A logo with a book and a white circle

Description automatically generated

**FCIS Book**

**Penetration Testing Report**

Table of Contents

[1 Executive Summary 3](#_Toc151854723)

[2 Assessment Overview 4](#_Toc151854724)

[2.1 Planning 5](#_Toc151854725)

[2.2 Discovery 5](#_Toc151854726)

[2.3 Attack 5](#_Toc151854727)

[2.4 Reporting 5](#_Toc151854728)

[3 Finding Severity Ratings 6](#_Toc151854729)

[4 Scope of Engagement 7](#_Toc151854730)

[4.1 Targets 7](#_Toc151854731)

[4.2 Scope Exclusions 7](#_Toc151854732)

[5 Findings Table 8](#_Toc151854733)

[6 Technical Findings 9](#_Toc151854734)

# Executive Summary

In the heart of Ain Shams University's bustling campus, a group of computer science students were buzzing with excitement. Inspired by the tale of Mark Zuckerberg, they Start their mission to create their own social networking platform, "FCIS Book" exclusively for their fellow FCIS peers.

FCIS Book was designed to be a digital hub where students could connect, share updates, and engage with each other. It offered a variety of features,

including:

* User profiles: Students could create personal profiles to showcase their interests, hobbies, and academic achievements.
* Comment sections: Users could leave comments on posts, sparking discussions and fostering a sense of community.
* Search functionality: A search bar allowed users to easily find friends
* Profile picture uploads: Students could personalize their profiles by uploading their own profile pictures.

As FCIS Book grew in popularity, whispers of hidden vulnerabilities began to circulate among the student body. Tales of flags tucked away in unexpected corners and opportunities for unauthorized access piqued the curiosity of a group of tech-savvy individuals, the self-proclaimed "FCIS Vuln Hunters."

Driven by a passion for cybersecurity, the Vuln Hunters embarked on a quest to uncover the secrets of FCIS Book. They carefully analyzed the website's code, scrutinized its functionalities, and delved into the depths of its user interactions.

Their investigations led them to discover hints of potential vulnerabilities,

subtle clues that suggested hidden flaws within the platform's security. These observations fuelled their determination to uncover the truth.

The Vuln Hunters' persistence paid off. They discovered vulnerabilities that allowed them to manipulate user profiles, inject malicious code, and even gain unauthorized access to sensitive information. These discoveries highlighted the importance of robust security measures, user input validation, and access control mechanisms.

The FCIS Book development team left a trail of breadcrumbs for the Vuln Hunters to follow. Amidst their coding, they stored comments and reminders in an unencrypted file named "secret.txt". This file became a roadmap for the Vuln Hunters, revealing a cryptic message that signaled the presence of a hidden vulnerability. The Vuln Hunters were poised to uncover the platform's deepest secrets, armed with this newfound knowledge.

The FCIS Book development team's overreliance on the GET method for home page interactions proved to be their undoing. The Vuln Hunters, armed with their knowledge of various HTTP methods and web proxies, exploited the home page's weaknesses, uncovering hidden vulnerabilities and potential entry points for unauthorized access.

The FCIS Book Vuln Hunters' relentless pursuit led them to the super user's password, a coveted credential that unlocked the platform's core functionalities. Empowered with this newfound power, they could delve deeper into the system, uncovering hidden secrets and potential vulnerabilities. Their journey had transformed them from mere students into cybersecurity enthusiasts, ready to safeguard their digital realm

Unfazed by the "super" user's claims of invincibility, the FCIS Book Vuln Hunters embarked on a mission to crack his password, a challenge that would test their skills. Through persistent effort, they finally gained access to the account, only to discover a password far from "super" in strength. This discovery served as a humbling reminder to the "super" user and highlighted the importance of staying vigilant and implementing robust security measures.

The FCIS Book Vuln Hunters' final challenge was to breach another user's password, a feat that demanded both skill and perseverance. Through relentless attacks and cryptographic expertise, they cracked the code, gaining access to another user's account.

As the Vuln Hunters shared their findings with the FCIS Book development team, they hoped that their insights would contribute to the platform's overall security and strengthen the trust of its users.

Their journey through the vulnerabilities of FCIS Book served as a testament to their curiosity, ingenuity, and unwavering determination to protect their digital realm.

This report presents the output of an internal penetration test that was conducted against FCIS book. It is recommended to fix the found vulnerabilities.

The assessment showed:

***Seven*** Easy vulnerabilities.

In addition to, ***FIVE*** Medium vulnerabilities

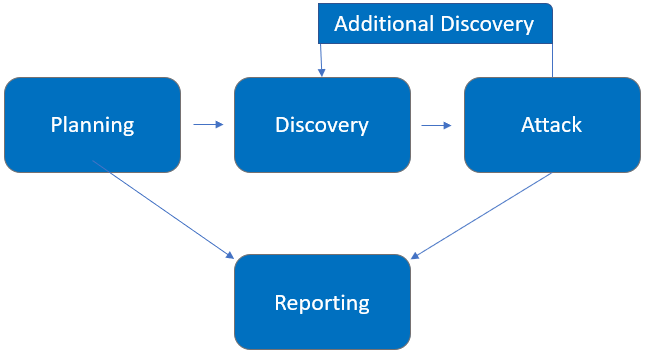
And ***FIVE*** Bonus vulnerabilities

# Assessment Overview

From Dec 2nd, 2023 to Dec 17th, 2023, FCIS book engaged You to evaluate the security posture of its infrastructure compared to current industry best practices that included an external penetration test. All testing performed is based on the NIST *SP 800-115 Technical Guide to Information Security Testing and Assessment, OWASP Testing Guide (v4), and customized testing frameworks*.

Phases of penetration testing activities include the following:

* Planning – Customer goals are gathered and rules of engagement obtained.
* Discovery – Perform scanning and enumeration to identify potential vulnerabilities, weak areas, and exploits.
* Attack – Confirm potential vulnerabilities through exploitation and perform additional discovery upon new access.
* Reporting – Document all found vulnerabilities and exploits, failed attempts, and company strengths and weaknesses.



## Planning

Planning in penetration testing involves collaboratively defining the scope, goals, and guidelines for the test. This phase establishes the boundaries of the assessment, such as which systems are included and the desired objectives—whether it's uncovering vulnerabilities, testing response procedures, or assessing controls. Ethical and legal considerations are addressed, resources are allocated, and a communication plan is set for a successful testing engagement. This strategic groundwork ensures alignment between the testing team and the client, leading to a well-structured and effective penetration test.

## Discovery

* Information Gathering: Collect relevant information about the internal infrastructure, including IP ranges, domain names, and network architecture.
* Open Source Intelligence (OSINT): Utilize publicly available sources to gather additional information about employees, technology stack, and potential weak points.
* Vulnerability Scanning: Perform automated vulnerability scans on the identified systems to uncover known security vulnerabilities.
* Network Mapping: Create a comprehensive map of the internal network, identifying live hosts, open ports, and services.

## Attack

* Manual Vulnerability Assessment: Conduct an in-depth manual assessment of the identified vulnerabilities to validate their severity and potential impact.
* Exploitation: Attempt to exploit the identified vulnerabilities to determine their feasibility and potential impact on the internal systems.
* Privilege Escalation: If necessary, attempt to escalate privileges to gain deeper access into the systems and network.
* Lateral Movement: Explore the internal network for lateral movement opportunities, simulating an attacker's attempt to pivot within the environment.
* Data Exfiltration (If Agreed Upon): Simulate the extraction of sensitive data from the internal systems to demonstrate potential data breaches.

