TATENDA KENNETH CHAITWA

Web Application Firewall Home Lab using SafeLine WAF

Contents

[Introduction 2](#_Toc203473533)

[How safeline WAF works 2](#_Toc203473534)

[Lab environment setup 2](#_Toc203473535)

[Ubuntu server configuration 3](#_Toc203473536)

[DNS Resolution Setup 5](#_Toc203473537)

[Creating a Self-Signed SSL Certificate 5](#_Toc203473538)

[Installing and Configuring SafeLine WAF 7](#_Toc203473539)

[**Importing the Self-Signed Certi**fi**cate** 8](#_Toc203473540)

[Onboarding the DVWA Application 8](#_Toc203473541)

[SafeLine WAF Advanced Configurations 10](#_Toc203473542)

[Issues faced when installing WAF 13](#_Toc203473543)

[Setting up DNS server 16](#_Toc203473544)

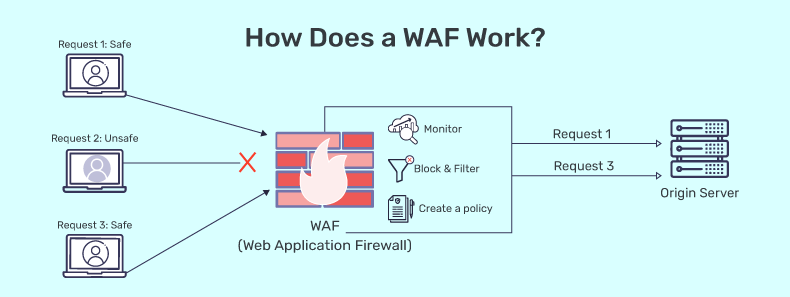
# Introduction

In this lab I built a cybersecurity home lab using kali Linux, ubuntu and safeline WAF

Basic setup

* Virtual box (hypervisor)
* 2 VMS kali (attacker), ubuntu (defender)
* Use bridged adapter (to ensure devices are on the same network)
* The machine is going to run using an Apache webserver
* The application that’ll be running is dvwa

# How safeline WAF works



# Lab environment setup

1. Install kali Linux and ubuntu virtual machines
2. Enable bridged networking:

This ensures the VMs appear on the same network as your host (and each other):

* **Open VirtualBox >**>> select your VM (e.g., Ubuntu Server) >>> **Settings** >>> **Network**.
* **Adapter 1**: Choose **Bridged Adapter** from the “Attached to” drop-down.
* Select your host’s network interface (Ethernet or Wi-Fi).
* Click **OK**.
* Grab the IP address of the machine after you install net-tools in section 4 by running the command “ifconfig” from the terminal.

Repeat process for both machines

# Ubuntu server configuration

**Initial System Updates and Installations**

**Update the Package List and Upgrade**: sudo apt-get update

sudo apt-get upgrade -y

**Install Net-Tools** (for ifconfig and other network utilities):

sudo apt-get install -y net-tools

**Install OpenSSL**:

sudo apt-get install -y openssl

**Installing and Con**fi**guring LAMP Stack**

**Install Apache2, PHP, and MySQL**:

sudo apt-get install -y apache2 php php-mysql MySQL-server

**Secure the MySQL Installation** (optional but recommended):

sudo mysql\_secure\_installation

Follow the prompts (set a root password, remove anonymous users, disable remote root login, etc.).

**Installing and Con**fi**guring Damn Vulnerable Web App (DVWA)**

**Clone DVWA** (or download):

cd /var/www/html

sudo git clone https://github.com/digininja/DVWA.git

If git is not installed, install it first:

sudo apt-get install -y git

**Set File Permissions**:

sudo chown -R www-data: www-data DVWA

sudo chmod -R 755 DVWA

**Con**fi**gure DVWA Database**:

DVWA has a config file at DVWA/config/config.inc.php. Update it if necessary: $DBMS = 'MySQL’.

$db = 'dvwa';

$user = 'dvwa\_user’.

$pass = 'p@ssw0rd';

$host = 'localhost';

**Create a new database and user in MySQL:**

sudo MySQL -u root -p

CREATE DATABASE dvwa;

CREATE USER 'dvwa\_user'@'localhost' IDENTIFIED BY 'p@ssw0rd';

GRANT ALL ON dvwa. \* TO 'dvwa\_user'@'localhost';

FLUSH PRIVILEGES.

exit;

**Initialize DVWA**:

Navigate to http://<Ubuntu IP>/DVWA/setup. Php in your browser. Click **Create/Reset Database**.

**4.4. Changing the DVWA Listening Port to 8080**

By default, Apache listens on port 80.

To change it to 8080:

**Edit the Apache Con**fi**guration**:

sudo nano /etc/apache2/ports.conf

**Change**: Listen 80

**to**: Listen to 8080

**Update the default Virtual Host** (optional if you want the default site on 8080):

sudo nano /etc/apache2/sites-available/000-default.conf

1. Change <VirtualHost \*:80> to <VirtualHost \*:8080>.

**Restart Apache**: sudo systemctl restart apache2

Now DVWA is accessible at http://<Ubuntu IP>:8080/DVWA.

# DNS Resolution Setup

**Using /etc/hosts for Local Resolution**

**Edit /etc/hosts on Both Ubuntu and Kali**:

sudo nano /etc/hosts

**Add a Line** (replace <Ubuntu IP> with the actual IP address): <Ubuntu IP> dvwa.local

Now you can access the DVWA app at http://dvwa.local:8080/DVWA in your browser (on Kali).

# Creating a Self-Signed SSL Certificate

**On Ubuntu, create a self-signed certificate to use for HTTPS:**

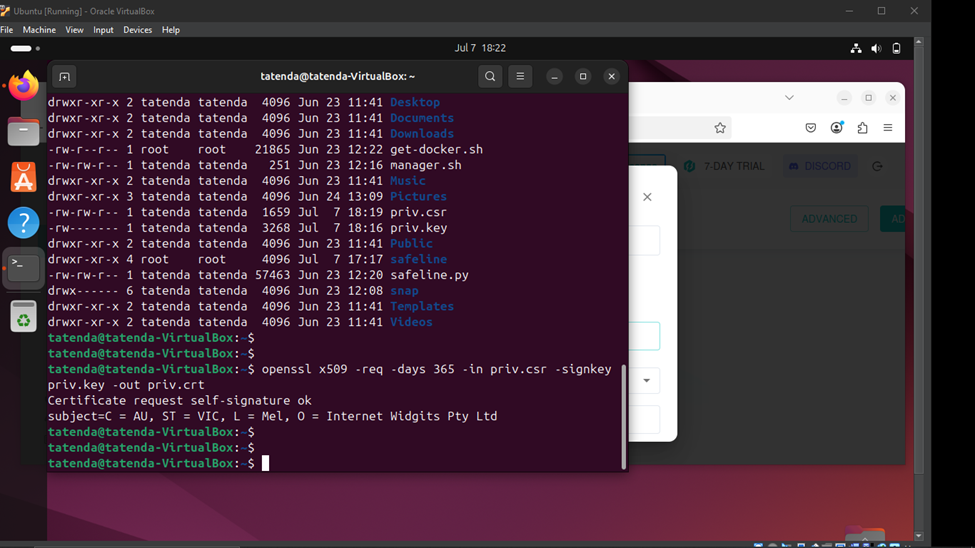
sudo mkdir /etc/ssl/dvwa

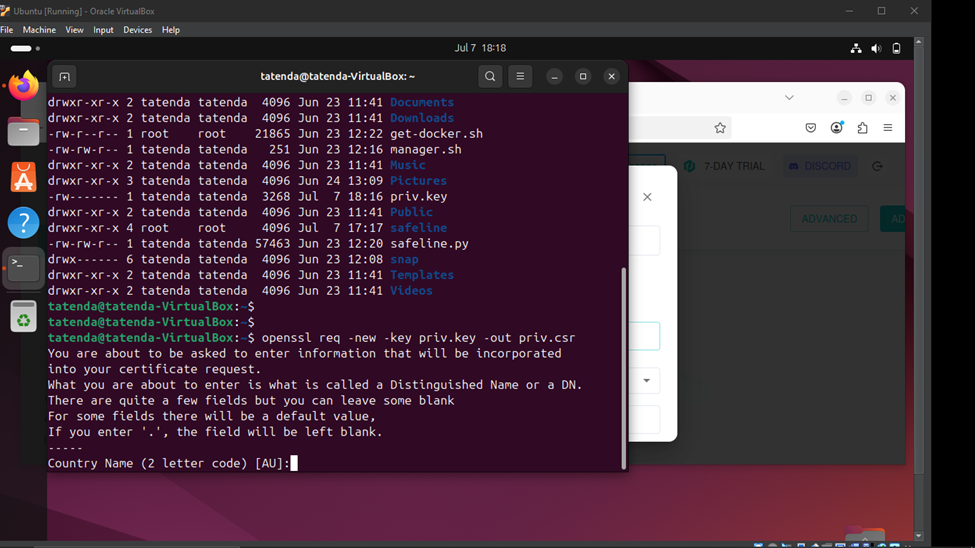
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \

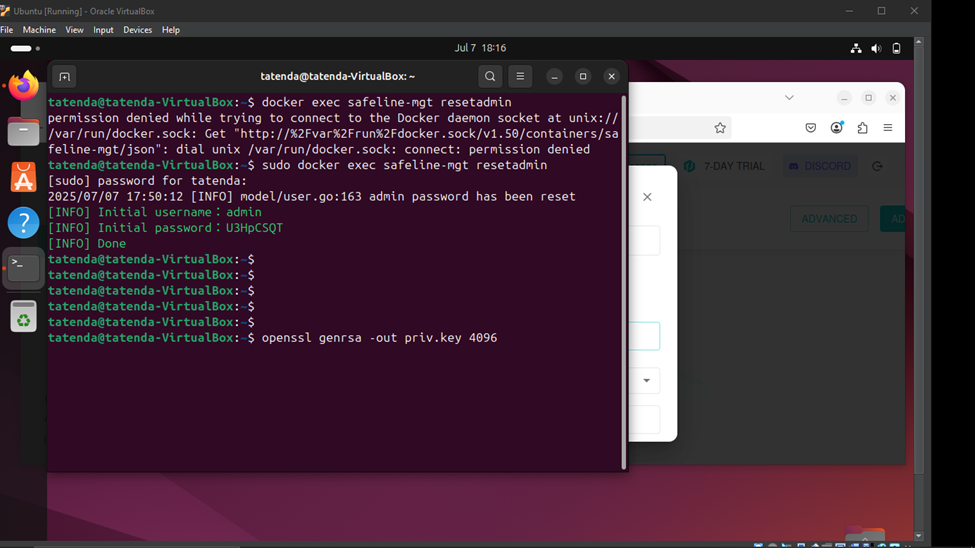
-keyout /etc/ssl/dvwa/dvwa.key \

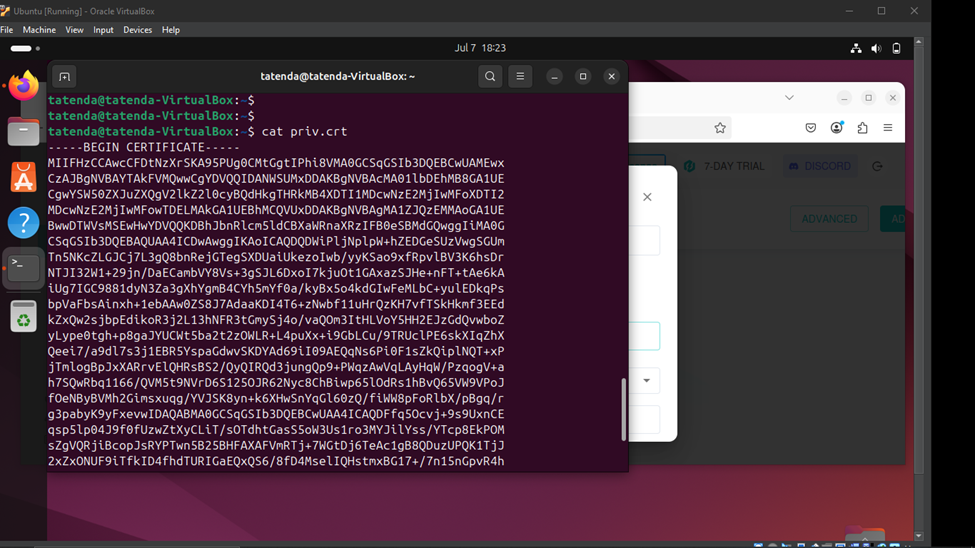
-out /etc/ssl/dvwa/dvwa.crt \

The image below shows phases using different commands and the last image shows the final certificate after successful prompting









# Installing and Configuring SafeLine WAF

**Automatic Deployment**

Head to the following website https://ly.safepoint.cloud/zfYZD3l and select “Install”

Safeline WAF typically provides an **automatic install script** or a manual install method, follow the steps below for the automatic install:

1. **Use the following command on your Ubuntu CLI**

**-** bash -c "$(curl -fsSLk https://waf.chaitin.com/release/latest/manager.sh)" -- --en

2. **Follow On-Screen Prompts** to complete installation.

3. You will be provided with an admin password and username as well as the URL to access Safeline, typically on port 9443

# **Importing the Self-Signed Certi**fi**cate**

In the SafeLine WAF interface (web UI or config files), import the .crt and .key files you created under the SSL certificate section.

● **Certi**fi**cate File**: /etc/ssl/dvwa/dvwa.crt

● **Key File**: /etc/ssl/dvwa/dvwa.key

Follow the SafeLine documentation on how to manage certificates (it could be through a GUI or CLI).

# Onboarding the DVWA Application

Use the SafeLine WAF management console or interface to **add a new application**:

1. **Add the DNS name -** www.dvwa.local or whatever Domain name you have selected

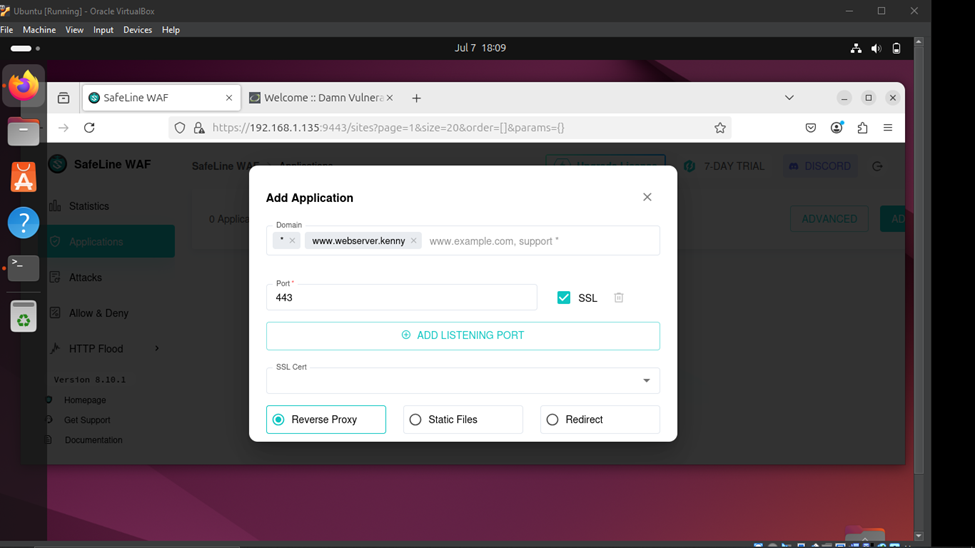
2. **Backend URL (Reverse Proxy):** http://<UbuntuIP>:8080

