Project Title: Security DMZ(Demilitarized Zone)

Duration: 01/08/2025 – 27/08/2025

Organization: Sunbeam Institute Of Information Technology

Submitted by: Neha Sanjay Kumbharde

Under the guidance of

Mr. Gajanan Taur Sir

DECLARATION

We declare that this written submission represents our ideas in our own words and where others ideas or words have been included; we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed**.**

Place: Pune

Date:

Neha Sanjay Kumbharde (**93886**)

CERTIFICATE

This is to certify that the project report entitled "Security DMZ(Demilitarized Zone)", submitted by Neha Kumbharde **is the bonafide work completed under our supervision and guidance in partial fulfillment for the Internship Program of Sunbeam Institute of Information Technology, Pune**

Place: Pune

Date:

Technologies Used and Setup Overview:

1. Database Server – MySQL/MariaDB:

The database server was set up by installing the required database software from the package manager.

A new database schema was created for the project, and user access privileges were assigned so only the web server could connect.

This ensured secure storage of application data and restricted access.

2. Web Server – Apache2 + Flask:

A web server environment was prepared using Apache2 to handle HTTP requests.

Flask, a lightweight Python framework, was added to run the application logic.

The application was then cloned from the project repository and deployed on the server.

Configuration changes were made so that the application could connect to the database server and be accessed from client browsers.

3. SSL/TLS Certificates:

A domain name was mapped to the web server using local DNS/hosts file entries.

A self-signed SSL certificate was generated and applied to the web server.

This enabled secure HTTPS communication between users and the application, protecting sensitive data during transmission.

4. Mail Server – Postfix + Dovecot:

A mail server environment was created using Postfix for sending emails and Dovecot for receiving and storing them.

The mail server was configured with the organization's local domain.

User accounts were created to allow testing of sending and receiving emails.

This provided a functional internal email system for the project environment.

5. **Webmail – Roundcube:**

Roundcube was set up as a webmail client so that emails could be accessed through a web browser.

The interface was connected to the mail server's database and configured with the local domain.

This allowed users to log in with their mail IDs and send or receive emails directly through the browser.

6. **Monitoring – Nagios XI:**

A dedicated monitoring server was installed and configured with Nagios XI.

Web and mail servers were added as monitored hosts.

Key services like HTTP, HTTPS, SMTP, IMAP, and system resources (CPU, memory) were tracked.

Notification alerts were enabled through the mail server, ensuring that administrators received warnings if any service failed.

7. **Intrusion Detection – Snort:**

Snort was deployed on the web server as a network intrusion detection system.

It was configured to monitor incoming traffic and identify suspicious patterns such as SYN floods, ICMP floods, and abnormal packets.

Custom rules were added to detect common attack types, allowing early identification of potential threats.

8. **System Hardening – Fail2ban:**

Fail2ban was installed on all servers to prevent brute-force login attempts.

The system monitored failed login attempts on services like SSH.

Malicious IP addresses were automatically blocked for a period of time after repeated failures.

This provided an extra layer of protection against unauthorized access.

9. **System Hardening – iptables:**

The database server was further secured using firewall rules.

Only the web server's IP address was permitted to connect to the database service.

All other connections to the database port were denied, reducing the attack surface.

10. **Firewall – pfSense:**

A pfSense firewall was deployed on a separate virtual machine to control network traffic.

Firewall rules were defined to allow only the necessary services:

HTTP/HTTPS access to the web server

Mail services (SMTP, IMAP, POP3) to the mail server

SSH access restricted to administrative systems only

This created a strong network perimeter, ensuring only trusted traffic could pass through.

Day 02 (04/08/2025) – **Setup of Database & Web Server**

Objective:

- **Configure and setup two machines:**

  Database Server

  Web Server

- **Clone application from GitHub and deploy.**

Implementation:

1. **Database Server:**
   - Installed MySQL/MariaDB database server.
   - Created database and imported schema provided by the project.
   - Configured user privileges to allow only web server access.