## Reporting

Reporting the findings of the penetration tests is integral to the fulfilment of the previously mentioned strategic motivations and driving forces behind engaging in such a process. Hence, once the above tasks are completed, a documentation scheme is followed to report the results across different levels including technical and management levels. The report is going to be focused on the real risk behind the findings to maximize business value. Also, whenever applicable, a detailed recommendation is included on how to fix the leveraged vulnerabilities and / or minimize their threat.

# Finding Severity Ratings

The following table defines levels of severity and corresponding CVSS score range that are used throughout the document to assess vulnerability and risk impact.

|  |  |  |
| --- | --- | --- |
| **Severity** | **CVSS V3**  **Score Range** | **Definition** |
| Critical | 9.0-10.0 | Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately. |
| High | 7.0-8.9 | Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible. |
| Moderate | 4.0-6.9 | Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved. |
| Low | 0.1-3.9 | Vulnerabilities are non-exploitable but would reduce an organization’s attack surface. It is advised to form a plan of action and patch during the next maintenance window. |
| Informational | N/A | No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation. |

**Risk Factors**

Risk is measured by two factors: Likelihood and Impact:

**Likelihood**

Likelihood measures the potential of a vulnerability being exploited. Ratings are given based on the difficulty of the attack, the available tools, attacker skill level, and client environment.

**Impact**

Impact measures the potential vulnerability’s effect on operations, including confidentiality, integrity, and availability of client systems and/or data, reputational harm, and financial loss.

# Scope of Engagement

The scope of this engagement includes performing Internal Penetration Testing for

FCIS book.

## Targets

|  |  |
| --- | --- |
| **Assessment** | **Details** |
| Internal Penetration Test | Local host |

## Scope Exclusions

Per client request, you will not perform any of the following attacks during testing:

• Denial of Service (DoS)

• Phishing/Social Engineering

All other attacks not specified above were permitted by FCIS book.

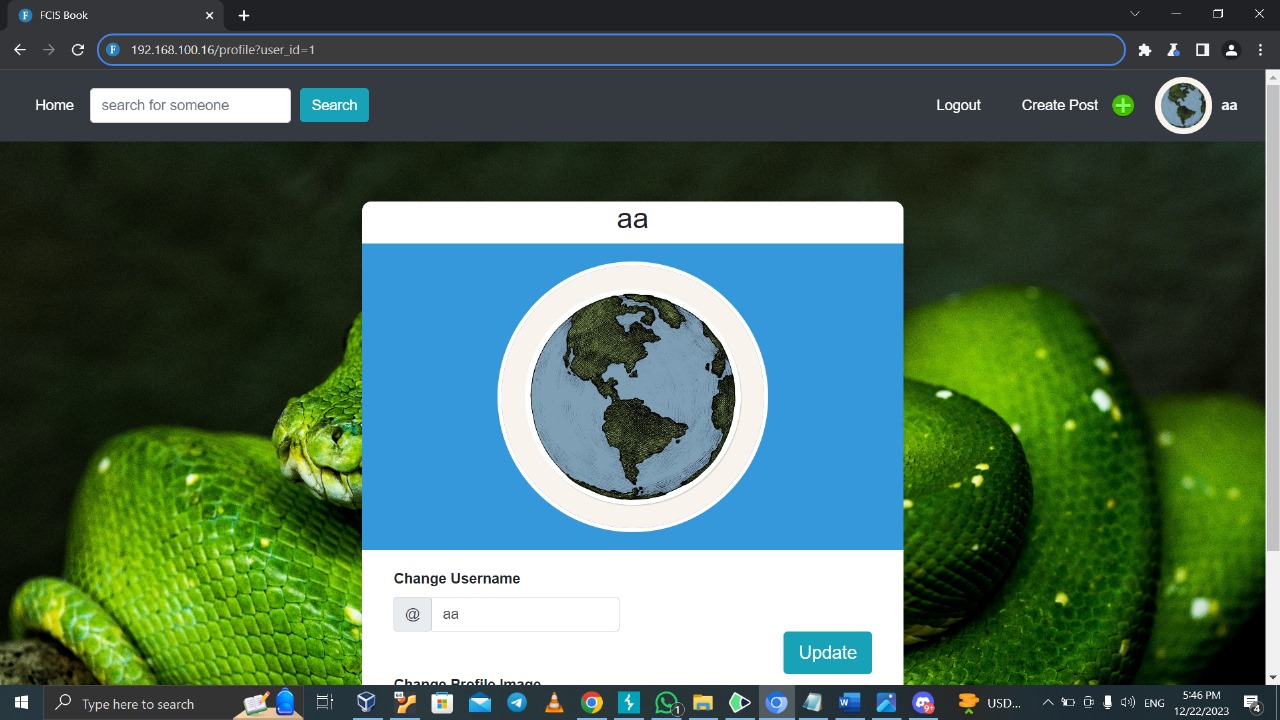
# Findings Table

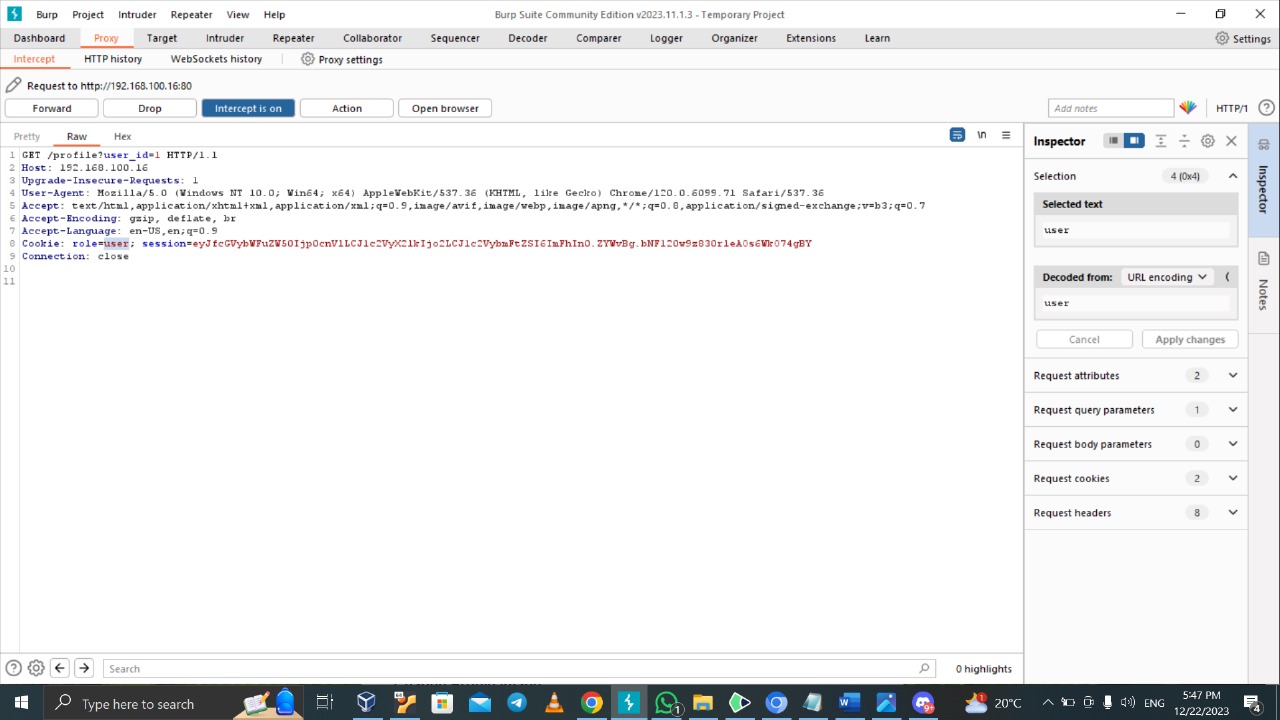
|  |  |
| --- | --- |
| **Finding** | **Risk** |
| Remote Code Execution on | High |
| Outdated Software version at | Medium |
| Server Header information at | Low |

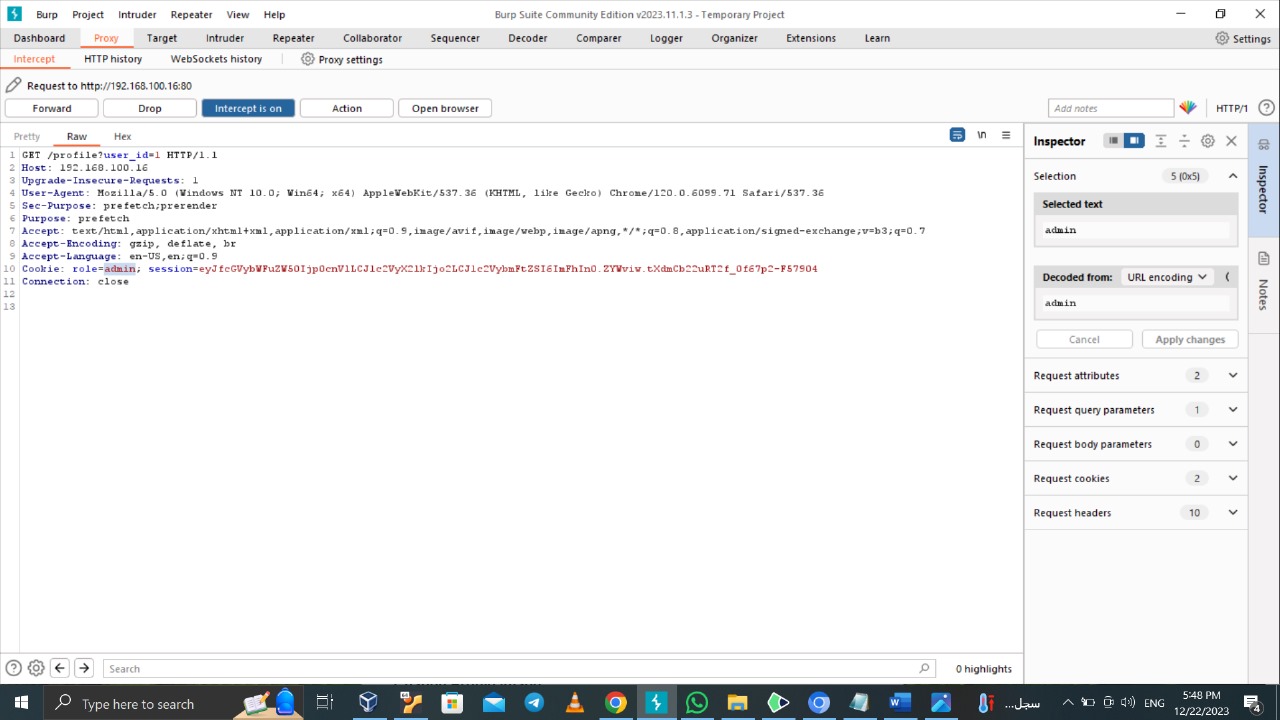
# Technical Findings

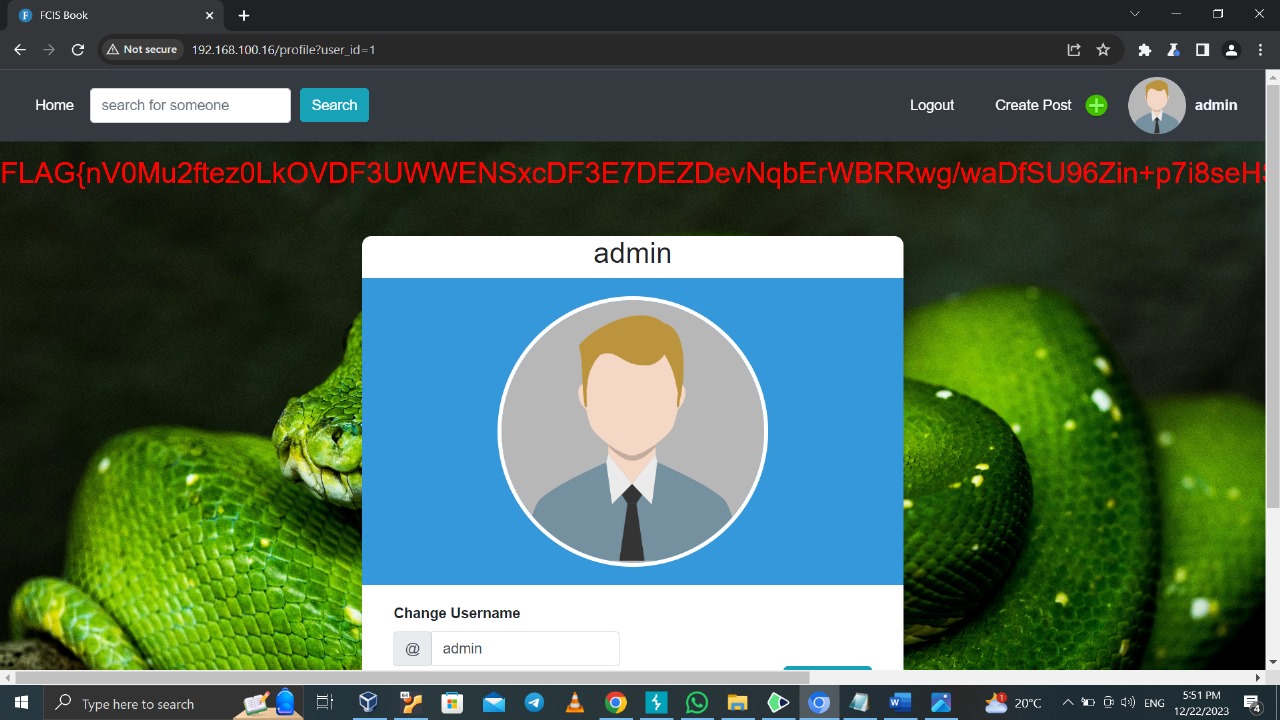
**1-Broken access control**

|  |  |
| --- | --- |
| **Description** | * Broken access control is one of important vulnerabilities * it could be vertical or horizontal * We find and apply vertical access control , we login with our account and make intercept on so we find on the request (something) so we change it to admin and it make us have the admin access |
| **Recommendation** | * Access control checks should always be performed on the server side, so any coming from client side should be treat as dangerous * control checks instead of role-based access control. |
| **Affected system** | It affects CIA   * Confidentiality: attacker can view admin page * Integrity: attacker can update as admin * Availability: attacker can delete users |
| **Threat level** | High |



