## Part 1: Virtual Interface Configuration

**1. Create Virtual Interfaces on Kali Linux**

To create two virtual interfaces, we will first configure network interfaces using ifconfig or ip commands, and then assign them static IP addresses within the same subnet as the original interface.

**Step 1: Check Current Network Interfaces**

Start by checking your current network interfaces:

ip a

**Step 2: Create Virtual Interfaces**

To create the first virtual interface ( eth0:1), run:

sudo ip link add link eth0 name eth0:1

sudo ip addr add 192.168.43.101/24

sudo ip link set eth0:1 up

For the second virtual interface (eth0:2):

sudo ip link add link eth0 name eth0:2

sudo ip addr add 192.168.43.102/24

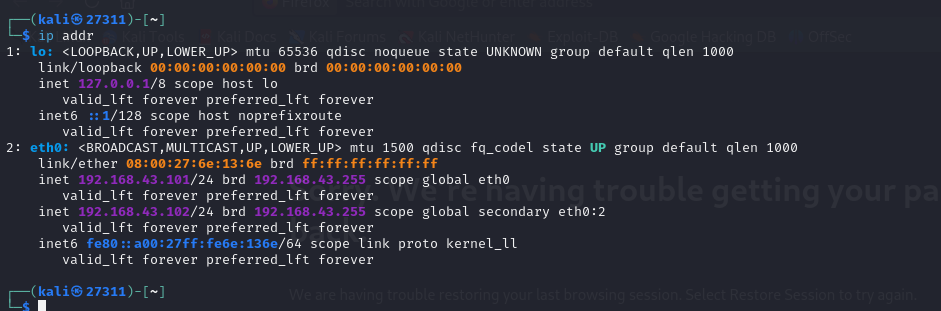
sudo ip link set eth0:2 up

**Step 3: Verify Virtual Interfaces**

Check the new virtual interfaces with:

ip a

You should see eth0:1 and eth0:2 with the respective IP addresses.



**Step 4: Ensure Connectivity**

Verify that both virtual interfaces are reachable from other machines or VMs on the same network by using the ping command:

ping 192.168.43.101

ping 192.168.43.102

**2. Static IP Configuration (Optional)**

To make these virtual interfaces persist after a reboot, you can configure static IPs in the /etc/network/interfaces file.

I Edited the network interfaces file :

sudo nano /etc/network/interfaces

Add the following:

/etc/network/interfaces

# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/\*

# Loopback interface (local system use only)

auto lo

iface lo inet loopback

# Main network interface (bridged to Wi-Fi, gets IP automatically)

auto eth0

iface eth0 inet static

address 192.168.43.101

netmask 255.255.255.0

gateway 192.168.43.1

dns-nameservers 8.8.8.8 1.1.1.1

# First virtual interface - for AUCA web server

auto eth0:1

iface eth0:1 inet static

address 192.168.43.101

netmask 255.255.255.0

# Second virtual interface - for portfolio system

auto eth0:2

iface eth0:2 inet static

address 192.168.43.102

netmask 255.255.255.0

**Restart the network service:**

sudo systemctl restart networking

### Apache Setup for AUCA Education Web Page (27311.auca.ac.rw)

#### 1. Install Apache and Create Directory

sudo apt update

sudo apt install apache2 -y

sudo mkdir -p /var/www/auca/registration.auca.ac.rw

#### 2. Add a Static Web Page

To simulate a real institutional web presence, I utilized the wget tool to mirror the AUCA Registration webpage. This approach ensured an authentic and complete static copy of the official AUCA content, preserving design, structure, and usability. By using the following command, all required resources such as images, stylesheets, and links were recursively fetched and adapted for local hosting:

wget --mirror --convert-links --adjust-extension --page-requisites --no-parent https://registration.auca.ac.rw/ -P /var/www/auca/registration.auca.ac.rw

This allowed the Apache server to deliver a realistic, navigable AUCA Education web page on the configured virtual interface.

#### 3. Create SSL Certificates

sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \

-keyout /etc/apache2/ssl/auca.key \

-out /etc/apache2/ssl/auca.crt

#### 4. Create Virtual Host Config File

sudo nano /etc/apache2/sites-available/27311.auca.ac.rw.conf

Add:

<VirtualHost 192.168.43.101:8443>

ServerName 27311.auca.ac.rw

DocumentRoot /var/www/auca/registration.auca.ac.rw

SSLEngine on

SSLCertificateFile /etc/apache2/ssl/auca.crt

SSLCertificateKeyFile /etc/apache2/ssl/auca.key

<Directory /var/www/auca/registration.auca.ac.rw>

Options -Indexes +FollowSymLinks

AllowOverride All

</Directory>

ErrorLog ${APACHE\_LOG\_DIR}/auca\_ssl\_error.log

CustomLog ${APACHE\_LOG\_DIR}/auca\_ssl\_access.log combined

</VirtualHost>

#### 5. Enable Modules and Site

sudo a2enmod ssl

sudo a2ensite 27311.auca.ac.rw.conf

sudo systemctl restart apache2

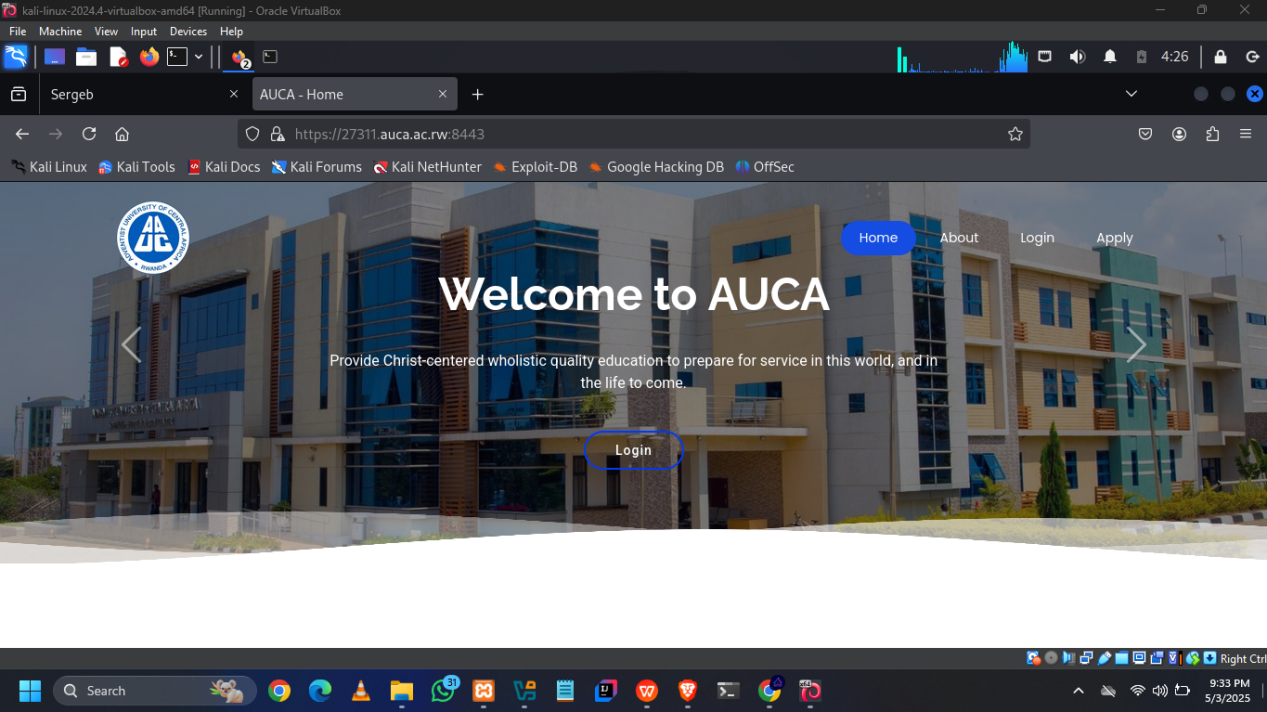
#### 6. Add to /etc/hosts

sudo nano /etc/hosts

Add:

192.168.43.101 27311.auca.ac.rw

Test: <https://27311.auca.ac.rw:8443>



### Nginx Setup for Dynamic Portfolio (portfolio.auca.ac.rw)

#### 1. Install Nginx and PHP-FPM

sudo apt install nginx php php-fpm -y

#### 2. Set Up Portfolio Directory

sudo mkdir -p /var/www/portfolio.auca.ac.rw

Created index.html, contact.php in DIrectory

#### 3. Use MySQL to Store Profile Data

Create a portfolio database and manually insert:

CREATE DATABASE contact\_form\_db;

USE contact\_form\_db;

CREATE TABLE contacts (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL,

message TEXT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

is\_read BOOLEAN DEFAULT FALSE

);

#### 4. Configure Nginx HTTPS Block

sudo nano /etc/nginx/sites-available/portfolio.auca.ac.rw

server {

listen 192.168.43.102:443 ssl http2;

server\_name portfolio.auca.ac.rw;

ssl\_certificate /etc/apache2/ssl/auca.crt;

ssl\_certificate\_key /etc/apache2/ssl/auca.key;

root /var/www/portfolio.auca.ac.rw;

index index.html contact.php;

location / {

try\_files $uri $uri/ =404;

}

location ~ \.php$ {

include snippets/fastcgi-php.conf;

fastcgi\_pass unix:/var/run/php/php7.4-fpm.sock;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

include fastcgi\_params;

}

location ~ /\.ht {

deny all;

}

error\_page 404 /404.html;

}

server {

listen 192.168.43.102:80;

server\_name portfolio.auca.ac.rw;

return 301 https://$server\_name$request\_uri;

}

#### 5. Enable and Test

sudo ln -s /etc/nginx/sites-available/portfolio.auca.ac.rw /etc/nginx/sites-enabled/