     CREATE DATABASE kartvyapath;

     CREATE USER 'kartvya'@'webserver_ip' IDENTIFIED BY 'StrongPassword';

     GRANT ALL PRIVILEGES ON kartvyapath. TO 'kartvya'@'webserver_ip';*

     FLUSH PRIVILEGES;

2. **Web Server:**
   - Installed Apache2 + Flask with mod_wsgi.
   - Cloned application:

     git clone https://github.com/sunbeam-ditiss/KartvyaPath-Project.git
   - Configured virtual host for Flask app.
   - Updated DB connection settings in config.py to point to Database server.

3. **Verification:**

- Accessed application via http://webserver_ip/ → Application running successfully with DB connectivity.

Day 03 (05/08/2025) – SSL/TLS Configuration

Objective:

- Access Flask application on https://todo.sunbeam.local.
- Create and configure SSL certificates.

Implementation:

1. **Configured DNS/local hosts file:**

   192.168.x.x todo.sunbeam.local

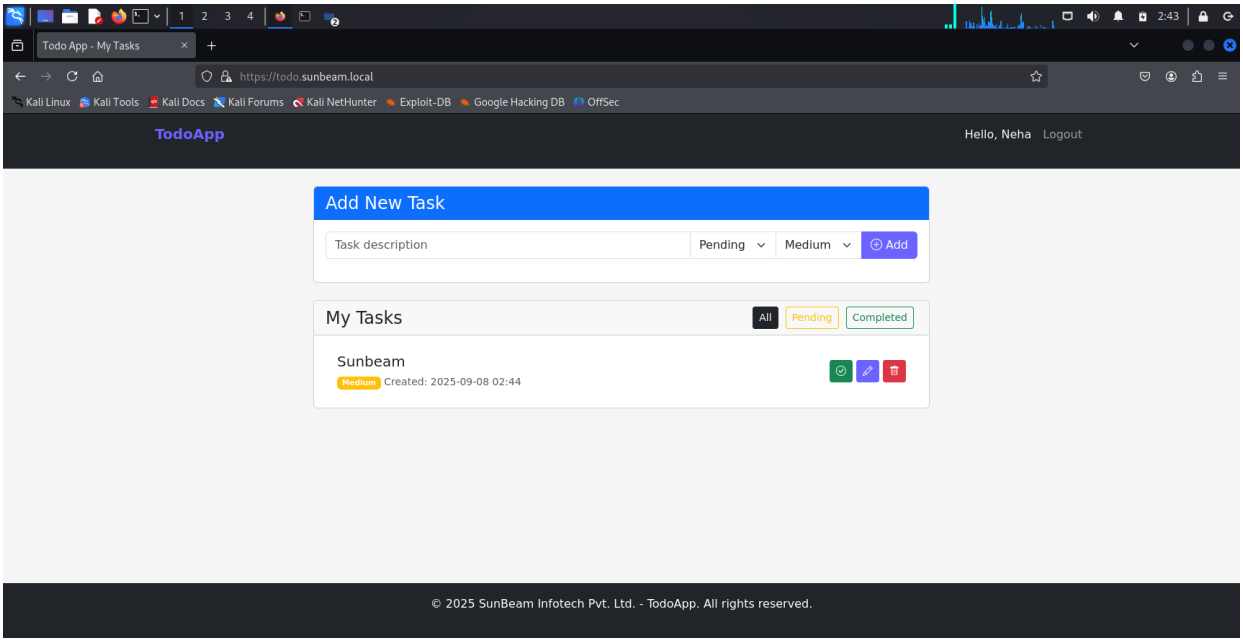
2. **Generated self-signed certificate:**

   openssl req -newkey rsa:2048 -nodes -keyout todo.key -x509 -days 365 -out todo.crt

3. **Configured Apache2 VirtualHost with SSL:**

```
<VirtualHost *:443>
    ServerName todo.sunbeam.local
    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/todo.crt
    SSLCertificateKeyFile /etc/ssl/private/todo.key
    WSGIScriptAlias / /var/www/todo/wsgi.py
</VirtualHost>
```

4. **Restarted Apache2 → Application accessible at**: https://todo.sunbeam.local.



