Web Application Firewall for DVWA

By Shubodaya Kumar

Date: 22/07/2025

Email: hnshubodaya@gmail.com

Contents

1 Setting Up Your Web Security Lab 4

2 What You'll Need Before Starting 4

3 Building the Virtual Lab 5

3.1 Install VirtualBox 5

3.2 Create the Kali Linux VM 5

3.3 Create the Ubuntu Server VM 6

3.4 Configure Network Settings 6

3.5 (Optional) Install Guest Additions 6

4 Preparing the Web Server 7

4.1 System Updates and Tools 7

4.2 Install Apache, MySQL, and PHP 7

4.3 Deploy DVWA (Damn Vulnerable Web App) 8

4.4 Change Apache Port to 8080 8

4.5 Add Sample Data for Testing 9

5 DNS Setup for Local Access 10

5.1 Quick Method: Edit `/etc/hosts` 10

5.2 (Optional) Set Up a Local DNS Server with BIND 10

6 Creating a Basic SSL Certificate 12

7 Installing and Using SafeLine WAF 13

7.1 Quick Deployment of SafeLine WAF 13

7.2 Add Your SSL Certificate to SafeLine 14

7.3 Link DVWA to SafeLine WAF 14

8 Testing with SQL Injection 14

8.1 Launching the Attack from Kali 15

8.2 Watching SafeLine Block the Attack 15

9 Exploring Extra WAF Features 16
9.1 HTTP Flood / DoS Protection 16
9.2 Add a Login Prompt (WAF Authentication Gateway) 16
9.3 Blocking by IP Address 17
10 Final Thoughts and What to Try Next 17
11 Quick Reference and Troubleshooting Tips 18
11.1 Troubleshooting Tips 18
11.2 Security Best Practices 19
12 References 19

1 Setting Up Your Web Security Lab

This guide walks you through building a full-featured web security lab at home. You'll simulate real-world attacks on a deliberately vulnerable application and defend against them using a Web Application Firewall (WAF). Everything runs inside virtual machines, so you can experiment freely without messing up your main system.

Here's what you'll be doing:

- Set up a vulnerable web application called DVWA on an Ubuntu server.
- Use Kali Linux to simulate attacks like SQL injection.
- Install and configure SafeLine WAF to detect and block those attacks.
- Explore additional WAF security features like flood protection and access control.

By the end of this lab, you'll have a working environment to test, learn, and practice offensive and defensive cybersecurity techniques. If you hit any roadblocks, don't stress most issues can be solved with a quick online search or by checking logs on each layer of your setup.

2 What You'll Need Before Starting

Before diving into the lab setup, make sure you've got the right tools and resources in place. Here's a quick checklist to get everything ready:

Hardware Requirements

- A physical host machine (your main computer) with:
    - **At least 8 GB of RAM** (more is better)
    - **50 GB+ of free disk space**
    - A reliable internet connection

Software You'll Be Using

- **VirtualBox** – for running virtual machines
- **Kali Linux** – to simulate attacks
- **Ubuntu Server** – to host the vulnerable web app
- **DVWA (Damn Vulnerable Web App)** – the actual target
- **SafeLine WAF** – to detect and block attacks

Basic Skills

- Some experience with the Linux terminal: installing packages, editing config files, restarting services, etc.

- Comfort with basic networking concepts (IP addresses, ports, DNS, etc.)

3 Building the Virtual Lab

Now let's set up your virtual machines. You'll run two:

- One Ubuntu Server (to host DVWA)
- One Kali Linux (to attack the app and test your defenses)

We'll be using **VirtualBox** to run both systems on your host machine.

