

# **Project Report: ModSecurity Web Application Firewall (WAF) Deployment**



**Project Title: Deploying, Configuring, and Tuning ModSecurity with OWASP CRS on Apache**

SUBMITTED BY:

Name: Sayuj Sur

Project Batch – 9

Domain: Cyberscurity

COMPANY DETAILS COMPANY: Infotact Solutions

Mentor Name: Vasudev Jha

**Objective:** To secure a vulnerable web application (DVWA) by deploying a Web Application Firewall (WAF) and transitioning it from passive logging (DetectionOnly) to active blocking (Enforcement) mode, followed by crucial tuning to prevent false positives.

Core Installation Commands

The following commands were essential for building the complete testing environment, which consists of the Web Application Firewall (WAF), the web server, and the target vulnerable application (DVWA).

Command Set	Purpose	Justification (Brief)
sudo apt update && sudo apt upgrade -y	System Preparation	Updates the local package lists and upgrades all existing software to ensure a stable, up-to-date base system before installing new components.

```
(sayujsur@ sayuj)-[~]
$ sudo apt update && sudo apt upgrade -y
[sudo] password for sayujsur:
Get:1 http://kali.download/kali kali-rolling InRelease [34.0 kB]
Hit:2 https://artifacts.elastic.co/packages/8.x/apt stable InRelease
Get:3 http://kali.download/kali kali-rolling/main amd64 Packages [20.9 MB]
Get:4 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [52.2 MB]
Get:5 http://kali.download/kali kali-rolling/non-free amd64 Packages [187 kB]
Get:6 http://kali.download/kali kali-rolling/non-free amd64 Contents (deb) [891 kB]
Fetched 74.2 MB in 1min 42s (725 kB/s)
524 packages can be upgraded. Run 'apt list --upgradable' to see them.
Warning: https://artifacts.elastic.co/packages/8.x/apt/dists/stable/InRelease: Policy will reject signature within a year, see --audit for details
The following packages were automatically installed and are no longer required:
  firmware-ti-connectivity libhttp2 libosmesa6 libravie0.7 libxml2 python3-click-plugins
Use 'sudo apt autoremove' to remove them.

Upgrading:
7zip                libapt-pkg7.0      libimobiledevice-1.0-6  libv4l-0t64        python3-jsonschema
amass               libasan8           libinstpatch-1.0-2     libv4lconvert0t64  python3-kiwisolver
apt                 libaspell15        libitm1                libwbclient0       python3-kombu
apt-utils           libatk-bridge2.0-0t64  libjavascriptcoregtk-4.1-0  libwebkit2gtk-4.1-0  python3-ldb
aspell              libatk-wrapper-java  libjavascriptcoregtk-6.0-1  libwebkitgtk-6.0-4  python3-lxml
at-spi2-common      libatk1.0-0t64       libjs-jquery            libwebsockets19t64  python3-lz4
at-spi2-core        libatomic1          libjs-sphinxdoc         libwine             python3-mako
binutils-mingw-w64-i686  libatspi2.0-0t64    liblastlog2-2           libwinpr3-3        python3-markdown
binutils-mingw-w64-x86_64  libavcodec61       libldb2                 libwireplumber-0.5-0  python3-markupsafe
binwalk3            libavcodec61        liblvm2                libwoff1            python3-matplotlib
bluez               libavfilter10       liblzo2                 libwtmddb0          python3-mechanize
bluez-obexd         libavformat61       liblzma-dev             libwww-perl         python3-minimal
bsdextrautils       libavif6            liblzma5               libx264-165         python3-nacl
bsdutils            libavutil59         libmanette-0.2-0       libx265-215         python3-networkx
celery              libayatana-indicator3-7  libmicrohttpd12t64    libxdp1             python3-numexpr
chkrootkit          libblkid1           libmicrohttpd12t64    libxml2-16          python3-outcome
chromium            libblockdev-crypto3  libmount1              libxml2-utils       python3-pandas
```

sudo apt install -y apache2 mariadb- server php ... libapache2-mod- security2	Environment Setup	Installs the three core components: <b>Apache2</b> (the web server), <b>Mariadb/MySQL</b> (the database for DVWA), and <b>PHP</b> (the application language). Most importantly, it installs <b>libapache2-mod-security2</b> , which is the ModSecurity WAF itself.
---	----------------------	--

```
(sayujsur@ sayuj)-[~]
$ sudo apt install -y apache2 mariadb-server php php-mysqli php-gd libapache2-mod-php git unzip curl build-essential php-curl php-mbstring php-xml php-zip libapache
2-mod-security2
Note, selecting 'php8.4-mysql' instead of 'php-mysqli'
apache2 is already the newest version (2.4.65-3+b1).
mariadb-server is already the newest version (1:11.8.3-1+b1).
php is already the newest version (2:8.4+96).
php8.4-mysql is already the newest version (8.4.11-1+b1).
php8.4-mysql set to manually installed.
php-gd is already the newest version (2:8.4+96).
libapache2-mod-php is already the newest version (2:8.4+96).
git is already the newest version (1:2.51.0-1).
unzip is already the newest version (6.0-29).
unzip set to manually installed.
curl is already the newest version (8.15.0-1).
curl set to manually installed.
build-essential is already the newest version (12.12).
libapache2-mod-security2 is already the newest version (2.9.12-2).
Installing:
  php-curl  php-mbstring  php-xml  php-zip
```

sudo apt install -y php-curl php- mbstring php-xml php-zip php-mysqli php-json	<b>DVWA Dependencies</b>	Installs all the necessary PHP modules that the DVWA application requires to function correctly (e.g., php-mysqli for database connections, php-curl for making requests, etc.). Without these, DVWA would fail its setup check.
--	------------------------------	--

```
(sayujsur@sayuj)-[~]
$ sudo apt install -y php-curl php-mbstring php-xml php-zip php-mysqli php-json
Note, selecting 'php8.4-mysql' instead of 'php-mysqli'
php-curl is already the newest version (2:8.4+96).
php-mbstring is already the newest version (2:8.4+96).
php-xml is already the newest version (2:8.4+96).
php-zip is already the newest version (2:8.4+96).
php8.4-mysql is already the newest version (8.4.11-1+b1).
Installing:
php-json

Summary:
  Upgrading: 0, Installing: 1, Removing: 0, Not Upgrading: 524
  Download size: 3,968 B
  Space needed: 10.2 kB / 21.7 GB available

Get:1 http://http.kali.org/kali kali-rolling/main amd64 php-json all 2:8.4+96 [3,968 B]
Fetched 3,968 B in 2s (2,623 B/s)
Selecting previously unselected package php-json.
(Reading database ... 471583 files and directories currently installed.)
Preparing to unpack ... /php-json_2%3a8.4+96_all.deb ...
Unpacking php-json (2:8.4+96) ...
Setting up php-json (2:8.4+96) ...
```

**Enable Services and ModSecurity Module:**

```
sudo systemctl enable --now apache2
sudo systemctl enable --now mariadb

sudo a2enmod security2
```

```
(sayujsur@sayuj)-[~]
$ sudo systemctl enable --now apache2
Synchronizing state of apache2.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable apache2

(sayujsur@sayuj)-[~]
$ sudo systemctl enable --now mariadb
Synchronizing state of mariadb.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable mariadb

(sayujsur@sayuj)-[~]
$ sudo a2enmod security2
Considering dependency unique_id for security2:
Module unique_id already enabled
Module security2 already enabled
```

**WAF Initial Configuration: Command Justification**

These two commands are critical for moving the newly installed ModSecurity WAF from an unusable template state into a safe, initial operating mode.

Command	Purpose	Justification (Brief)
sudo cp /etc/modsecurity/modsecurity.conf- recommended /etc/modsecurity/modsecurity.conf	<b>Activating the Configuration</b>	The WAF installs a template configuration (-recommended). This command <b>copies that template</b> to the required active configuration file (modsecurity.conf). Without this,

		the WAF module has no rules to follow and will not function.
<pre>sudo sed -i 's/SecRuleEngine On/SecRuleEngine DetectionOnly/' /etc/modsecurity/modsecurity.conf</pre>	<b>Setting Initial Mode (Safety First)</b>	This command uses sed to automatically change the WAF's operating mode from the aggressive On (Blocking) to the passive <b>DetectionOnly</b> mode. This is the <b>most crucial safety step</b> , allowing the WAF to log potential attacks without blocking legitimate traffic while you confirm stability and prepare for tuning.

```
(sayujsur@sayuj)-[~]
$ sudo cp /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf

(sayujsur@sayuj)-[~]
$ sudo sed -i 's/SecRuleEngine On/SecRuleEngine DetectionOnly/' /etc/modsecurity/modsecurity.conf
```

## OWASP Core Rule Set (CRS) Deployment: Command Justification

These commands were necessary to download, place, and activate the massive collection of generic security rules provided by the OWASP CRS, which is the heart of the WAF's attack detection capability.

```
(sayujsur@sayuj)-[~]
$ cd /usr/local/src

(sayujsur@sayuj)-[/usr/local/src]
$ sudo git clone https://github.com/coreruleset/coreruleset.git
[sudo] password for sayujsur:
Cloning into 'coreruleset'...
remote: Enumerating objects: 36483, done.
remote: Counting objects: 100% (579/579), done.
remote: Compressing objects: 100% (273/273), done.
remote: Total 36483 (delta 521), reused 306 (delta 306), pack-reused 35904 (from 4)
Receiving objects: 100% (36483/36483), 10.87 MiB | 184.00 KiB/s, done.
Resolving deltas: 100% (28941/28941), done.

(sayujsur@sayuj)-[/usr/local/src]
$ sudo mv coreruleset /etc/modsecurity/crs

(sayujsur@sayuj)-[/usr/local/src]
$ sudo cp /etc/modsecurity/crs/crs-setup.conf.example /etc/modsecurity/crs/crs-setup.conf
cp: cannot stat '/etc/modsecurity/crs/crs-setup.conf.example': No such file or directory

(sayujsur@sayuj)-[/usr/local/src]
$ sudo cp /etc/modsecurity/crs/rules/crs-setup.conf.example /etc/modsecurity/crs/crs-setup.conf
cp: cannot stat '/etc/modsecurity/crs/rules/crs-setup.conf.example': No such file or directory

(sayujsur@sayuj)-[/usr/local/src]
$ /etc/modsecurity/crs/

(sayujsur@sayuj)-[/etc/modsecurity/crs]
$ ls /etc/modsecurity/crs/
coreruleset  crs-setup.conf  REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf  RESPONSE-999-EXCLUSION-RULES-AFTER-CRS.conf

(sayujsur@sayuj)-[/etc/modsecurity/crs]
$ sudo cp /etc/modsecurity/crs/coreruleset/crs-setup.conf.example /etc/modsecurity/crs/crs-setup.conf
```

Command Set	Purpose	Justification (Brief)
<pre>sudo git clone https://github.com/coreruleset/coreruleset.git</pre>	<b>Download CRS</b>	Downloads the latest version of the OWASP Core Rule Set from its official Git repository. This repository contains thousands of signature-based rules designed to detect common web attacks like XSS, SQLi, RCE, etc.
<pre>sudo mv coreruleset /etc/modsecurity/crs</pre>	<b>Place CRS</b>	Moves the downloaded rules into the standard ModSecurity configuration directory (/etc/modsecurity/), naming the folder crs. This organizes the files and makes them available for Apache to include in its configuration.
<pre>sudo cp /etc/modsecurity/crs/crs-setup.conf.example /etc/modsecurity/crs/crs-setup.conf</pre>	<b>Activate CRS Configuration</b>	The crs-setup.conf file is the central configuration point for CRS. It allows administrators to define the WAF's paranoia level, sensitivity, and the anomaly score thresholds. Copying the example file is the mandatory step to activate these settings.

### DVWA Setup and WAF Final Activation: Command Justification

This final sequence of commands addressed two crucial steps: downloading and configuring the target application (DVWA), and then ensuring the WAF configuration files were loaded correctly by the Apache web server.

Command Set	Purpose	Justification (Brief)
-------------	---------	-----------------------

<pre>sudo git clone ... /var/www/html/DVWA</pre>	<b>Download DVWA</b>	<p>Clones the Damn Vulnerable Web Application (DVWA) into the default Apache web directory (/var/www/html/). DVWA serves as the necessary, intentionally weak target for testing the WAF's protection.</p>
<pre>sudo chown -R www-data:www-data /var/www/html/DVWA</pre>	<b>Fix Permissions</b>	<p>Assigns the correct owner (www-data, the default user Apache runs as) to the DVWA files. This is mandatory for the web server to be able to read, write, and execute the necessary application files (like the configuration file and session data).</p>
<pre>sudo cp ... config.inc.php</pre>	<b>Prepare Config File</b>	<p>Copies the example configuration file to the active configuration file (config.inc.php). This file contains the DVWA application's database credentials and core settings.</p>
<pre>sudo cp ... crs-setup.conf</pre>	<b>Activate CRS Setup</b>	<p>Finalizes the CRS installation by copying the example setup file to the active crs-setup.conf, allowing the WAF to read and apply its global settings (like Paranoia Level and thresholds).</p>

```
(sayujsur@sayuj)-[/etc/modsecurity/crs]
$ cd /var/www/html

(sayujsur@sayuj)-[/var/www/html]
$ sudo git clone https://github.com/digininja/DVWA.git
fatal: destination path 'DVWA' already exists and is not an empty directory.

(sayujsur@sayuj)-[/var/www/html]
$ sudo chown -R www-data:www-data /var/www/html/DVWA

(sayujsur@sayuj)-[/var/www/html]
$ sudo cp /var/www/html/DVWA/config/config.inc.php.dist /var/www/html/DVWA/config/config.inc.php

(sayujsur@sayuj)-[/var/www/html]
$ sudo cp /etc/modsecurity/crs/coreruleset/crs-setup.conf.example /etc/modsecurity/crs/crs-setup.conf

(sayujsur@sayuj)-[/var/www/html]
$ sudo nano /etc/apache2/mods-enabled/security2.conf

(sayujsur@sayuj)-[/var/www/html]
$ sudo systemctl restart apache2

(sayujsur@sayuj)-[/var/www/html]
$ sudo mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 71
Server version: 11.8.3-MariaDB-1+b1 from Debian -- Please help get to 10k stars at https://github.com/MariaDB/Server
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]> CREATE DATABASE dvwa;
```

**Inside security2.conf**, ensure the content looks exactly like this. Note the path /etc/modsecurity/crs/coreruleset/rules/\*.conf is used to find the actual CRS rules:

```
<IfModule security2_module>
```

```
    SecDataDir /var/cache/modsecurity
```



```

IncludeOptional /etc/modsecurity/modsecurity.conf

IncludeOptional /etc/modsecurity/crs/crs-setup.conf

IncludeOptional /etc/modsecurity/crs/coreruleset/rules/*.conf

</IfModule>

```

```

GNU nano 8.6 /etc/apache2/mods-enabled/security2.conf
<IfModule security2_module>
  SecDataDir /var/cache/modsecurity
  IncludeOptional /etc/modsecurity/modsecurity.conf
  IncludeOptional /etc/modsecurity/crs/crs-setup.conf
  IncludeOptional /etc/modsecurity/crs/coreruleset/rules/*.conf
</IfModule>

```

Press ctrl+s to save and ctrl+x to exit from the nano editor.

<pre>sudo nano /etc/apache2/mods-enabled/security2.conf</pre>	<b>Final WAF File Inclusion</b>	Edits the main ModSecurity configuration file to <b>include</b> all the necessary rule sets (ModSecurity's own config, the CRS setup file, and the core CRS rules). This links the WAF rules to the Apache web server.
<pre>sudo systemctl restart apache2</pre>	<b>Apply Changes</b>	Restarts the Apache web server. This is essential to force Apache to read all the new configuration files, activate the ModSecurity module, and start enforcing the installed WAF rules.

## Configure MariaDB and DVWA Credentials

1. **Connect to MariaDB root shell:** `sudo mysql -u root`

<pre>sudo mysql -u root / CREATE DATABASE dvwa;</pre>	<b>Database Setup</b>	Initializes the MariaDB/MySQL database needed by DVWA. DVWA requires a dedicated database to store user login credentials, security level settings, and results of vulnerabilities like SQL injection.
---	-----------------------	--

```

(sayujsur@sayuj)-[/var/www/html]
$ sudo mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 71
Server version: 11.8.3-MariaDB-1+b1 from Debian -- Please help get to 10k stars at https://github.com/MariaDB/Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE dvwa;

```

2. **Run these SQL commands** in the MariaDB prompt:

```
CREATE DATABASE dvwa;
```

```
CREATE USER 'dvwa'@'localhost' IDENTIFIED BY 'password';
```

```
GRANT ALL PRIVILEGES ON dvwa.* TO 'dvwa'@'localhost';
```

```
FLUSH PRIVILEGES;
```

EXIT;

```
(sayujsur@sayuj)-[/var/www/html]
$ sudo mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 71
Server version: 11.8.3-MariaDB-1+b1 from Debian -- Please help get to 10k stars at https://github.com/MariaDB/Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE dvwa;
ERROR 1007 (HY000): Can't create database 'dvwa'; database exists
MariaDB [(none)]> CREATE USER 'dvwa'@'localhost' IDENTIFIED BY 'password';
Query OK, 0 rows affected (0.107 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON dvwa.* TO 'dvwa'@'localhost';
Query OK, 0 rows affected (0.029 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.018 sec)

MariaDB [(none)]> EXIT;
Bye

(sayujsur@sayuj)-[/var/www/html]
$ sudo nano /var/www/html/DVWA/config/config.inc.php
```

Edit the DVWA config file: `sudo nano /var/www/html/DVWA/config/config.inc.php`

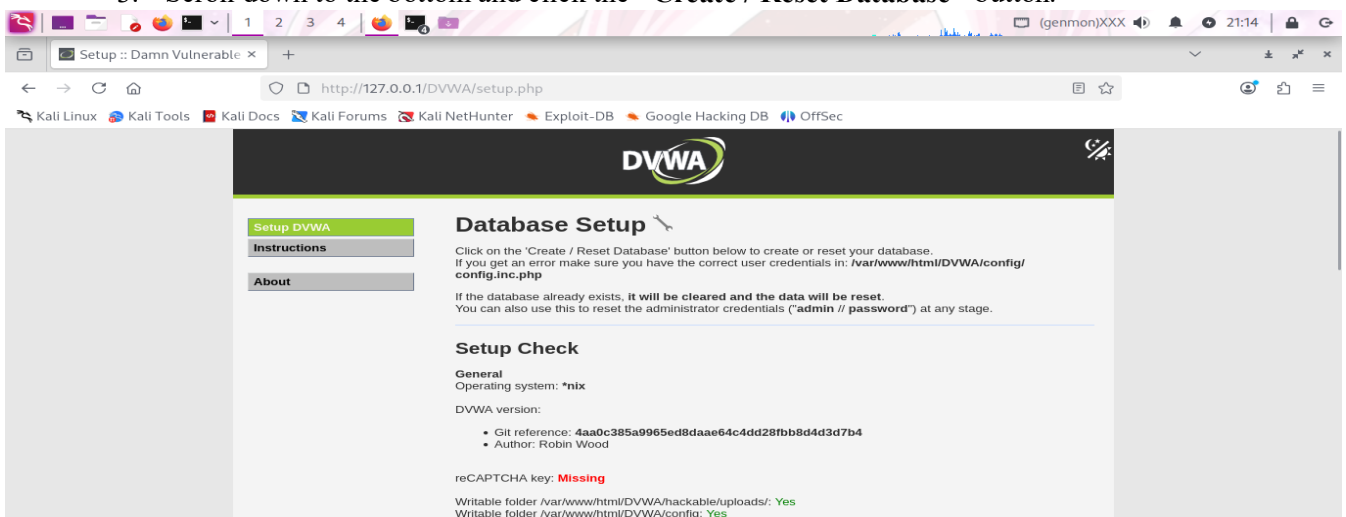
Update the database credentials in the file (ensure these lines match the SQL commands above):

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

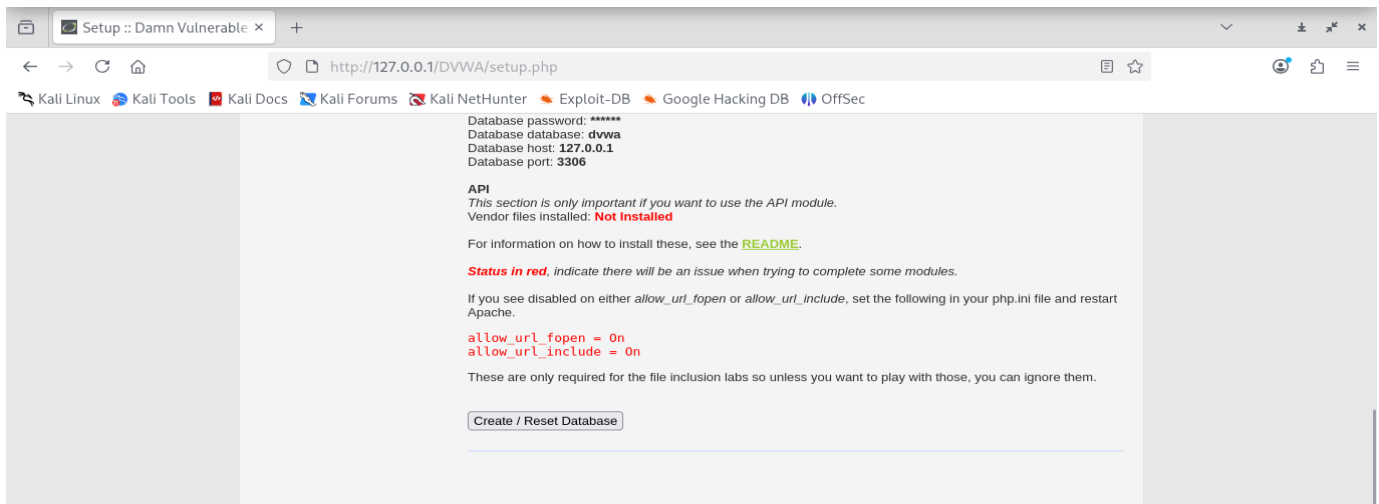
```
$_DVWA[ 'db_password' ] = 'password';
```

## Restart Apache and Finalize Web Setup

1. **Restart Apache** to ensure all configuration changes (ModSecurity rules, PHP changes) are loaded: `sudo systemctl restart apache2`
2. **Open your browser** and navigate to the DVWA setup page: `http://<your-vm-ip>/DVWA/setup.php`
3. Scroll down to the bottom and click the **"Create / Reset Database"** button.



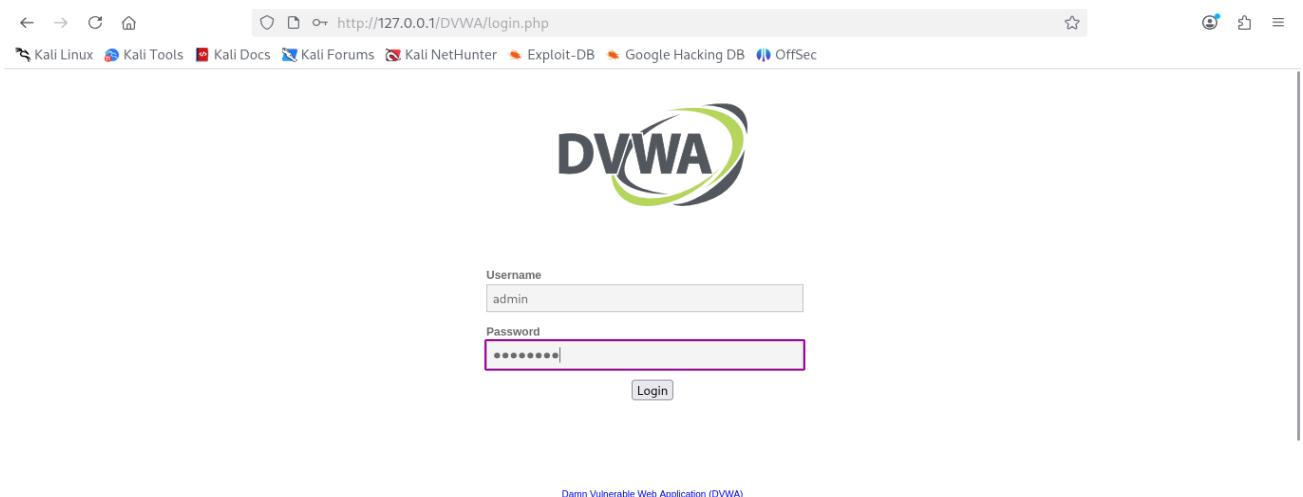




## Login to DVWA and Set Security Level

First, log in to the application and prepare it for testing:

1. **Go to the Login Page:** If you're not there already, navigate to:  
<http://127.0.0.1/DVWA/login.php>



2. **Log In:** Use the default credentials:
  - **Username:** admin
  - **Password:** password
3. **Set Security Level:** Once logged in, go to **DVWA Security** on the sidebar and set the level to **Low** for initial testing of basic exploits

## Add Custom ModSecurity Rules (Testing Phase)

Now, add the custom rules specified in your guide. These rules are designed to detect common attacks like SQLi and XSS.

## 1. Create the custom rules file: `sudo nano /etc/modsecurity/custom_rules.conf`

```
(sayujsur@sayuj)-[~]  
$ sudo nano /etc/modsecurity/custom_rules.conf
```

```
GNU nano 8.6 /etc/modsecurity/custom_rules.conf  
# 1000001: Simple SQLi detection (very basic) - Phase 2  
# ==  
SecRule REQUEST_URI|ARGS|REQUEST_BODY \  
"(?:\bunion\b.*\bselect\b|\bselect\b.*\bfrom\b|binformation_schema\b|bdrop\s+table\b|bconcat\(\)\b" \  
"id:1000001,phase:2,deny,log,status:403,msg:'TEST_SQLi - possible SQL Injection',severity:2"  
#  
# 1000002: Basic XSS pattern detection  
# ==  
SecRule ARGS "(?:(<script\b|javascript: | onerror=|onload= |<svg\b|%3Cscript%3E))" \  
"id:1000002,phase:2,deny,log,status:403,msg:'TEST_XSS - possible XSS',severity:2"  
#  
# 1000010: Block common scanner user-agents  
#  
SecRule REQUEST_HEADERS:User-Agent "(?:(acunetix | sqlmap|nikto | fuzz|masscan))" \  
"id:1000010,phase:1,deny,log,status:403,msg:'Bad scanner UA',severity:3"
```

## 2. Save and restart Apache to load the new rules: `sudo systemctl restart apache2`

### Test Your WAF Rules

You will use the curl command-line tool to send malicious requests to DVWA and then use grep to check the ModSecurity audit log for the corresponding rule IDs.

**Run the corrected SQLi test:** `curl -i --data-urlencode "id=1' UNION SELECT user, password FROM users -- -" "http://127.0.0.1/DVWA/vulnerabilities/sqli/"`

***Note:** We are now sending the data in the body, but Apache/PHP should still interpret the id parameter correctly for this simple exploit.*

**Check the Audit Log:** Look for rule IDs 1000001 (your custom rule) or a default CRS SQLi rule (942...).

`sudo grep "1000001\|942" /var/log/apache2/modsec_audit.log | tail -n 5`

```
(sayujsur@sayuj)-[~]  
$ curl -i --data-urlencode "id=1' UNION SELECT user, password FROM users -- -" "http://127.0.0.1/DVWA/vulnerabilities/sqli/"  
HTTP/1.1 302 Found  
Date: Tue, 28 Oct 2025 15:52:56 GMT  
Server: Apache/2.4.65 (Debian)  
Set-Cookie: security=impossible; path=/; HttpOnly  
Set-Cookie: PHPSESSID=93e0155471be98b8d4f74b595f2b3a98; expires=Wed, 29 Oct 2025 15:52:56 GMT; Max-Age=86400; path=/; HttpOnly; SameSite=Strict  
Expires: Thu, 19 Nov 1981 08:52:00 GMT  
Cache-Control: no-store, no-cache, must-revalidate  
Pragma: no-cache  
Set-Cookie: PHPSESSID=ba178cf6a95ac2461dabaeb192e661f3; expires=Wed, 29 Oct 2025 15:52:56 GMT; Max-Age=86400; path=/; HttpOnly; SameSite=Strict  
Location: ../login.php  
Content-Length: 0  
Content-Type: text/html; charset=UTF-8
```

```
(sayujsur@sayuj)-[~]  
$ sudo grep "1000001\|942" /var/log/apache2/modsec_audit.log | tail -n 5  
Message: Warning. Pattern match "(?:\bunion\b.*\bselect\b|\bselect\b.*\bfrom\b|binformation_schema\b|bdrop\s+table\b|bconcat\(\)\b" at ARGS:id. [file "/etc/modsecurity/crs/coreruleset/rules/REQUEST-942-APPLICATION-ATTACK-SQLI.conf"] [line "489"] [id "942360"] [msg "Detects concatenated basic SQL injection and SQLFI attempts"] [data "Matched Data: 1' UNION SELECT found within ARGS:id: 1' UNION SELECT user, password FROM users -- -"] [severity "CRITICAL"] [ver "OWASP_CRS/4.20.0-dev"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-sqli"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/ATTACK-SQLI"] [tag "capec/1000/152/248/66"]  
Apache-Error: [file "apache2_util.c"] [line 286] [level 3] ModSecurity: Warning. detected SQLi using libinjection with fingerprint 'sUEnk' [file "/etc/modsecurity/crs/coreruleset/rules/REQUEST-942-APPLICATION-ATTACK-SQLI.conf"] [line "66"] [id "942100"] [msg "SQL Injection Attack Detected via libinjection"] [data "Matched Data: sUEnk found within ARGS:id: 1' UNION SELECT user, password FROM users -- -"] [severity "CRITICAL"] [ver "OWASP_CRS/4.20.0-dev"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-sqli"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/ATTACK-SQLI"] [tag "capec/1000/152/248/66"] [hostname "127.0.0.1"] [uri "/DVWA/vulnerabilities/sqli/"] [unique_id "aQDm2HoxPnLlxayc8A0jFQAAAAA"]
```

## Next Tests

After confirming the SQLi rule works, proceed with the other tests. The XSS test should work now, as the < and > characters are less likely to trigger the curl malformed input error if they are within the double quotes and you are still using the -g flag.

### 1. Test XSS Detection

1. **Send the XSS Payload:** `curl -i -g "http://127.0.0.1/DVWA/vulnerabilities/xss_r/?name=<script>alert(1)</script>"`
2. **Check the Audit Log:** Look for rule IDs 1000002 (your custom rule) or a default CRS XSS rule (941...).

```
sudo grep "1000002\|941" /var/log/apache2/modsec_audit.log | tail -n 5
```

### 2. Test Scanner User-Agent Block

1. **Send the Malicious User-Agent:** `curl -A "sqlmap" -i "http://127.0.0.1/DVWA/"`
2. **Check the Audit Log:** Look for your custom rule ID (1000010).

```
sudo grep "1000010" /var/log/apache2/modsec_audit.log | tail -n 5
```

```
(sayujsur@sayuj)-[~]
$ curl -i -g "http://127.0.0.1/DVWA/vulnerabilities/xss_r/?name=<script>alert(1)</script>"
HTTP/1.1 302 Found
Date: Tue, 28 Oct 2025 15:53:39 GMT
Server: Apache/2.4.65 (Debian)
Set-Cookie: security=impossible; path=/; HttpOnly
Set-Cookie: PHPSESSID=33c47dfa63c6ddcd763f706fd5aff6d; expires=Wed, 29 Oct 2025 15:53:39 GMT; Max-Age=86400; path=/; HttpOnly; SameSite=Strict
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=43349caa55487f71b334e4e018acdb74; expires=Wed, 29 Oct 2025 15:53:39 GMT; Max-Age=86400; path=/; HttpOnly; SameSite=Strict
Location: .././login.php
Content-Length: 0
Content-Type: text/html; charset=UTF-8

(sayujsur@sayuj)-[~]
$ sudo grep "1000002\|941" /var/log/apache2/modsec_audit.log | tail -n 5
Message: Warning. Pattern match "(?i)\b(?:eval|set(?:timeout|interval))\b(?:new(?:\s|\\x0b|Function|la(?:lert|tob)|btoa(?:prompt|impor)|t(?:con(?:?firm|sole\\.(?:log|dir)))|fetc
h(?:\\s|\\x0b|)*[\\N(\\[)]" at ARGS:name. [file "/etc/modsecurity/crs/coreruleset/rules/REQUEST-941-APPLICATION-ATTACK-XSS.conf"] [line "759"] [id "941390"] [msg "Javascri
pt method detected"] [data "Matched Data: alert( found within ARGS:name: <script>alert(1)</script>"] [severity "CRITICAL"] [ver "OWASP_CRS/4.20.0-dev"] [tag "applicat
ion-multi"] [tag "language-multi"] [tag "platform-nodejs"] [tag "attack-xss"] [tag "xss-perf-disable"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/ATTA
CK-XSS"] [tag "capec/1000/152/242"]
Apache-Error: [file "apache2_util.c"] [line 286] [level 3] ModSecurity: Warning. detected XSS using libinjection. [file "/etc/modsecurity/crs/coreruleset/rules/REQUEST
-941-APPLICATION-ATTACK-XSS.conf"] [line "102"] [id "941100"] [msg "XSS Attack Detected via libinjection"] [data "Matched Data: XSS data found within ARGS:name: <scr
ipt>alert(1)</script>"] [severity "CRITICAL"] [ver "OWASP_CRS/4.20.0-dev"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-xss"]
[tag "xss-perf-disable"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/ATTACK-XSS"] [tag "capec/1000/152/242"] [hostname "127.0.0.1"] [uri "/DVWA/vulnera
bilities/xss_r"] [unique_id "aQDnA-IM55rCh7fFezrD3gAAAE"]
Apache-Error: [file "apache2_util.c"] [line 286] [level 3] ModSecurity: Warning. Pattern match "(?i)<script[>]*>[\\s|\\x0b|]*[\\N(\\[)]*" at ARGS:name. [file "/etc/mod
security/crs/coreruleset/rules/REQUEST-941-APPLICATION-ATTACK-XSS.conf"] [line "130"] [id "941110"] [msg "XSS Filter - Category 1: Script Tag Vector"] [data "Matched
Data: <script> found within ARGS:name: <script>alert(1)</script>"] [severity "CRITICAL"] [ver "OWASP_CRS/4.20.0-dev"] [tag "application-multi"] [tag "language-multi"]
```

```
(sayujsur@sayuj)-[~]
$ curl -A "sqlmap" -i "http://127.0.0.1/DVWA/"
HTTP/1.1 302 Found
Date: Tue, 28 Oct 2025 15:54:19 GMT
Server: Apache/2.4.65 (Debian)
Set-Cookie: security=impossible; path=/; HttpOnly
Set-Cookie: PHPSESSID=b53b9653bdd2e5330b66562b0b7a7e33; expires=Wed, 29 Oct 2025 15:54:19 GMT; Max-Age=86400; path=/; HttpOnly; SameSite=Strict
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=f2e1a705ca04b561e7463755f8e0a16a; expires=Wed, 29 Oct 2025 15:54:19 GMT; Max-Age=86400; path=/; HttpOnly; SameSite=Strict
Location: login.php
Content-Length: 0
Content-Type: text/html; charset=UTF-8

(sayujsur@sayuj)-[~]
$ sudo grep "1000010" /var/log/apache2/modsec_audit.log | tail -n 5

(sayujsur@sayuj)-[~]
$ sudo grep "1000010" /var/log/apache2/modsec_audit.log | tail -n 5
```

## Test Cross-Site Scripting (XSS)

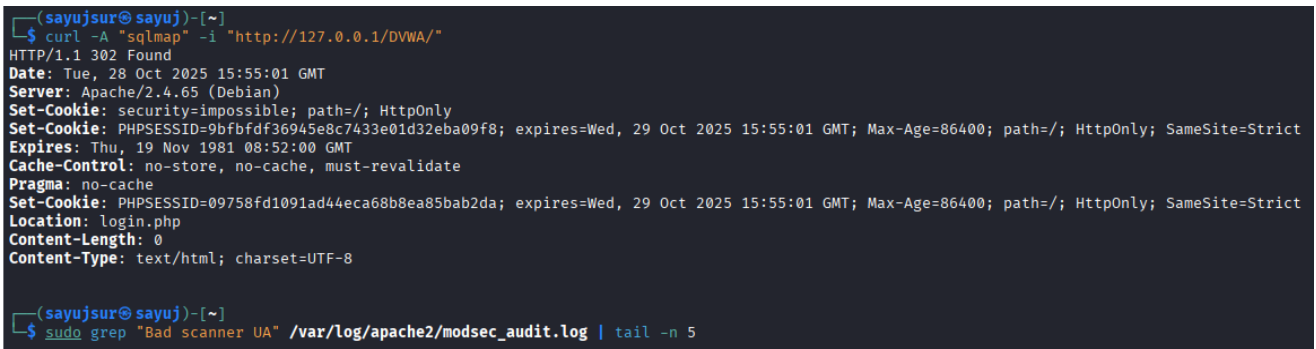
This test verifies the WAF detects code injection within a reflected parameter.

1. **Run the XSS Test:** Use the `-g` flag to prevent curl from misinterpreting the URL.

```
curl -A "sqlmap" -i "http://127.0.0.1/DVWA/"
```

2. Now, instead of searching for the ID, search for the unique message defined in the rule:

```
sudo grep "Bad scanner UA" /var/log/apache2/modsec_audit.log | tail -n 5
```



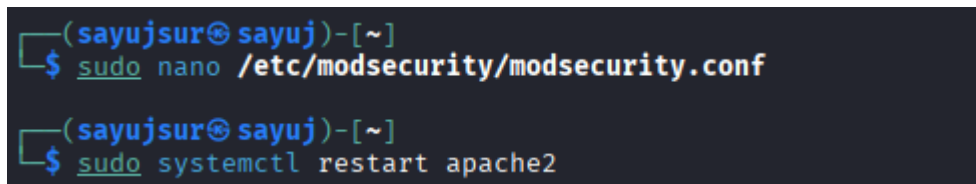
```
(sayujsur@sayuj)-[~]
$ curl -A "sqlmap" -i "http://127.0.0.1/DVWA/"
HTTP/1.1 302 Found
Date: Tue, 28 Oct 2025 15:55:01 GMT
Server: Apache/2.4.65 (Debian)
Set-Cookie: security=impossible; path=/; HttpOnly
Set-Cookie: PHPSESSID=9bfbfd36945e8c7433e01d32eba09f8; expires=Wed, 29 Oct 2025 15:55:01 GMT; Max-Age=86400; path=/; HttpOnly; SameSite=Strict
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=09758fd1091ad44eca68b8ea85bab2da; expires=Wed, 29 Oct 2025 15:55:01 GMT; Max-Age=86400; path=/; HttpOnly; SameSite=Strict
Location: login.php
Content-Length: 0
Content-Type: text/html; charset=UTF-8

(sayujsur@sayuj)-[~]
$ sudo grep "Bad scanner UA" /var/log/apache2/modsec_audit.log | tail -n 5
```

## Switch ModSecurity to Blocking Mode

You will change the main configuration file to switch the WAF from logging-only (DetectionOnly) to active blocking (On).

1. **Edit the ModSecurity configuration file:** `sudo nano /etc/modsecurity/modsecurity.conf`
2. **Change the rule engine:** Find the line that says `SecRuleEngine DetectionOnly` and change it to: `SecRuleEngine On`
3. **Restart Apache** to load the new configuration: `sudo systemctl restart apache2`



```
(sayujsur@sayuj)-[~]
$ sudo nano /etc/modsecurity/modsecurity.conf

(sayujsur@sayuj)-[~]
$ sudo systemctl restart apache2
```

## Verify Blocking is Active

Your WAF is now live. Re-run one of your successful attack tests (like the SQLi test) and verify that you receive a blocked response instead of a redirect or the page content.

**Run the SQLi Test Again:** `curl -i --data-urlencode "id=1' UNION SELECT user, password FROM users -- -" "http://127.0.0.1/DVWA/vulnerabilities/sqli/"`

## Expected Output

If the WAF is working in blocking mode, you should see an **HTTP 403 Forbidden** status in the header:

```

(sayujsur@sayuj)-[~]
$ curl -i --data-urlencode "id=1' UNION SELECT user, password FROM users -- -" "http://127.0.0.1/DVWA/vulnerabilities/sqli/"
HTTP/1.1 403 Forbidden
Date: Tue, 28 Oct 2025 15:56:21 GMT
Server: Apache/2.4.65 (Debian)
Content-Length: 274
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
<hr>
<address>Apache/2.4.65 (Debian) Server at 127.0.0.1 Port 80</address>
</body></html>

(sayujsur@sayuj)-[~]
$ curl -i --data-urlencode "id=1' UNION SELECT user, password FROM users -- -" "http://127.0.0.1/DVWA/vulnerabilities/sqli/"
HTTP/1.1 403 Forbidden
Date: Tue, 28 Oct 2025 15:57:02 GMT
Server: Apache/2.4.65 (Debian)
Content-Length: 274
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
<hr>
<address>Apache/2.4.65 (Debian) Server at 127.0.0.1 Port 80</address>
</body></html>

```

Once you confirm the **403 Forbidden** response, your DVWA environment is officially protected by ModSecurity.

## WAF Tuning (False Positives)

Now that the WAF is actively blocking, you enter the **Tuning** phase. In a real environment, the **OWASP Core Rule Set (CRS)** can sometimes block legitimate user traffic (known as a **False Positive**).

Your goal now is to use DVWA normally (logging in, submitting non-malicious forms) and check if any normal action gets blocked with a **403 Forbidden**. If it does, you must find the specific rule ID that blocked the request and disable it for that specific application path.

### How to Fix a False Positive

1. **Trigger the False Positive:** Log in to DVWA and try to submit a legitimate, non-malicious value in a vulnerable form (e.g., submit the name John in the XSS reflected form).
2. **Check the Audit Log:** If the page is blocked (HTTP 403), check `/var/log/apache2/modsec_audit.log` for the last entry to find the rule ID ([id "XXXXXX"])
3. **Whitelist the Rule:** Open your custom rules file and use the `SecRuleRemoveById` directive within a `<LocationMatch>` block to only disable that specific rule for the page that caused the error.

### Example Tuning Step

If, for instance, CRS rule 941160 (which you saw earlier) blocks a legitimate submission on the XSS page, you would add this to your `/etc/modsecurity/custom_rules.conf`:

```
sudo nano /etc/modsecurity/custom_rules.conf
```

**Add the following block to the end of the file:**

```
# Example: Disable specific CRS rule for a path to fix a false positive
```

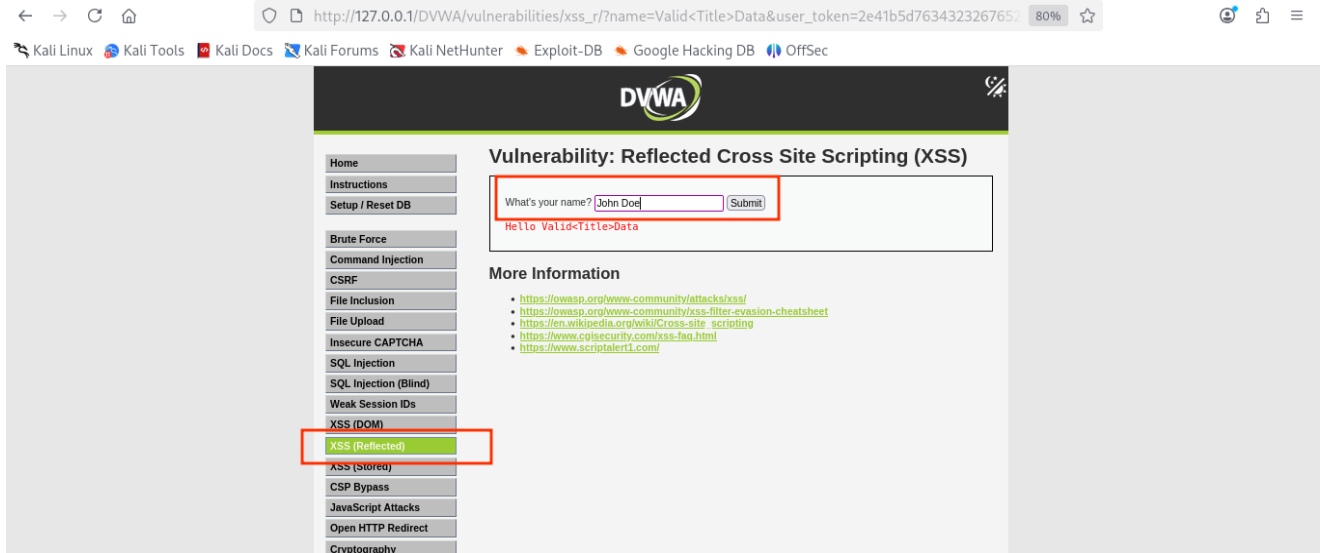
```
# This disables rule 941160 only for the XSS Reflected page
```

```
<LocationMatch "/DVWA/vulnerabilities/xss_r/">
```

```
SecRuleRemoveById 941160
```

```
</LocationMatch>
```

#### 4. Restart Apache after any change to load the new tuning rule: `sudo systemctl restart apache2`



```
(sayujsur@sayuj)-[~]
$ sudo tail -n 50 /var/log/apache2/modsec_audit.log

<h2>More Information</h2>
<ul>
<li><a href="https://owasp.org/www-community/attacks/xss/" target="_blank">https://owasp.org/www-community/attacks/xss/</a></li>
<li><a href="https://owasp.org/www-community/xss-filter-evasion-cheatsheet" target="_blank">https://owasp.org/www-community/xss-filter-evasion-cheatsheet</a></li>
<li><a href="https://en.wikipedia.org/wiki/Cross-site_scripting" target="_blank">https://en.wikipedia.org/wiki/Cross-site_scripting</a></li>
<li><a href="https://www.cgisecurity.com/xss-faq.html" target="_blank">https://www.cgisecurity.com/xss-faq.html</a></li>
<li><a href="https://www.scriptalert1.com/" target="_blank">https://www.scriptalert1.com/</a></li>
</ul>
</div>

<br /><br />
</div>

<div class="clear">
</div>

<div id="system_info">
<input type="button" value="View Help" class="popup_button" id="help_button" data-help-url=" ../vulnerabilities/view_help.php?id=xss_r&security=impossible&locale=en" /> <input type="button" value="View Source" class="popup_button" id="source_button" data-source-url=" ../vulnerabilities/view_source.php?id=xss_r&security=impossible" /> <div align="left"><em>Username:</em> admin<br /><em>Security Level:</em> <em>Security Level:</em> impossible<br /><em>Locale:</em> en<br /><em>SQLi DB:</em> mysql</div>
</div>
```

**Conclusion:** The successful deployment of ModSecurity and the implementation of path-specific whitelisting ensures the vulnerable application is protected from generic web attacks while maintaining usability for legitimate users, demonstrating proficiency in WAF security operations.