PROJECT NO :06

Web Application Security Assessment using OWASP Top 10 as a Framework for VAPT Testing

Target : DVWA [Damn Vulnerable Web Application]

1.Overview

During the testing of DVWA, we focused on identifying vulnerabilities related to SQL injection, brute force, and LFI attacks. Our approach involved using both automated tools and manual techniques to identify potential vulnerabilities.

2.List of teammates participated in the HUFFPOST:

S. No.	Name	Designation	Mobile No.
1	Mavuluri Datha Sushma	Team leader	7416427969
2	Kotte Vamsi	Team Member	6302150178
3	Kambham Sumanth	Team Member	7013709070
4	Madduluri Yabbeju	Team Member	78010709070

3. List of Vulnerable Parameter, Location discovered

S. No.	Vulnerability path	Name of the vulnerability	Reference CWE
1	http:// 10.0.2.4/001/ vulnerabilities/ sqli/	**SQL INJECTION**	**CWE-89**
2	http:// 10.0.2.4/001/ vulnerabilities/ brute/? username=test &pa ssword=test&L ogi n=Login#	BRUTEFORCE	**CWE-305**
3	http://10.0.2.4/ 001/ vulnerabilities/f i/? page=include.p hp	LOCAL FILE INCLUSION	**CWE-98**

4. Other Information

- **Tools Used :**

1. Kali Linux
2. FireFox Browser
3. BurpSuite

Application Details :

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

The aim of DVWA is to practice some of the most common web vulnerabilities, with various levels of difficulty, with a simple straightforward interface. Please note, there are both documented and undocumented vulnerabilities with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

Application Running on 10.0.2.4 localhost

MAIN VULNERABILITY REPRESENTATION FORMAT

1— 1.1 . Vulnerability Name: SQL Injection

CWE : CWE-89
OWASP Category: A03:2021 – Injection

Description: SQL injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database

Business Impact: A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to affect the execution of predefined SQL commands.

SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and become administrators of the database server.

SQL Injection is very common with PHP and ASP applications due to the prevalence of older functional interfaces. Due to the nature of programmatic interfaces available, J2EE and ASP.NET applications are less likely to have easily exploited SQL injections.

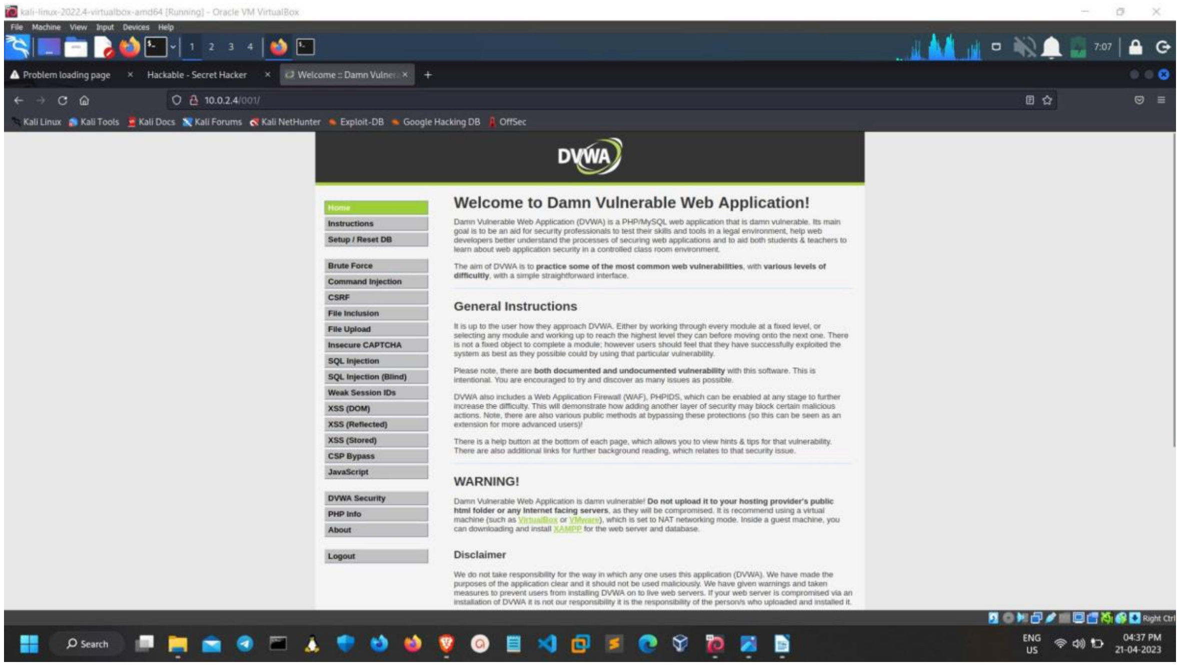
The severity of SQL Injection attacks is limited by the attacker's skill and imagination, and to a lesser extent, defense in depth countermeasures, such as low privilege connections to the database server and so on. In general, consider SQL Injection a high impact severity.

Vulnerability Path :http://10.0.2.4/001/

VulnerabilityParameter: http://10.0.2.4/001/vulnerabilities/sqli/

Steps to Reproduce :

Step 1. Access the URL



Welcome to Damn Vulnerable Web Application!

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

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with **various levels of difficultly**, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerability** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

DVWA also includes a Web Application Firewall (WAF), PHPIDS, which can be enabled at any stage to further increase the difficulty. This will demonstrate how adding another layer of security may block certain malicious actions. Note, there are also various public methods at bypassing these protections (so this can be seen as an extension for more advanced users)!

