

Name: SHILPA

Date:13.03.2023

Task: 3

1.commands execution vulnerability:

Command execution vulnerabilities can arise through the use of external libraries or third-party code. If the code is not properly vetted or updated, it may contain vulnerabilities that can be exploited by an attacker.

To mitigate command execution vulnerabilities, it is important to properly validate user input and to use up-to-date and secure third-party code. Additionally, it is recommended to use sandboxing or other isolation techniques to prevent an attacker from executing arbitrary commands on the system.

Command injection is an attack in which the goal is execution of arbitrary commands on the host operating system via a vulnerable application. Command injection attacks are possible when an application passes unsafe user supplied data (forms, cookies, HTTP headers etc.) to a system shell.

i. Security level: low



The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The top navigation bar includes links for Home, Instructions, Setup, Bruteforce, Command Execution (highlighted), CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: Command Execution". It features a "Ping for FREE" section with a text input field for an IP address and a "submit" button. Below the input field, the text "help", "index.php", and "source" is displayed in red. A "More info" section contains three links: <http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>, <http://www.ss54.com/bash/>, and <http://www.ss54.com/nit/>.

ii. Security level: medium



This screenshot is identical to the one for the low security level, showing the DVWA interface with the "Command Execution" vulnerability page. The navigation bar, main title, "Ping for FREE" section with IP input and "submit" button, red text output ("help", "index.php", "source"), and "More info" links are all present and match the previous image.

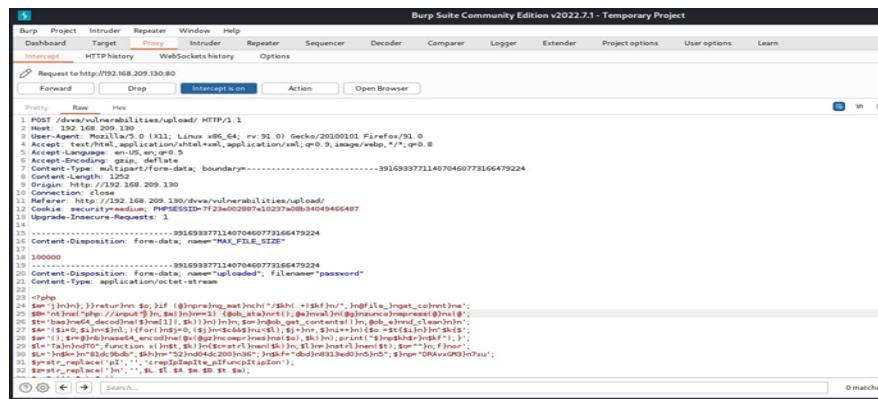
iii. Security level: high



2.File Upload Vulnerability:

File upload vulnerability is a security flaw that allows an attacker to upload malicious files to a web server, which can then be executed on the server or downloaded by other users. This vulnerability can be exploited by attackers to compromise the confidentiality, integrity, and availability of the web application and its data. There are several ways in which file upload vulnerabilities can arise. One common way is through inadequate file type validation. If a

i.Security level: low



ii.Security level: medium

Burp Suite Community Edition v2022.7.1 - Temporary Project

Burp	Project	Intruder	Repeater	Window	Help
Dashboard	Target	Proxy	Intruder	Repeater	Sequencer
Intercept	HTTP history	WebSockets history	Options	Decoder	Comparer
Logger	Extender	Project options	User options	Learn	

Request to http://192.168.209.130:80

Forward Drop Intercept Action Open Browser

Priority	Row	How
1	POST /dev/api/vulnerabilities/upload/ HTTP/1.1	
2	Host: 192.168.209.130	
3	User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0	
4	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8	
5	Accept-Language: en-US,en;q=0.5	
6	Accept-Encoding: gzip, deflate	
7	Content-Type: multipart/form-data; boundary=-----39169977114070460773166479224	
8	Content-Length: 125	
9	Origin: http://192.168.209.130	
10	Connection: Close	
11	Referer: http://192.168.209.130/dev/api/vulnerabilities/upload/	
12	Cookie: security_token=PdPzSGDwF7ZuecdBkVx1227vblB3KtM4G6g87	
13	Upgrade-Insecure-Requests: 1	
14		
15	-----39169977114070460773166479224	
16	Content-Disposition: form-data; name="MAX_FILE_SIZE"	
17		
18	1000000	
19	-----39169977114070460773166479224	
20	Content-Disposition: form-data; name="uploaded"; filename="password"	
21	Content-Type: application/octet-stream	
22		
23	<html>	
24	<!--[if]--><script>return true; if ([" "<script>alert(1)</script>"]<script>alert(1)</script>]<script>alert(1)</script>)</script>	
25	<!--[endif]--></script>	
26	<!--[if]--><script>document.write(1); </script>	
27	<!--[endif]--></script>	
28	<!--[if]--><script>document.write(1); </script>	
29	<!--[endif]--></script>	
30	<!--[if]--><script>document.write(1); </script>	
31	<!--[endif]--></script>	
32	<!--[if]--><script>document.write(1); </script>	
33	<!--[endif]--></script>	
34	<!--[if]--><script>document.write(1); </script>	
35	<!--[endif]--></script>	
36	<!--[if]--><script>document.write(1); </script>	
37	<!--[endif]--></script>	
38	<!--[if]--><script>document.write(1); </script>	
39	<!--[endif]--></script>	
40	<!--[if]--><script>document.write(1); </script>	
41	<!--[endif]--></script>	
42	<!--[if]--><script>document.write(1); </script>	
43	<!--[endif]--></script>	
44	<!--[if]--><script>document.write(1); </script>	
45	<!--[endif]--></script>	
46	<!--[if]--><script>document.write(1); </script>	
47	<!--[endif]--></script>	
48	<!--[if]--><script>document.write(1); </script>	
49	<!--[endif]--></script>	
50	<!--[if]--><script>document.write(1); </script>	
51	<!--[endif]--></script>	
52	<!--[if]--><script>document.write(1); </script>	
53	<!--[endif]--></script>	
54	<!--[if]--><script>document.write(1); </script>	
55	<!--[endif]--></script>	
56	<!--[if]--><script>document.write(1); </script>	
57	<!--[endif]--></script>	
58	<!--[if]--><script>document.write(1); </script>	
59	<!--[endif]--></script>	
60	<!--[if]--><script>document.write(1); </script>	
61	<!--[endif]--></script>	
62	<!--[if]--><script>document.write(1); </script>	
63	<!--[endif]--></script>	
64	<!--[if]--><script>document.write(1); </script>	
65	<!--[endif]--></script>	
66	<!--[if]--><script>document.write(1); </script>	
67	<!--[endif]--></script>	
68	<!--[if]--><script>document.write(1); </script>	
69	<!--[endif]--></script>	
70	<!--[if]--><script>document.write(1); </script>	
71	<!--[endif]--></script>	
72	<!--[if]--><script>document.write(1); </script>	
73	<!--[endif]--></script>	
74	<!--[if]--><script>document.write(1); </script>	
75	<!--[endif]--></script>	
76	<!--[if]--><script>document.write(1); </script>	
77	<!--[endif]--></script>	
78	<!--[if]--><script>document.write(1); </script>	
79	<!--[endif]--></script>	
80	<!--[if]--><script>document.write(1); </script>	
81	<!--[endif]--></script>	
82	<!--[if]--><script>document.write(1); </script>	
83	<!--[endif]--></script>	
84	<!--[if]--><script>document.write(1); </script>	
85	<!--[endif]--></script>	
86	<!--[if]--><script>document.write(1); </script>	
87	<!--[endif]--></script>	
88	<!--[if]--><script>document.write(1); </script>	
89	<!--[endif]--></script>	
90	<!--[if]--><script>document.write(1); </script>	
91	<!--[endif]--></script>	
92	<!--[if]-->	

