WEB APPLICATION FIREWALL

Implementation on DVWA Web Application

Mohammad Anas

CYBER SECURITY | Cohort 5

Table of Contents

Introduction:

The internet has become an essential part of our lives, and we depend on it to carry out many of our daily tasks. Web applications have also become increasingly popular, and they have provided us with easy access to information and services. However, with the increase in the number of web applications, the number of attacks targeting them has also increased. Hackers have become more sophisticated, and they can now exploit vulnerabilities in web applications to steal data or cause other malicious actions. One way to protect web applications from such attacks is by implementing a Web Application Firewall (WAF). In this report, we will implement an open-source WAF on a Linux machine and demonstrate how it can be used to block attacks on a web application.

What is a Web Application Firewall?

A WAF is a security system that sits between a web application and the internet and monitors and filters HTTP traffic. The WAF can block attacks that exploit vulnerabilities in web applications. It can also prevent unauthorized access to sensitive information and protect against Denial of Service (DoS) attacks. A WAF can detect and block attacks such as cross-site scripting (XSS), SQL injection, and other web-based attacks.

Open-Source WAF:

We have chosen ModSecurity as our open-source WAF. ModSecurity is a popular WAF that is widely used in the industry. It is a free and open-source application that can be easily integrated into Apache or NGINX web servers.

Steps to Implement Open-Source WAF:

The following are the steps that we followed to implement ModSecurity on a Linux machine and demonstrate how it works:

Install ModSecurity

We used the package manager on our Linux machine to install ModSecurity. On Kali Linux, we used the following command:

```
sudo apt-get install libapache2-mod-security2
```

```
sudo a2enmod headers
```

```
sudo systemctl restart apache2
```

These commands installs ModSecurity on the Apache web server.

Step 2: Configure ModSecurity

After installing ModSecurity, we need to configure it to work with our web application. The configuration file for ModSecurity is located at **/etc/modsecurity/modsecurity.conf**. We modified this file to suit our needs. The configuration file contains rules that specify what traffic should be blocked and what traffic should be allowed.

NOTE:

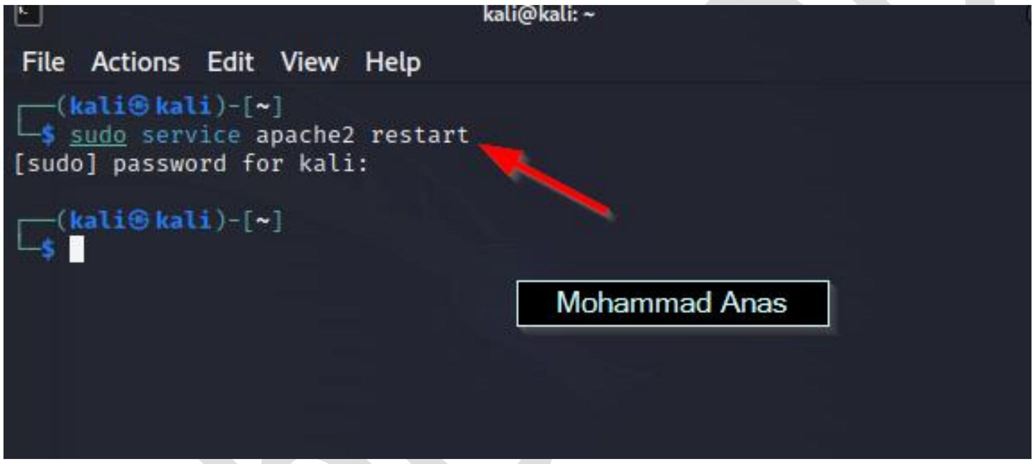
If you can see step by step guide of configuration of WAF then Click on Below Link:

WAF Configuration

Restart Apache

After configuring ModSecurity, we need to restart the Apache web server to apply the changes. We used the following command to restart Apache:

sudo service apache2 restart



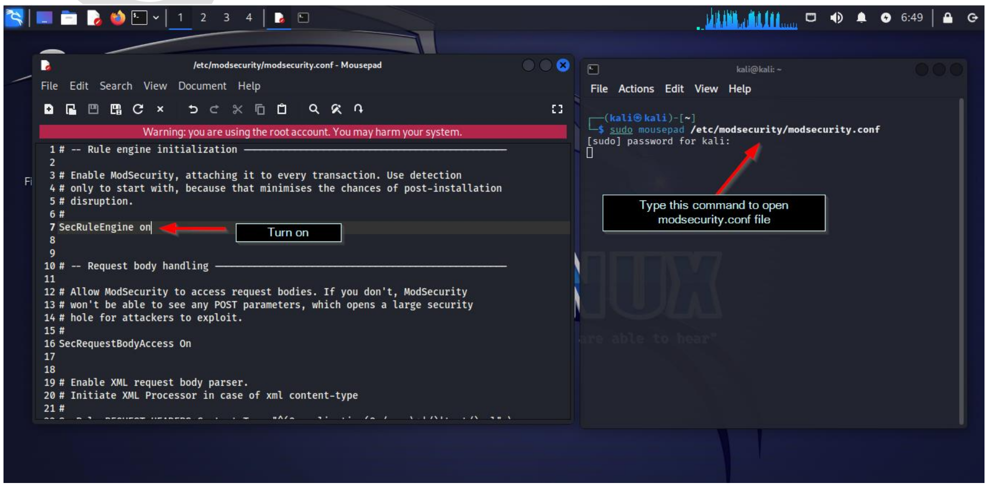
```
┌──(kali㉿kali)-[~]
└─$ sudo service apache2 restart
[sudo] password for kali:

┌──(kali㉿kali)-[~]
└─$ 
```

Turn On ModSecurity

1.1. We edited the configuration file for ModSecurity (/etc/modsecurity/modsecurity.conf) and set the following parameter:

sudo mousepad /etc/modsecurity/modsecurity.conf

SecRuleEngine On


/etc/modsecurity/modsecurity.conf - Mousepad
File Edit Search View Document Help
Warning: you are using the root account. You may harm your system.
Turn on
kali@kali: ~
File Actions Edit View Help
Type this command to open modsecurity.conf file
6:49

```
# -- Rule engine initialization ----------------------------------------------

# Enable ModSecurity, attaching it to every transaction. Use detection
# only to start with, because that minimises the chances of post-installation
# disruption.
#
SecRuleEngine on


# -- Request body handling ---------------------------------------------------

# Allow ModSecurity to access request bodies. If you don't, ModSecurity
# won't be able to see any POST parameters, which opens a large security
# hole for attackers to exploit.
#
SecRequestBodyAccess On


# Enable XML request body parser.
# Initiate XML Processor in case of xml content-type
#
```

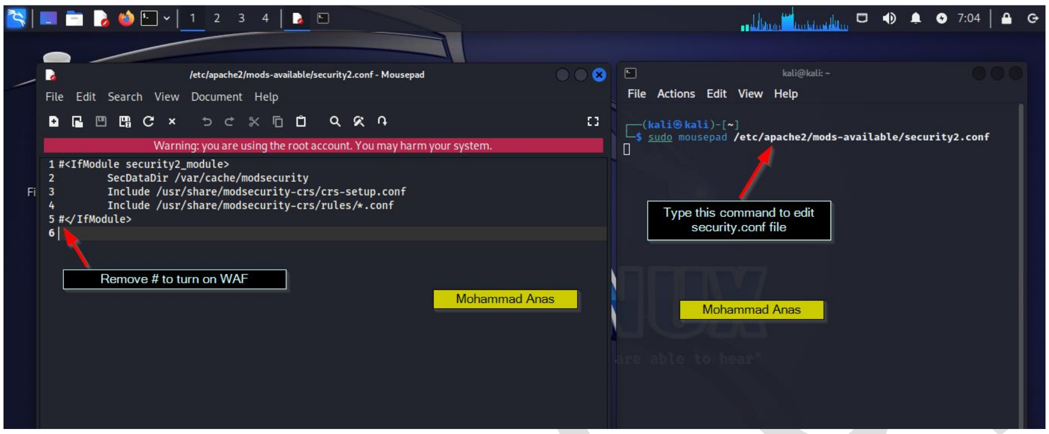
```
┌──(kali㉿kali)-[~]
└─$ sudo mousepad /etc/modsecurity/modsecurity.conf
[sudo] password for kali:
```

After changing the value from off to on save this file by **Ctrl+S** and exit.

1.2. Now edited the configuration file for turning on ModSecurity (/etc/apache2/mods available/security2.conf) and set the following parameter:

sudo mousepad /etc/apache2/mods-available/security2.conf

Remove # from the first line and last line and save this change by **Ctrl+S**



/etc/apache2/mods-available/security2.conf - Mousepad

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

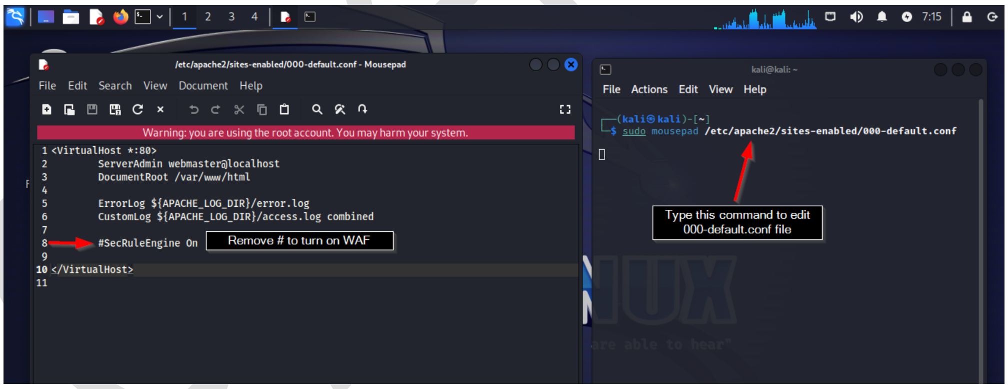
```
#<IfModule security2_module>
        SecDataDir /var/cache/modsecurity
        Include /usr/share/modsecurity-crs/crs-setup.conf
        Include /usr/share/modsecurity-crs/rules/*.conf
#</IfModule>
```

```
┌──(kali㉿kali)-[~]
└─$ sudo mousepad /etc/apache2/mods-available/security2.conf
```

1.3. Now edited the configuration file for turning on ModSecurity /etc/apache2/sites-enabled/000-default.conf) and set the following parameter:

sudo mousepad /etc/apache2/sites-enabled/000-default.conf

Remove # from the start of line 8 and save this change by **Ctrl+S**



/etc/apache2/sites-enabled/000-default.conf - Mousepad

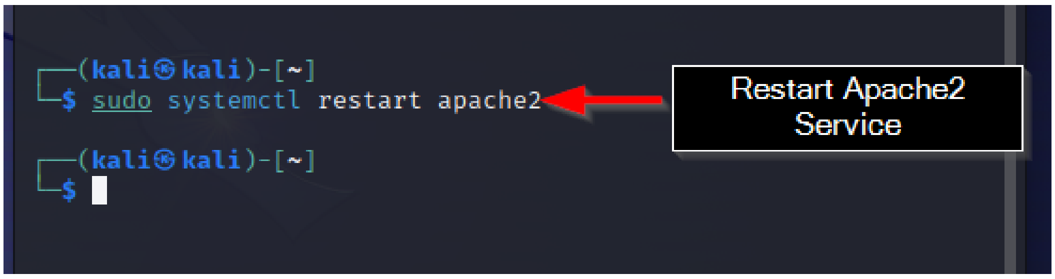
Warning: you are using the root account. You may harm your system.

```
1  <VirtualHost *:80>
2          ServerAdmin webmaster@localhost
3          DocumentRoot /var/www/html
4
5          ErrorLog ${APACHE_LOG_DIR}/error.log
6          CustomLog ${APACHE_LOG_DIR}/access.log combined
7
8          #SecRuleEngine On
9
10 </VirtualHost>
11
```

```
┌──(kali㉿kali)-[~]
└─$ sudo mousepad /etc/apache2/sites-enabled/000-default.conf
```

1.4. Now restart Apache2 server by typing following command.

sudo systemctl restart apache2



```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl restart apache2

┌──(kali㉿kali)-[~]
└─$ 
```

After restarting Apache, ModSecurity was turned on.

Testing ModSecurity

We tested ModSecurity by accessing our DVWA web application that install on Kali Linux if not install visit following link and trying to perform attacks. If ModSecurity is working properly, it should block the attack.

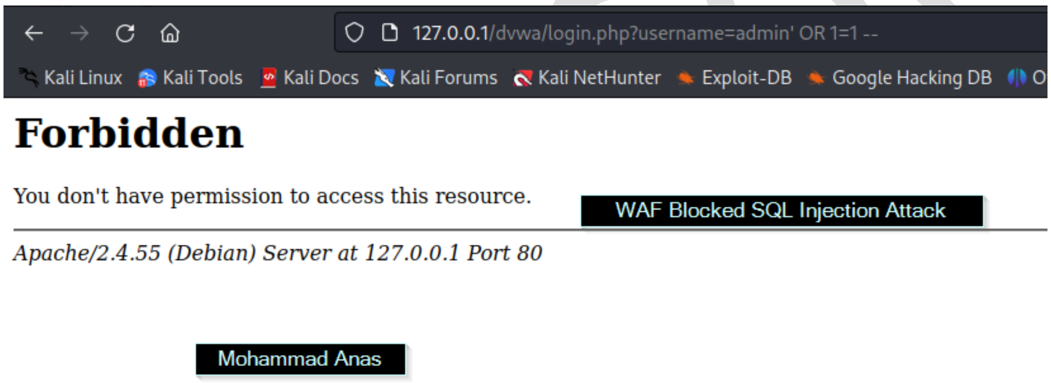
Install DVWA on Kali Linux

Perform a SQL injection attack

We accessed our web application and tried to perform a SQL injection attack. For example, we tried to access the following URL:

http://127.0.0.1/dvwa/login.php?username=admin' OR 1=1

If ModSecurity is working properly, it should block this attack and return a 403 Forbidden error.



Forbidden

You don't have permission to access this resource.

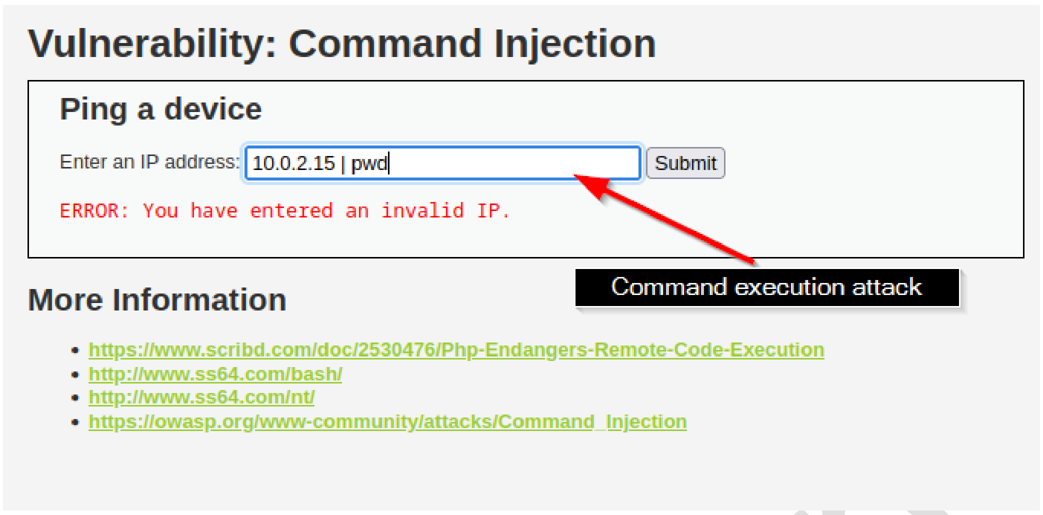
Apache/2.4.55 (Debian) Server at 127.0.0.1 Port 80

Here you can see Modsecurity blocked SQL Injection Attack.

Perform a Command Execution attack

We accessed our web application and tried to perform a SQL injection attack. For example I tried to enter IP Address with **pwd** command

10.0.2.15 | pwd


Vulnerability: Command Injection

Ping a device

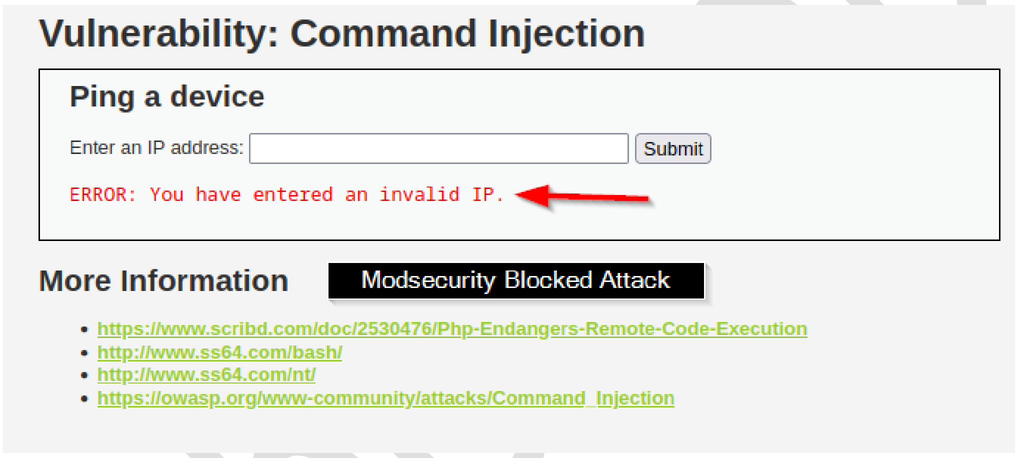
Enter an IP address: 10.0.2.15 | pwd

```
ERROR: You have entered an invalid IP.
```

More Information

- https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution
- http://www.ss64.com/bash/
- http://www.ss64.com/nt/
- https://owasp.org/www-community/attacks/Command_Injection

But Modsecurity blocked this attack and show a following error.


Vulnerability: Command Injection

Ping a device

Enter an IP address:

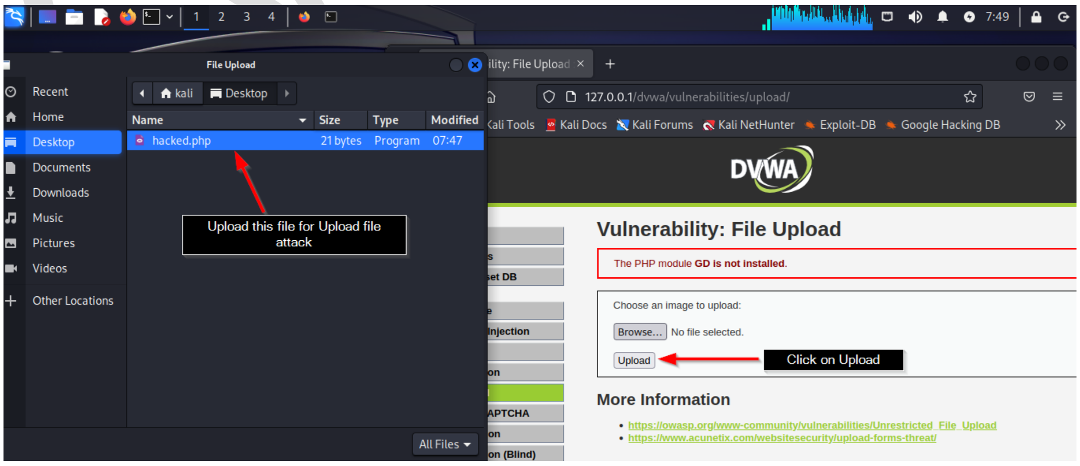
```
ERROR: You have entered an invalid IP.
```

More Information

- https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution
- http://www.ss64.com/bash/
- http://www.ss64.com/nt/
- https://owasp.org/www-community/attacks/Command_Injection

Perform a Malicious File Upload attack

Here I perform malicious file upload attack on DVWA by uploading a malicious file shown in blow image.



Name	Size	Type	Modified
hacked.php	21 bytes	Program	07:47

127.0.0.1/dvwa/vulnerabilities/upload/

Vulnerability: File Upload

The PHP module **GD is not installed**.

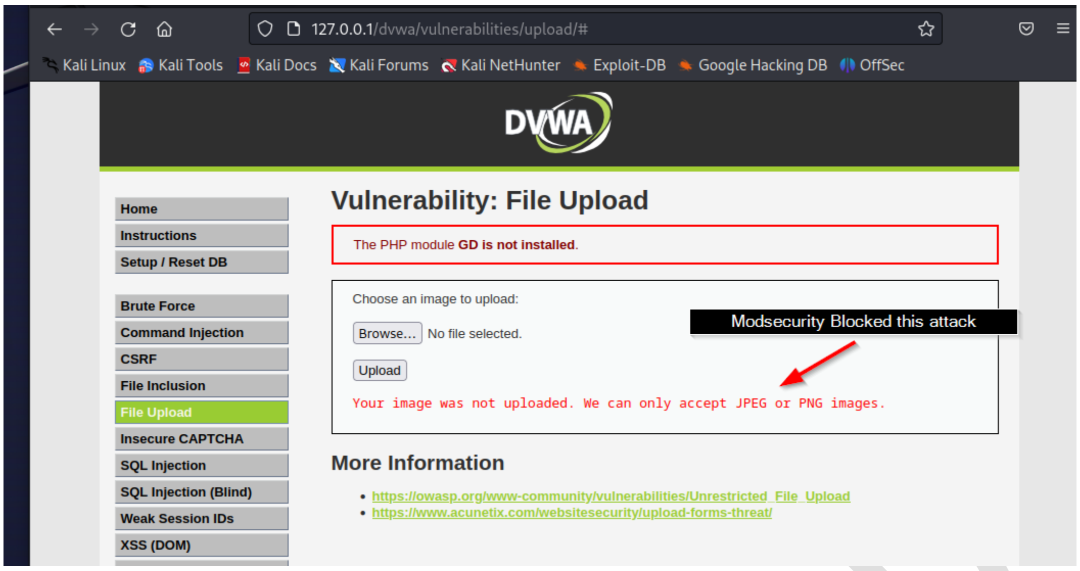
Choose an image to upload:

No file selected.

More Information

- https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
- https://www.acunetix.com/websitesecurity/upload-forms-threat/

After uploading that file and click on upload button web application show me a error because Modesecurity block this attack shown in below image.



127.0.0.1/dvwa/vulnerabilities/upload/#

Vulnerability: File Upload

The PHP module **GD is not installed**.

Choose an image to upload:

No file selected.

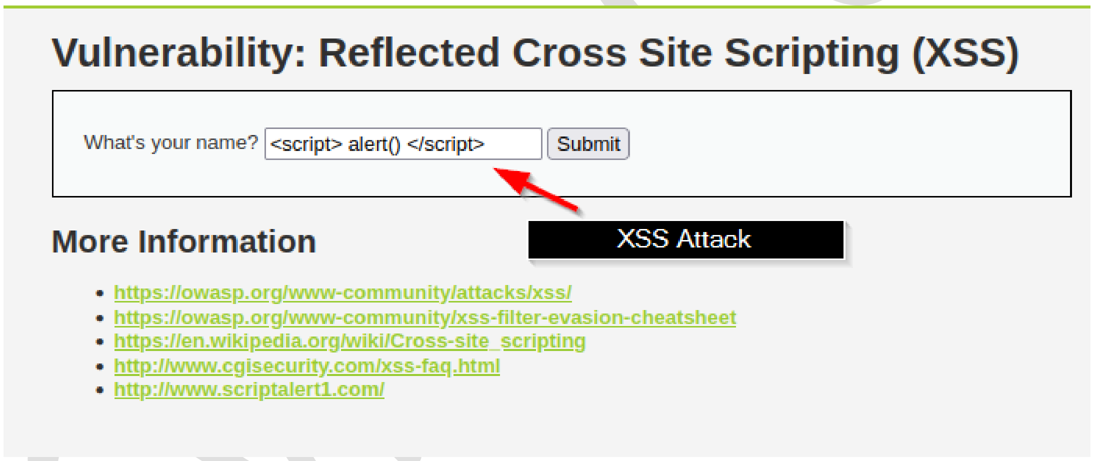
```
Your image was not uploaded. We can only accept JPEG or PNG images.
```

More Information

- https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
- https://www.acunetix.com/websitesecurity/upload-forms-threat/

Perform a XSS (Reflected) attack

Here I perform Cross-Site Scripting (XSS Reflected) attack on DVWA by uploading a malicious file shown in blow image.



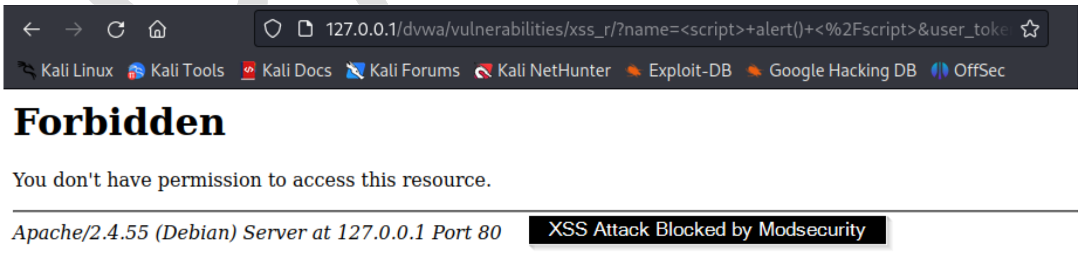
Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? <script> alert() </script>

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
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

Type a malicious script in input field and click on submit button web application show me a error because Modesecurity block this attack shown in below image.



127.0.0.1/dvwa/vulnerabilities/xss_r/?name=<script>+alert()+<%2Fscript>&user_toke

Forbidden

You don't have permission to access this resource.

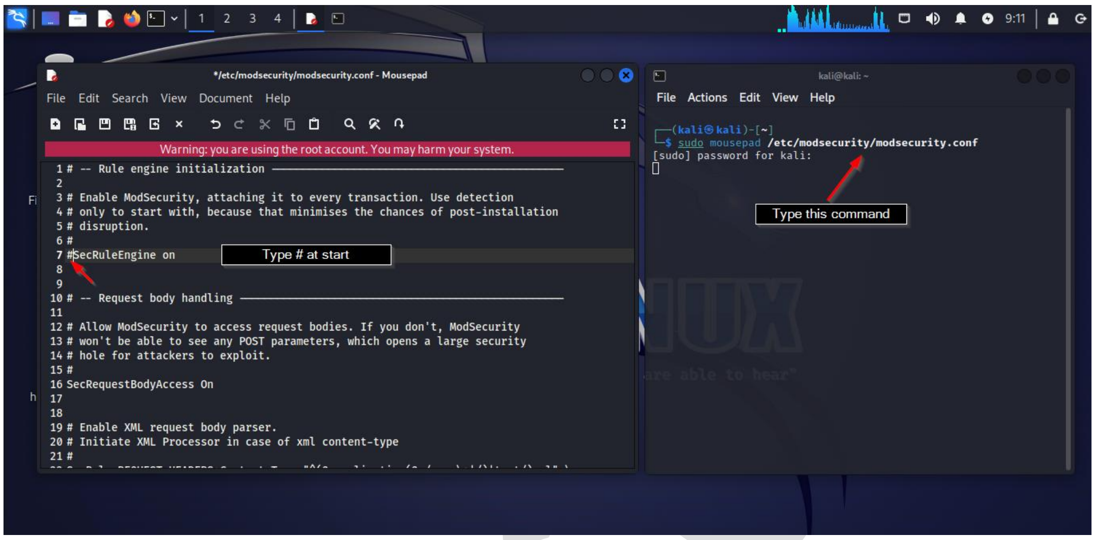
Apache/2.4.55 (Debian) Server at 127.0.0.1 Port 80

Turn off ModSecurity

2.1. We edited the configuration file for ModSecurity (/etc/modsecurity/modsecurity.conf) and set the following parameter:

sudo mousepad /etc/modsecurity/modsecurity.conf

SecRuleEngine off


*/etc/modsecurity/modsecurity.conf - Mousepad
File Edit Search View Document Help
Warning: you are using the root account. You may harm your system.
Type # at start
kali@kali: ~
File Actions Edit View Help
Type this command

```
# -- Rule engine initialization ----------------------------------------------

# Enable ModSecurity, attaching it to every transaction. Use detection
# only to start with, because that minimises the chances of post-installation
# disruption.
#
#SecRuleEngine on


# -- Request body handling ---------------------------------------------------

# Allow ModSecurity to access request bodies. If you don't, ModSecurity
# won't be able to see any POST parameters, which opens a large security
# hole for attackers to exploit.
#
SecRequestBodyAccess On


# Enable XML request body parser.
# Initiate XML Processor in case of xml content-type
#
```

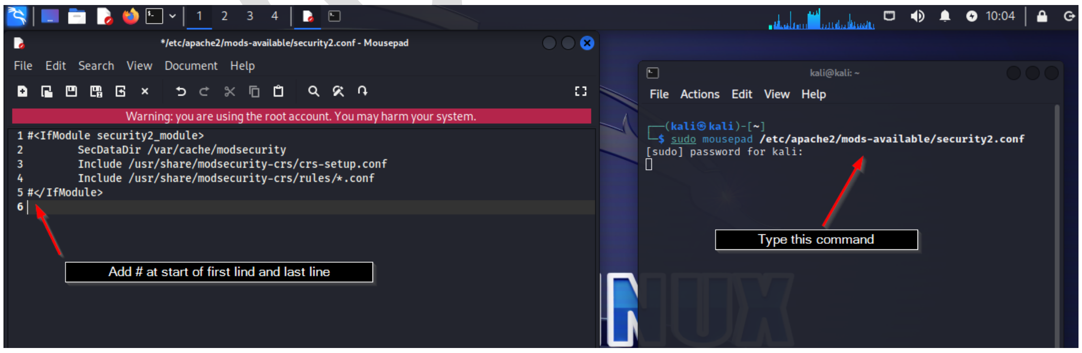
```
┌──(kali㉿kali)-[~]
└─$ sudo mousepad /etc/modsecurity/modsecurity.conf
[sudo] password for kali:
```

After changing the value from off to on save this file by **Ctrl+S** and exit.

2.2. Now edited the configuration file for turning off ModSecurity (/etc/apache2/mods available/security2.conf) and set the following parameter:

sudo mousepad /etc/apache2/mods-available/security2.conf

Add # from the first line and last line and save this change by **Ctrl+S**


*/etc/apache2/mods-available/security2.conf - Mousepad
File Edit Search View Document Help
Warning: you are using the root account. You may harm your system.
Add # at start of first lind and last line
kali@kali: ~
File Actions Edit View Help
Type this command

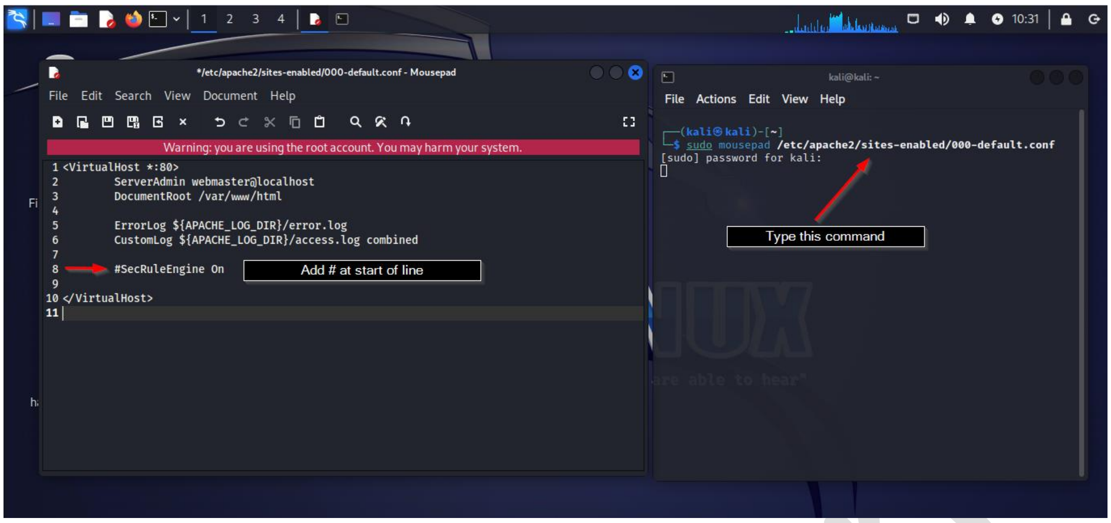
```
#<IfModule security2_module>
        SecDataDir /var/cache/modsecurity
        Include /usr/share/modsecurity-crs/crs-setup.conf
        Include /usr/share/modsecurity-crs/rules/*.conf
#</IfModule>
```

```
┌──(kali㉿kali)-[~]
└─$ sudo mousepad /etc/apache2/mods-available/security2.conf
[sudo] password for kali:
```

2.3. Now edited the configuration file for turning off ModSecurity (/etc/apache2/sites-enable/000-default.conf) and set the following parameter:

sudo mousepad /etc/apache2/sites-enable/000-default.conf

Add # from the start of line 8 and save this change by **Ctrl+S**

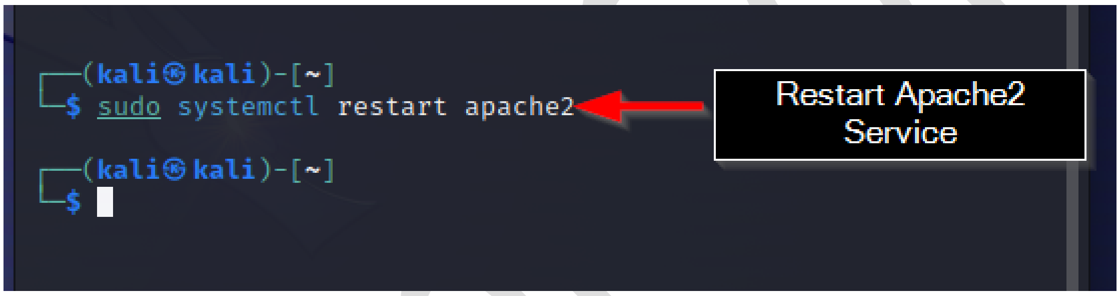

*/etc/apache2/sites-enabled/000-default.conf - Mousepad
File Edit Search View Document Help
Warning: you are using the root account. You may harm your system.
Add # at start of line
kali@kali: ~
File Actions Edit View Help
Type this command

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        #SecRuleEngine On

</VirtualHost>
```

```
┌──(kali㉿kali)-[~]
└─$ sudo mousepad /etc/apache2/sites-enabled/000-default.conf
[sudo] password for kali:
```

2.4. Now restart Apache2 server by typing following command.

sudo systemctl restart apache2



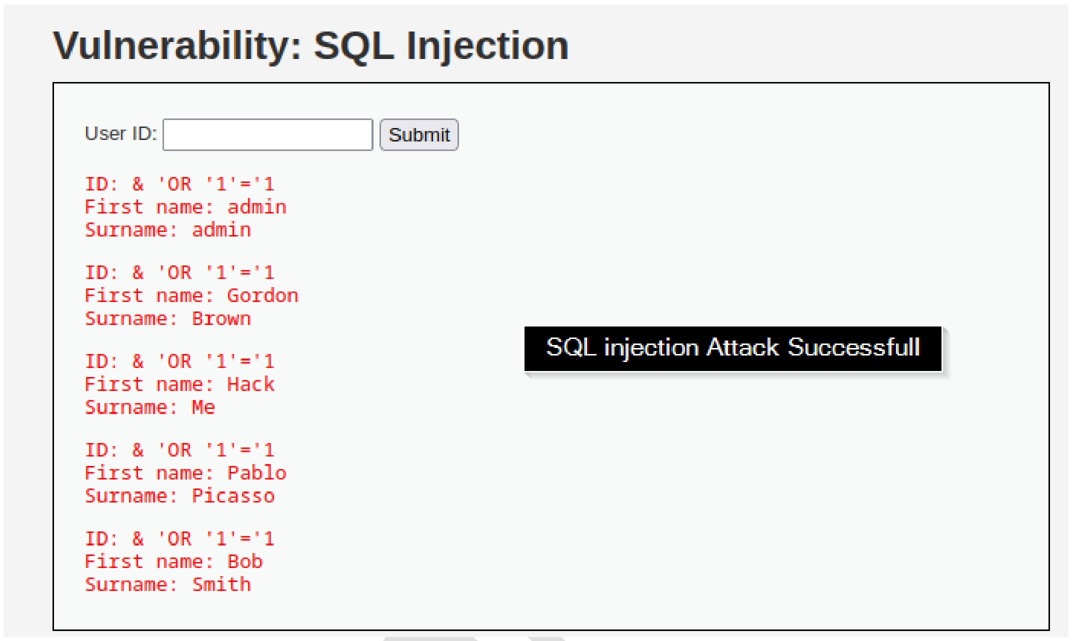
```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl restart apache2

┌──(kali㉿kali)-[~]
└─$ 
```

After restarting Apache, ModSecurity was turned off.

Testing ModSecurity (Modsecurity Off Now)

Perform a SQL injection attack

We accessed our web application and tried to perform the same SQL injection attack that we did before. This time, since ModSecurity was turned off, the attack was successful, as shown in image.

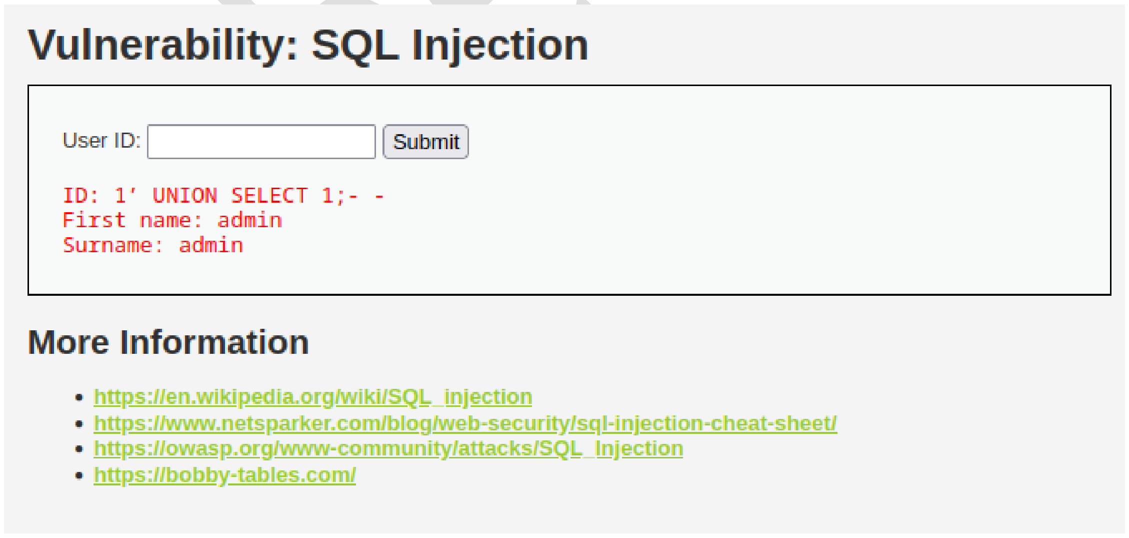


Vulnerability: SQL Injection

User ID: Submit

```
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
```

SQL injection Attack Successfull



Vulnerability: SQL Injection

User ID: Submit

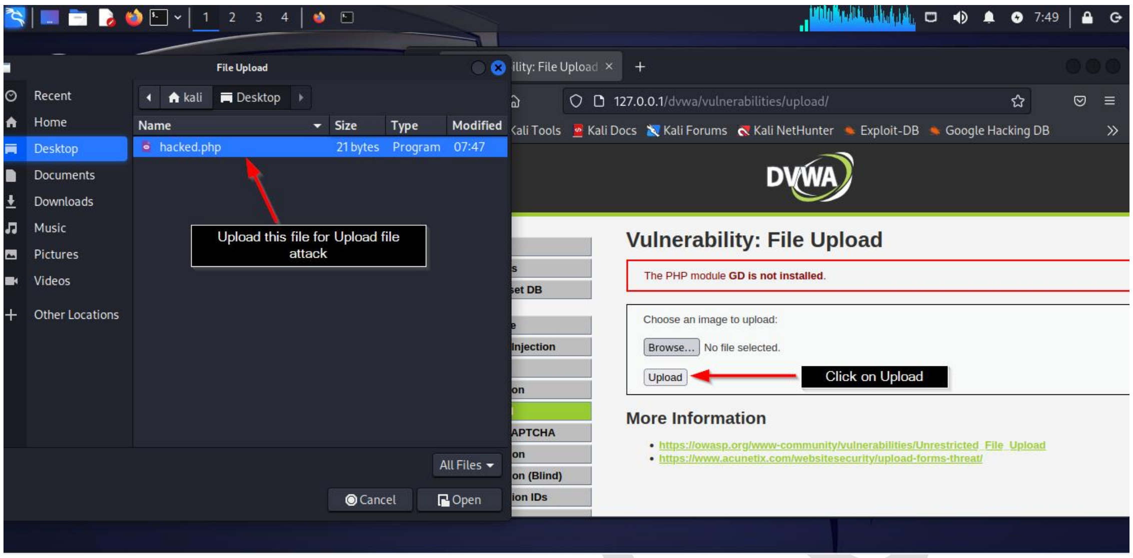
```
ID: 1' UNION SELECT 1;- -
First name: admin
Surname: admin
```

More Information

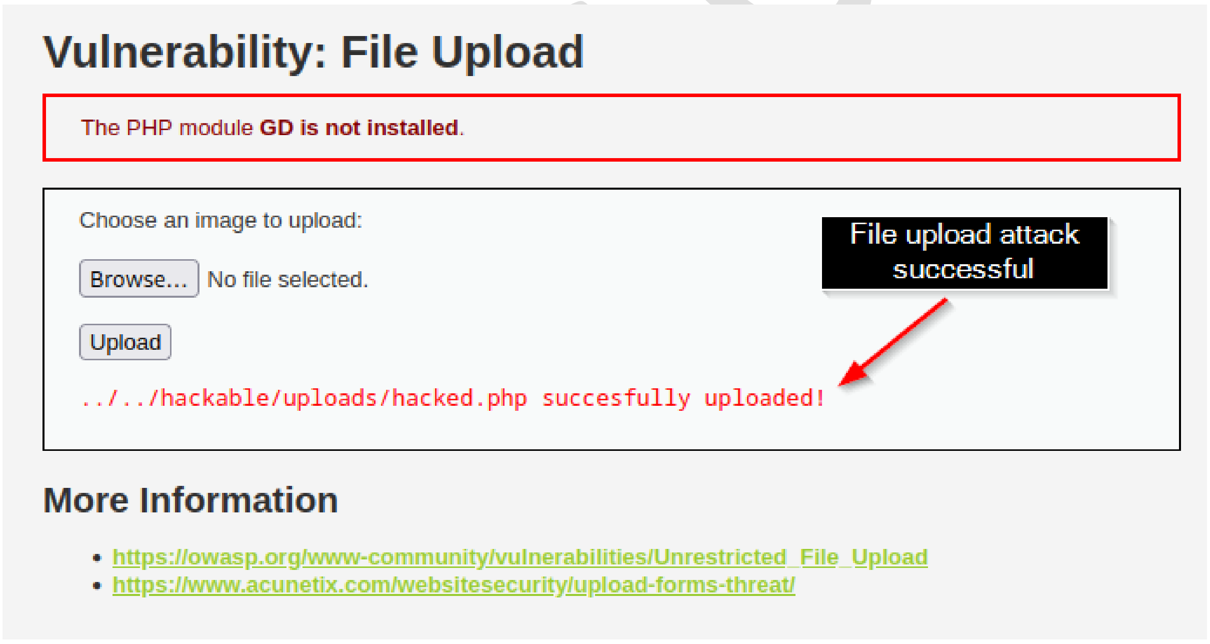
- https://en.wikipedia.org/wiki/SQL_injection
- https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
- https://owasp.org/www-community/attacks/SQL_Injection
- https://bobby-tables.com/

Perform a Malicious File Upload attack

Here I perform malicious file upload attack on DVWA by uploading a malicious file shown in blow image.



Now this time Malicious File upload attack successful, as you can show in below image.



Vulnerability: File Upload

The PHP module **GD is not installed**.

Choose an image to upload:

Browse... No file selected.

Upload

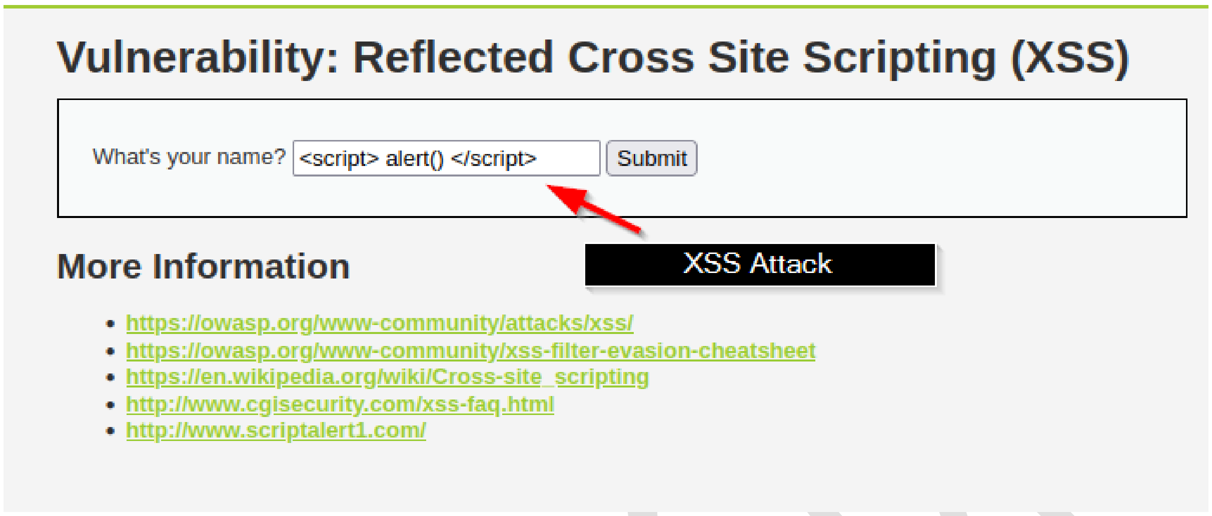
```
../../hackable/uploads/hacked.php succesfully uploaded!
```

More Information

- https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
- https://www.acunetix.com/websitesecurity/upload-forms-threat/

Perform a XSS (Reflected) attack

Here I perform Cross-Site Scripting (XSS Reflected) attack on DVWA by uploading a malicious file shown in blow image.



Vulnerability: Reflected Cross Site Scripting (XSS)

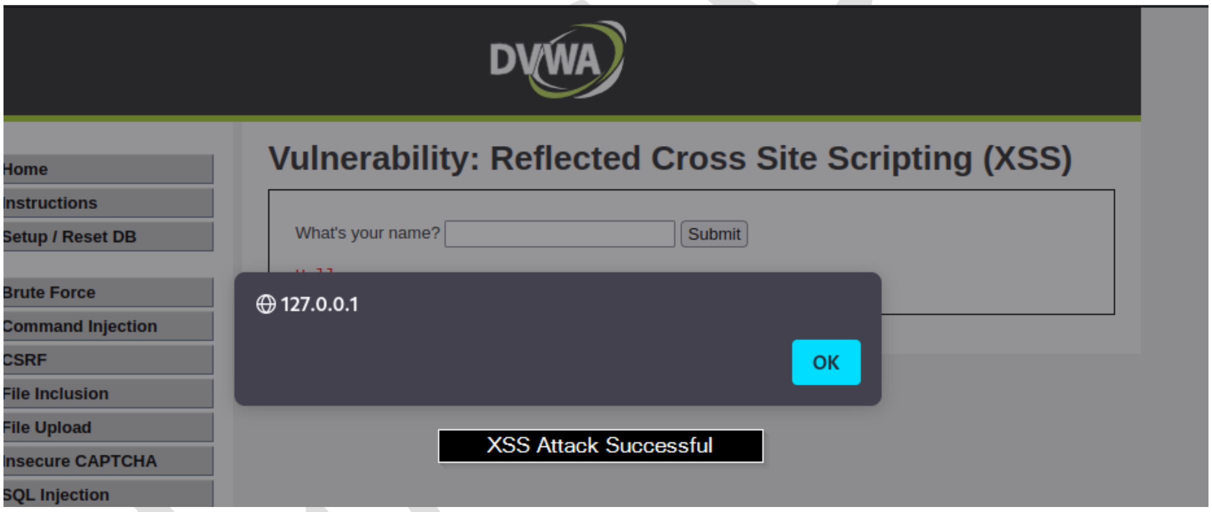
What's your name? <script> alert() </script> Submit

XSS Attack

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
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

Type a malicious script in input field and click on submit button here you can see XSS attack successfully launched, shown in below image.



Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? Submit

127.0.0.1

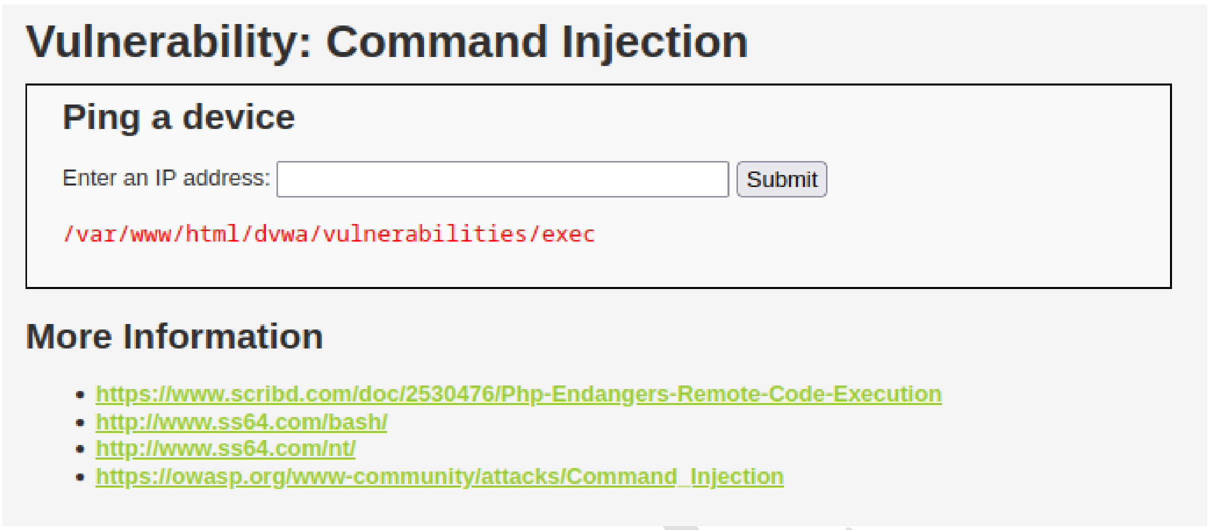
OK

XSS Attack Successful

Perform a Command Execution attack

We accessed our web application and tried to perform a SQL injection attack. For example I tried to enter IP Address with pwd command.

10.0.2.15 | pwd



Vulnerability: Command Injection

Ping a device

```
/var/www/html/dvwa/vulnerabilities/exec
```

More Information

- https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution
- http://www.ss64.com/bash/
- http://www.ss64.com/nt/
- https://owasp.org/www-community/attacks/Command_Injection

This time attack successful

Conclusion:

In this report, we have demonstrated how to implement an open-source WAF on a Linux machine and how it can be used to protect a web application from attacks. We have shown how ModSecurity can be used to block SQL injection attacks, Malicious File Upload Attack and XSS Attack on DVWA web application. By implementing a WAF, we can increase the security of our web application and protect it from attacks that can compromise sensitive data or cause other malicious actions. We recommend that all web application developers and administrators implement a WAF to protect their applications from attacks.

CYBER-COP
Your security is our top priority.

End Page