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

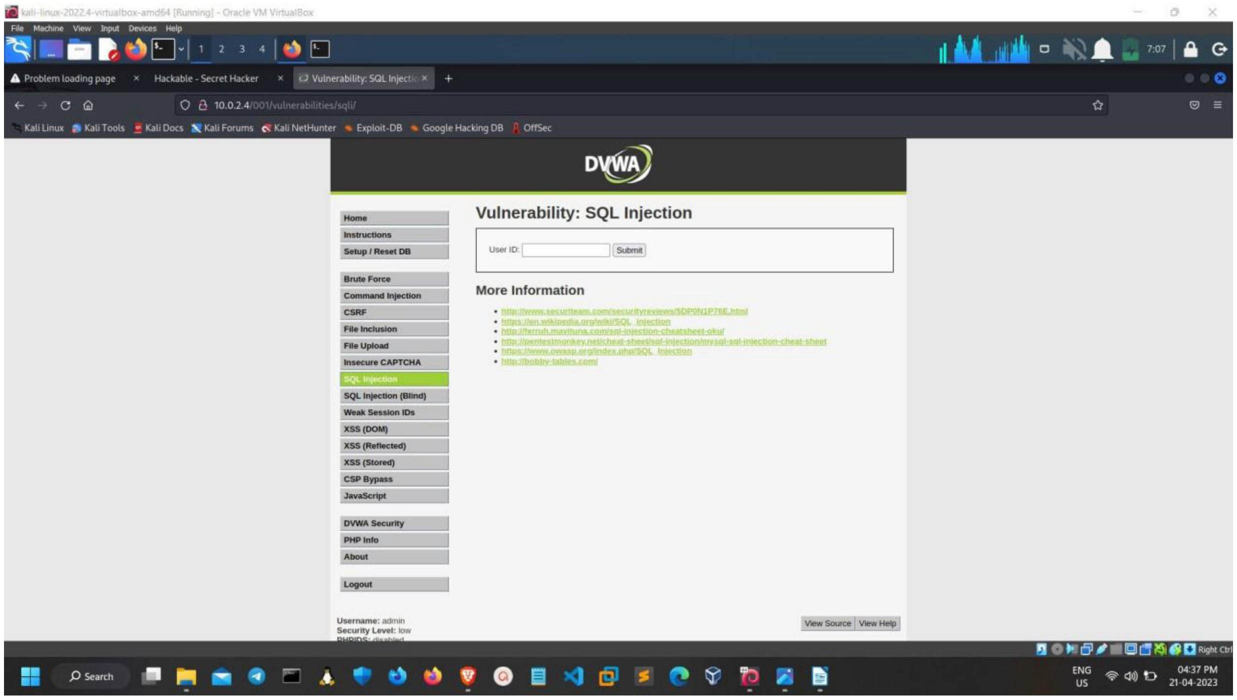
WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public html folder or any Internet facing servers**, as they will be compromised. It is recommend using a virtual machine (such as VirtualBox or VMware), which is set to NAT networking mode. Inside a guest machine, you can downloading and install XAMPP for the web server and database.

Disclaimer

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

Step 2: Go to the Sql Injection page



Vulnerability: SQL Injection

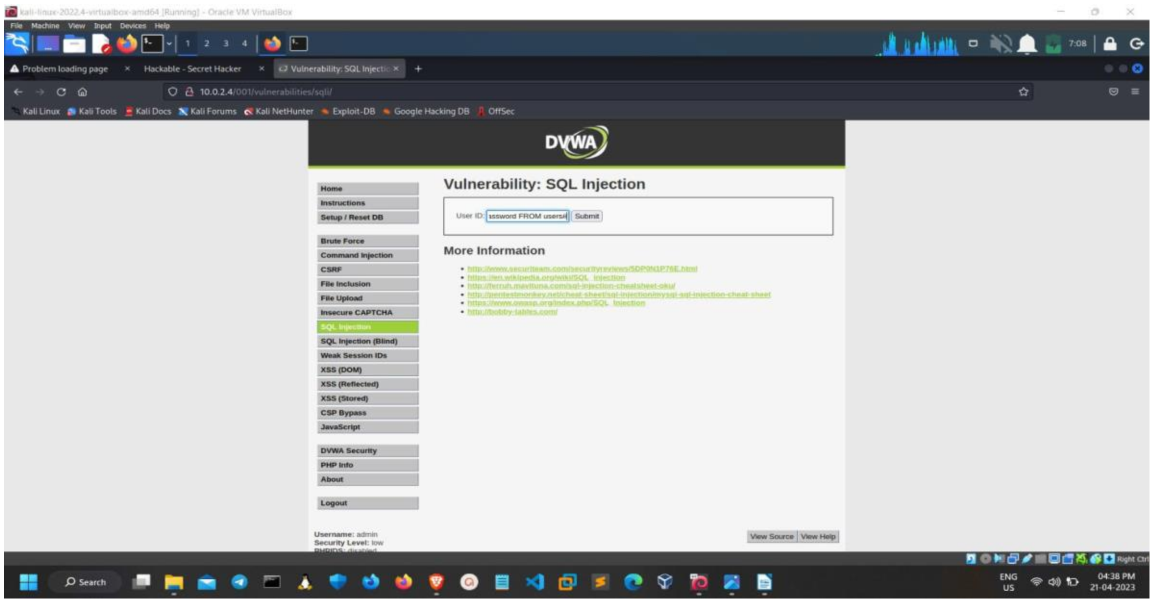
User ID: Submit

More Information

- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://www.owasp.org/index.php/SQL_Injection
- http://bobby-tables.com/

Username: admin
Security Level: low

Step 3: Now you will be redirected to the dashboard where we will enter the sql query "1' UNION SELECT user, password FROM users#" in Userid field



Vulnerability: SQL Injection

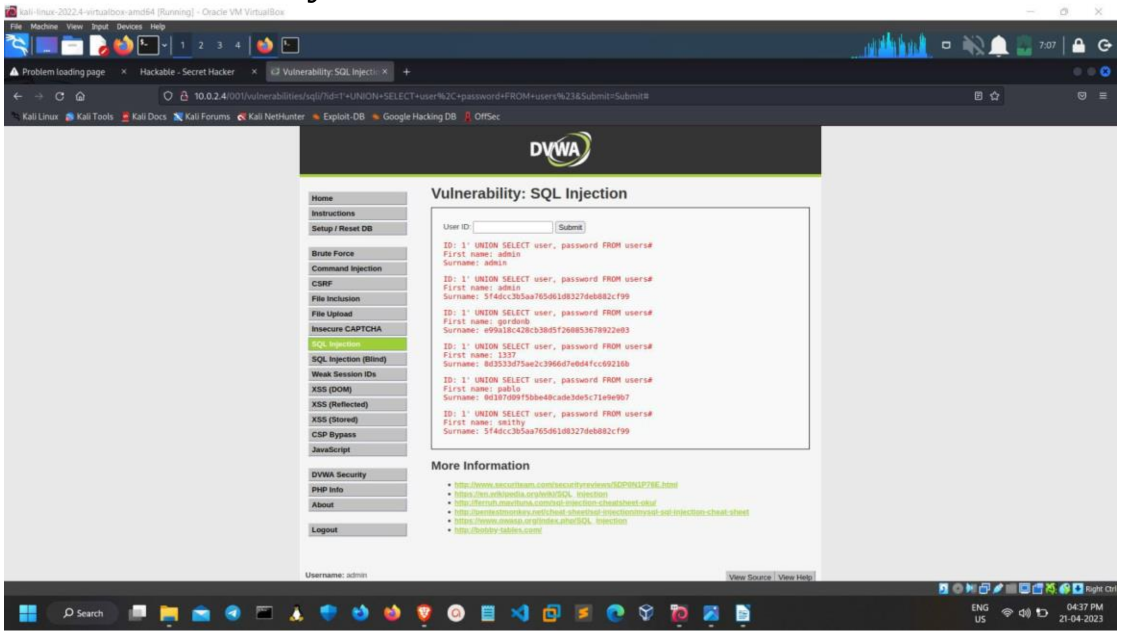
User ID: ssword FROM users# Submit

More Information

- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://www.owasp.org/index.php/SQL_Injection
- http://bobby-tables.com/

Username: admin
Security Level: low

Step 4:-after entering the query ,press submit .now we retrieved the data successfully .



`10.0.2.4/001/vulnerabilities/sqli/?id=1'+UNION+SELECT+user%2C+password+FROM+users%23&Submit=Submit#`

```
ID: 1' UNION SELECT user, password FROM users#
First name: admin
Surname: admin

ID: 1' UNION SELECT user, password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' UNION SELECT user, password FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' UNION SELECT user, password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' UNION SELECT user, password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' UNION SELECT user, password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

Recommendation:

- Do not rely on client-side input validation.
- Use a database user with restricted privileges.
- Use prepared statements and query parameterization.
- Scan your code for SQL injection vulnerabilities.

1— 1.2 . Vulnerability Name: Login BruteForce

CWE : CWE-305

OWASP Category: A2:2017-Broken Authentication

Description: Brute-force attacks are often used for attacking authentication and discovering hidden content/pages within a web application.

Business Impact:

The breach can have far-reaching effects on both users and businesses. They include:

•Identity theft – stealing someone's identity to access their accounts, such as bank accounts or credit cards. This enables the attacker to purchase goods using these details. In addition, information such as social security numbers can be sold for use in other cyber attacks.

•Loss of data – due to loss of confidentiality if data is stolen which could destroy company reputation. Additionally, there may be reputational damage caused by a leak of sensitive customer information that leads to public distrust and dissatisfaction with the business.

•Downtime – this refers to system outages where websites or computer networks cannot be accessed due to a cyber attack. This is costly to the business in terms of lost revenue, customer satisfaction as well as loss of image.

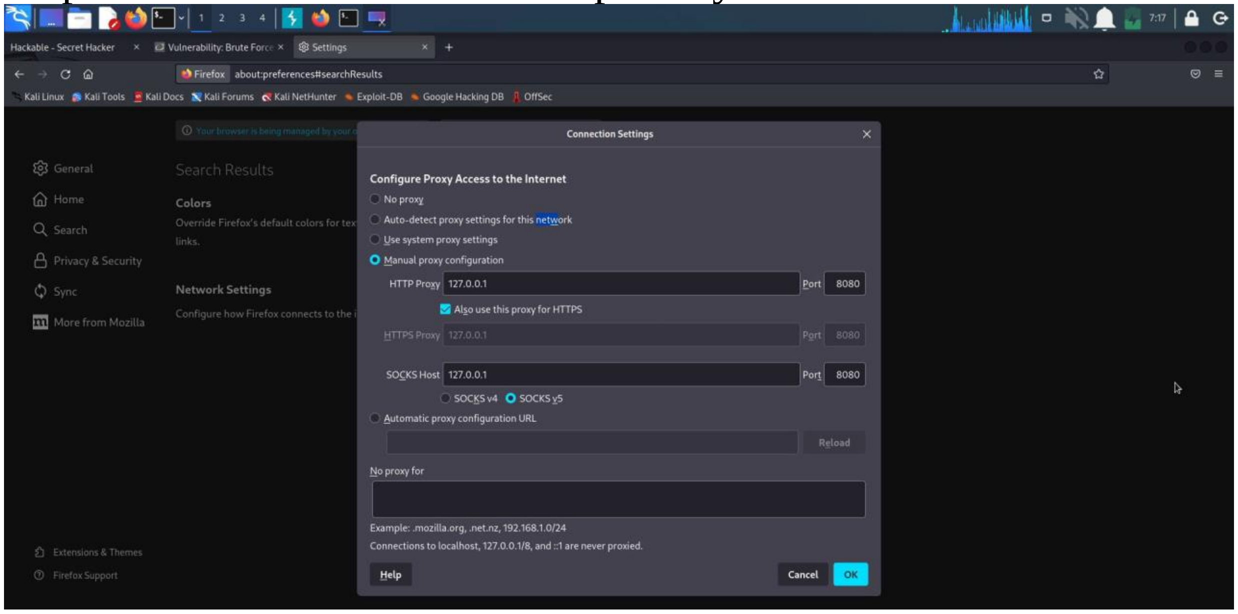
In short, hackers can use broken authentication attacks and session hijacking to gain access to the system by forging session data, such as cookies, and stealing login credentials. Thus, it would be best if you never compromised with your web applications' security.

Vulnerability Path :http://10.0.2.4/001/vulnerabilities

Vulnerability Parameter: http://10.0.2.4/001/vulnerabilities/brute/?username=test&password=test&Login=Login#

Steps to Reproduce :

Step 1. Connect Browser To Burp Proxy


Connection Settings

Configure Proxy Access to the Internet

No proxy
Auto-detect proxy settings for this network
Use system proxy settings
[✓ Manual proxy configuration]

HTTP Proxy 127.0.0.1 Port 8080

[✓ Also use this proxy for HTTPS]

HTTPS Proxy 127.0.0.1 Port 8080

SOCKS Host 127.0.0.1 Port 8080

SOCKS v4 [✓ SOCKS v5]

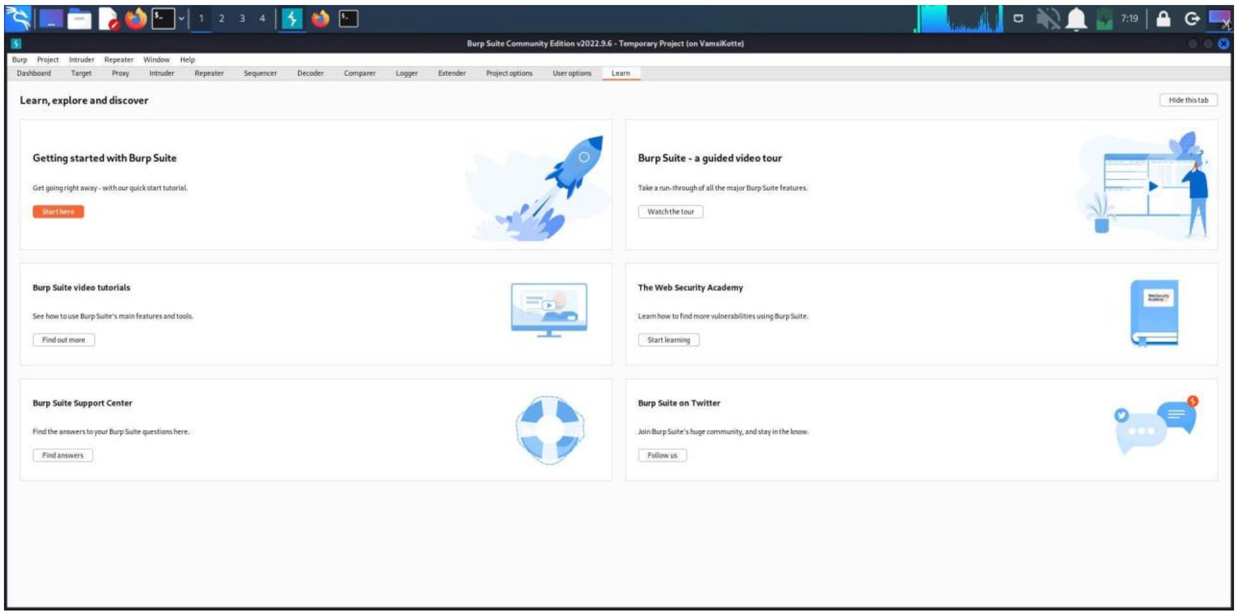
Automatic proxy configuration URL

No proxy for

Example: .mozilla.org, .net.nz, 192.168.1.0/24

Connections to localhost, 127.0.0.1/8, and ::1 are never proxied.

Step 2. Open Burp and Start intercept


Burp Suite Community Edition v2022.9.6 - Temporary Project (on VamsiKotte)

Learn, explore and discover

Getting started with Burp Suite

Get going right away - with our quick start tutorial.

Burp Suite - a guided video tour

Take a run-through of all the major Burp Suite features.

Burp Suite video tutorials

See how to use Burp Suite's main features and tools.

The Web Security Academy

Learn how to find more vulnerabilities using Burp Suite.

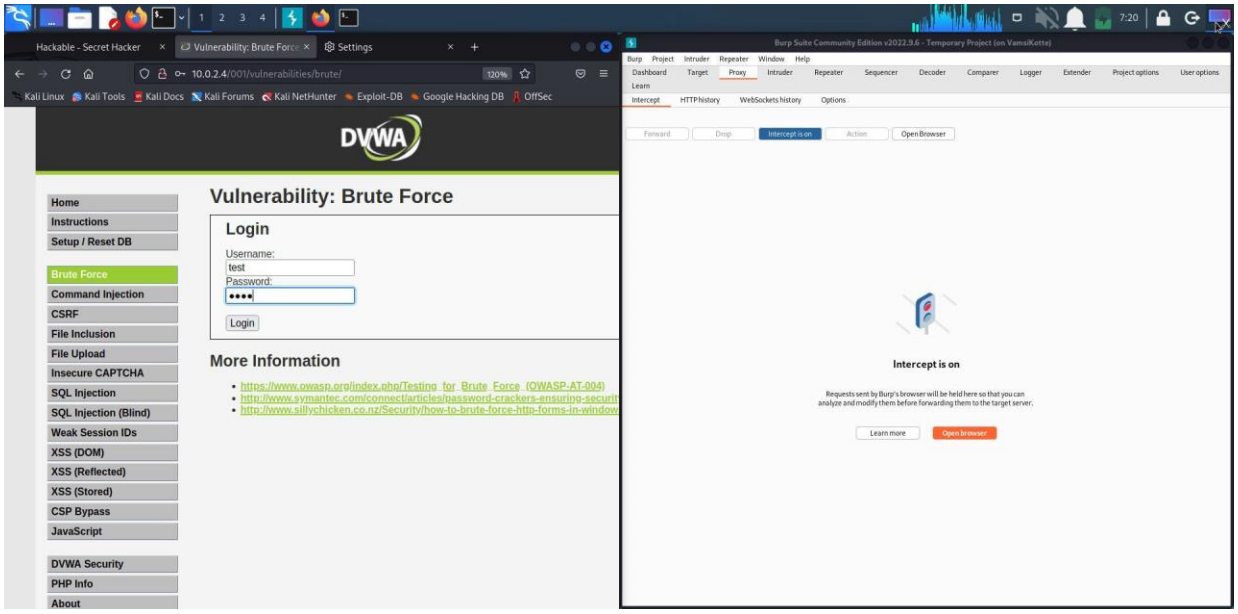
Burp Suite Support Center

Find the answers to your Burp Suite questions here.

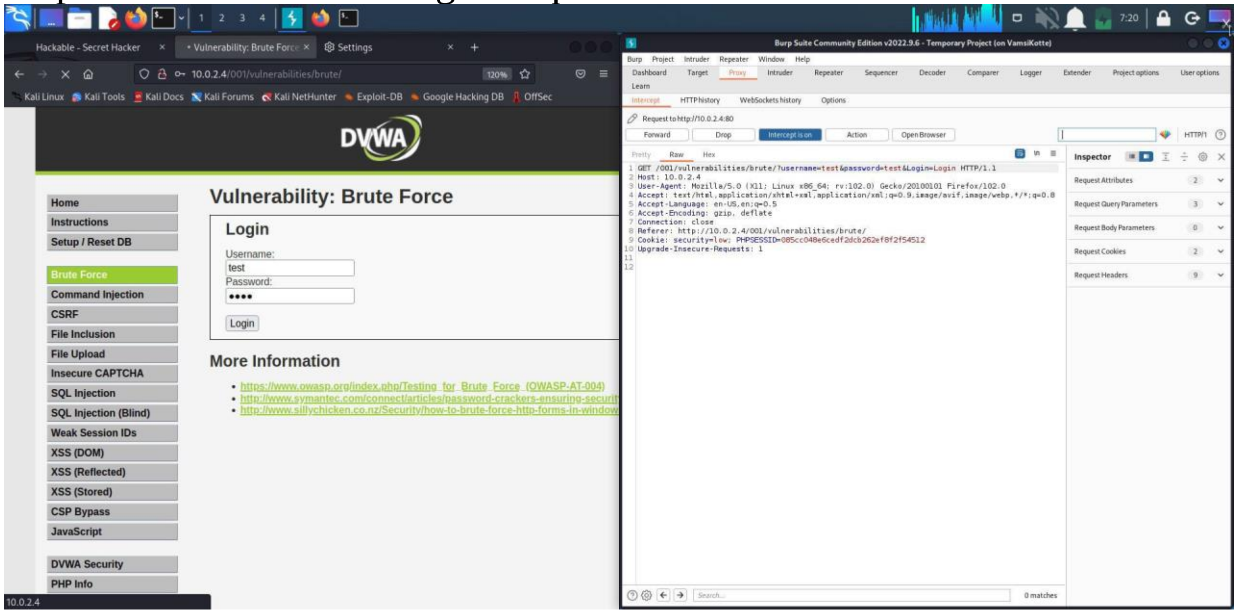
Burp Suite on Twitter

Join Burp Suite's huge community, and stay in the know.

Step 3 . Now access the url.give some fake credentials example “test:test”

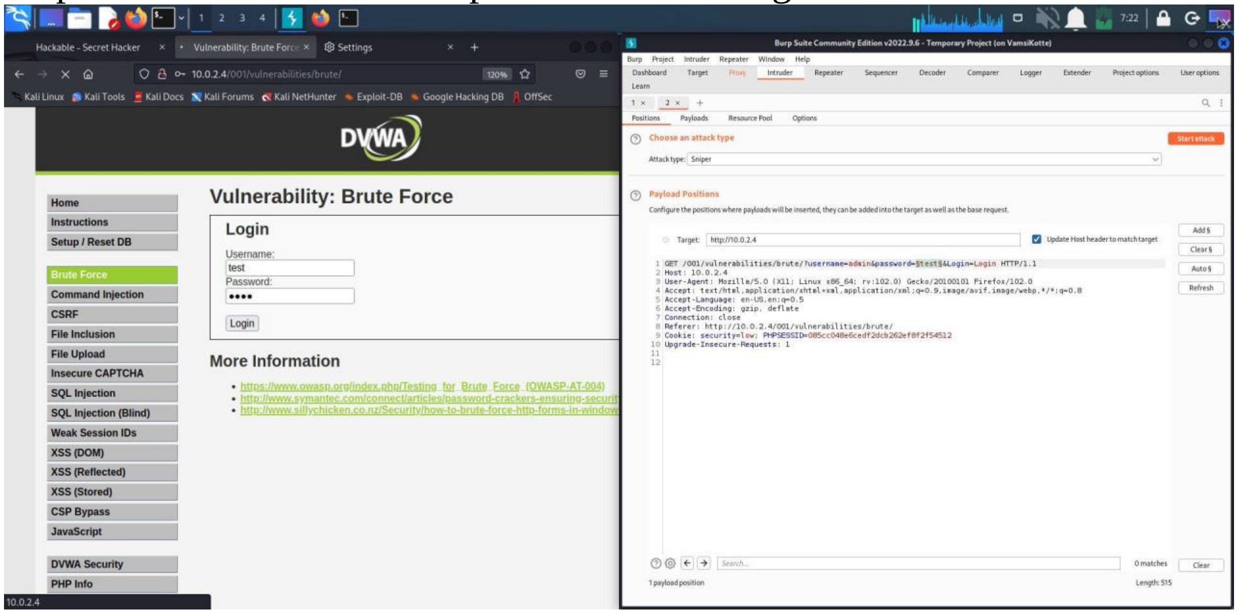

10.0.2.4/001/vulnerabilities/brute/
Vulnerability: Brute Force
Login
Username:
test
Password:
Login
More Information
Intercept is on
Requests sent by Burp's browser will be held here so that you can analyze and modify them before forwarding them to the target server.

Step 4. Now Send The Caught Request To Intruder


Burp Suite Community Edition v2022.9.6 - Temporary Project (on VamsiKotte)
Request to http://10.0.2.4:80
Forward
Drop
Intercept is on
Action
Open Browser
Inspector

```
GET /001/vulnerabilities/brute/?username=test&password=test&Login=Login HTTP/1.1
Host: 10.0.2.4
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://10.0.2.4/001/vulnerabilities/brute/
Cookie: security=low; PHPSESSID=085cc048e6cedf2dcb262ef8f2f54512
Upgrade-Insecure-Requests: 1
```

Step 5 .Now Add Position to password and change username to “admin”



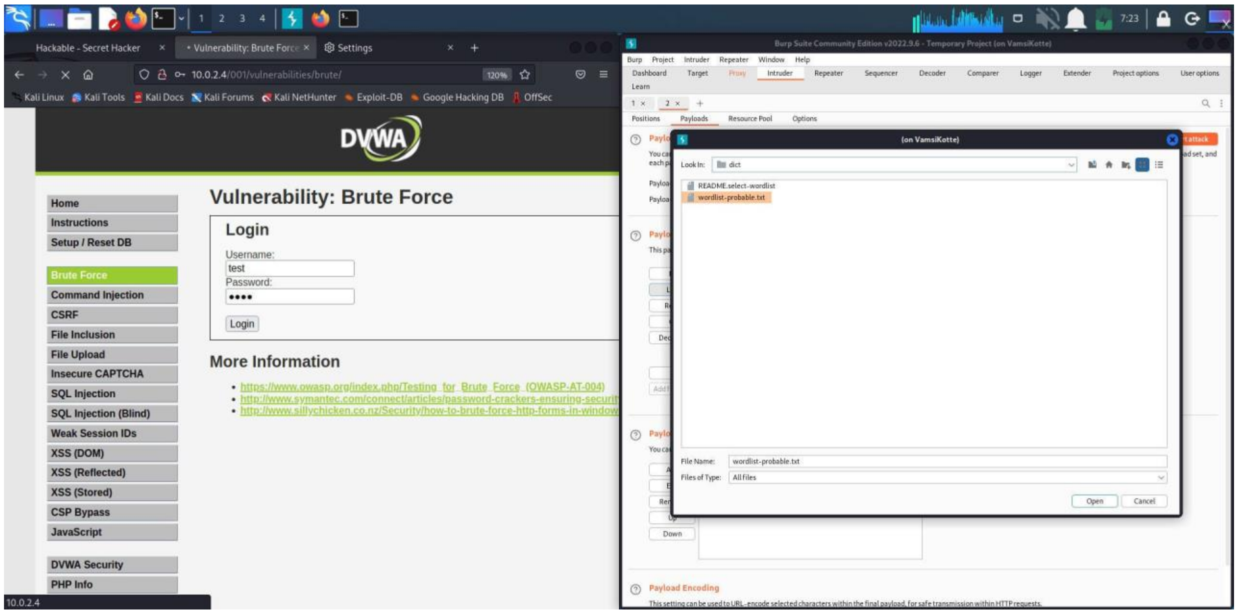
Attack type: Sniper

Target: http://10.0.2.4

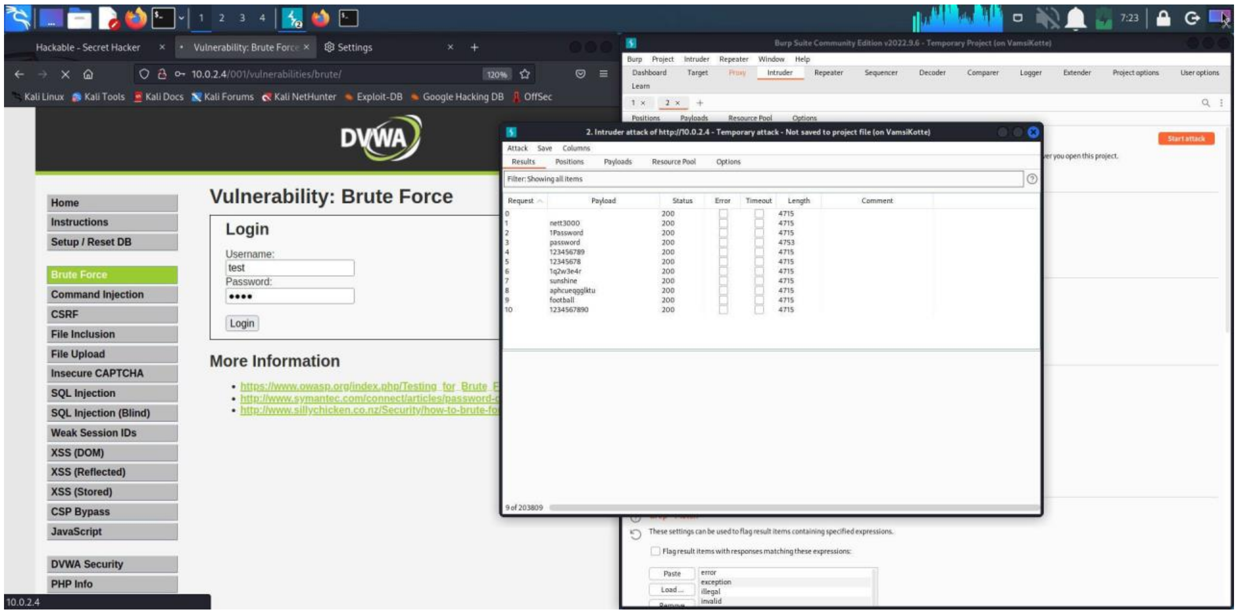
```
GET /001/vulnerabilities/brute/?username=admin&password=§test§&Login=Login HTTP/1.1
Host: 10.0.2.4
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://10.0.2.4/001/vulnerabilities/brute/
Cookie: security=low; PHPSESSID=085cc048e6cedf2dcb262ef8f2f54512
Upgrade-Insecure-Requests: 1
```

Burp Suite Community Edition v2022.9.6 - Temporary Project (on VamsiKotte)
Vulnerability: Brute Force
Login
Username:
Password:
1 payload position
Length: 515

Step 6. load the payload wordlist. I used the kali default “/usr/share/dict/wordlist-probable.txt ”



Step 7. Start The Attack



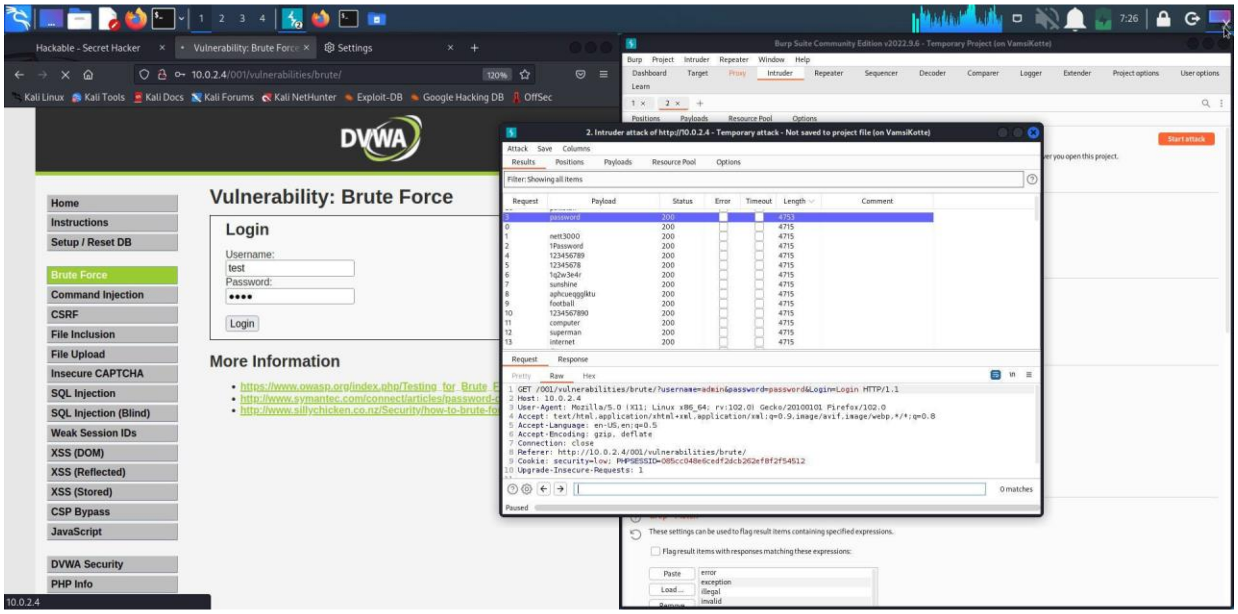
2. Intruder attack of http://10.0.2.4 - Temporary attack - Not saved to project file (on VamsiKotte)

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
0		200	☐	☐	4715	
1	nett3000	200	☐	☐	4715	
2	1Password	200	☐	☐	4715	
3	password	200	☐	☐	4753	
4	123456789	200	☐	☐	4715	
5	12345678	200	☐	☐	4715	
6	1q2w3e4r	200	☐	☐	4715	
7	sunshine	200	☐	☐	4715	
8	aphcueqgglktu	200	☐	☐	4715	
9	football	200	☐	☐	4715	
10	1234567890	200	☐	☐	4715	

9 of 203809

Step 8 . Wait untill it completed. And sort by Desc.Size



2. Intruder attack of http://10.0.2.4 - Temporary attack - Not saved to project file (on VamsiKotte)

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
3	password	200	☐	☐	4753	
0		200	☐	☐	4715	
1	nett3000	200	☐	☐	4715	
2	1Password	200	☐	☐	4715	
4	123456789	200	☐	☐	4715	
5	12345678	200	☐	☐	4715	
6	1q2w3e4r	200	☐	☐	4715	
7	sunshine	200	☐	☐	4715	
8	aphcueqgglktu	200	☐	☐	4715	
9	football	200	☐	☐	4715	
10	1234567890	200	☐	☐	4715	
11	computer	200	☐	☐	4715	
12	superman	200	☐	☐	4715	
13	internet	200	☐	☐	4715	

```
GET /001/vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
Host: 10.0.2.4
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://10.0.2.4/001/vulnerabilities/brute/
Cookie: security=low; PHPSESSID=085cc048e6cedf2dcb262ef8f2f54512
Upgrade-Insecure-Requests: 1
```

0 matches

Paused

Recommendation:

- You can keep brute force attacks at bay and drastically improve your data security by having a strong password policy, limiting login attempts, enabling two-factor authentication, using CAPTCHAs, and blocking malicious IP addresses.

1— 1.3 . Vulnerability Name:- File Inclusion

CWE : - CWE-98

OWASP Category:-A06 Vulnerable and Outdated Components.

Description:-

Local file inclusion (also known as LFI) is the process of including files, that are already locally present on the server, through the exploiting of vulnerable inclusion procedures implemented in the application.

Business Impact::-

The impact of a Local File Inclusion attack can vary based on the exploitation and the read permissions of the webserver user. Based on these factors, an attacker can gather usernames via an /etc/passwd file, harvest useful information from log files, or combine this vulnerability with other attack vectors (such as file upload vulnerability) to execute commands remotely.

Let’s take a closer look at three possible outcomes of local file inclusion:

1. Information disclosure

Although not the worst outcome of a local file inclusion vulnerability, information disclosure can reveal important information about the application and its configuration. That information can be valuable to an attacker to gain a deeper understanding of the application and can help them detect and exploit other vulnerabilities.

2. Directory Traversal

A local file inclusion vulnerability can lead to Directory Traversal attacks, where an attacker will try to find and access files on the web server to gain more useful information, such as log files. Log files can reveal the structure of the application or expose paths to sensitive files.

3. Remote Code Execution

Combined with a file upload vulnerability, a Local File vulnerability can lead to remote code execution. In this case the attacker would use LFI to execute the unwanted file.

Vulnerability Path :-

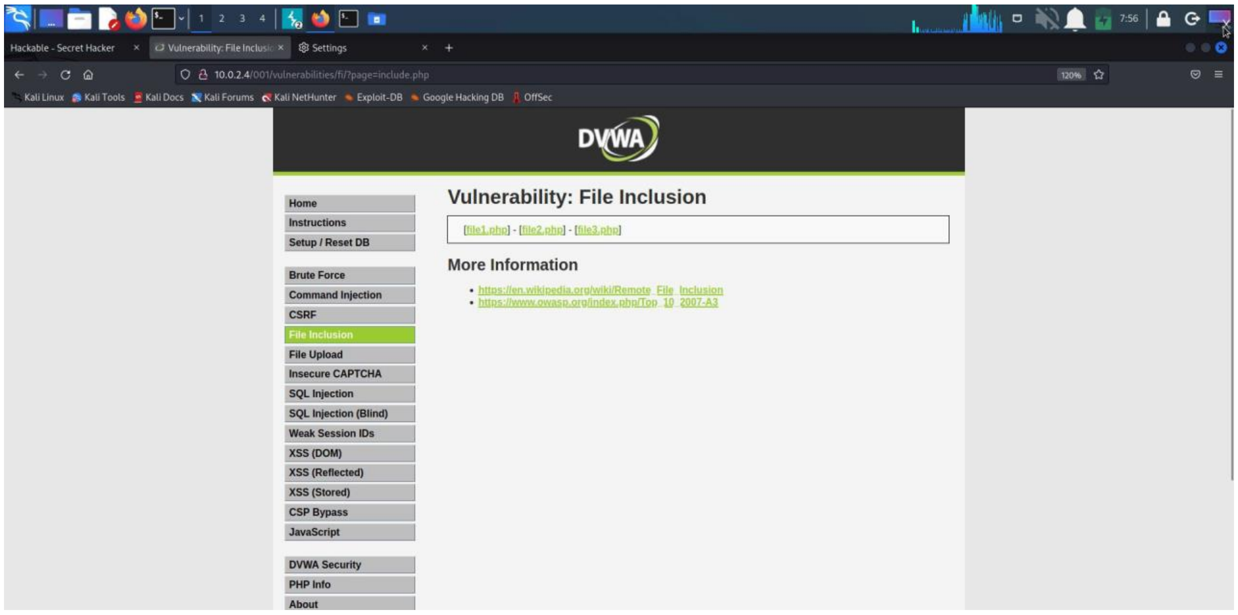
http://10.0.2.4/001/vulnerabilities

Vulnerability Parameter:-

http://10.0.2.4/001/vulnerabilities/fi/?page=include.php

Steps to Reproduce :-

Step 1 . Go To Url Open Any one file Given.



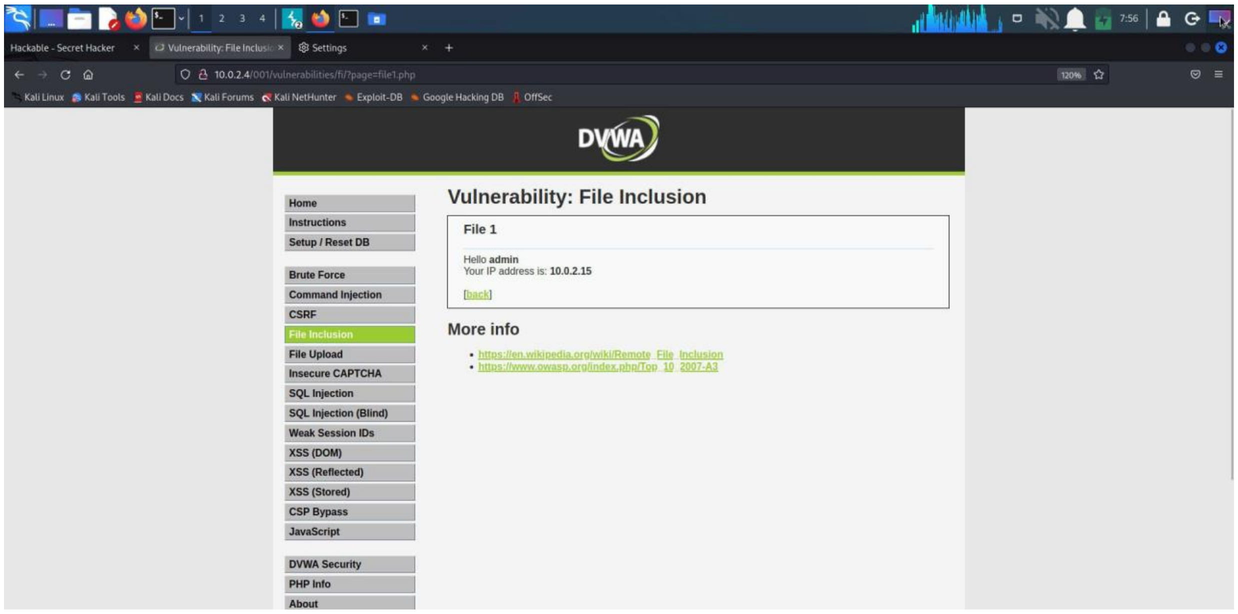
Vulnerability: File Inclusion

[file1.php] - [file2.php] - [file3.php]

More Information

- https://en.wikipedia.org/wiki/Remote_File_Inclusion
- https://www.owasp.org/index.php/Top_10_2007-A3

Step 2. Observe The Url.



10.0.2.4/001/vulnerabilities/fi/?page=file1.php

Vulnerability: File Inclusion

File 1

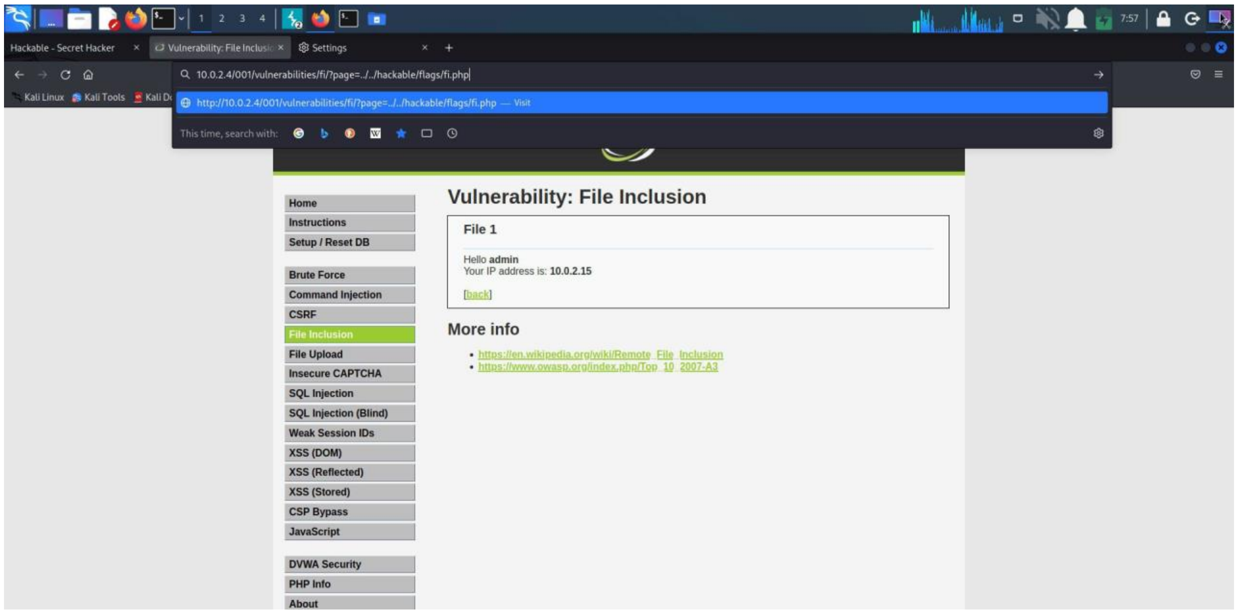
Hello admin
Your IP address is: 10.0.2.15

[back]

More info

- https://en.wikipedia.org/wiki/Remote_File_Inclusion
- https://www.owasp.org/index.php/Top_10_2007-A3

Step 3. Change the url’s “page” parameter to “../../hackable/flags/fi.php. ” , to change through file directories and to access fi.php file.



10.0.2.4/001/vulnerabilities/fi/?page=../../hackable/flags/fi.php

http://10.0.2.4/001/vulnerabilities/fi/?page=../../hackable/flags/fi.php — Visit

Vulnerability: File Inclusion

File 1

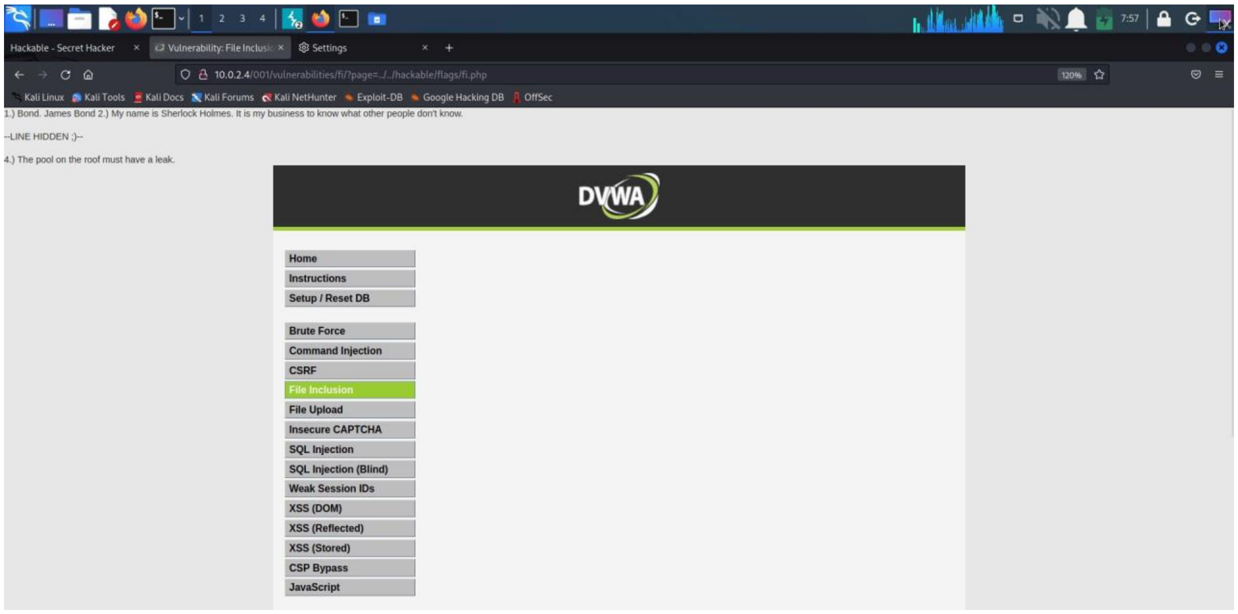
Hello admin
Your IP address is: 10.0.2.15

[back]

More info

- https://en.wikipedia.org/wiki/Remote_File_Inclusion
- https://www.owasp.org/index.php/Top_10_2007-A3

Step 4. we opened an unauth-file



```
10.0.2.4/001/vulnerabilities/fi/?page=../../hackable/flags/fi.php
```

1.) Bond. James Bond 2.) My name is Sherlock Holmes. It is my business to know what other people don't know.

--LINE HIDDEN ;)--

4.) The pool on the roof must have a leak.

Recommendation:-

•**ID assignation** – save your file paths in a secure database and give an ID for every single one, this way users only get to see their ID without viewing or altering the path

•**Whitelisting** – use verified and secured whitelist files and ignore everything else

•**Use databases** – don’t include files on a web server that can be compromised, use a database instead

•**Better server instructions** – make the server send download headers automatically instead of executing files in a specified directory.

SUMMARY :

SQL Injection: During the SQL injection testing, we identified several vulnerabilities within the DVWA application that allowed an attacker to execute arbitrary SQL queries against the database. This was achieved by manipulating the input fields of the application and injecting malicious SQL code, allowing an attacker to bypass the application's security mechanisms and access sensitive information.

Brute Force: In the brute force testing, we attempted to gain unauthorized access to the DVWA application by guessing the credentials of a valid user account. We used a combination of automated tools and manual techniques to guess the username and password of the application. Through this testing, we identified several vulnerabilities in the application's authentication mechanism, allowing an attacker to gain unauthorized access to the application.

LFI: During the LFI testing, we identified several vulnerabilities within the DVWA application that allowed an attacker to read sensitive files on the target system by exploiting a Local File Inclusion vulnerability. This was achieved by manipulating the input fields of the application and injecting malicious code, allowing an attacker to read sensitive files such as /etc/passwd, which could be used to further compromise the target system.

Overall, we were able to identify several vulnerabilities in the DVWA application through SQL injection, brute force, and LFI testing. These vulnerabilities can be exploited by attackers to gain unauthorized access to sensitive information or compromise the target system.