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

DVWA

Vulnerability: File Upload

Choose an image to upload:

Browse...

No file selected.

Upload

../../../../hackable/uploads/pass successfully uploaded!

More Information

- https://www.owasp.org/index.php/Unrestricted_File_Upload
- <https://www.acunetix.com/websitesecurity/upload-forms-threat/>

iii. Security level: high



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

Vulnerability: File Upload

Choose an image to upload:

No file selected.

../../hackable/uploads/pass succesfully uploaded!


More Information

- https://www.vowasp.org/index.php/Unrestricted_File_Upload
- <https://www.acunetix.com/website-security/upload-forms-threat/>

3.Sql Injection Vulnerability:

SQL injection vulnerability is a type of security flaw that allows an attacker to inject malicious SQL code into an application's database. This vulnerability can be exploited by attackers to compromise the confidentiality, integrity, and availability of the database and its data. Once an attacker has successfully injected malicious SQL code, they can perform various malicious actions such as accessing sensitive data, modifying or deleting existing data, or even taking complete control of the database server.

i.Security level: low



[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](#)

Vulnerability: SQL Injection

User ID:

ID: %' or '0' = '0
First name: admin
Surname: admin

ID: %' or '0' = '0
First name: Gordon
Surname: Brown


ID: %' or '0' = '0
First name: Hack
Surname: Me

ID: %' or '0' = '0
First name: Pablo
Surname: Picasso

ID: %' or '0' = '0
First name: Bob
Surname: Smith

More info
<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

ii.Security level: medium



[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](#)

Vulnerability: SQL Injection

User ID:

ID: %' or '0' = '0
First name: admin
Surname: admin

ID: %' or '0' = '0
First name: Gordon
Surname: Brown

ID: %' or '0' = '0
First name: Hack
Surname: Me

ID: %' or '0' = '0
First name: Pablo
Surname: Picasso

ID: %' or '0' = '0
First name: Bob
Surname: Smith

More info
<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

iii. Security level: high

DVWA

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

Vulnerability: SQL Injection

User ID:

ID: %' or '0' = '0
First name: admin
Surname: admin

ID: %' or '0' = '0
First name: Gordon
Surname: Brown

ID: %' or '0' = '0
First name: Hack
Surname: Me

ID: %' or '0' = '0
First name: Pablo
Surname: Picasso

ID: %' or '0' = '0
First name: Bob
Surname: Smith

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unbwiz.net/techtips/sql-injection.html>

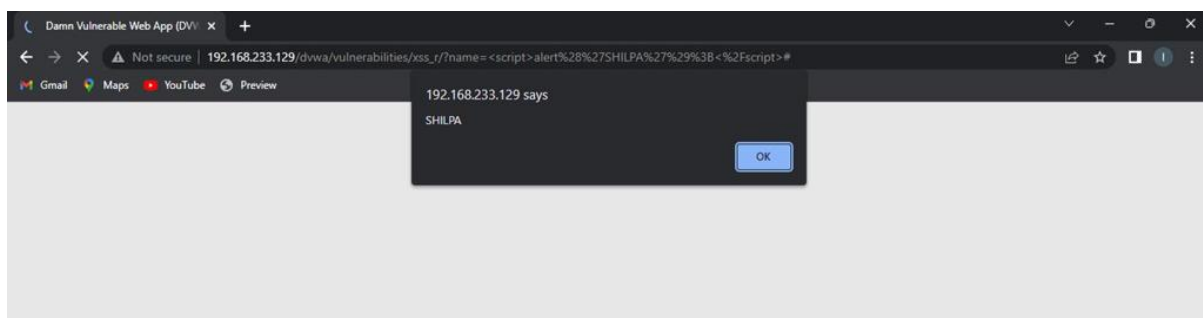
4. Cross-Site Scripting:

Cross-site scripting (XSS) is a type of web security vulnerability where an attacker is able to inject malicious code, usually in the form of scripts, into a web page viewed by other users. This can allow the attacker to steal sensitive information, such as login credentials or personal data, or to modify the content of the page in a way that can harm users.

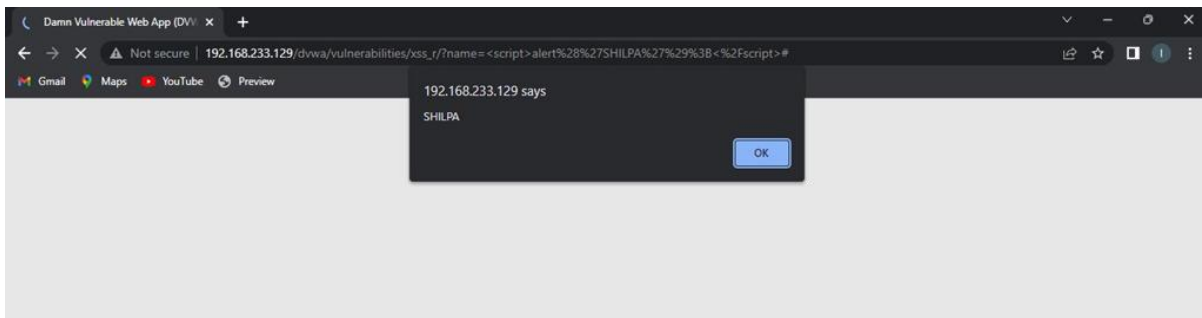
XSS attacks can occur when a web application does not properly validate or sanitize user input, such as in the case of comment boxes or search bars. An attacker can inject malicious code, such as a script that steals cookies or submits a form, into the web page by entering it as user input. This code is then executed by the browser of other users viewing the web page

i. Security level: low

XSS-reflected:

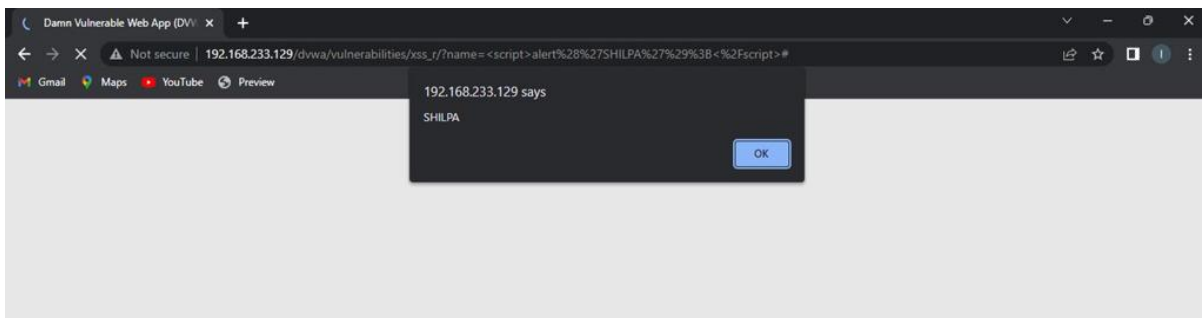


XSS-Stored:

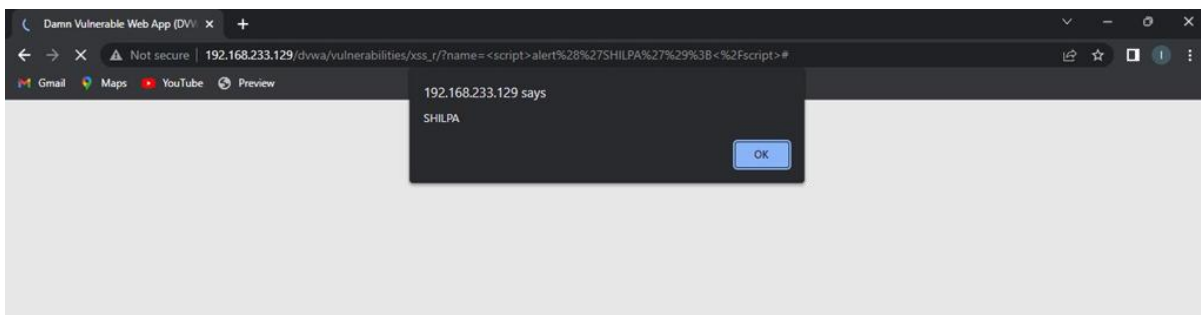


ii.Security level: medium

XSS-reflected:

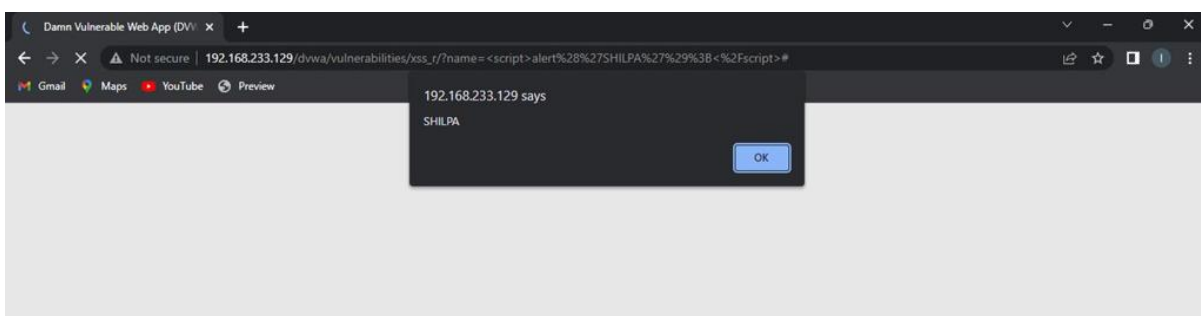


XSS-Stored:

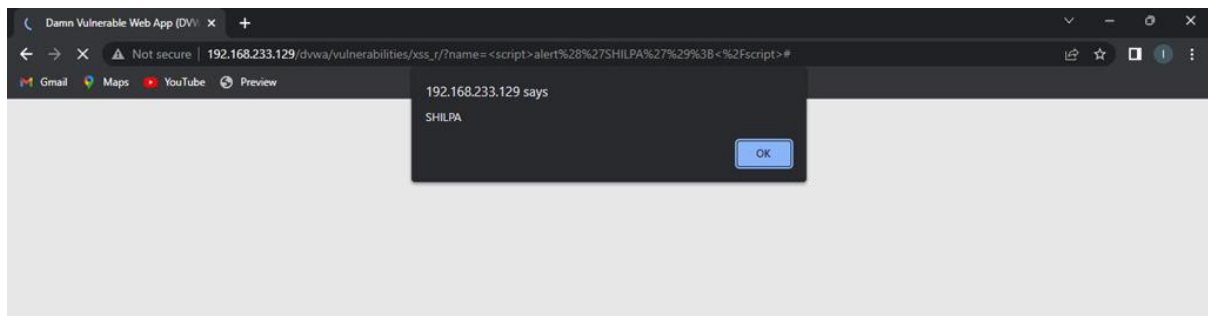


iii. Security level: high

XSS-reflected:



XSS-Stored:

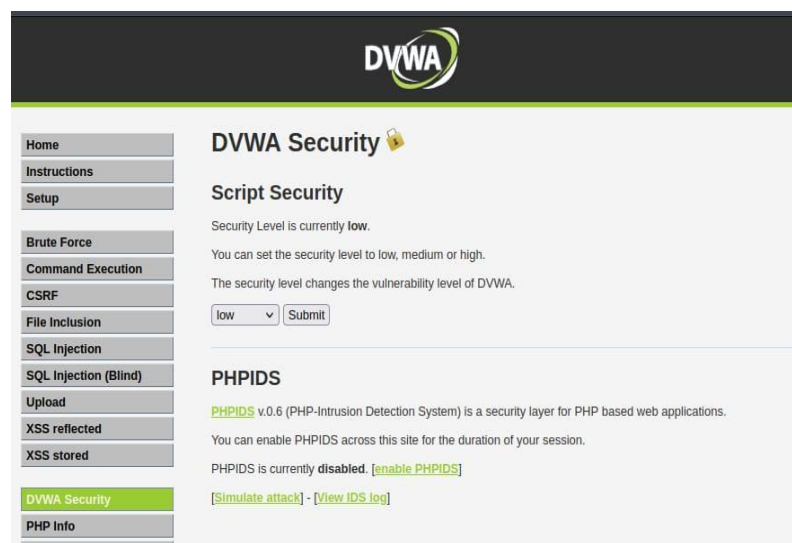
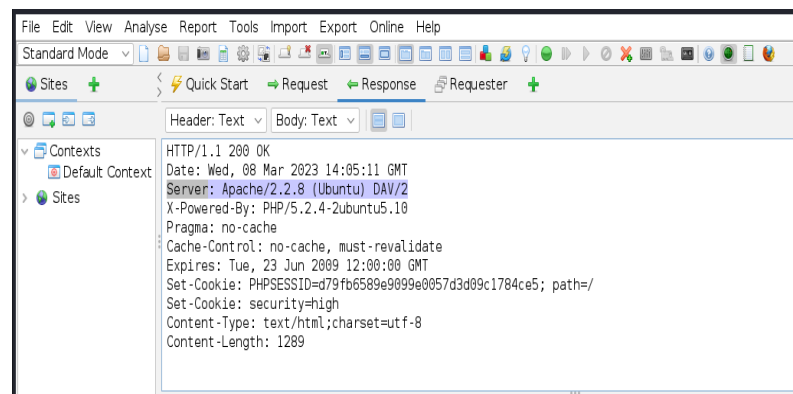


