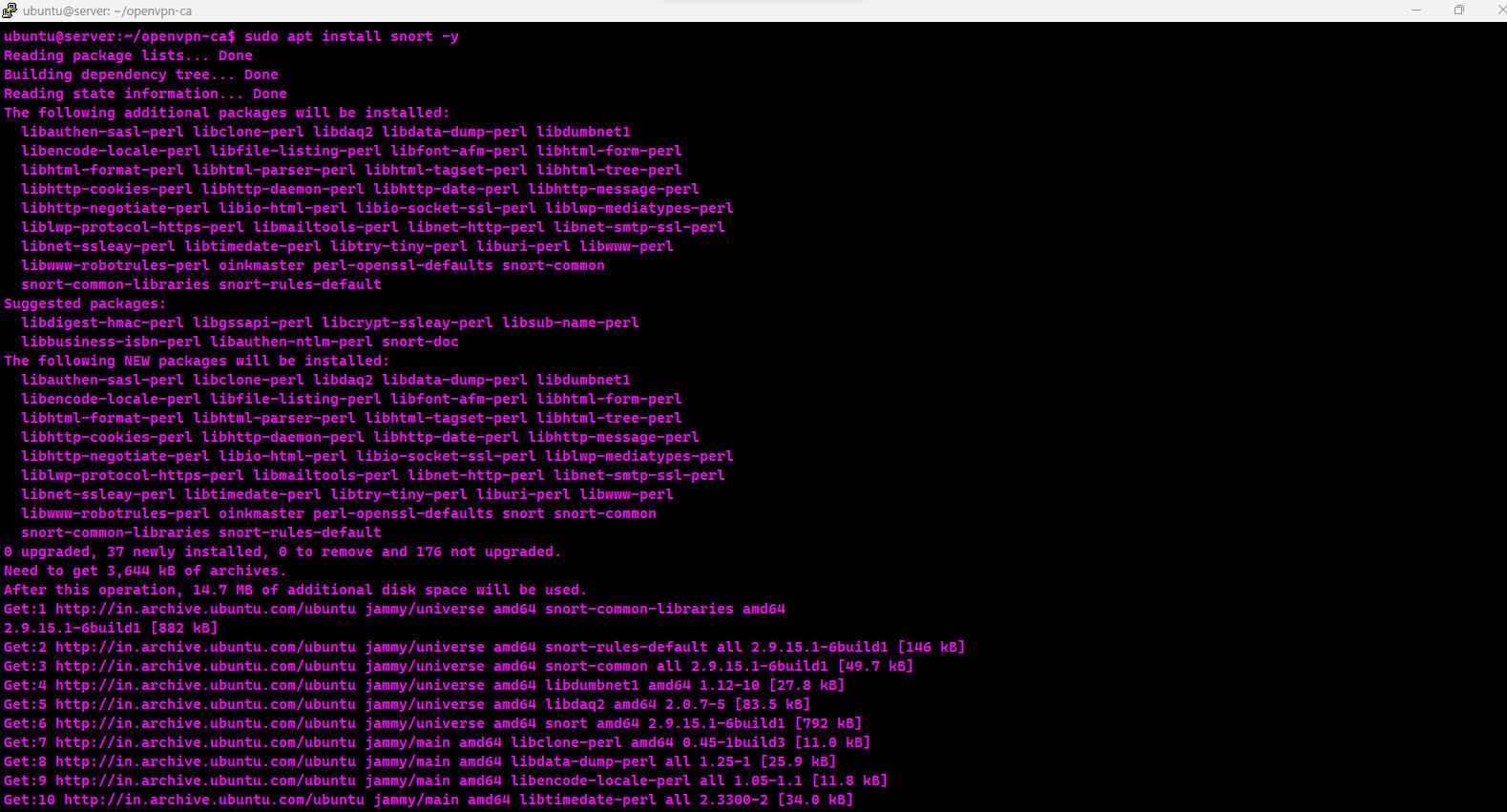


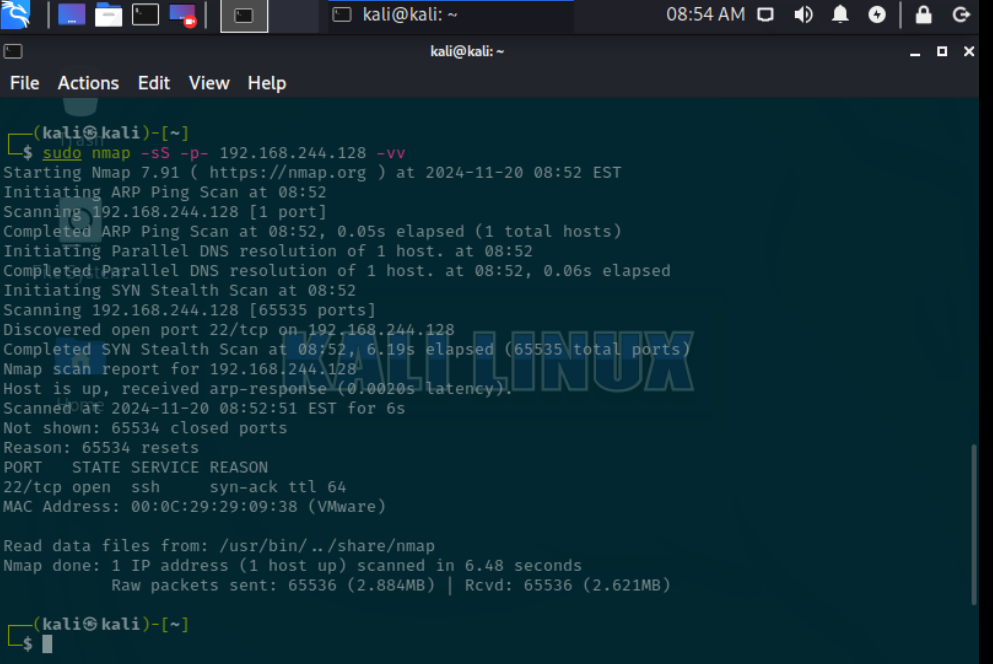
**IDS/IPS Implementation**

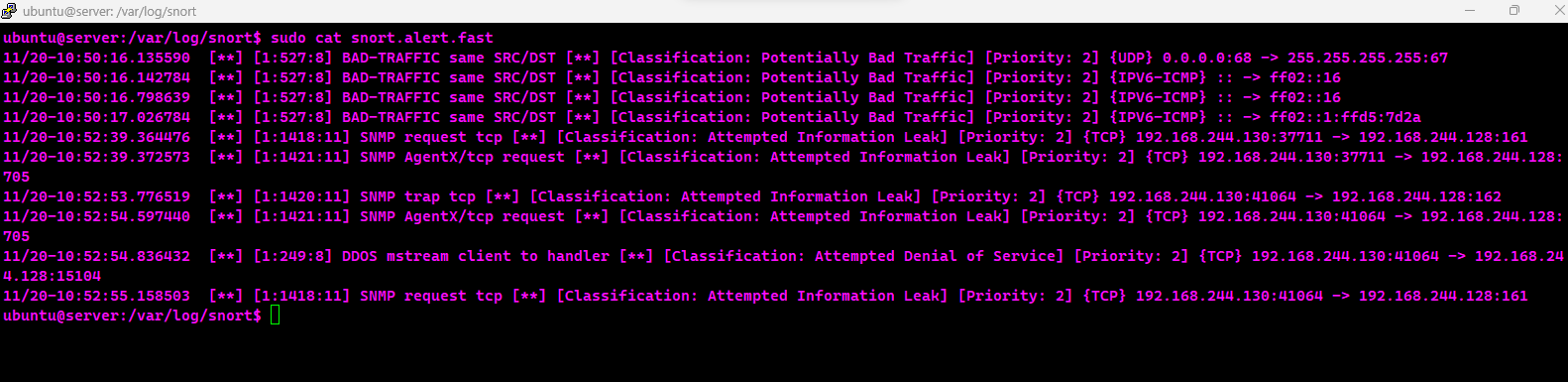
The IDS/IPS implemented is Snort, we download snort to the Ubuntu server and also installed the default rules and set up custom rule**:**

*alert tcp $EXTERNAL\_NET any -> $HOME\_NET 22 (msg:"SSH Brute Force"; flags:S; threshold:type both, track by\_dst, count 5, seconds 60; sid:1000001;)*

To test the IDS, we triggered attack traffic from the attacking machine, in this case, Kali Linux, and followed up with the Snort log to identify whether it was able to detect any rule violation, which it actually detected.







**Testing and Security Assessment**

To contact the implemented security in the network we simulated attacks from the network default gateway using both nmap and metasploit framework.The namp scan showed negative result with all ports filtered by the firewall and but SSH port was luckcly exposed which also was difficult to exploit through bruteforce and other attack vector due to strong ssh cyphers / encryption.We tried to utilize ssh exploit to obtain a reverse shell but it also didn’t work.This simplified well implemented security but thesome security misconfiguration can lead to compromise like exposing ssh to the internet.

The penetration testing of attempting to open the OpenSSH service by Metasploit was futile in the attempt. Although the service discovered was in port 22, the exploitation did not give the attacker a session. This could be due to several factors, including:

* A more secure version of OpenSSH, for instance use of more secure authentication methods (such as keys instead of passwords).
* The system likely having settings as fail2ban, rate-limiting or firewall that prevents brute force/ exploiting.
* The particular Metasploit exploit used may not have been suitable for the particular SSH version in operation on the target system

**Mitigation**

Configuration optimization for security misconfiguration