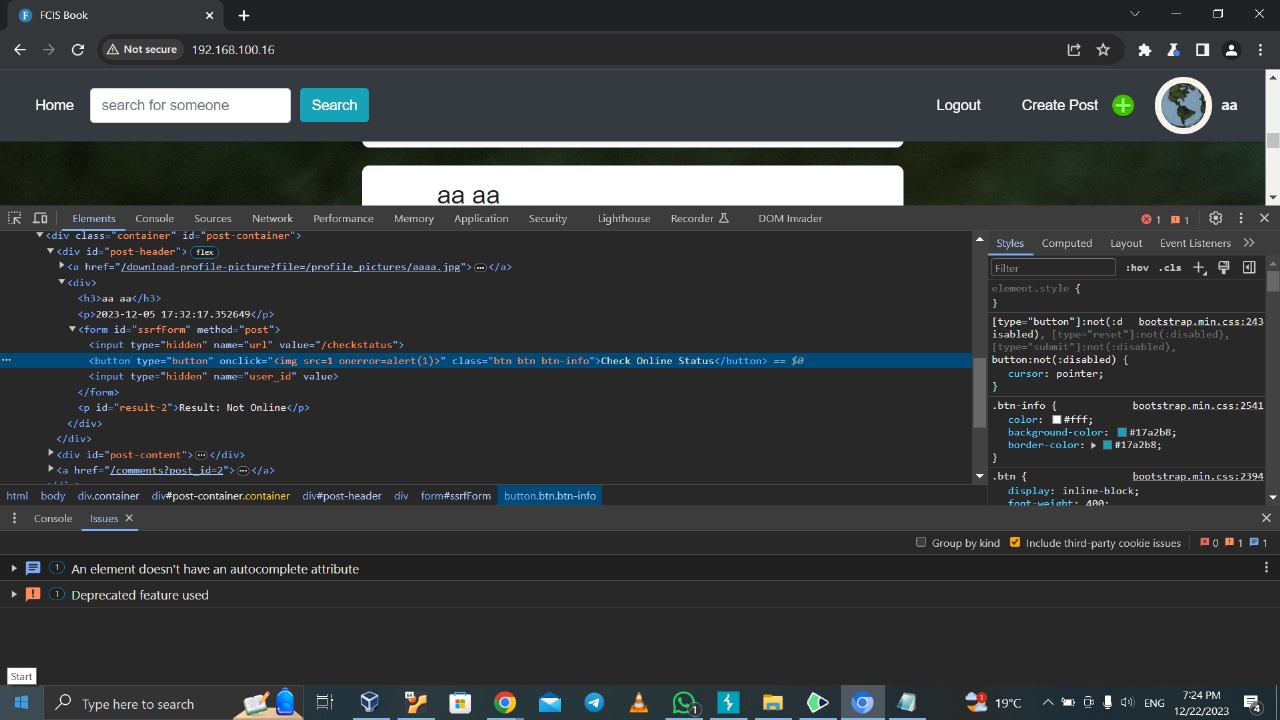


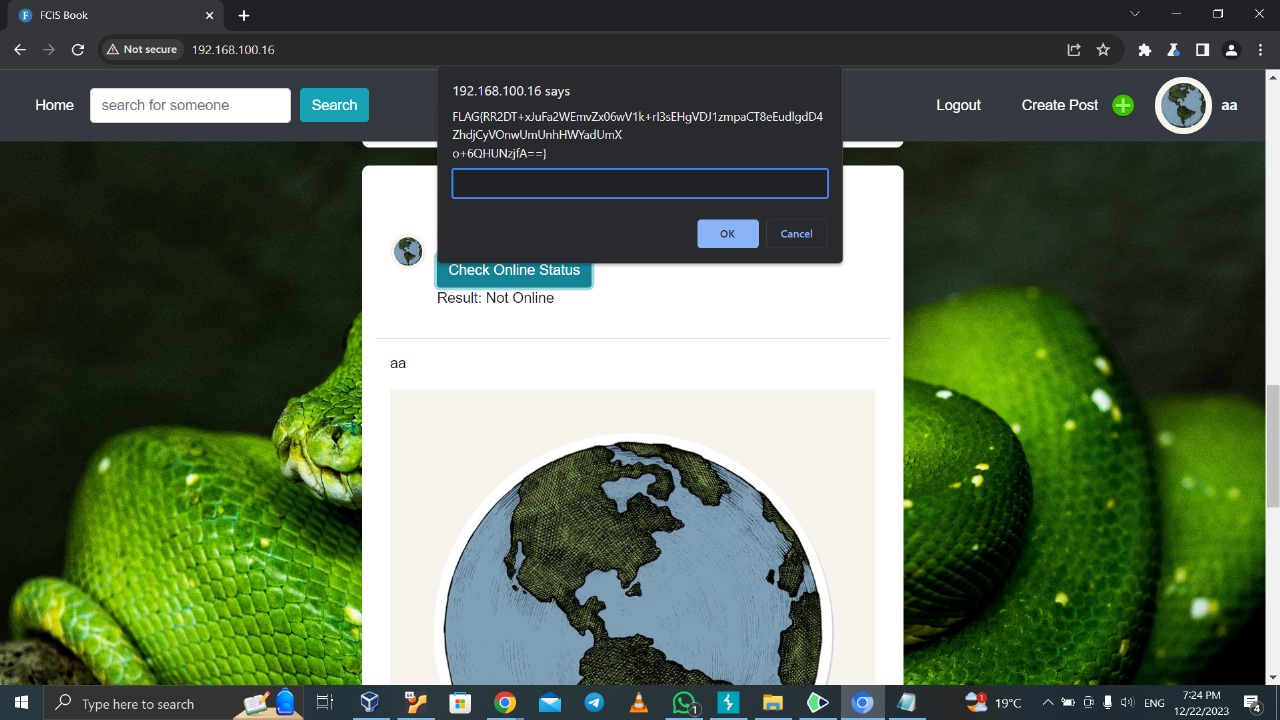


****

**2-XSS(DOM)**

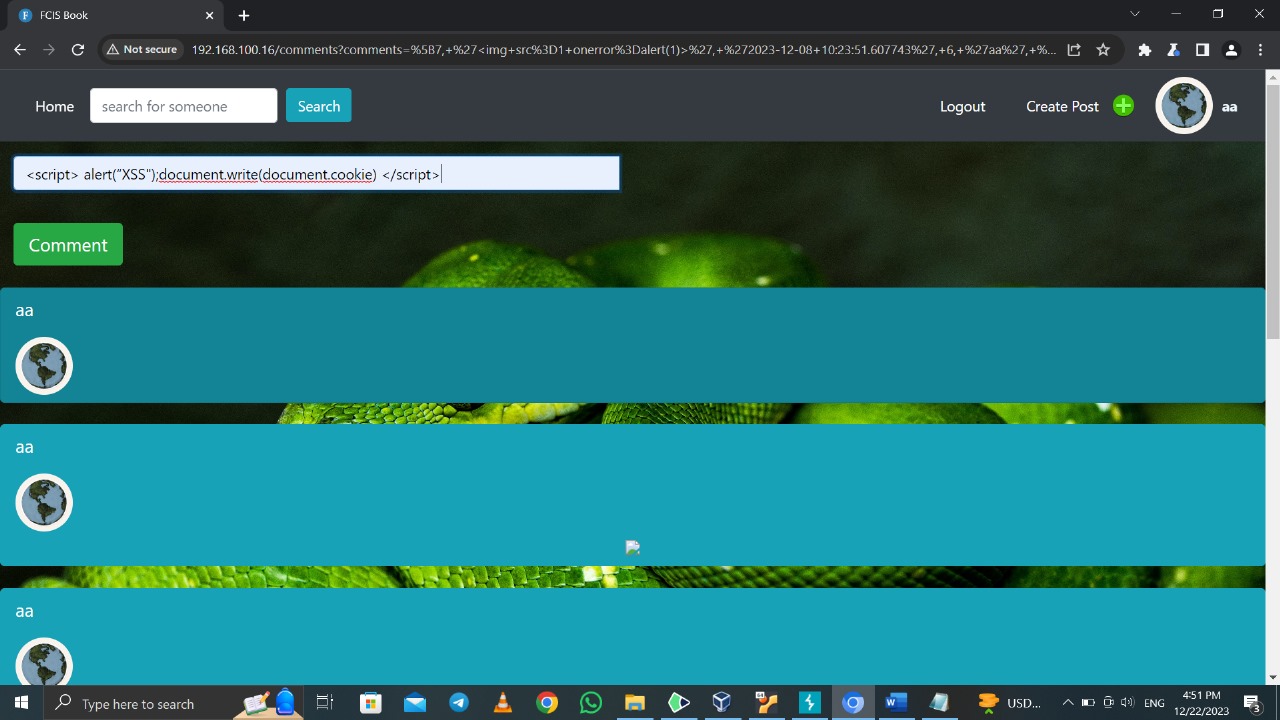
|  |  |
| --- | --- |
| **Description** | * XSS have several types we concern here DOM * It is exploiting a script inserted in input field or by other way, so it could change the content and behaviour of web page add or remove element or change the content * We open inspect and we show on check status button and we insert on click java script code and it show alert |
| **Recommendation** | * Encode any user-controllable data on output. * Filter input on arrival.Only accept input based on what is expected or valid. |
| **Affected system** | Search |
| **Threat level** | Moderate |

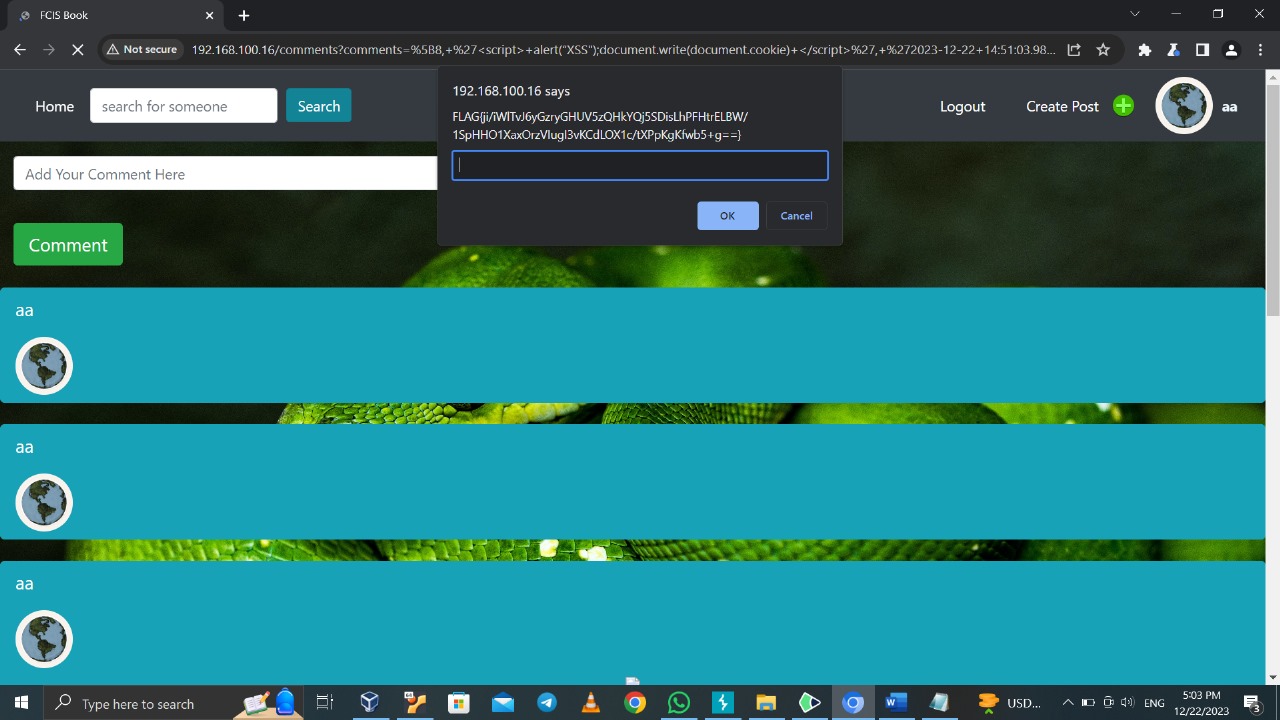




**3-XSS (stored)**

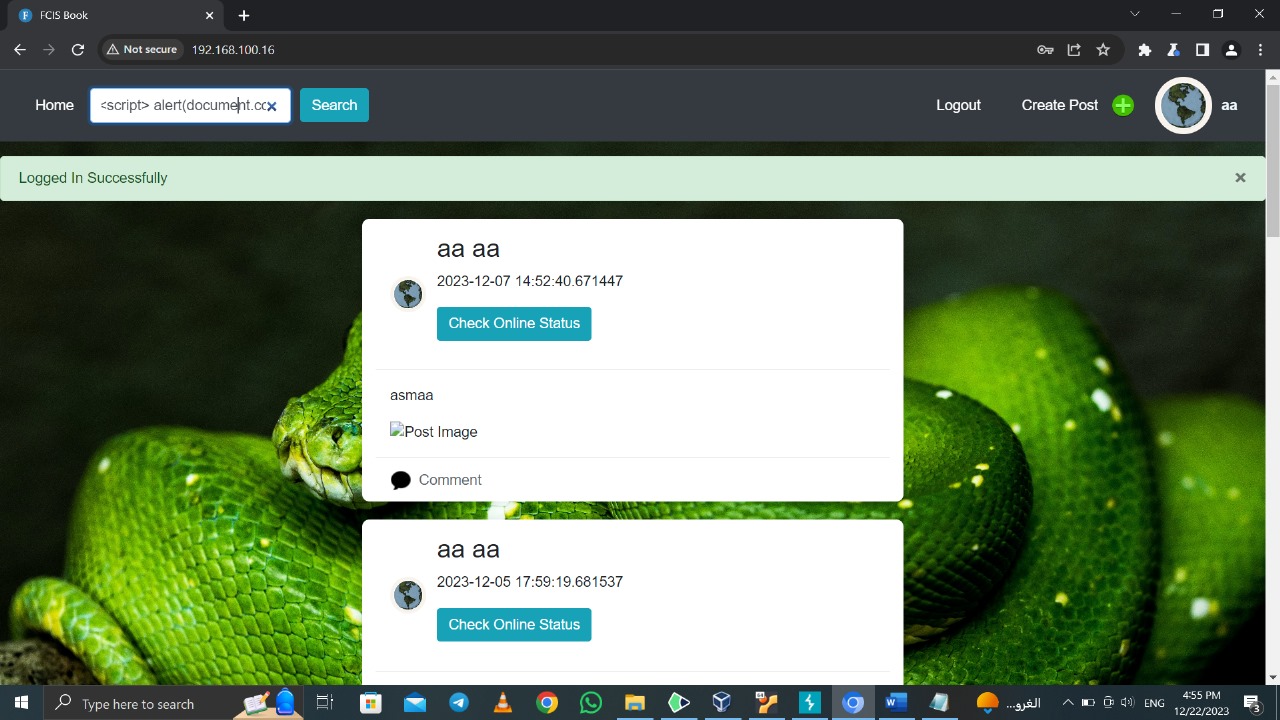
|  |  |
| --- | --- |
| **Description** | * As we mention on [vulnerability 2](#_Technical_Findings) the XSS vulnerability and its types * we try to do stored XSS by using java script code it just alert on comment and it is done. |
| **Recommendation** | * Encode any user-controllable data on output. * Filter input on arrival.Only accept input based on what is expected or valid. |
| **Affected system** | * The vulnerability affects the client-side of the web application, particularly the comment functionality * It could effect on **Confidentiality** |
| **Threat level** | High |

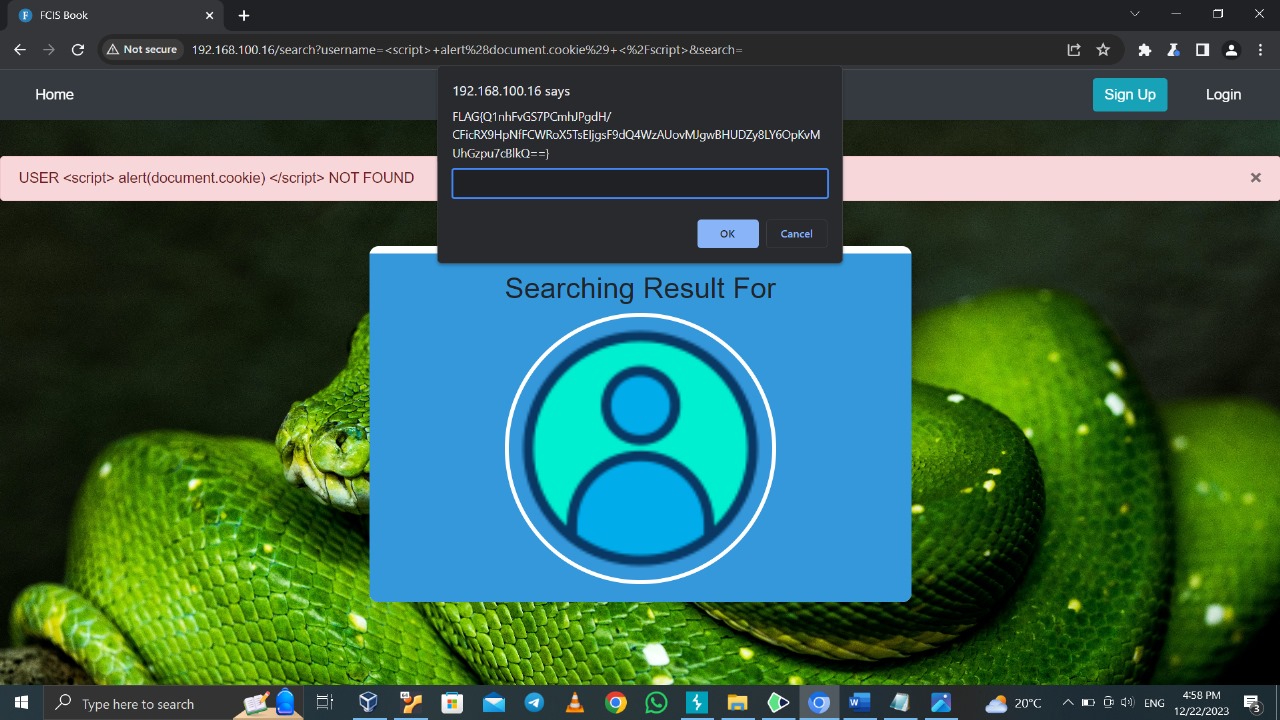




**4- XSS (reflected)**

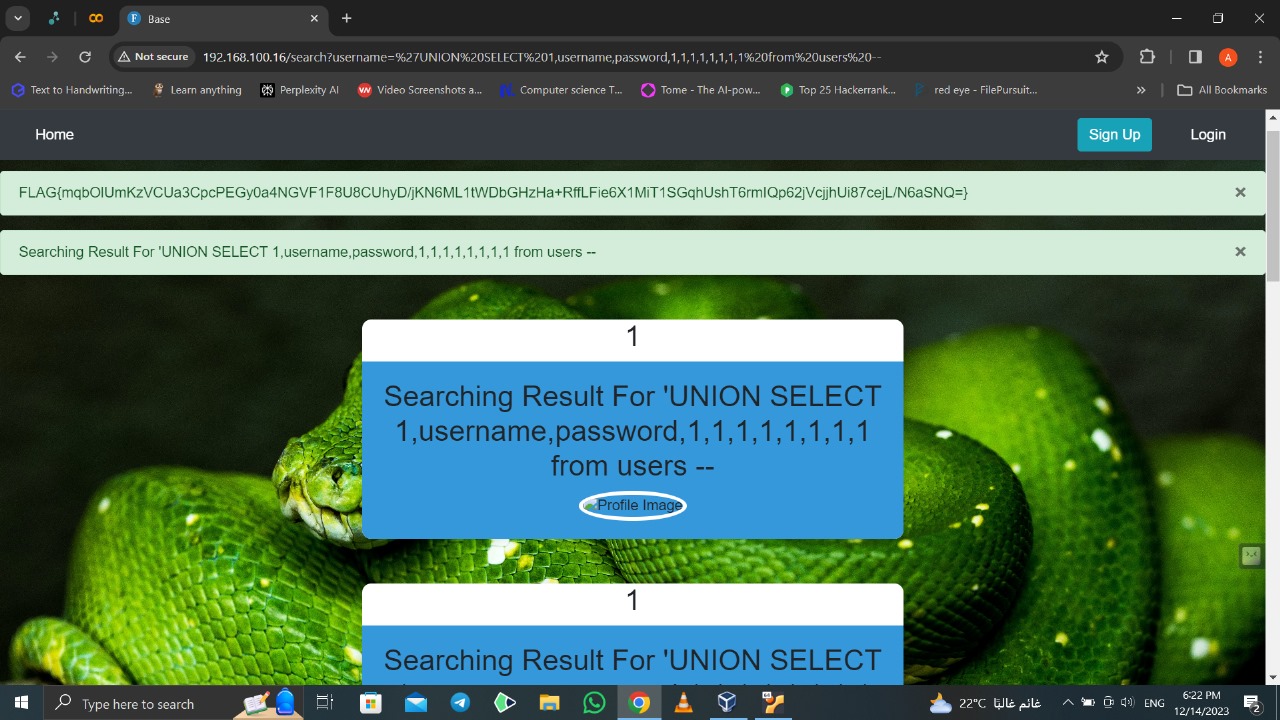
|  |  |
| --- | --- |
| **Description** | * We concern here on reflected on XSS we try to insert alert on search and it reflect |
| **Recommendation** | * Encode any user-controllable data on output. * Filter input on arrival.Only accept input based on what is expected or valid. |
| **Affect systems** | * The vulnerability affects the client-side of the web application, particularly the search functionality. |
| **Threat level** | Moderate |





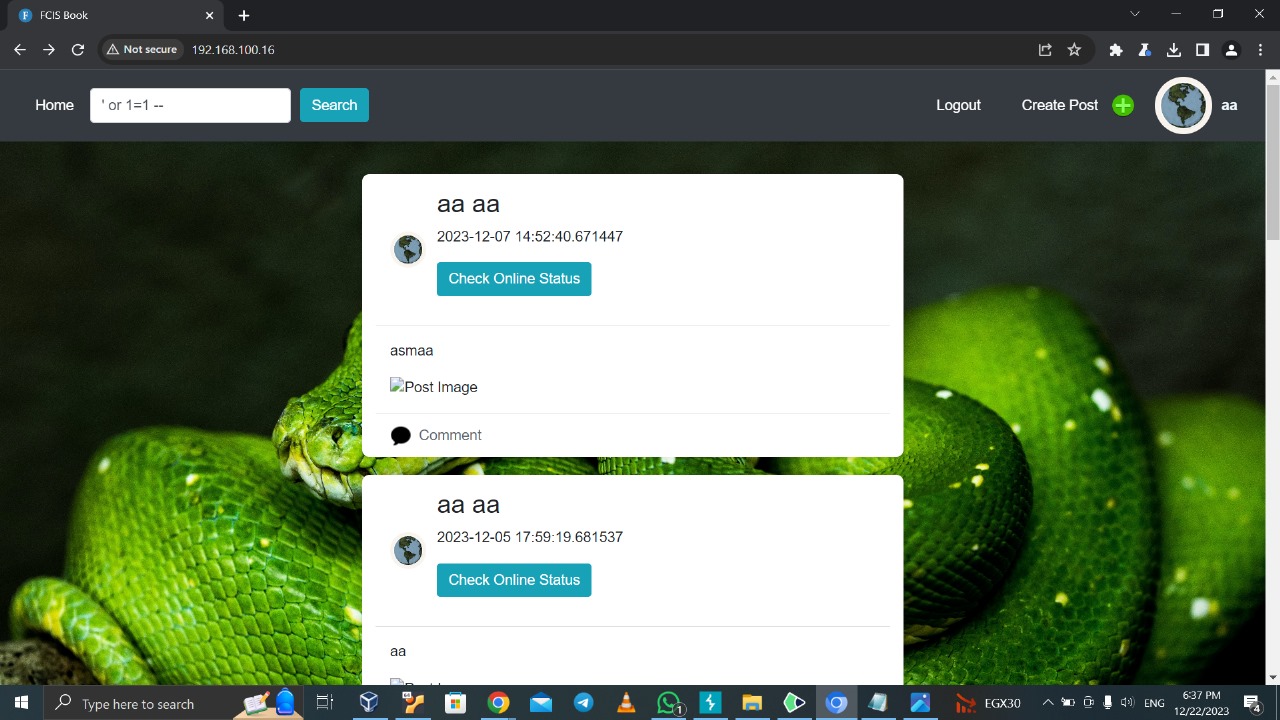
**5-SQL injection (Union and error)**

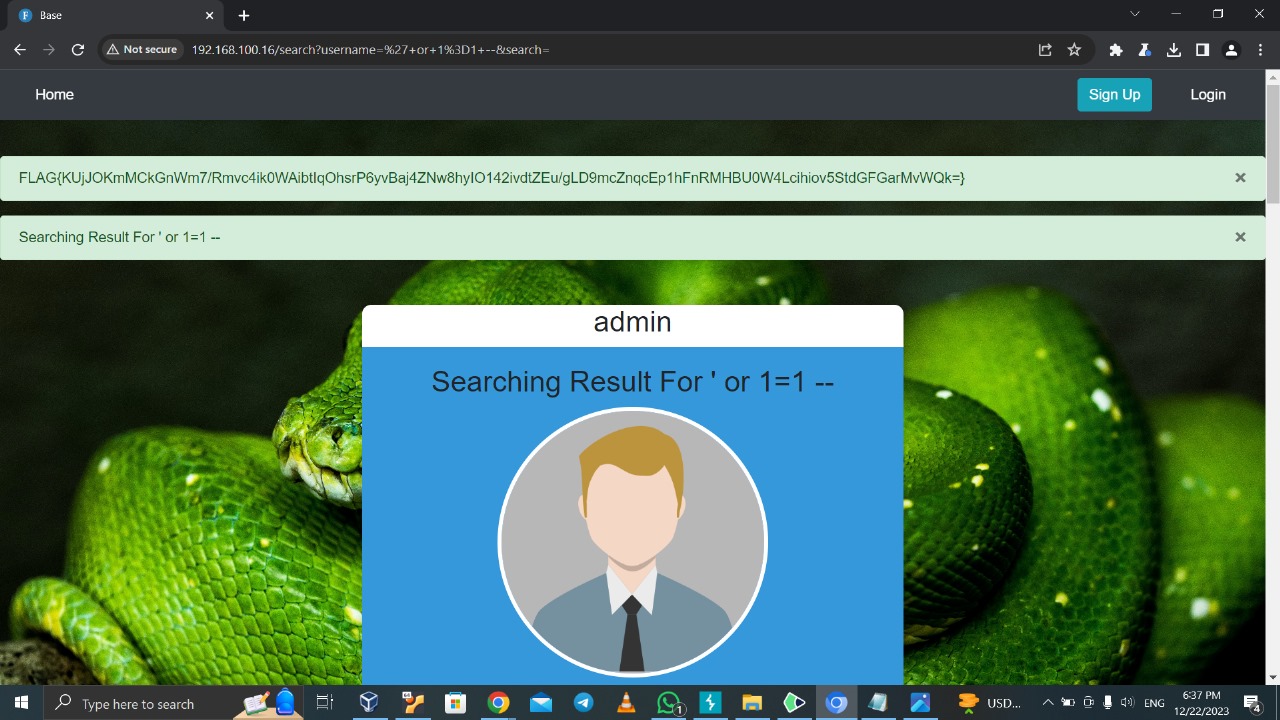
|  |  |
| --- | --- |
| **Description** | * SQL injection is one of important vulnerabilities * It inserts SQL code so it retrieves data from database. * It has several types in band, inferential and out of band. * We search on a several users we notice in URL something on database (username =) so we insert single quote to test if a message error as response have information we can use and the error message have information on database specifically table name so we apply here**(SQL injection error based),** so we know the name of columns from comments and we insert SQL command (UNION) and select columns username and password and we insert placeholders to make union command right |
| **Recommendation** | * Make validations prevent insert SQL commands * make errors general and do not have information about database that could be used |
| **Affected system** | User information, it effects on:   * confidentiality as attacker could view usernames and passwords, * it also could affect integrity as we could alter columns * availability we could also delete tables |
| **Threat level** | Critical |