Day 04 (06/08/2025) – Mail Server Setup

Objective:

- Install and configure a mail server with domain sunbeam.local.
- Create 2 user accounts for testing.

Implementation:

- Installed Postfix (SMTP) + Dovecot (IMAP/POP3).
- Configured /etc/postfix/main.cf with domain:

myhostname = mail.sunbeam.local

mydomain = sunbeam.local

myorigin = /etc/mailname

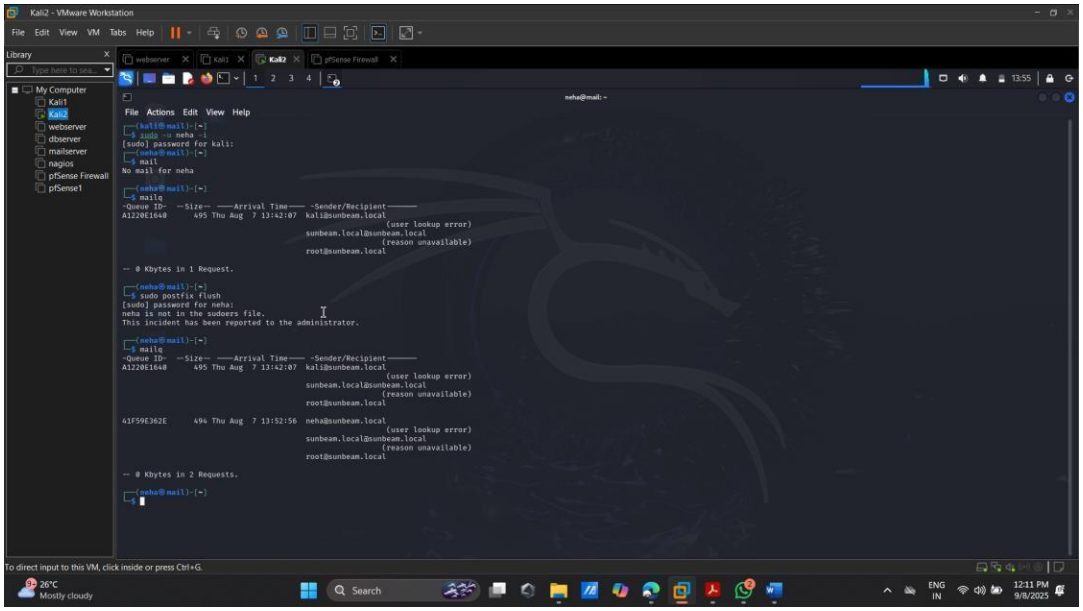
inet_interfaces = all

- Added user accounts:

sudo adduser user1

sudo adduser user2

- Tested email delivery with telnet localhost 25 and IMAP using Thunderbird.



<u>Day 05 (07/08/2025)</u> – Webmail Configuration

Objective:

- Setup Webmail for Mail Server.
- Access via http://mail.sunbeam.local.

Implementation:

- Installed Roundcube Webmail on Apache2.
- Configured DB for Roundcube.
- Updated /etc/apache2/sites-available/mail.conf:

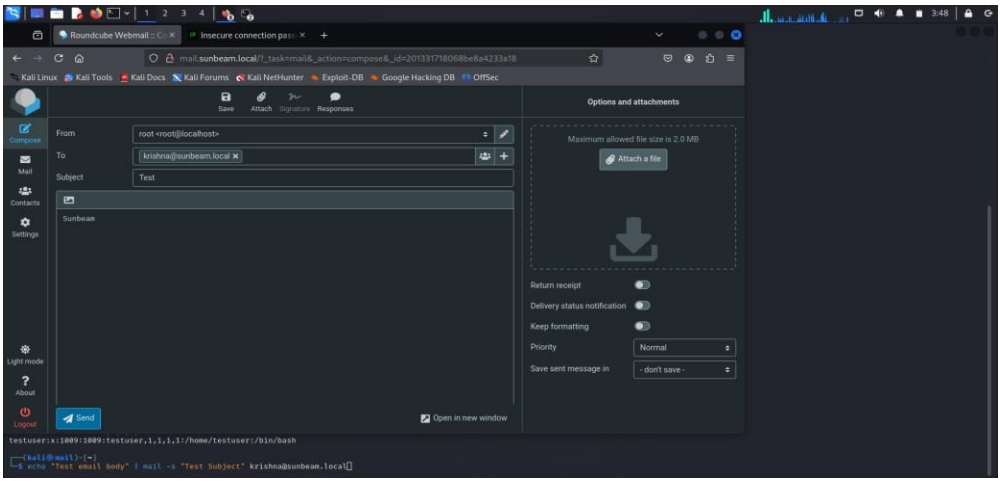
*<VirtualHost *:80>*

ServerName mail.sunbeam.local

DocumentRoot /var/www/roundcube

</VirtualHost>

- Verified access via browser → Login with user1@sunbeam.local.



Day 06 (08/08/2025) – Pending Work Completion

- Reviewed configurations of DB server, Web server, Mail server.
- Verified application, SSL, and mail functionality.

Day 07 (11/08/2025) – Monitoring with Nagios XI

Objective:

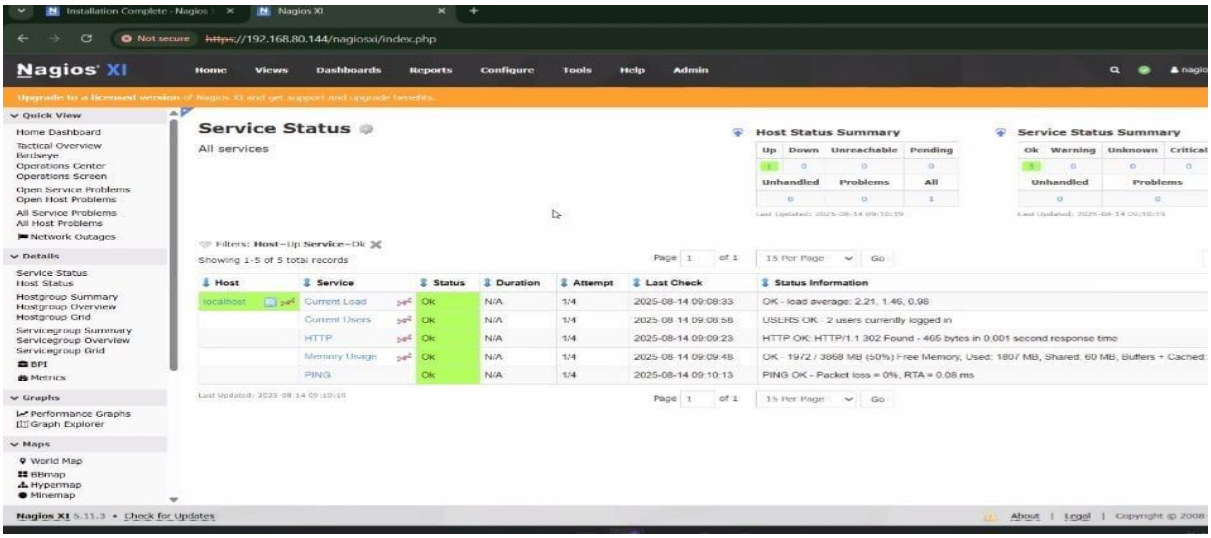
- Monitor services on Web & Mail server.
- Configure alerts via mail server.

Implementation:

1. **Installed Nagios XI on a monitoring server.**
2. **Added hosts and services:**

- **Web Server** → HTTP, HTTPS, Apache2, CPU, Memory.
- **Mail Server** → SMTP, IMAP, POP3, CPU, Memory.

3. **Configured Postfix mail relay on Nagios to send alerts to @sunbeam.local.**
4. **Verified by stopping Apache2 service → Alert mail received.**



Day 09 (13/08/2025) – Intrusion Detection with Snort

Objective:

- Configure Snort on Web Server.
- Write custom IDS rules.

Rules Implemented:

1. **SYN Flood Detection**

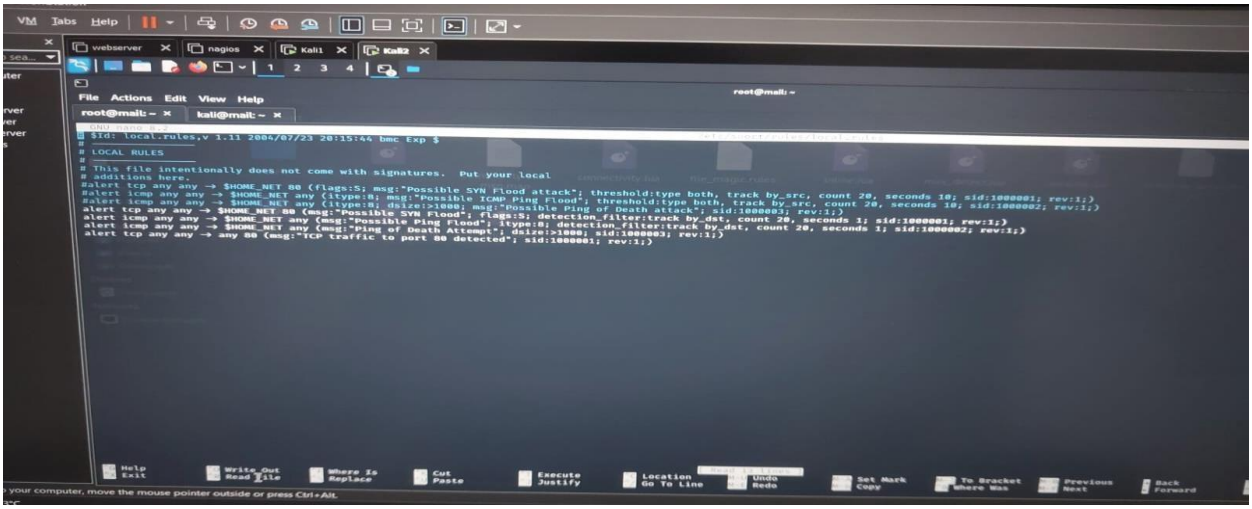
alert tcp any any -> $HOME_NET 80 (flags:S; msg:"SYN Flood detected"; threshold:type both, track by_src, count 20, seconds 1; sid:1000001;)

2. **Ping Flood**

alert icmp any any -> $HOME_NET any (msg:"ICMP Ping Flood detected"; dsize:0; threshold:type both, track by_src, count 50, seconds 2; sid:1000002;)

3. **Ping of Death**

alert icmp any any -> $HOME_NET any (msg:"Ping of Death detected"; dsize:>1500; sid:1000003;)



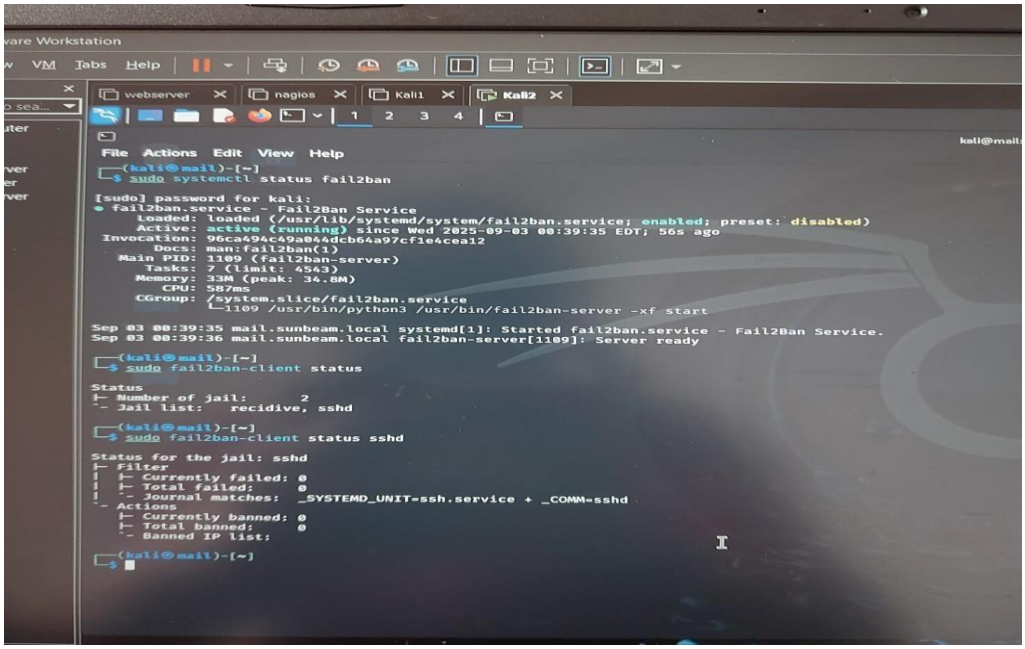
Day 11 (18/08/2025) – System Hardening

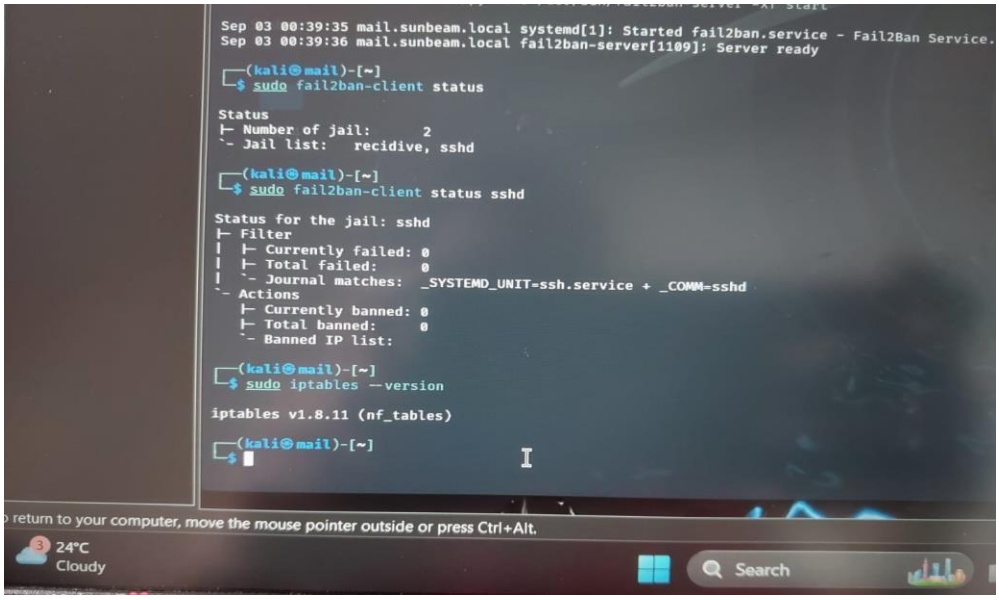
Objective:

- Protect all servers with fail2ban.
- Secure DB server using iptables.
- Install and configure pfSense firewall.

Implementation:

1. **Fail2ban (SSH Protection)**
   - Installed fail2ban on all servers.
   - Enabled jail for SSH (/etc/fail2ban/jail.local).



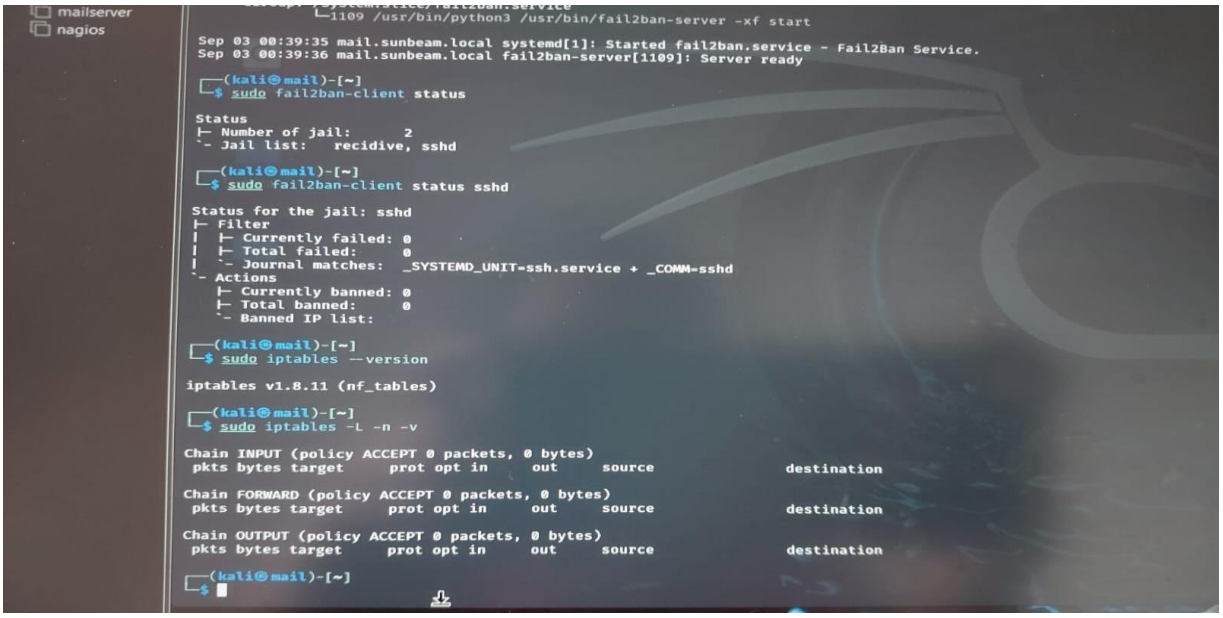


2. DB Server iptables Rules (Allow only Web Server access):

- iptables -A INPUT -p tcp -s webserver_ip --dport 3306 -j ACCEPT
- iptables -A INPUT -p tcp --dport 3306 -j DROP

iptables -A INPUT -p tcp -s webserver_ip --dport 3306 -j ACCEPT

iptables -A INPUT -p tcp --dport 3306 -j DROP



3. pfSense Firewall

- Installed pfSense on separate VM.
- Configured firewall rules:

    1. Allow HTTP/HTTPS to Web Server.
    2. Allow SMTP/IMAP/POP3 to Mail Server.
    3. Allow SSH only from admin machine.

Conclusion:

- Successfully deployed a multi-server environment with Web, Database, and Mail servers.
- Secured application with SSL/TLS certificates.
- Configured monitoring (Nagios XI) and intrusion detection (Snort).
- Implemented system hardening with fail2ban, iptables, and pfSense.
- Entire setup provides a secure, monitored, and resilient infrastructure for sunbeam.local.