5.Sensitive Information Disclosure:

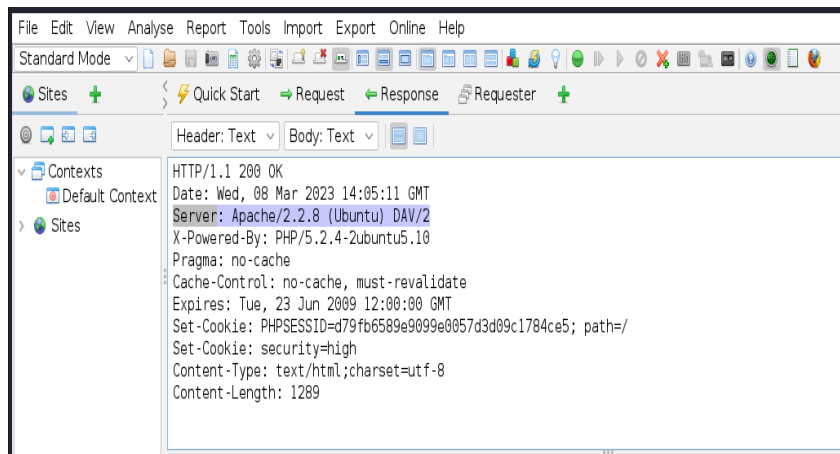
Information disclosure, also known as information leakage, is when a website unintentionally reveals sensitive information to its users. Depending on the context, websites may leak all kinds of information to a potential attacker.

Sensitive information disclosure is a type of security flaw that occurs when sensitive data, such as personal information or confidential business data, is exposed to unauthorized users or parties. This vulnerability can be exploited by attackers to compromise the confidentiality, integrity, and availability of the sensitive data.

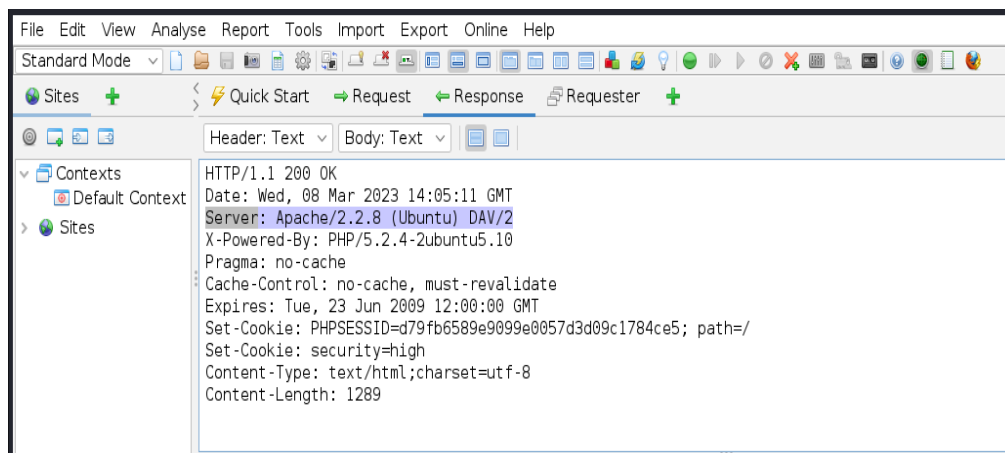
i.Security level: low

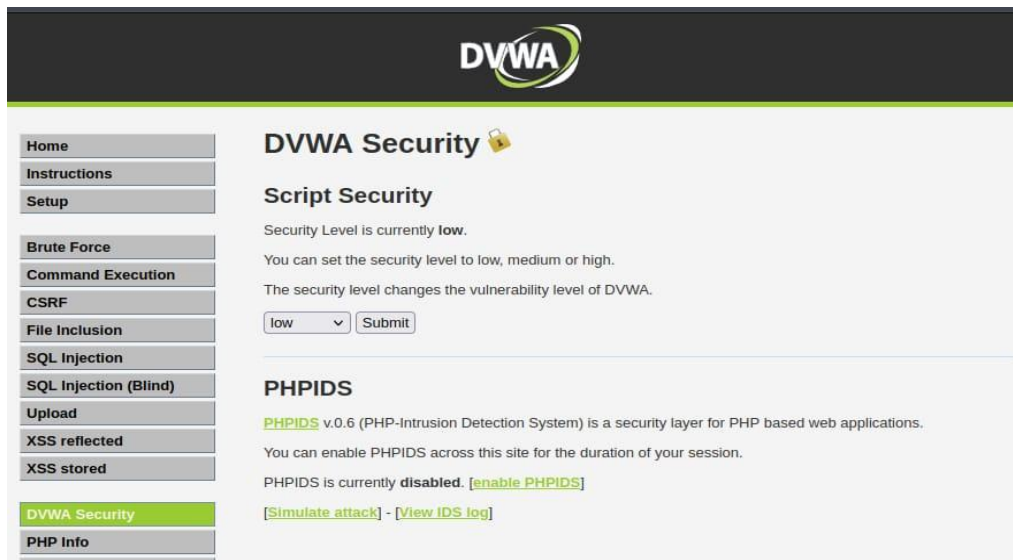


ii. Security level: medium



iii. Security level: high





6. Local File Inclusion:

A File Inclusion Vulnerability is a type of Vulnerability commonly found in PHP based websites and it is used to affect the web applications. This issue generally occurs when an application is trying to get some information from a particular server where the inputs for getting a particular file location are not treated as a trusted source.

It generally refers to an inclusion attack where an attacker can supply a valid input to get a response from a web server. In response, an attacker will be able to judge whether the input which he supplied is valid or not. If it is valid, then whatever/whichever file an attacker wants to see they can easily access it.

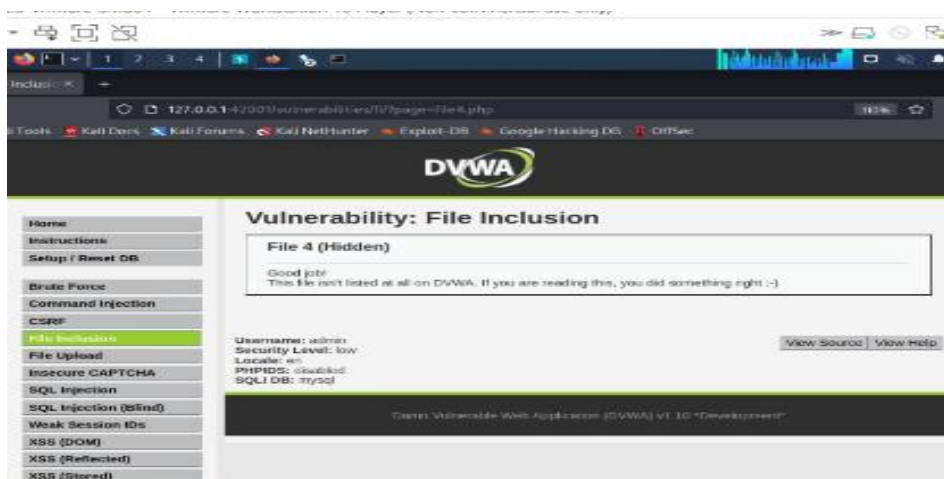
i. Security level: low



ii. Security level: medium



iii.Security level: high

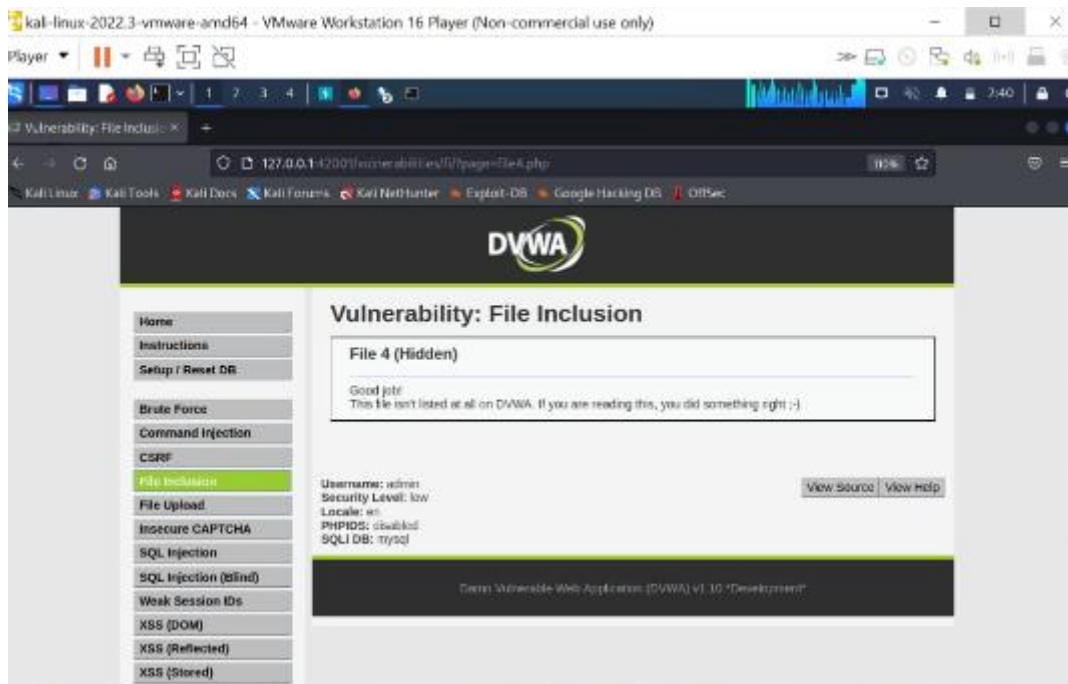


7.Remote File Inclusion:

Remote File Inclusion (RFI) is a type of security vulnerability that occurs when an application allows an attacker to include and execute remote files on the server. The vulnerability typically arises from insufficient input validation or poor coding practices in the application.

RFI typically occurs when a web application allows user input to control the path or URL of a file that is included or executed on the server. An attacker can exploit this vulnerability by manipulating the input to point to a remote file that they control, which can be used to inject malicious code onto the server.

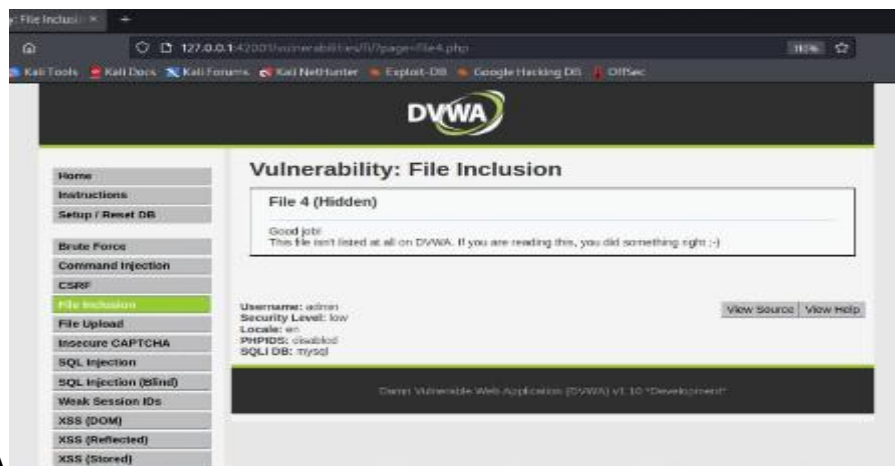
i.Security level: low



ii. Security level: medium



iii. Security level: high

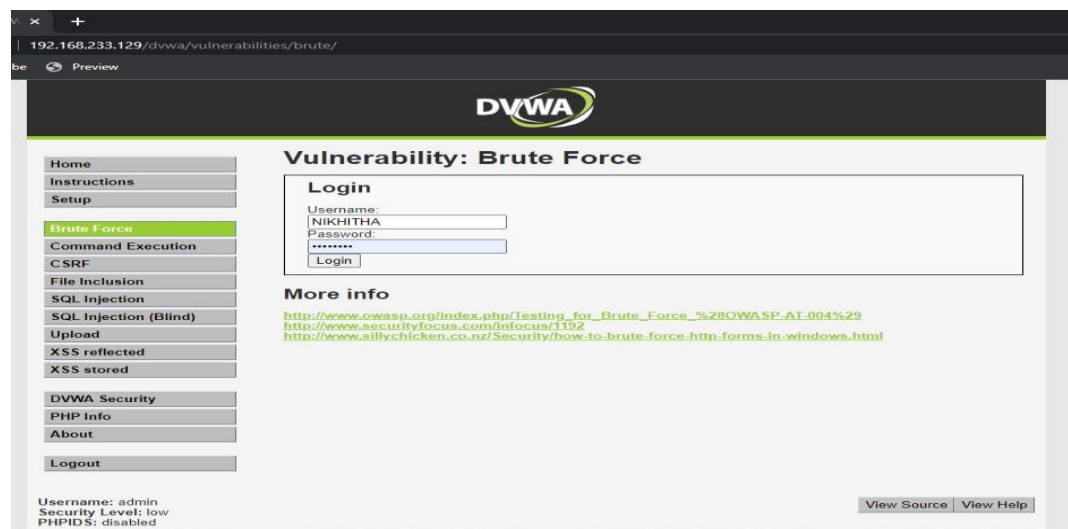
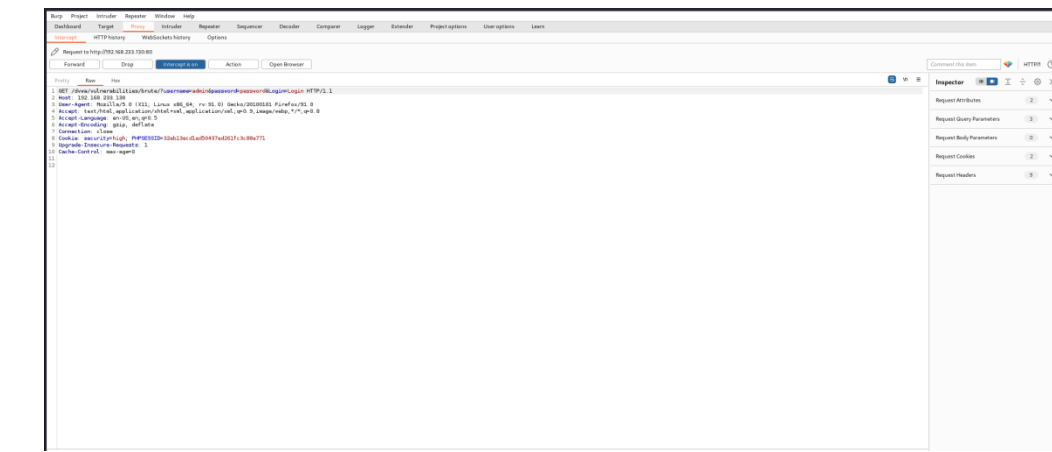


8.Bruteforce Attack:

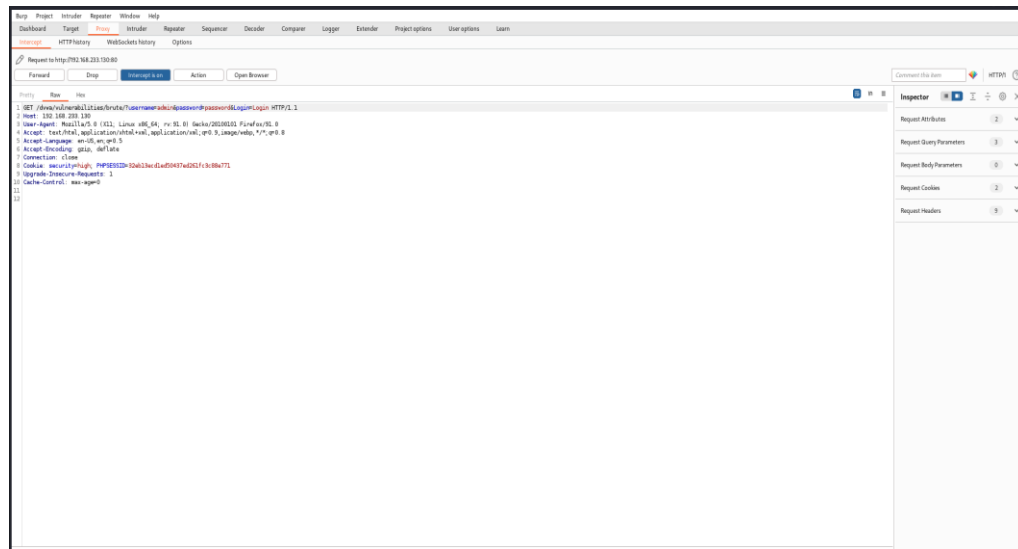
A brute force attack is a type of cyber attack in which an attacker tries to guess a password or encryption key by systematically trying every possible combination until the correct one is found. This type of attack can be used to gain unauthorized access to a system, steal sensitive data, or launch further attacks.

Brute force attacks are typically automated and use specialized software or scripts to generate and try large numbers of password or key combinations. The attack can take a long time, depending on the length and complexity of the password or key being targeted, and the computing power available .

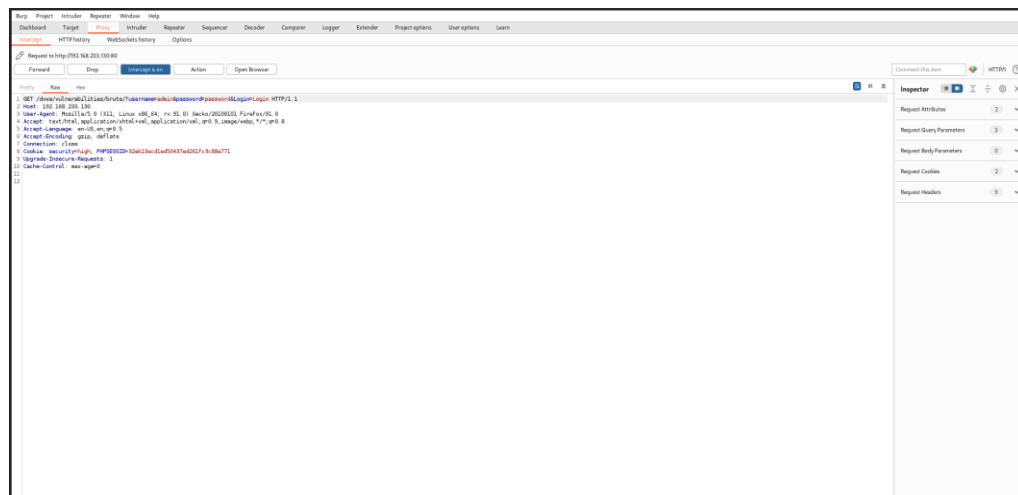
i.Security level: low



ii.Security level:medium



iii.Security level:high





9.forced browsing vulnerability:

Forced browsing, also known as directory traversal, is a type of security vulnerability that occurs when an attacker is able to access files and directories on a web server that are not intended to be publicly accessible. This vulnerability can be exploited by attackers to gain unauthorized access to sensitive information or to launch further attacks.

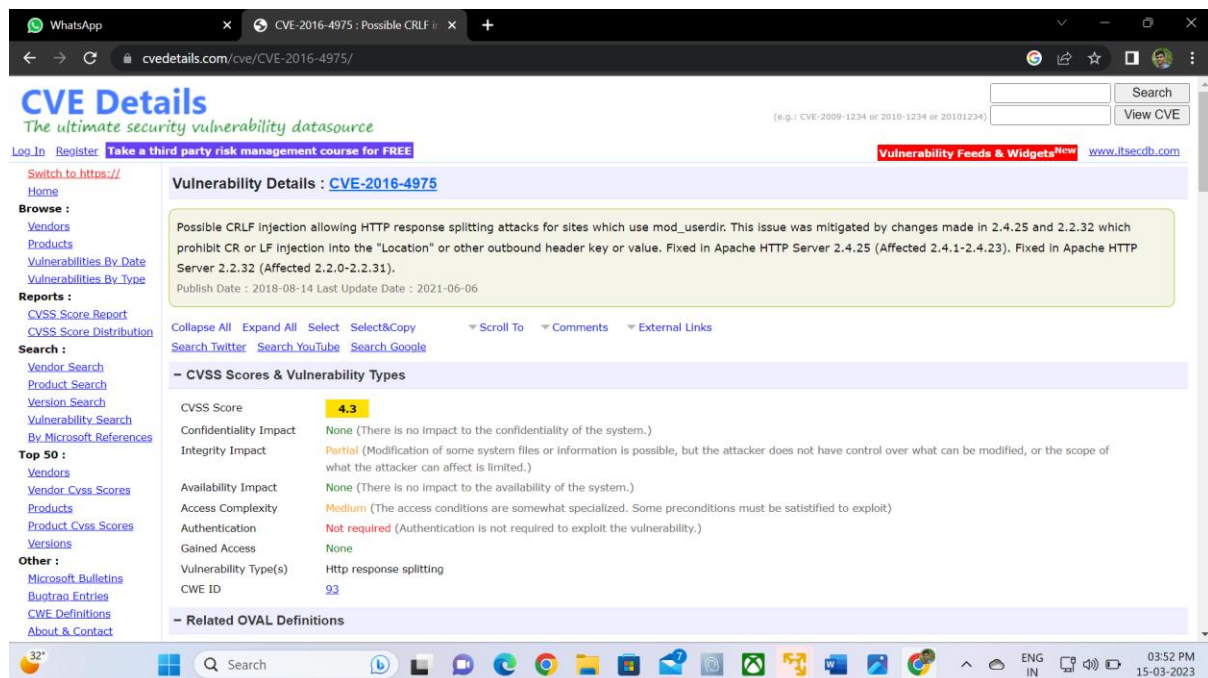
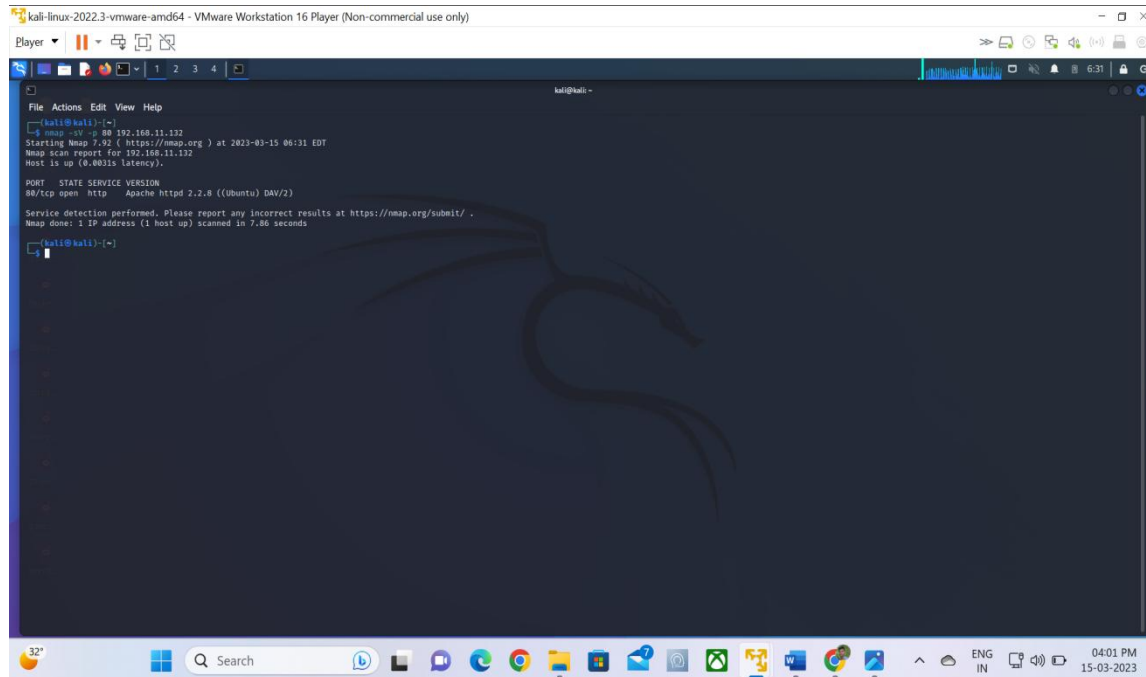
To mitigate forced browsing vulnerabilities, it is important to implement proper access controls and to validate user input. Web servers should be configured to restrict access to sensitive files and directories and to use secure file permissions. Input validation and sanitization can help prevent attacks that attempt to modify URLs or inject malicious code. Web application firewalls can also be used to detect and block forced browsing attacks.

10.components with known vulnerability:

Using Components with Known Vulnerabilities According to OWASP: Using Components with Known Vulnerabilities Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate severe data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine the app.

These attacks have become commonplace because it is far easier for an attacker to use a known weakness than create a specific program or attack methodology to search out vulnerabilities themselves. This fact should put known component vulnerabilities high on your

security priority list to mitigate.



