## 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**  **2**  **3**  **4** | **Mavuluri Datha Sushma**  **Kotte Vamsi**  **Kambham Sumanth**  **Madduluri Yabbeju** | **Team leader**  **Team Member**  **Team Member**  **Team Member** | **7416427969**  **6302150178**  **7013709070**  **78010709070** |

**3. List of Vulnerable Parameter, Location discovered**

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

1. **Other Information**
   * **Tools Used :**
2. Kali Linux
3. FireFox Browser
4. 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.

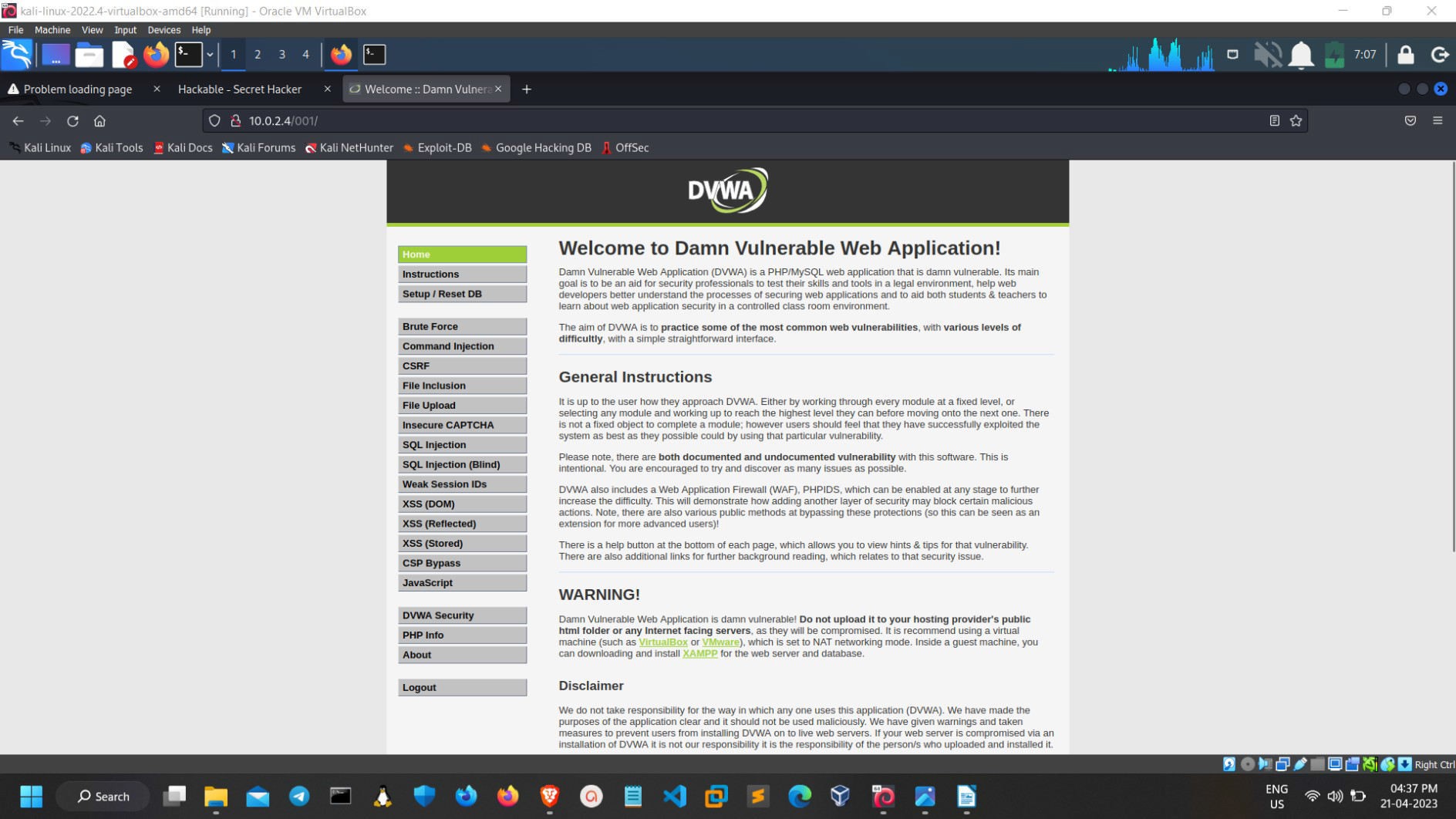
The severity of SQL Injection attacks is limited by the attacker’s skill and imagination, and to a lesser extent, defense in depth countermeasures, such as low privilege connections to the database server and so on. In general, consider SQL Injection a high impact severity.

**Vulnerability Path** :<http://10.0.2.4/001/>

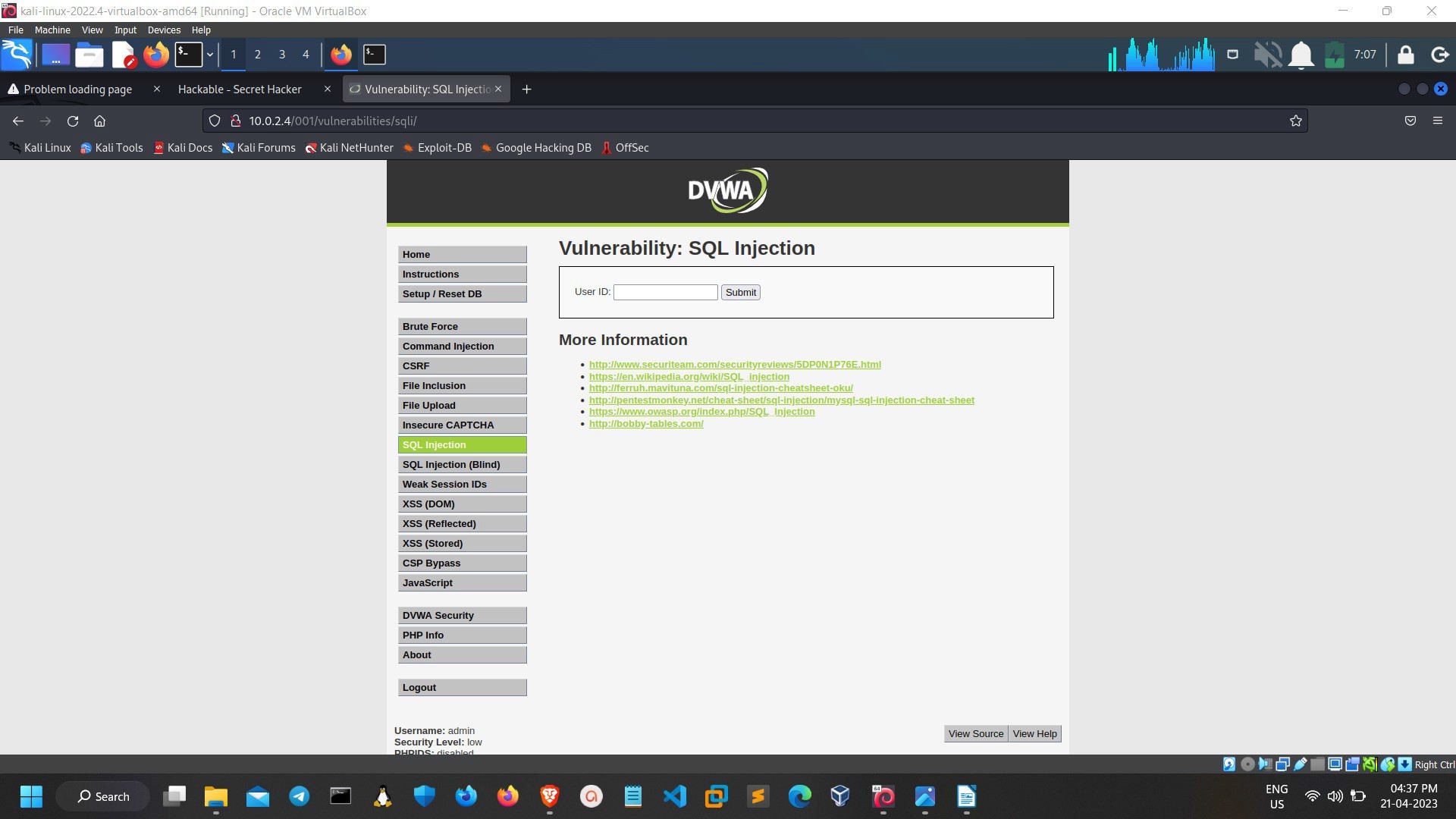
**VulnerabilityParameter**: <http://10.0.2.4/001/vulnerabilities/sqli/>

#### Steps to Reproduce :

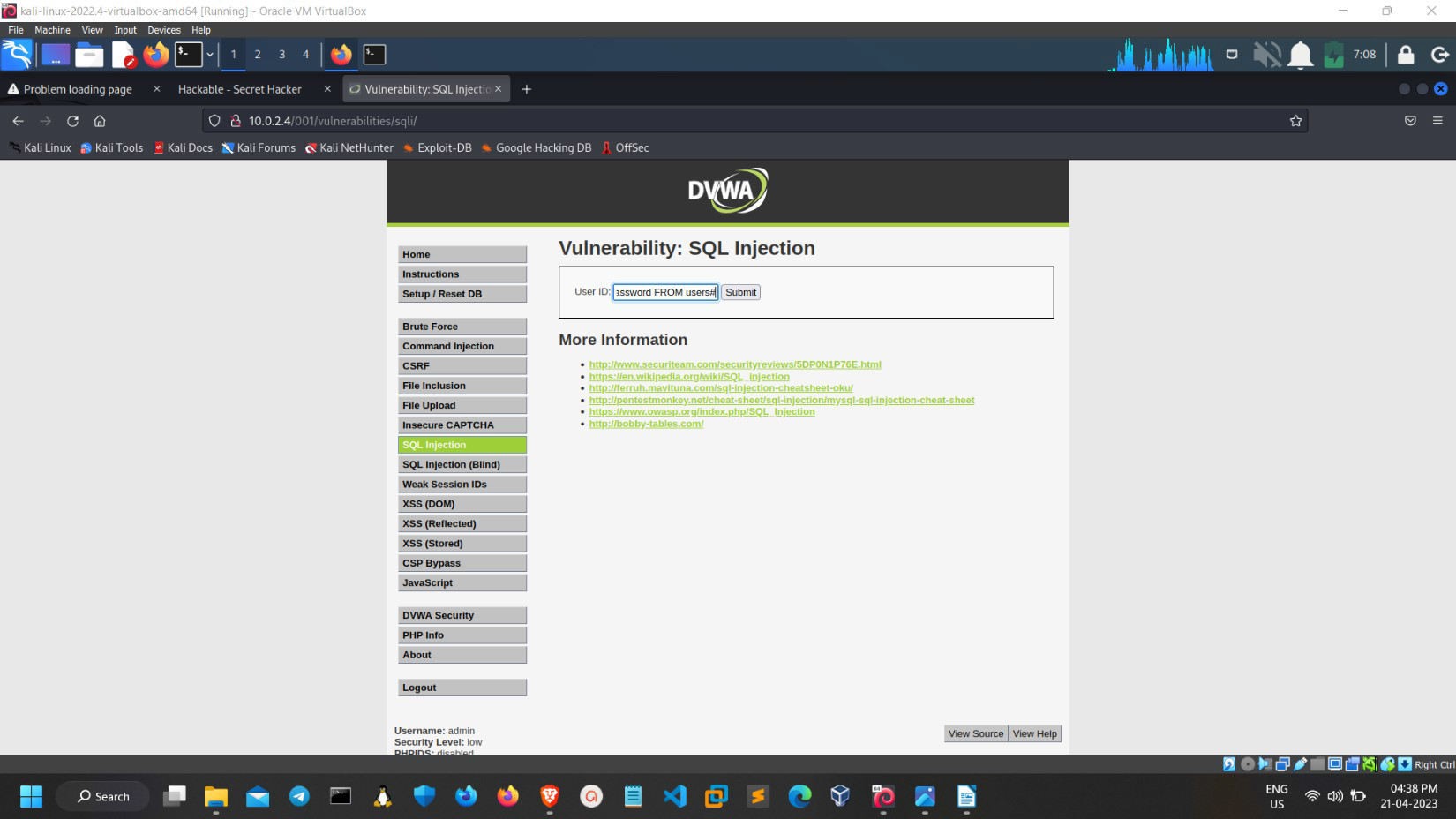
##### Step 1. Access the URL



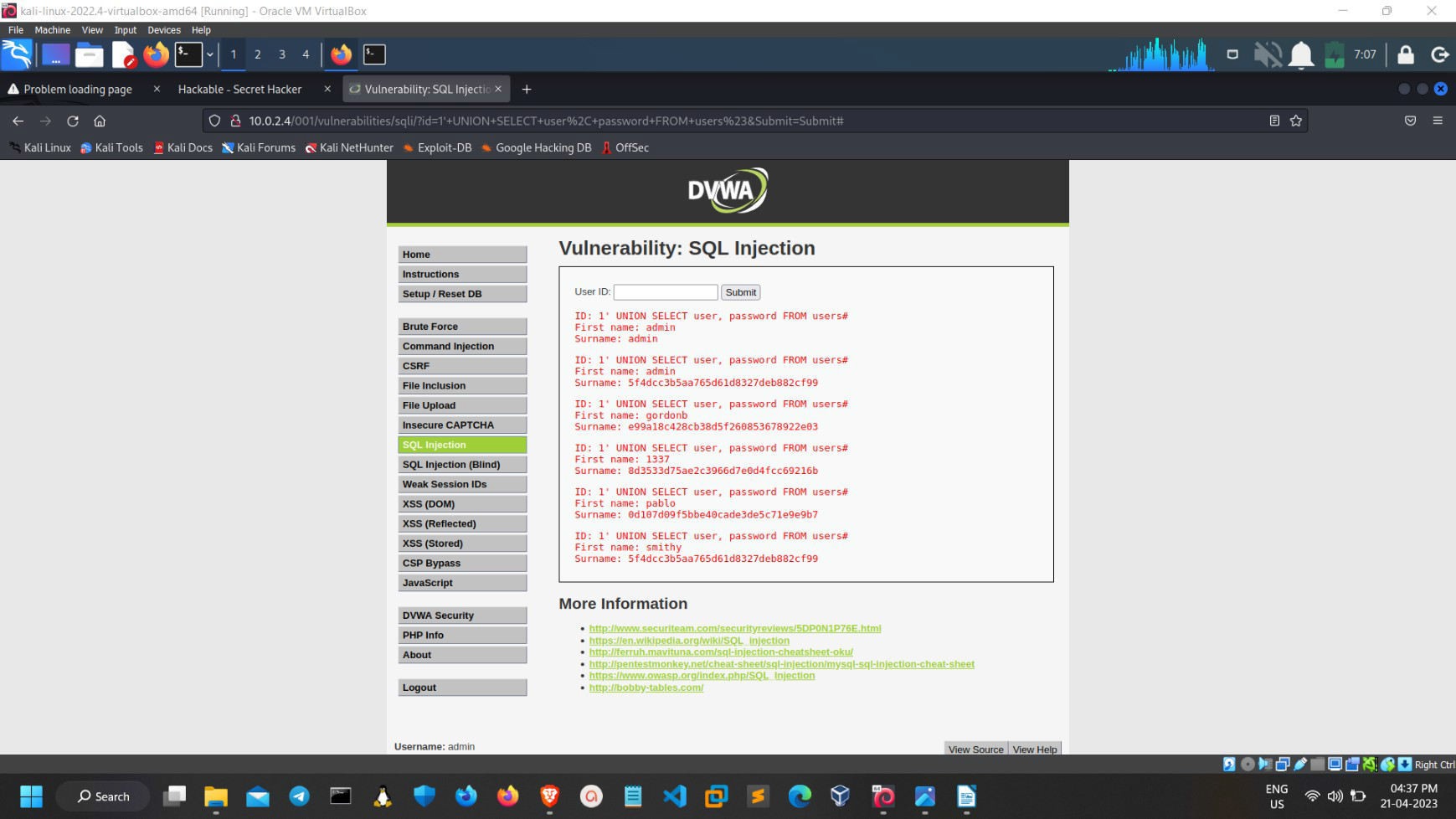
Step 2: Go to the Sql Injection page



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



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



#### 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.

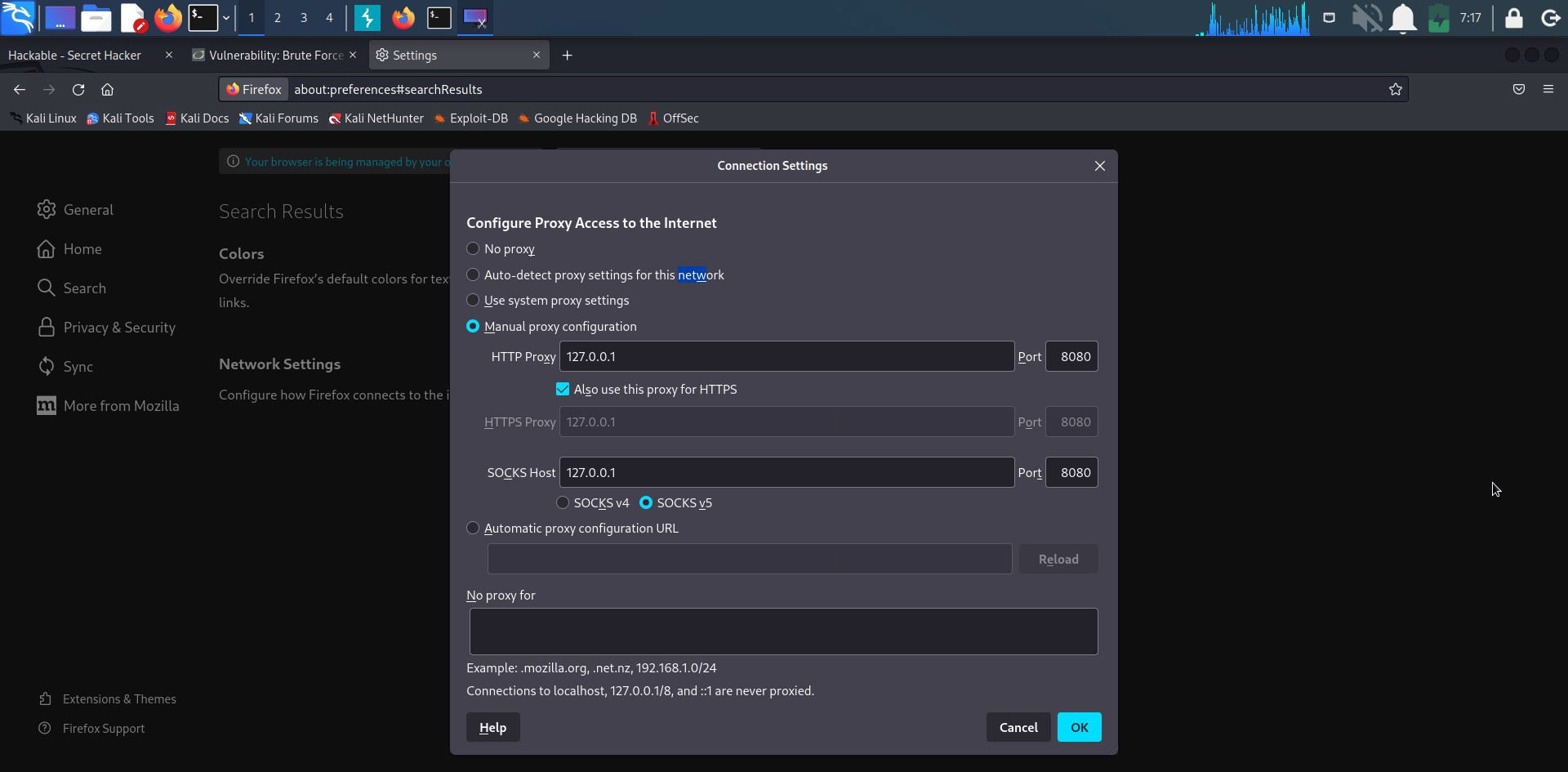
In short, hackers can use broken authentication attacks and session hijacking to gain access to the system by forging session data, such as cookies, and stealing login credentials. Thus, it would be best if you never compromised with your web applications' security.

**Vulnerability Path** :<http://10.0.2.4/001/vulnerabilities>

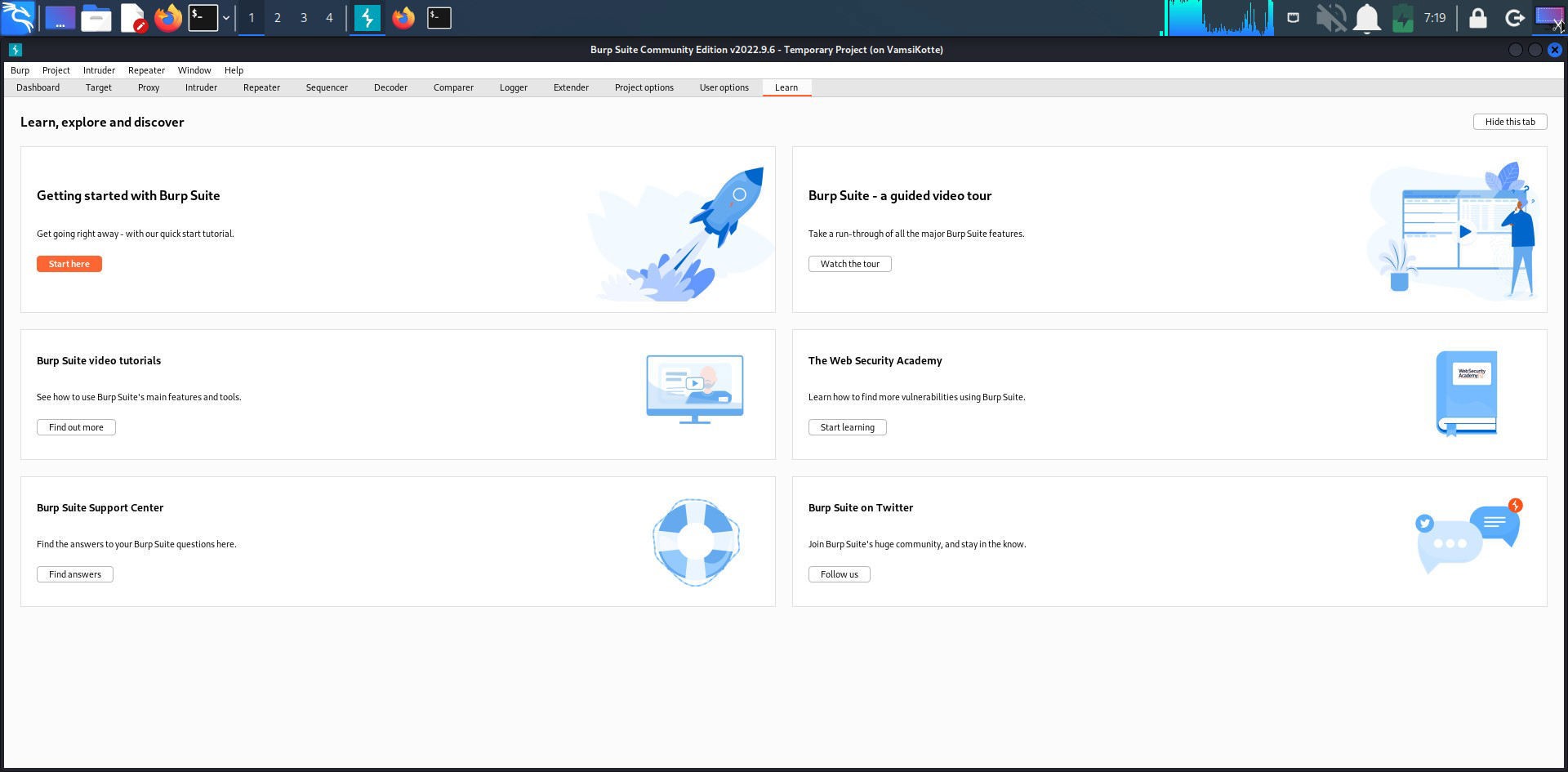
**Vulnerability Parameter**: [http://10.0.2.4/001/vulnerabilities/brute/?](http://10.0.2.4/001/vulnerabilities/brute/) username=test&password=test&Login=Login#

#### Steps to Reproduce :

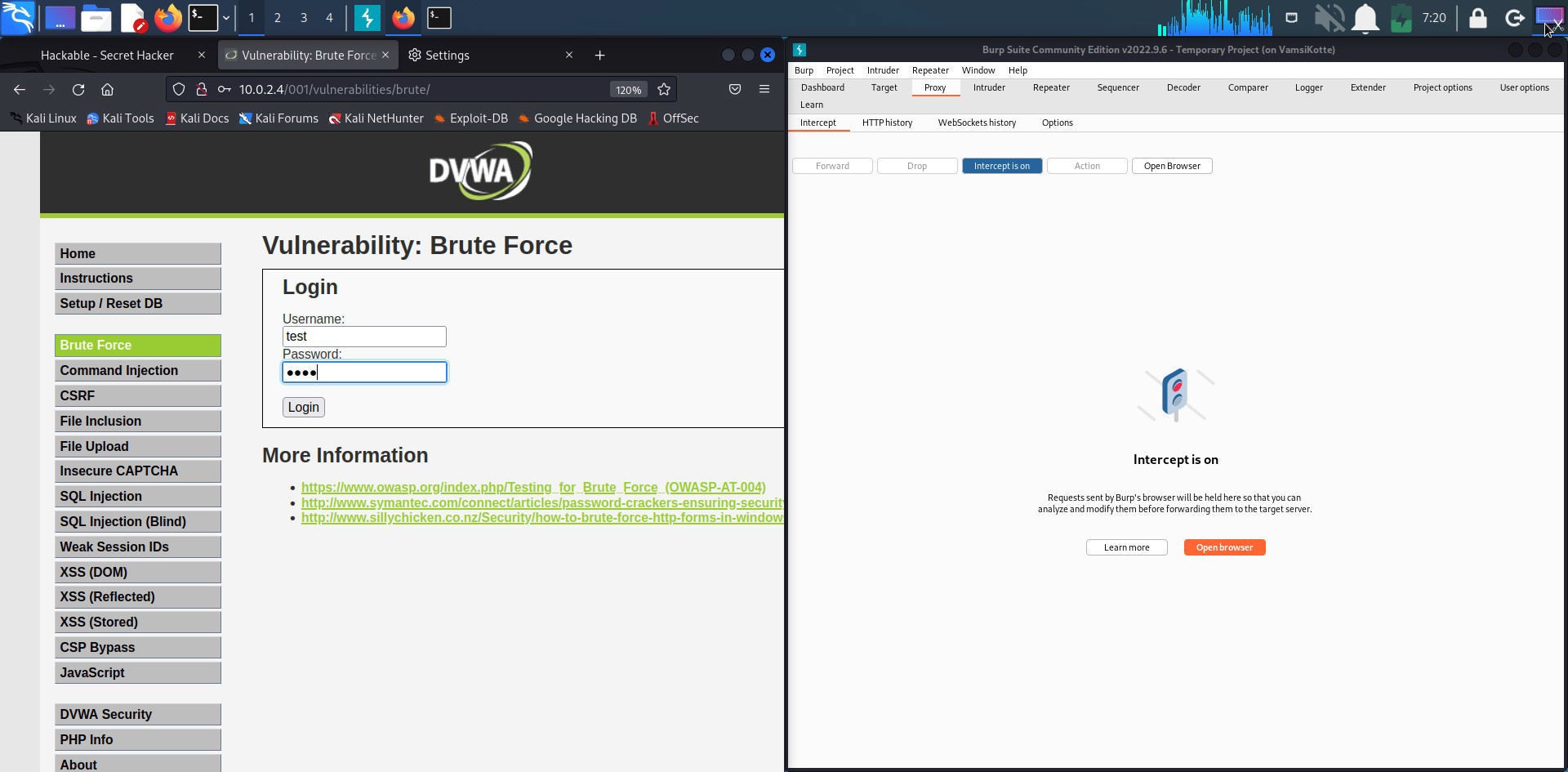
##### Step 1. Connect Browser To Burp Proxy



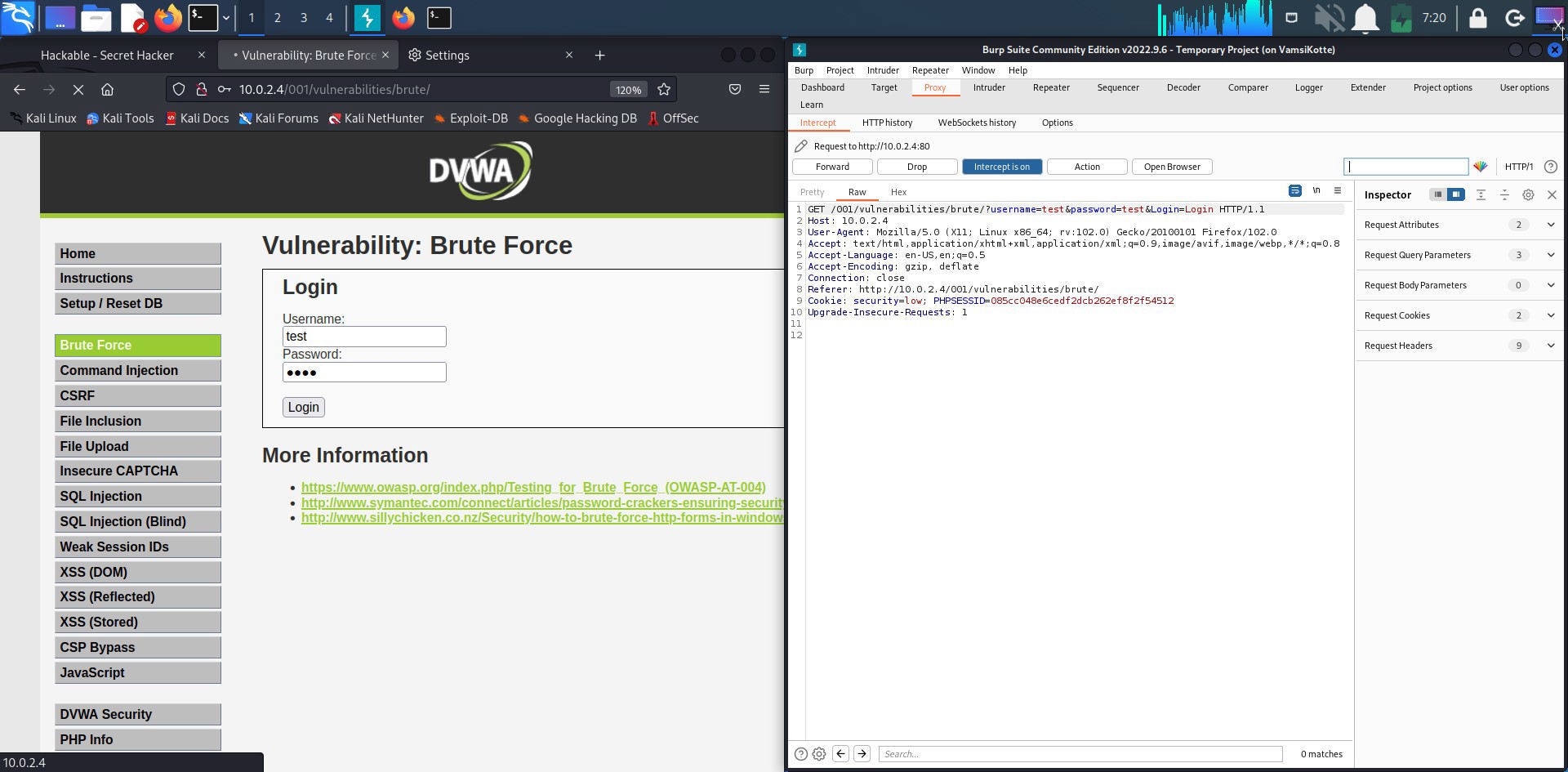
Step 2. Open Burp and Start intercept



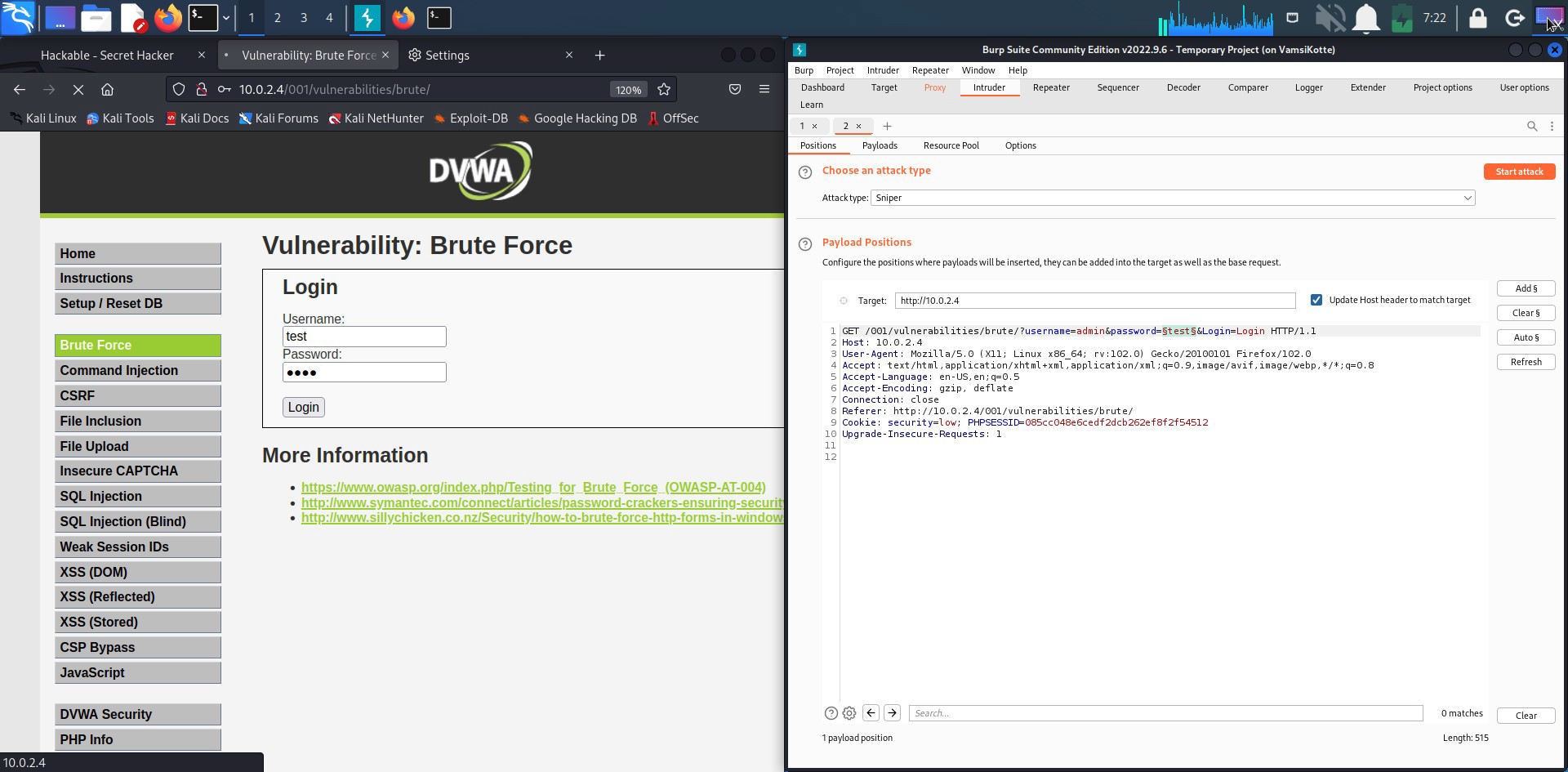
Step 3 . Now access the url.give some fake credentials example “test:test”



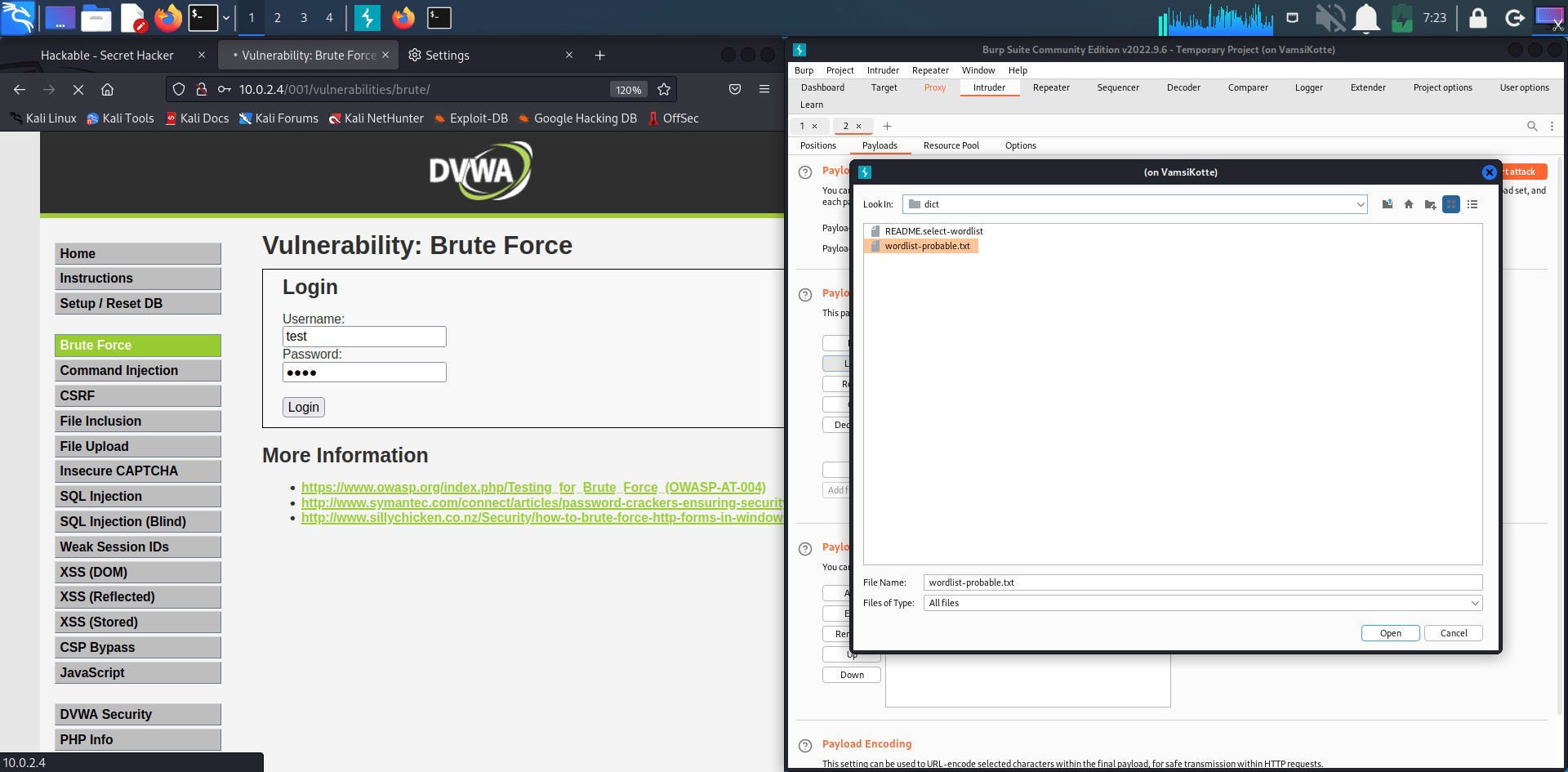
Step 4. Now Send The Caught Request To Intruder



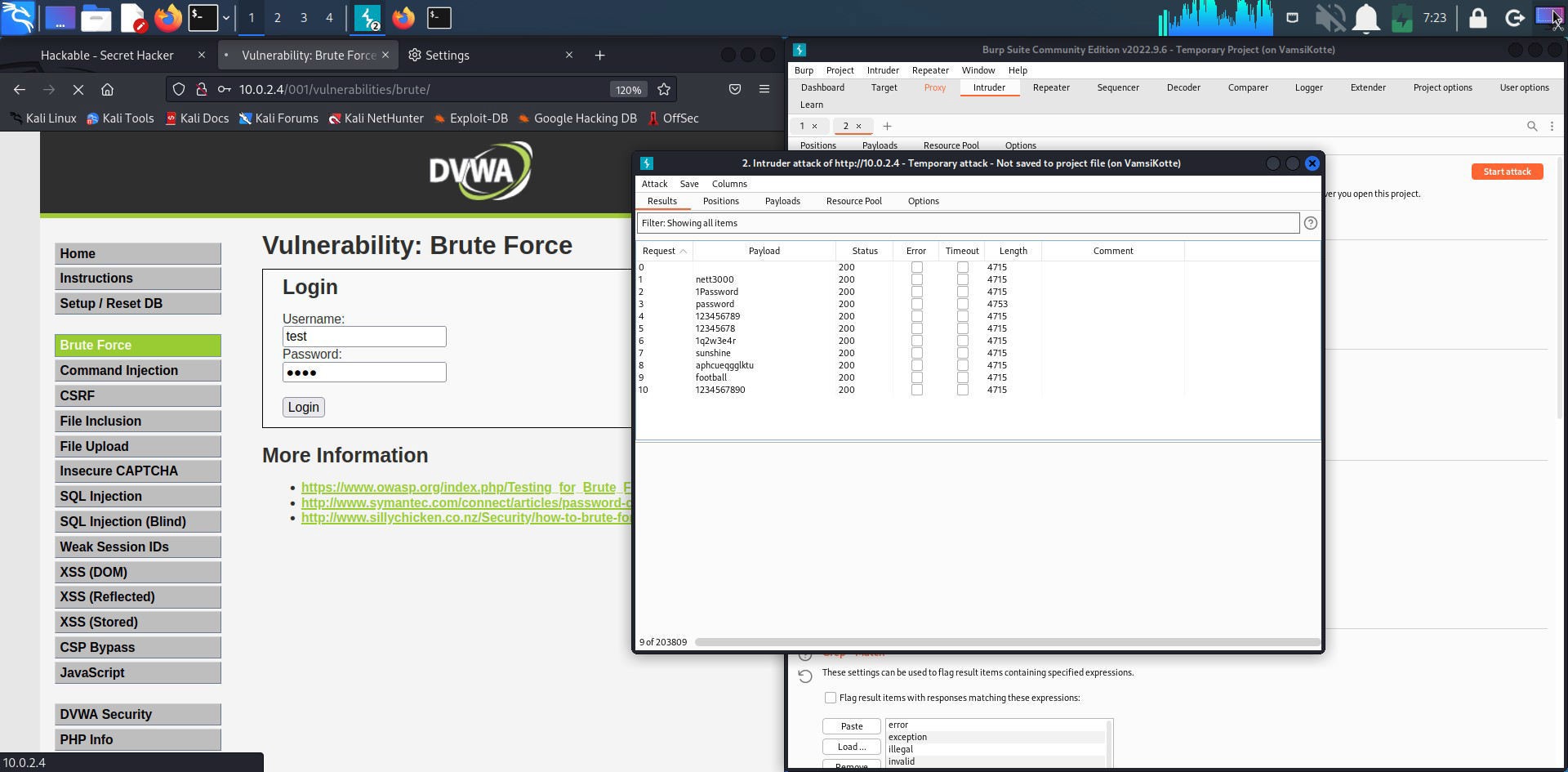
Step 5 .Now Add Position to password and change username to “admin”



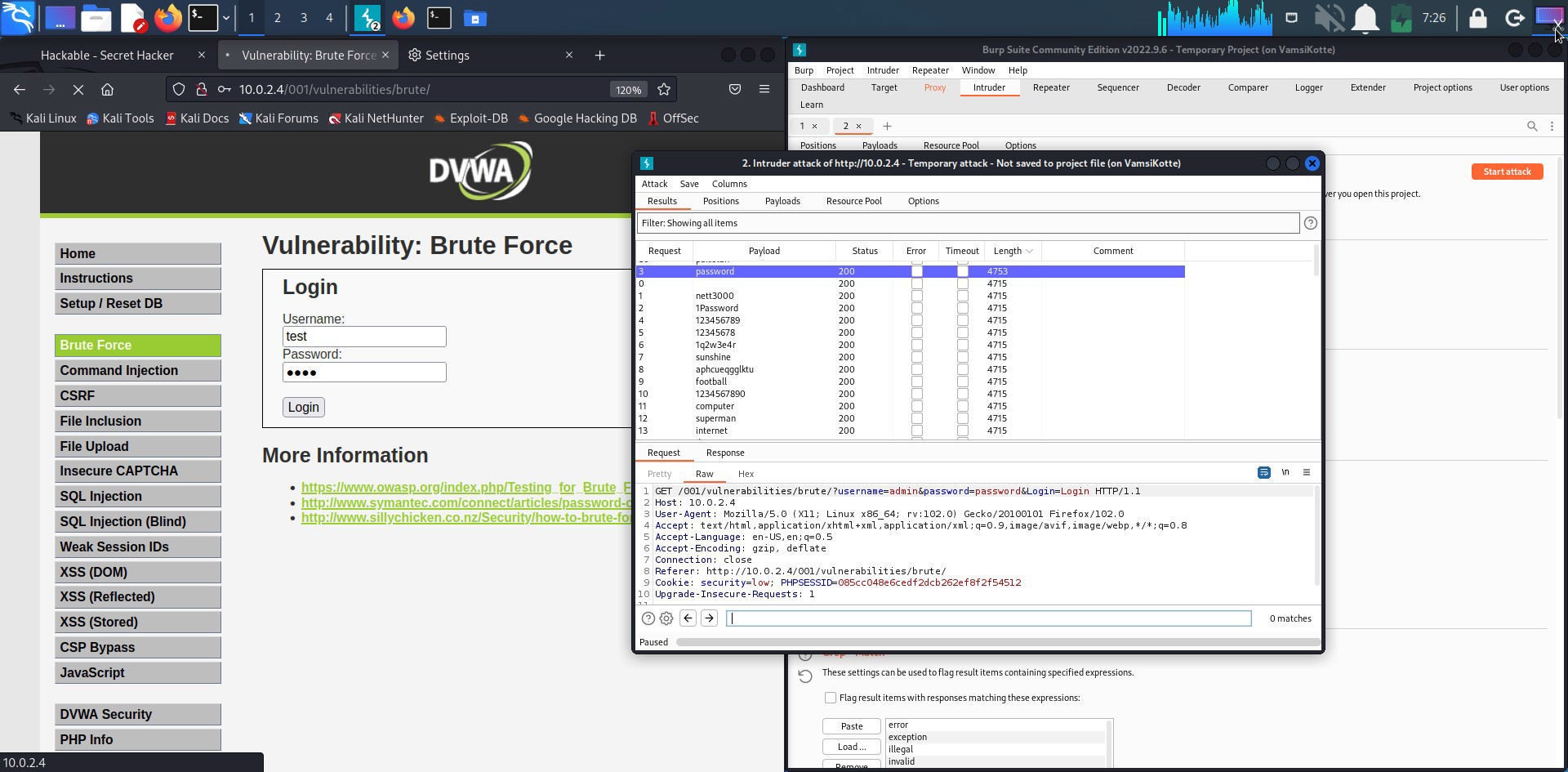
Step 6. load the payload wordlist. I used the kali default “/usr/share/dict/wordlist-probable.txt ”



Step 7. Start The Attack



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



#### 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.

1. 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.

1. Remote Code Execution

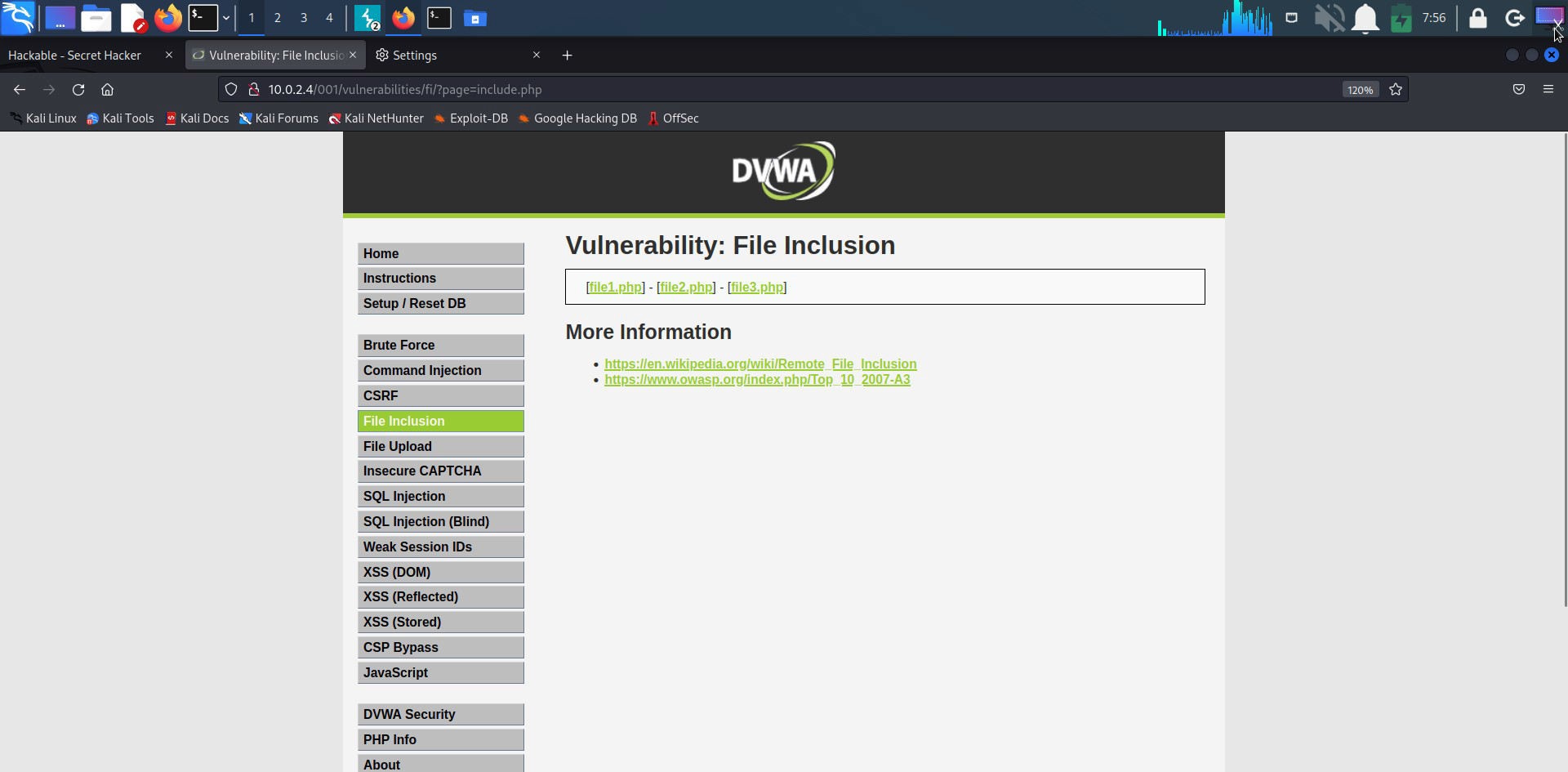
Combined with a file upload vulnerability, a Local File vulnerability can lead to remote code execution. In this case the attacker would use LFI to execute the unwanted file.

**Vulnerability Path** :- <http://10.0.2.4/001/vulnerabilities>

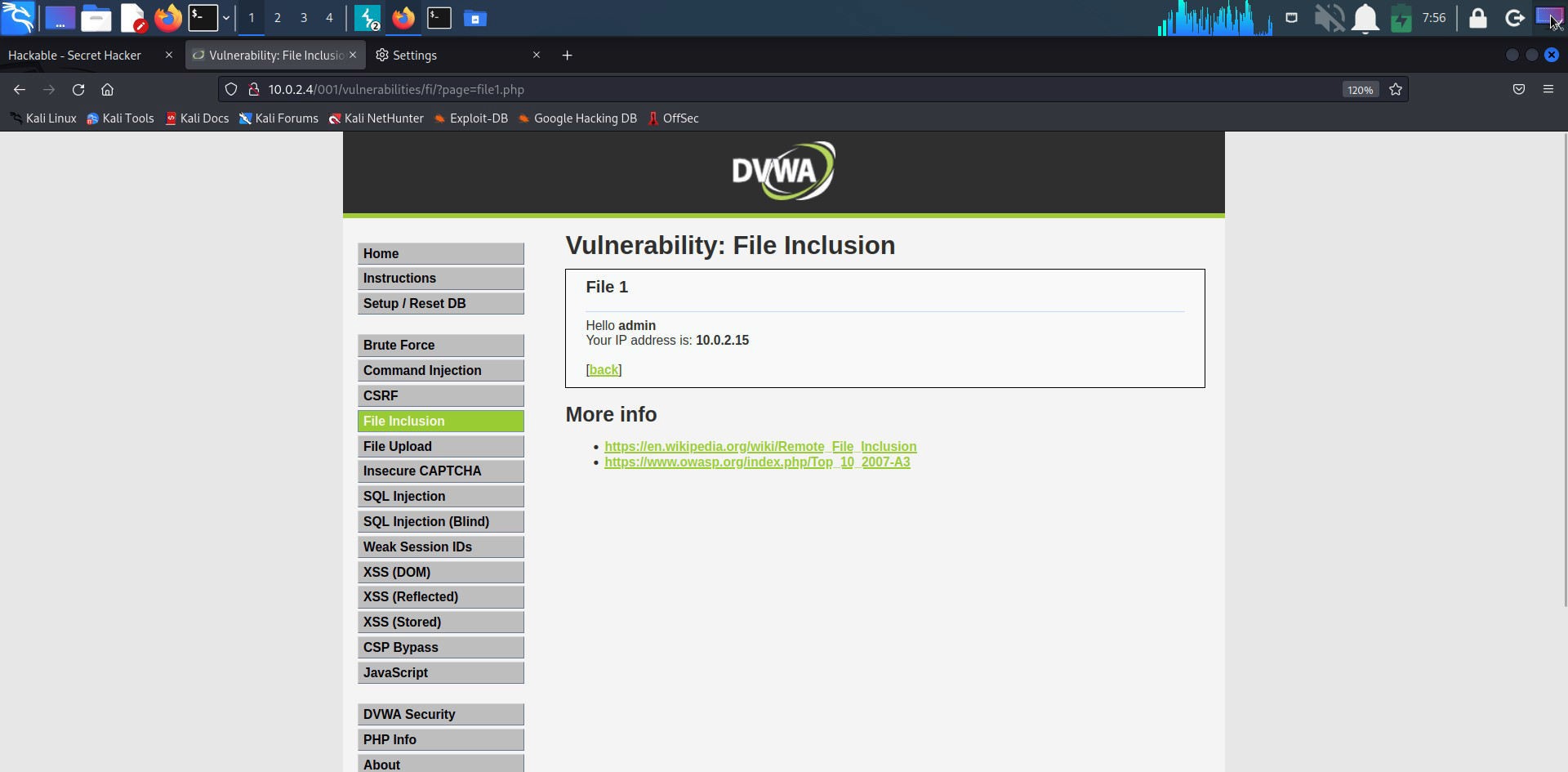
**Vulnerability Parameter**:- <http://10.0.2.4/001/vulnerabilities/fi/?page=include.php>

###### Steps to Reproduce :-

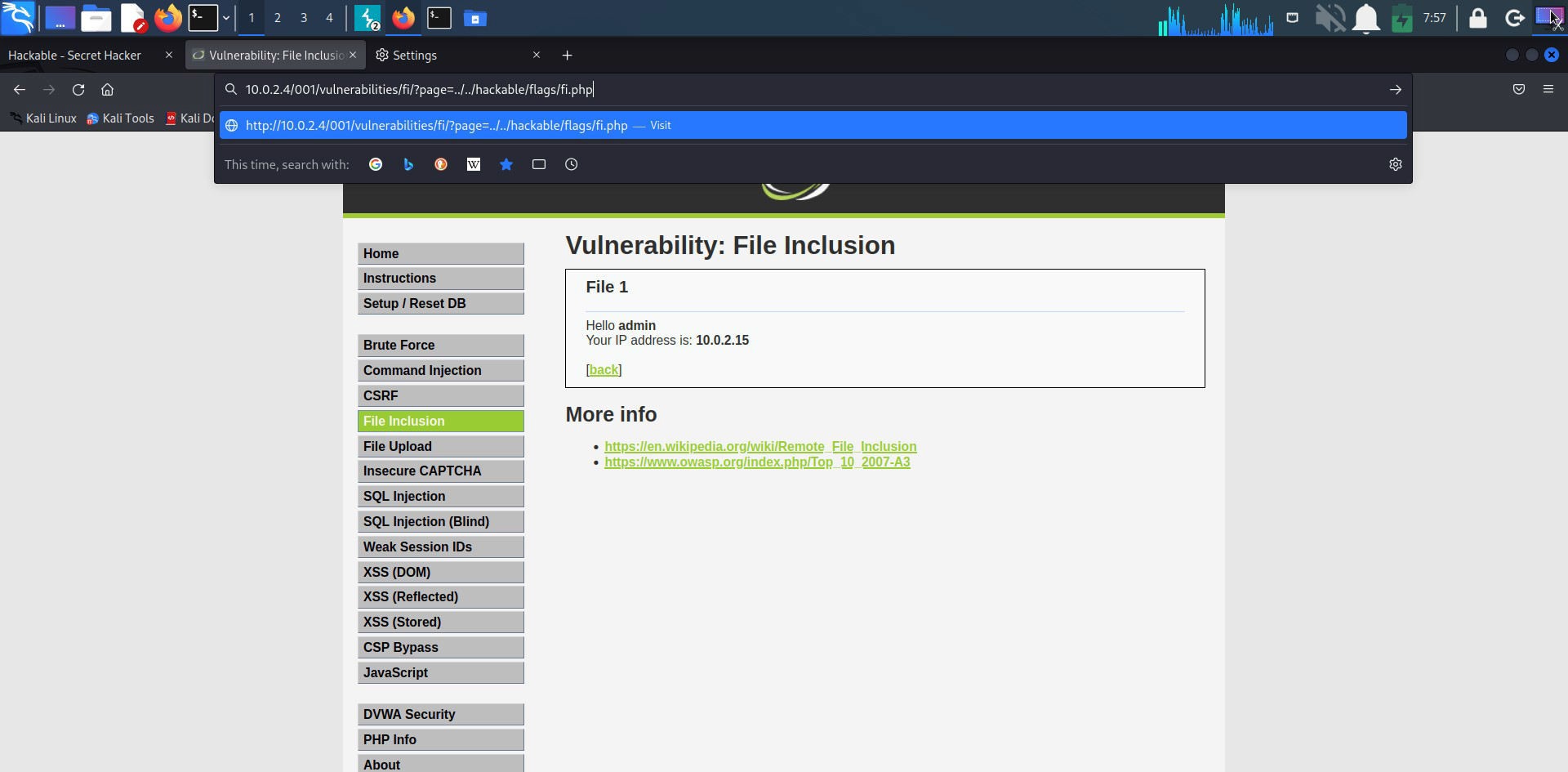
Step 1 . Go To Url Open Any one file Given.



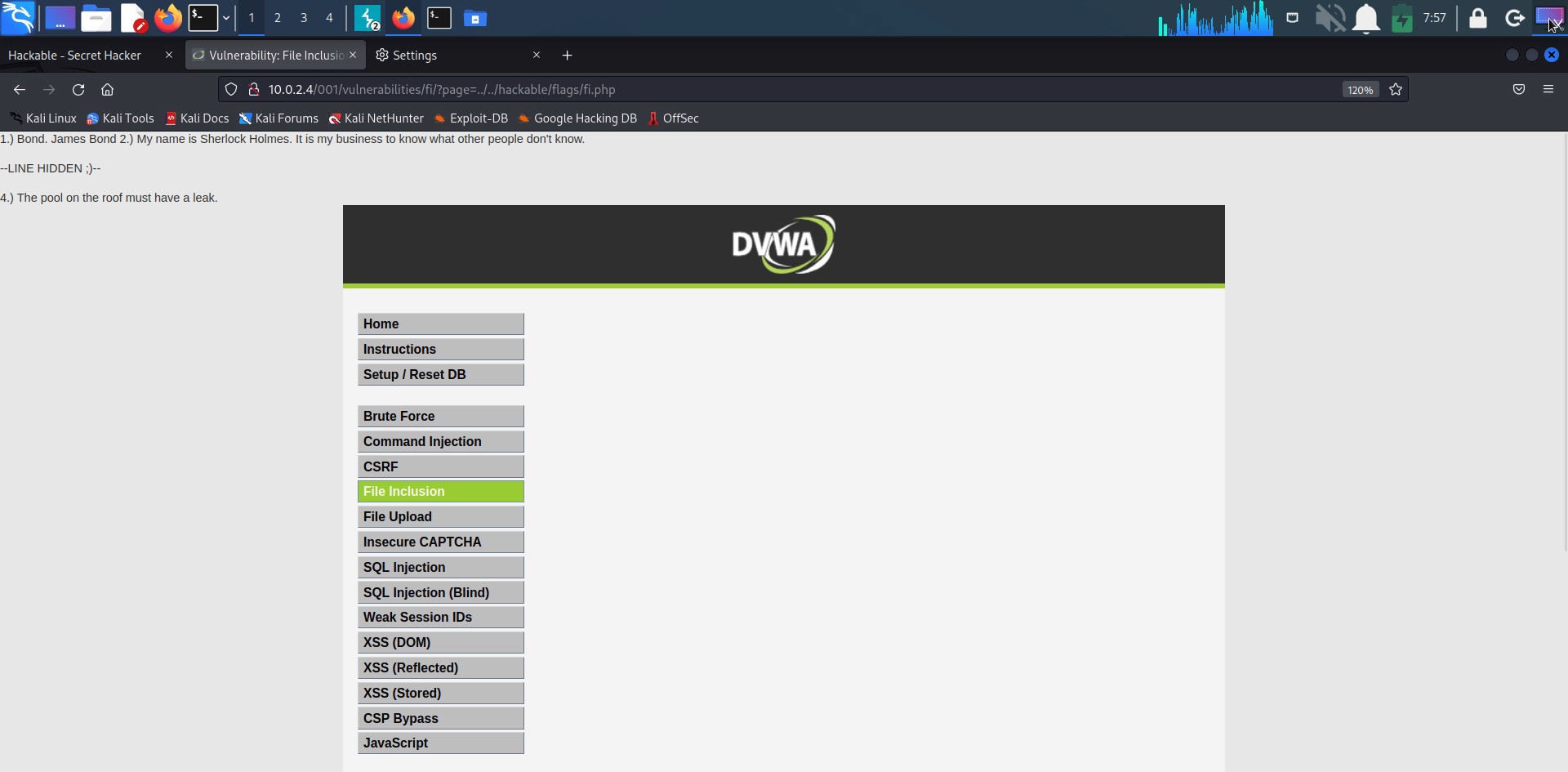
**Step 2. Observe The Url.**



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



Step 4. we opened an unauth-file



**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.