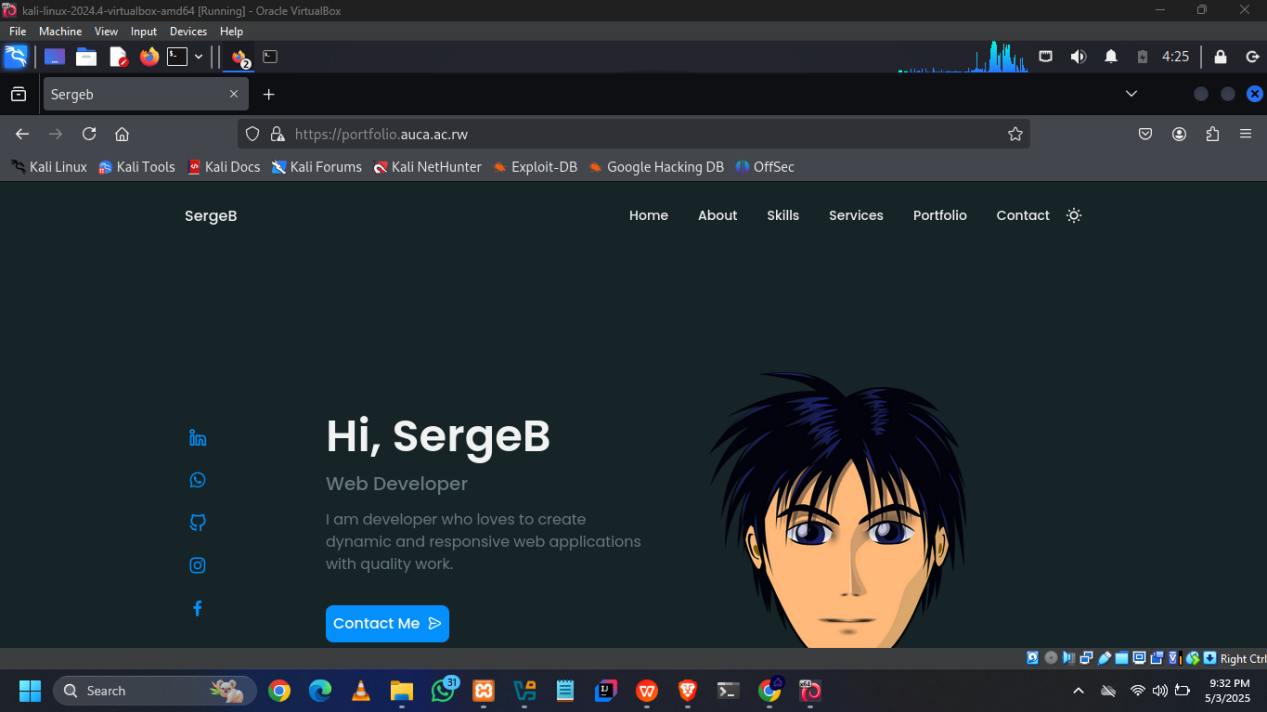
sudo nginx -t

sudo systemctl restart nginx

#### 6. Add to /etc/hosts

192.168.43.102 portfolio.auca.ac.rw

Test: <https://portfolio.auca.ac.rw>



# Implementation Using ModSecurity and OWASP-CRS

In modern web architecture, securing web applications from common vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), and Local File Inclusion is essential. This documentation outlines the step-by-step configuration and deployment of a Web Application Firewall (WAF) using **ModSecurity** and the **OWASP Core Rule Set (CRS)**. The setup applies to the web server project developed in this course and is deployed on an Ubuntu system using Apache2.

## 2. Installing ModSecurity

### Step 1: Install ModSecurity for Apache

sudo apt install libapache2-mod-security2 -y

### Step 2: Enable Apache headers Module

sudo a2enmod headers

### Step 3: Restart Apache2

sudo systemctl restart apache2

## 3. Configuring ModSecurity

### Step 1: Prepare Configuration File

Copy the default configuration:

sudo cp /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf

### Step 2: Edit Configuration

Open the file:

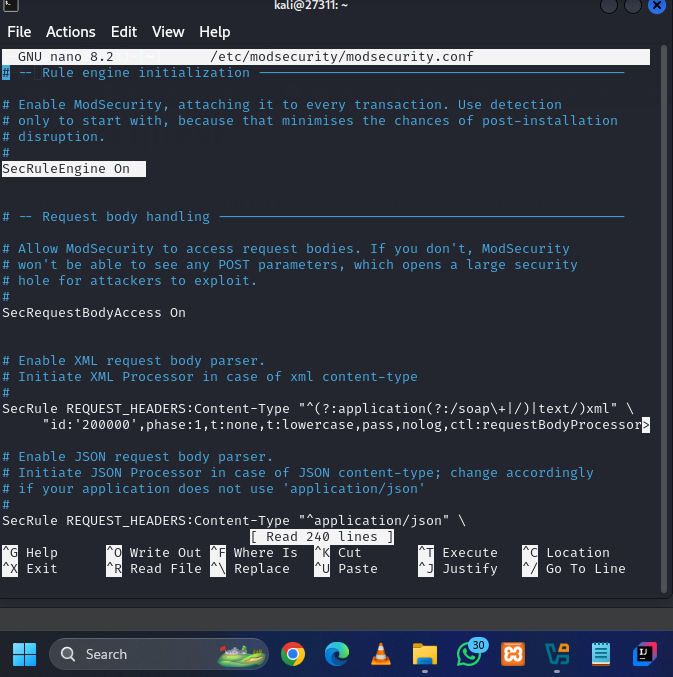
sudo nano /etc/modsecurity/modsecurity.conf

Change:

SecRuleEngine DetectionOnly

To:

SecRuleEngine On



### Step 3: Restart Apache

sudo systemctl restart apache2

## 4. Setting Up OWASP ModSecurity Core Rule Set (CRS)

### Step 1: Remove Default Rules

sudo rm -rf /usr/share/modsecurity-crs

### Step 2: Clone OWASP CRS

sudo git clone https://github.com/coreruleset/coreruleset /usr/share/modsecurity-crs

### Step 3: Rename Configuration Files

sudo mv /usr/share/modsecurity-crs/crs-setup.conf.example /usr/share/modsecurity-crs/crs-setup.conf

## 4. Enabling OWASP-CRS in Apache Configuration

### Step 1: Modify security2.conf

sudo nano /etc/apache2/mods-available/security2.conf

Append:

Include /usr/share/modsecurity-crs/crs-setup.conf

Include /usr/share/modsecurity-crs/rules/\*.conf



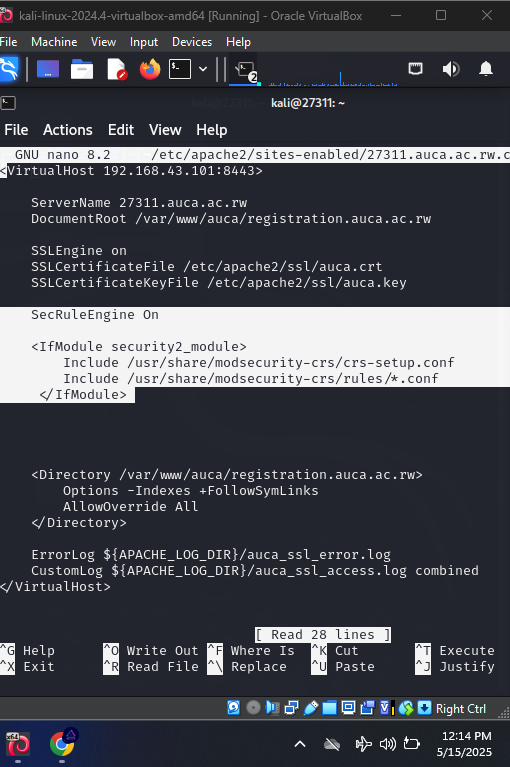
### Step 2: Configure Virtual Host

Edit site configuration,

sudo nano /etc/apache2/sites-enabled/27311.auca.ac.rw.conf

Inside the <VirtualHost> block, add:

SecRuleEngine On



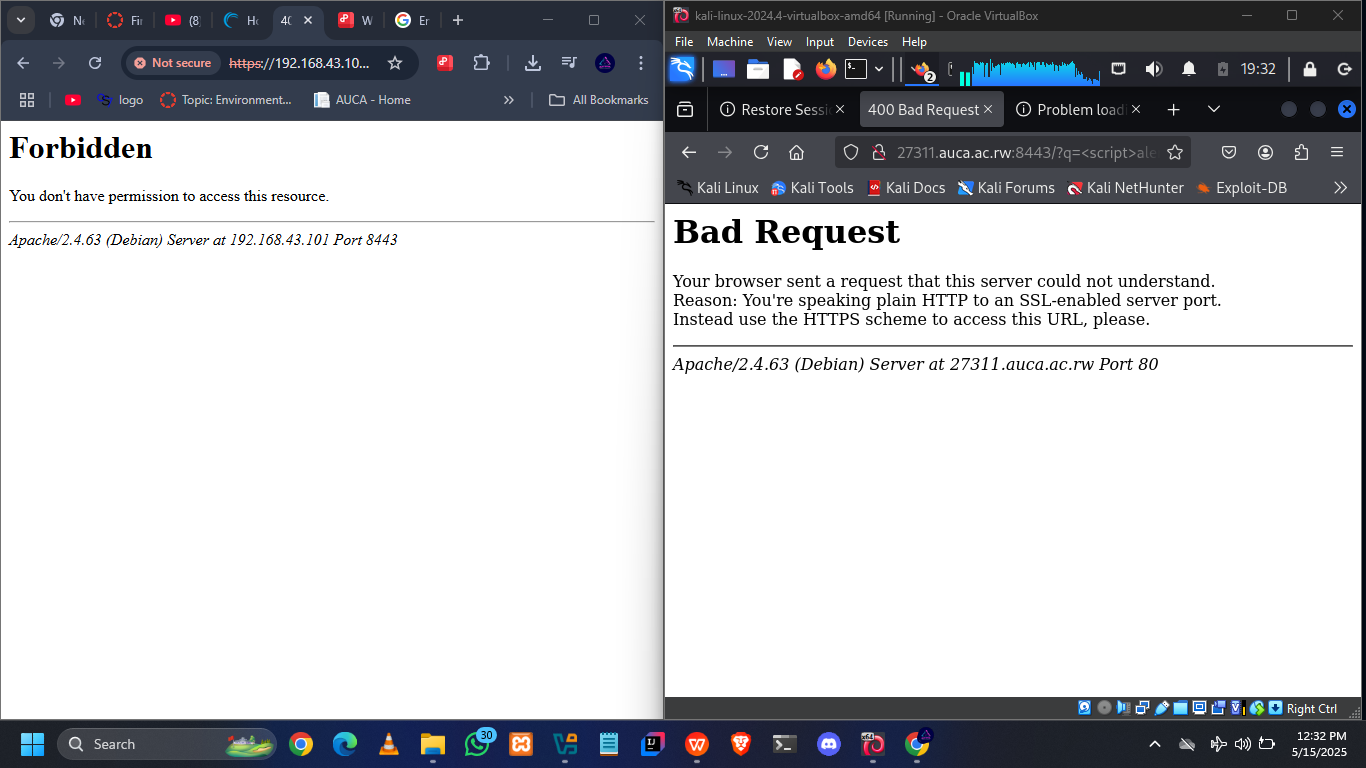
## 6. Final Step: Restart Apache

sudo systemctl restart apache2

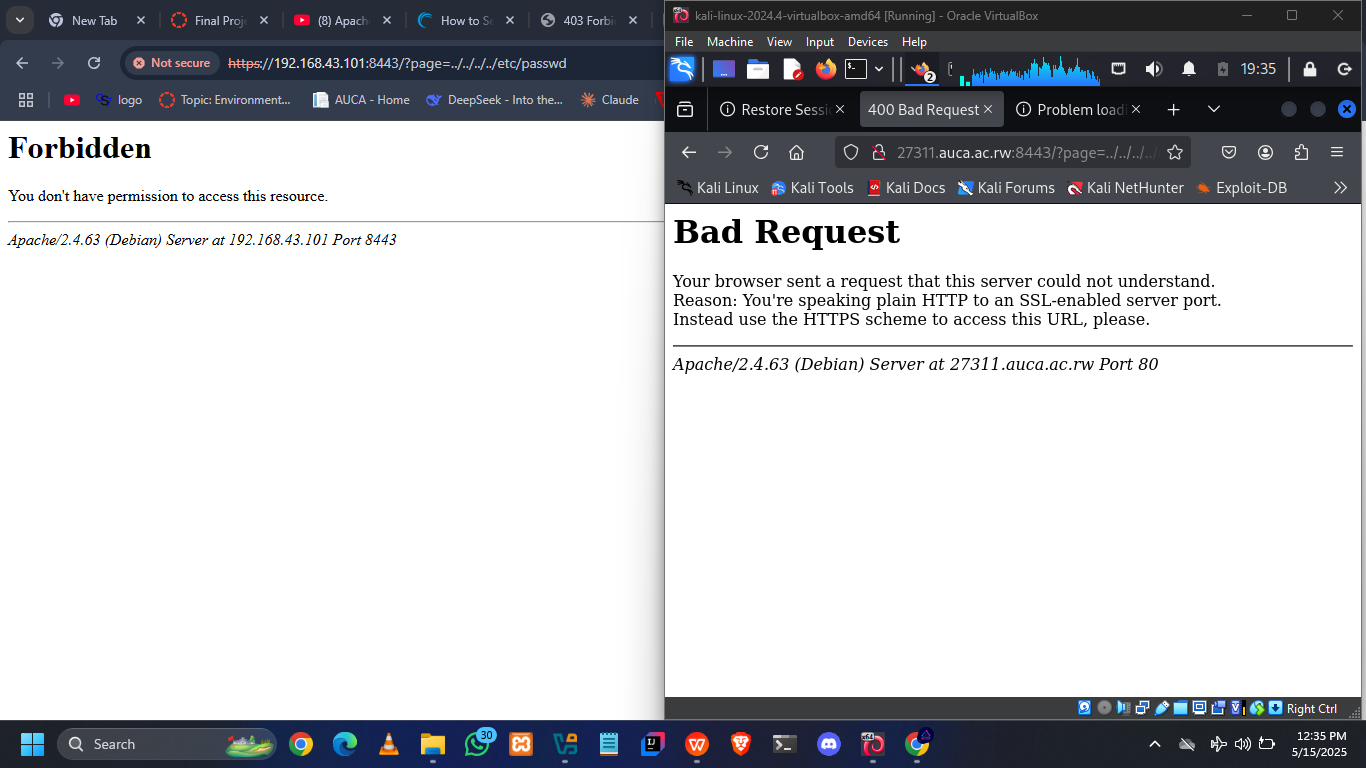
## 7. Testing ModSecurity

To test the WAF, attempt to access a URL with suspicious query parameters, such as:

<https://27311.auca.ac.rw:8443/?query=<script>alert('xss')</script>>



<https://27311.auca.ac.rw:8443/?page=../../../../etc/passwd>



**Deployment and Configuration of NGINX with ModSecurity and OWASP Core Rule Set**

## ****Introduction****

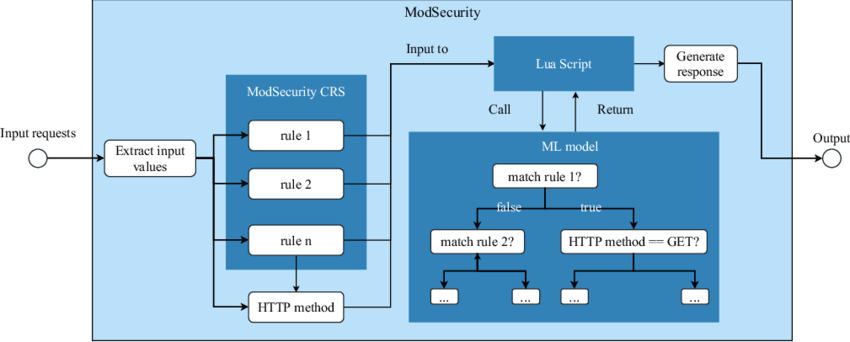
In the modern cybersecurity landscape, web applications are constantly targeted by malicious actors. To mitigate these threats, Web Application Firewalls (WAFs) like **ModSecurity** can be integrated with high-performance web servers like **NGINX**, enhanced further using the **OWASP Core Rule Set (CRS)**. This report demonstrates a full setup of an NGINX-based reverse proxy integrated with ModSecurity and OWASP CRS to provide real-time attack detection and mitigation.

## ****Objective****

To build a secure, production-ready reverse proxy server using **NGINX**, configured with **ModSecurity**  and the **OWASP Core Rule Set** to act as a Web Application Firewall.

## ****System Requirements****

| **Component** | **Version/Spec** |
| --- | --- |
| OS | Ubuntu 20.04 / 22.04 LTS,LINUX DISTROS, |
| Web Server | NGINX 1.24.0 |
| WAF | ModSecurity v3.x |
| Rule Set | OWASP CRS v4.x |
| SSL | Self-signed or Let's Encrypt |
| CPU/RAM | 2 Cores, 2GB+ RAM |