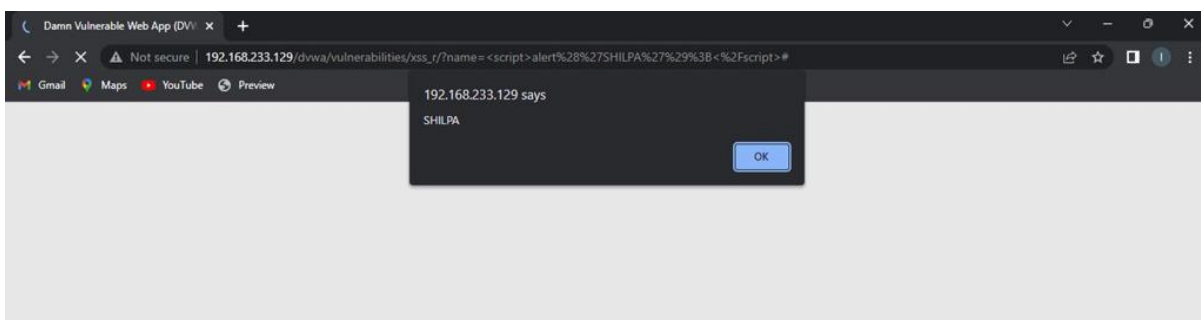
11.HTML injection:

HTML injection vulnerabilities typically arise from insufficient input validation or sanitization in a web application. Attackers can inject malicious code into input fields, such as login forms, comment sections, or search fields, and the code is then displayed to other users who visit the page. When the code is executed in the browser of the victim, it can steal cookies, session tokens, or other sensitive information, or redirect the victim to a malicious website.

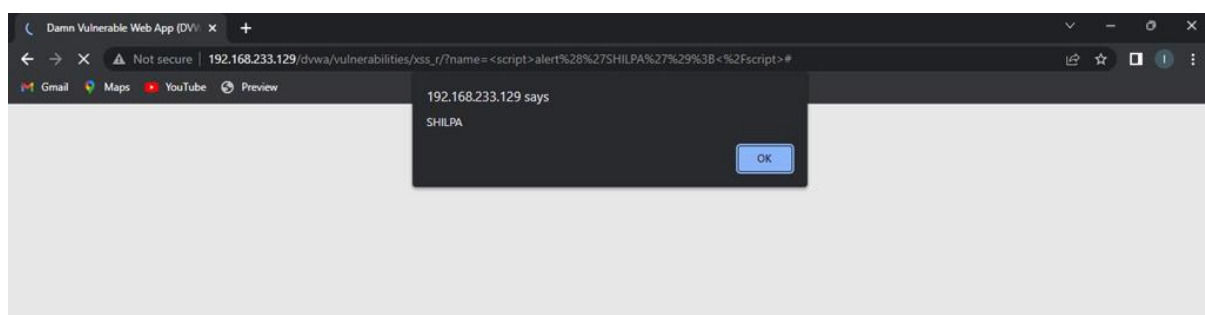
Developers should also avoid using unsanitized user input in HTML output, and use frameworks that provide built-in security features, such as template engines that automatically sanitize user input. It is also important to provide security awareness training to users to help them identify and avoid phishing attacks and other forms of social engineering.

i.Security level: Low:

Reflected-HTML

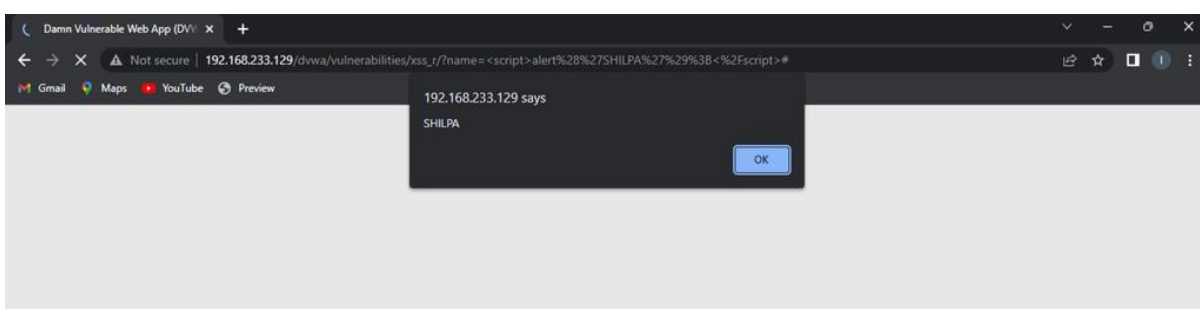


Stored-HTML:

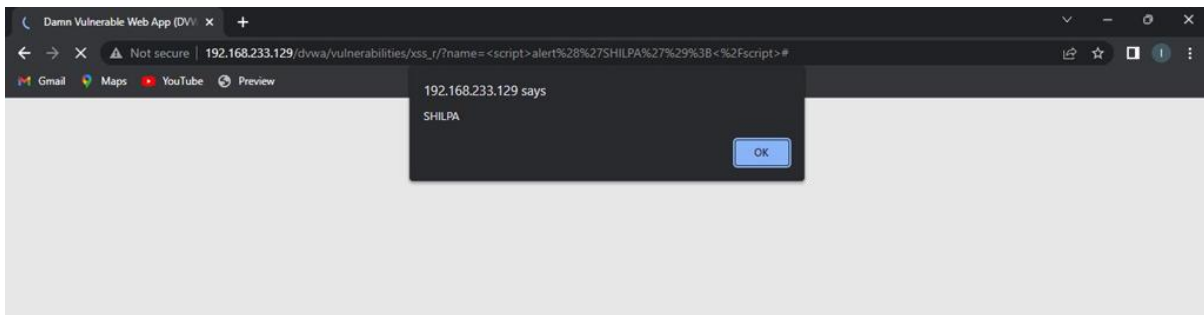


ii. Security level: Medium:

Reflected-HTML

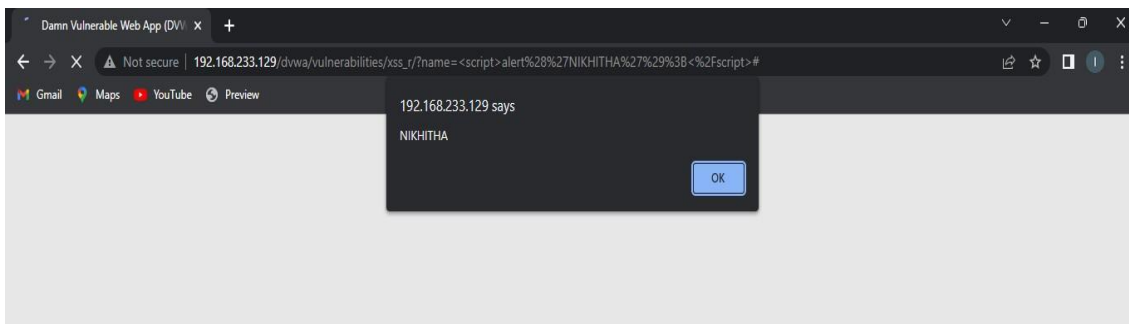


Stored-HTML:



iii. Security level: High

Reflected-HTML



Stored-HTML:

