

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



SECURITY PRO

PROJECT # 4

W A F & D V W A



SECURITY PRO



INTRODUCTION

Name: Arslan Ali

Qualification: BS Software Engineering

Cohort: IEC Cybersecurity Cohort -5

Instructor: Shahzaib Ali Khan

Project: Web Application Firewall Implementation

Firewall: Modsecurity

Web: DVWA



SECURITY PRO



DVWA

Username: admin

Password: password

Login

```
o access official Ubuntu documentation, please visit:  
http://help.ubuntu.com/  
o mail.  
sfadmin@metasploitable:~$ ifconfig  
eth0      Link encap:Ethernet HWaddr 08:00:27:4b:b5:27  
          inet addr:192.168.0.101 Bcast:192.168.0.255 Mask:255.255.255.0  
          inet6 addr: fe80::a00:27ff:fe4b:b527/64 Scope:Link  
             UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
             RX packets:45 errors:0 dropped:0 overruns:0 frame:0  
             TX packets:71 errors:0 dropped:0 overruns:0 carrier:0  
             collisions:0 txqueuelen:1000  
             RX bytes:5805 (5.6 KB) TX bytes:7419 (7.2 KB)  
             Base address:0xd020 Memory:f0200000-f0220000  
  
o Link encap:Local Loopback  
    inet addr:127.0.0.1 Mask:255.0.0.0  
    inet6 addr: ::1/128 Scope:Host  
       UP LOOPBACK RUNNING MTU:16436 Metric:1  
       RX packets:92 errors:0 dropped:0 overruns:0 frame:0  
       TX packets:92 errors:0 dropped:0 overruns:0 carrier:0  
       collisions:0 txqueuelen:0  
       RX bytes:19393 (18.9 KB) TX bytes:19393 (18.9 KB)  
sfadmin@metasploitable:~$
```

DAMN VULNERABLE WEB APPLICATION (DVWA)

Username: admin

Password: password

In this project, I installed the Virtual box on my laptop and then install the mirror of metasploitable 2. The default username and password of the metasploitable 2 is "msfadmin" and find ip address of this machine and open it in Firefox

[Home](#)[Instructions](#)[Setup](#)[Brute Force](#)[Command Execution](#)[CSRF](#)[File Inclusion](#)[SQL Injection](#)[SQL Injection \(Blind\)](#)[Upload](#)[XSS reflected](#)[XSS stored](#)[DVWA Security](#)[PHP Info](#)[About](#)

Welcome to Damn Vulnerable Web App!

Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a class room environment.

WARNING!

Damn Vulnerable Web App is damn vulnerable! Do not upload it to your hosting provider's public html folder or any internet facing web server as it will be compromised. We recommend downloading and installing [XAMPP](#) onto a local machine inside your LAN which is used solely for testing.

Disclaimer

We do not take responsibility for the way in which any one uses this application. We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

General Instructions

The help button allows you to view hits/tips for each vulnerability and for each security level on their respective page.

You have logged in as 'admin'



TESTING THE DVWA



Before applying the firewall

All the attacks are performed easily like XSS, SQL injection, Command line, and Malicious file upload.

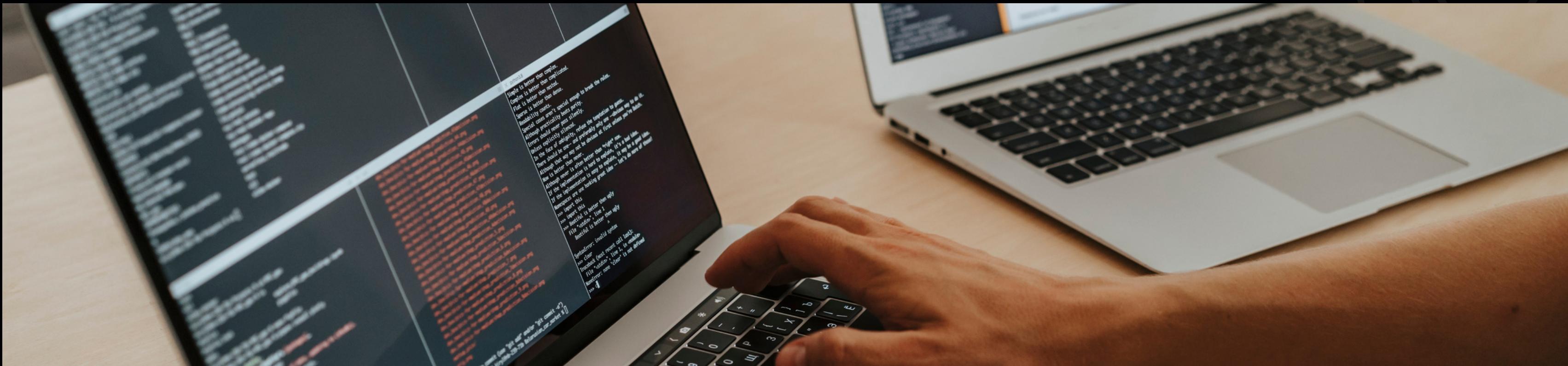


After applying the firewall

After Applying the firewall all the attacks and malicious traffic is blocked. We can see it practically



SECURITY-PRO



TEST DVWA WITHOUT FIREWALL

Malicious File Upload
SQL Injection & SQL Blind
XSS Reflected & Stored
HTML Injection Reflected & Stored
Command Line:
File Inclusion

XSS ATTACK(STORED)

The screenshot shows the DVWA application's XSS stored page. On the left, a sidebar lists various security vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored (which is highlighted in green), DVWA Security, PHP Info, and About. The main content area has a title "Vulnerability: Stored Cross Site Scripting (XSS)". It contains two input fields: "Name *" with the value ">" and "Message *" with the value "<script>alert(document.cookie)</script>". Below these fields is a "Sign Guestbook" button. To the right, there are several examples of stored XSS attacks, each consisting of a "Name:" field and a "Message:" field. One example shows "Name: test" and "Message: This is a test comment.". Another shows "Name: arslan" and "Message: hello boss". A third shows "Name: >" and "Message: ". A fourth shows "Name: a" and "Message: ". At the bottom, there is a "Name: arslan" entry. The DVWA logo is at the top of the page.

The screenshot shows the DVWA application's XSS stored page with a confirmation dialog box overlaid. The dialog box contains the IP address "192.168.1.5", the session ID "PHPSESSID=0615bd9c5aff79a8b6302413838723e1", and a checkbox labeled "Don't allow 192.168.1 to prompt you again". There is an "OK" button on the right side of the dialog. The background page shows the same interface as the previous screenshot, with the "XSS stored" vulnerability selected in the sidebar. The DVWA logo is at the top of the page.

Code:

```
<script>alert(document.cookie);</script>
```

Severity:

High

Risk:

- Session hijacking
- Malware distribution
- Steal cookies
- Website defacement



XSS ATTACK(REFLECTED)



The screenshot shows a Firefox browser window on a Kali Linux desktop. The address bar shows the URL `192.168.1.5/dvwa/vulnerabilities/xss_r/?name=<script>alert(2)<%2Fscript>#`. The DVWA interface displays a form with the placeholder "What's your name?" and a red error message "Hello". The left sidebar menu is visible, showing various exploit categories, with "XSS reflected" selected.

The screenshot shows the same DVWA interface after the exploit was triggered. A modal dialog box appears with the text "192.168.1.5" and "It is a Vunlability". An "OK" button is visible at the bottom right of the dialog.

Code:

```
<script>alert("Hello i am error")</script>
```

Severity:

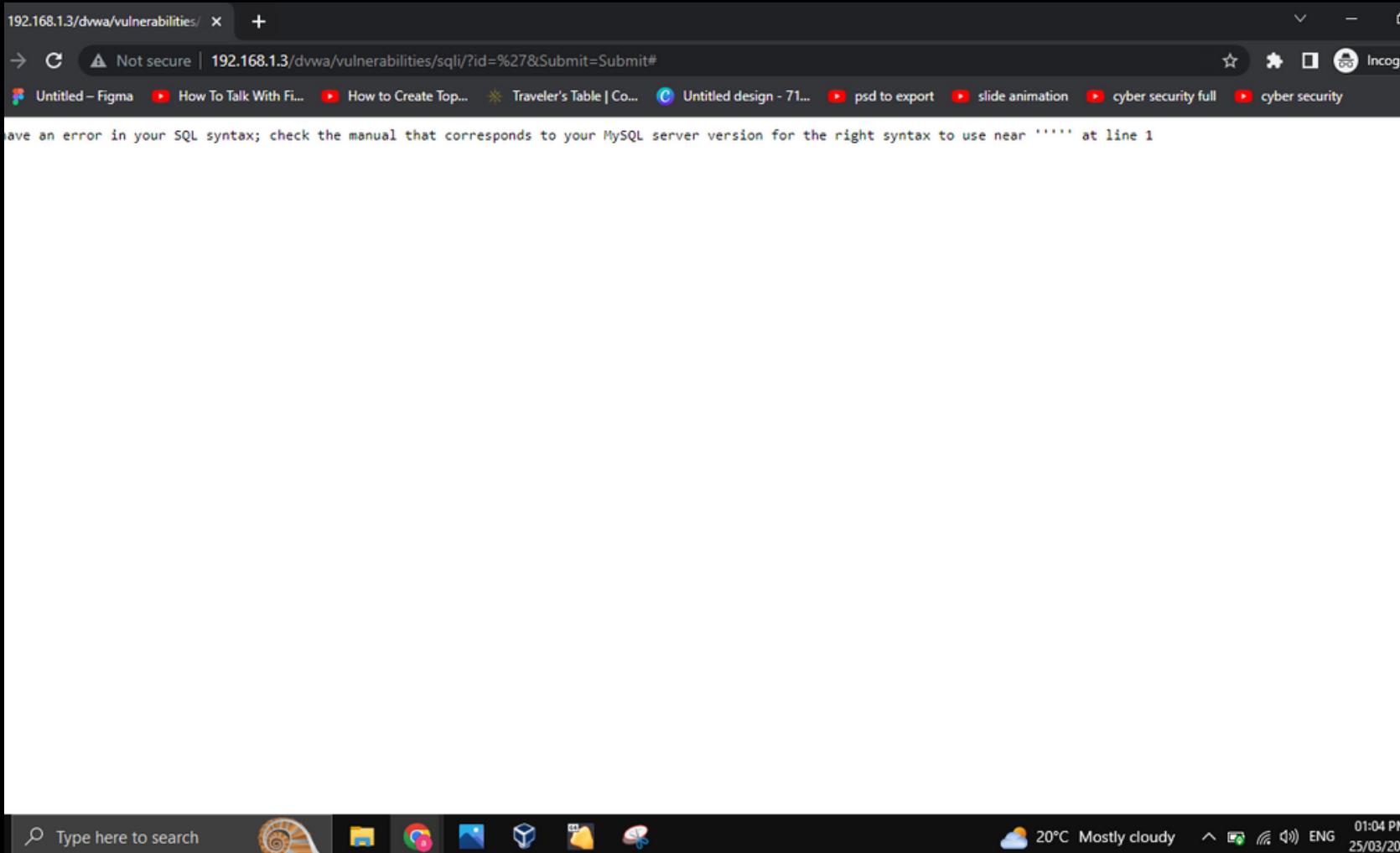
High

Risk:

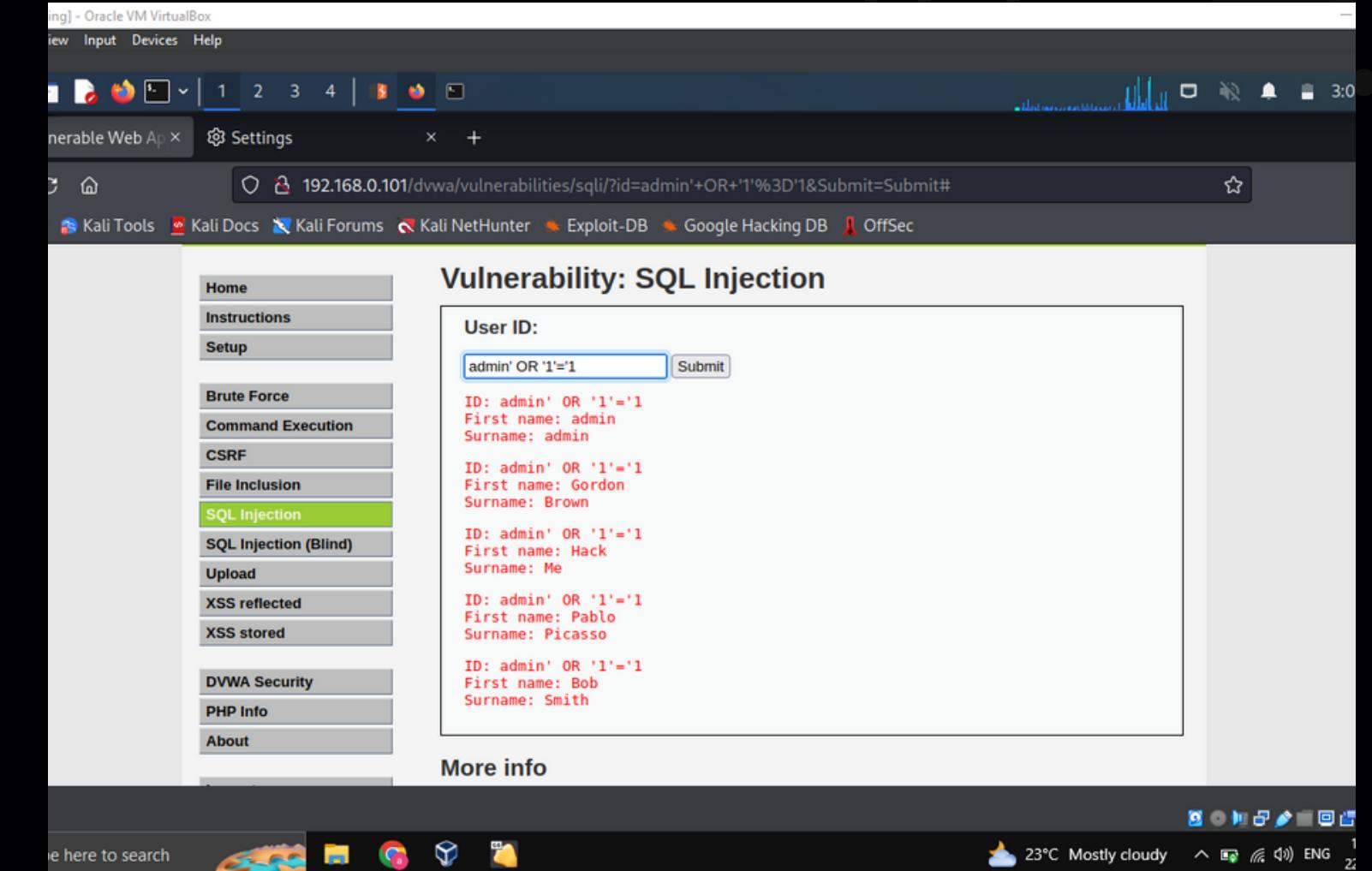
- Steal sensitive information
- Steal cookies
- Modify the appearance of a website



SQL INJECTION



A screenshot of a web browser window. The address bar shows the URL `192.168.1.3/dvwa/vulnerabilities/sqli/?id=%27&Submit=Submit#`. The page content displays an error message: "You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '''' at line 1". Below the error message is a navigation menu with items like "Home", "Instructions", "Setup", "Brute Force", "Command Execution", "CSRF", "File Inclusion", "SQL Injection" (which is highlighted in green), "SQL Injection (Blind)", "Upload", "XSS reflected", "XSS stored", "DVWA Security", "PHP Info", and "About". At the bottom of the screen, there is a taskbar with various icons and a system status bar showing the date and time.



A screenshot of a Kali Linux desktop environment. The main window title is "Vulnerable Web App" and the sub-title is "192.168.0.101/dvwa/vulnerabilities/sqli/?id=admin' OR '1'='1&Submit=Submit#". On the left, a sidebar menu lists the same items as the first screenshot. The main content area shows a form with a "User ID" input field containing "admin' OR '1='1". Below the form, a list of user records is displayed, each with an ID, first name, and surname. The first record is redacted. The list includes:

ID	First name	Surname
ID: admin' OR '1='1	First name: admin	Surname: admin
ID: admin' OR '1='1	First name: Gordon	Surname: Brown
ID: admin' OR '1='1	First name: Hack	Surname: Me
ID: admin' OR '1='1	First name: Pablo	Surname: Picasso
ID: admin' OR '1='1	First name: Bob	Surname: Smith

Code:

`admin' OR '1='1`

Severity:

Critical

Risk:

- Unauthorized data access
- Data manipulation or deletion
- Server compromise



BLIND SQL INJECTION



Damn Vulnerable Web App (DVWA) - Not secure | 192.168.1.3/dvwa/vulnerabilities/sql_injection/?id=%27&Submit=Submit#

User ID: Submit

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQl_injection
<http://www.unixwiz.net/tctips/sql-injection.html>

Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout

Username: admin

Damn Vulnerable Web App - Oracle VM VirtualBox

User ID: Submit

ID: 2' OR '1'='1
First name: admin
Surname: admin

ID: 2' OR '1'='1
First name: Gordon
Surname: Brown

ID: 2' OR '1'='1
First name: Hack
Surname: Me

ID: 2' OR '1'='1
First name: Pablo
Surname: Picasso

ID: 2' OR '1'='1
First name: Bob
Surname: Smith

Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About

Code:

`admin' OR '1'='1`

Severity:

Critical

Risk:

- Unauthorized data access
- Data manipulation or deletion
- Server compromise



HTML INJECTION (STORED)



Kali Linux [Running] - Oracle VM VirtualBox

Damn Vulnerable Web App | cookies - Google Search

192.168.1.5/dvwa/vulnerabilities/xss_s/

DVWA

Vulnerability: Stored Cross Site Scripting (XSS)

Name * >1
</textarea><h1>I want to hack you</h1>

Message * <textarea><h1>I want to hack you</h1>

Sign Guestbook

Name: test
Message: This is a test comment.

Name: arslan
Message: hello boss

Name: >

More info

<http://ha.ckers.org/xss.html>
http://en.wikipedia.org/wiki/Cross-site_scripting
<http://www.cgisecurity.com/xss-faq.html>

Kali Linux [Running] - Oracle VM VirtualBox

Damn Vulnerable Web App | cookies - Google Search

192.168.1.5/dvwa/vulnerabilities/xss_s/

DVWA

Usernames: admin

Name: >
Message:

I want to hack you

More info

<http://ha.ckers.org/xss.html>
http://en.wikipedia.org/wiki/Cross-site_scripting
<http://www.cgisecurity.com/xss-faq.html>

Code:

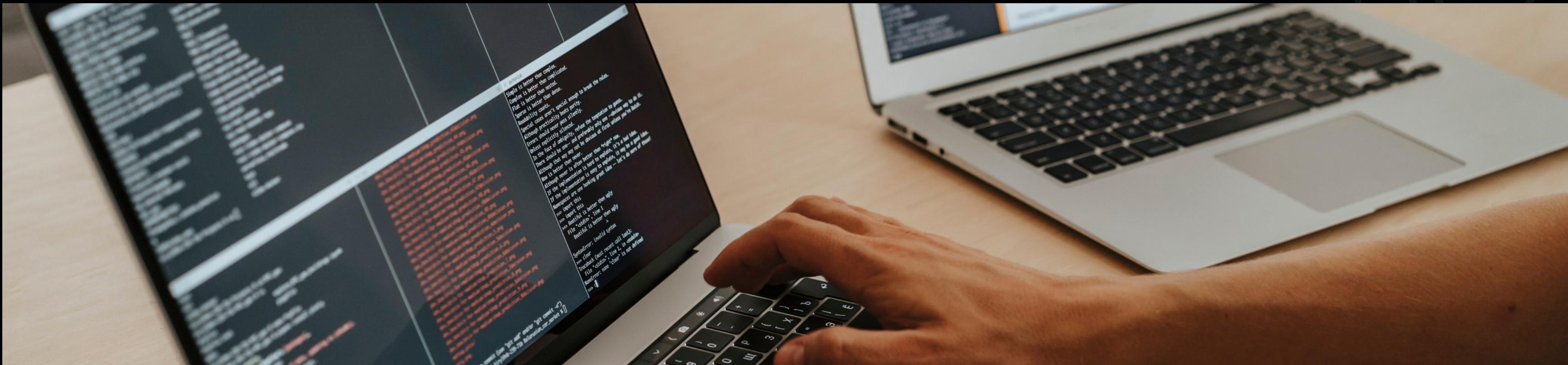
/textarea><h1>I want to hack You</h1>

Severity:

High

Risk:

- Session hijacking
- Malware distribution
- Steal cookies
- Website defacement



TEST DVWA WITH FIREWALL

1st we install the ModSecurity AND after it install it and implement it and than again test the DVWA



```
└─# sudo nano /etc/apt/sources.list
└─(root㉿kali)-[~/home/mrhacker]
└─# sudo apt update
Get:1 http://kali.cs.nycu.edu.tw/kali kali-rolling InRelease [41.2 kB]
Get:2 http://kali.cs.nycu.edu.tw/kali kali-rolling/contrib Sources [77.2 kB]
Get:3 http://kali.cs.nycu.edu.tw/kali kali-rolling/main Sources [15.7 MB]
Get:4 http://kali.cs.nycu.edu.tw/kali kali-rolling/non-free Sources [130 kB]
Get:5 http://kali.cs.nycu.edu.tw/kali kali-rolling/main amd64 Packages [19.3 MB]
Get:6 http://kali.cs.nycu.edu.tw/kali kali-rolling/main amd64 Contents (deb) [44.7 MB]
Get:7 http://kali.cs.nycu.edu.tw/kali kali-rolling/contrib amd64 Packages [116 kB]
Get:8 http://kali.cs.nycu.edu.tw/kali kali-rolling/contrib amd64 Contents (deb) [172 kB]
Get:9 http://kali.cs.nycu.edu.tw/kali kali-rolling/non-free amd64 Packages [217 kB]
Get:10 http://kali.cs.nycu.edu.tw/kali kali-rolling/non-free amd64 Contents (deb) [927 kB]
Fetched 81.3 MB in 7min 20s (185 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
1741 packages can be upgraded. Run 'apt list --upgradable' to see them.
└─# sudo apt install libapache2-mod-security2 -y
└─(root㉿kali)-[~/home/mrhacker]
```

INSTALL MODSECURITY

All of the 1st we update our apt in the Kali Linux terminal. After it we will install Modsecurity by using this command:

"sudo apt install libapache2-mod-security2 -y"



```
Setting up liblua5.1-0:amd64 (5.1.5-9) ...
Setting up libc6-i386 (2.36-8) ...
Setting up libc-dev-bin (2.36-8) ...
Setting up libapache2-mod-security2 (2.9.7-1) ...
apache2_invoke: Enable module security2
Setting up libc6-dev:amd64 (2.36-8) ...
Processing triggers for libc-bin (2.36-8) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for kali-menu (2022.3.1) ...
```

```
(root㉿kali)-[~/home/mrhacker]
└─# sudo a2enmod headers      Setup
Enabling module headers.
To activate the new configuration, you need to run:
  systemctl restart apache2  Brute Force
```

```
(root㉿kali)-[~/home/mrhacker]
└─# sudo systemctl restart apache2
```

```
(root㉿kali)-[~/home/mrhacker]
└─# sudo cp /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf
```

Vulnerability: File Upload

Choose an image to upload:

Browse... No file selected.

Upload

Your image was not uploaded.

More info

http://192.168.1.100/dvwa/vulnerabilities/file_upload/

INSTALL MODSECURITY

Now we will install the the header of Apache-2 server and also restart it by using the following commands:

"sudo a2enmod headers"

"sudo systemctl restart apache2"



```
Setting up liblua5.1-0:amd64 (5.1.5-9) ...
Setting up libc6-i386 (2.36-8) ...
Setting up libc-dev-bin (2.36-8) ...
Setting up libapache2-mod-security2 (2.9.7-1) ...
apache2_invoke: Enable module security2
Setting up libc6-dev:amd64 (2.36-8) ...
Processing triggers for libc-bin (2.36-8) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for kali-menu (2022.3.1) ...
```

```
(root㉿kali)-[~/home/mrhacker]
└─# sudo a2enmod headers      Setup
Enabling module headers.
To activate the new configuration, you need to run:
  systemctl restart apache2  Brute Force
```

```
(root㉿kali)-[~/home/mrhacker]
└─# sudo systemctl restart apache2
```

```
(root㉿kali)-[~/home/mrhacker]
└─# sudo cp /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf
  http://www.modsecurity.org/documentation/file-upload
  http://www.modsecurity.org/documentation/file-upload
```

Vulnerability: File Upload

Choose an image to upload:

Browse... No file selected.

Upload

Your image was not uploaded.

More info

<http://www.modsecurity.org/documentation/file-upload>

CONFIGURING MODSECURITY

Now we will Configuring ModSecurity by using the following commands:

```
"sudo cp /etc/modsecurity/modsecurity.conf-
recommended
/etc/modsecurity/modsecurity.conf
"
```



```
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  # The ServerName directive sets the request scheme, hostname and port that
11  # the server uses to identify itself. This is used when creating
12  # redirection URLs. In the context of virtual hosts, the ServerName
13  # specifies what hostname must appear in the request's Host: header to
14  # match this virtual host. For the default virtual host (this file) this
15  # value is not decisive as it is used as a last resort host regardless.
16  # However, you must set it for any further virtual host explicitly.
17  #ServerName www.example.com
18
19  ServerAdmin webmaster@localhost
20  DocumentRoot /var/www/html
21
22  # vim: syntax=apache ts=4 sw=4 sts=4 sr=0 et=1
```

CONFIGURING MODSECURITY

We will change the file which has the following path address file and after this, we will restart the server again.

`/etc/modsecurity/modsecurity.conf`
`SecRuleEngine On`



```
Preparing to unpack .../git_1%3a2.39.2-1.1_amd64.deb ...
Unpacking git (1:2.39.2-1.1) over (1:2.35.1-1) ...
Preparing to unpack .../git-man_1%3a2.39.2-1.1_all.deb ...
Unpacking git-man (1:2.39.2-1.1) over (1:2.35.1-1) ...
Setting up git-man (1:2.39.2-1.1) ...
Setting up git (1:2.39.2-1.1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for kali-menu (2022.3.1) ...
```



Vulnerability: File Upload

```
(root㉿kali)-[~/home/mrhacker]
└─# sudo git clone https://github.com/coreruleset/coreruleset /usr/share/modsecurity-crs
Cloning into '/usr/share/modsecurity-crs' ...
remote: Enumerating objects: 25894, done.
remote: Counting objects: 100% (85/85), done.
remote: Compressing objects: 100% (47/47), done.
remote: Total 25894 (delta 41), reused 74 (delta 38), pack-reused 25809
Receiving objects: 100% (25894/25894), 6.48 MiB | 221.00 KiB/s, done.
Resolving deltas: 100% (20239/20239), done.
```

```
(root㉿kali)-[~/home/mrhacker]
└─# sudo mv /usr/share/modsecurity-crs/crs-setup.conf.example /usr/share/modsecurity-crs/crs-setup.conf
More info
http://owasp.org/owasp-modsecurity-crs-project/owasp-modsecurity-crs-faq.html#How_to_upload_a_new_rule_set
```

OWASP CORE RULE SET APPLY ON FIREWALL

OWASP Core Rule Set applies on firewallModSecurity by using the following commands:

- 1) Remove current rules of ModSecurity:
`"sudo rm -rf /usr/share/modsecurity-crs"`
- 2) we installed the git by using this command
`"sudo apt install git"`
- 3) we Clone the OWASP-CRS GitHub repository
`"sudo git clon https://github.com/coreruleset/coreruleset /usr/share/modsecurity-crs"`



OWASP CORE RULE SET & ENABLING MODSECURITY

1) Now we should rename the " crs-setup.conf".example to "crs-setup.conf" by the following command:

```
"sudo mv /usr/share/modsecurity-crs/crs-setup.conf.example /usr/share/modsecurity-crs/crs-setup.conf"
```

2) Rename the default request exclusion rule file by using this 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"
```

3) After this we must be edit the file "security2.conf" and paste the following code:

```
""SecDataDir /var/cache/modsecurity Include /usr/share/modsecurity-crs/crs-setup.conf Include  
/usr/share/modsecurity-crs/rules/*.conf"
```

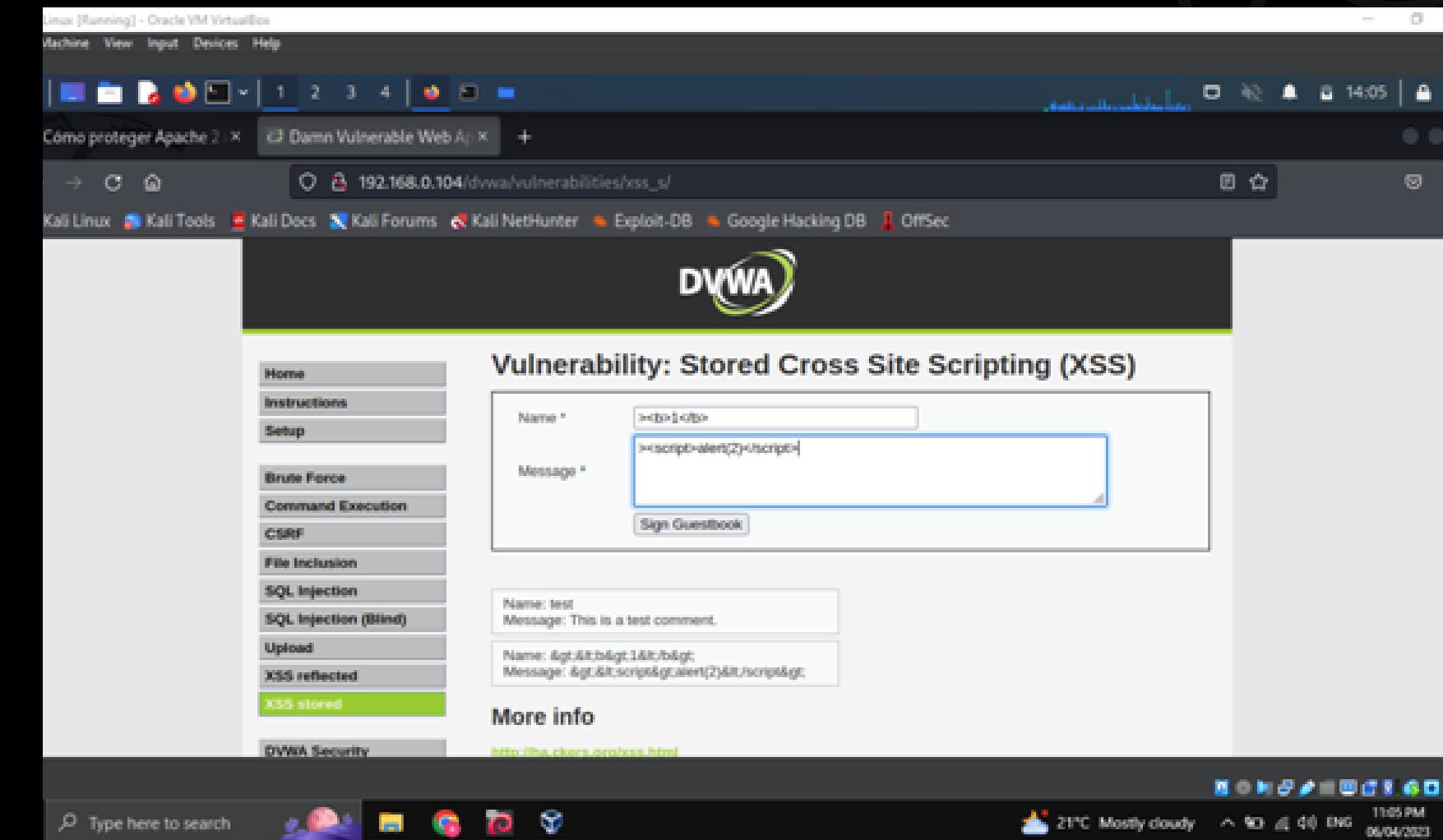
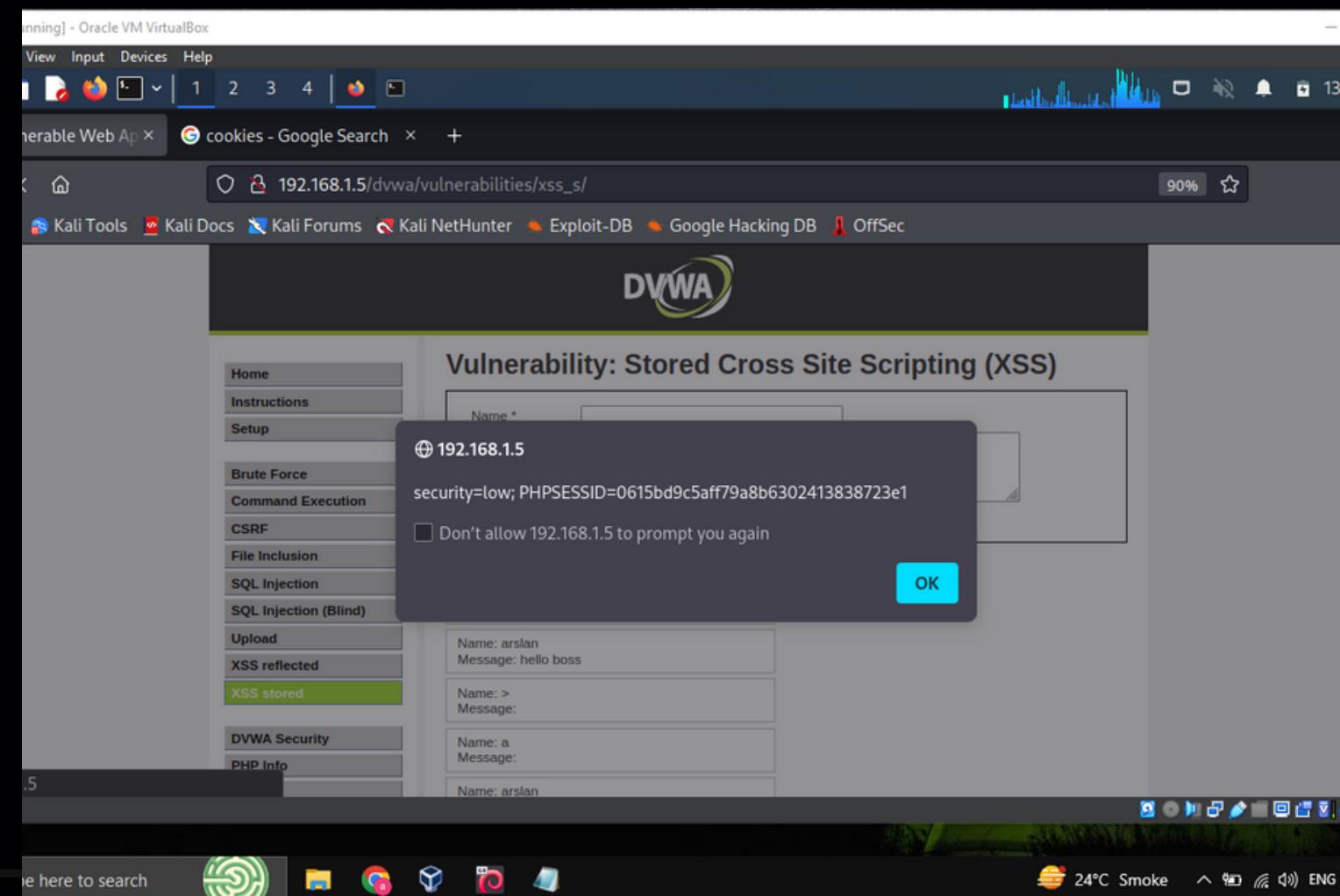
3) After this we must be edit the file "/etc/apache2/sites-enabled/000-default.conf" and paste the following code:

```
""" ServerAdmin webmaster@localhost  
DocumentRoot /var/www/html  
ErrorLog ${APACHE_LOG_DIR}/error.log  
CustomLog ${APACHE_LOG_DIR}/access.log combined
```



```
1 # -- Rule engine initialisation
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 #
```

NOW ITS TIME TO TEST THE FIREWALL MODSECURITY



XSS ATTACK(STORED)



SQL INJECTION



Oracle VM VirtualBox

Input Devices Help

Apache 2 Damn Vulnerable Web App +

192.168.0.104/dvwa/vulnerabilities/sql/

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

DVWA

Vulnerability: SQL Injection

User ID: admin' OR '1='1

Submit

More info

<http://www.securiteam.com/securityreviews/SDP2HCP76L.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/tips/sql-injection.html>

Home Instructions Setup

Brute Force Command Execution CSRF File Inclusion SQL Injection SQL Injection (Blind) Upload XSS reflected XSS stored

DVWA Security

Search

21°C Mostly cloudy

VM VirtualBox

Devices Help

Apache 2 Damn Vulnerable Web App +

192.168.0.104/dvwa/vulnerabilities/sql/?id=admin'+OR+'1%3D1&Submit=Submit

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

DVWA

Vulnerability: SQL Injection

User ID:

Submit

More info

<http://www.securiteam.com/securityreviews/SDP2HCP76L.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/tips/sql-injection.html>

Home Instructions Setup

Brute Force Command Execution CSRF File Inclusion SQL Injection SQL Injection (Blind) Upload XSS reflected XSS stored

DVWA Security

Search

21°C Mostly cloudy



MALICIOUS FILE



Kali Linux [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Como proteger Apache 2 | Damn Vulnerable Web App

192.168.0.104/dvwa/vulnerabilities/upload/

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

DVWA

Vulnerability: File Upload

Choose an image to upload:
Browse... No file selected.

Upload

Your image was not uploaded.

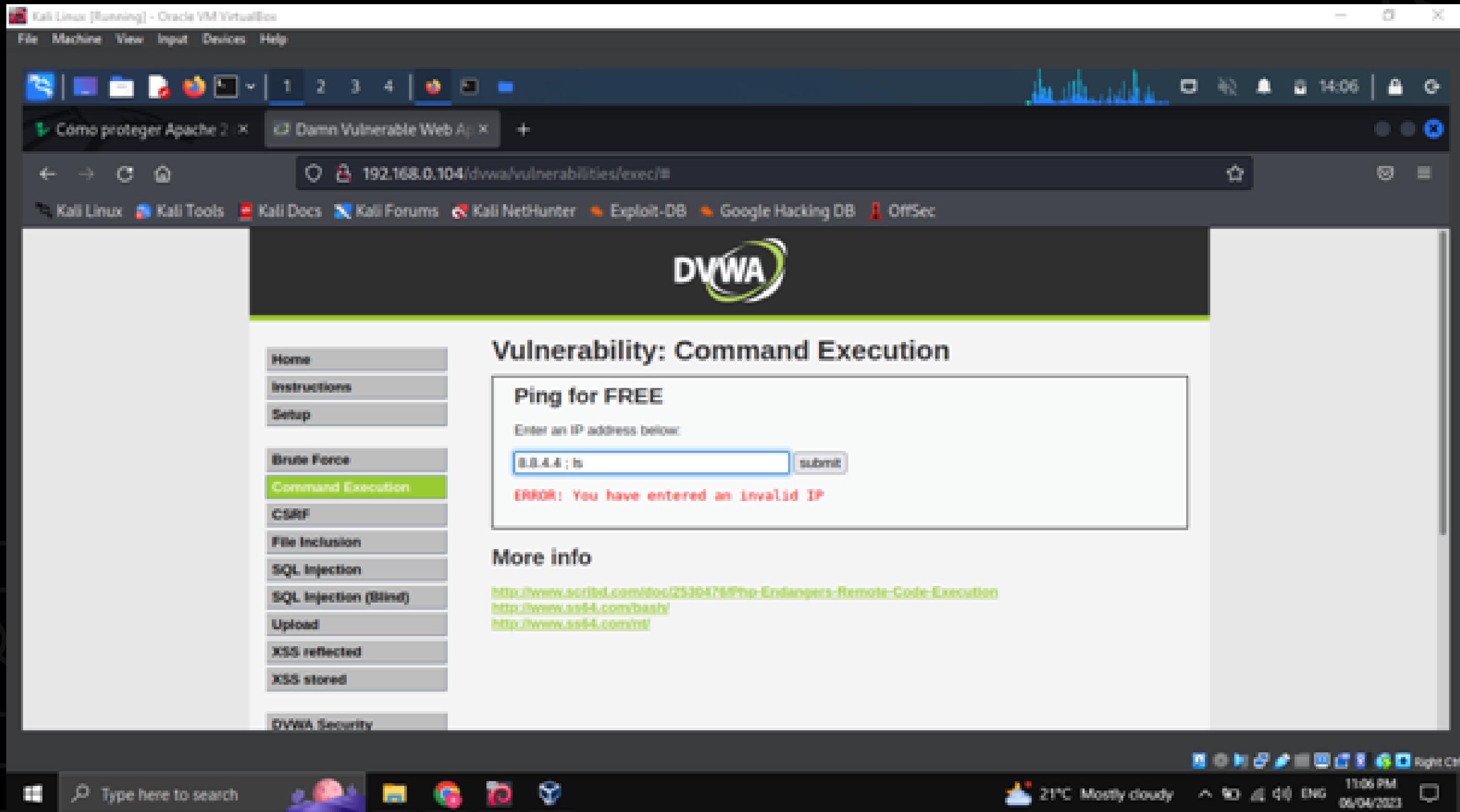
More info

http://www.owasp.org/index.php/Unrestricted_File_Upload
<http://blogs.securteam.com/index.php/archives/1268>
<http://www.usenix.com/web/base/security/upload-forms-threat.htm>

Type here to search

21°C Mostly cloudy 11:02 PM 06/04/2023

COMMAND LINE:



HTML INJECTIONS

This screenshot shows a Kali Linux desktop environment with a running Oracle VM VirtualBox window. Inside the VirtualBox window, a Firefox browser is open to the DVWA (Damn Vulnerable Web Application) 'Reflected Cross Site Scripting (XSS)' page. The URL is `http://192.168.0.104/dvwa/vulnerabilities/xss_r/`. The DVWA navigation menu on the left includes options like Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected (which is highlighted in green), XSS stored, and DVWA Security. The main content area displays a form with the placeholder 'What's your name?' and a red error message below it: 'Hello! To <h>Hack this now</h>'.

This screenshot shows a Kali Linux desktop environment with a running Oracle VM VirtualBox window. Inside the VirtualBox window, a Firefox browser is open to the DVWA 'Reflected Cross Site Scripting (XSS)' page. The URL is `http://192.168.0.104/dvwa/vulnerabilities/xss_r/`. The DVWA navigation menu on the left includes options like Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected (which is highlighted in green), XSS stored, and DVWA Security. The main content area displays a form with the placeholder 'What's your name?' and a red error message below it: 'Hello! To <h>Hack this now</h>'.



WAF PROJECT 4

THANK YOU

S E C U R I T Y P R O

