

Name:SHREYAS HEGDE

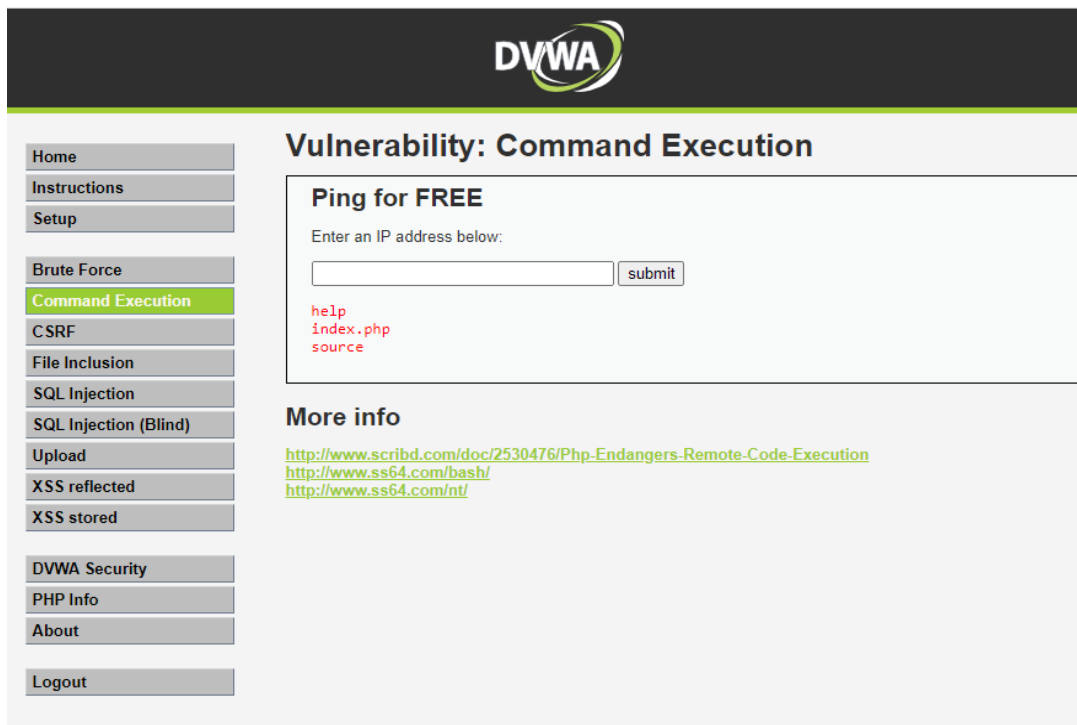
Date:13.03.2023

Task: 3

1.commands execution vulnerability:

A command execution vulnerability, also known as a command injection vulnerability, is a type of security vulnerability that occurs when an attacker is able to execute unauthorized commands on a target system or application. This vulnerability arises when an application allows user-supplied input to be executed as a command by the operating system or application without proper validation or sanitization.

Low



The screenshot shows the DVWA web application interface. At the top, there is a dark header with the DVWA logo. Below the header, a sidebar on the left contains a list of navigation links: Home, Instructions, Setup, Brute Force, Command Execution (highlighted in green), 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 section 'Ping for FREE' with a text input field and a 'submit' button. Below the input field, the text 'help', 'index.php', and 'source' is displayed in red. Underneath, there is a 'More info' section with three green hyperlinks: <http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>, <http://www.ss64.com/bash/>, and <http://www.ss64.com/nt/>.

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: Command Execution

Ping for FREE

Enter an IP address below:

```
help
index.php
source
```

More info

<http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
<http://www.ss64.com/bash/>
<http://www.ss64.com/nt/>

High



[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: Command Execution

Ping for FREE

Enter an IP address below:

```
help
index.php
source
```

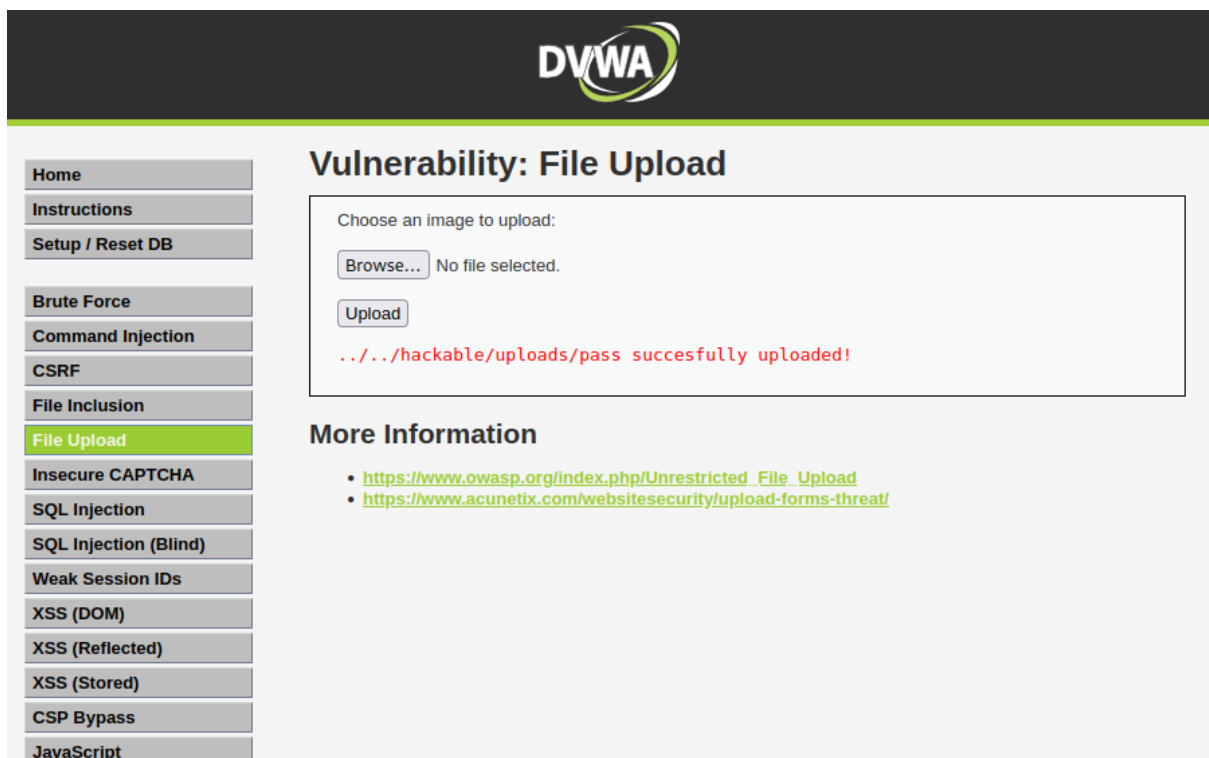
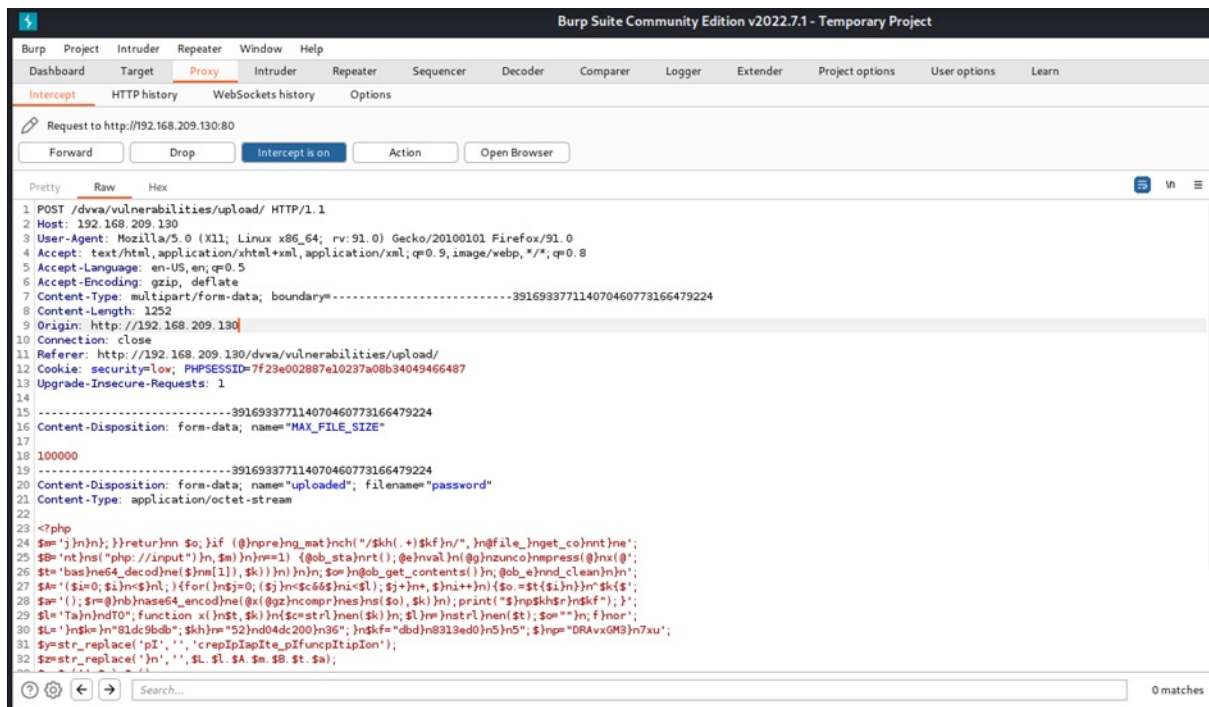
More info

<http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
<http://www.ss64.com/bash/>
<http://www.ss64.com/nt/>

2.file upload vulnerability:

A file upload vulnerability is a type of security vulnerability that allows an attacker to upload and execute malicious files on a target system or application. This vulnerability occurs when an application or website allows users to upload files without proper validation or sanitization of the uploaded content.

Low



Medium

Burp Suite Community Edition v2022.7.1 - Temporary Project

Dashboard Target **Proxy** Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn

Intercept HTTP history WebSockets history Options

Request to http://192.168.209.130:80

Forward Drop **Intercept is on** Action Open Browser

Pretty Raw Hex

```
1 POST /dvwa/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=-----391693377114070460773166479224
8 Content-Length: 1252
9 Origin: http://192.168.209.130
10 Connection: close
11 Referer: http://192.168.209.130/dvwa/vulnerabilities/upload/
12 Cookie: security=low; PHPSESSID=7f23e002887e10237a08b34049466487
13 Upgrade-Insecure-Requests: 1
14
15 -----391693377114070460773166479224
16 Content-Disposition: form-data; name="MAX_FILE_SIZE"
17
18 100000
19 -----391693377114070460773166479224
20 Content-Disposition: form-data; name="uploaded"; filename="password"
21 Content-Type: application/octet-stream
22
23 <?php
24 $a='';
25 $b='nt';
26 $t='bas';
27 $a='';
28 $a='';
29 $t='';
30 $t='';
31 $t='';
32 $t='';
```

0 matches

DVWA

Vulnerability: File Upload

Choose an image to upload:

No file selected.

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

More Information

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

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

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

JavaScript

Vulnerability: File Upload

Choose an image to upload:

Browse...

No file selected.

Upload

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

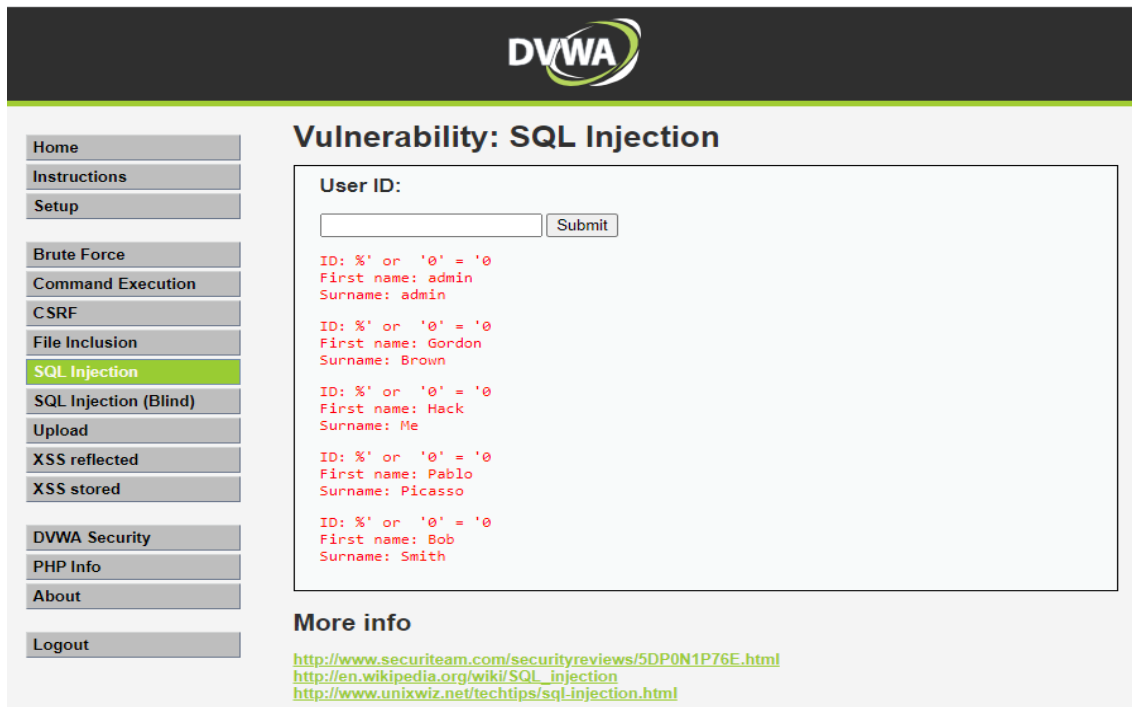
More Information

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

3.sql injection vulnerability:

SQL injection is a type of security vulnerability that occurs when an attacker is able to manipulate or inject malicious SQL code into an application's input fields, such as a login form or search box. This can allow the attacker to bypass authentication, execute unauthorized commands, or gain access to sensitive data stored in the application's database.

Low



The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The top header features the DVWA logo. On the left is a navigation menu with buttons for Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (highlighted in green), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: SQL Injection". It contains a "User ID:" label, an input field, and a "Submit" button. Below the input field, the application displays the results of the query in red text, showing the first five users: admin, Gordon Brown, Hack Me, Pablo Picasso, and Bob Smith. Each result line starts with "ID: '% or '0' = '0'". At the bottom, under "More info", there are three links: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, http://en.wikipedia.org/wiki/SQL_injection, and <http://www.unixwiz.net/techtips/sql-injection.html>.

DVWA

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>

Medium



[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 Security](#)
[PHP Info](#)
[About](#)

[Logout](#)

Vulnerability: SQL Injection


User ID:

ID: 3
First name: Hack
Surname: Me

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/>

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](#)
[JavaScript](#)

[DVWA Security](#)
[PHP Info](#)
[About](#)

[Logout](#)

Vulnerability: SQL Injection

Click [here to change your ID](#).

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

More Information

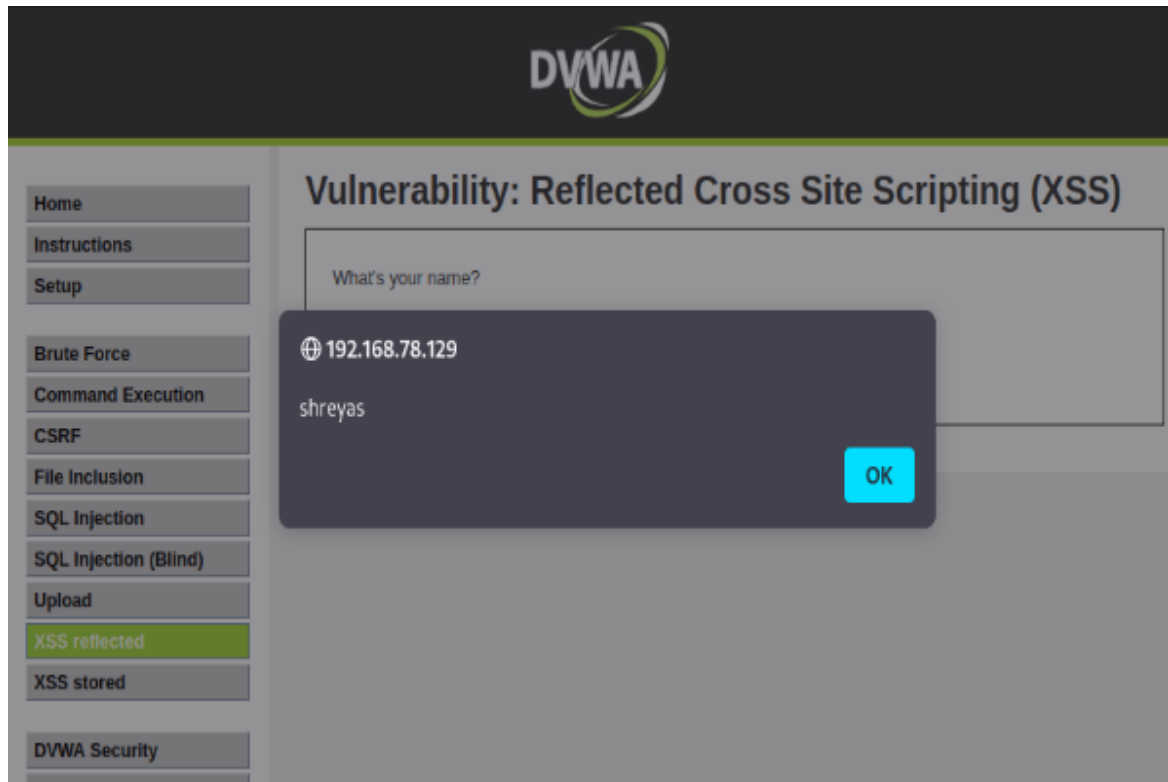
- https://en.wikipedia.org/wiki/SQL_injection
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- https://owasp.org/www-community/attacks/SQL_injection
- <https://bobby-tables.com/>

4.cross-site scripting:


Cross-site scripting (XSS) is a type of security vulnerability that allows attackers to inject malicious scripts into web pages viewed by other users. This occurs when an application does not properly validate or sanitize user input, allowing an attacker to inject malicious code into a web page that is then executed by a victim's web browser.

Xss-reflected:

Low



Medium



[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 Security](#)
[PHP Info](#)
[About](#)

[Logout](#)

Vulnerability: Reflected Cross Site Scripting (XSS)


What's your name?

Hello alert('SHREYAS');

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/>

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](#)
[JavaScript](#)

[DVWA Security](#)
[PHP Info](#)
[About](#)

[Logout](#)

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

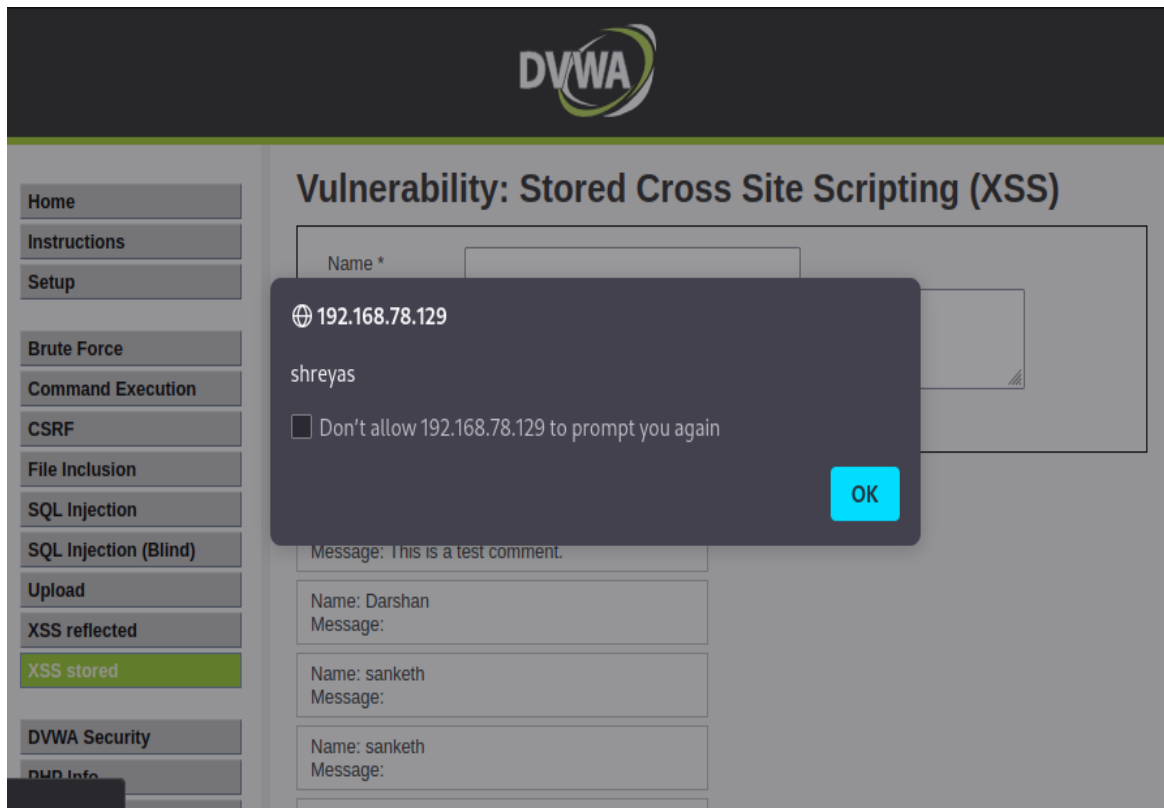
Hello >

More Information

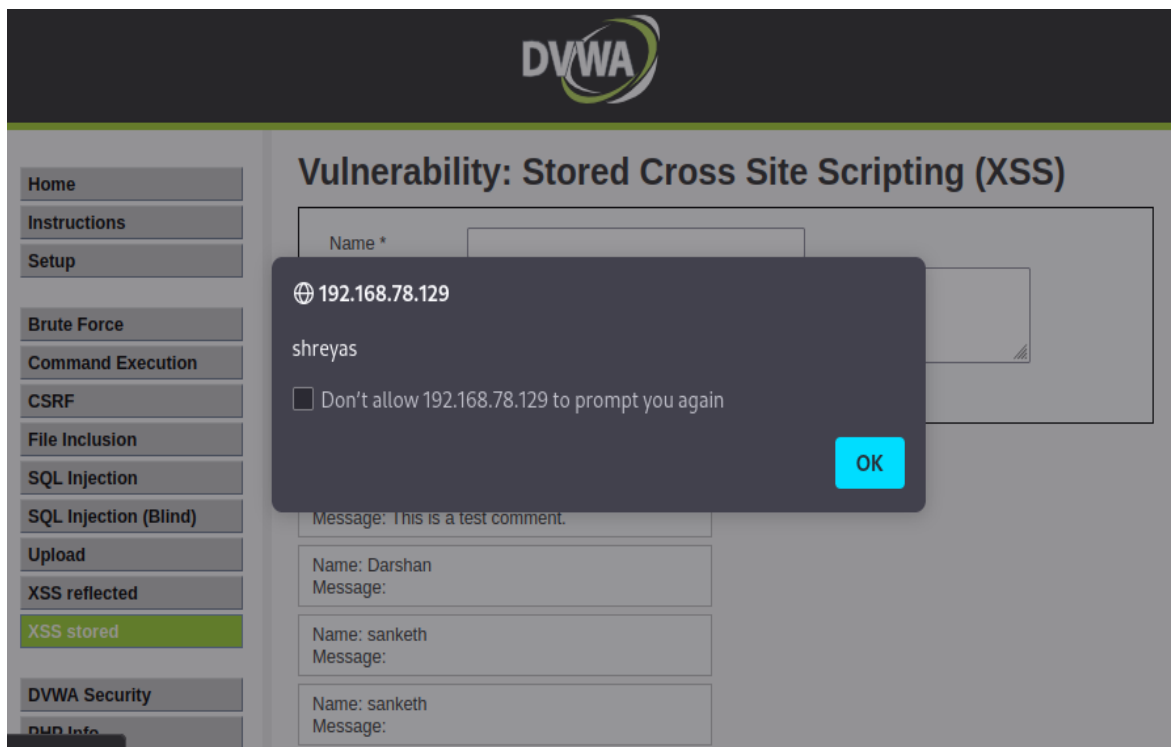
- <https://owasp.org/www-community/attacks/xss/>
- <https://owasp.org/www-community/xss-filter-evasion-cheatsheet>
- https://en.wikipedia.org/wiki/Cross-site_scripting
- <http://www.cgisecurity.com/xss-faq.html>
- <http://www.scriptalert1.com/>

Xss-Stored:

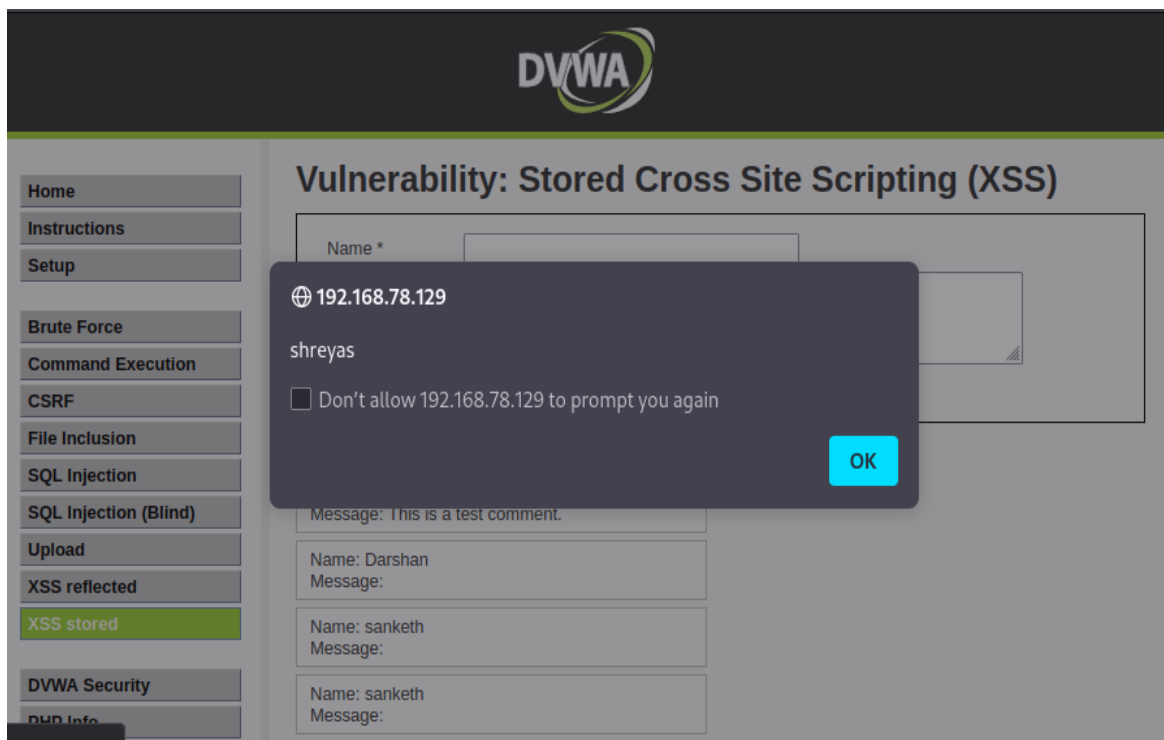
Low



Medium



High



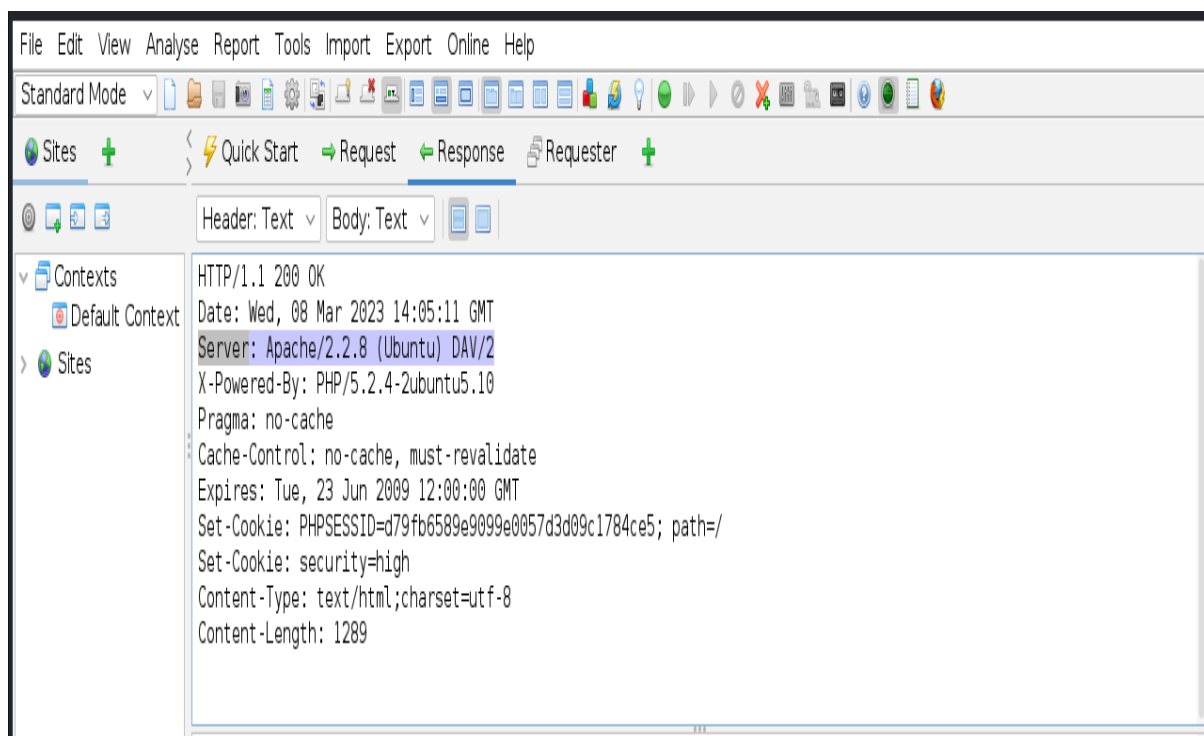
5.sensitive information disclosure:

Sensitive information disclosure is a type of security vulnerability that occurs when an application or system reveals sensitive information to unauthorized users. This can include personal information, such as names, addresses, social security numbers, or financial information, as well as system information, such as server logs, database credentials, or other configuration details.

Low



The screenshot shows the DVWA Security page. On the left is a sidebar with navigation links: 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 Security (highlighted), PHP Info, About, and Logout. The main content area is titled 'DVWA Security' and 'Security Level'. It states 'Security level is currently: low.' and explains that the security level can be set to low, medium, high, or impossible. A list of four levels is provided: 1. Low (completely vulnerable, no security measures), 2. Medium (basic exploitation techniques), 3. High (extension to medium difficulty, harder or alternative bad practices), and 4. Impossible (secure against all vulnerabilities). A dropdown menu is set to 'Low' with a 'Submit' button. Below this is the 'PHPIDS' section, which describes it as a security layer for PHP based web applications. It states 'PHPIDS v0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.' and 'PHPIDS works by filtering any user supplied input against a blacklist of potentially malicious code.' It also mentions 'You can enable PHPIDS across this site for the duration of your session.' and shows 'PHPIDS is currently: disabled.' with links to 'Enable PHPIDS', 'Simulate attack', and 'View IDS log'.



The screenshot shows a web browser window with the title 'File Edit View Analyse Report Tools Import Export Online Help'. The address bar shows 'Standard Mode'. The browser is displaying the 'Request' tab. The 'Request' tab shows the following headers: HTTP/1.1 200 OK, Date: Wed, 08 Mar 2023 14:05:11 GMT, Server: Apache/2.2.8 (Ubuntu) DAV/2, X-Powered-By: PHP/5.2.4-2ubuntu5.10, Pragma: no-cache, Cache-Control: no-cache, must-revalidate, Expires: Tue, 23 Jun 2009 12:00:00 GMT, Set-Cookie: PHPSESSID=d79fb6589e9099e0057d3d09c1784ce5; path=/, Set-Cookie: security=high, Content-Type: text/html; charset=utf-8, Content-Length: 1289.

Medium



The screenshot shows the DVWA Security page. The security level is set to 'Medium'. The page includes a sidebar with navigation links and a main content area with a 'Security Level' section and a 'PHPIDS' section.

DVWA Security

Security Level

Security level is currently: **low**.

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

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code. Prior to DVWA v1.9, this level was known as 'high'.

Medium Submit

PHPIDS

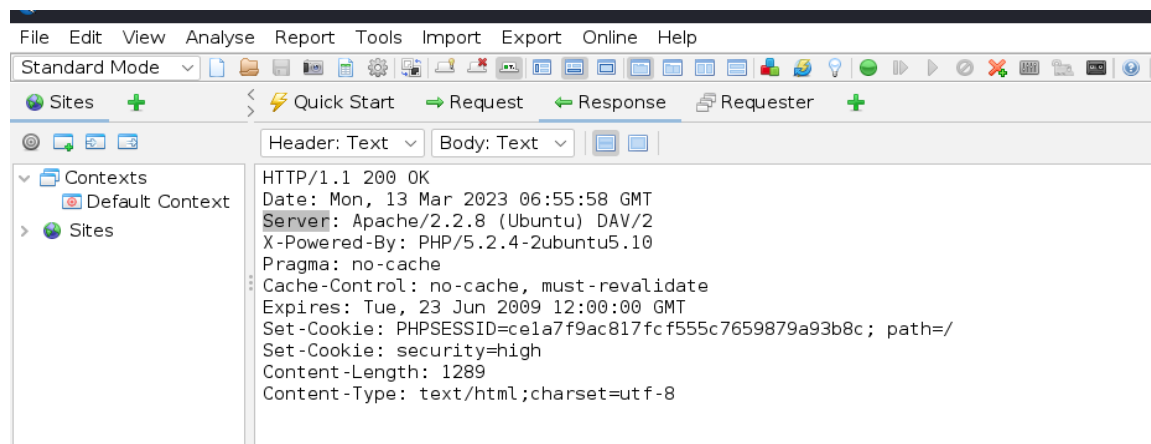
PHPIDS v0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

PHPIDS works by filtering any user supplied input against a blacklist of potentially malicious code. It is used in DVWA to serve as a live example of how Web Application Firewalls (WAFs) can help improve security and in some cases how WAFs can be circumvented.

You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently: **disabled**. [Enable PHPIDS]

[Simulate attack] - [View IDS log]



The screenshot shows a web browser window displaying the response of the DVWA Security page. The response is an HTTP 200 OK status with various headers and cookies.

File Edit View Analyse Report Tools Import Export Online Help

Standard Mode

Sites Quick Start Request Response Requester

Header: Text Body: Text

Contexts Default Context Sites

HTTP/1.1 200 OK
Date: Mon, 13 Mar 2023 06:55:58 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Pragma: no-cache
Cache-Control: no-cache, must-revalidate
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Set-Cookie: PHPSESSID=ce1a7f9ac817fcf555c7659879a93b8c; path=/
Set-Cookie: security=high
Content-Length: 1289
Content-Type: text/html; charset=utf-8

High



The screenshot shows the DVWA Security page. The security level is set to 'High'. The page includes a sidebar with navigation links and a main content area with a 'Security Level' section and a 'PHPIDS' section.

DVWA Security

Security Level

Security level is currently: **low**.

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

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code. Prior to DVWA v1.9, this level was known as 'high'.

High Submit

PHPIDS

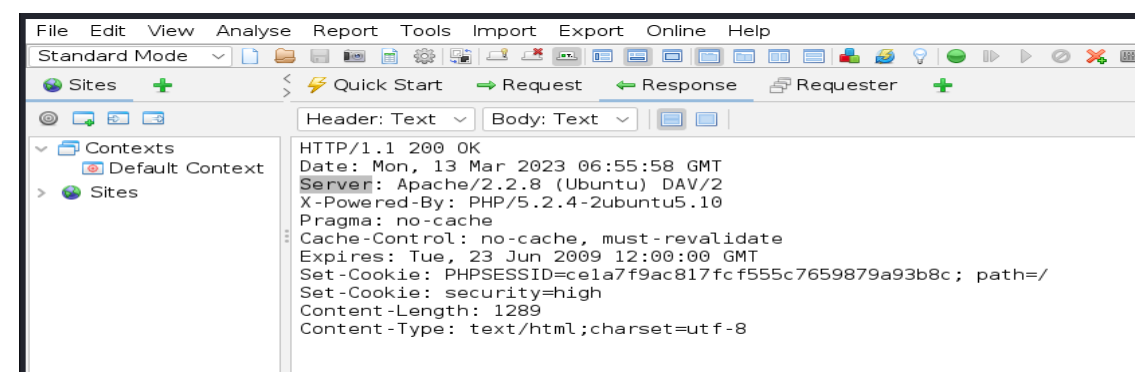
PHPIDS v0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

PHPIDS works by filtering any user supplied input against a blacklist of potentially malicious code. It is used in DVWA to serve as a live example of how Web Application Firewalls (WAFs) can help improve security and in some cases how WAFs can be circumvented.

You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently: **disabled**. [Enable PHPIDS]

[Simulate attack] - [View IDS log]



The screenshot shows a web browser window displaying the response of the DVWA Security page. The response is an HTTP 200 OK status with various headers and cookies.

File Edit View Analyse Report Tools Import Export Online Help

Standard Mode

Sites Quick Start Request Response Requester

Header: Text Body: Text


Contexts Default Context Sites

HTTP/1.1 200 OK
Date: Mon, 13 Mar 2023 06:55:58 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Pragma: no-cache
Cache-Control: no-cache, must-revalidate
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Set-Cookie: PHPSESSID=ce1a7f9ac817fcf555c7659879a93b8c; path=/
Set-Cookie: security=high
Content-Length: 1289
Content-Type: text/html; charset=utf-8

6.local file inclusion:

Local File Inclusion (LFI) is a type of security vulnerability that occurs when an application allows a user to include a local file in a web page without proper validation or sanitization. This can allow an attacker to access files on the server that are not intended to be accessible, including configuration files, system logs, or sensitive user data.

Low



[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 Security](#)
[PHP Info](#)
[About](#)

[Logout](#)

Vulnerability: File Inclusion

File 4 (Hidden)

Good job!
This file isn't listed at all on DVWA. If you are reading this, you did something right ;-)


Username: admin
Security Level: low
Locale: en
PHPIDS: disabled
SQLi DB: mysql

[View Source](#) [View Help](#)

Damn Vulnerable Web Application (DVWA) v1.10 *Development*

Medium

[Exploit-DB](#) [Google Hacking DB](#) [OnSec](#)



[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 Security](#)
[PHP Info](#)
[About](#)

[Logout](#)

Vulnerability: File Inclusion

File 4 (Hidden)


Good job!
This file isn't listed at all on DVWA. If you are reading this, you did something right :-)

Username: admin
Security Level: medium
Locale: en
PHPIDS: disabled
SQLi DB: mysql

[View Source](#) [View Help](#)

Damn Vulnerable Web Application (DVWA) v1.10 *Development*

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](#)
[JavaScript](#)

[DVWA Security](#)
[PHP Info](#)
[About](#)

[Logout](#)

Vulnerability: File Inclusion

File 4 (Hidden)

Good job!
This file isn't listed at all on DVWA. If you are reading this, you did something right :-)

Username: admin
Security Level: high
Locale: en
PHPIDS: disabled
SQLi DB: mysql


[View Source](#) [View Help](#)

Damn Vulnerable Web Application (DVWA) v1.10 *Development*

7.remote file inclusion:

Remote File Inclusion (RFI) is a type of security vulnerability that occurs when an application allows a user to include a remote file in a web page without proper validation or sanitization. This can allow an attacker to execute malicious code on the server or to access files on a remote server that are not intended to be accessible.

Low



[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 Security](#)
[PHP Info](#)
[About](#)

[Logout](#)

Vulnerability: File Inclusion

File 4 (Hidden)

Good job!
This file isn't listed at all on DVWA. If you are reading this, you did something right ;-)


Username: admin
Security Level: low
Locale: en
PHPIDS: disabled
SQLi DB: mysql

[View Source](#) [View Help](#)

Damn Vulnerable Web Application (DVWA) v1.10 *Development*

Medium

Exploit-DBGoogle Hacking DBOnSec



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 Security

PHP Info

About

Logout

Vulnerability: File Inclusion

File 4 (Hidden)


Good job!
This file isn't listed at all on DVWA. If you are reading this, you did something right ;-)

Username: admin
Security Level: medium
Locale: en
PHPIDS: disabled
SQLI DB: mysql

[View Source](#)[View Help](#)

Damn Vulnerable Web Application (DVWA) v1.10 *Development*

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

JavaScript

DVWA Security

PHP info

About

Logout

Vulnerability: File Inclusion

File 4 (Hidden)

Good job!
This file isn't listed at all on DVWA. If you are reading this, you did something right ;-)

Username: admin
Security Level: high
Locale: en
PHPIDS: disabled
SQLI DB: mysql

[View Source](#)[View Help](#)

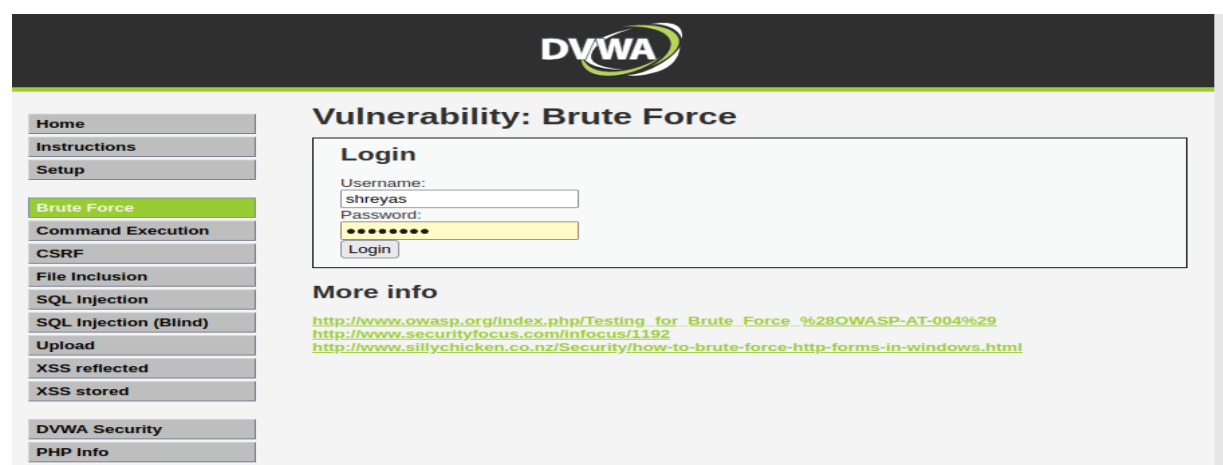
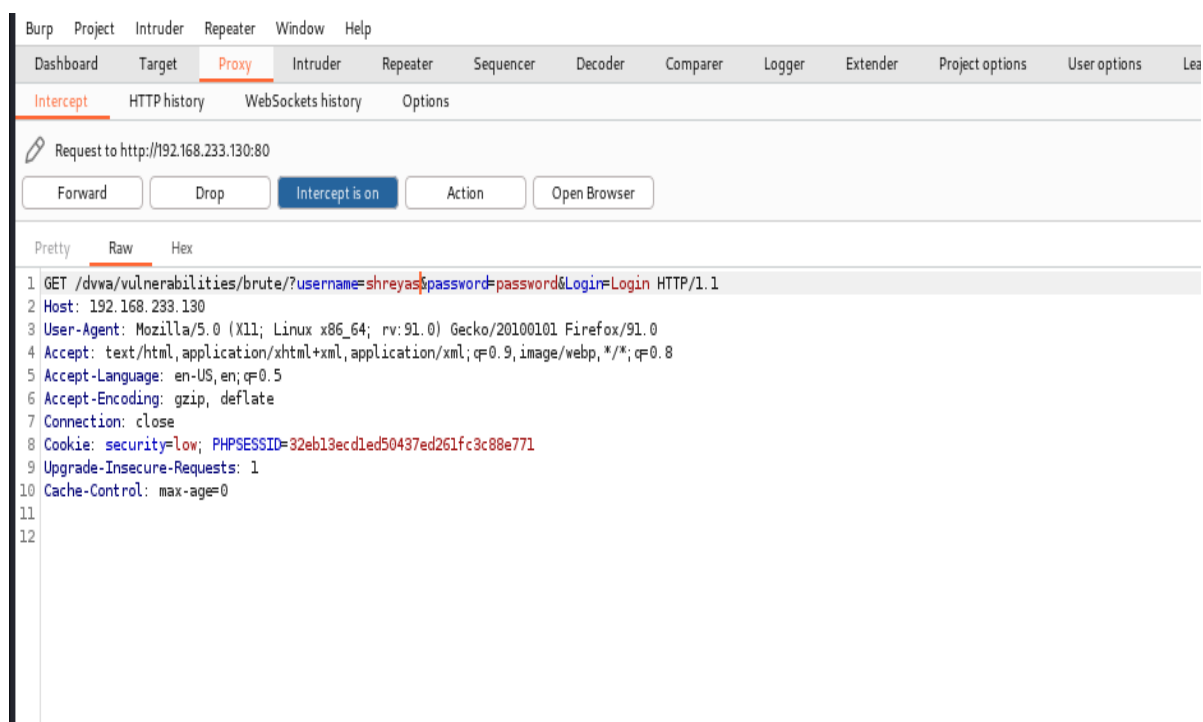
Damn Vulnerable Web Application (DVWA) v1.10 *Development*

8.bruteforce attack:

A brute force attack is a type of cyberattack that involves systematically trying every possible combination of characters to guess a password or encryption key. Attackers use specialized software or tools to automate the process of guessing passwords or keys, with the goal of gaining unauthorized access to a system or application.

Brute force attacks can be particularly effective against weak passwords or encryption keys, such as those that are short or use common words or patterns. Attackers can also use techniques such as dictionary attacks, which use a pre-generated list of common passwords to speed up the guessing process.

Low



Medium

Request to http://192.168.233.130:80

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex

```
1 GET /dvwa/vulnerabilities/brute/?username=shreyas&password=password&Login=Login HTTP/1.1
2 Host: 192.168.233.130
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 Connection: close
8 Cookie: security=medium; PHPSESSID=32eb13ecd1ed50437ed261fc3c88e771
9 Upgrade-Insecure-Requests: 1
10 Cache-Control: max-age=0
11
12
```

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

Vulnerability: Brute Force

Login

Username:

Password:

Login

More info

http://www.owasp.org/index.php/Testing_for_Brute_Force_%28OWASP-AT-004%29
<http://www.securityfocus.com/infocus/1192>
<http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html>

High

Request to http://192.168.233.130:80

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex

```
1 GET /dvwa/vulnerabilities/brute/?username=shreyas&password=password&Login=Login HTTP/1.1
2 Host: 192.168.233.130
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 Connection: close
8 Cookie: security=high; PHPSESSID=32eb13ecd1ed50437ed261fc3c88e771
9 Upgrade-Insecure-Requests: 1
10 Cache-Control: max-age=0
11
12
```

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

Vulnerability: Brute Force

Login

Username:

Password:

Login

More info

http://www.owasp.org/index.php/Testing_for_Brute_Force_%28OWASP-AT-004%29
<http://www.securityfocus.com/infocus/1192>
<http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html>

9.forced browsing vulnerability:

Forced browsing is a type of security vulnerability that occurs when an attacker is able to access unauthorized resources or data by manually or programmatically guessing or manipulating URLs or directory structures. This can allow an attacker to access sensitive information or functionality that is not intended to be accessible to the general public. Forced browsing attacks can occur when an application does not properly enforce access controls or input validation, allowing attackers to access files or resources that are not intended to be publicly accessible. This can include sensitive data such as user account information, financial records, or confidential business data.

To protect against forced browsing attacks, it is important to implement proper access controls and input validation to ensure that only authorized users are able to access sensitive resources or data. This can include using authentication and authorization mechanisms, restricting access to specific IP addresses or user agents, and using encryption and other security measures to protect sensitive data.

10.components with known vulnerability:

Components with known vulnerabilities refer to software libraries, frameworks, or other components that have known security vulnerabilities or weaknesses that can be exploited by attackers. These components are often used in the development of software applications, and may include third-party libraries, open source software, or other components that are commonly used by developers.

```
File Actions Edit View Help
(kali@kali)-[~]
$ nmap -sV -p 80 192.168.11.132
Starting Nmap 7.92 ( https://nmap.org ) at 2023-03-15 06:31 EDT
Nmap scan report for 192.168.11.132
Host is up (0.0031s latency).

PORT      STATE SERVICE
80/tcp    open  http
Apache httpd 2.2.8 ((Ubuntu) DAV/2)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.86 seconds

(kali@kali)-[~]
$
```

CVE Details

The ultimate security vulnerability datasource

[Log In](#) [Register](#) [Take a third party risk management course for FREE](#)

[Switch to https://](#)
[Home](#)

Browse :

- [Vendors](#)
- [Products](#)
- [Vulnerabilities By Date](#)
- [Vulnerabilities By Type](#)

Reports :

- [CVSS Score Report](#)
- [CVSS Score Distribution](#)

Search :

- [Vendor Search](#)
- [Product Search](#)
- [Version Search](#)
- [Vulnerability Search](#)
- [By Microsoft References](#)

Top 50 :

- [Vendors](#)
- [Vendor Cvss Scores](#)
- [Products](#)
- [Product Cvss Scores](#)
- [Versions](#)

Other :

- [Microsoft Bulletins](#)
- [Bugtraq Entries](#)
- [CVE Definitions](#)
- [About & Contact](#)

(e.g.: CVE-2009-1234 or 2010-1234 or 20101234)

Vulnerability Feeds & Widgets [www.itsecdb.com](#)

Vulnerability Details : CVE-2016-4975

Possible CRLF injection allowing HTTP response splitting attacks for sites which use mod_userdir. This issue was mitigated by changes made in 2.4.25 and 2.2.32 which prohibit CR or LF injection into the "Location" or other outbound header key or value. Fixed in Apache HTTP Server 2.4.25 (Affected 2.4.1-2.4.23). Fixed in Apache HTTP Server 2.2.32 (Affected 2.2.0-2.2.31).

Publish Date : 2018-08-14 Last Update Date : 2021-06-06

[Collapse All](#) [Expand All](#) [Select](#) [Select&Copy](#) [Scroll To](#) [Comments](#) [External Links](#)

[Search Twitter](#) [Search YouTube](#) [Search Google](#)

CVSS Scores & Vulnerability Types

CVSS Score	4.3
Confidentiality Impact	None (There is no impact to the confidentiality of the system.)
Integrity Impact	Partial (Modification of some system files or information is possible, but the attacker does not have control over what can be modified, or the scope of what the attacker can affect is limited.)
Availability Impact	None (There is no impact to the availability of the system.)
Access Complexity	Medium (The access conditions are somewhat specialized. Some preconditions must be satisfied to exploit)
Authentication	Not required (Authentication is not required to exploit the vulnerability.)
Gained Access	None
Vulnerability Type(s)	Http response splitting
CWE ID	93

Related OVAL Definitions

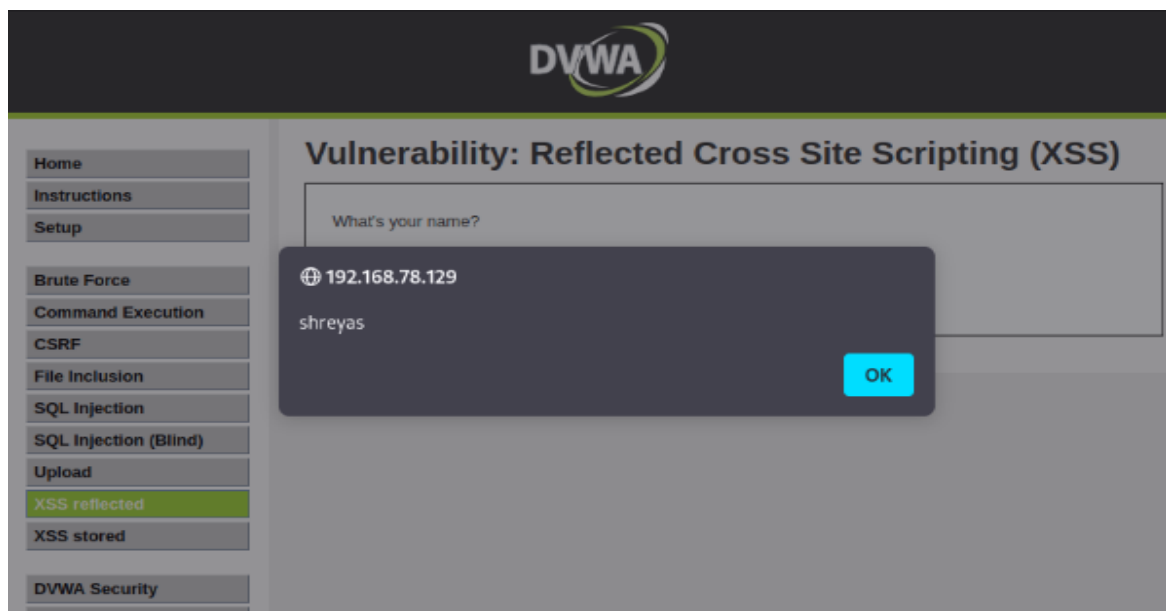
11.html injection:

HTML injection, also known as HTML injection attack or HTML code injection, is a type of web security vulnerability that allows an attacker to insert malicious HTML code into a web page. This code is then executed by the victim's web browser, potentially allowing the attacker to steal sensitive information or launch further attacks.

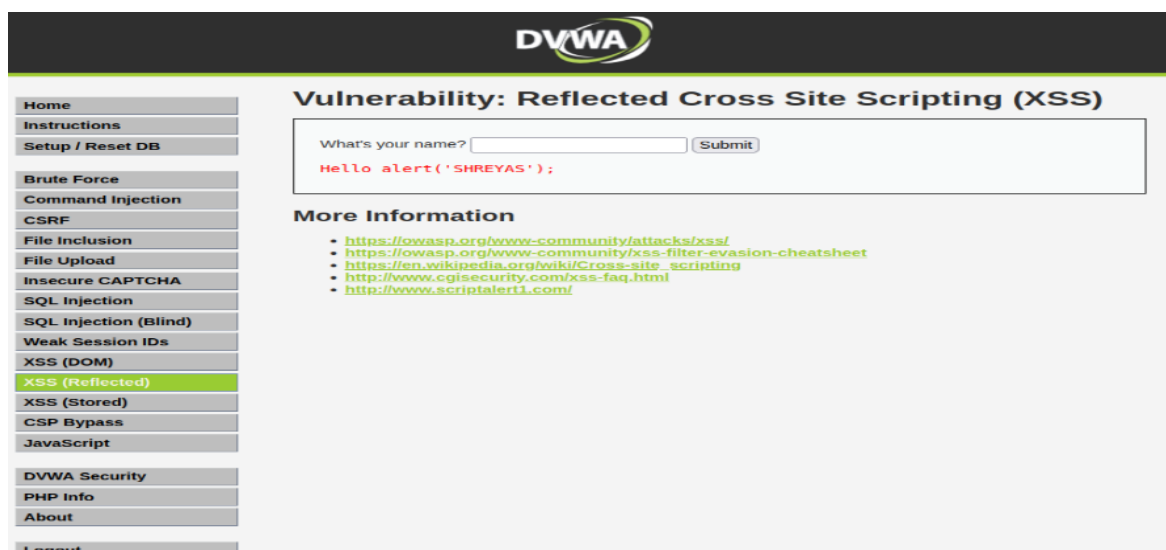
HTML injection attacks can occur when an application does not properly validate or sanitize user input, allowing an attacker to inject malicious HTML code into a web page that is viewed by other users. This can occur in a variety of ways, such as through input fields, cookies, or other mechanisms that allow users to input data.

Xss-reflected:

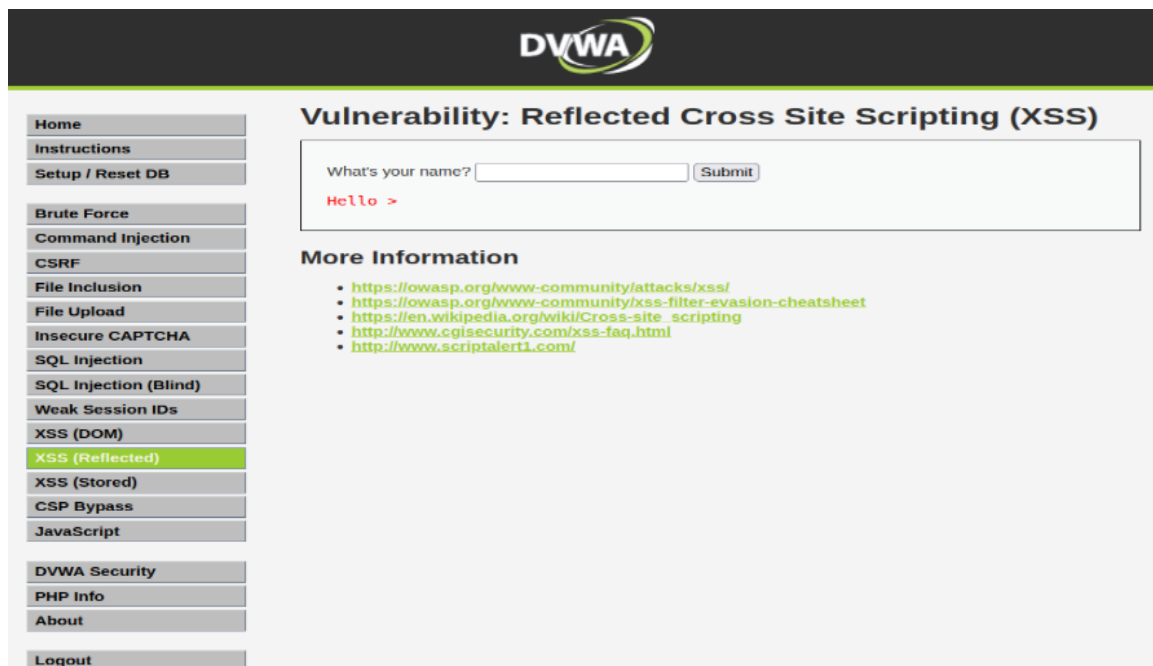
Low



Medium



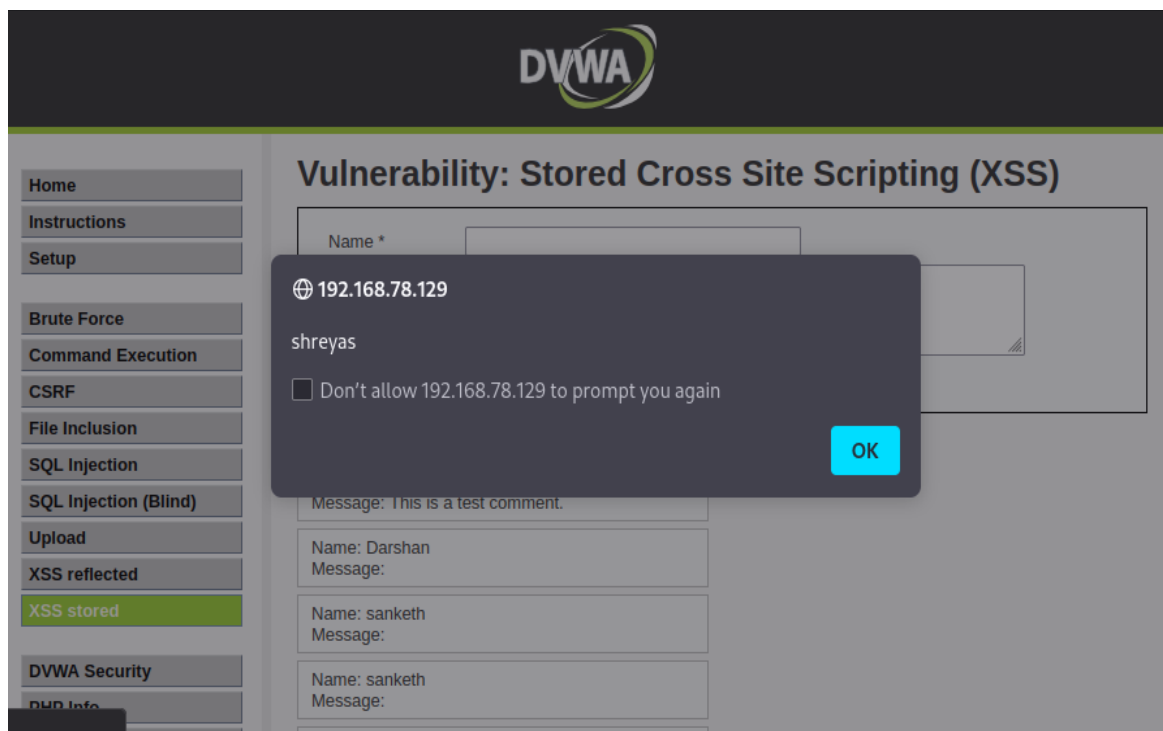
High



The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The top header features the DVWA logo. On the left, a sidebar contains a list of navigation links: 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) (highlighted in green), XSS (Stored), CSP Bypass, JavaScript, DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: Reflected Cross Site Scripting (XSS)". It contains a form with the label "What's your name?" and a "Submit" button. Below the form, the text "Hello >" is displayed in red. Under the heading "More Information", there is a list of links: <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>, and <http://www.scriptalert1.com/>.

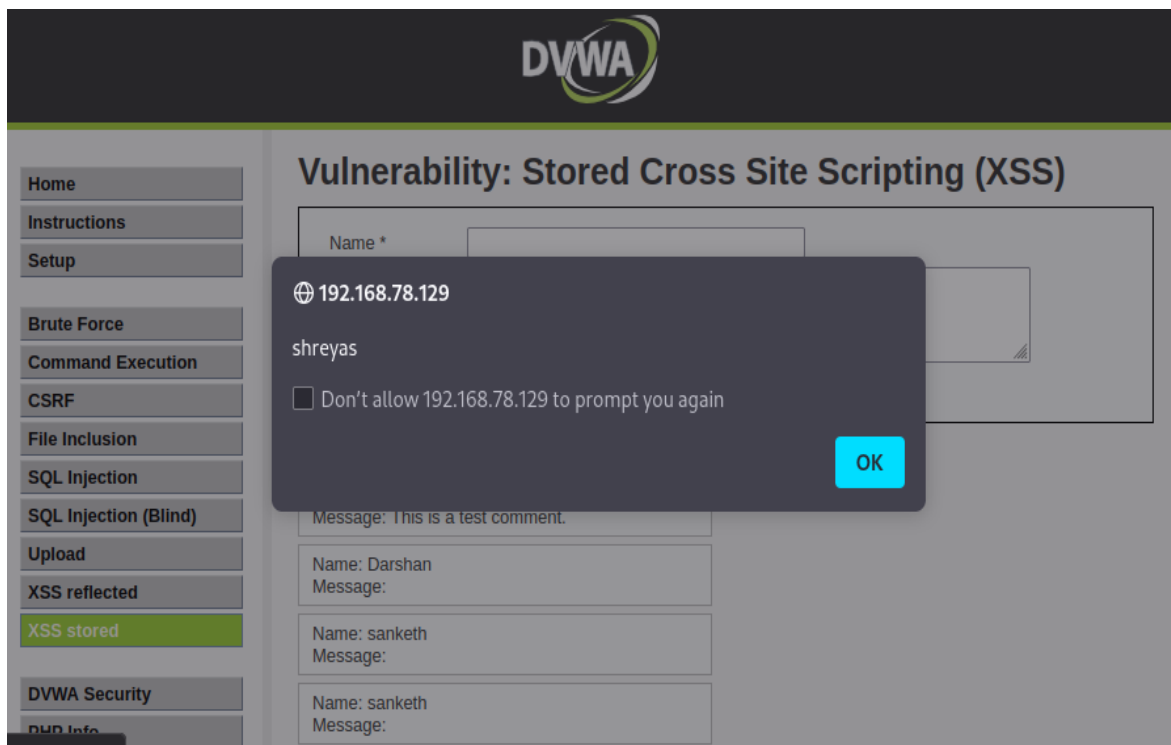
Xss-Stored:

Low



The screenshot shows the DVWA interface for the "Vulnerability: Stored Cross Site Scripting (XSS)" section. The sidebar on the left is similar to the first screenshot, but "XSS stored" is highlighted in green. The main content area shows a form with a "Name *" field. A modal dialog box is overlaid on the form, displaying a warning from the browser: "192.168.78.129 shreyas" and a checkbox labeled "Don't allow 192.168.78.129 to prompt you again". Below the dialog, the form shows a "Message: This is a test comment." and several input fields for "Name" and "Message". The "Name" field contains "Darshan" and "sanketh", and the "Message" field contains "sanketh".

Medium



High