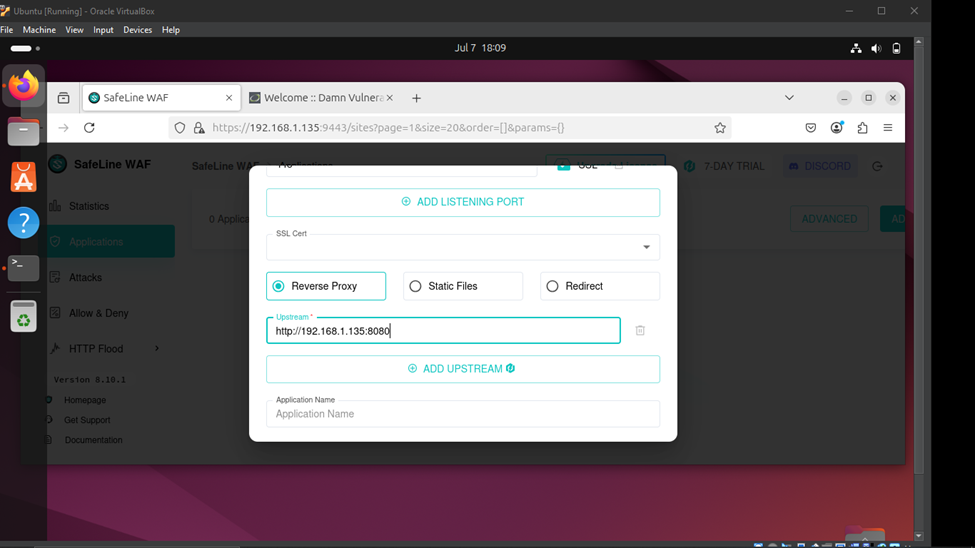
3. Delete port 80 and only leave port 443

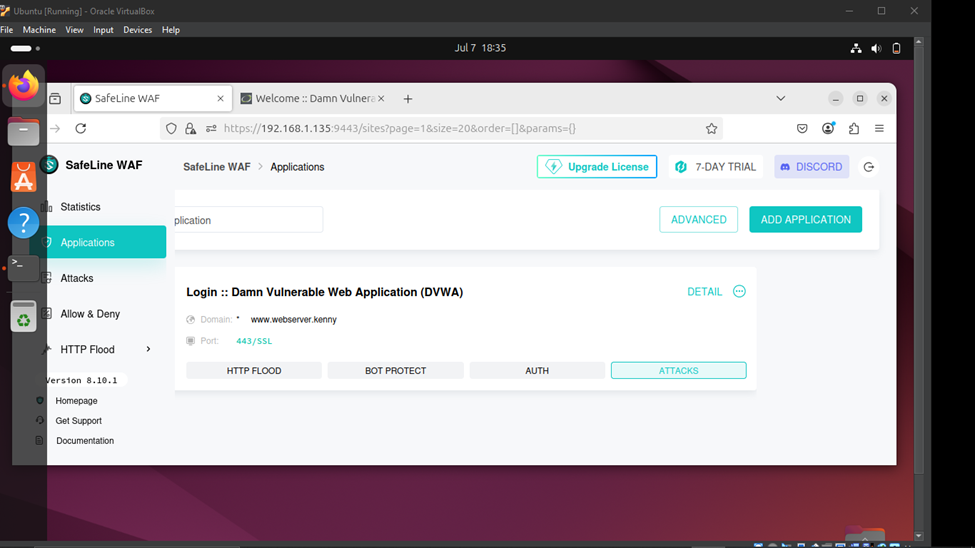
4. **Virtual Host** (for WAF): e.g., dvwa-waf.local or re-use the same domain.

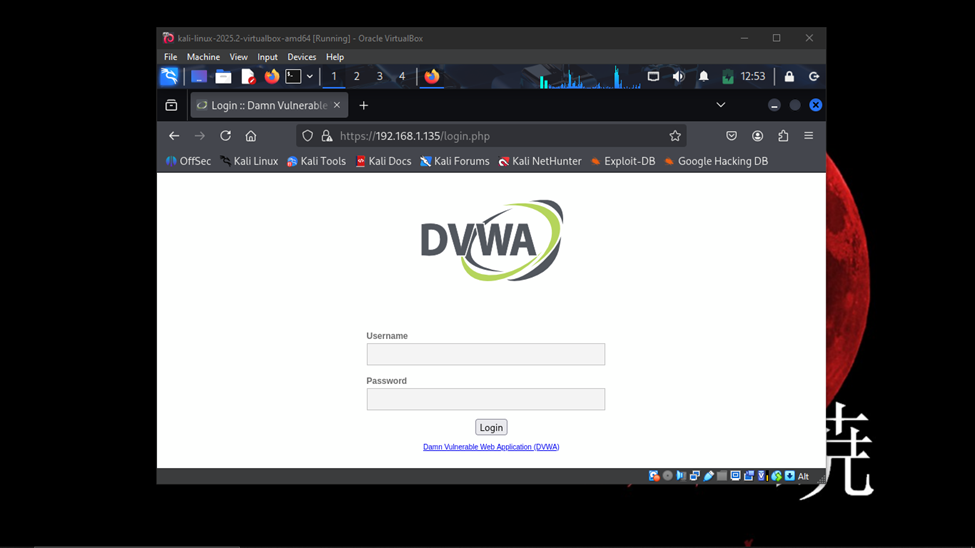
5. **Attach the SSL Certi**fi**cate** (if you want the WAF to serve HTTPS).

The images below represent phases in the processes and the last image showcases how trying to access the dvwa application from the attack machine using http://192.168.2.135 it is translated to https://192.168.2.135 which is more secure.









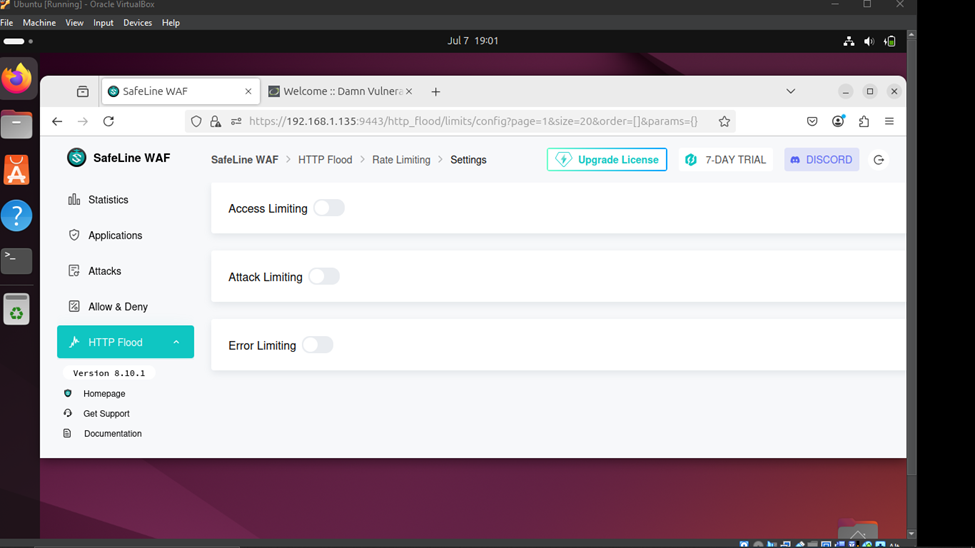
# SafeLine WAF Advanced Configurations

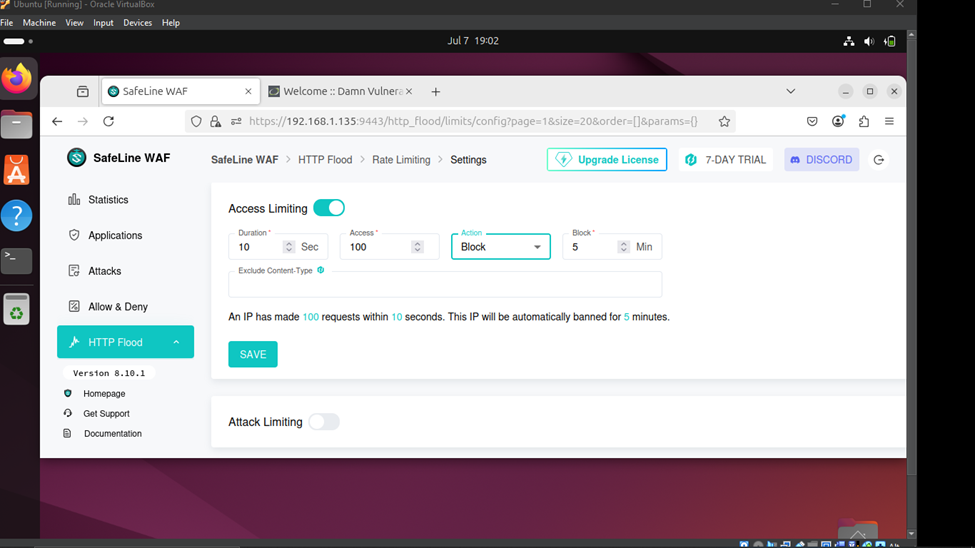
**HTTP Flood Defense**

1. **Enable** HTTP Flood/DoS settings in SafeLine WAF.

2. Configure **Thresholds** (requests per second) and **Penalty** or **Ban** duration.

3. **Test** by sending multiple requests from Kali (e.g., using ab, siege, or other load tools).





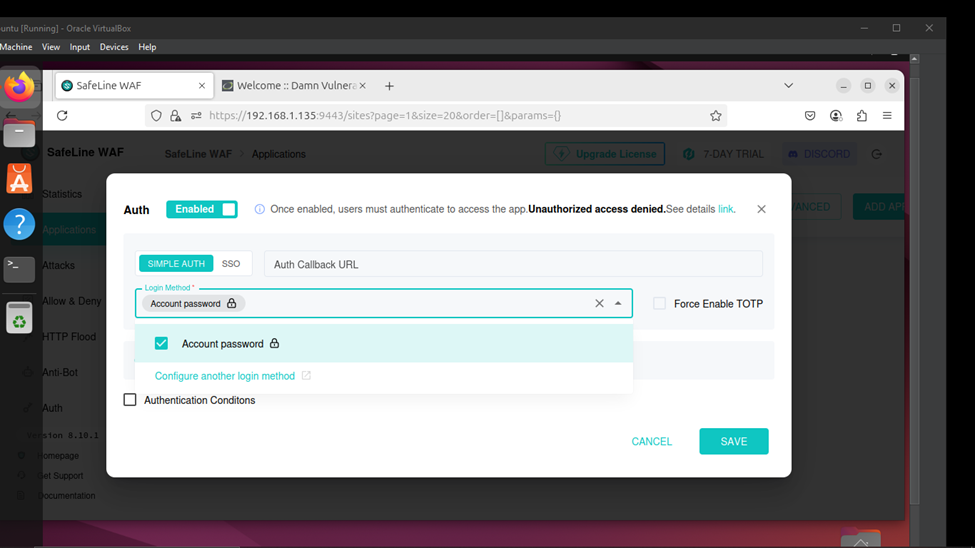
**Authentication Sign-In**

SafeLine supports an **auth gateway**:

1. **Enable** Auth Sign-In in the WAF policy for DVWA.

2. **Con**fi**gure** username/password or integrate with an external auth provider.

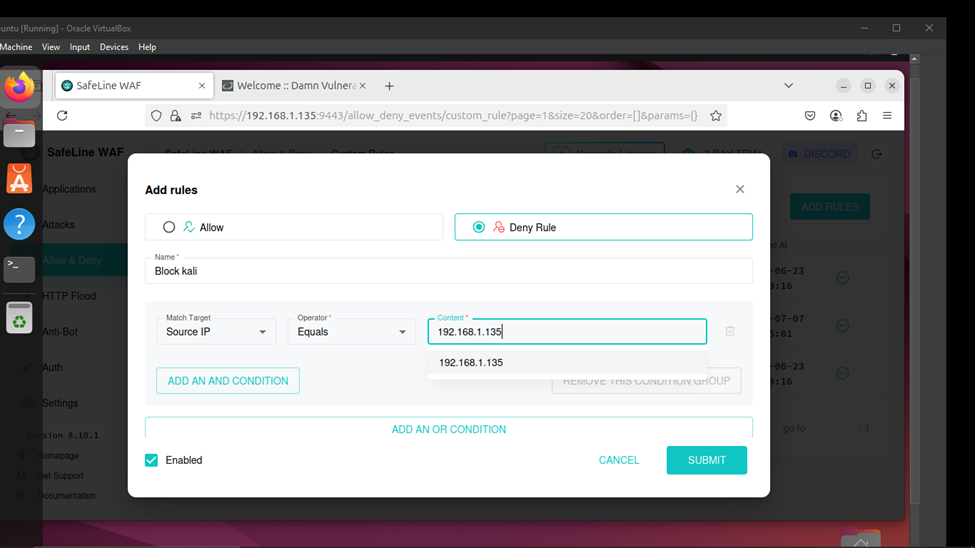
3. Attempt to access DVWA from Kali; the WAF should prompt for credentials before passing traffic to the server.



**Custom Deny Rules (Blocking Kali IP)**

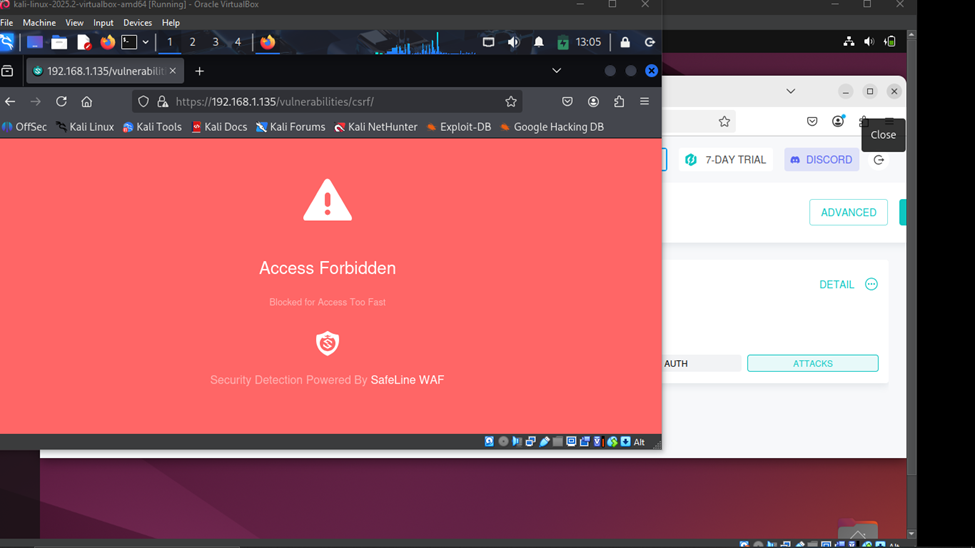
1. **Identify** the IP of your Kali VM (e.g., 192.168.1.50).

2. **Add a Deny Rule** in SafeLine WAF: ○ Match Source IP: 192.168.1.50

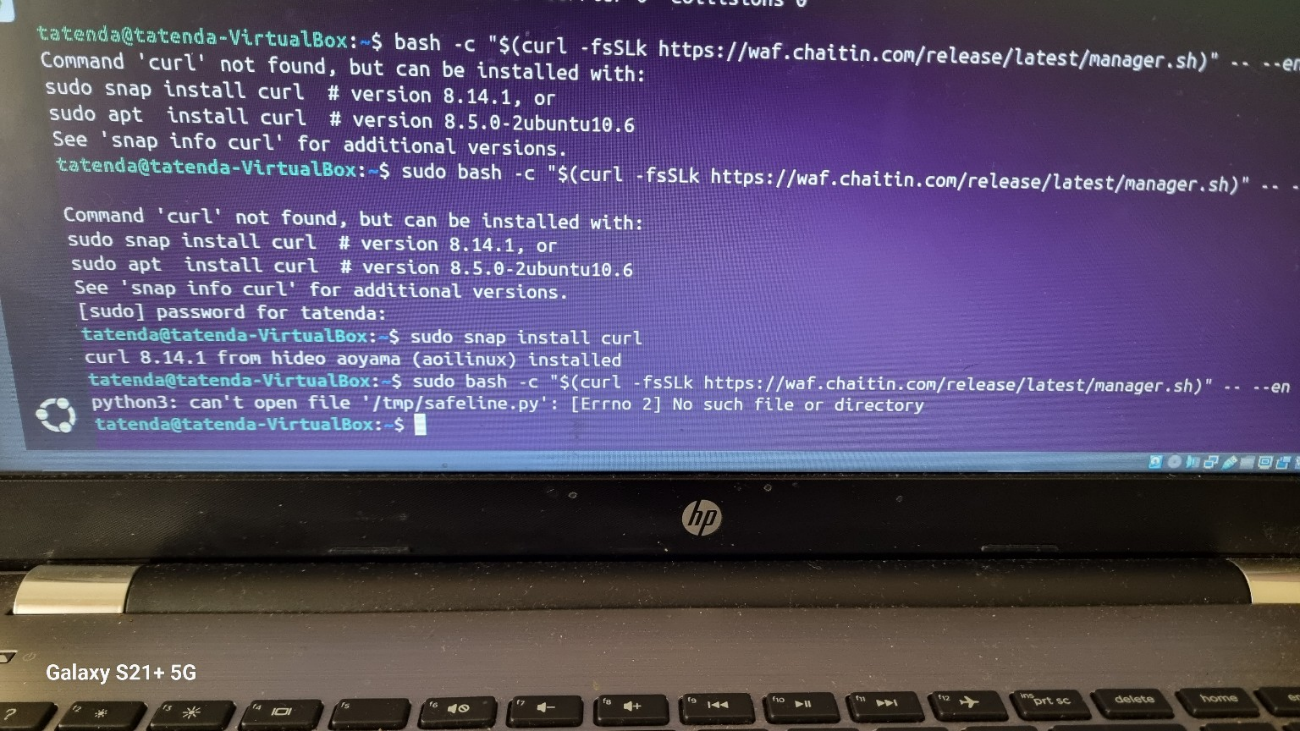


Action: Block or deny.

1. **Test** from Kali: You should receive a blocked response as shown below



# Issues faced when installing WAF



Error >>> python3: can’t open file ‘/tmp/safeline.py’: [Errno 2] No such file or directory

The error means the script expected to download a file called safeline.py to /tmp/, but that file doesn’t exist.

Fix steps

1. Try downloading the script manually and see what it is doing:

* curl -fsSL <https://waf>.chaitin.com/release/latest/manager.sh -o manager.sh
* cat manager.sh

A computer screen with white text

AI-generated content may be incorrect.

This part means:

If you’re not root, print an error and exit.

But there’s a bug — the if statement is empty, and the echo and exit commands run regardless of your privileges.

So even when you run it with Sudo, it exits early and never downloads safeline.py.

**Solution:**

**: Run the Python script directly**

* curl -fsSL <https://waf>.chaitin.com/release/latest/manager.py -o safeline.py
* sudo python3 safeline.py –en

or

Open the script in a text editor:

* nano manager.sh

Then fix the if block to enforce root privileges:

#!/bin/bash

set -e

if [[ $EUID -ne 0 ]]; then

echo "This script must be run as root"

exit 1

fi

URL="https://waf.chaitin.com/release/latest/manager.py"

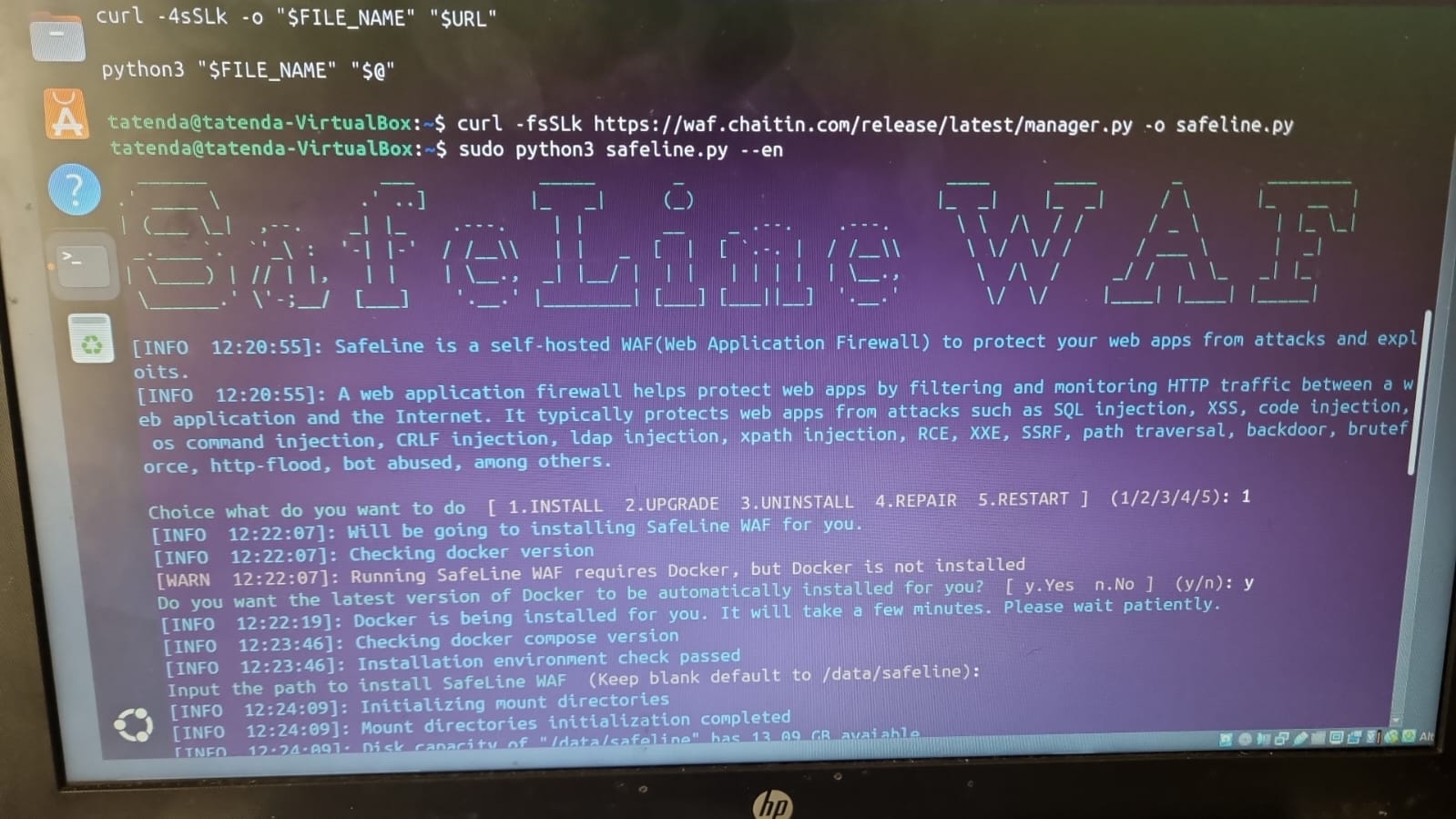
FILE\_NAME="/tmp/safeline.py"

curl -4sSLk -o "$FILE\_NAME" "$URL"

python3 "$FILE\_NAME" "$@"

Then run it again:

* sudo ./manager.sh –en



# Setting up DNS server

Setting up DNS-like behavior using /etc/hosts is a simple way to map domain names to IP addresses *locally* on a system—without needing a DNS server. This is useful for testing or working in small private networks.

**What is /etc/hosts?**

/etc/hosts is a plain-text file used by the operating system to map hostnames (like example.local) to IP addresses (like 192.168.1.10) **before** querying DNS servers.

**How to Set It Up**

1. **Open the file with a text editor (as root):**

sudo nano /etc/hosts

1. Add entries like this (adjust for your actual IPs):

127.0.0.1 localhost

192.168.1.10 myserver.local myserver

10.0.0.15 webapp.local webapp

1. **Save and exit** the editor (in Nano: Ctrl + X, then Y, then Enter)
2. Use ping or ssh to test the hostname:

ping myserver.local

ssh user@webapp.local