3.1 Install VirtualBox

1. Go to the official site: https://www.virtualbox.org/wiki/Downloads
2. Download the version for your OS (Windows/macOS/Linux)
3. Run the installer and follow the prompts
4. (Optional) Install the **VirtualBox Extension Pack** for features like USB 2.0/3.0 support

3.2 Create the Kali Linux VM

1. **Download the Kali ISO**
    - Get it from https://www.kali.org/get-kali
    - Choose the **64-bit installer ISO**
2. **Create a new VM in VirtualBox**
    - Name: KaliLinux
    - Type: Linux, Version: Debian (64-bit)
    - RAM: at least **2 GB**
    - Disk: ~20 GB, dynamically allocated
3. **Attach the ISO & Install**
    - Go to **Settings > Storage** and attach the ISO under the IDE controller
    - Start the VM and complete the installation (graphical install recommended)
    - Create a username and password (e.g., kali / kali)

3.3 Create the Ubuntu Server VM

1. **Download Ubuntu Server**
    - Get it from https://ubuntu.com/download/server
    - Use the latest LTS version (e.g., 22.04 LTS)
2. **Create the VM**
    - Name: UbuntuServer
    - Type: Linux, Version: Ubuntu (64-bit)
    - RAM: **2 GB or more**
    - Disk: ~20 GB, dynamically allocated
3. **Install Ubuntu Server**
    - Attach the ISO under **Settings > Storage**
    - Start the VM and follow the prompts
    - Create a user/password (e.g., ubuntu / ubuntu)
    - (Optional) Install the **OpenSSH server** if you want remote access via SSH

3.4 Configure Network Settings

To make both VMs communicate with each other and your host, use **bridged networking**:

1. In VirtualBox, select your VM → **Settings > Network**
2. Set **Adapter 1** to Bridged Adapter
3. Choose your physical network interface (Wi-Fi or Ethernet)
4. Click **OK**

Repeat this for both VMs. After setup, install net-tools on each and run ifconfig to get their IP addresses.

3.5 (Optional) Install Guest Additions

Guest Additions improve usability like screen resolution, shared clipboard, and folder sharing.

On each VM:

1. While the VM is running, go to **Devices > Insert Guest Additions CD Image**
2. Run the following:

```
sudo apt-get update

sudo apt-get install build-essential dkms linux-headers-$(uname
-r)

sudo mount /dev/cdrom /media/cdrom

sudo /media/cdrom/VBoxLinuxAdditions.run
```

3. Reboot the VM.

4 Preparing the Web Server

Your Ubuntu Server will act as the target system. You'll update it, install the LAMP stack (Linux, Apache, MySQL, PHP), and deploy the Damn Vulnerable Web App (DVWA).

4.1 System Updates and Tools

First, update your packages and install essential tools:

```
sudo apt-get update

sudo apt-get upgrade -y

sudo apt-get install -y net-tools openssl
```

You'll need net-tools for basic networking commands like ifconfig, and openssl later for generating an SSL certificate.

4.2 Install Apache, MySQL, and PHP

Install the core components of a LAMP stack:

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

Once installed, secure your MySQL setup (optional but recommended):

```
sudo mysql_secure_installation
```

Follow the prompts to:

- Set a root password
- Remove anonymous users
- Disallow remote root login
- Remove the test database
- Reload privileges

4.3 Deploy DVWA (Damn Vulnerable Web App)

Clone DVWA into the web root:

```
cd /var/www/html

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

If git isn't installed:

```
sudo apt-get install -y git
```

Fix the permissions:

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

sudo chmod -R 755 DVWA
```

Now configure the database connection. Open:

```
sudo nano /var/www/html/DVWA/config/config.inc.php
```

Make sure it looks like this:

```
$_DVWA[ 'db_user' ] = 'dvwa_user';

$_DVWA[ 'db_password' ] = 'p@ssw0rd';

$_DVWA[ 'db_database' ] = 'dvwa';
```

4.4 Change Apache Port to 8080

By default, Apache listens on port 80. To move it to 8080:

Edit the ports config:

```
sudo nano /etc/apache2/ports.conf
```

Change:

```
Listen 80
```

to:

```
Listen 8080
```

Now update the virtual host:

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

Change:

```
<VirtualHost *:80>
```

to:

```
<VirtualHost *:8080>
```

Restart Apache:

```
sudo systemctl restart apache2
```

Now DVWA should be available at:
http://<Ubuntu-IP>:8080/DVWA

4.5 Add Sample Data for Testing

Let's drop in some dummy data to use in your SQL injection tests.

```
sudo mysql -u root -p
```

Then inside the MySQL shell:

```
CREATE DATABASE dvwa;

CREATE USER 'dvwa_user'@'localhost' IDENTIFIED BY 'p@ssw0rd';

GRANT ALL ON dvwa.* TO 'dvwa_user'@'localhost';

FLUSH PRIVILEGES;

USE dvwa;


CREATE TABLE test_users (

    id INT NOT NULL AUTO_INCREMENT,

    username VARCHAR(50) NOT NULL,

    password VARCHAR(50) NOT NULL,

    PRIMARY KEY (id)

);


INSERT INTO test_users (username, password) VALUES

('alice', 'alice123'),

('bob', 'bob123'),
```

```
('admin', 'admin123');



exit;
```

Finally, initialize the DVWA database:

Open your browser and go to:
http://<Ubuntu-IP>:8080/DVWA/setup.php
Click **Create/Reset Database**

5 DNS Setup for Local Access

You've got DVWA running, but typing `http://192.168.x.x:8080/DVWA` isn't ideal. Let's fix that with a custom local domain. You've got two options: a simple `hosts` file edit, or a full DNS server setup using BIND.

5.1 Quick Method: Edit `/etc/hosts`

This is the easiest way for both your Ubuntu and Kali VMs to resolve a domain name like `dvwa.local`.

On **both machines** (Ubuntu and Kali):

```
sudo nano /etc/hosts
```

Add this line (replace `<Ubuntu-IP>` with the actual IP address of your Ubuntu VM):

```
<Ubuntu-IP>    dvwa.local
```

Now from your Kali machine, you can access the app like this:

```
http://dvwa.local:8080/DVWA
```

And yes, this works even if you're offline.

5.2 (Optional) Set Up a Local DNS Server with BIND

If you'd rather practice DNS configuration or want to scale your lab later, set up BIND on your Ubuntu server.

Step 1: Install BIND

```
sudo apt-get install -y bind9
```

Step 2: Define a New Zone

```
sudo nano /etc/bind/named.conf.local
```

Add:

```
zone "dvwa.local" {
    type master;
    file "/etc/bind/zones/db.dvwa.local";
};
```

Step 3: Create the Zone File

```
sudo mkdir -p /etc/bind/zones
sudo nano /etc/bind/zones/db.dvwa.local
```

Paste this (replace <Ubuntu-IP> with your server IP):

```
$TTL    604800
@       IN      SOA     ns.dvwa.local. admin.dvwa.local. (
                          1         ; Serial
                          604800    ; Refresh
                          86400     ; Retry
                          2419200   ; Expire
                          604800 )  ; Negative Cache TTL

@       IN      NS      ns.dvwa.local.
ns      IN      A       <Ubuntu-IP>
www     IN      A       <Ubuntu-IP>
```

Step 4: Restart BIND

```
sudo systemctl restart bind9
```

Step 5: Point Kali to the New DNS Server

On Kali:

```
sudo nano /etc/resolv.conf
```

Add:

```
nameserver <Ubuntu-IP>
```

Step 6: Test It

On Kali, run:

```
dig www.dvwa.local
```

If DNS is working, you should get a valid response.

6 Creating a Basic SSL Certificate

Now that DVWA is running and accessible, let's add HTTPS support with a self-signed SSL certificate. This is important for testing how the WAF handles encrypted traffic.

On your **Ubuntu server**, follow these steps:

Step 1: Create a Directory for SSL Files

```
sudo mkdir /etc/ssl/dvwa
```

Step 2: Generate the SSL Certificate and Key

Run the following command. It'll create both the certificate (dvwa.crt) and the private key (dvwa.key):

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

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

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

You'll be asked for details like country, organization name, etc. You can enter whatever you want or just press Enter through it.

Next Steps (Optional for Now)

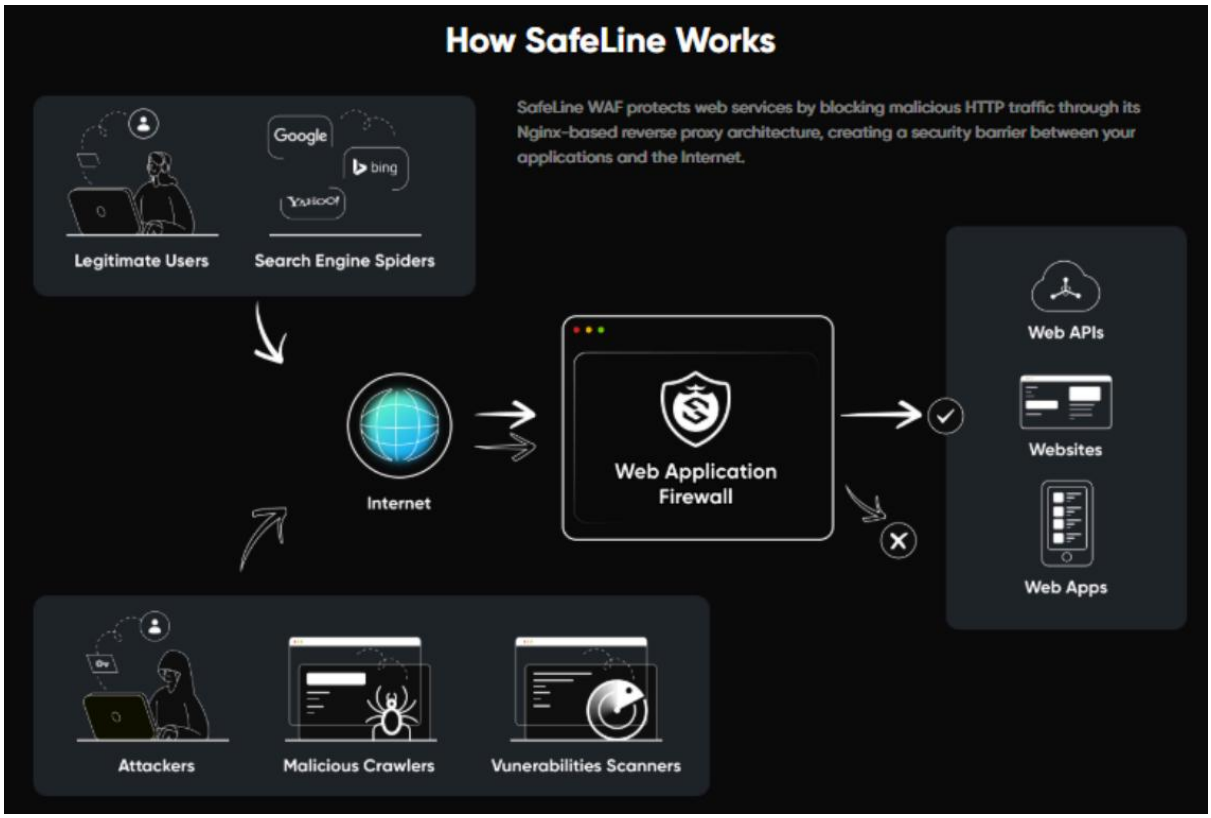
You don't need to configure Apache to use this certificate just yet. SafeLine WAF will use it when we onboard DVWA into the WAF interface.

So, keep the file paths handy:

- Certificate: /etc/ssl/dvwa/dvwa.crt
- Key: /etc/ssl/dvwa/dvwa.key

7 Installing and Using SafeLine WAF

With your web app up and running, now it's time to bring in SafeLine WAF. This firewall will sit in front of DVWA and inspect traffic for malicious behaviour.



7.1 Quick Deployment of SafeLine WAF

SafeLine offers an easy script-based install. Here's how to get it running on your **Ubuntu server**:

Step 1: Run the install command

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

If that URL ever breaks, the original source is also linked from: https://ly.safepoint.cloud/zfYZD3l

Step 2: Follow the prompts

- You'll get a **web login URL** (usually on port 9443)
- It will also give you a default **admin username and password**

Open a browser and go to:

```
https://<Ubuntu-IP>:9443
```

The page might throw a security warning due to the self-signed cert—just bypass it.

7.2 Add Your SSL Certificate to SafeLine

Inside the SafeLine interface:

1. Go to the **SSL certificate** section
2. Upload the certificate and key you created earlier:
    - Certificate file: /etc/ssl/dvwa/dvwa.crt
    - Key file: /etc/ssl/dvwa/dvwa.key
3. Save and apply the changes

This allows SafeLine to terminate HTTPS traffic coming into your DVWA app.

7.3 Link DVWA to SafeLine WAF

Now onboard the DVWA application into the WAF:

1. **Add an application/domain**
    - Name: dvwa.local or www.dvwa.local
    - Backend URL: http://<Ubuntu-IP>:8080
2. **Update the ports**
    - Remove port 80
    - Keep only port 443 (for HTTPS)
3. **Set virtual host (optional)**
    - You can reuse dvwa.local or create a new one like dvwa-waf.local
4. **Attach your SSL cert**
    - Use the cert/key you just uploaded

Once that's saved, SafeLine will now act as a reverse proxy and firewall for DVWA.

8 Testing with SQL Injection

Now that SafeLine WAF is in front of your DVWA site, it's time to simulate a real attack and observe how the WAF handles it. You'll use Kali Linux to perform a basic SQL injection and check whether SafeLine detects or blocks it.

8.1 Launching the Attack from Kali

1. **Open Firefox or any browser inside Kali**
2. Visit your DVWA site via HTTPS through the WAF:

```
https://dvwa.local
```

If you used a different domain during setup (like dvwa-waf.local), use that instead.

3. **Log in to DVWA**
   Default credentials (unless you changed them):

```
Username: admin

Password: password
```

4. **Set DVWA Security Level to Low**
   - Go to the **DVWA Security** tab
   - Set the security level to **Low** and click **Submit**
5. **Go to the SQL Injection section**
   - In the DVWA sidebar, click **"SQL Injection"**
6. **Try a simple SQL injection**

Enter this in the user ID field:

```
admin' OR '1'='1
```

Click **Submit**

8.2 Watching SafeLine Block the Attack

If SafeLine WAF is in **blocking mode**, one of two things will happen:

- You'll see a **block page** instead of a DVWA response
- Or DVWA will appear to load, but **WAF logs** will show that the injection was detected and stopped

To confirm, log into the SafeLine web interface:

- Navigate to **Logs > Web Protection** (or wherever WAF logs are displayed)
- Look for entries showing blocked SQL injection attempts from your Kali VM's IP

9 Exploring Extra WAF Features

SafeLine does more than just block SQL injections. This section shows how to test additional defences like rate-limiting, user authentication, and IP-based access control - all useful in real-world scenarios.

9.1 HTTP Flood / DoS Protection

You can simulate a simple Denial-of-Service-style attack using tools like ab or siege from Kali, and configure SafeLine to detect and block it.

Step 1: Enable HTTP Flood Protection

- In SafeLine's interface, go to the **security policy** or **traffic control** section
- Enable **HTTP Flood** or **Rate Limiting**

Step 2: Configure the thresholds

- Set a limit on requests per second (e.g., 10 req/sec)
- Choose a ban duration (e.g., 5 minutes)

Step 3: Launch a test flood from Kali

Example using ab (ApacheBench):

```
ab -n 1000 -c 50 https://dvwa.local/
```

If set up correctly, SafeLine should start throttling or blocking the flood traffic, and you'll see errors in the ab output.

9.2 Add a Login Prompt (WAF Authentication Gateway)

You can require users to authenticate *before* even reaching DVWA—acting like a login wall.

Step 1: Enable Auth Sign-In

- In SafeLine, edit the DVWA app's WAF policy
- Turn on **authentication gateway** or similar

Step 2: Set credentials

- Either configure a simple username/password pair, or integrate with an existing identity provider (like LDAP or SSO)

Step 3: Test it from Kali

- When you visit https://dvwa.local, SafeLine should now ask for a username and password *before* forwarding traffic to DVWA

This is useful for staging sites, admin panels, or testing isolated environments.

9.3 Blocking by IP Address

Let's say you want to block your Kali VM entirely.

Step 1: Find Kali's IP

Run this on Kali:

```
ifconfig
```

Note the IP address (e.g., 192.168.1.50)

Step 2: Add a deny rule in SafeLine

- Go to the WAF policy or IP control section
- Add a rule like:

```
Source IP: 192.168.1.50

Action: Block
```

Step 3: Test it

Try accessing the DVWA site again from Kali. You should see a block message or get no response at all.

10 Final Thoughts and What to Try Next

You've now built a complete web security lab from scratch. You installed and configured a vulnerable app (DVWA), simulated real attacks from Kali, and set up SafeLine WAF to catch and block them.

That's no small thing.

Here's a recap of what you've achieved:

- Built isolated VMs with Kali Linux and Ubuntu Server
- Installed and configured Apache, MySQL, PHP, and DVWA
- Set up SafeLine WAF and integrated it as a reverse proxy
- Simulated SQL injections and HTTP floods
- Explored SafeLine's advanced protection features like IP blocking and login prompts

But this doesn't have to stop here.

Keep Exploring

Here are a few solid next steps:

- **Try other vulnerable web apps** like OWASP Juice Shop, bWAPP, or WebGoat
- **Test different attack types** - XSS, file inclusion, command injection, CSRF
- **Dig into WAF logs** to understand how requests are analyzed and classified
- **Pair this lab with a SIEM or IDS/IPS** to start collecting and correlating events
- **Write your own deny rules** in SafeLine to block specific patterns or headers
- **Play with SafeLine in detection-only mode** to monitor without blocking

This kind of hands-on lab is a powerful learning tool. It's not just about breaking stuff—it's about observing, adjusting, and really understanding how traffic flows, how attacks behave, and how defences respond.

11 Quick Reference and Troubleshooting Tips

Even in a test lab, things can (and will) go wrong. This section covers quick fixes, common issues, and general advice to help keep your setup clean, stable, and useful.

11.1 Troubleshooting Tips

Issue: One VM can't reach the other
Fix:

- Make sure both VMs use **bridged networking**
- Run ifconfig on both VMs to confirm their IPs
- Check firewall settings (e.g., ufw) on Ubuntu

Issue: DVWA isn't loading
Fix:

- Check Apache with: sudo systemctl status apache2
- Verify DVWA folder permissions: www-data should own the files
- Confirm you're using the correct port: http://<IP>:8080/DVWA

Issue: SQL injection test didn't trigger the WAF
Fix:

- Confirm WAF is in **blocking mode** (not just logging)
- Make sure traffic is going through the WAF, not bypassing it

- Double-check the domain mapping (dvwa.local) resolves correctly

Issue: SafeLine UI not loading
Fix:

- Make sure you're using https://<Ubuntu-IP>:9443
- Check the service status and logs if needed
- Accept the browser warning for the self-signed SSL certificate

11.2 Security Best Practices

- **Isolate the lab** from your main network (use host-only or separate VLANs if possible)
- **Don't reuse lab passwords** anywhere else
- **Keep your lab up-to-date** - run system updates regularly on both Kali and Ubuntu
- **Take snapshots** in VirtualBox so you can roll back after making changes or breaking something

Lab Maintenance Checklist

- Apply system updates once a month
- Backup important configs (WAF policies, Apache settings, etc.)
- Periodically test WAF rules to make sure they still work
- Add new vulnerable apps or services to keep learning

This lab is a solid launchpad for anyone serious about web application security. It's also flexible you can evolve it as your skills grow. Use it to test tools, attacks, policies, and ideas without fear of damaging anything important.

When something breaks (and it will), treat it as a learning moment. That's how you get sharp.

12 References

This lab guide was created as a personal rewrite and walkthrough based on the structure and steps shown in the original work by **Royden Rahul Rebello**.

- Original video tutorial:
  Building a Web Application Firewall Home Lab using SafeLine WAF – by Royden Rahul Rebello

His project served as the core inspiration for this setup. This document rephrases and reorganizes the same content with a different style and structure for personal learning and documentation purposes.