

# Project 2: WAF Rule Development Lab

This guide walks you through setting up **ModSecurity**, a Web Application Firewall (WAF), to block a basic SQL Injection attack.

## Step 1: Setup and Installation

1. **Install Apache & ModSecurity:** On a fresh Ubuntu Server VM, install the Apache web server and the ModSecurity module.

*sudo apt update*

```
sudo apt install -y apache2 libapache2-mod-security2
```

- ## 2. **Enable ModSecurity**: Enable the module and restart Apache.

*sudo a2enmod security2*

*sudo systemctl restart apache2*

```
Warning: you are using the root account. You may harm your system.

1 # -- Rule engine initialization -----
2
3 # Enable ModSecurity, attaching it to every transaction. Use detection
4 # only to start with, because that minimises the chances of post-installation
5 # disruption.
6 #
7 SecRuleEngine On
8
9
10 # -- Request body handling -----
11
12 # Allow ModSecurity to access request bodies. If you don't, ModSecurity
13 # won't be able to see any POST parameters, which opens a large security
14 # hole for attackers to exploit.
15 #
16 SecRequestBodyAccess On
17
18
19 # Enable XML request body parser.
20 # Initiate XML Processor in case of xml content-type
21 #
22 SecRule REQUEST_HEADERS:Content-Type "^(?:application(?:/soap\+|/)|text/)\xml" \
23     "id:'200000',phase:1,t:none,t:lowercase,pass,nolog,ctl:requestBodyProcessor=XML"
24
25 # Enable JSON request body parser.
26 # Initiate JSON Processor in case of JSON content-type; change accordingly
27 # if your application does not use 'application/json'
28 #
29 SecRule REQUEST_HEADERS:Content-Type "^application/json" \
30     "id:'200001',phase:1,t:none,t:lowercase,pass,nolog,ctl:requestBodyProcessor=JSON"
31
32 # Sample rule to enable JSON request body parser for more subtypes.
33 # Uncomment or adapt this rule if you want to engage the JSON
34 # Processor for "+json" subtypes
35 #
36 #SecRule REQUEST_HEADERS:Content-Type "application/[a-z0-9.-]+[+]json" \
37 #     "id:'200006',phase:1,t:none,t:lowercase,pass,nolog,ctl:requestBodyProcessor=JSON"
```

3. Configure ModSecurity: Turn the engine on by editing the configuration file

```
—(root@kali)-[~]
-# sudo a2enmod headers
Enabling module headers.
To activate the new configuration, you need to run:
systemctl restart apache2

—(root@kali)-[~]
-# systemctl restart apache2
Unknown command verb 'restart', did you mean 'restart'?

—(root@kali)-[~]
-# systemctl restart apache2

—(root@kali)-[~]
-# 

—(root@kali)-[~]
-# sudo cp /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf
cp: cannot stat '/etc/modsecurity/modsecurity.conf-recommended': No such file or directory

—(root@kali)-[~]
-# systemctl restart apache2

—(root@kali)-[~]
-# sudo rm -rf /usr/share/modsecurity-crs

—(root@kali)-[~]
-# sudo apt install libapache2-mod-security2 -y
```

```

  upgrading, or, installing, or, removing, or, not upgrading.
[~] (root㉿kali)-[~]
└─# sudo git clone https://github.com/coreruleset/coreruleset /usr/share/modsecurity-crs
Cloning into '/usr/share/modsecurity-crs'...
remote: Enumerating objects: 36267, done.
remote: Counting objects: 100% (304/304), done.
remote: Compressing objects: 100% (176/176), done.
remote: Total 36267 (delta 263), reused 128 (delta 128), pack-reused 35963 (from 3)
Receiving objects: 100% (36267/36267), 10.92 MiB | 5.19 MiB/s, done.
Resolving deltas: 100% (28760/28760), done.

[~] (root㉿kali)-[~]
└─# g

[~] (root㉿kali)-[~]
└─# sudo mv /usr/share/modsecurity-crs/crs-setup.conf.example /usr/share/modsecurity-crs/crs-setup.conf

[~] (root㉿kali)-[~]
└─# sudo mv /usr/share/modsecurity-crs/rules/REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf.example /usr/share/modsecurity-crs/rules/REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf

[~] (root㉿kali)-[~]
└─#

```

*sudo nano /etc/modsecurity/modsecurity.conf*

Find the line SecRuleEngine DetectionOnly and change it to SecRuleEngine On.

### Step 2: Deploy a Vulnerable App

For simplicity, we'll create a single vulnerable PHP page instead of installing a complex application like DVWA.

#### 1. Install PHP:

*sudo apt install -y php libapache2-mod-php*

```

[~] (root㉿kali)-[/var/www/html]
└─# sudo git clone https://github.com/ethicalhack3r/DVWA
  Cloning into 'DVWA'...
remote: Enumerating objects: 5373, done.
remote: Total 5373 (delta 0), reused 0 (delta 0), pack-reused 5373 (from 1)
Receiving objects: 100% (5373/5373), 2.57 MiB | 508.00 KiB/s, done.
Resolving deltas: 100% (2673/2673), done.

[~] (root㉿kali)-[/var/www/html]
└─# ls
DVWA  index.html  index.nginx-debian.html

[~] (root㉿kali)-[/var/www/html]
└─#

```

### Step 3: Create and Test a WAF Rule

#### 1. Create a Custom Rules File:



```
root@kali:~# systemctl status mysql
● mariadb.service - MariaDB 11.8.3 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; disabled; preset: disabled)
     Active: active (running) since Wed 2025-09-17 13:26:40 EDT; 29s ago
   Invocation: 39d3d1dc3c3f46f59861b2c5204cab0f
     Docs: man:mariadb(8)
           https://mariadb.com/kb/en/library/systemd/
   Process: 114383 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /var/run/mysql
   Process: 114385 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] && VAR= || 
   Process: 114461 ExecStartPost=/bin/rm -f /run/mysqld/wsrep-start-position (code=exited
   Process: 114463 ExecStartPost=/etc/mysql/debian-start (code=exited, status=0/SUCCESS)
 Main PID: 114439 (mariadb)
   Status: "Taking your SQL requests now ... "
     Tasks: 15 (limit: 14275)
    Memory: 126.1M (peak: 130.9M)
      CPU: 2.017s
     CGroup: /system.slice/mariadb.service
             └─114439 /usr/sbin/mariadb

Sep 17 13:26:40 kali mariadb[114439]: 2025-09-17 13:26:40 0 [Note] Plugin 'wsrep-provider'
Sep 17 13:26:40 kali mariadb[114439]: 2025-09-17 13:26:40 0 [Note] InnoDB: Loading buffer
Sep 17 13:26:40 kali mariadb[114439]: 2025-09-17 13:26:40 0 [Note] InnoDB: Buffer pool(s)
Sep 17 13:26:40 kali mariadb[114439]: 2025-09-17 13:26:40 0 [Note] Server socket created
Sep 17 13:26:40 kali mariadb[114439]: 2025-09-17 13:26:40 0 [Note] mariadb: Event Scheduler
Sep 17 13:26:40 kali mariadb[114439]: 2025-09-17 13:26:40 0 [Note] /usr/sbin/mariadb: re
Sep 17 13:26:40 kali mariadb[114439]: Version: '11.8.3-MariaDB-1+b1' from 'Debian' socket:
Sep 17 13:26:40 kali /etc/mysql/debian-start[114465]: Upgrading MariaDB tables if necessar
Sep 17 13:26:40 kali systemd[1]: Started mariadb.service - MariaDB 11.8.3 database server.
Sep 17 13:26:41 kali mariadb[114439]: 2025-09-17 13:26:41 0 [Note] InnoDB: Memory pressur

—(root@kali)-[/var/www/html/DVWA/config]
root@kali:~# sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 31
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)]> 
```

```

cd: no such file or directory: /etc/php/7.4/apache2

[root@kali]~#
# php -v
PHP 8.4.11 (cli) (built: Aug 15 2025 23:34:18) (NTS)
Copyright (c) The PHP Group
Built by Debian
Zend Engine v4.4.11, Copyright (c) Zend Technologies
    with Zend OPcache v8.4.11, Copyright (c), by Zend Technologies

[root@kali]~#
# cd /etc/php/8.4/apache2

[root@kali]/etc/php/8.4/apache2#
# sudo systemctl start apache2

[root@kali]/etc/php/8.4/apache2#
# systemctl status apache2
● apache2.service - The Apache HTTP Server
    Loaded: loaded (/usr/lib/systemd/system/apache2.service; disabled; preset: disabled)
    Active: active (running) since Wed 2025-09-17 12:45:04 EDT; 1h 3min ago
      Invocation: 13ab5ba8e2df4bb2a6067673f842af35
        Docs: https://httpd.apache.org/docs/2.4/
    Process: 93454 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 93466 (apache2)
     Tasks: 6 (limit: 2163)
    Memory: 35.9M (peak: 36.2M)
      CPU: 540ms
     CGroup: /system.slice/apache2.service
             ├─93466 /usr/sbin/apache2 -k start
             ├─93469 /usr/sbin/apache2 -k start
             ├─93470 /usr/sbin/apache2 -k start
             ├─93471 /usr/sbin/apache2 -k start
             ├─93472 /usr/sbin/apache2 -k start
             └─93473 /usr/sbin/apache2 -k start

Sep 17 12:45:03 kali systemd[1]: Starting apache2.service - The Apache HTTP Server ...
Sep 17 12:45:04 kali systemd[1]: Started apache2.service - The Apache HTTP Server.

[root@kali]/etc/php/8.4/apache2#
# 
```

127.0.0.0/DVWA/setup.php

OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB >

**DVWA**

**Database Setup**

Click on the 'Create / Reset Database' button below to create or reset your database. If you get an error make sure you have the correct user credentials in: `/var/www/html/config.inc.php`

If the database already exists, **it will be cleared and the data will be reset**. You can also use this to reset the administrator credentials ("admin // password").

**Setup Check**

**General**  
Operating system: **\*nix**  
DVWA version:  

- Git reference: **c6e3d05c503cc6c02feca78cab5b4b747ba83d**
- Author: Robin Wood

reCAPTCHA key: **Missing**  
Writable folder `/var/www/html/DVWA/hackable/uploads/`: **Yes**  
Writable folder `/var/www/html/DVWA/config/`: **Yes**

**Apache**  
Web Server SERVER\_NAME: **127.0.0.0**  
mod\_rewrite: **Not Enabled**  
mod\_rewrite is required for the AP labs.

**PHP**  
PHP version: **8.4.11**  
PHP function display\_errors: **Disabled**  
PHP function display\_startup\_errors: **Disabled**  
PHP function allow\_url\_include: **Disabled**  
PHP function allow\_url\_fopen: **Enabled**  
PHP module gd: **Missing - Only an issue if you want to play with captchas**  
PHP module mysql: **Installed**

```
sudo nano /etc/apache2/conf-available/modsecurity-custom.conf
```



# DVWA Security

## Security Level

Security level is currently: **low**.

You can set the security level to low, medium, high or impossible. The security level of DVWA:

1. Low - This security level is completely vulnerable and **has no security** implemented. It is used as an example of how web application vulnerabilities manifest through browser and server side code. It also acts as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices** that a developer has tried but failed to secure an application. It also acts as a challenge for more advanced exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **good security practices** to attempt to secure the code. The vulnerability may not allow for full exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is impossible to exploit the source code to the secure source code.

Prior to DVWA v1.9, this level was known as 'high'.

Security level set to low

→ ⌂ ⌂ 127.0.0.0/DVWA/vulnerabilities/sql/?id=1+or+1%3D+1&Submit=Submit#

OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB

**DVWA**

**Vulnerability: SQL Injection**

User ID:  Submit

ID: 1' or 1 = '1  
First name: admin  
Surname: admin

ID: 1' or 1 = '1  
First name: Gordon  
Surname: Brown

ID: 1' or 1 = '1  
First name: Hack  
Surname: Me

ID: 1' or 1 = '1  
First name: Pablo  
Surname: Picasso

ID: 1' or 1 = '1  
First name: Bob  
Surname: Smith

**More Information**

- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- [https://owasp.org/www-community/attacks/SQL\\_Injection](https://owasp.org/www-community/attacks/SQL_Injection)
- <https://bobby-tables.com/>

Home Instructions Setup / Reset DB

Brute Force Command Injection CSRF File Inclusion File Upload Insecure CAPTCHA

**SQL Injection**

SQL Injection (Blind) Weak Session IDs XSS (DOM) XSS (Reflected) XSS (Stored) CSP Bypass JavaScript Authorisation Bypass Open HTTP Redirect Cryptography API

DVWA Security

→ ⌂ ⌂ 127.0.0.0/DVWA/vulnerabilities/sql/

OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB

**DVWA**

**Vulnerability: SQL Injection**

User ID: 1' or 1 = '1 Submit

**More Information**

- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- [https://owasp.org/www-community/attacks/SQL\\_Injection](https://owasp.org/www-community/attacks/SQL_Injection)
- <https://bobby-tables.com/>

Home Instructions Setup / Reset DB

Brute Force Command Injection CSRF File Inclusion File Upload Insecure CAPTCHA

**SQL Injection**

SQL Injection (Blind) Weak Session IDs XSS (DOM) XSS (Reflected) XSS (Stored) CSP Bypass JavaScript Authorisation Bypass Open HTTP Redirect Cryptography API

DVWA Security

```
root@kali:~/usr/share
# sudo git clone https://github.com/coreruleset/coreruleset.git modsecurity-crs
Cloning into 'modsecurity-crs'...
remote: Enumerating objects: 36275, done.
remote: Counting objects: 100% (307/307), done.
remote: Compressing objects: 100% (177/177), done.
remote: Total 36275 (delta 265), reused 130 (delta 130), pack-reused 35968 (from 4)
Receiving objects: 100% (36275/36275), 10.89 MiB | 2.75 MiB/s, done.
Resolving deltas: 100% (28762/28762), done.

root@kali:~/usr/share
# cd /usr/share/modsecurity-crs
root@kali:~/usr/share/modsecurity-crs
# sudo cp crs-setup.conf.example crs-setup.conf
root@kali:~/usr/share/modsecurity-crs
# ls -l /usr/share/modsecurity-crs/crs-setup.conf
-rw-r--r-- 1 root root 36362 Sep 19 03:11 /usr/share/modsecurity-crs/crs-setup.conf

root@kali:~/usr/share/modsecurity-crs
# cp /usr/share/modsecurity-crs/rules/
asp-dotnet-error.data REQUEST-913-SCANNER-DETECTION.conf REQUEST-944-APPLICATION-ATTACK-JAVA.conf
as-errors.data REQUEST-920-PROTOCOL-ENFORCEMENT.conf REQUEST-949-BLOCKING-EVALUATION.conf
java-classes.data REQUEST-921-PROTOCOL-ATTACK.conf RESPONSE-950-DATA-LEAKAGES.conf
jfi-os-files.data REQUEST-922-MULTIPART-ATTACK.conf RESPONSE-951-DATA-LEAKAGES-SQL.conf
php-errors.data REQUEST-930-APPLICATION-ATTACK-LFI.conf RESPONSE-952-DATA-LEAKAGES-JAVA.conf
php-errors-pl2.data REQUEST-931-APPLICATION-ATTACK-RFI.conf RESPONSE-953-DATA-LEAKAGES-PHP.conf
php-function-names-933150.data REQUEST-932-APPLICATION-ATTACK-RCE.conf RESPONSE-954-DATA-LEAKAGES-IIS.conf
php-variables.data REQUEST-933-APPLICATION-ATTACK-PHP.conf RESPONSE-955-WEBSHELLS.conf
REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf.example REQUEST-934-APPLICATION-ATTACK-GENERIC.conf RESPONSE-956-DATA-LEAKAGES-RUBY.conf
REQUEST-901-INITIALIZATION.conf REQUEST-941-APPLICATION-ATTACK-XSS.conf RESPONSE-959-BLOCKING-EVALUATION.conf
REQUEST-905-COMMON-EXCEPTIONS.conf REQUEST-942-APPLICATION-ATTACK-SQLI.conf RESPONSE-980-CORRELATION.conf
REQUEST-911-METHOD-ENFORCEMENT.conf REQUEST-943-APPLICATION-ATTACK-SESSION-FIXATION.conf RESPONSE-999-EXCLUSION-RULES-AFTER-CRS.conf.example
restricted-files.data
restricted-upload.data
ruby-errors.data
scanners-user-agents.data
sql-errors.data
ssrf.data
unix-shell-builtins.data
unix-shell.data
web-shells-asp.data
web-shells-php.data
windows-powershell-commands.data

root@kali:~/usr/share/modsecurity-crs
#
```

```
sudo systemctl restart apache2
```

## 2. Test the WAF:

- **Benign Request:** Open a web browser and go to `http://<VM_IP>/login.php`. Enter a normal username like `admin` and click `Login`. It should work.
- **Attack Request:** Now, enter a basic SQLi probe into the username field: `test' UNION SELECT 1,2,3--`
- **Verify the Block:** When you click `Login`, you will receive a **403 Forbidden** error from the server. The WAF has blocked your request. You can see the block message in Apache's error log (`sudo tail /var/log/apache2/error.log`).



## Project 3: Threat Intel Processor

This guide explains how to build a Python script that fetches malicious IPs from the **AbuseIPDB** threat feed and checks them against a sample log file.

### Step 1: Setup and API Key

#### 1. Install Python and Libraries:

```
sudo apt update
```

```
sudo apt install -y python3 python3-pip
```

```
pip3 install requests
```

#### 2. Get an AbuseIPDB API Key:

- Go to <https://www.abuseipdb.com/> and create a free account.