

Name:AMBIKA

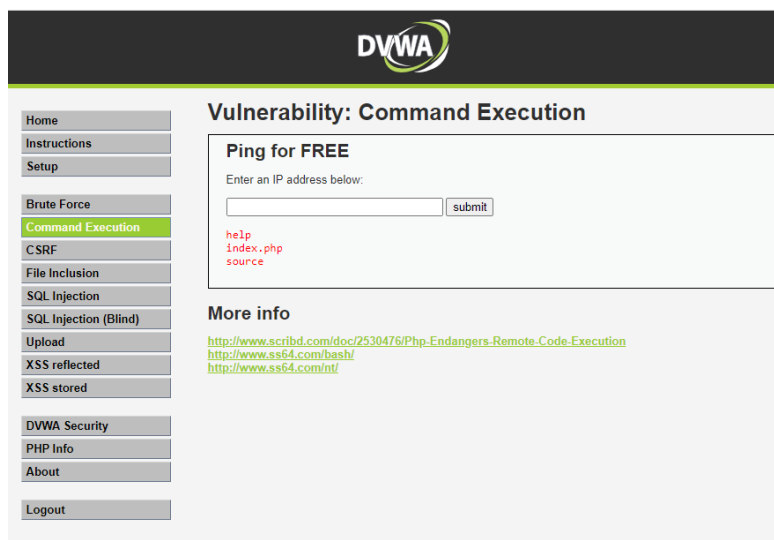
Date:13.03.2023

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.




The screenshot shows the DVWA web application interface. At the top is the DVWA logo. On the left is a sidebar menu with buttons for 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 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/>.

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.



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!

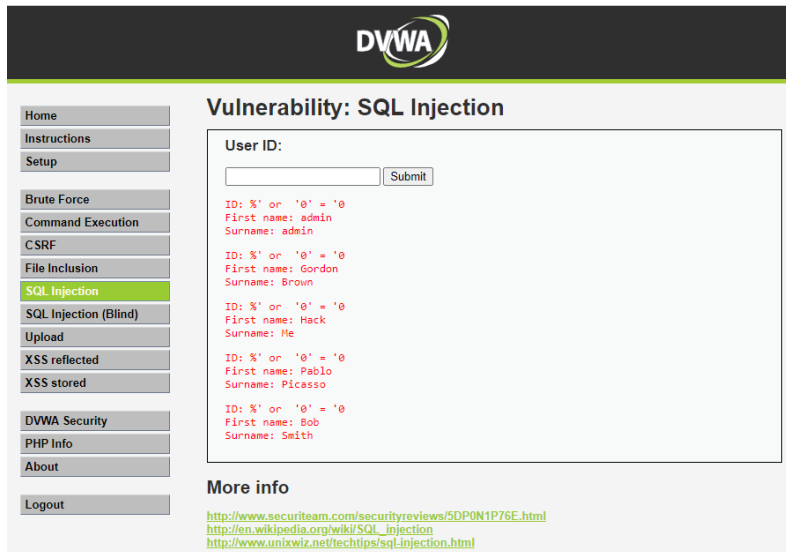
More Information

- https://www.owasp.org/index.php/Unrestricted_File_Upload
- <https://www.acunetix.com/websitesecurity/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.



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 links: 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, a text input field, and a "Submit" button. Below the input field, there are four lines of red text, each representing a successful SQL injection payload and the resulting user data:

```
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: He

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

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

Below the main content area, there is a "More info" section with three links:

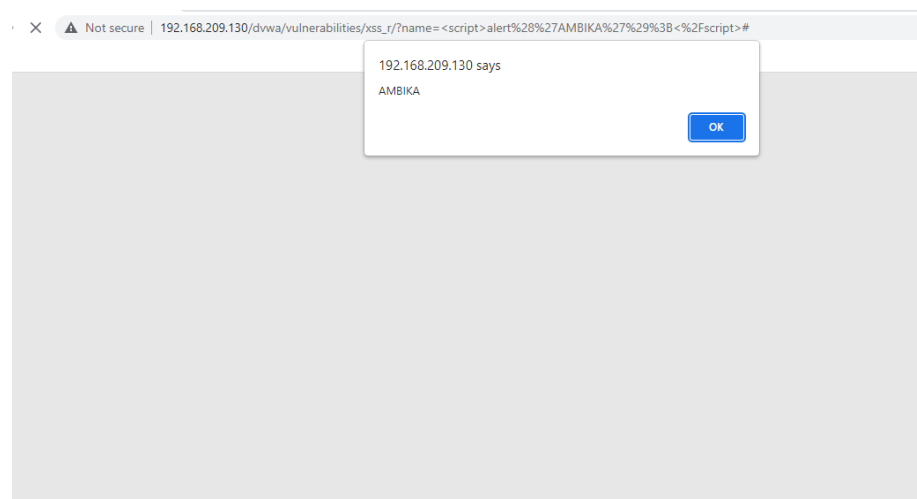
- <http://www.securiteam.com/securityreviews/SDP0N1P76E.html>
- http://en.wikipedia.org/wiki/SQL_injection
- <http://www.unixwiz.net/techtips/sql-injection.html>

4.cross-site scripting:

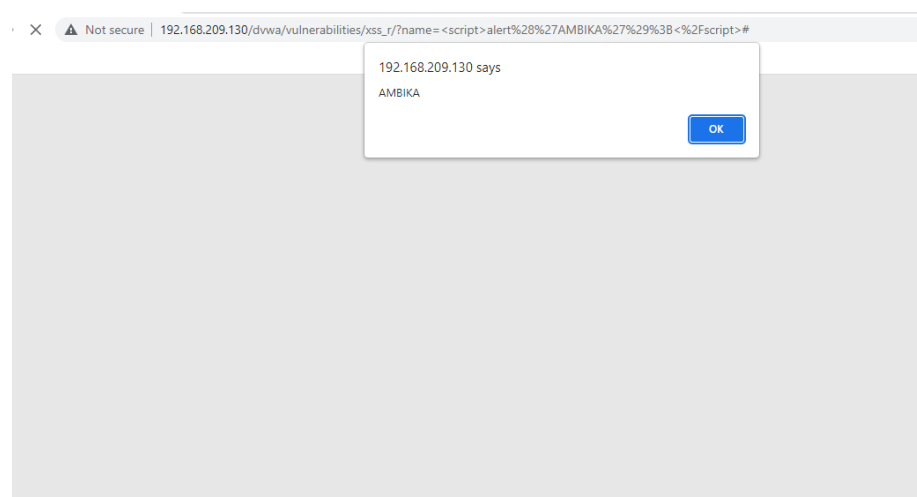
Cross-site scripting (XSS) is a type of web security vulnerability where an attacker is able to inject malicious code, usually in the form of scripts, into a web page viewed by other users. This can allow the attacker to steal sensitive information, such as login credentials or personal data, or to modify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

Xss-reflected:



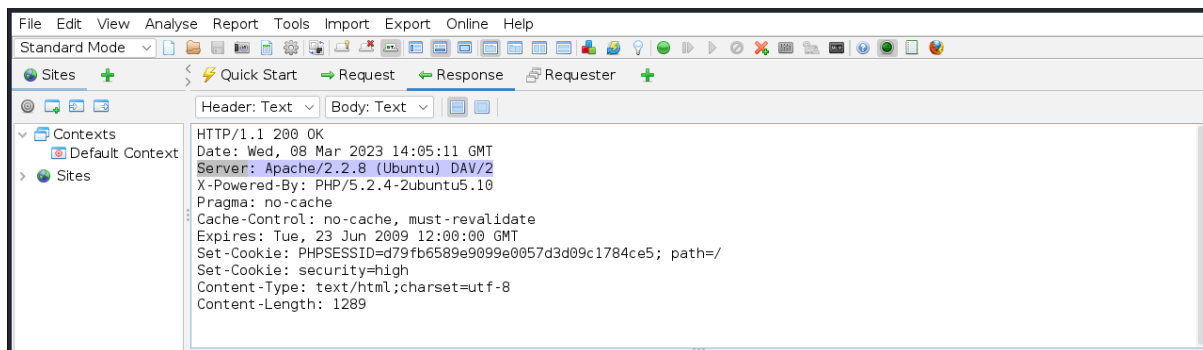
Xss-stored:



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:

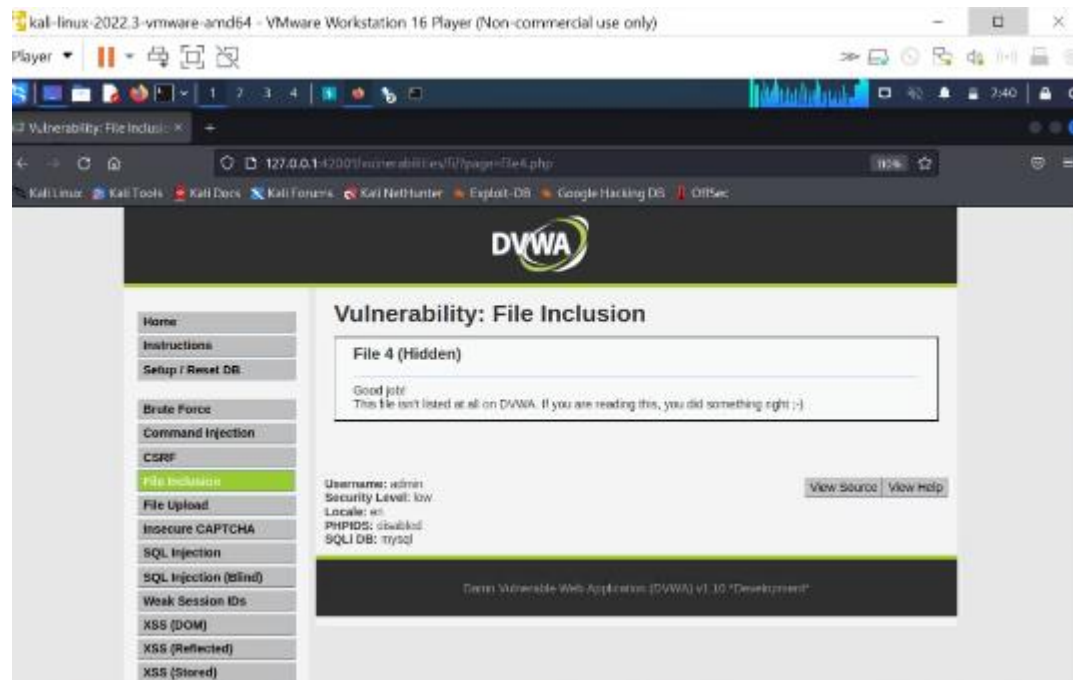
The dangers of leaking sensitive user or business data are fairly obvious, but disclosing technical information can sometimes be just as serious. Although some of this information will be of limited use, it can potentially be a starting point for exposing an additional attack surface, which may contain 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.



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.

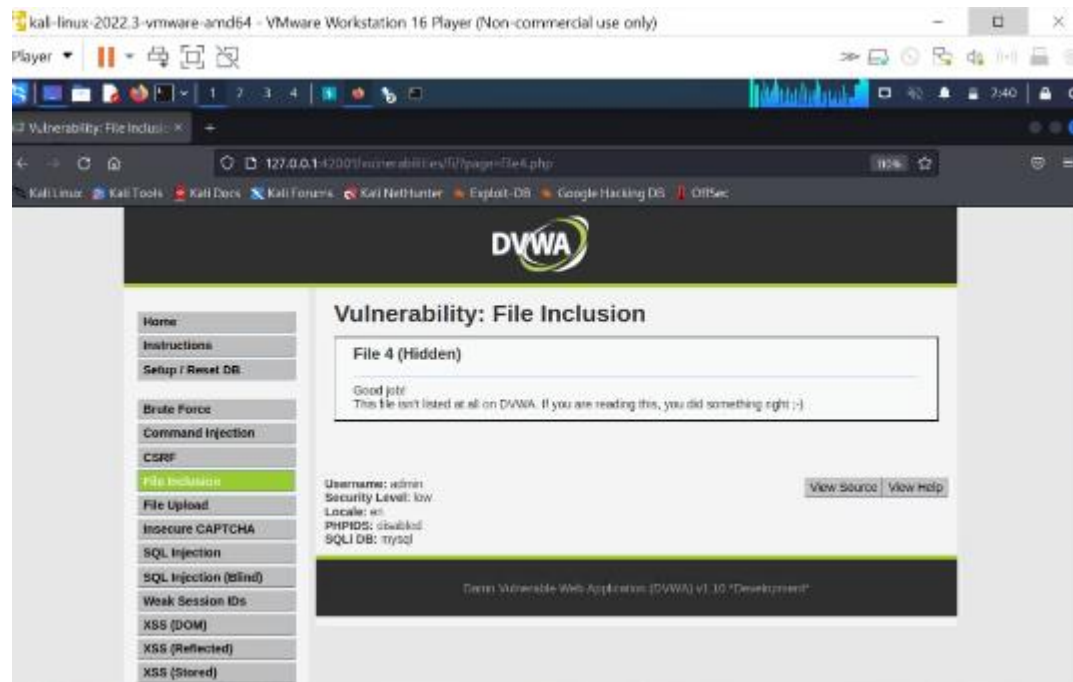
This type of attack is also known as Local File Inclusion (LFI) or Path Traversal. It can occur in web applications that accept user input and do not properly validate or sanitize it before using it in file operations. Attackers can exploit this vulnerability to access files on the server, including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



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.

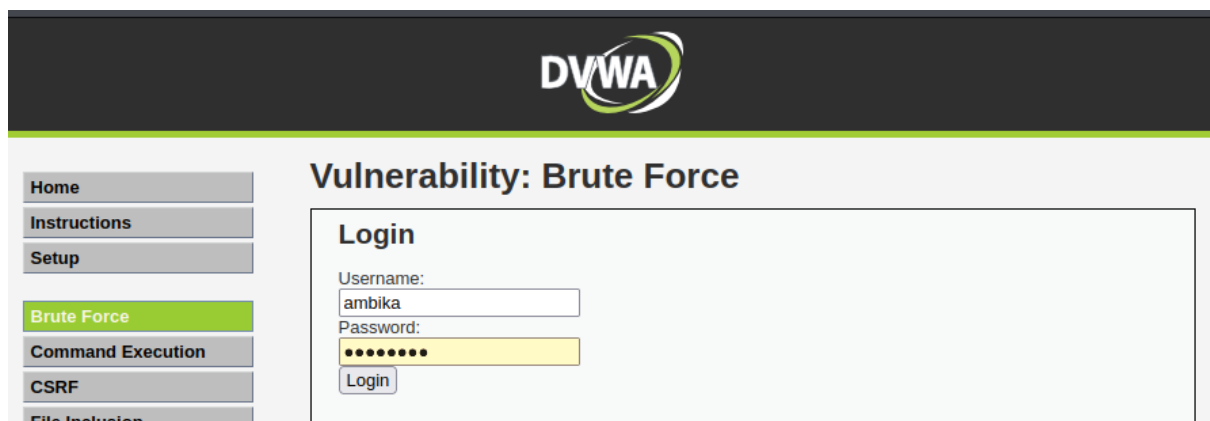
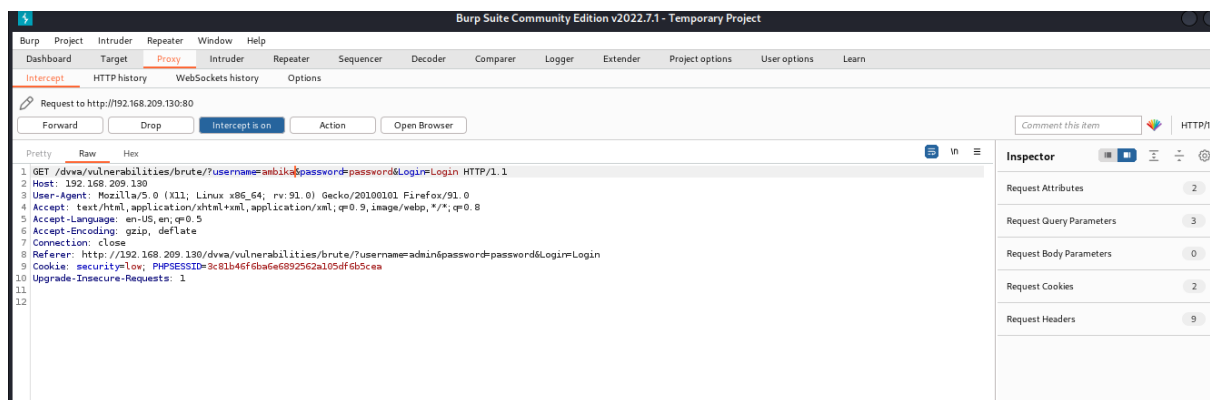


8.bruteforce attack:

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.



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.

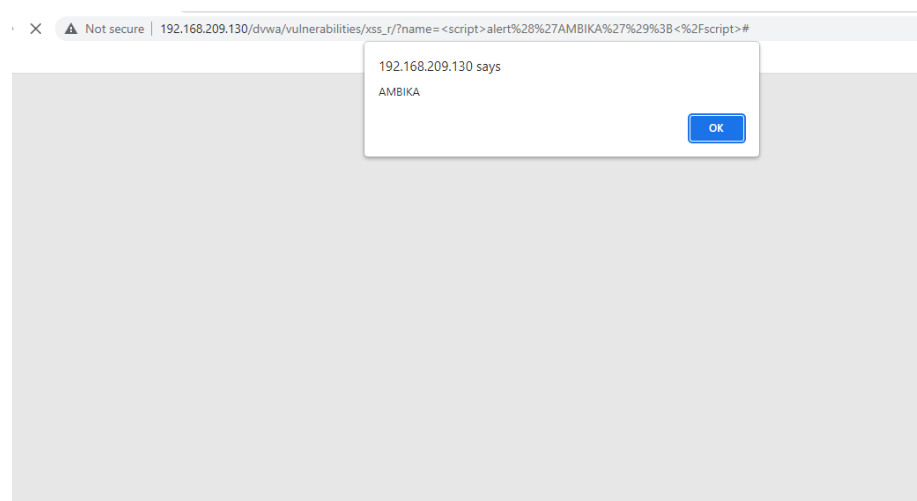
These attacks have become commonplace because it is far easier for an attacker to use a known weakness than create a specific program or attack methodology to search out vulnerabilities themselves. This fact should put known component vulnerabilities high on your security priority list to mitigate.

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.

Xss reflected:



xss stored:

