



## WEB APPLICATION FIREWALL ON DVWA

*Date: April 07<sup>th</sup>, 2023*  
*Project: 4*  
*Version 1.0*

## CONTENTS

WEB APPLICATION FIREWALL ON DVWA .....	1
Introduction: - .....	3
What is Web Application Firewall? .....	3
What is Open Source WAF?.....	4
What is Open Source Mod Security? .....	4
Installation of ModSecurity. ....	5
Configuration of ModSecurity. ....	5
Setting up the OWASP ModSecurity Core Rule Set.....	7
Enabling ModSecurity in Apache2.....	7
Download DVWA in Localhost: .....	9
Configure Database DVWA .....	10
Configure Apache Server for DVWA .....	11
Testing ModSecurity.....	12
1.Sql Injection .....	13
2. Cross-Site Scripting (XSS) .....	14
3.Local File Inclusion (LFI) .....	14
Turn Off ModSecurity .....	15
Testing ModSecurity off .....	17
1.Sql Injection: .....	18
2.File Inclusion.....	19
3.Cross-Site Scripting (XSS) .....	19
Conclusion: .....	20
References Links:.....	20
End Page .....	21

## INTRODUCTION: -

The internet has become an indispensable aspect of modern life, serving as a critical tool for performing a multitude of daily tasks. Simultaneously, the growing popularity of web applications has provided unparalleled access to information and services. However, with this increase in web application usage comes a corresponding uptick in attacks targeting them. Malicious actors have become increasingly sophisticated, exploiting vulnerabilities in web applications to engage in activities such as data theft or other nefarious actions. One effective means of safeguarding web applications against such attacks is through the implementation of a Web Application Firewall (WAF). This report outlines the steps for installing an open-source WAF on a Linux-based system and demonstrates how it can be leveraged to thwart attacks on a web application.

## WHAT IS WEB APPLICATION FIREWALL?

WAF stands for Web Application Firewall. It is a type of firewall designed to protect web applications from various types of attacks, such as SQL injection, cross-site scripting, and other malicious web-based attacks.

A WAF is typically implemented as a software or hardware solution that sits between the web server and the internet, and it examines every request and response that passes through it to identify and block any malicious traffic.

A WAF can protect web applications by using a number of techniques, including signature-based detection, behavior-based detection, and anomaly detection. Signature-based detection involves the use of pre-defined signatures or patterns to identify known attacks, while behavior-based detection analyzes the behavior of the web application to detect any anomalies or suspicious activity.

WAFs can be deployed in a number of different ways, including as a standalone appliance, as a software module installed on a web server, or as a cloud-based service.

Some of the benefits of using a WAF include improved security, reduced risk of data breaches, and improved compliance with industry standards and regulations. However, WAFs can also introduce some additional complexity and potential performance issues, so careful planning and configuration are essential to ensure that the benefits outweigh the costs.

## WHAT IS OPEN SOURCE WAF?

An Open Source WAF is a type of Web Application Firewall that is built on open-source software and can be freely downloaded, installed, and customized by users. The source code for open-source WAFs is available to the public, allowing users to modify and improve the software to meet their specific needs. This provides an opportunity for a wide range of users to collaborate on the development and improvement of the software, resulting in a more robust and effective solution.

Open-source WAFs typically work by analyzing incoming web traffic and identifying potential security threats. This is done by analyzing the request headers and content of each request and comparing it to a set of predefined rules. These rules can be customized to reflect the specific needs of the application being protected. For example, a rule could be created to block SQL injection attacks, cross-site scripting attacks, or other common attack types.

One of the key advantages of open-source WAFs is their flexibility. Because the source code is freely available, users can modify the software to meet their specific needs. This allows for a high degree of customization and can help to ensure that the WAF is optimized for the specific web application being protected.

Another advantage of open-source WAFs is that they can be more cost-effective than commercial solutions. Because the software is freely available, there are no licensing fees, and users can often leverage existing infrastructure to support the WAF.

However, it's important to note that open-source WAFs require a certain level of technical expertise to set up and maintain, which may be a barrier to entry for some users. Additionally, open-source software may not always have the same level of support and resources as commercial solutions, although many open-source communities have active user bases that can provide support and assistance.

## WHAT IS OPEN SOURCE MOD SECURITY?

ModSecurity is an open-source Web Application Firewall (WAF) that provides protection against a variety of web-based attacks. It is a widely used WAF and is supported by many web servers, including Apache and Nginx. ModSecurity can be installed on Linux, Windows, and other operating systems.

ModSecurity works by analyzing incoming HTTP traffic and comparing it to a set of predefined rules. The rules can be customized to reflect the specific needs of the

application being protected. ModSecurity uses a rule language known as ModSecurity Rule Language (MRL), which is designed to be flexible and easy to use.

ModSecurity can protect web applications against a range of attacks, including SQL injection, cross-site scripting, and file inclusion attacks. It can also help to prevent brute-force attacks and DDoS attacks.

One of the advantages of ModSecurity is its flexibility. It can be used as a standalone WAF or integrated with other security solutions, such as intrusion detection systems (IDS) and security information and event management (SIEM) systems. ModSecurity can also be configured to generate alerts and log events, which can be used to monitor and analyze traffic patterns and identify potential security threats.

Another advantage of ModSecurity is its active user community. The software is open-source, which means that users can access the source code and modify it to meet their specific needs. The community also provides support and assistance to users through forums and other online resources.

ModSecurity can be complex to configure and requires a certain level of technical expertise. Additionally, some users may find the rule language to be challenging to work with. Nevertheless, ModSecurity is a powerful and flexible open-source WAF that can provide valuable protection against a range of web-based attacks.

## INSTALLATION OF MODSECURITY.

On our Linux machine, we installed ModSecurity using the package manager. Specifically, we utilized the following commands on Kali Linux to complete the installation process:

1. "sudo apt-get install libapache2-mod-security2"
2. "sudo a2enmod headers"
3. "sudo systemctl restart apache2".

These commands enable the integration of ModSecurity with the Apache web server.

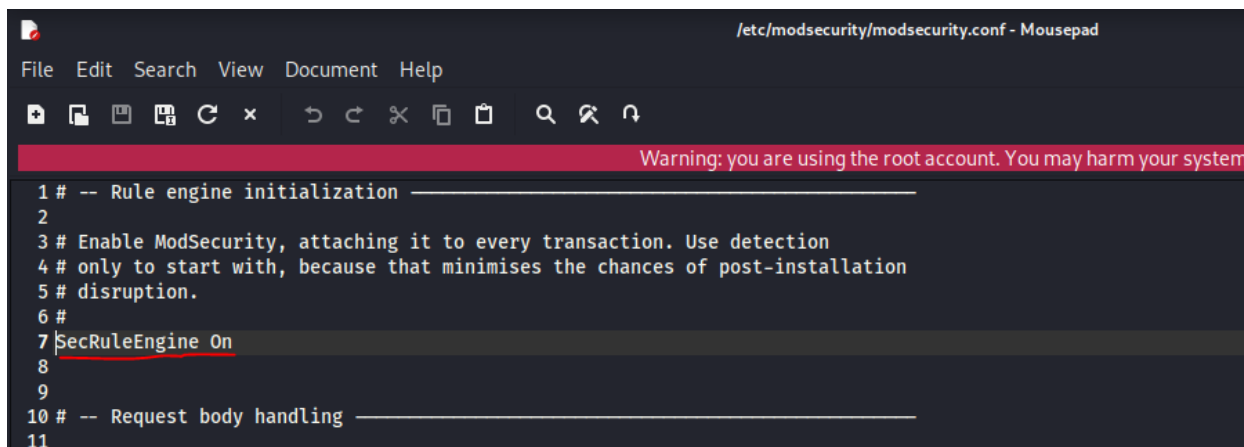
## CONFIGURATION OF MODSECURITY.

Following the installation of ModSecurity, the next step is to configure it to provide protection to our web application. ModSecurity utilizes a configuration file, which can be found at `/etc/modsecurity/modsecurity.conf`. To tailor the configuration to our specific requirements, we made necessary modifications to this file. The configuration file includes rules that define which types of incoming traffic should be allowed or blocked by the WAF.

ModSecurity, being a firewall, relies on a set of rules to function effectively. To implement the OWASP Core Rule Set, the first step is to prepare the ModSecurity configuration file.

To achieve this, execute the command `sudo cp /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf` in the terminal to remove the `.recommended` extension from the configuration file name.

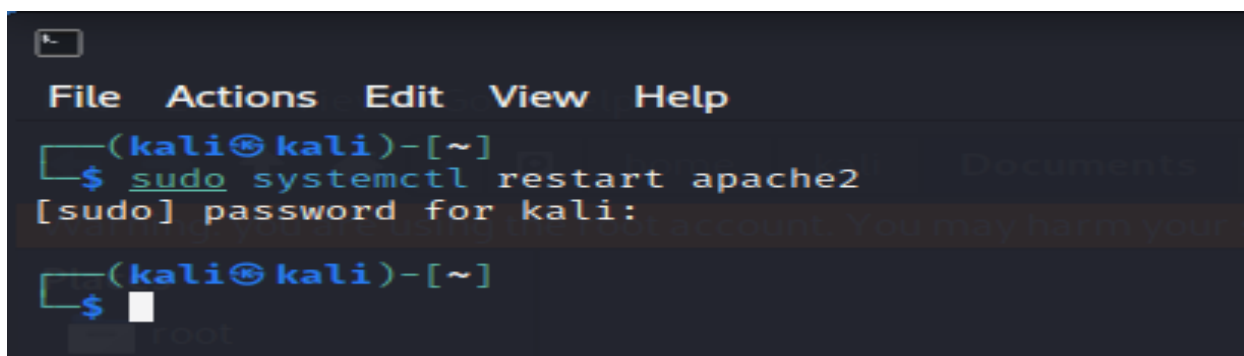
Next, open the `conf` file using a text editor, such as vim, and change the value for `SecRuleEngine` to `On`. This will enable the ModSecurity engine to inspect incoming traffic and apply the rules defined in the OWASP Core Rule Set.



```
File Edit Search View Document Help
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
```

The `secRuleEngine On` and save it.

To activate the changes, restart Apache by running the following command with sudo privileges: `sudo systemctl restart apache2`. Once done, ModSecurity will be configured and ready to function. The subsequent step is to establish a rule set that can actively shield your web server from potential attacks.



```
File Actions Edit View Help
(kali@kali)-[~]
$ sudo systemctl restart apache2
[sudo] password for kali:
(kali@kali)-[~]
$
```

## SETTING UP THE OWASP MODSECURITY CORE RULE SET.

The ModSecurity Core Rule Set (CRS) from OWASP is a collection of general attack detection rules, designed to be used with ModSecurity or web application firewalls that are compatible. The CRS serves the purpose of shielding web applications from various types of attacks, including the OWASP Top Ten, while minimizing the number of false alerts generated. With the CRS, web applications can receive protection from a range of common attack categories, such as Cross Site Scripting, SQL Injection, and Local File Inclusion.

**To set up the OWASP-CRS, follow the procedures outlined below.**

1. First, delete the current rule set that comes prepackaged with ModSecurity by running the following command:  
Command: `- sudo rm -rf /usr/share/ModSecurity-crs`
2. Ensure that git is installed:  
Command: `- sudo apt install git`
3. Clone the OWASP-CRS GitHub repository into the `/usr/share/modsecurity-crs` directory:  
Command: `sudo git clone https://github.com/coreruleset/coreruleset /usr/share/modsecurity-crs`
4. Rename the `crs-setup.conf.example` to `crs-setup.conf`:  
Command: `- sudo mv /usr/share/modsecurity-crs/crs-setup.conf.example /usr/share/modsecurity-crs/crs-setup.conf`
5. Rename the default request exclusion rule file:  
Command: `- 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`

You should now have the OWASP-CRS setup and ready to be used in your Apache configuration.

## ENABLING MODSECURITY IN APACHE2

To begin using ModSecurity, enable it in the Apache configuration file by following the steps outlined below:

1. Using a text editor such as vim, edit the `/etc/apache2/mods-available/security2.conf` file to include the OWASP-CRS files you have downloaded:

```
File Edit Search View Document Help
Warning: you are using the root account. You may harm your system.
1 <IfModule security2_module>
2     SecDataDir /var/cache/modsecurity
3     Include /usr/share/modsecurity-crs/crs-setup.conf
4     Include /usr/share/modsecurity-crs/rules/*.conf
5
6     # Default Debian dir for modsecurity's persistent data
7     SecDataDir /var/cache/modsecurity
8
9     # Include all the *.conf files in /etc/modsecurity.
10    # Keeping your local configuration in that directory
11    # will allow for an easy upgrade of THIS file and
12    # make your life easier
13    IncludeOptional /etc/modsecurity/*.conf
14
15    # Include OWASP ModSecurity CRS rules if installed
16    IncludeOptional /usr/share/modsecurity-crs/*.load
17 </IfModule>
18
```

Copy and paste it then save the file.

2. In /etc/apache2/sites-enabled/000-default.conf file VirtualHost block, include the SecRuleEngine directive set to On.

```
File Edit Search View Document Help
Warning: you are using the root account. You may harm your system.
1 <VirtualHost *:80>
2
3     ServerAdmin webmaster@localhost
4     DocumentRoot /var/www/html
5
6     ErrorLog ${APACHE_LOG_DIR}/error.log
7     CustomLog ${APACHE_LOG_DIR}/access.log combined
8
9     SecRuleEngine On
10
11    # The ServerName directive sets the request scheme, hostname and port that
12    # the server uses to identify itself. This is used when creating
13    # redirection URLs. In the context of virtual hosts, the ServerName
14    # specifies what hostname must appear in the request's Host: header to
15    # match this virtual host. For the default virtual host (this file) this
16    # value is not decisive as it is used as a last resort host regardless.
17    # However, you must set it for any further virtual host explicitly.
18    #ServerName www.example.com
19
20
```

SecRuleEngine On then save it.



3. Restart the Apache2 service to apply the configuration:

```

kali@kali: ~
File Actions Edit View Help
(kali@kali)-[~]
$ sudo systemctl restart apache2
[sudo] password for kali:
(kali@kali)-[~]
$

```

There is a completed setting of ModSecurity. Now download DVWA in localhost.

### DOWNLOAD DVWA IN LOCALHOST:

There are some steps to install DVWA manually on your localhost. With command in terminal.

1. `cd /var/www/html`
2. `sudo git clone https://github.com/ethicalhack3r/DVWA`

```

(kali@kali)-[~]
$ cd /var/www/html/
(kali@kali)-[/var/www/html]
$ ls
DVWA  index.html  index.nginx-debian.html
(kali@kali)-[/var/www/html]
$

```

3. Give DVWA all permission to read, write, and execute.
4. `sudo chmod -R 777 DVWA`

To set up and configure DVWA, we will need to navigate to the `/dvwa/config` directory. Use the command below:

5. `cd DVWA/config`

Run the `ls` command to see the contents of the config directory.

You should see a file with the name `config.inc.php.dist`. That file contains the default DVWA configurations.

We will not tamper with it, and it will act as our backup if things go south. Instead, we will create a copy of this file with the name `config.inc.php` that we will use to configure DVWA. Use the command below.

6. `sudo cp config.inc.php.dist config.inc.php`

You can use the `ls` command to check if the file was copied successfully.

To make the required configurations, open the "`config.inc.php`" file using the nano editor with the command `sudo nano config.inc.php`. Once the file is open, navigate to the relevant section where you can see parameters like "`db_database`", "`db_user`", "`db_password`", etc. These parameters are typically located in a section of the file where database configurations are present. You are free to modify these values as per your requirements, but it is recommended to make a note of the values as they will be needed for the database setup process. As an example, you can set the value of "`db_user`" to "`userDVWA`" and "`db_password`" to "`dvwa`".

```
# Database variables
# WARNING: The database specified under db_database WILL BE ENTIRELY DELETED during setup.
# Please use a database dedicated to DVWA.
#
# If you are using MariaDB then you cannot use root, you must use create a dedicated DVWA user.
# See README.md for more information on this.
$_DVWA = array();
$_DVWA[ 'db_server' ]   = '127.0.0.1';
$_DVWA[ 'db_database' ] = 'dvwa';
$_DVWA[ 'db_user' ]     = 'userDVWA';
$_DVWA[ 'db_password' ] = 'dvwa';
$_DVWA[ 'db_port' ]    = '3306';
```

## CONFIGURE DATABASE DVWA

By default, Kali Linux comes installed with the MariaDB relational database management system. You, therefore, don't need to install any packages. First, start the `mysql` service with the command below.

1. `sudo systemctl start mysql`

You can check whether the service is running with the command:

2. `systemctl status mysql`

To log in to the database, use the command below. In our case, we are using root since that is the superuser name set on our system. If you have something different, then you will need to replace the root.

3. `sudo mysql -u root -p`

You will be prompted for a password. However, since we haven't set any yet, just hit Enter to continue.

We will first create a new user using the credentials we set in the config.inc.php file in the DVWA directory. Execute the command below, replacing the username and password with your preset credentials.

4. `create user 'userDVWA'@'127.0.0.1' identified by "dvwa";`

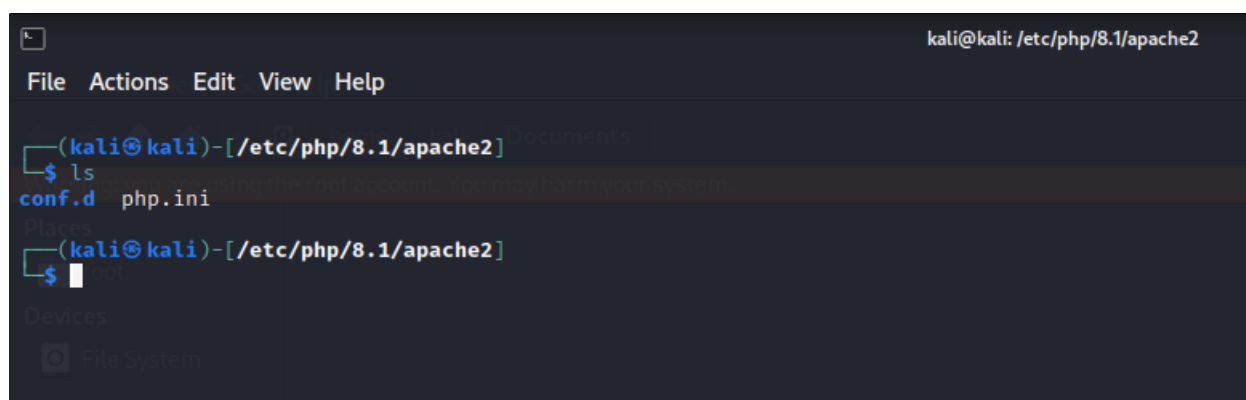
We now need to grant this user total privilege over the dvwa database. Execute the command below, replacing the username and password with your credentials.

5. `grant all privileges on dvwa.* to 'userDVWA'@'127.0.0.1' identified by 'dvwa';`

## CONFIGURE APACHE SERVER FOR DVWA

The Apache web server comes installed by default on Kali Linux. Therefore, we don't have to need to install any additional packages.

To get started configuring Apache2, launch the Terminal and navigate the `/etc/php/8.1/apache2` directory.



```
kali@kali: /etc/php/8.1/apache2
File Actions Edit View Help
(kali@kali)~[/etc/php/8.1/apache2]
$ ls
conf.d  php.ini
(kali@kali)~[/etc/php/8.1/apache2]
$
```

Open Php.ini file in nano.

```
;;;;;;;;;;  
; Fopen wrappers ;  
;;;;;;;;;;  
  
; Whether to allow the treatment of URLs (like http:// or ftp://) as files.  
; https://php.net/allow-url-fopen  
allow_url_fopen = On  
  
; Whether to allow include/require to open URLs (like https:// or ftp://) as files.  
; https://php.net/allow-url-include  
allow_url_include = On  
  
; Define the anonymous ftp password (your email address). PHP's default setting  
; for this is empty.  
; https://php.net/from
```

Allow\_url\_fopen = On and allow\_url\_include = On as well.

There is completed The DVWA installation and configuration.

## TESTING MODSECURITY

ModSecurity is a web application firewall engine that is designed to provide real-time protection for web applications against various types of attacks. It operates as a module within the Apache web server and is capable of monitoring incoming web traffic and analyzing it against a set of predefined rules. ModSecurity can detect and prevent attacks such as cross-site scripting (XSS), SQL injection, and file inclusion attacks.

To test ModSecurity, you can perform various attack scenarios on your web application and observe the behavior of ModSecurity in detecting and preventing these attacks. Some common test cases are:

1. Cross-Site Scripting (XSS) Attack: This attack involves injecting malicious scripts into a web page viewed by other users. To test for this attack, you can try entering a script tag with some malicious code in a form field or a URL parameter. ModSecurity should detect and block such requests.
2. SQL Injection Attack: This attack involves exploiting vulnerabilities in web applications that allow malicious SQL statements to be executed by the database. To test for this attack, you can try entering some SQL injection commands in a form field or a URL parameter. ModSecurity should detect and block such requests.
3. Remote File Inclusion (RFI) Attack: This attack involves exploiting vulnerabilities in web applications that allow remote files to be included and executed on the server. To test for this attack, you can try entering a remote URL that points to a malicious file in a form field or a URL parameter. ModSecurity should detect and block such requests.

4. Local File Inclusion (LFI) Attack: This attack involves exploiting vulnerabilities in web applications that allow local files to be included and executed on the server. To test for this attack, you can try entering a local file path that points to a sensitive file in a form field or a URL parameter. ModSecurity should detect and block such requests.

After performing these test cases, you can review the ModSecurity logs to see the details of the blocked requests and the rules that were triggered. This information can be used to fine-tune the ModSecurity rules and improve the overall security of your web application.

## 1. SQL INJECTION

To test whether ModSecurity is working correctly, we attempted to perform a SQL injection attack on our web application by accessing a specific URL: <http://127.0.0.1/dvwa/login.php?username=' OR 1=1>. This URL contains a malicious payload that attempts to exploit a vulnerability in the login page to gain unauthorized access to the system. However, if ModSecurity is functioning properly, it should detect and block this attack, preventing the payload from executing and returning a 403 Forbidden error.

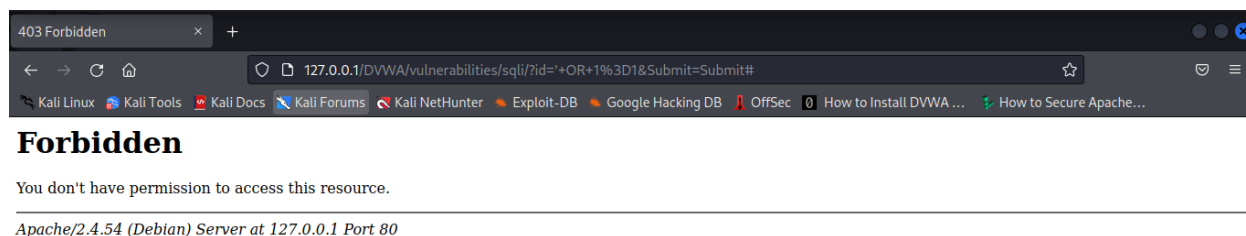
### Vulnerability: SQL Injection

User ID:

### 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/>

Submit the query. It will block by ModSecurity Firewall.



## 2. CROSS-SITE SCRIPTING (XSS)

In this scenario, I attempted to perform a Cross-Site Scripting (XSS Reflected) attack on DVWA by uploading a malicious file, as illustrated in the image below. The malicious file contained a script that could potentially execute harmful actions on the victim's browser, such as stealing cookies or injecting more malicious code. If ModSecurity is functioning properly, it should detect and block this attack, preventing the malicious file from being uploaded and executed on the web application.

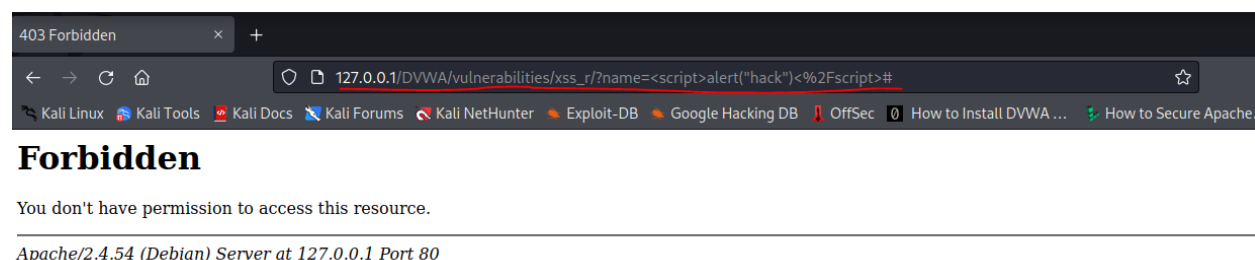
### Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

#### More Information

- <https://owasp.org/www-community/attacks/xss/>
- <https://owasp.org/www-community/xss-filter-evasion-cheatsheet>
- [https://en.wikipedia.org/wiki/Cross-site\\_scripting](https://en.wikipedia.org/wiki/Cross-site_scripting)
- <http://www.cgisecurity.com/xss-faq.html>
- <http://www.scriptalert1.com/>

Submit the script in textbox. It will block by ModSecurity Firewall.

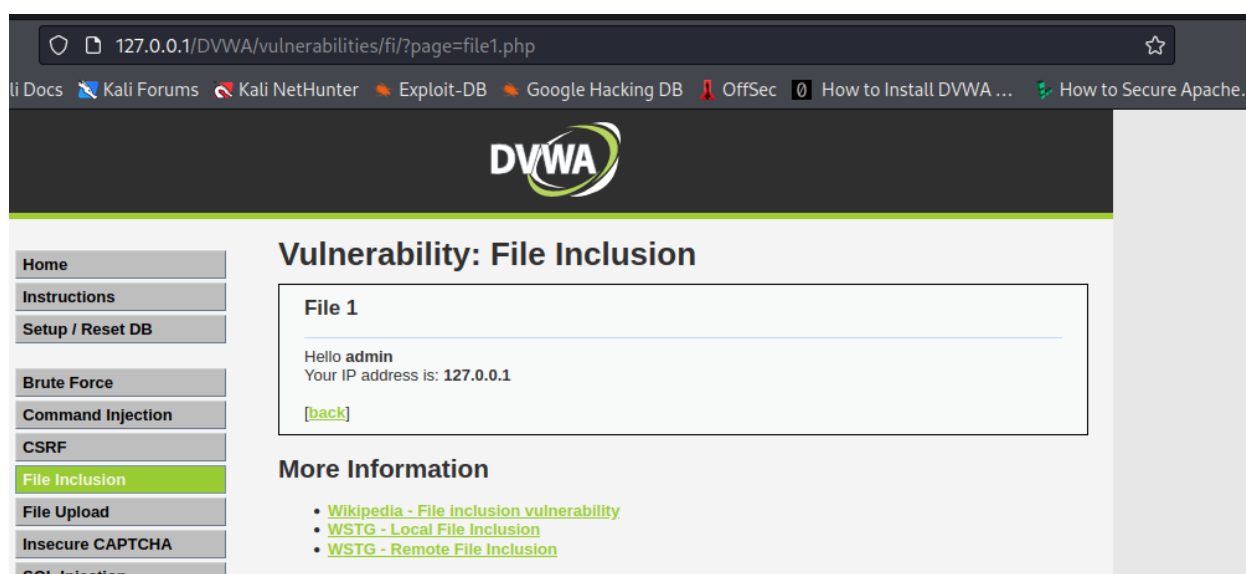


## 3. LOCAL FILE INCLUSION (LFI)

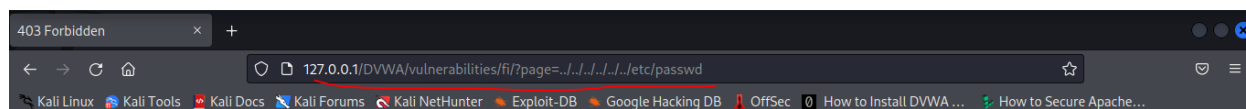
LFI is a type of web application vulnerability that occurs when an application allows a user to include a file from the local file system, but does not properly sanitize or validate the input. This can lead to an attacker being able to read sensitive information, execute arbitrary code, or even gain full control of the system.

ModSecurity is a web application firewall (WAF) that can be used to protect against LFI attacks by inspecting incoming HTTP requests and blocking any that match a predefined set of rules. The rule provided in the previous message looks for the presence of the characters `./` or `../` in the input parameters, which can indicate an LFI attack. If these characters are found, the request is denied and a log message is generated.

Implementing this rule requires ModSecurity to be installed and configured on your web server, and adding the rule to your ModSecurity configuration file. Note that the rule may generate false positives if your application legitimately uses `./` or `../` in input parameters, and you may need to adjust the rule or whitelist certain parameters to avoid blocking legitimate requests.



Use in url `.././.././.././.././etc/passwd`. Then see ModSecurity will block the request.



## Forbidden

You don't have permission to access this resource.

Apache/2.4.54 (Debian) Server at 127.0.0.1 Port 80

## TURN OFF MODSECURITY

We modified the configuration file of ModSecurity located at `/etc/modsecurity/modsecurity.conf` and updated the parameter with the following value:

### WAF MOD Security Firewall On DVWA

Penetration Testing

Copyright © Noman Ahmed

Open a file to turn off SecRuleEngine off.

```

File Edit Search View Document Help
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 Off
8
9
10 # -- Request body handling
11

```

Change the SecRuleEngine Vlaue off and then save the document.

Now we edited the ModSecurity configuration file located at `/etc/apache2/mods-available/security2.conf` to turn off ModSecurity, and updated the parameter with the following value:

```

File Edit Search View Document Help
Warning: you are using the root account. You may harm your system.

1 #<IfModule security2_module>
2     SecDataDir /var/cache/modsecurity
3     Include /usr/share/modsecurity-crs/crs-setup.conf
4     Include /usr/share/modsecurity-crs/rules/*.conf
5
6     # Default Debian dir for modsecurity's persistent data
7     SecDataDir /var/cache/modsecurity
8
9     # Include all the *.conf files in /etc/modsecurity.
10    # Keeping your local configuration in that directory
11    # will allow for an easy upgrade of THIS file and
12    # make your life easier
13    IncludeOptional /etc/modsecurity/*.conf
14
15    # Include OWASP ModSecurity CRS rules if installed
16    IncludeOptional /usr/share/modsecurity-crs/*.load
17 #</IfModule>
18

```

Comment the tag and then save it.

Now We modified the configuration file for the default Apache site located at `/etc/apache2/sites-enabled/000-default.conf` to turn off ModSecurity, and updated the parameter with the following value:



```
File Edit Search View Document Help
Warning: you are using the root account. You may harm your system.
1 <VirtualHost *:80>
2
3     ServerAdmin webmaster@localhost
4     DocumentRoot /var/www/html
5
6     ErrorLog ${APACHE_LOG_DIR}/error.log
7     CustomLog ${APACHE_LOG_DIR}/access.log combined
8
9     #SecRuleEngine On
10
11     # The ServerName directive sets the request scheme, hostname and port that
12     # the server uses to identify itself. This is used when creating
13     # redirection URLs. In the context of virtual hosts, the ServerName
14     # specifies what hostname must appear in the request's Host: header to
15     # match this virtual host. For the default virtual host (this file) this
16     # value is not decisive as it is used as a last resort host regardless.
17     # However, you must set it for any further virtual host explicitly.
18     #ServerName www.example.com
19
```

Now we comment on the SecRuleEngine and save the document.

Now we restart the Apache2 server and restart the mMySQL.

```
kali@kali: /etc/php/8.1/apache2
File Actions Edit View Help
(kali@kali)-[/etc/php/8.1/apache2]
$ sudo systemctl restart mysql
[sudo] password for kali:
(kali@kali)-[/etc/php/8.1/apache2]
$ sudo systemctl restart apache2
(kali@kali)-[/etc/php/8.1/apache2]
$
```

## TESTING MODSECURITY OFF

When ModSecurity is turned off, the web application is no longer protected by the firewall and is vulnerable to web-based attacks such as cross-site scripting (XSS), SQL injection, and Local File Inclusion (LFI).

To test whether ModSecurity is turned off, you can attempt to exploit a known vulnerability in your web application. If the exploit is successful, it indicates that ModSecurity is not protecting the application. However, it is important to note that turning off ModSecurity should only be done temporarily for testing purposes, and should not be used as a permanent solution to security issues.

## 1.SQL INJECTION:

### Vulnerability: SQL Injection

User ID:

ID: ' OR 1=1 #  
First name: admin  
Surname: admin

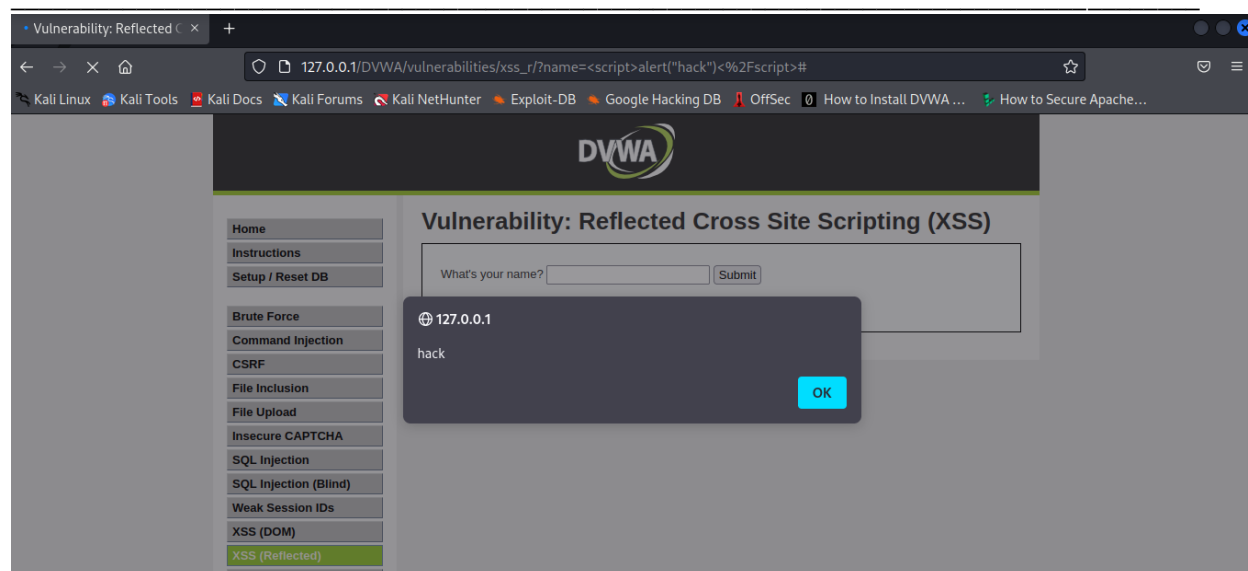
ID: ' OR 1=1 #  
First name: Gordon  
Surname: Brown

ID: ' OR 1=1 #  
First name: Hack  
Surname: Me

ID: ' OR 1=1 #  
First name: Pablo  
Surname: Picasso

ID: ' OR 1=1 #  
First name: Bob  
Surname: Smith





## CONCLUSION:

This report has demonstrated the implementation of an open-source web application firewall (WAF) on a Linux machine and its capability to protect web applications from various types of attacks. Specifically, the report illustrated how ModSecurity can be utilized to prevent SQL injection attacks, Malicious File Upload Attacks, and XSS Attacks on the DVWA web application.

The implementation of a WAF significantly enhances the security posture of a web application, providing an additional layer of defense against attacks that may compromise sensitive data or cause other malicious outcomes. We recommend that web application developers and administrators consider the implementation of a WAF to safeguard their applications against potential threats.

## REFERENCES LINKS:

1. [How to Secure Apache 2 With ModSecurity | Linode](#)
2. [How to Install DVWA on Kali Linux for Pentesting Practice - NoobLinux](#)



END PAGE