SURICATA IDS/IPS IMPLEMENTATION FOR NETWORK THREAT DETECTION

Practical Implementation & Indonesian Legal Analysis

Author: Reihan Daniswara Pramudito

Date: December 2024

KEY FINDINGS

- **Intrusion Detection & Prevention**:
  - **SSH Anomaly Detection**: Suricata flagged SSH-on-non-standard-port attempts
  - **HTTP Protocol Alerts**: Suricata detected all malformed HTTP requests
- **Active Blocking & Firewall Integration**:
  - **Drop Actions Enabled**: IPS mode dropped all flagged SSH/HTTP packets
  - **NFQUEUE Bypass**: Maintained uninterrupted SSH access while scanning traffics
- **Web Server Hardening**:
  - **SSL/TLS Deployment**: Generated and installed self-signed certificates with OpenSSL

TOOLS USED

- **Network Security**: Suricata
- **Web Hardening**: OpenSSL, Apache HTTPD, firewall-cmd

Disclaimer:

This report was created for educational purposes. Findings are based on a simulated network scenario on Rocky Linux. Tools, methodologies, and conclusions are intended for learning and do not represent an actual organizational environment.

Table of Contents

1.0 Network Configuration

1.1 Setting up Hostname on Server and Client using FQDN

1.1.1 Rocky Linux (Server)

Step 1: Open the terminal and type sudo nano /etc/hostname

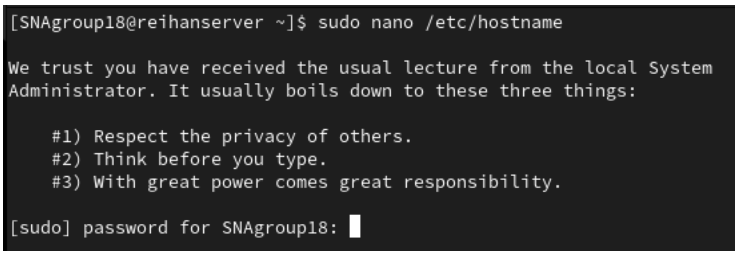


Figure 1.1.1: Hostname

Step 2: Type the server's name that is being used into the sudo nano /etc/hostname, as you can see our group number is 18, therefore we have named the server as server.snagroup18.com

- This step is configuring our server's name to the Fully Qualified Domain Name (FQDN) format
- Only whoever with the root admin privileges can make the changes for the server's name.

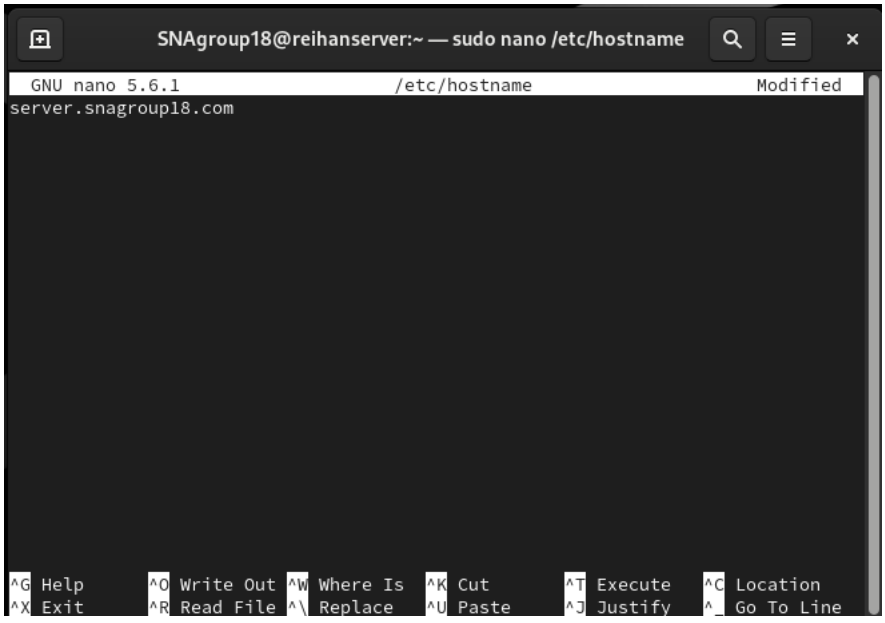


Figure 1.1.2: Server Name

Step 3: Reboot your system

- By restarting the system, we can make sure the configuration works properly

Figure 1.1.3: Reboot

Step 4: Now once your system has rebooted, type in hostname and enter, and as you can see it shows our server's name which is server.snagroup18.com. This means it has we have successfully configured using the FQDN format

Figure 1.1.4: Hostname

1.1.2 Ubuntu Linux (Client)

Step 1: Now we are going to repeat the same step we took for rocky. Open the terminal and type sudo nano /etc/hostname and enter your password to enter.

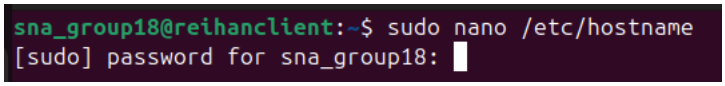


Figure 1.1.5: Hostname

Step 2: Once entered, type in the server's name but this time change the server to "client". The purpose is to understand a different between rocky and ubuntu so that we don't get confused.

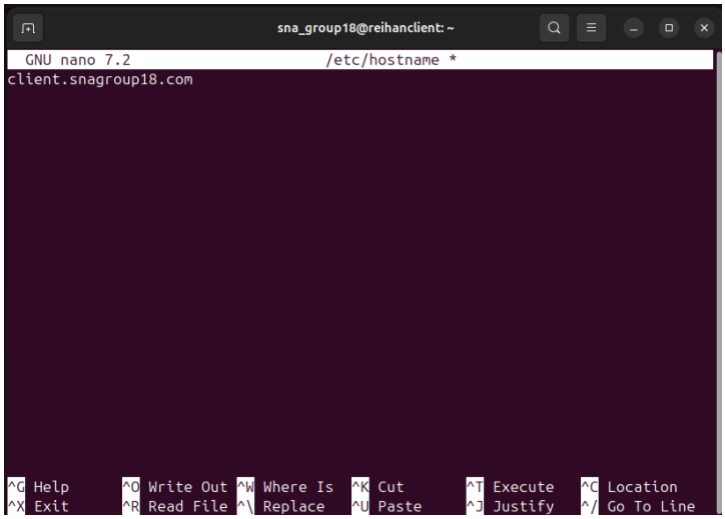


Figure 1.1.6: Client Name

Step 2: Reboot the system again, and check the configuration whether it works properly.

```
sna_group18@reihanclient:~$ reboot
```

Figure 1.1.7: Reboot

Step 3: Once rebooted, type in hostname and check once again whether the configuration has updated. If is has it will show the client name such as the picture below.

```
sna_group18@client:~/Desktop$ hostname
client.snagroup18.com
```

Figure 1.1.8: Hostname

1.2 Setting up Static IP Address on the Rocky Linux (Server)

Step 1: Type in sudo rpm -q NetworkManager

- This is to check whether the NetworkManager has been install into our server
- The NetworkManager is installed by default by Rocky Linux.

```
[SNAgroup18@server ~]$ sudo rpm -q NetworkManager
[sudo] password for SNAgroup18:
NetworkManager-1.46.0-19.el9_4.x86_64
```

Figure 1.2.1: Check NetworkManager

Step 2: Type sudo systemctl enable NetworkManager and sudo systemctl start NetworkManager

- The point of this is to enable the NetworkManager to boot up
- This will remind the system to tell to start the NetworkManager service whenever the system starts

```
[SNAgroup18@server ~]$ sudo systemctl enable NetworkManager
[SNAgroup18@server ~]$ sudo systemctl start NetworkManager
```

Figure 1.2.2: Enable NetworkManager

Step 3: Type in sudo systemctl status NetworkManager

- This command sudo systecmtl status NetworkManager is used to check the status of the NetworkManager service on a Linux System.
- Running this command will also show whether the NetworkManager service is active (running), inactive, or failed.
- If there are any sort of errors or warning in the logs associated with the service, it will be showed by running the sudo systemctl status NetworkManager

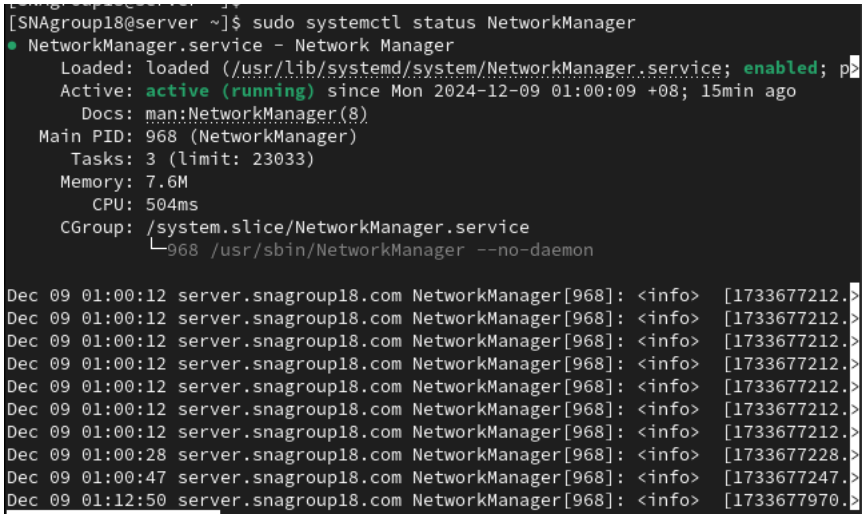


Figure 1.2.3: NetworkManager Status

2.0 Network File System (NFS) Configuration

Step 1: Open your rocky terminal and type in sudo dnf install nfs-utils.

- This is to configure and manage NFS servers (systems that share directions over the network)
- This command is also enable and use NFS protocol versions, for an example NFSv3, or even NFSv4

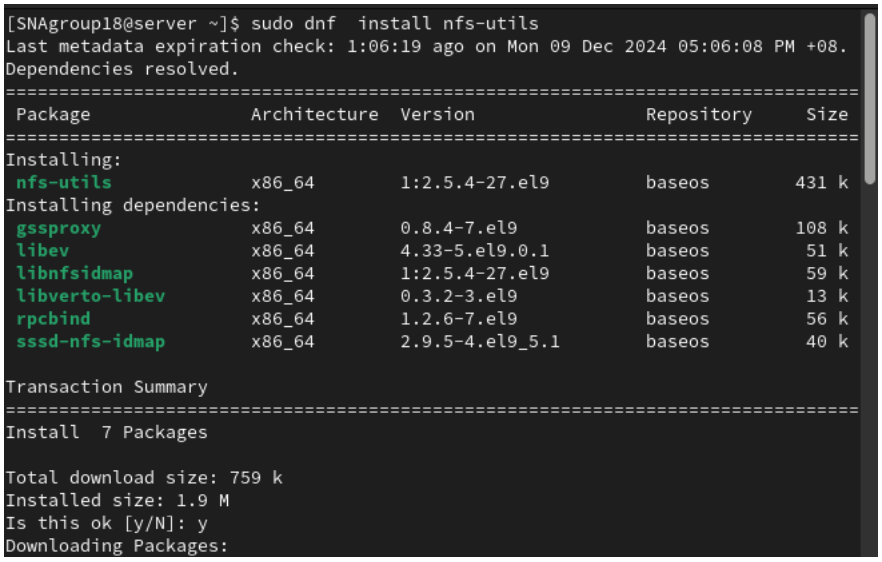


Figure 1.2.3: Install NFS

Step 2: Tye sudo systemctl enable nfs-server –now

- This command is used when setting up an NFS server.
- The NFS server is running and ready to server shared directions immediately
- The service will also start automatically on subsequent reboots.

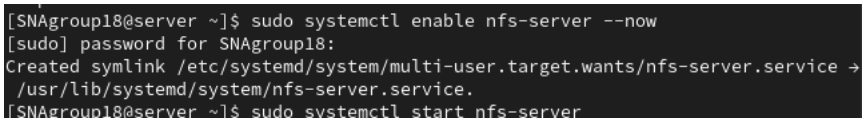


Figure 1.2.4: Enable NFS

Step 3: Type sudo systemctl status nfs-server.

- When we run this command, we can see the detailed information about the NFS server service including the status, service, main process ID and logs
- This will also help us to see whether the server is running active (running), inactive, or failed.
- We can also see how long the service has been running as well as how the recent entries related to the service such as any errors or warnings.

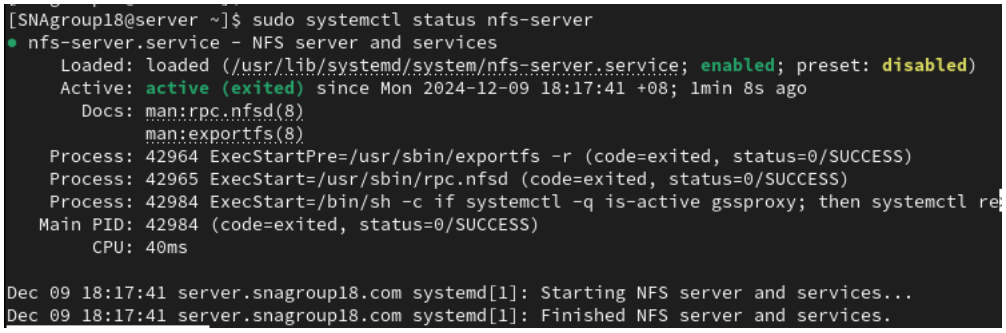


Figure 1.2.5: NFS Status

Step 4: Type in sudo cat /proc/fs/mfsd/versions

- This command to the check the NFS versions supported and enabled on the system. This file provides details about the configuration of the NFS server regarding protocol versions.

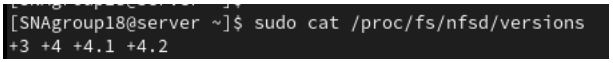


Figure 1.2.6: Check NFS Version

Step 5: Type in sudo mkdir /var/nfs/general -p

- This command creates a directory for sharing files over the Network File systems.
- This directory will likely be used as as a shared location for NFS clients. It's common to create directories under /var for server-related services.

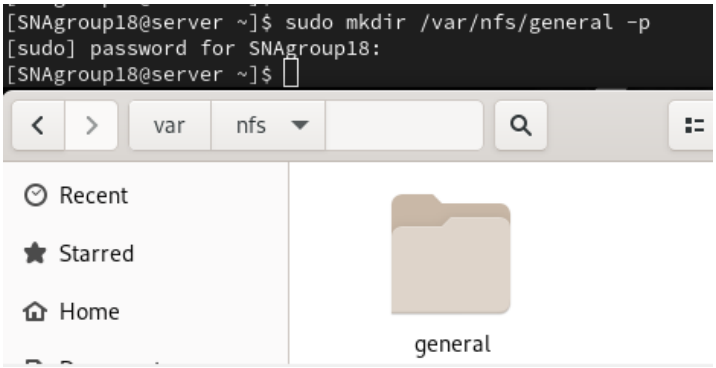


Figure 1.2.7: Creates Directory

Step 6: Type sudo ls -dl /var/nfs/general/

- This command is used to inspect detailed information about the directory that we created earlier
- It also displays the ownership, permissions and other attributes of the directory.

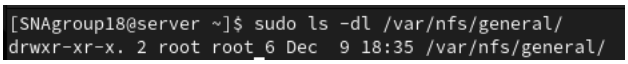


Figure 1.2.8: Check Ownership

Step 7: Type sudo chown nobody:nobody /var/nfs/general/

- This command is used to change the ownership of the directory to the nobody user and group
- This is often done as a part of setting up an nfs share to ensure proper permissions for clients accessing the shared directory.

Figure 1.2.9: Change Ownership

Step 8: Type in sudo nano /etc/exports

- This command opens the file for editing suing the nano text editor, with administrative privileges. This file is used to configure and manage for NFS (Network File System) shares that are made available to clients.

Figure 1.3.0: Open File

Step 9: Type in the the path of the directory which us /var/nfs/general and followed by the ip address

- Once done click the ctrl x and Y to save the input that we have put into the etc/exports

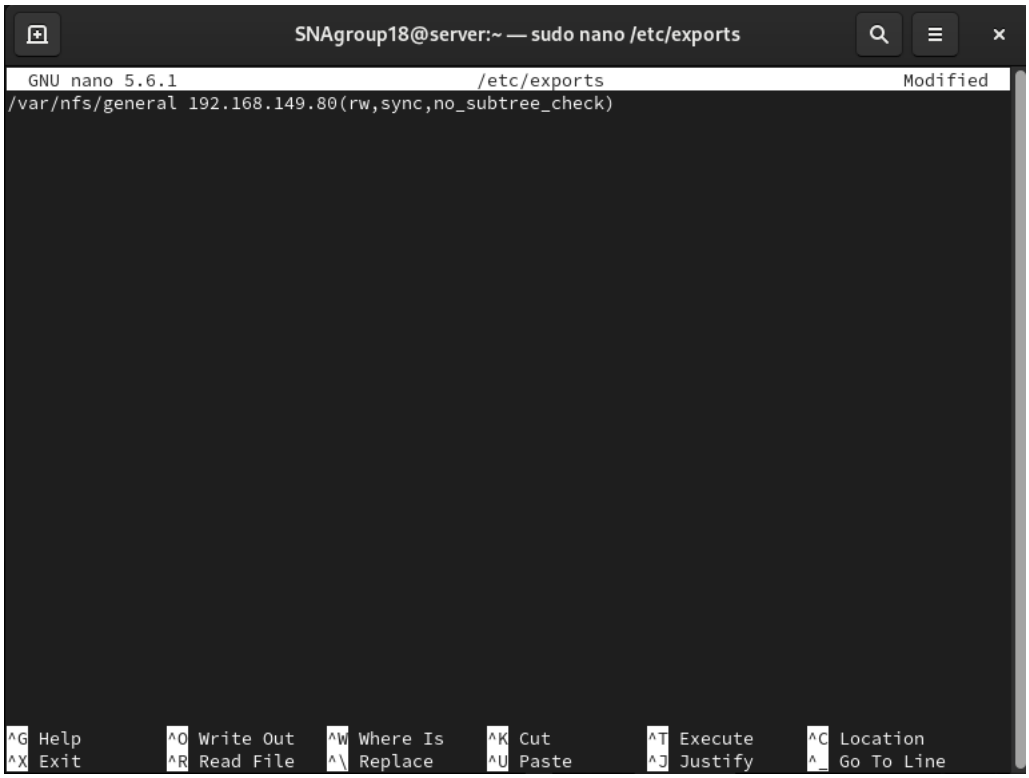


Figure 1.3.1: Path Directory

Step 10: Type in sudo exportfs -arv

- The command is used to refresh and display the currents NFS shares that are exported on the server.
- It is a command for managing NFS exports, typically after making changes to the /etc/exports file

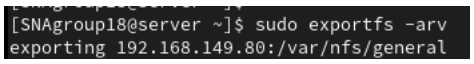


Figure 1.3.2: Display NFS

Step 11: Type in sudo exportfs -s

- This is used to display a summary of all the NFS exports on the server, including the exported directories, their permissions, and the client systems that have access to them.
- It also shows which directories are exported but without the detailed verbose output that includes every option.

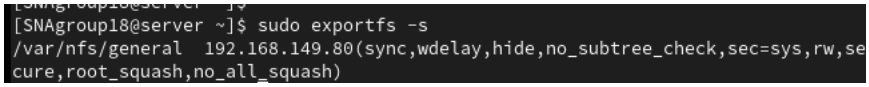


Figure 1.3.3: Display Summary

Step 12: Type sudo systemctl enable –now rpcbind ngs-server and sudo systemctl restart nfs-server

- The command which enable helps to starts the rpcbind and nfs-server services immediately.
- This command will ensure both the rpcbind and nfs-services start immediately and are enabled to start at boot
- As for the system restart, the command will restart the NFS server without rebooting the system

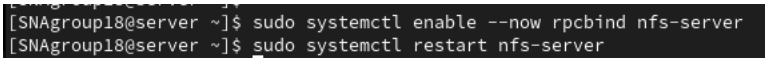


Figure 1.3.4: Enable rpcbind

Step 13: Type sudo systemctl status nfs-server

- This command is to used to check the status of nfs servive
- It also provides detailed information about the currents state of the NFS service including whether it is active (running), inactive (stopped) or in a failed state

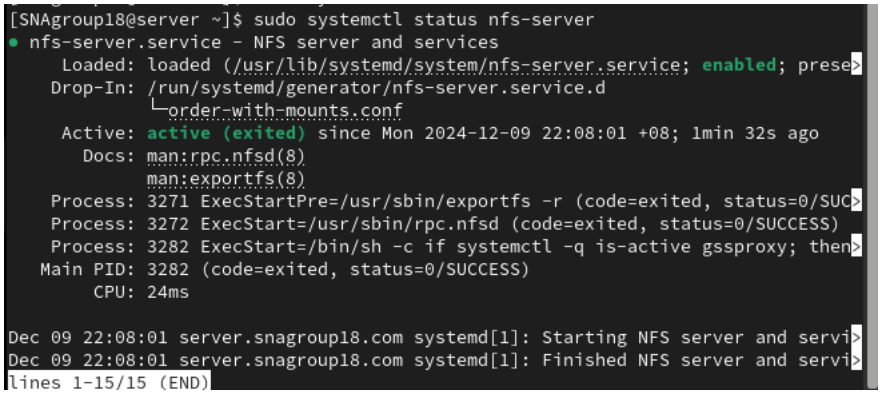


Figure 1.3.5: NFS Status

Step 14: Type sudo firewall-cmd –add-service={nfs,nfs3,rpc-bind,mountd} –permanent

sudo direwall-cmd –reload

sudo firewall-cmd –zone=public –list-all

- The first command adds the necessary firewall rules to allow NFS-related traffic through the firewall. Since NFS works over multiple ports and services, opening these services ensures that NFS shares are accessible to clients
- The second command is to ensure that new rules or modifications take effect immediately
- The third command is to used to display all the firewall rules and settings for the public zone, this shows which services, ports, and other rules are allowed in the public zone.

Figure 1.3.6: Firewall Rules NFS

Step 15: Type in sudo systemctl daemon-reload

- This command is used to reload the system manager configuration, this command is typically used after making changes to unit files or after installing new packages that includes unit files.

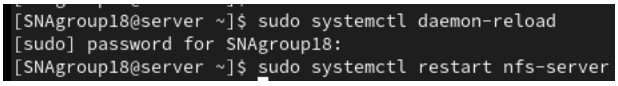


Figure 1.3.7: Daemon Reload

Step 16: Type sudo systemctl status nfs-server

- This command is used to check the status of the nfs server service on a linux system
- It also provides detailed information about the nfs server such whether it is active (running), inactive failed.

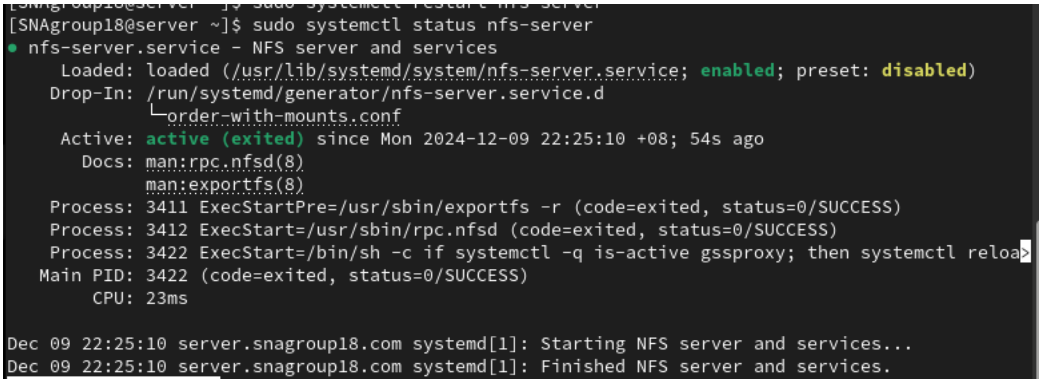


Figure 1.3.8: NFS Status

Step 17: Type in cd /var/nfs/general

- This command is used to change the current directory to the /var/nfs/general in the filesystem

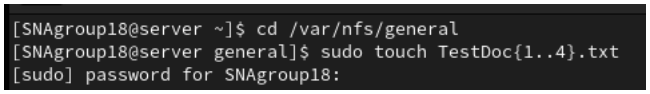


Figure 1.3.9: Change current directory

Step 18: Type ls -l

- This command is used to list files and directories in the current directory with all the detailed information

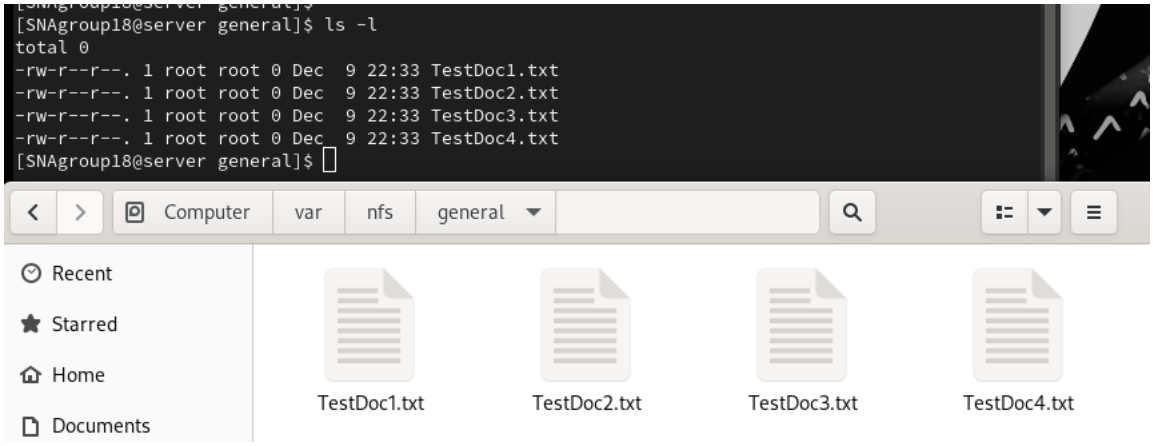


Figure 1.4.0: List Files

Step 19: Now open ubuntu and type in sudo apt install nfs-kernel-server

- This command is used to install the NFS server on a system that uses APT for package management, such as Debian-based distribution.
- This command also installs the NFS server package, which provide the necessary tools and service for setting up an NFS server
- After installation, the system can share directories with client over the network using the NFS protocol

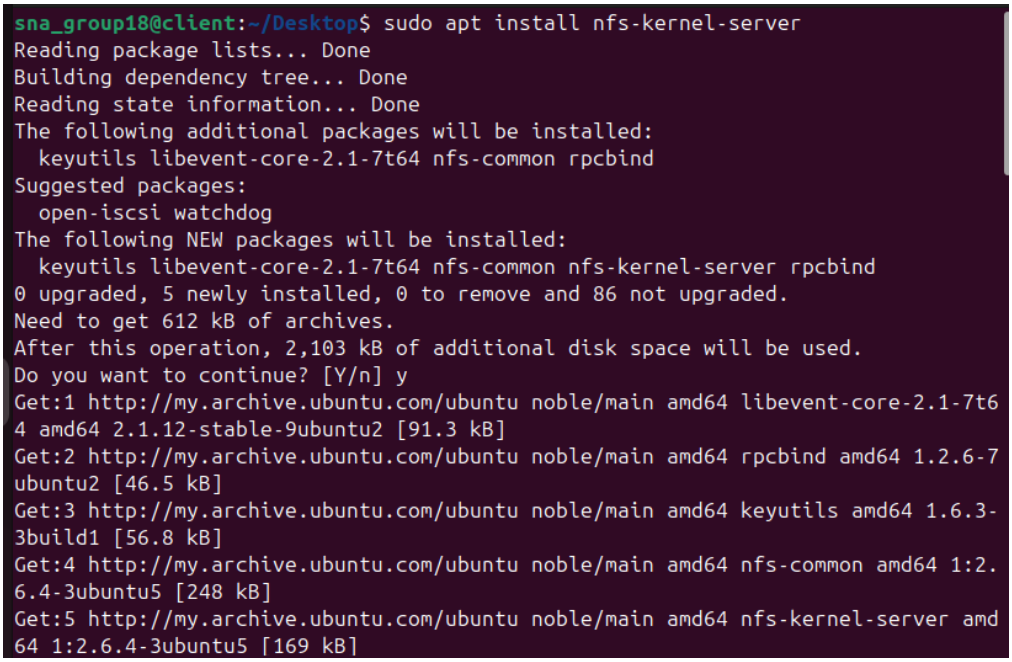


Figure 1.4.1: Install NFS

Step 20: Type sudo apt install nfs-common

- This command is used to install the NFS client utilities. These utilities are necessary for mouthing and accessing shared directories from an NFS server

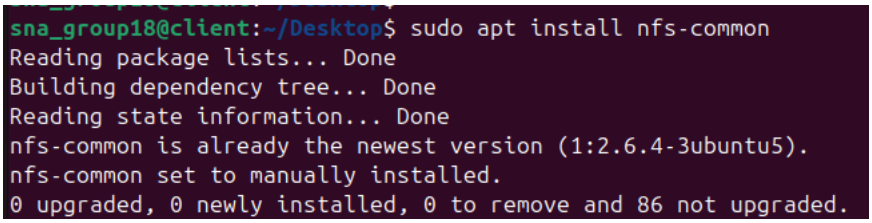


Figure 1.4.2: Install NFS Common

Step 21: Type sudo mkdir /nfs/shared_from_server -p

- This command is used to create a directory on your system.

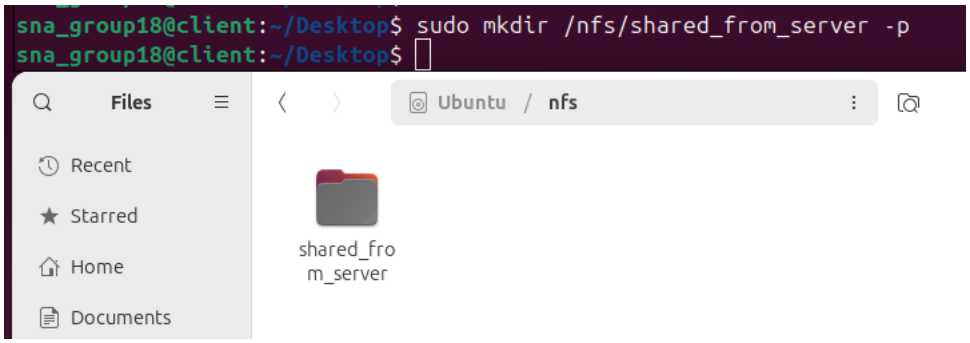


Figure 1.4.3: Create Directory

Step 22: Type sudo mount 192.168.149.4:/var/nfs/general /nfs/shared_from_server

- This command is used to mount and NFS share from a remote NFS server onto the local system
- As for the command df -h, this used to display disk space usage on the mounted files system the system in a human-readable format

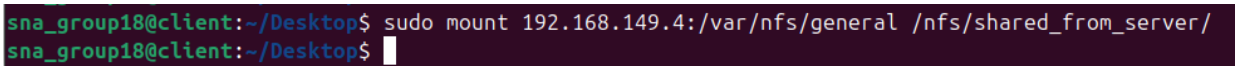


Figure 1.4.4: Mount NFS

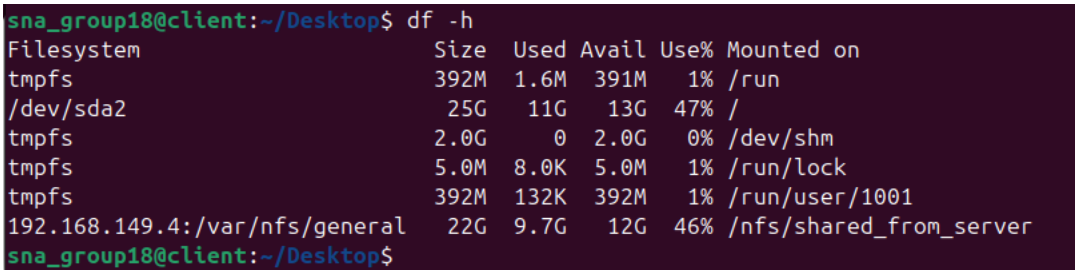


Figure 1.4.5: Display Disk Space

Step 23: Type ls -l /nfs/shared_from-server/

- This command is used to list the contents of the directory in longlisting format

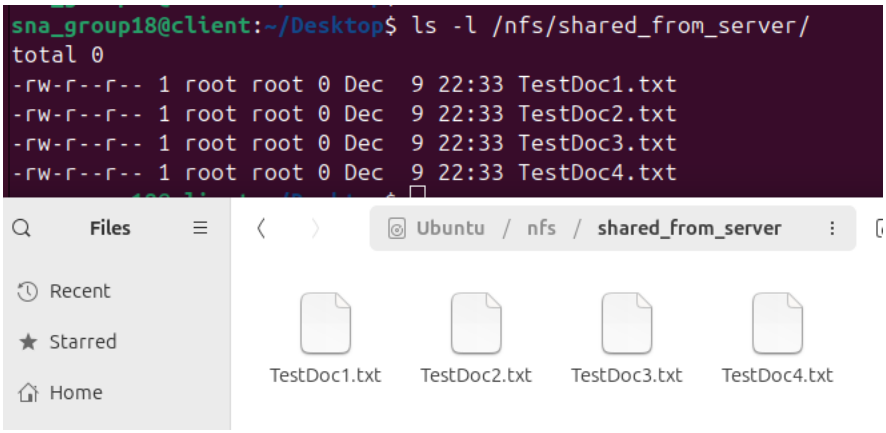


Figure 1.4.6: List Content

Step 24: Type sudo touch /nfs/shared_from_server/sent_from_clientUbuntu.txt

- This command is used to create an empty file in the directory

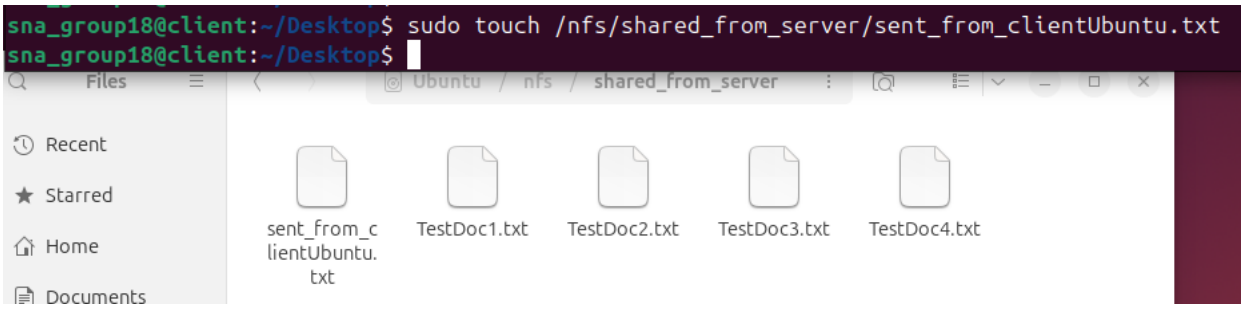


Figure 1.4.7: Create File

Step 25: Now boot up rocky and open the terminal, type ls -l /var/nfs/general/

- This command is used to list the contents of the directory in long lisitng format
- It also shows the file permissions, sizes, owners, and modification times

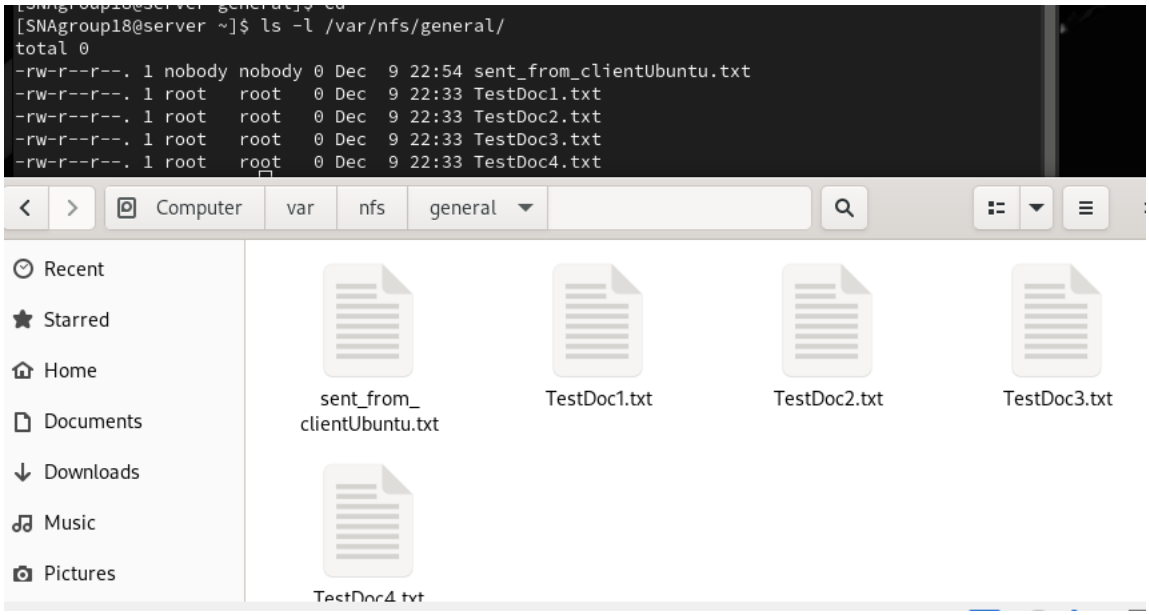


Figure 1.4.8 List Content

Step 26: Type cd /var/nfs/general/ and ls -l

- As you can see, it will list all the available long list format as shown below
- Now for the sudo mkdir, is to create a new directory in the working directory
- And back to the ls -l, it starts to list down all available and created files.

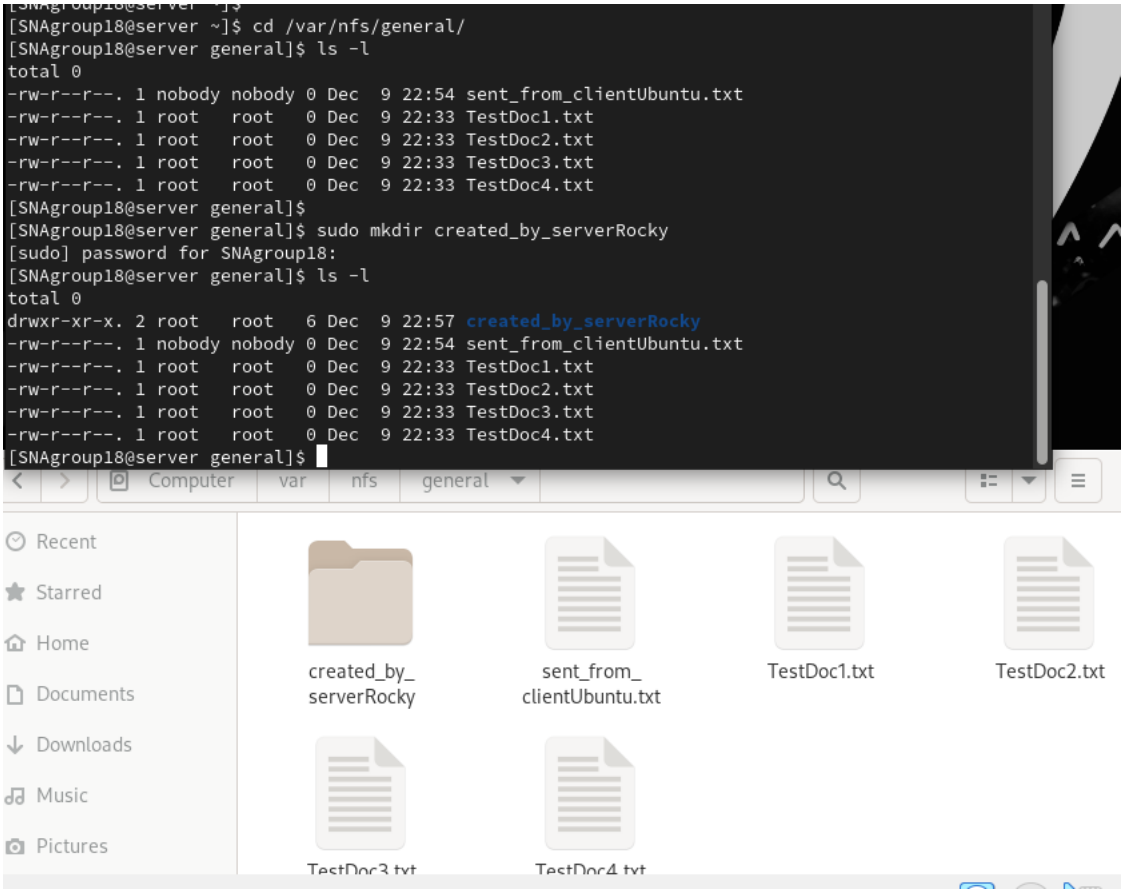


Figure 1.4.9: List File

Step 27: Open the terminal from ubuntu and type is ls -l /nfs/shared_from_server/

- As you can see, it lists down the long listing format and we also have that there is a file created by the server and the testing work well

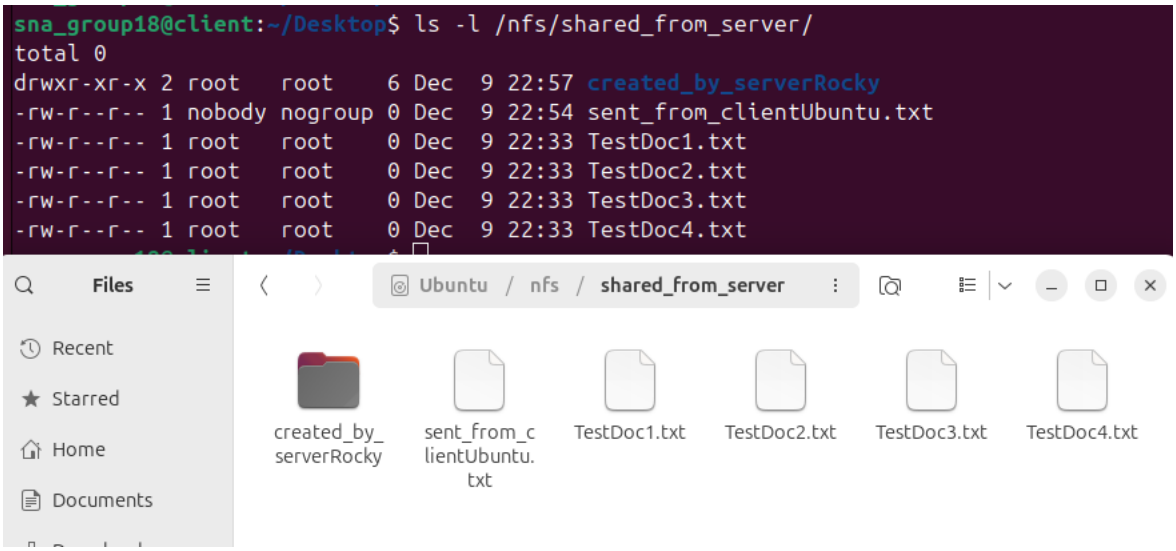
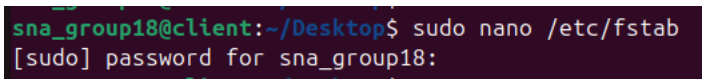


Figure 1.5.0: Files Created

Step 28: Type sudo nano /etc/fstab

- After entering your password, this command will direct you to edit the file using the nano text editor with the superuser root privileges.



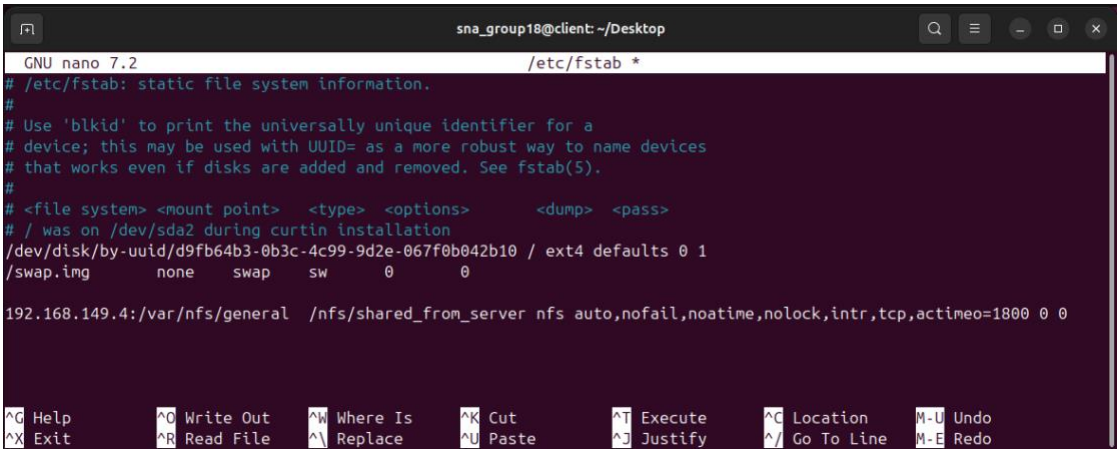


Figure 1.5.1: Editing File

Step 29: Type sudo mount -a

- This will mount all files systems listed in the file that are not already mounted, it is typically used to apply changes made to the file without needing to reboot the system

Figure 1.5.2: Mount

Step 30: Reboot your system

Figure 1.5.3: Reboot

Step 31: Command ls -l /nfs/shared_from_server/ to check if the configuration works

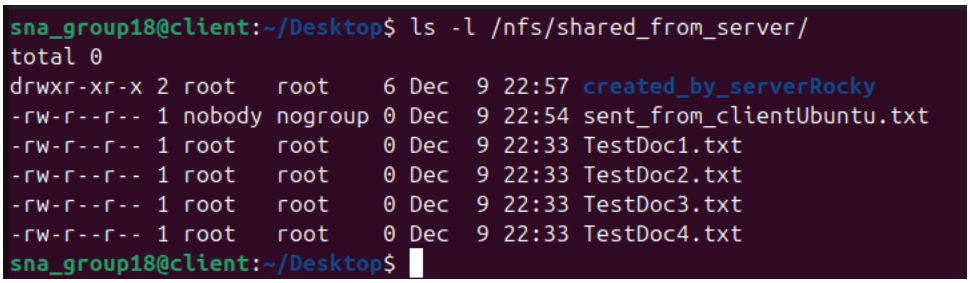


Figure 1.5.4: Check Configuration

3.0 Apache Web Server

3.1 Configure Apache Web Server

Step 1: First, we need to install the httpd package in our server using “sudo dnf install https”.

```
[SNAgroup18@server ~]$ sudo dnf install httpd
[sudo] password for SNAgroup18:
Last metadata expiration check: 0:02:09 ago on Tue 10 Dec 2024 01:16:14 PM +08.
Dependencies resolved.
================================================================================
 Package               Arch        Version               Repository      Size
================================================================================
Installing:
 httpd                 x86_64      2.4.62-1.el9          appstream       45 k
Installing dependencies:
 apr                   x86_64      1.7.0-12.el9_3        appstream      122 k
 apr-util              x86_64      1.6.1-23.el9          appstream       94 k
 apr-util-bdb          x86_64      1.6.1-23.el9          appstream       12 k
 httpd-core            x86_64      2.4.62-1.el9          appstream      1.4 M
 httpd-filesystem      noarch      2.4.62-1.el9          appstream       12 k
 httpd-tools           x86_64      2.4.62-1.el9          appstream       79 k
 rocky-logos-httpd     noarch      90.15-2.el9           appstream       24 k
Installing weak dependencies:
 apr-util-openssl      x86_64      1.6.1-23.el9          appstream       14 k
 mod_http2             x86_64      2.0.26-2.el9_4.1      appstream      163 k
 mod_lua               x86_64      2.4.62-1.el9          appstream       58 k

Transaction Summary
================================================================================
```

Figure 3.1.1: Installing Httpd Package

Step 2: Next we need to rename the welcome.conf file to welcome.conf.org and move it to a backup directory using the 'mv' command.

- The .org extension signifies that it is an original backup, allowing for easy identification later. This ensures the default configuration remains preserved as a reference while you work on your new website.

```
[SNAgroup18@server ~]$ sudo mv /etc/httpd/conf.d/welcome.conf /etc/httpd/conf.d/welcome.conf.org
```

Figure 3.1.2: Rename welcome.conf file

Step 3: Next, we need to open the main configuration file for Apache http by using the “sudo mv/etc/httpd/conf.d/welcome.conf /etc/httpd/conf.d/welcome.conf.org.

- Make sure you are in the “httpd.conf” path

```
[SNAgroup18@server ~]$ sudo vi /etc/httpd/conf/httpd.conf
```

Figure 3.1.3: Opening Configuration File

Step 4: Inside the main configuration file, you will need to do some configuration.

- First, you will need to scroll down until you see **ServerAdmin,** set the server admin by adding the email address which is root@snagroup18.com
- Next, scroll down to the **ServerName**. You will need to set the server name to server.snagroup18.com:80. This is for our server to identify by the email address and operates on port 80, which is the standard port for HTTP traffic.

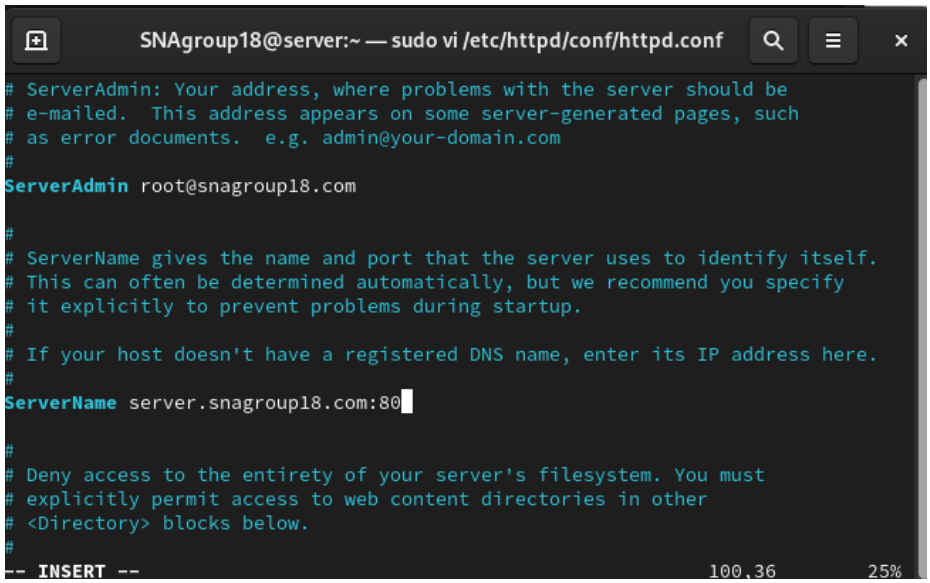


Figure 3.1.4: Editing ServerAdmin and ServerName

Step 5: Then, scroll down again to the option directive. Make sure that the indexes option is remove and only leave the FollowSymLinks. Next, we need to scroll down again to the AllowOverride setting and change from “none” to “ALL”.

- This enables the server to follow symbolic links, which function as references to other files or directories.
- This will also enables website settings to be managed through **.**htaccess files, removing the need to alter the primary server configuration files.

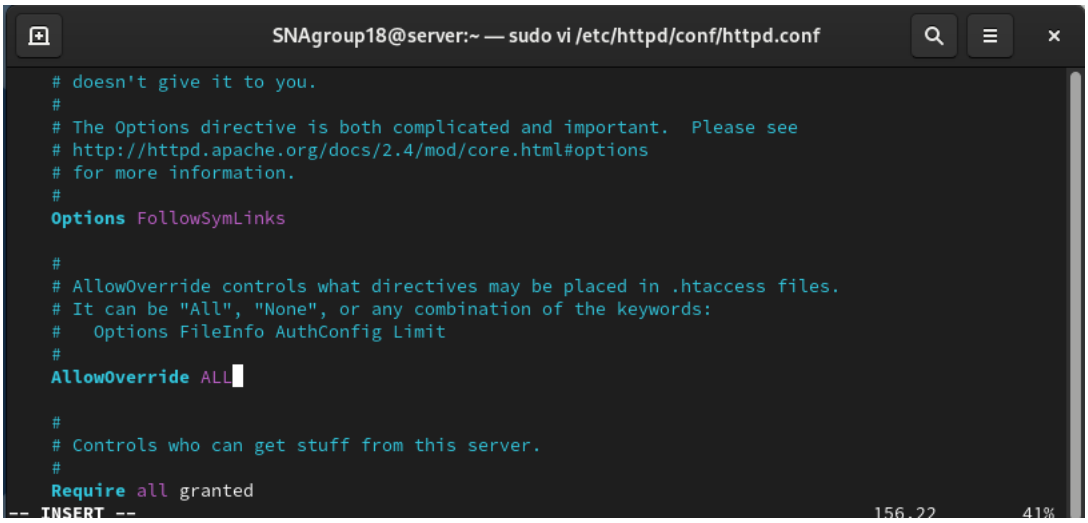


Figure 3.1.5: Editing Options Directory and AllowOverride

Step 6: Next, we need to scroll down until you see the **DirectoryIndex**. In the directory module, you will need to add index.php and index.cgi to the DirectoryIndex.

- This sets up Apache to search for these files in the defined sequence and serve the first one it finds when a directory is accessed without specifying a particular file.

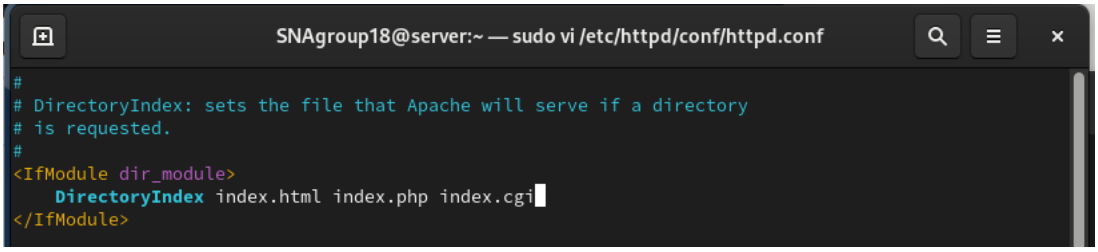


Figure 3.1.6: Editing DirectoryIndex

Step 7: Closed the main configuration file, and now you can the Http Apache service in our server. Use the “sudo systemctl enable –now httpd” command to enable the service.

Figure 3.1.7: Enable Httpd Service

Step 8: Lastly, we need to add our HTTP service to the firewall rules. We can make sure that the service is set to permanent so that whenever our server is reboot the service will still run.

- Use “sudo firewall-cmd –add-service=http” to add the http service to the firewall.
- Use “sudo firewall-cmd –runtime-to-permanent” to set the service to permanent.

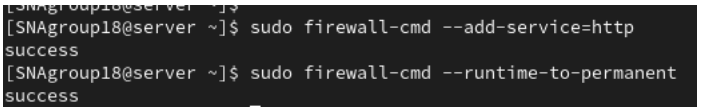


Figure 3.1.8: Adding Httpd service and Permanent Setup

3.2 Create a Website and Verify the Web Server

Next, we will be moving on to create a website. We will create a simple HTML file for the homepage of the web server.

Step 1: We need to open the index.html file in the html directory and edit to add the html coding.

- Use “sudo vi /var/www/html/index.html”

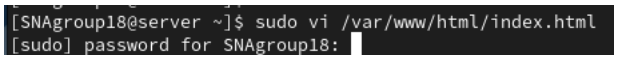


Figure 3.2.1: Opening index.html File

Step 2: In the html file, we will need to configure as per the HTML code below:

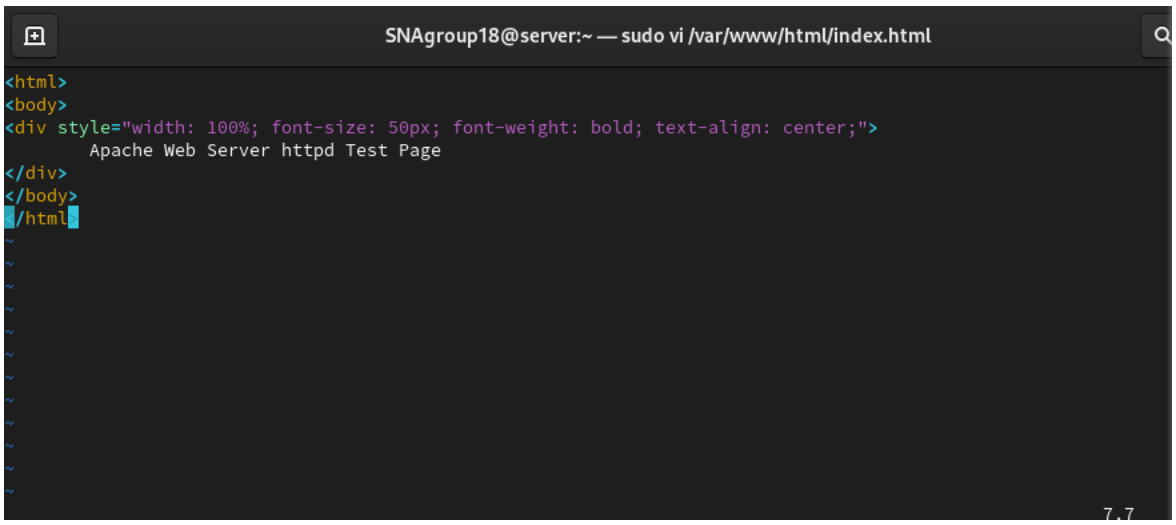


Figure 3.2.2: Editing Html Code

Code	Explanation
<html> <body>	This two code defines the structure of the homepage of the web server.
	The div elements uses inline CSS to style the text for the homepage of the web server.
Width: 100%;	The width makes the content span the full width of the homepage.
Font-size: 50px;	This code sets the font size in the homepage.
Font-weight: bold;	This code makes the text become bold.
Text-align: center;	This code centers the content or the text in the homepage.
Content	Displays any content in the homepage, for example, “Apache Web Server httpd Test Page”.

Next, we will need to verify that our web server is set up successfully. First, we will text the web server on our Rocky server.

Step 3: Open mozilla firefox, in the search bar type in the server name that we have configured before.

- For example, we used “snagroup18.com”

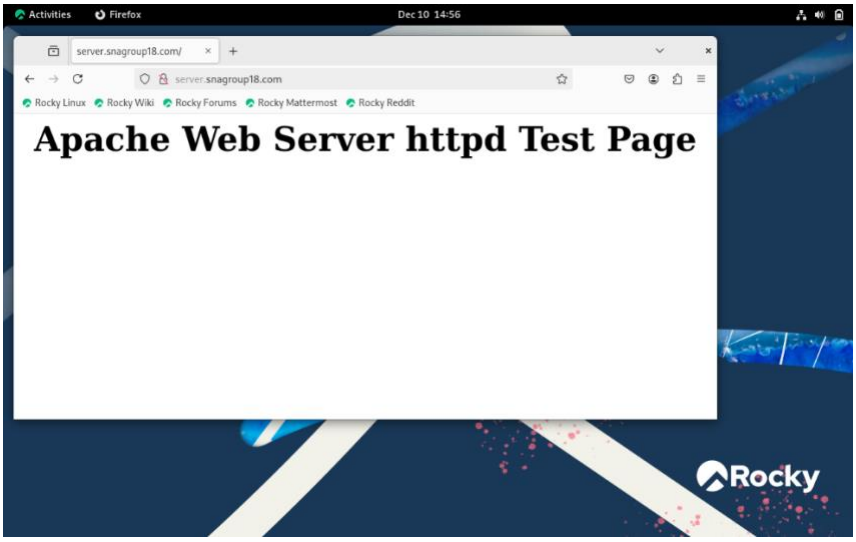


Figure 3.2.3: WebServer On Rocky

Next, open your Ubuntu Client. Do the same steps as in our Rocky server which search for the server name in the mozilla firefox. If both rocky server and ubuntu client can display the web server meaning we successfully verify that the web server is working.

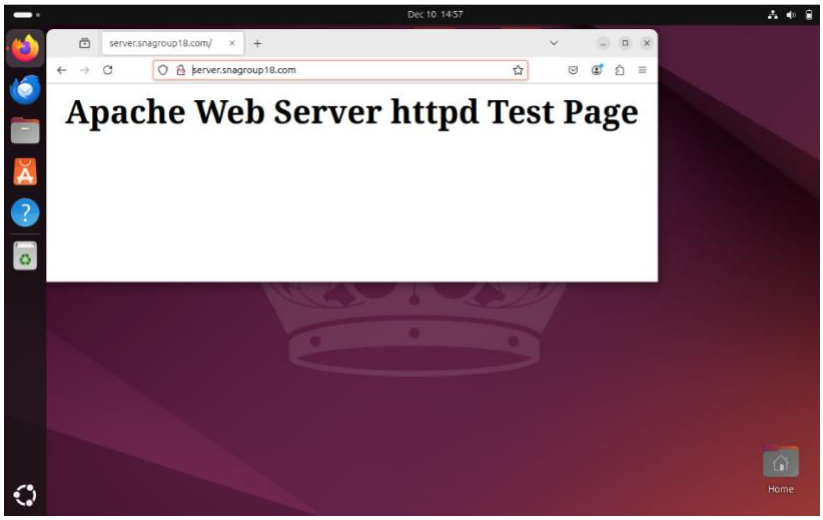


Figure 3.2.4: WebServer On Ubuntu

3.3 Configure Virtual Host

```
[SNAgroup18@server ~]$ sudo vi /etc/httpd/conf.d/vhost.conf
[sudo] password for SNAgroup18:
```

Figure 3.3.1: Accessing the Virtual Host Configuration File

Figure 3.3.1 showcases the command that is being executed "sudo vi /etc/httpd/conf.d/vhost.conf" which serves the purpose of accessing the virtual host "vhost.conf" configuration file, in the vi editor. The user password is requested as we are entering a superuser/elevated privileged mode by entering the command "sudo".

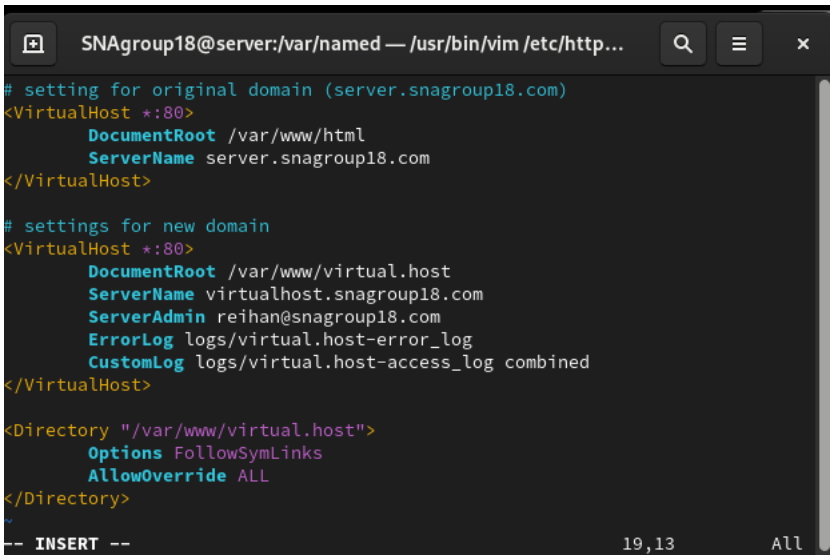


Figure 3.3.2: Virtual Host Configuration Information

In *Figure 3.3.2*, displays the content added within the virtual host configuration. In the figure, 2 separate blocks can be seen, which are the settings for the original domain and the new domain. Within the old/original domain, the directory that hosts every files under this site is specified under the "DocumentRoot" for which the directory "/var/www/html" is stated. The domain name for the virtual host "server.snagroup18.com" is also specified as the server name for this section. Meanwhile, below the settings of the old domain, the new domain settings is set by ensuring that several key subblocks are added as shown below:

- DocumentRoot: /var/www/virtual.host
- ServerName: virtualhost.snagroup18.com

- ServerAdmin: reihan@snagroup18.com
- ErrorLog: logs/virtual.host-error_log
- CustomLog: logs/virtual.host-access_log combined

From the 5 stated key information used for the new domain, each has its own purpose for which, the document root is to ensure that the host directory that handles this site is added in. The hostname of the virtual host is indicated to make certain that the Apache web server has the knowledge to identify which route requests is to be directed to this server name. The server administrator is included, for identifying the responsible administrator if any errors were to occur. The error and custom logs are added for tracking purposes of any errors and access. The permissions for directory "/var/www/virtual.host" is added under the directory block to permit symbolic links and ensure that overriding is enabled.

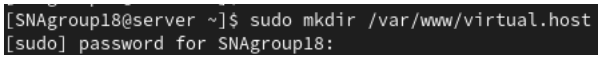


Figure 3.3.3: Generating Directory for Virtual Host

Figure 3.3.3 shows the command used to create a new directory called "/var/www/virtual.host" by entering the privileged user (sudo) mode and creating the file with command "mkdir". The purpose of creating this directory is to store every file that are destined to the new virtual host of "virtualhost.snagroup18.com".

Figure 3.3.4: Setting Permissions for /var/www/virtual.host/

Figure 3.3.4 is where the permissions for the new directory created in Figure 3.3.3 is set, for which, the permission is set to 755, which informs us that the owner is allowed to perform all read, write and execute activities within the directory, while group and other users are only allowed to read and execute without having access to being able to write. Setting such permissions are crucial for web server directory access.

Figure 3.3.5: Switching to elevated privileged user (root)

In *Figure 3.3.5*, showcases the user mode switching process, for which, the user is given the elevated privileged access by entering the command "su root", which prompts for a password from the user for user verification, and switches the user to a root user.

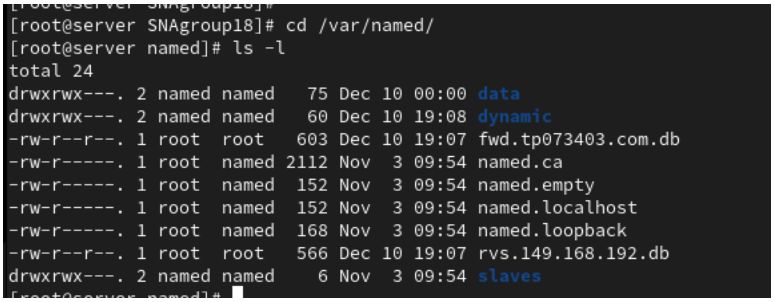


Figure 3.3.6: Listing all files in /var/named/

In *Figure 3.3.6*, displays the command "cd /var/named", that is used to swap current working directory to the named directory, which is the continued with the command "ls -l" which showcases the entire content within the /var/named/ directory.

Figure 3.3.7: Editing the Foward Zone File

Figure 3.3.7 shows the command "nano fwd.tp073403.com.db" which opens the forward zone file. It is opened for editing purposes using the nano editor. This file contains key information regarding the domain names and its set on IP addresses.

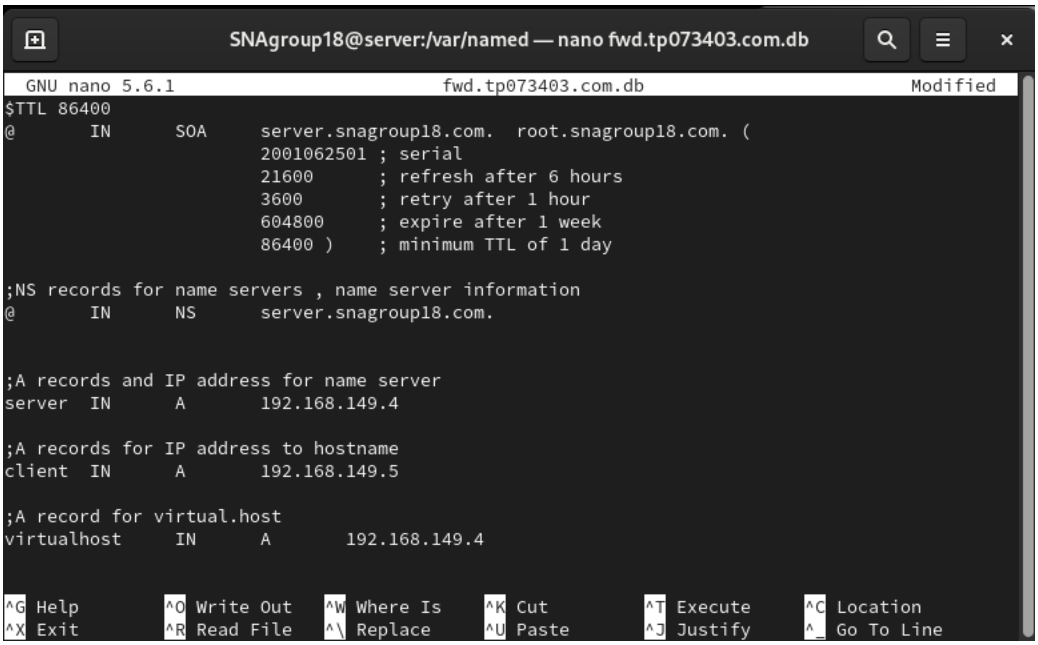


Figure 3.3.8: Foward Zone File "fwd.tp073403.com.db" Configuration

In *Figure 3.3.8*, showcases the information that were added to the forward zone file, some of the key takeouts are as such:

- NS Record
- A Record (server, client, and virtual host)

In the NS record, it declares the authorized name server for the domain (server.snagroup18.com). The A record for the name server maps the server.snagroup18.com with the server IP address of (192.168.149.4) while, the hostname of client.snagroup18.com is mapped to the clients IP address of (192.168.149.5). The virtual host on the other hand, resolves the virtualhost.snagroup18.com to 192.168.149.4. The purpose for mapping all 3 of the above stated domain names is to ensure that forward lookups are set to the right hostname and their respective IP addresses.

```
[root@server named]# nano rvs.149.168.192.db
```

Figure 3.3.9: Editing the Reverse Zone File

Figure 3.3.9 displays the command "rvs.149.168.192.db" which opens the reverse zone file. Like the forward zone file, the purpose of opening the file is to perform editing upon the file using the command-line editor of nano editor. This file characterizes how IP addresses are set back to the domain names.

Figure 3.3.10: Reverse Zone File "rvs.149.168.192.db" Configuration

Figure 3.3.10, displays the crucial information that was added in the reverse zone file configuration. Two key takeaways are:

- NS Record
- Pointer Record (PTR)

The NS record declares the authorized name server which is the "server.snagroup18.com" for reverse lookup purposes. The Pointer Record (PTR) is present, for the purpose of mapping IP addresses to a domain name. As seen in the figure above, the IP address of (192.168.149.4) is resolved to the domain name of "server.snagroup18.com". Next, the IP address of (192.168.149.5) is mapped to the domain name of "client.snagroup18.com". Meanwhile, the virtual host resolves

on the hostname of “virtualhost.snagroup18.com”. By configuring all 3 of the stated IP addresses to be mapped onto its respective hostnames, it makes certain, that proper reverse lookups performed are functioning effectively.

Figure 3.3.11: Restaring the Apache Web Server (httpd)

In *Figure 3.3.11*, showcases the command “sudo systemctl restart httpd” that is utilized for the purpose of restarting the Apache HTTP server. Restarting the service allows for all the changes made upon the configuration files to be applied, up, and running.

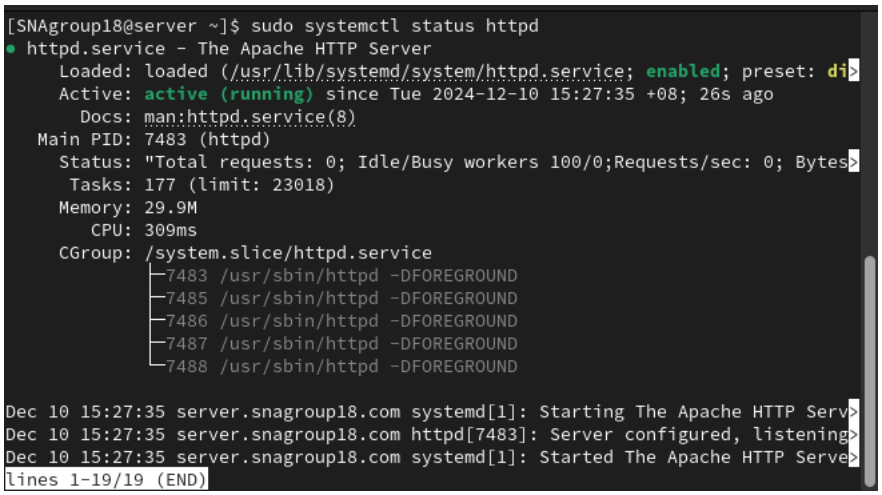


Figure 3.3.12: Checking the status of the Apache HTTP Web Server

Figure 3.3.12, portrays the Apache HTTP Web Server status check, for which, the command “sudo systemctl status httpd” is used. The output showcases several key information to verify the state of the running Apache service. As we can observe, the Apache HTTP Server (httpd.service) is currently in an active (running) state, which gives us the knowledge that the service is up and running in an active state, without any errors occurred. Also displayed in *Figure 3.3.12*, are the Process ID (PID), CPU details, as well as several lines of logs which informs us that the server is in a starting state, has been configured, and has started its functionalities.

3.4 Create a Website and Verify the Virtual Host

Figure 3.4.1: Creating a new HTML Page

Figure 3.4.1 showcases the command used to create a new HTML test page (index.html), that uses the default text editor of (vi), for which is under the root document of "/var/www/virtual.host/index.html" that sets up for "virtualhost.snagroup18.com" which is the virtual host.

Figure 3.4.2: HTML Test Page configuration

From the above Figure 3.4.2, showcases the content within the HTML Test Page which is utlizied to perform testing upon the Virtual Host to verify its functionality and content delivery. The figure displays several key commands that is crucial to ensure for a successful HTML test page such as:

Code	Function
<html>	Indicating the opening tag for the HTML document.
<body>	The opening tag for the body of the HTML document, which us utilized for displaying visual contents of the webpage.
	Is used to perform content styling and placement.
width: 100%;	Ensuring that the “div” spans out entirely using 100% of its width of the container.
font-size: 40px;	Sets the size of font to 40 pixels.
font-weight: bold;	Sets the weight of font to bold.
text-align: center;	Ensures that the text is positioned to the center of the container.
“Virtual Host Test Page”	The content that will be displayed in the webpage using the style and position format stated.
	Closing tag for the “div” element, indicating the end of the styling container.
</body>	Closing tag for the “body” element, specifying the end of the webpage visual contents.
</html>	Closing tag for the “html” element, marking the end of the HTML document.

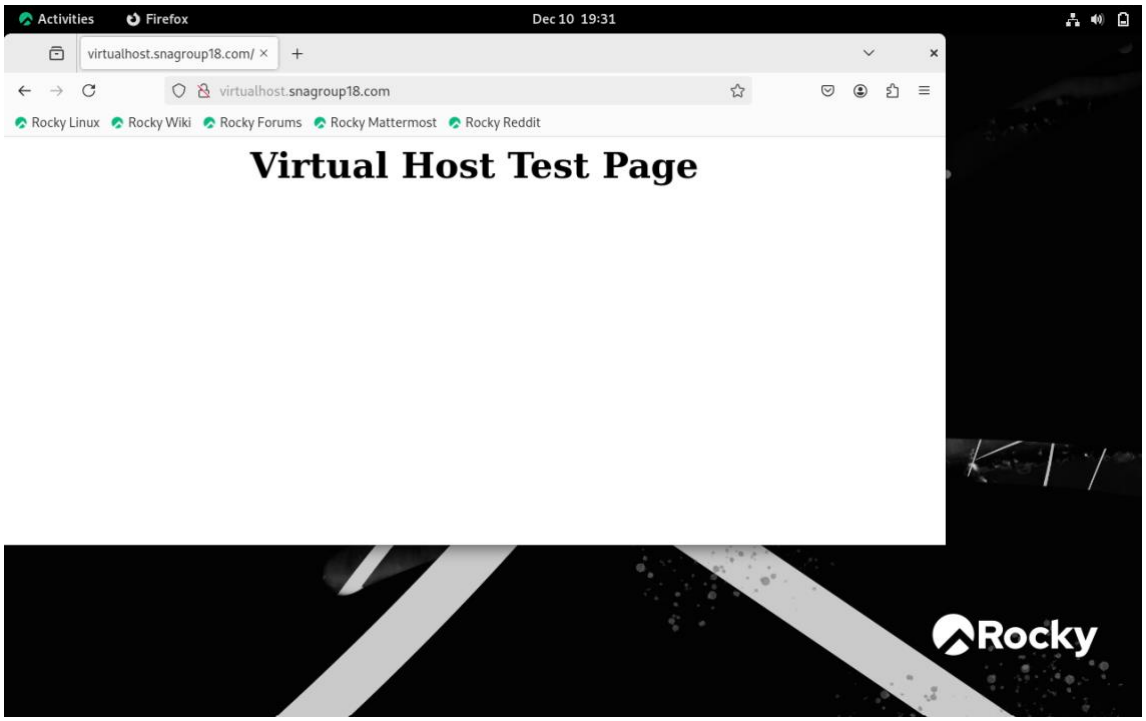


Figure 3.4.3: Testing on Browser – Rocky Linux

Figure 3.4.3 showcases the verification of ensuring that the configuration made upon the virtual host is working well and as expected. In the figure, the Mozilla Firefox web browser was utilized, and the URL of "virtualhost.snagroup18.com" is entered. Upon executing the URL, the right content was displayed "Virtual Host Test Page" which indicates that web server is resolving with the virtual host hostname (virtualhost.snagroup18.com) to the content discovered in /var/www/virtual.host/.

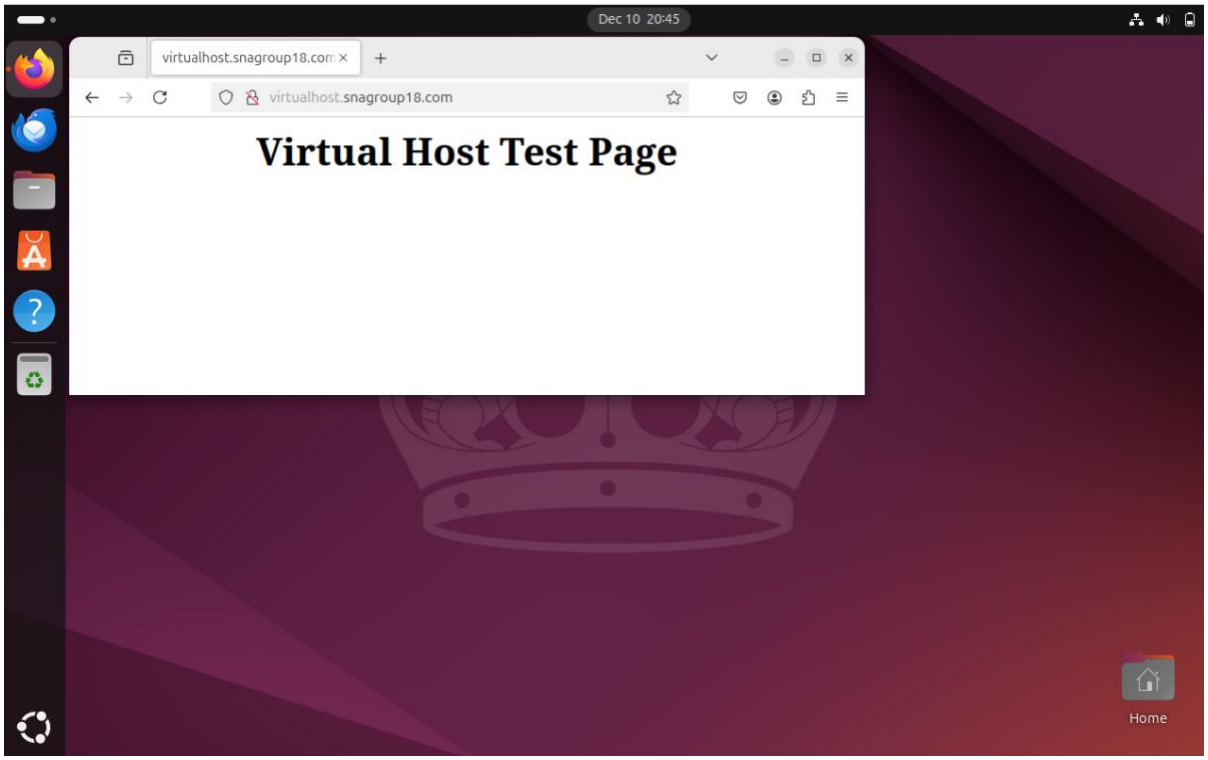


Figure 3.4.4: Testing on Browser – Ubuntu Client

In *Figure 3.4.4*, the similar processes are performed, for which, is to verify that virtual host is reachable from other systems (Ubuntu). The Mozilla Firefox web browser was used to demonstrate that the virtual host is reachable. In the URL, the "virtualhost.snagroup18.com" was entered and executed which resulted in the display of the correct content "Virtual Host Test Page", verifying that the virtual host is indeed reachable across different platforms. This validates that the configurations made, permits for outer systems like Ubuntu, to be able to gain access upon the virtual host.

3.5 Configure Certificate Authority (CA)

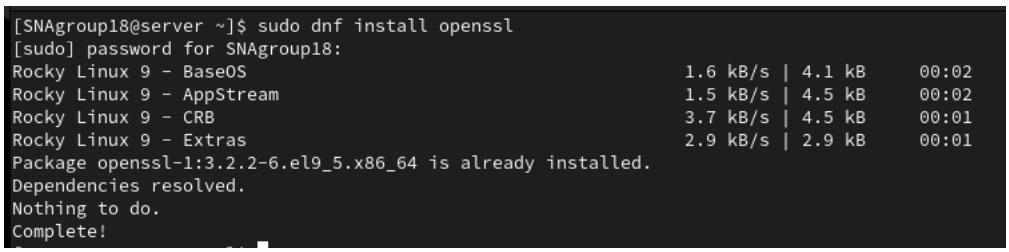


Figure 3.5.1 : Installation of openssl

The user attempted to install the OpenSSL package on a Rocky Linux 9 system using the `dnf` package manager. However, the installation was unsuccessful as the package was already installed. The `dnf` package manager verified the presence of the OpenSSL package and its dependencies, and reported that no further action was required. This indicates a successful completion of the installation process, even though the package was already present.

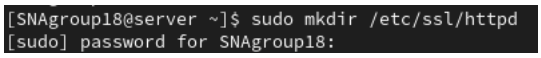


Figure 3.5.2 : sudo mkdir /etc/ssl/httpd

The image shows a user on a Rocky Linux 9 system attempting to create a directory named `/etc/ssl/httpd` using the `sudo` command. This command requires elevated privileges to modify system files. The user is prompted for their password to authenticate the `sudo` request.

Figure 3.5.3 : Create directory

The command generates a self-signed SSL certificate and private key pair for use with the Apache HTTP server. The certificate, valid for 365 days, is stored in `/etc/ssl/httpd/httpd-selfsigned.pem`. However, using self-signed certificates in production environments is not recommended due to potential security risks and browser warnings. It's best suited for testing or development purposes.

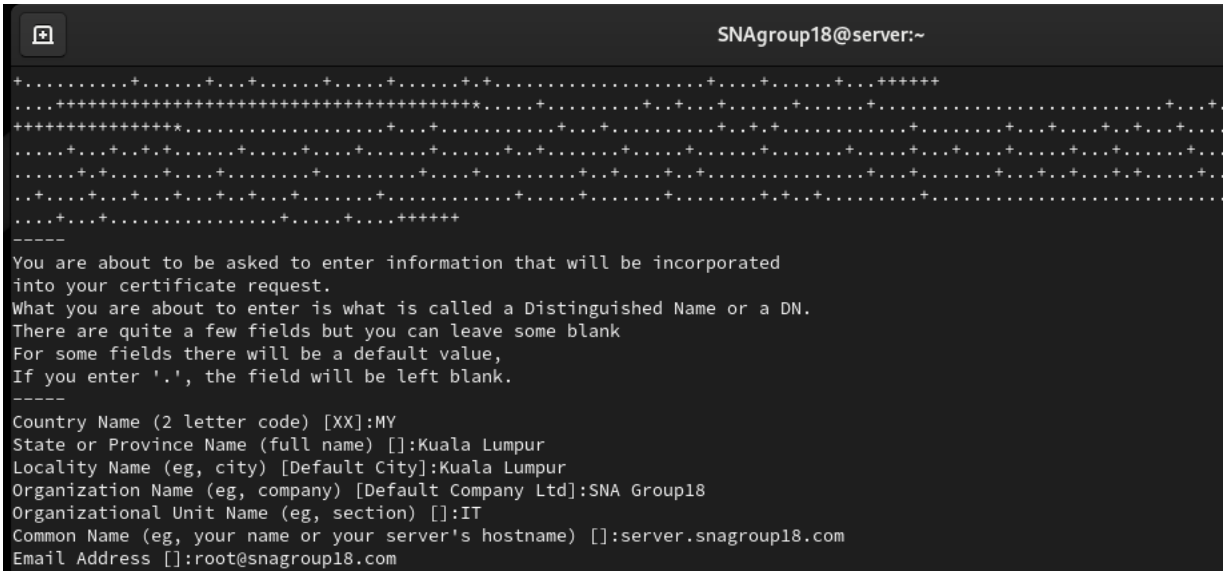


Figure 3.5.4 : Certificate Signing Request

The image displays a terminal session where a user is generating a Certificate Signing Request (CSR) using OpenSSL. The CSR will contain information such as the country, state, locality, organization, and common name of the entity requesting the certificate. This information, which includes details like Malaysia, Kuala Lumpur, SNA Group18, and server.snagroup18.com, will be embedded within the CSR. Once generated, the CSR can be submitted to a Certificate Authority (CA) to obtain a signed SSL certificate for secure web server communication.

Figure 3.5.5 : Genearting Signing Request

The terminal output shows the contents of the directory /etc/ssl/httpd/. It lists a single file named httpd-selfsigned.pem. This file likely contains a self-signed SSL certificate and its corresponding private key, which were previously generated using the OpenSSL command openssl req -x509 -nodes -keyout /etc/ssl/httpd/httpd-selfsigned.pem -out /etc/ssl/httpd/httpd-selfsigned.pem -days 365 -newkey rsa:2048.

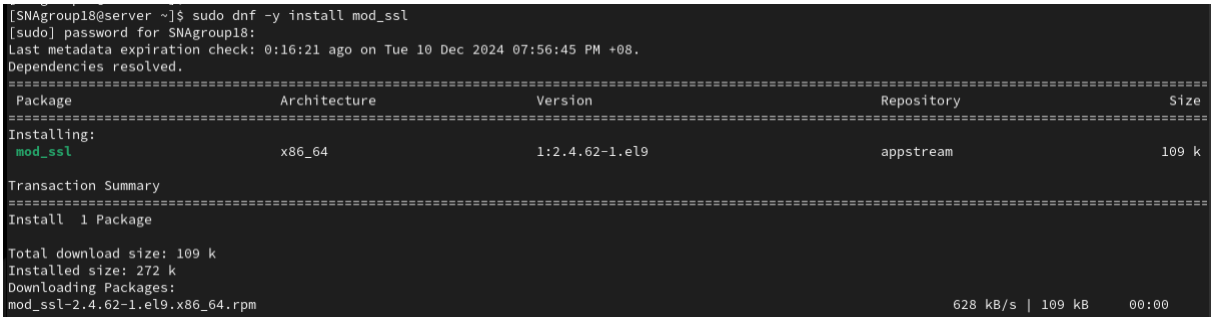


Figure 3.5.6 : Installation the mod-ssl package

The image shows a terminal session where the user is installing the mod_ssl package using the dnf package manager on a Rocky Linux system. The mod_ssl package is an Apache HTTP Server module that enables SSL/TLS encryption for secure communication.

The dnf package manager resolves the dependencies and downloads the mod_ssl package. Once the download is complete, the installation process begins, and the package is installed on the system.

Figure 3.5.7 : Open ssl confiuration file

The command sudo vi /etc/httpd/conf.d/ssl.conf is used to open the SSL configuration file /etc/httpd/conf.d/ssl.conf using the vi text editor with root privileges. This file contains the configuration settings for enabling SSL/TLS encryption on an Apache HTTP server.

By opening this file, the user can modify the SSL configuration settings, such as the SSL certificate and private key locations, SSL protocols, ciphers, and other relevant parameters.

Figure 3.5.8 : SSL configuration file

Apache HTTP server's SSL configuration file. It defines a virtual host for the domain server.snagroup18.com on port 443. This configuration ensures that when a client accesses the website using HTTPS, the server will serve content from the /var/www/html directory and use SSL/TLS encryption for secure communication.

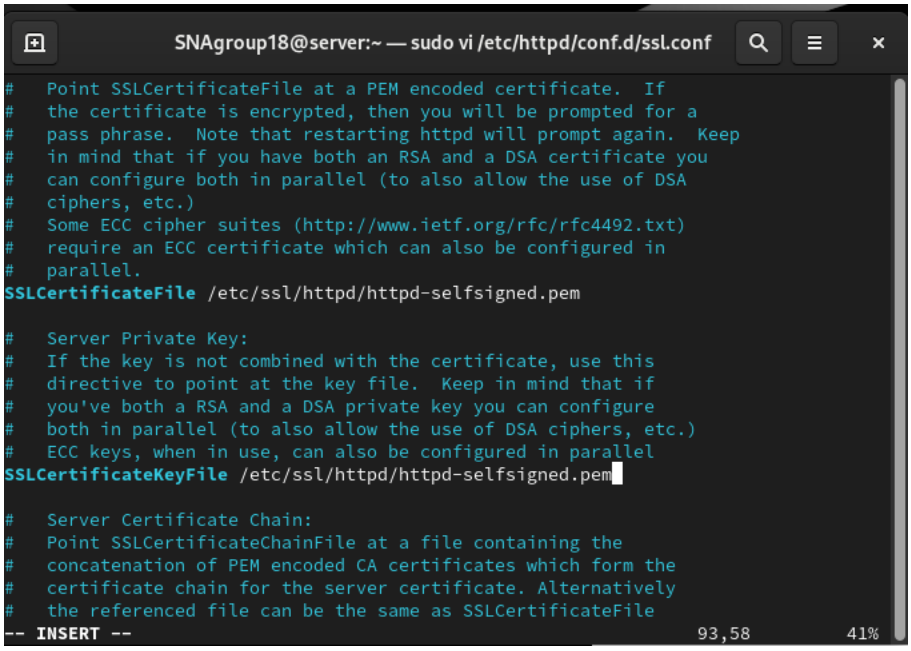


Figure 3.5.9 : SSL Configuration file

The Apache HTTP server's SSL configuration file specifies the location of the self-signed SSL certificate and private key files, both located in /etc/ssl/httpd/httpd-selfsigned.pem. While this configuration enables SSL/TLS encryption, it's crucial to remember that self-signed certificates are not suitable for production environments due to security concerns and browser warnings. They are primarily used for testing or development purposes.

```
[SNAgroup18@server ~]$ sudo systemctl restart httpd
[sudo] password for SNAgroup18:
```

Figure 3.5.10 : Restart Httpd

```
[SNAgroup18@server ~]$ sudo vi /etc/httpd/conf.d/vhost.conf
```

Figure 3.5.11: Open configuration with root privileges

The first command, sudo systemctl restart httpd, restarts the Apache HTTP server. This is often done after making changes to the server's configuration to apply the changes.

The second command, sudo vi /etc/httpd/conf.d/vhost.conf, opens the virtual host configuration file /etc/httpd/conf.d/vhost.conf using the vi text editor with root privileges. This file contains the configuration settings for one or more virtual hosts, which are used to define how the server handles requests for different domains or subdomains.

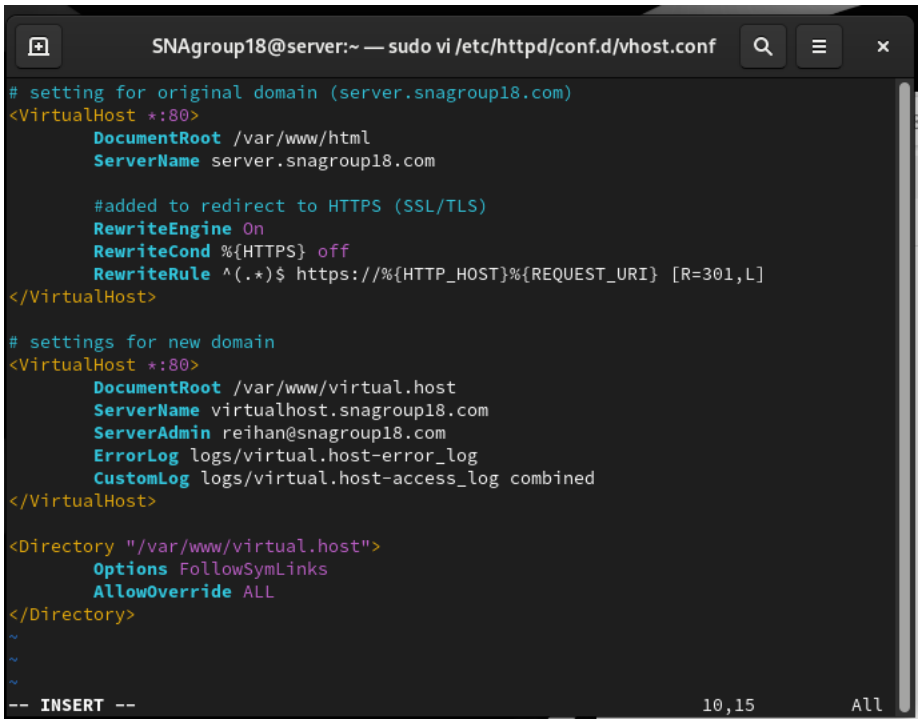


Figure 3.5.12 : using the vi text editor with root privileges

The configuration includes settings for document root directories, server names, error logging, and redirection from HTTP to HTTPS. This ensures that requests to the specified domains are handled correctly and securely.

```
[SNAgroup18@server ~]$ sudo systemctl restart httpd
```

Figure 3.5.13: Restarting httpd

The command sudo systemctl restart httpd is used to restart the Apache HTTP server. This is often done after making changes to the server's configuration to apply the changes. The sudo command is used to execute the command with root privileges, which is necessary to restart the server.

```
[SNAgroup18@server ~]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disab>
     Active: active (running) since Tue 2024-12-10 20:42:07 +08; 26s ago
       Docs: man:httpd.service(8)
   Main PID: 13401 (httpd)
     Status: "Total requests: 0; Idle/Busy workers 100/0;Requests/sec: 0; Bytes se>
      Tasks: 177 (limit: 23018)
     Memory: 26.7M
        CPU: 250ms
     CGroup: /system.slice/httpd.service
             ├─13401 /usr/sbin/httpd -DFOREGROUND
             ├─13402 /usr/sbin/httpd -DFOREGROUND
             ├─13403 /usr/sbin/httpd -DFOREGROUND
             ├─13404 /usr/sbin/httpd -DFOREGROUND
             └─13405 /usr/sbin/httpd -DFOREGROUND

Dec 10 20:42:07 server.snagroup18.com systemd[1]: Starting The Apache HTTP Server.>
Dec 10 20:42:07 server.snagroup18.com httpd[13401]: Server configured, listening o>
Dec 10 20:42:07 server.snagroup18.com systemd[1]: Started The Apache HTTP Server.
```

Figure 3.5.14 : Check the Apache HTTP server status

The terminal output shows the status of the Apache HTTP server. The server is currently active and running. It was started on Tuesday, December 10, 2024 at 20:42:07. The server is currently handling 0 requests and has 100 idle worker processes.

```
[SNAgroup18@server ~]$ sudo firewall-cmd --add-service=https
success
[SNAgroup18@server ~]$ sudo firewall-cmd --runtime-to-permanent
success
```

Figure 3.5.15 : sudo firewall – cmd – add-service=https

The user has opened port 443 (HTTPS) in the firewall by adding the HTTPS service and making the rule permanent. This allows incoming HTTPS traffic to reach the server.

3.6 Verify Certificate and HTTPS

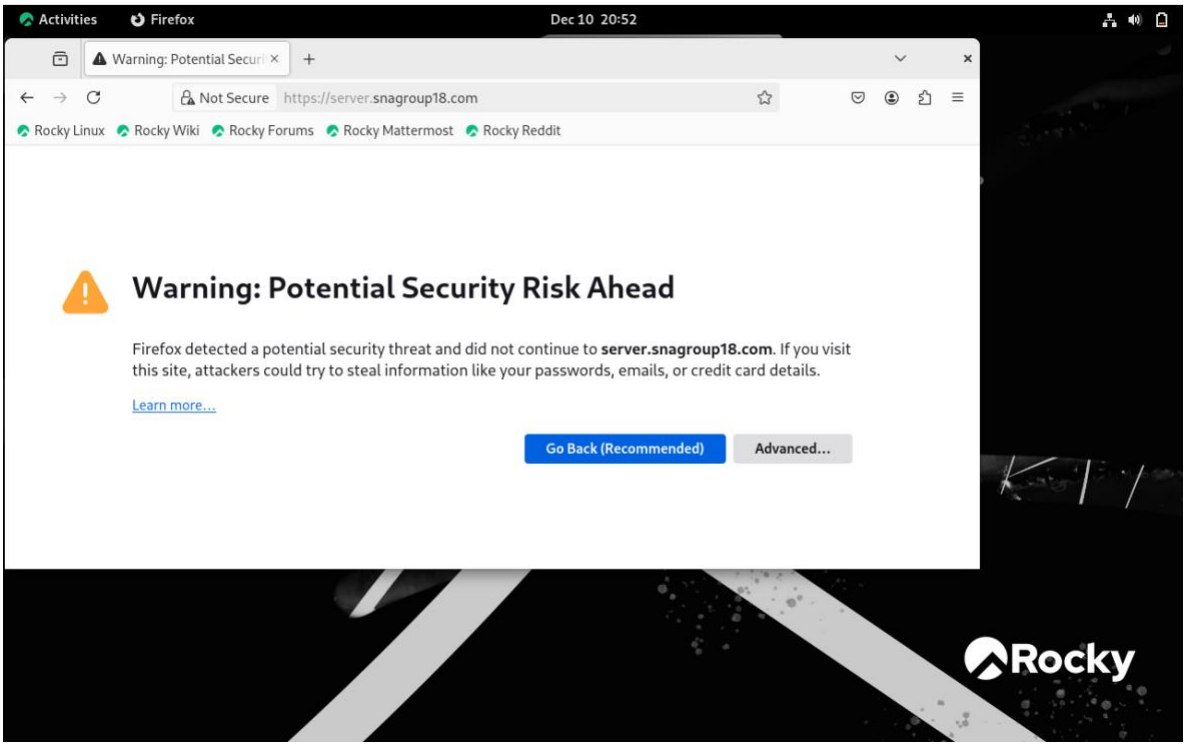


Figure 3.6.1: Potential Security Risk Warning

The warning indicates that Firefox detected a potential security threat and did not proceed to the website. This is likely due to the use of a self-signed SSL certificate, which is not trusted by default by browsers.

Self-signed certificates are often used for testing or development purposes, but they are not recommended for production environments. In production environments, it is important to use SSL certificates issued by a trusted Certificate Authority (CA) to ensure secure communication and avoid browser warnings.

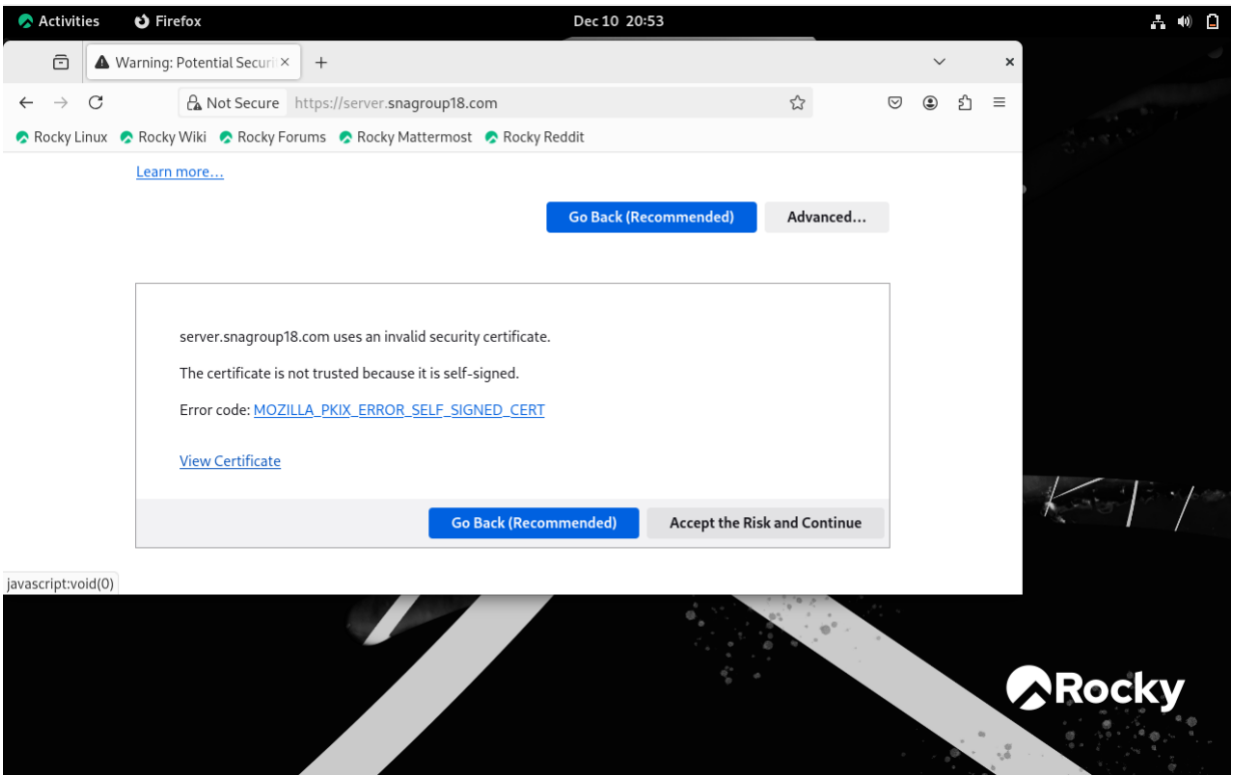


Figure 3.6.2 : Potential Secure

The image shows a Firefox browser displaying a security warning when trying to access the website server.snagroup18.com. The warning indicates that Firefox detected a potential security threat and did not proceed to the website. This is likely due to the use of a self-signed SSL certificate, which is not trusted by default by browsers.

Self-signed certificates are often used for testing or development purposes, but they are not recommended for production environments. In production environments, it is important to use SSL certificates issued by a trusted Certificate Authority (CA) to ensure secure communication and avoid browser warnings.

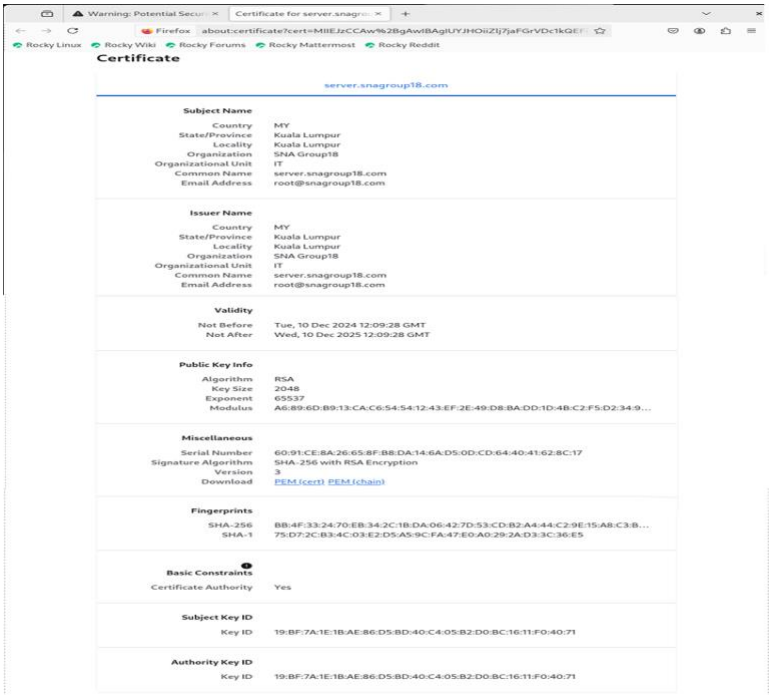


Figure 3.6.3 : Self – signed Certificate

This certificate is self-signed, meaning it was not issued by a trusted Certificate Authority. Self-signed certificates are often used for testing or development purposes, but they are not recommended for production environments due to security concerns and browser warnings.

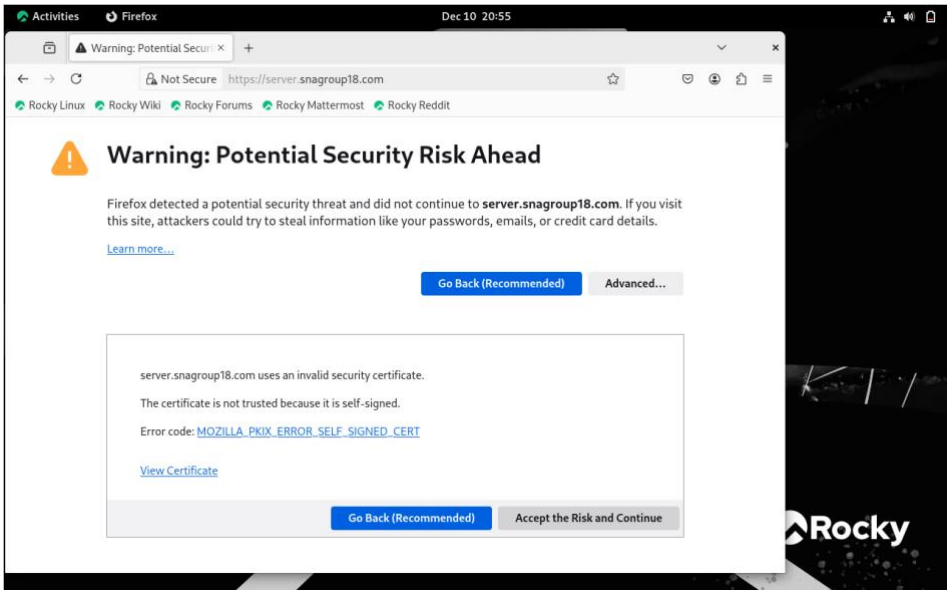


Figure 3.6.4 : Potential Security Risk

The warning indicates that Firefox detected a potential security threat and did not proceed to the website. This is likely due to the use of a self-signed SSL certificate, which is not trusted by default by browsers.

Self-signed certificates are often used for testing or development purposes, but they are not recommended for production environments. In production environments, it is important to use SSL certificates issued by a trusted Certificate Authority (CA) to ensure secure communication and avoid browser warnings.

Figure 3.6.5 : Apache Web Server httpd test page

Firefox browser displaying the "Apache Web Server httpd Test Page" on the website server.snagroup18.com. This indicates that the Apache HTTP server is running successfully and serving web pages. The website is likely being used for testing purposes, as it is displaying a simple test page.

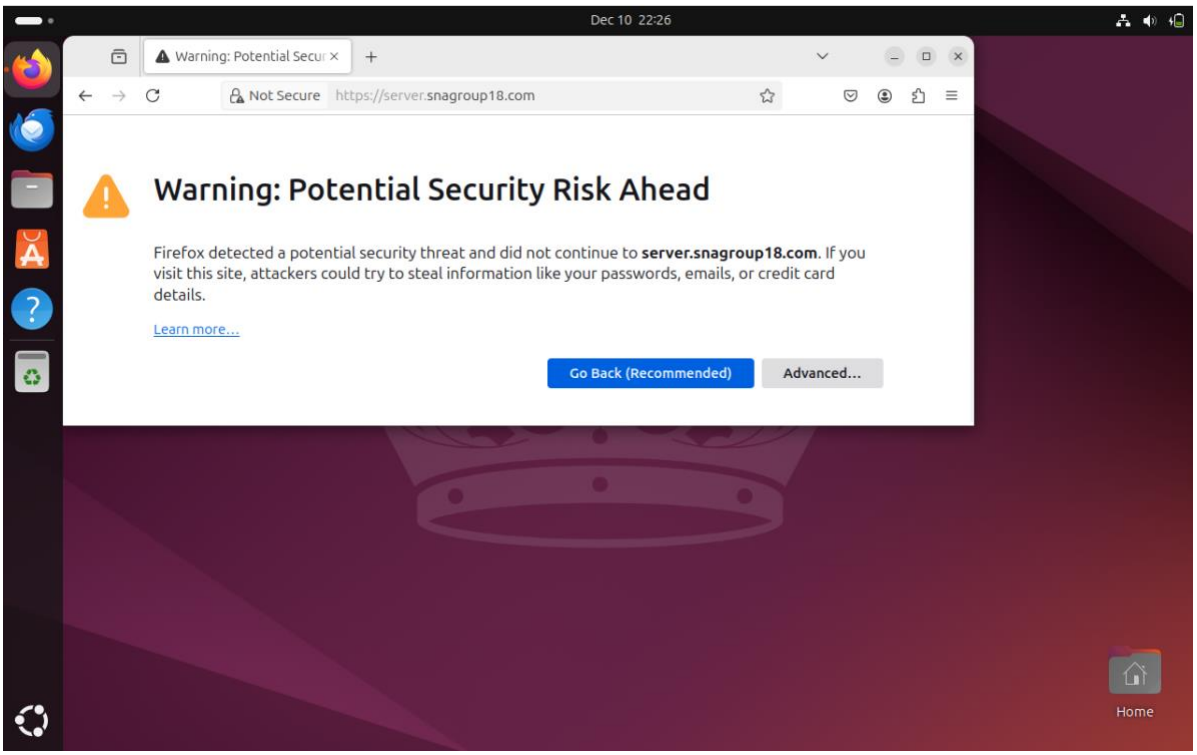


Figure 3.6.6 : Potential Security Risk

Next is for Ubuntu Clien,t with th The warning indicates that Firefox detected a potential security threat and did not proceed to the website. This is likely due to the use of a self-signed SSL certificate, which is not trusted by default by browsers.

Self-signed certificates are often used for testing or development purposes, but they are not recommended for production environments. In production environments, it is important to use SSL certificates issued by a trusted Certificate Authority (CA) to ensure secure communication and avoid browser warnings.

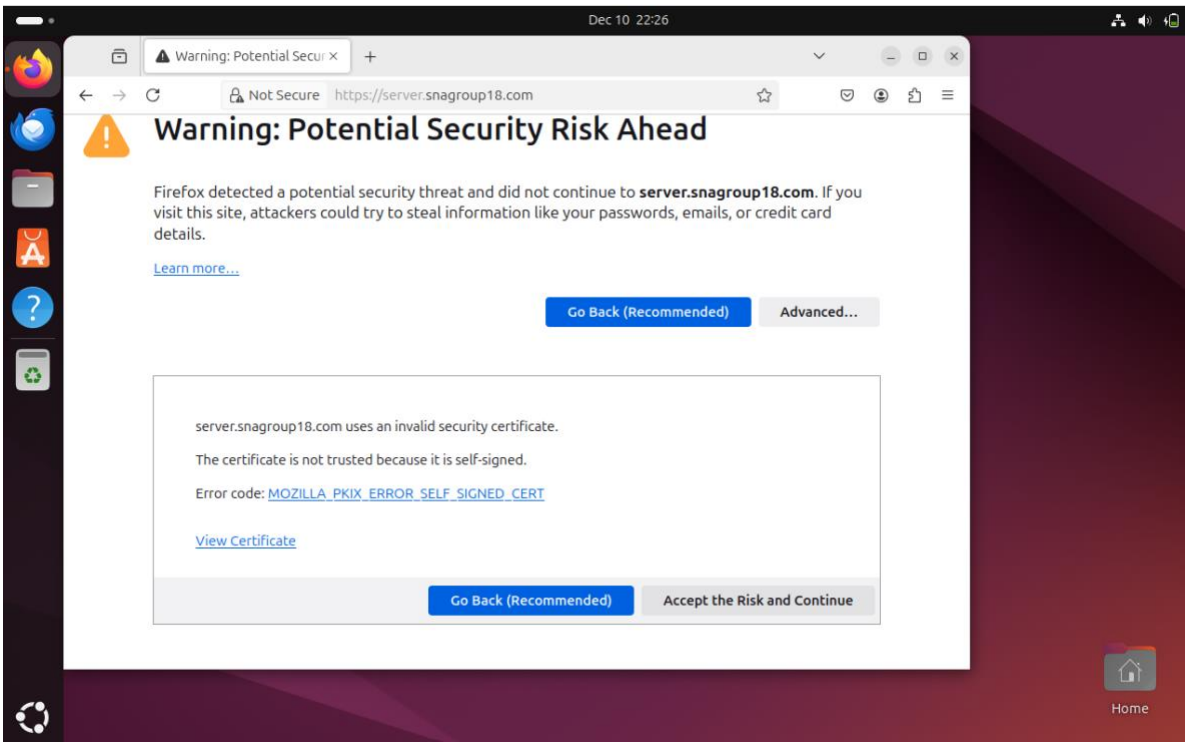


Figure 3.6.7 : Potential Security Risk

Self-signed certificates are often used for testing or development purposes, but they are not recommended for production environments. In production environments, it is important to use SSL certificates issued by a trusted Certificate Authority (CA) to ensure secure communication and avoid browser warnings.

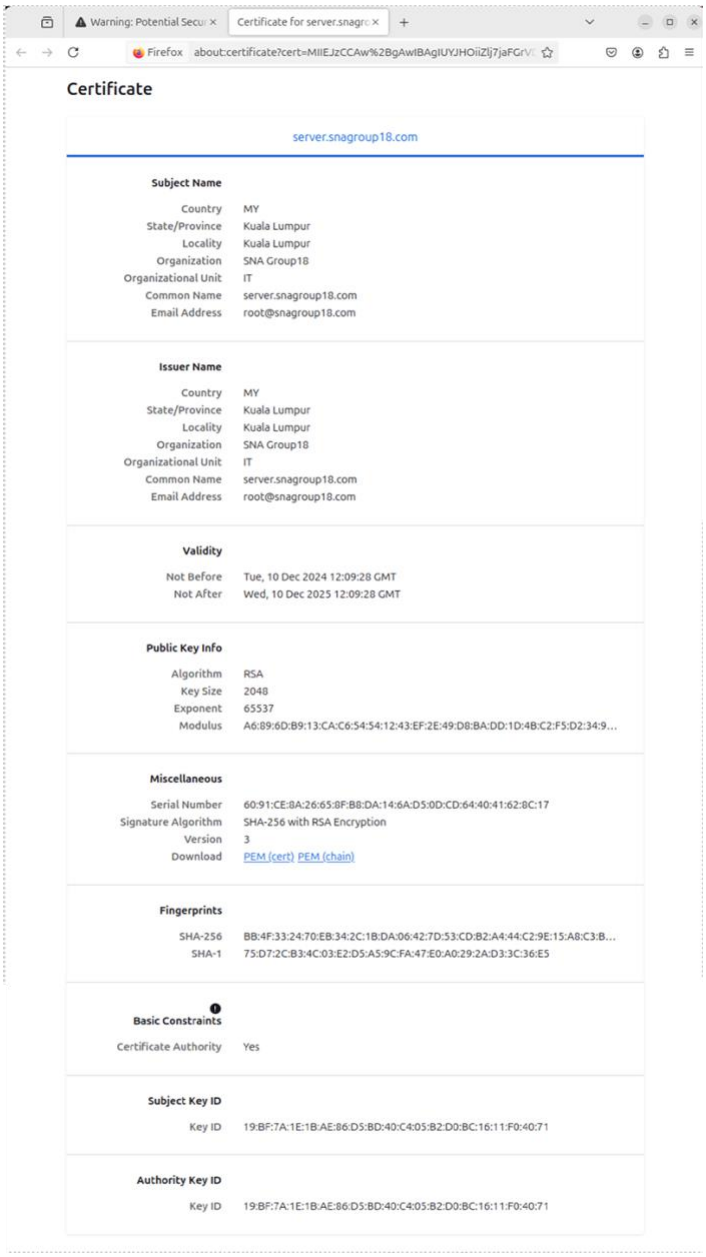


Figure 3.6.8 : Certificate

This certificate is self-signed, meaning it was not issued by a trusted Certificate Authority. Self-signed certificates are often used for testing or development purposes, but they are not recommended for production environments due to security concerns and browser warnings.

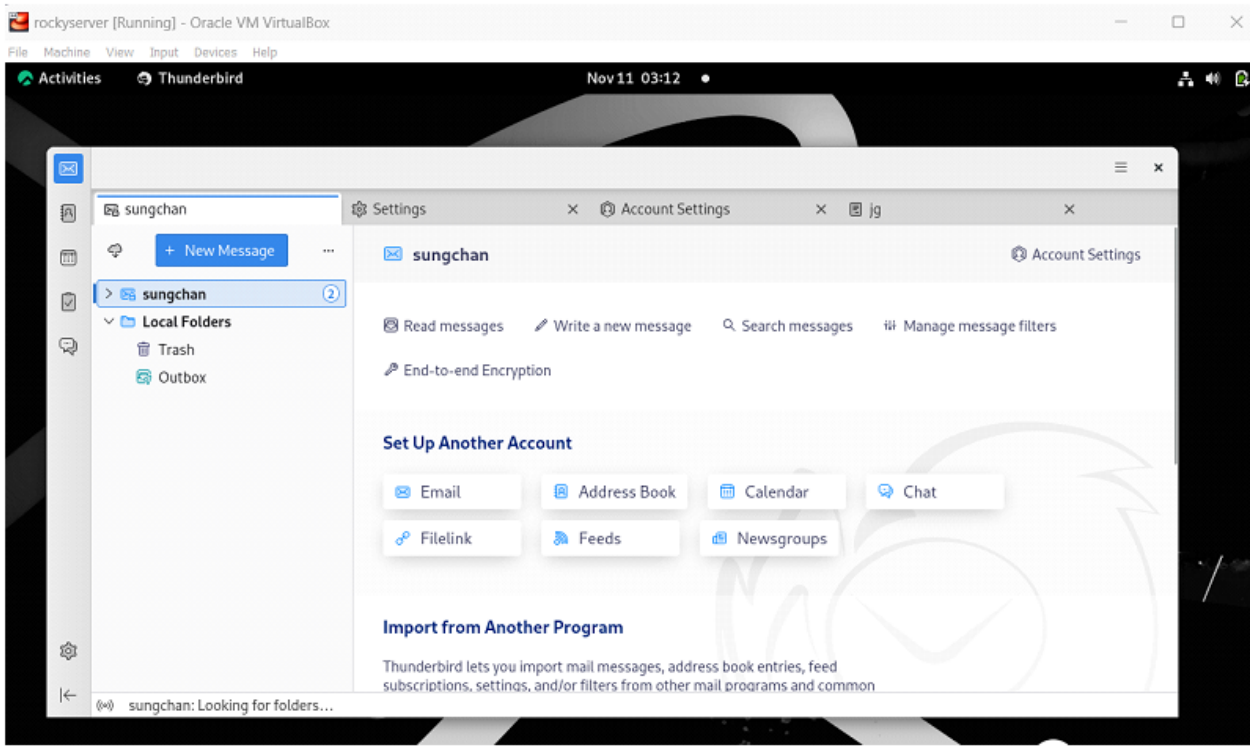


Figure 3.6.8 : MozillaThunderbird

On the client email side, open Mozilla Thunderbird, right-click on the email account, select Account Settings, and then choose Server Settings from the side panel.

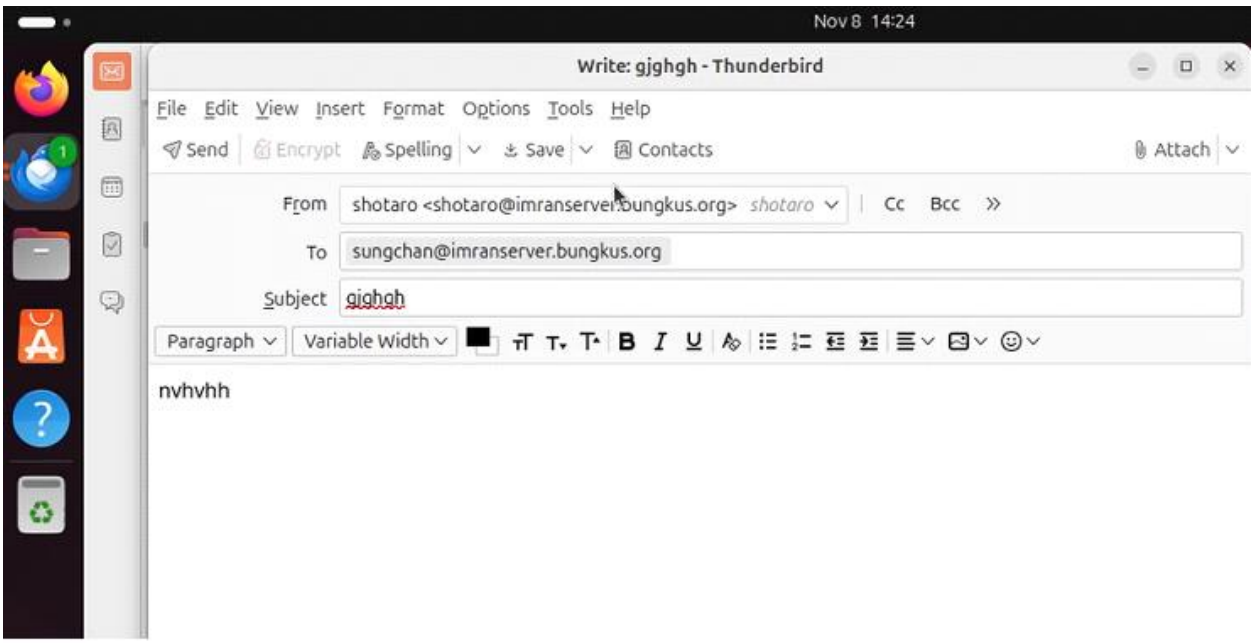


Figure 3.6.9 : Send email to the server

we can try to send email to the server side. Click on New Message. Send the email to sungchan@imranserver.bungkus.org, which is the email address for my rocky server. Enter any subject and message and click on the send button.

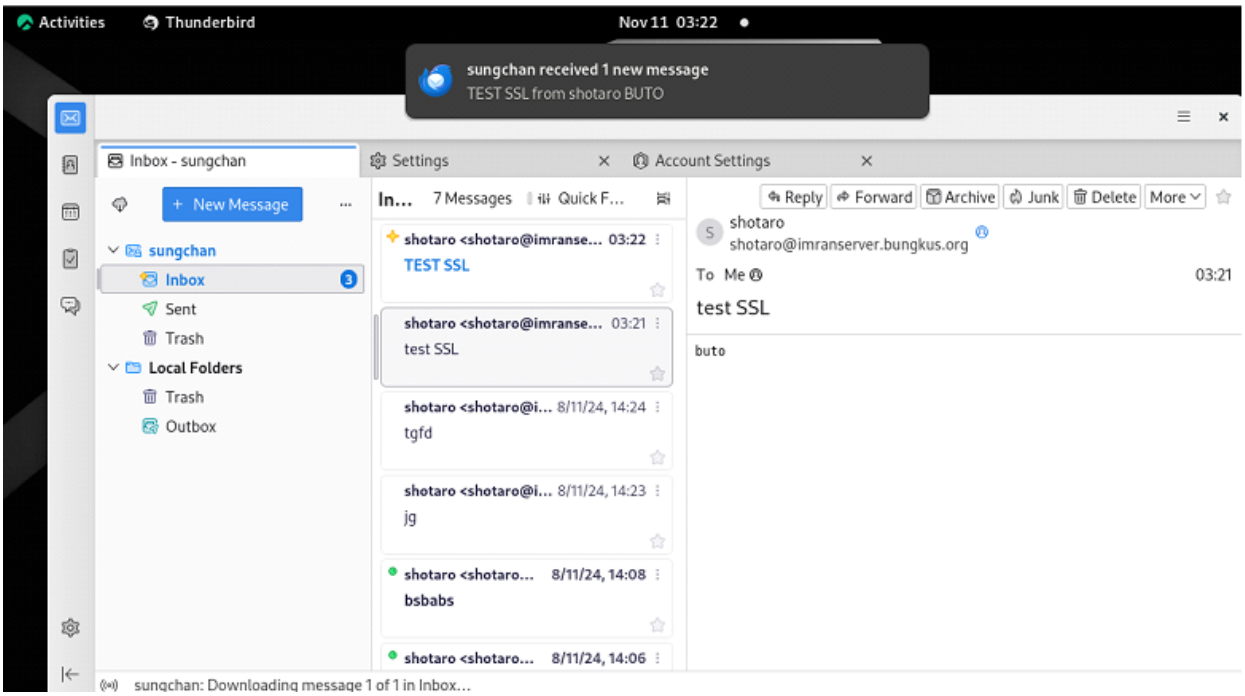


Figure 3.6.10 : Click New Message

The message is sent successfully after clicking the “Get Messages” icon.

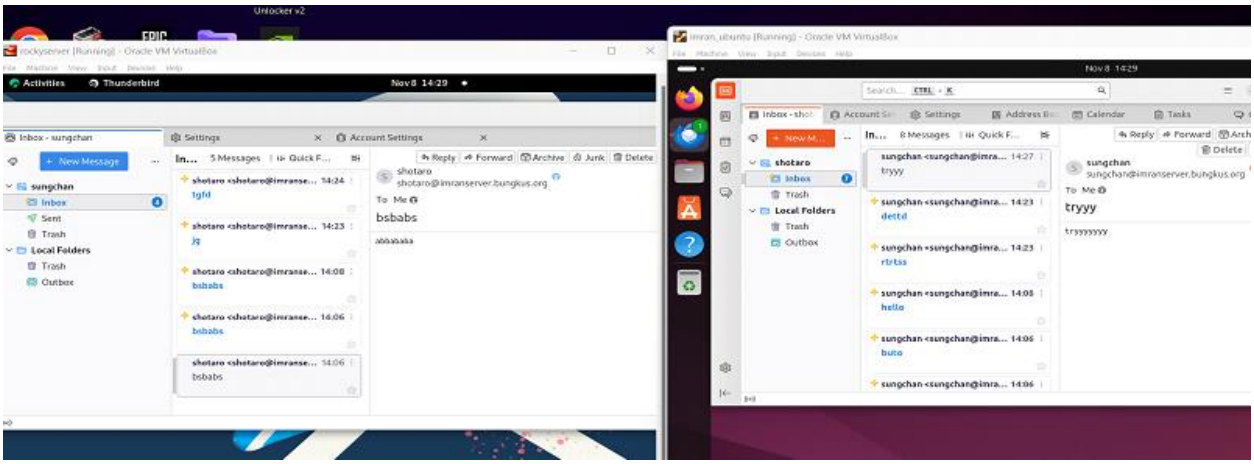


Figure 3.6.11 : Server email to recieve message

As you can see, It has been confirmed that the server email is able to receive messages sent from the client email. Both of the server and client email able to receive messages respectively.

4.0 Suricata IDS/IPS Configuration

Suricata is a network security monitoring tool that uses set of rules to examine and process network traffic. When a suspicious event occurs, Suricata could generate log events, trigger alerts, and drop traffic. Suricata could be set to run in passive mode, as an intrusion detection system (IDS), or in active mode, as an intrusion prevention system (IPS). IDS, the default mode for Suricata, scans network traffic for suspicious activity, and alert administrators for investigations and generate logs of the alerts. Suricata could also be set as an IPS which could do what IDS does but could also block traffic that matches specific rules set.

4.1 Configure Suricata as an Intrusion Detection System (IDS)

In the Rocky machine, install the dnf-copr plugin to community manage the Community Projects (copr) repositories, a community-driven repositories using the command:

- **sudo dnf install 'dnf-command(copr)'**

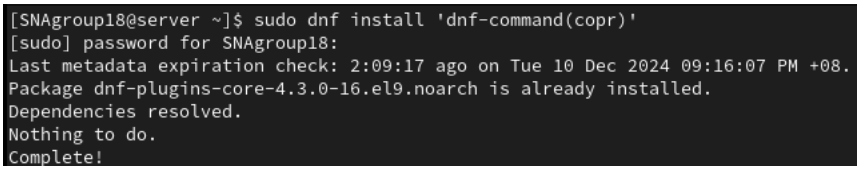


Figure 4.1.1: Installing dnf-copr login

Then enable Suricata from the COPR repository using the following command:

- **sudo dnf copr enable @oisf/suricata-6.0**

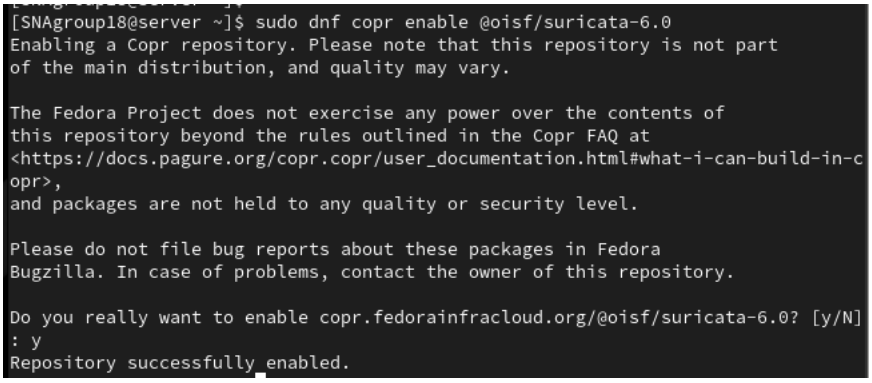


Figure 4.1.2: Enable suricata

Install the epel repository, which contains dependencies for Suricata, using the command:

- **sudo dnf install epel-release**

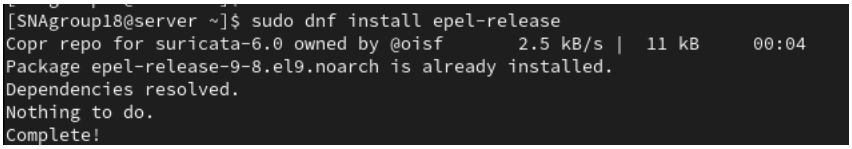


Figure 4.1.3: Installing epel repository

After all required repositories are installed and enabled, install Suricata using the command:

- **sudo dnf install suricata**

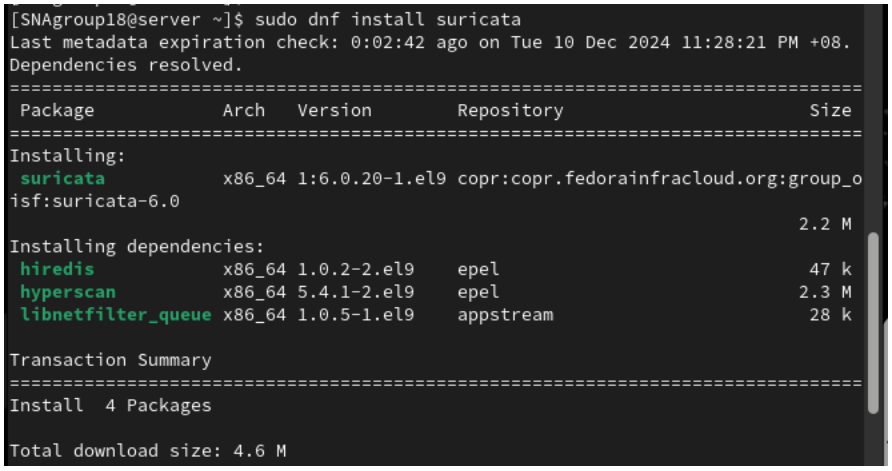


Figure 4.1.4: Installing Suricata

Enable the Suricata service using the following commands:

- **sudo systemctl enable suricata.service**
- **systemctl**: use to manage system and services

Figure 4.1.5: enable suricata service

There is no error message which means that it was enabled successfully.

Now stop the service to ensure that configuration and changes that would be made will be validated and loaded when Suricata starts up again.

- **sudo systemctl stop suricata.service**
- **systemctl**: use to manage system and services

Figure 4.1.6: Stopping suricata service

Enable Community ID so that other tools, like Elasticsearch (part of ELK stack to create SIEM), could be used with Suricata.

Edit the configuration file for Suricata using the command below:

- **sudo vi /etc/suricata/suricata.yaml**
- **vi**: terminal-based text editor
- **/etc/suricata/suricata.yaml**: the directory of the Suricata configuration file

Figure 4.1.7: Editing Suricata Configuration File

With the configuration file open using “vi”, do the following edit to the file:

- At line 132 change “community-id” to be equal to “true”
- **community-id: true**
- Enable community id feature

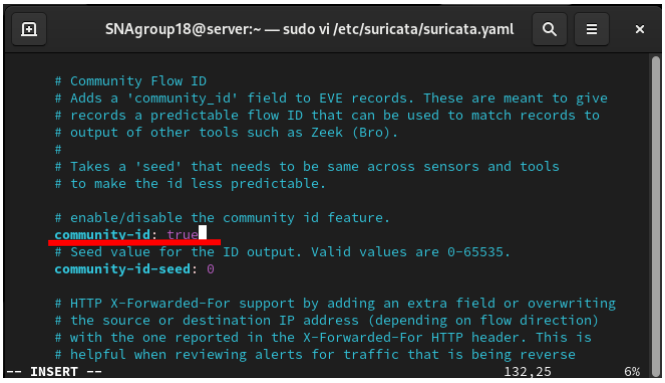


Figure 4.1.8: Set community id

Then save and quit the configuration file.

Determine the device name of the default network interface using the command below:

- **ip -p -j route show default**
- **ip**: utility for managing network interfaces, routing, and IP addresses in Linux
- **-p**: format output to be more readable
- **-j**: outputs the result in JSON format, structure that's easy to read
- **route show**: displays the system's routing table
- **default**: show only the default gateway

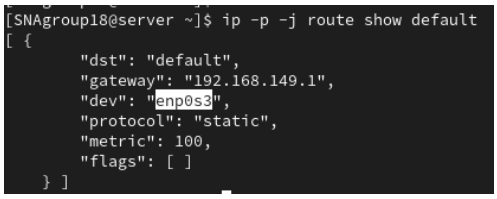


Figure 4.1.9: Determine Device Name

In this case the device name is "enp0s3". This would be used to specify the network interface that Suricata should inspect the traffic of. The configuration file for Suricata would be modify.

Edit the configuration file for Suricata using the command below:

- **sudo vi /etc/suricata/suricata.yaml**
- **vi**: terminal-based text editor
- **/etc/suricata/suricata.yaml**: the directory of the Suricata configuration file

Figure 4.1.10: Editing Suricata File

With the configuration file open using "vi", do the following edit to the file:

- At line 596 change "interface" to be the device of the default network
- **interface: enp0s3**
- "enp0s3" is the network interface Suricata will use to capture traffic

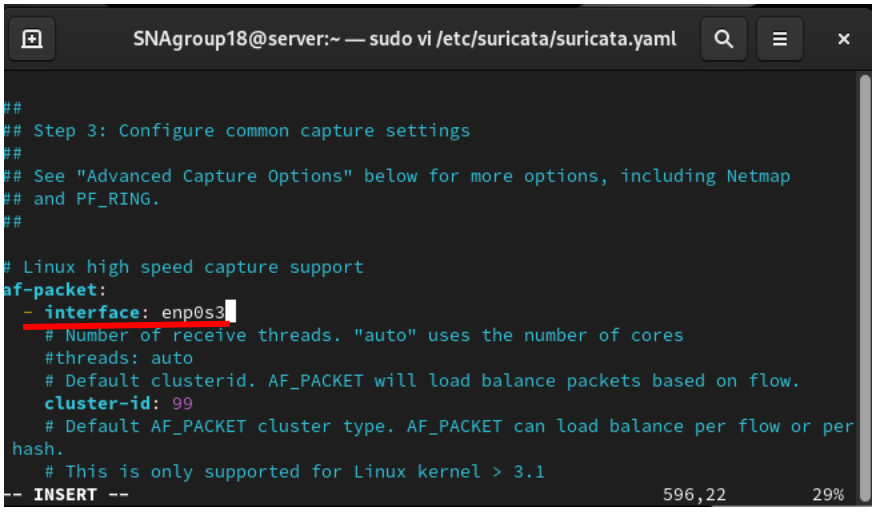


Figure 4.1.11: Setting Interface

Scroll to the end of the file and add the following:

- **detect-engine:**

  -rule-reload: true

- enables Suricata to reload detection rules automatically during runtime

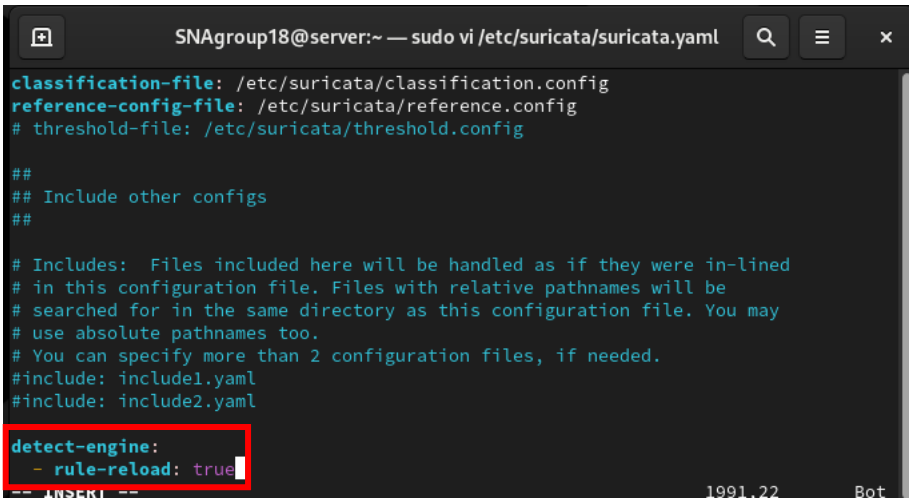


Figure 4.1.12: Set Rule-Reload

Then save and quit the configuration file.

Download the latest rulesets for Suricata from external providers using the command:

- **sudo suricata-update**

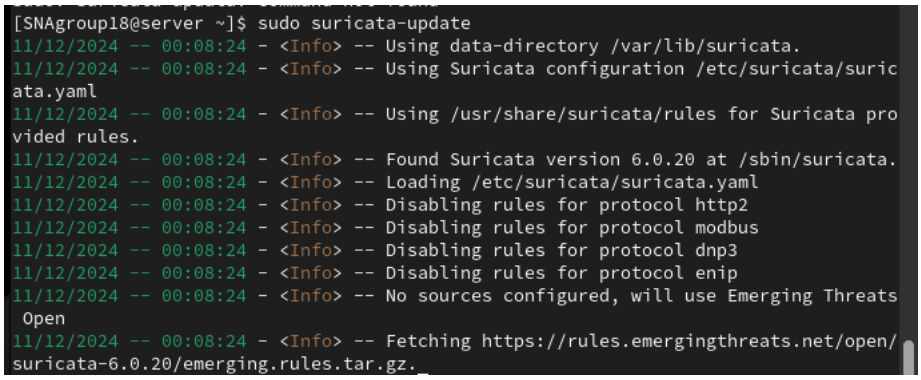


Figure 4.1.13: Downloading Latest Suricata

Add “list-sources” to the end of the previous command to get a list of rule providers:

- **sudo suricata-update list-sources**

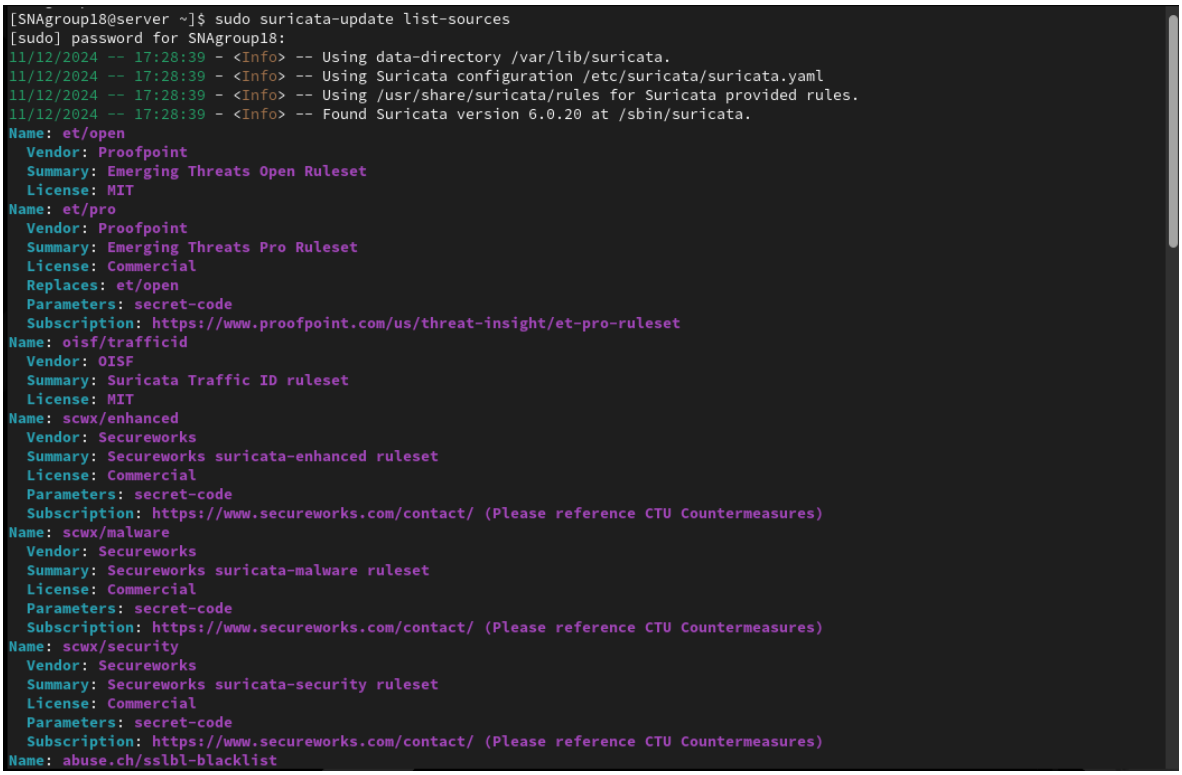


Figure 4.1.14: Listing Rule Providers

Important sections for each ruleset are “Name:” is the name of the ruleset, “Vendor:” is the name of the provider of the ruleset, “Summary:” is a brief description about the ruleset. Some of the rulesets are free while others are commercial/paid.

To add a specific ruleset to Suricata service, the name of the ruleset is required. For example, a ruleset for threat hunting by tgreen, like below, wants to be enabled in Suricata.

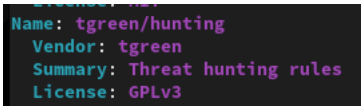


Figure 4.1.15: Adding Specific Ruleset

The name “tgreen/hunting” would be used in the command to enable this ruleset:

- **sudo suricata-update enable-source tgreen/hunting**
- **suricata-update**: manage Suricata's rule sets

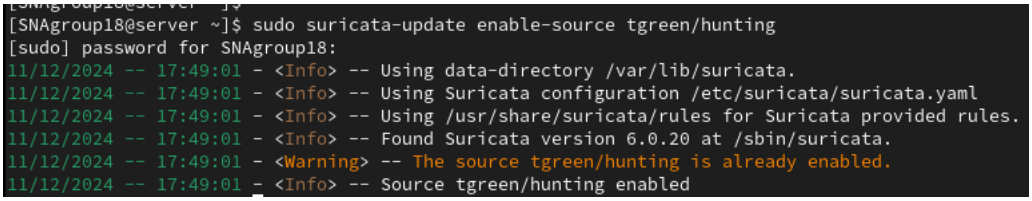


Figure 4.1.16: Enable ruleset

Edit the system configuration file for Suricata to use the network interface from “af-packet” in the Suricata configuration file. By default, it is set to “eth0” and needs to be change in order to be able to run the Suricata service.

- **sudo nano /etc/sysconfig/suricata**
- **nano**: terminal-based text editor

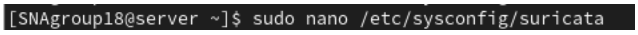


Figure 4.1.17: Editing Configuring File

With the configuration file open using “nano”, do the following edit to the file:

- Switch “eth0” to “--af-packet”
- **OPTIONS=“--af-packet –user suricata”**
- Instructs Suricata to use the “af_packet”
- Set Suricata to run as the non-root user suricata

Figure 4.1.18: Setting Option To Daemon

Then save and quit the configuration file.

Validate changes made to the configuration file and rules of Suricata using the command:

- **sudo suricata -T -c /etc/suricata/suricata.yaml -v**
- **suricata**: initiates Suricata's functionality
- **-T**: run Suricata in test mode
- **-c /etc/suricata/suricata.yaml**: location of Suricata's main configuration file
- **-v**: verbose mode, provide a more detailed information

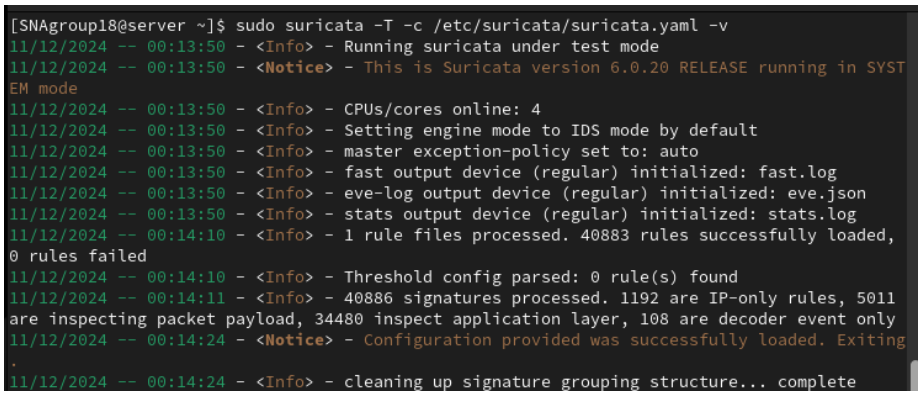


Figure 4.1.19: Validating Changes

There is no error which means that it was successful.

Restart Suricata to apply changes made and check the status using the following:

- **sudo systemctl restart suricata.service**
- **systemctl**: use to manage system and services
- **restart**: to reload (stop and then start) a service

```
[SNAgroup18@server ~]$ sudo systemctl restart suricata.service
```

Figure 4.1.20: Restarting Suricata Service

- **sudo systemctl status suricata.service**
- **status**: to display the current status of a service

```
[SNAgroup18@server ~]$ sudo systemctl status suricata.service
● suricata.service - Suricata Intrusion Detection Service
     Loaded: loaded (/usr/lib/systemd/system/suricata.service; enabled; preset: disabled)
    Drop-In: /etc/systemd/system/suricata.service.d
             └─override.conf
     Active: active (running) since Wed 2024-12-11 00:51:19 +08; 3min 6s ago
       Docs: man:suricata(1)
    Process: 6360 ExecStartPre=/bin/rm -f /var/run/suricata.pid (code=exited, status=0/SUCCESS)
   Main PID: 6361 (Suricata-Main)
      Tasks: 10 (limit: 23018)
     Memory: 469.5M
        CPU: 37.393s
     CGroup: /system.slice/suricata.service
             └─6361 /sbin/suricata -c /etc/suricata/suricata.yaml -i enp0s3 --pidfile /var/run/suricata.pid --af-packet --user suricata

Dec 11 00:51:19 server.snagroup18.com systemd[1]: Starting Suricata Intrusion Detection Service...
Dec 11 00:51:19 server.snagroup18.com systemd[1]: Started Suricata Intrusion Detection Service.
Dec 11 00:51:19 server.snagroup18.com suricata[6361]: 11/12/2024 -- 00:51:19 - <Info> - Multiple af-packet option without interface on
Dec 11 00:51:19 server.snagroup18.com suricata[6361]: 11/12/2024 -- 00:51:19 - <Notice> - This is Suricata version 6.0.20 RELEASE runn
Dec 11 00:51:52 server.snagroup18.com suricata[6361]: 11/12/2024 -- 00:51:52 - <Notice> - all 4 packet processing threads, 4 management
```

Figure 4.1.21: Display Suricata Status

The service would start during boot since the service is “enabled” in the “Loaded” option. The service is also currently active and running shown by “active (running)” in the “Active” option.

Suricata might take a while to start. To check if Suricata finished starting use the following:

- **sudo tail -f /var/log/suricata/suricata.log**
- **tail**: used to view the end of a file
- **-f**: stands for "follow", it keeps the file open and continuously display new lines as they are added to the file
- **/var/log/suricata/suricata.log**: path of Suricata's log file

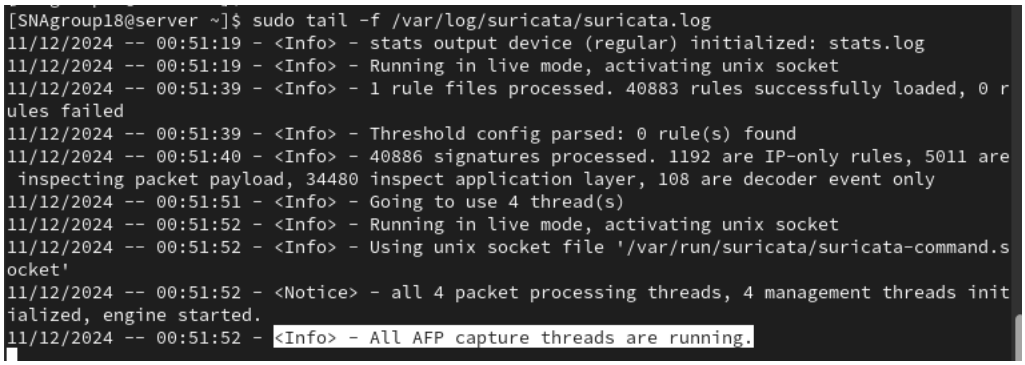


Figure 4.1.22: Checking Suricata Log

Watch for the line with the string “All AFP capture threads are running.”, which means that Suricata finished starting and is running and ready to inspect traffic.

4.2 Verify Suricata IDS and Test Intrusion Detection (Passive Mode)

Test whether Suricata IDS could detect suspicious traffic and generate an alert log for those events. Suricata’s Quickstart documentation recommends testing the IDS using ET Open ruleset, specifically rule number 2100498. Rule number 2100498 in ET Open ruleset detect suspicious activity based on pattern or event in HTTP traffic and an alert is triggered when it occurs. To test this the “curl” command could be used to generate a HTTP request to trigger an alert.

- **curl http://testmynids.org/uid/index.html**
- **curl**: tool used to send a request to a specified URL
- **http://testmynids.org/uid/index.html**: the URL to access, should trigger ET Open rule number 2100498

Figure 4.2.1: Test Suricata IDS

A HTTP respond is received, and the content of the page is shown. This is because IDS do not prevent but only detect. Check the Suricata log for entry with the rule number 2100498 to see if the “curl” done was detected. Search entries that match the rule number using the command:

- **sudo grep 2100498 /var/log/suricata/fast.log**
- **grep**: search for patterns in files
- **2100498**: the rule number to search for
- **/var/log/suricata/fast.log**: the log file where Suricata writes alerts and events

Figure 4.2.2: Check suricata log entry

These are the important findings from the log entry. There is a log entry with the rule number 2100498. The classification is “Potentially Bad Traffic” which means that there might be suspicious activity and require further investigation. The priority of the alert is 2 which is quite high, the smaller the value the higher the priority of the entry to be investigated. The protocol used was TCP and the source IP address is 18.67.181.16 with the port 80, and the destination IP address is 192.168.149.4 (the IP address of the Rocky machine) on the port 36940.

Suricata also logs event in another file, with the nickname EVE log. This uses JSON format entries and is made to be more readable for machines. Before proceeding, jq needs to be installed, a command-line JSON processor, to query, read, and filter the log file.

- **sudo dnf install jq**

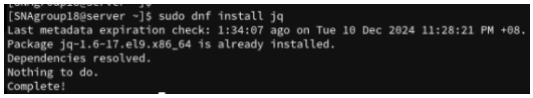


Figure 4.2.3: Installing jq

After jq is installed, filter events in the EVE log to show entry with the rule number 2100498

- **jq 'select(.alert .signature_id==2100498)' /var/log/suricata/eve.json**
- **jq**: JSON processor to query JSON data
- **'select(.alert .signature_id==2100498)'**: select and return JSON entries where the “signature_id” in the “alert” object is equal to 2100498, the rule number
- **/var/log/suricata/eve.json**: the EVE JSON log file where Suricata writes alerts

Figure 4.2.4: Filtering Events

These are the important findings from the log entry. The network interface on which this traffic was observed is “enp0s3”. The event is an alert, which is when Suricata detects a network activity that matches the rule. The source IP address is 18.67.181.16 with the port 80 and the destination IP address is 192.168.149.4 (the IP address of the Rocky machine) on the port 36940, and it used TCP. After detecting this Suricata “allowed” the connection and was not drop or blocked, which is a typical behaviour for an IDS. The rule number is 2100498. The category is “Potentially Bad Traffic”, and the severity of the alert is 2.

4.3 Configure Custom Rule in Suricata

Besides using rules from rulesets created by providers, Suricata also allows the creation of custom rules. Firstly, find Rocky server's IP address to be used in the custom rules using below:

- **ip -brief address show**
- **ip**: utility for managing network interfaces, routing, and IP addresses in Linux
- **-brief**: output the information in a simple format
- **address**: show IP addresses associated with the system's network interface(s)
- **show**: display the information, command would not provide output without this

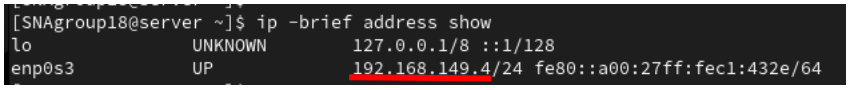


Figure 4.3.1: Show IP Address

From the output above the IP address of the Rocky server machine is 192.168.149.4.

To create custom rules, edit the local rules file for Suricata using the command below:

- **sudo vi /var/lib/suricata/rules/local.rules**
- **vi**: terminal-based text editor
- **/var/lib/suricata/rules/local.rules**: the directory to define custom Suricata rules

Figure 4.3.2: Editing Local Rules

With the custom rules file open using "vi", do the following edit to the file:

- **alert ssh any any -> 192.168.149.4 !22 (msg:"SSH TRAFFIC on non-SSH port"; flow:to_client, not_established; classtype: misc-attack; target: dest_ip; sid:1000000;)**
- Generate an alert for SSH traffic to the Rocky server with IP address 192.168.149.4 on non-SSH ports (all ports except port 22)
- **alert http any any -> 192.168.149.4 !80 (msg:"HTTP REQUEST on non-HTTP port"; flow:to_client, not_established; classtype:misc-activity; sid:1000002;)**
- Generate an alert for HTTP traffic to the Rocky server with IP address 192.168.149.4 on non-standard HTTP ports (all ports except port 80)

- **alert tls any any -> 192.168.149.4 !443 (msg:"TLS TRAFFIC on non-TLS HTTP port"; flow:to_client, not_established; classtype:misc-activity; sid:1000004;)**
- Generate an alert for TLS traffic to the Rocky server with IP address 192.168.149.4 on non-standard HTTPs ports (all ports except port 443)

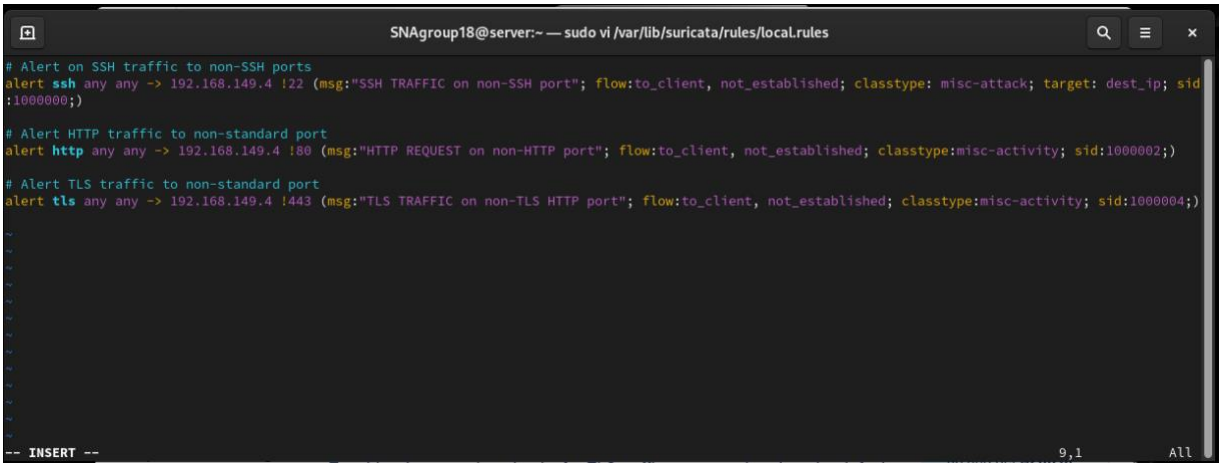


Figure 4.3.3: Set SSH Traffic

Then save and quit the configuration file.

Set Suricata to use the custom rule by editing the configuration file for Suricata using the command below:

- **sudo vi /etc/suricata/suricata.yaml**
- **vi**: terminal-based text editor
- **/etc/suricata/suricata.yaml**: the directory of Suricata configuration file

```
[SNAgroup18@server ~]$ sudo vi /etc/suricata/suricata.yaml
```

Figure 4.3.4: Set Custom Rule

With the configuration file open using “vi”, do the following edit to the file:

- At the “rule-files” section add “local.rules”
- **rule-files:**

  -suricata.rules

  - local.rules
- Set Suricata to load and use both suricata.rules (default rules) and local.rules (custom rules)

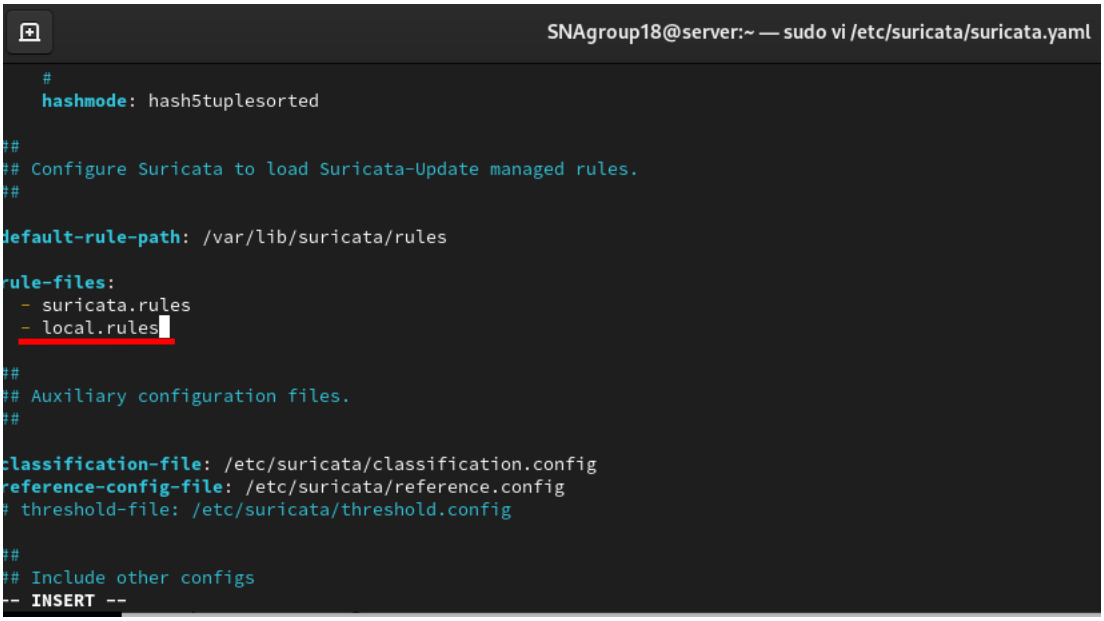


Figure 4.3.5: Adding Local Rules

Then save and quit the configuration file.

Validate changes made to the configuration file and rules of Suricata using the command:

- **sudo suricata -T -c /etc/suricata/suricata.yaml -v**
- **suricata**: initiates Suricata's functionality
- **-T**: run Suricata in test mode
- **-c /etc/suricata/suricata.yaml**: location of Suricata's main configuration file
- **-v**: verbose mode, provide a more detailed information

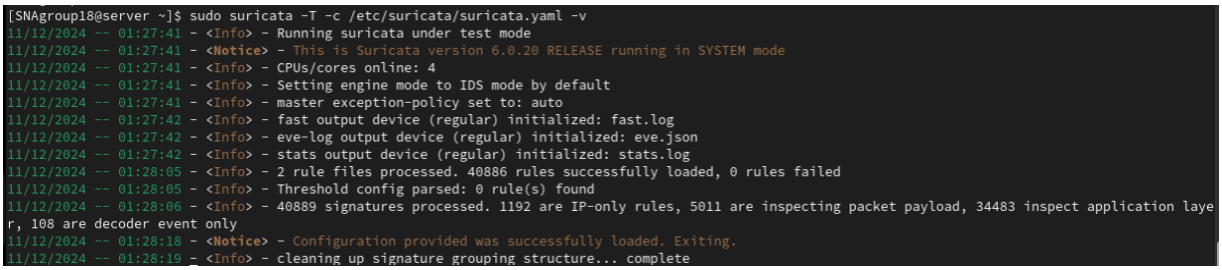


Figure 4.3.6: Validating Suricata rules

There is no error which means that it was successful.

4.4 Configure Suricata as an Intrusion Prevention System (IPS)

Change rules action to “drop” or “reject” and not “alert” so that in IPS mode, Suricata would actively block traffics matching these rules. The “drop” action would drop/remove a packet and any subsequent packets that belong to the network flow while the “reject” action would send a reset packet to both client and server for TCP and ICMP error packet for other protocols.

Edit the local rules file for Suricata using the command below:

- **sudo vi /var/lib/suricata/rules/local.rules**
- **vi**: terminal-based text editor
- **/var/lib/suricata/rules/local.rules**: the directory to define custom Suricata rules

Figure 4.4.1: Editing Suricata Rules File

With the custom rules file open using “vi”, do the following edit to the file:

- **Switch** all “**alert**” to “**drop**”
- Set Suricata to actively block traffics matching the rule instead of just logging it

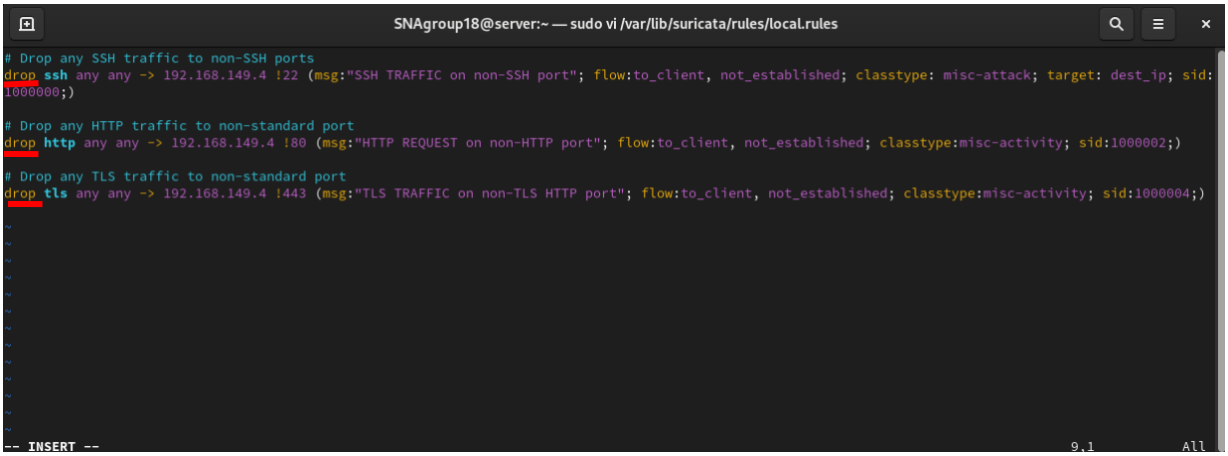


Figure 4.4.2: Set Block Traffics

Note: In this documentation only custom rules in local.rules are switch from “alert” to “drop” because switching all rules (including rules in suricata.rules) risk blocking legitimate access to the network or server. Usually, let Suricata to run for few days or weeks to collect alerts then analyse the entries and only switch the action of relevant rules to drop/reject.

By default, Suricata runs on IDS mode and would not actively block network traffic. Edit the system configuration file for Suricata to enable IPS mode.

- **sudo vi /etc/sysconfig/suricata**
- **vi**: terminal-based text editor
- **/etc/sysconfig/suricata**: the directory of system configuration file for Suricata

Figure 4.4.3: Edit Suricata File

With the configuration file open using “vi”, do the following edit to the file:

- Comment out the current OPTIONS line
- **# OPTIONS=“--af-packet --user suricata”**
- Add a new OPTIONS line
- **OPTIONS=“-q 0 -vvv --user suricata”**
- Sets Suricata to run in IPS mode

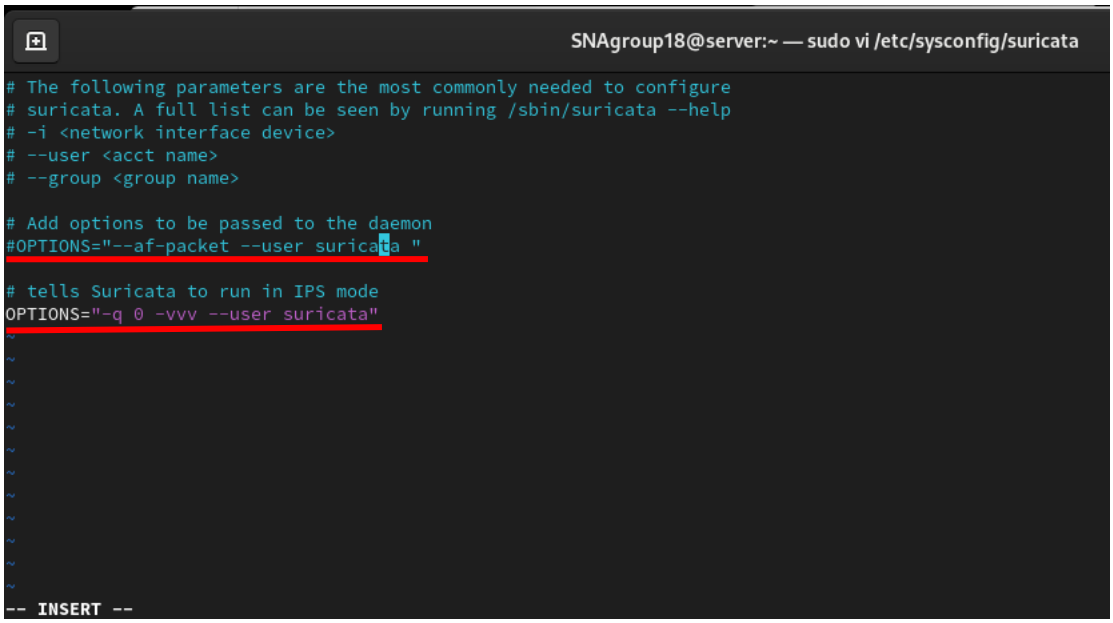


Figure 4.4.4: Set Options Line

Then save and quit the configuration file.

Restart Suricata to apply changes made and check the status using the following:

- **sudo systemctl restart suricata.service**
- **systemctl**: use to manage system and services
- **restart**: to reload (stop and then start) a service

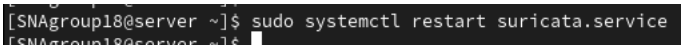


Figure 4.4.5: Restaring Suricata Service

- **sudo systemctl status suricata.service**
- **status**: to display the current status of a service

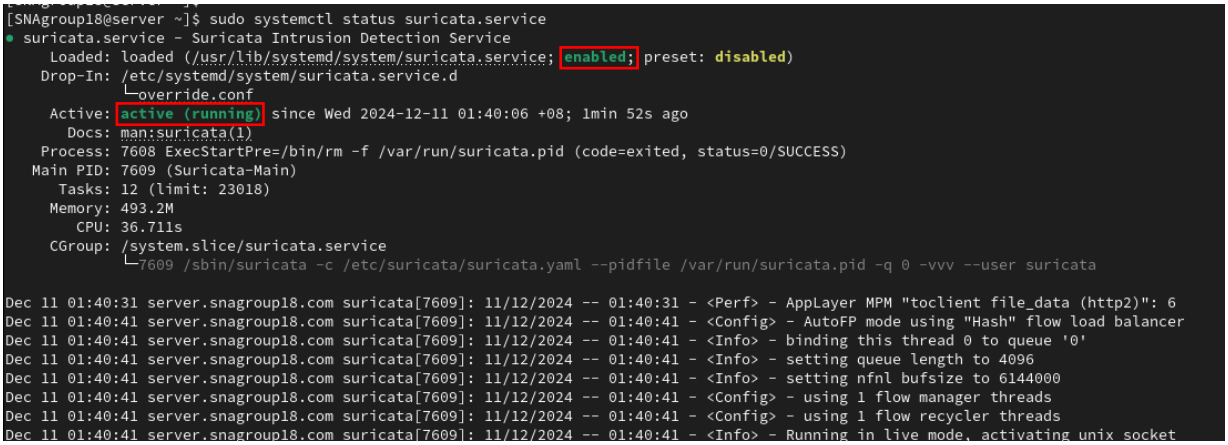


Figure 4.4.6: Checking Suricata Status

The service would start during boot since the service is “enabled” in the “Loaded” option. The service is also currently active and running shown by “active (running)” in the “Active” option.

After Suricata is set to process traffic in IPS mode, configure the firewall to direct incoming packets to Suricata.

Set the firewall to direct incoming and outgoing SSH traffic (port 22) to bypass the Suricata (NFQUEUE) so SSH would still be accessible and not delayed by Suricata.

- **sudo firewall-cmd --permanent --direct --add-rule ipv4 filter INPUT 0 -p tcp --dport 22 -j NFQUEUE --queue-bypass**
- **sudo firewall-cmd --permanent --direct --add-rule ipv4 filter OUTPUT 0 -p tcp --sport 22 -j NFQUEUE --queue-bypass**

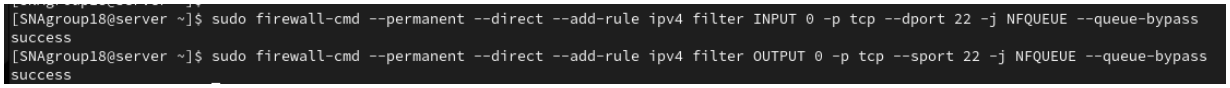


Figure 4.4.7: Set Permanent to Firewall

Set the FORWARD rule to direct all forwarded traffic (traffic that is passing through the server) to Suricata for inspection.

- **sudo firewall-cmd --permanent --direct --add-rule ipv4 filter FORWARD 0 -j NFQUEUE**

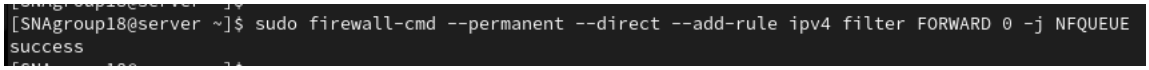


Figure 4.4.8: Set Forward Rule

Set the firewall to direct all remaining incoming and outgoing traffic that is not SSH traffic to Suricata for inspection.

- **sudo firewall-cmd --permanent --direct --add-rule ipv4 filter INPUT 1 -j NFQUEUE**
- **sudo firewall-cmd --permanent --direct --add-rule ipv4 filter OUTPUT 1 -j NFQUEUE**

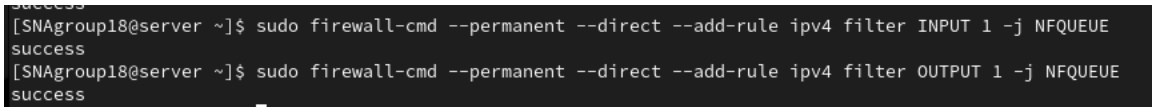


Figure 4.4.9: Set Incoming and Outgoing Traffic

After all firewall rules and configured, reload the firewall rules to ensure that changes are made immediately.

- **sudo firewall-cmd –reload**
- **firewall-cmd**: tool to manage firewall rules on Linux

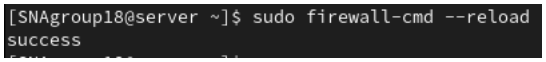


Figure 4.4.10: Reloading Firewall

4.5 Verify Suricata IPS and Test Intrusion Prevention (Active Mode)

Test whether Suricata in IPS mode could detect suspicious traffic, generate an alert log for those events, and block it. Like when testing the IDS, Rule number 2100498 in ET Open ruleset would be used for the testing. Using the same method as well, with the “curl” command to generate a HTTP request to trigger an alert.

Create a custom rule by editing the local rules file for Suricata using the command below:

- **sudo vi /var/lib/suricata/rules/local.rules**
- **vi**: terminal-based text editor
- **/var/lib/suricata/rules/local.rules**: the directory to define custom Suricata rules

```
[SNAgroup18@server ~]$ sudo vi /var/lib/suricata/rules/local.rules
```

Figure 4.5.1: Creating Custom Rule

With the custom rules file open using “vi”, do the following edit to the file:

- Add the Rule number 2100498 from the ET Open ruleset and replace “alert” with “drop”
- **drop ip any any -> any any (msg:"GPL ATTACK_RESPONSE id check returned root"; content:"uid=0|28|root|29|"; classtype:bad-unknown; sid:2100498; rev:7; metadata:created_at 2010_09_23, updated_at 2010_09_23;)**
- Set to block the traffic rather than just logging or create alert

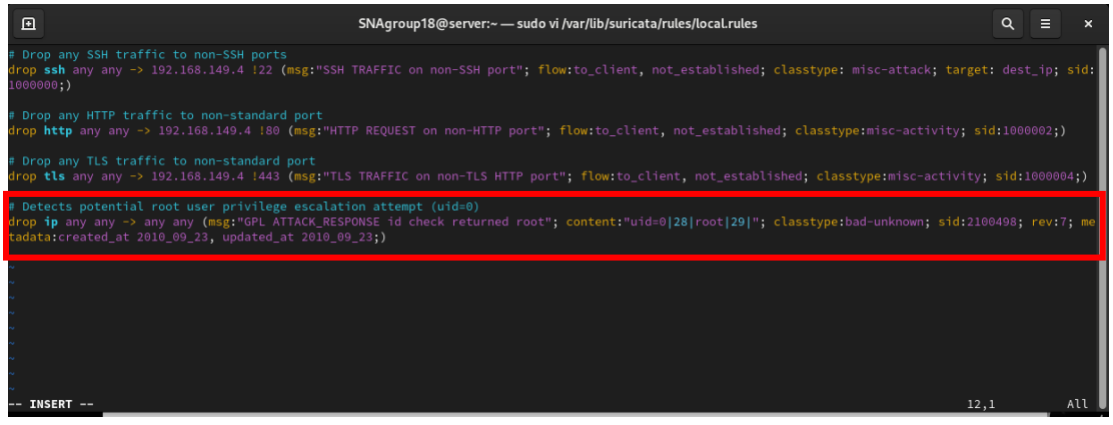


Figure 4.5.2: Adding Rule Number

Then save and quit the configuration file.

Reload Suricata's rulesets to save changes made to the ruleset files using the command:

- **sudo kill -usr2 $(pidof suricata)**
- **kill –usr2**: sends the SIGUSR2 signal
- SIGUSR2 signal instruct Suricata to reload its rules

Figure 4.5.3: Reloading Suricata Changes

Now test the IPS by using "curl" to generate a HTTP request to trigger the rule.

- **curl --max-time 5 http://testmynids.org/uid/index.html**
- **curl**: tool used to send a request to a specified URL
- **--max-time 5**: limit the maximum time for the request to 5 seconds
- **http://testmynids.org/uid/index.html**: the URL to access, should trigger ET Open rule number 2100498

Figure 4.5.4: Testing IPS

The output shows that no data was able to be received from the server within the 5 second time limit. This request timed out indicates Suricata blocked the HTTP response unlike with IDS.

Check the Suricata EVE log for entry with the rule number 2100498 to confirm that Suricata dropped the HTTP response using the following command:

- **jq 'select(.alert .signature_id==2100498)' /var/log/suricata/eve.json**
- **jq**: JSON processor to query JSON data
- **'select(.alert .signature_id==2100498)'**: select and return JSON entries where the "signature_id" in the "alert" object is equal to 2100498, the rule number
- **/var/log/suricata/eve.json**: the EVE JSON log file where Suricata writes alerts

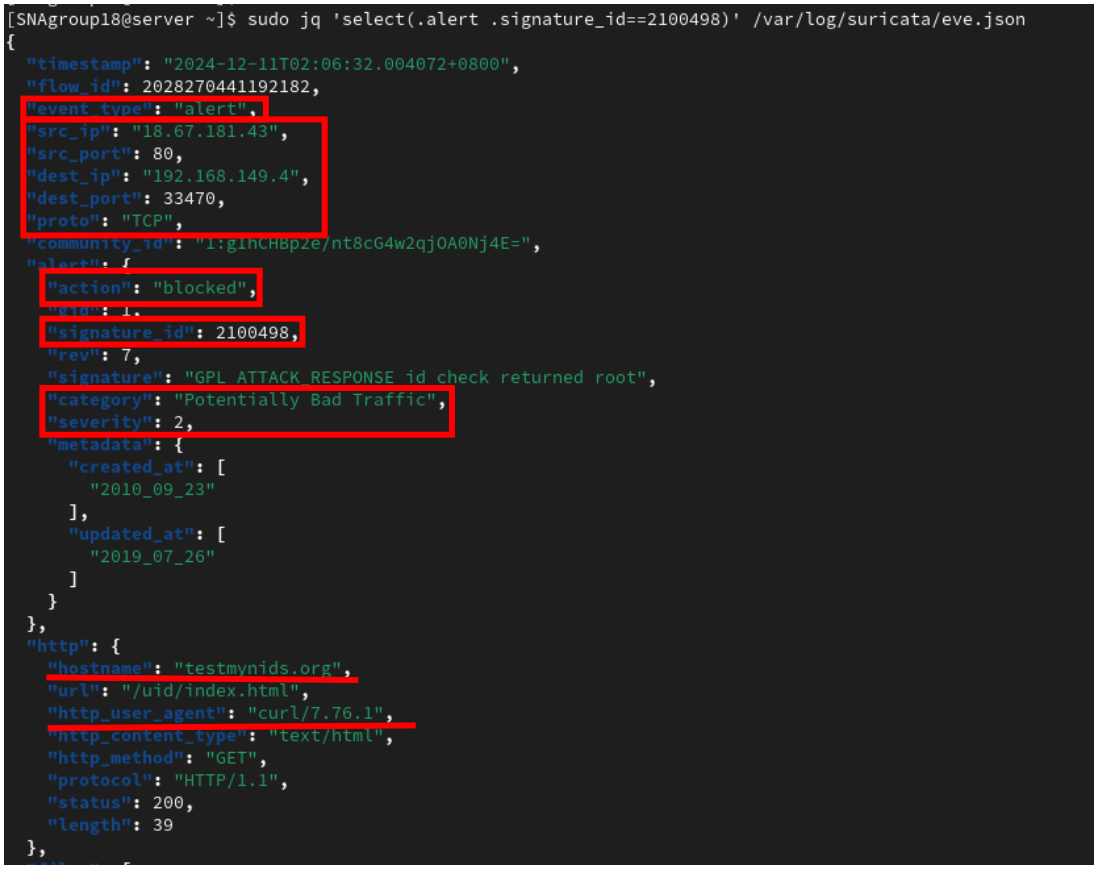


Figure 4.5.5: Checking Log Entry

These are the important findings from the log entry. The event is an alert, which is when Suricata detects a network activity that matches the rule. The source IP address is 18.67.181.43 with the port 80 and the destination IP address is 192.168.149.4 (the IP address of the Rocky machine) on the port 33470, and it used TCP. After detecting this Suricata “blocked” the connection which shows that the IPS was working. The rule number is 2100498. The category is “Potentially Bad Traffic”, and the severity of the alert is 2. It was from the hostname “testmynids.org” which was the URL that the request was sent to from the user agent “curl”.

5.0 References

Brad. (2022, March 16). *How to Create Your Own SSL Certificate Authority for Local HTTPS Development*. Delicious Brains. https://deliciousbrains.com/ssl-certificate-authority-for-local-https-development/

Cloud infrastructure for developers. (n.d.). DigitalOcean. https://www.digitalocean.com/

Cruz, J. D. (2021, December 13). Installing the Apache Server and Creating a Sample HTML Page. *Medium*. https://medium.com/@purplejen/installing-the-apache-server-and-creating-a-sample-html-page-33b1938bf0b8

Fajardini, J. (2024, April 23). *What is Suricata*. Suricata. https://suricata.io/2024/04/15/what-is-suricata/

Gomez, J. (2023, March 17). *How to install apache on Rocky Linux 9*. LinuxTeck. https://www.linuxteck.com/how-to-install-apache-on-rocky-linux/

Informix Servers 14.10. (n.d.). https://www.ibm.com/docs/en/informix-servers/14.10?topic=openssl-setting-up-ca

OISF. (n.d.). *Quickstart guide — Suricata 8.0.0-dev documentation*. Suricata. https://docs.suricata.io/en/latest/quickstart.html#alerting

Red Hat, Inc. (n.d.). *Red Hat Product Documentation*. Red Hat Enterprise Linux. https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/9

Sah, J. (2024, September 25). *How to Install Apache Webserver on Rocky Linux 9*. Retrieved from docs.vultr.com: https://docs.vultr.com/how-to-install-apache-webserver-on-rocky-linux-9

Setting Up a Rocky Linux 9 Web Server – Answertopia. (n.d.). https://www.answertopia.com/rocky-linux/setting-up-a-rocky-linux-web-server/