## ****Setup Steps****

### Install Prerequisites

sudo apt update

sudo apt install -y git gcc make libpcre3 libpcre3-dev zlib1g zlib1g-dev \

libssl-dev libxml2 libxml2-dev libyajl-dev apache2-dev pkgconf build-essential

### Build and Install ModSecurity

cd /opt

git clone --depth 1 -b v3/master https://github.com/SpiderLabs/ModSecuritycd ModSecurity

git submodule init && git submodule update

./build.sh && ./configure

make && sudo make install

### 5.3 Compile NGINX with ModSecurity-nginx Module

cd /opt

git clone https://github.com/SpiderLabs/ModSecurity-nginx.git

wget http://nginx.org/download/nginx-1.24.0.tar.gz

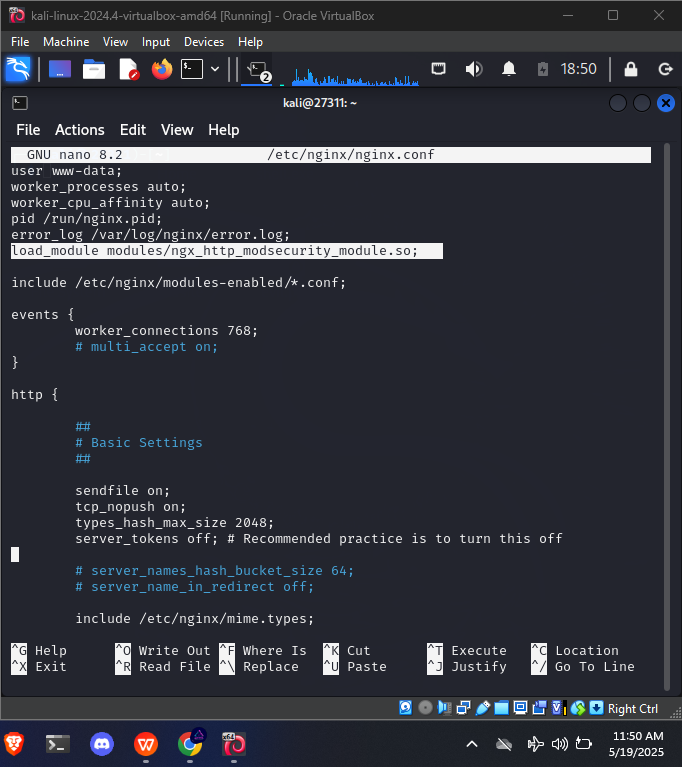
tar -xzvf nginx-1.24.0.tar.gzcd nginx-1.24.0

./configure --with-compat --add-dynamic-module=../ModSecurity-nginx

make modules

sudo cp objs/ngx\_http\_modsecurity\_module.so /etc/nginx/modules/

### 5.4 Enable ModSecurity in NGINX



### 5.5 Setup OWASP CRS

cd /etc/nginx

sudo git clone https://github.com/coreruleset/coreruleset.gitmv coreruleset owasp-crscp owasp-crs/crs-setup.conf.example owasp-crs/crs-setup.conf

### 5.6 Configure ModSecurity Rules

mkdir /etc/nginx/modseccp /opt/ModSecurity/modsecurity.conf-recommended /etc/nginx/modsec/modsecurity.conf

sed -i 's/SecRuleEngine DetectionOnly/SecRuleEngine On/' /etc/nginx/modsec/modsecurity.conf

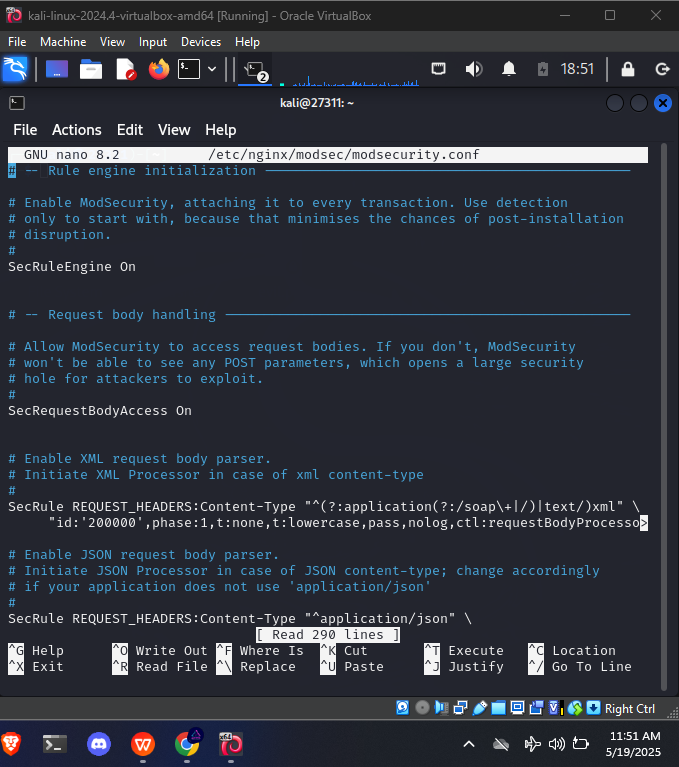
cat << 'EOF' | sudo tee /etc/nginx/modsec/main.conf

Include /etc/nginx/modsec/modsecurity.conf

Include /etc/nginx/owasp-crs/crs-setup.conf

Include /etc/nginx/owasp-crs/rules/\*.conf

EOF

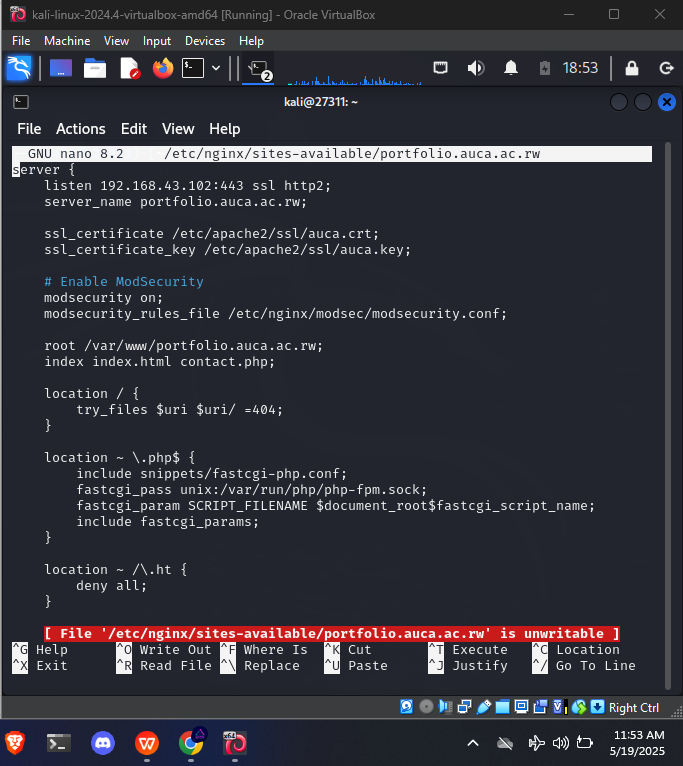


### 5.7 Reverse Proxy Configuration with HTTPS

Enable and restart:

ln -s /etc/nginx/sites-available/portfolio.auca.ac.conf /etc/nginx/sites-enabled/

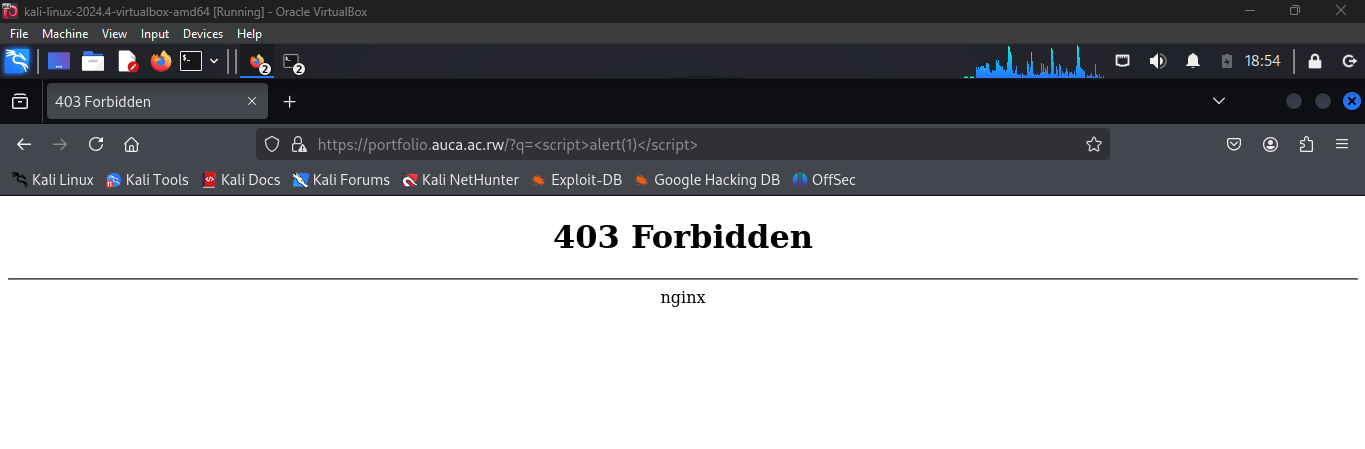
nginx -t && systemctl reload nginx



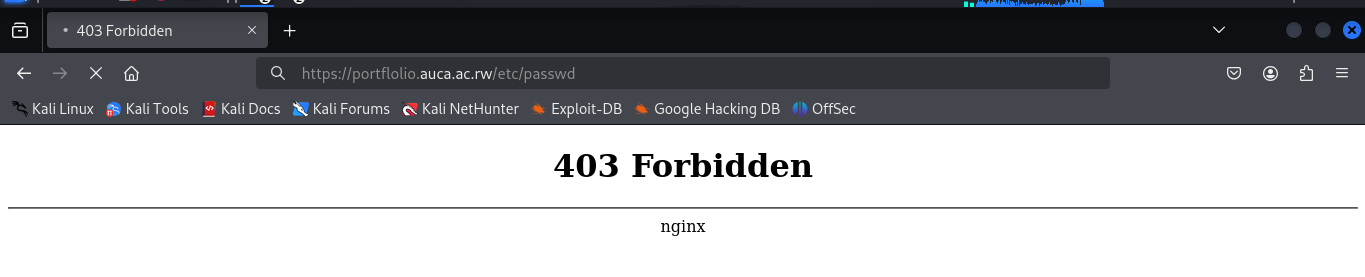
## 🧪 ****6. Testing the WAF****

### Test Injection:

https://portflolio.auca.ac.rw/?q=<script>alert(1)</script>



<https://portflolio.auca.ac.rw/etc/passwd>



**Secure SSH Access and Restricted Web Directory Environment for Web Server**

Configure a secure SSH environment for the user webaccess, allowing them to access only the web directory (/var/www/html) without being able to navigate or execute commands outside it.

## Summary of Key Actions

Created a new restricted user: webaccess

Assigned rbash as the default shell

Restricted SSH access to specific IP addresses

Enabled 2FA for SSH (placeholder for instructions)

Allowed HTTPS traffic and SSH from whitelisted IPs via UFW

Set up rbash with only essential commands

Locked the user to /var/www/

## Step-by-Step Implementation

### 1. Create the Restricted User

sudo adduser webaccess

### 2. Set Shell to rbash

sudo chsh -s /bin/rbash webaccess

### 3. Setup Limited Command Access

sudo mkdir /home/webaccess/bin

sudo cp /bin/ls /home/webaccess/bin/

sudo cp /bin/cat /home/webaccess/bin/

sudo cp /bin/nano /home/webaccess/bin/

sudo cp /bin/clear /home/webaccess/bin/

Sudo cp /bin/systemctl /home/webaccess/bin

### 4. Configure .bash\_profile

sudo nano /home/webaccess/.bash\_profile

PATH=$HOME/bin

export PATH

cd /var/www/html

Ensure it's owned and readable:

sudo chown webaccess:webaccess /home/webaccess/.bash\_profile

sudo chmod 644 /home/webaccess/.bash\_profile

### 5. Set Directory Permissions

sudo chown -R root:root /var/www

sudo chmod -R 755 /var/www

sudo chown -R webaccess:webaccess /var/www/

sudo chmod o+x /var/www

## UFW CONFIGURATION

### Objective:

Allow **only HTTPS (port 443)** to the public.

Allow **SSH (port 22)** only from 192.168.43.8. ( only my windows to ssh our server )

Block **all other traffic**.

### Steps:

**1. Reset UFW (if necessary)**

sudo ufw reset

**2. Default deny incoming, allow outgoing**

sudo ufw default deny incoming

sudo ufw default allow outgoing ( for like getting update, give responses)

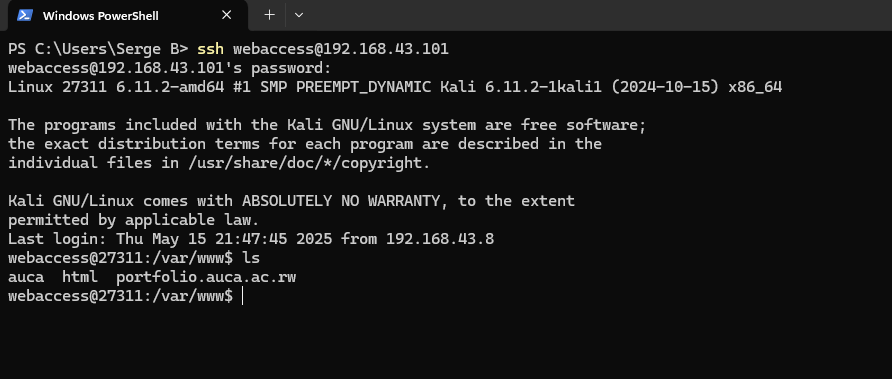
**3. Allow HTTPS (port 443)**

sudo ufw allow 443/tcp

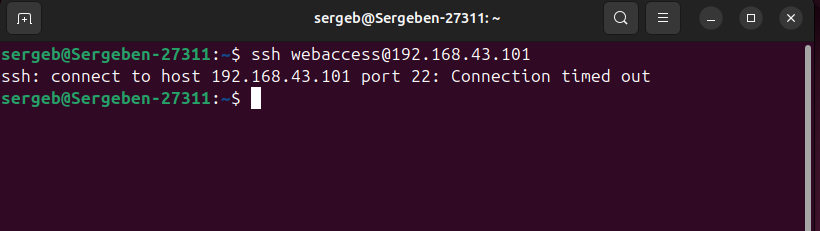
**4. Allow SSH ONLY from 192.168.43.8**

sudo ufw allow from 192.168.43.8 to any port 22 proto tcp

You see that now MY windows can ssh my server because is one configured in my firewall rules to only get access on port 22 via tcp protocal



My ubuntu can not get ssh connection because is not allowed by firewall to connect to ssh port

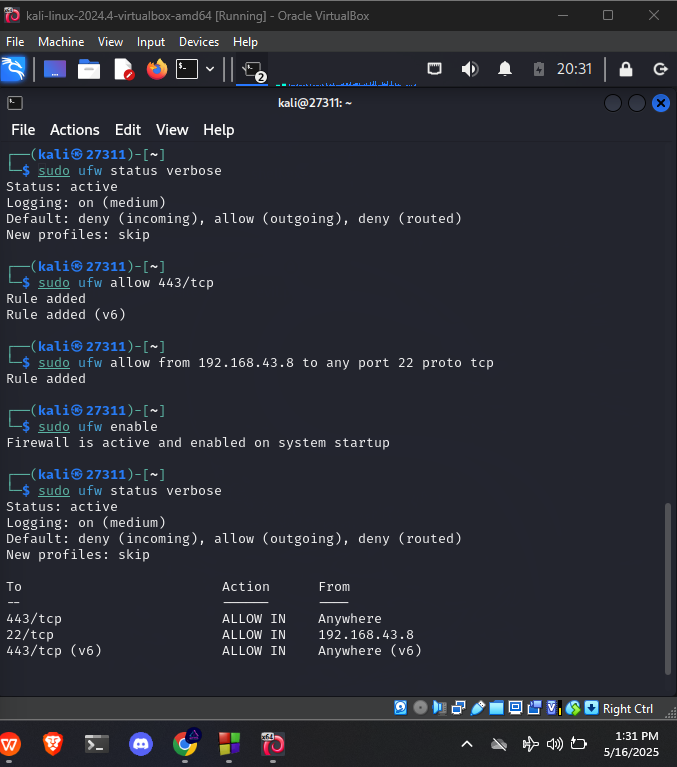


5. Enable UFW

sudo ufw enable

**6. Check UFW status**

sudo ufw status verbose



## Additional Security Notes

Root or sudo users still have full access to /var/www

webaccess is sandboxed using rbash, preventing escape

(This will help us when we have other system on our linux (ex: mysql) from being access by attacker in case he managed to trick user to give otp( social engineering ) and will sandbox him from previlage escalation)