**6-SQL (inferential)**

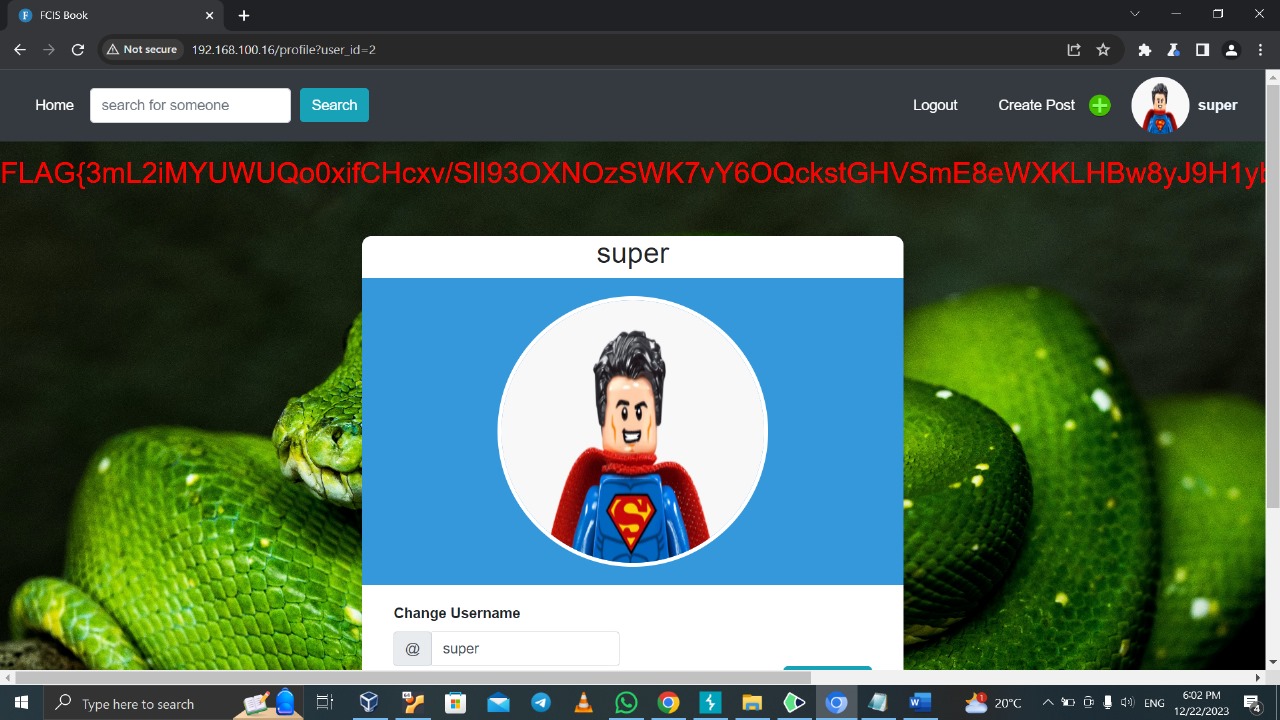
|  |  |
| --- | --- |
| **Description** | * Attacker send a query for example if user is existing on database or not and depend on it the response will be reflect on the page * We insert SQL command on search (‘ or 1==1 --) |
| **Recommendation** | * Make validations prevent insert SQL commands |
| **Affected systems** | * Database and the system * It effects **confidentiality**: we can bypass authentication and gain unauthorized access * It effects **integrity** and availability also |
| **Threat level** | High |





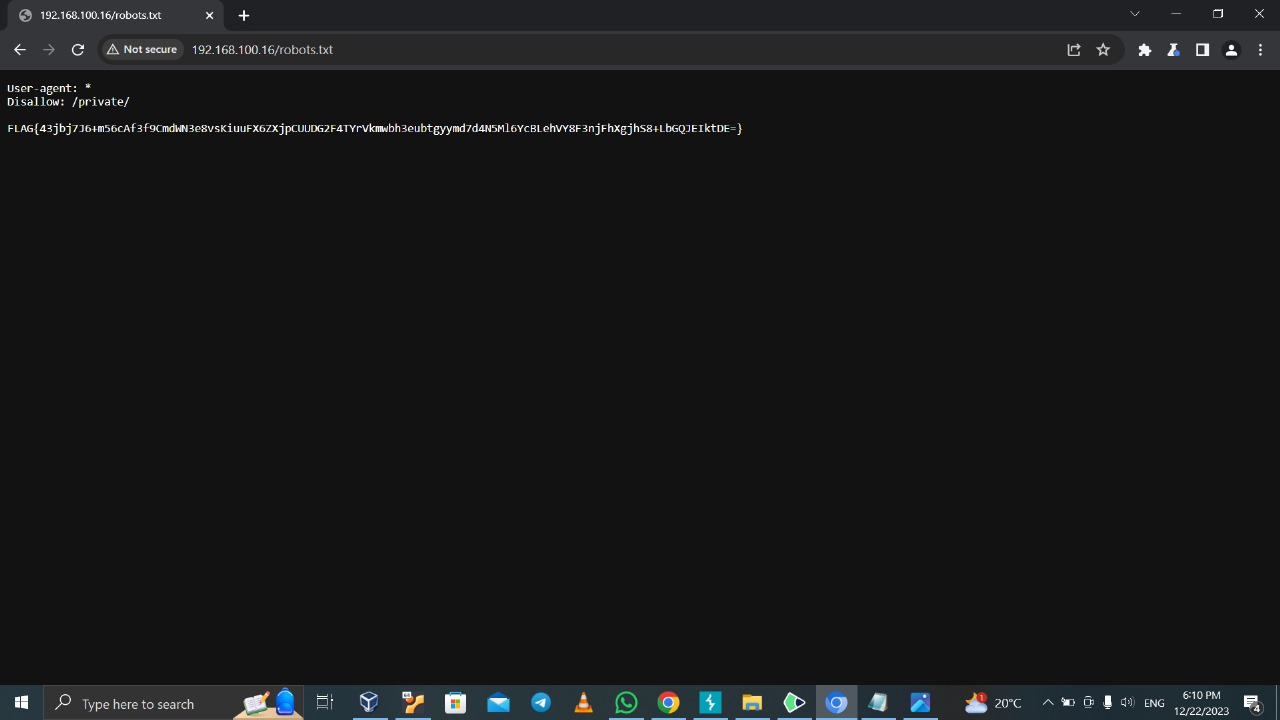
**7-IDOR**

|  |  |
| --- | --- |
| **Description** | * It is a type of access control vulnerability when application provide direct access to objects based in user supplied input. * We open our profile and notice on our URL we have (for example profile =1) so we change it to a several numbers and we access other accounts |
| **Recommendation** | * Prevent direct access to objects on database from URL |
| **Affect systems** | * Users accounts it effects on confidentiality as we can show other users profile |
| **Threat level** | High |

****

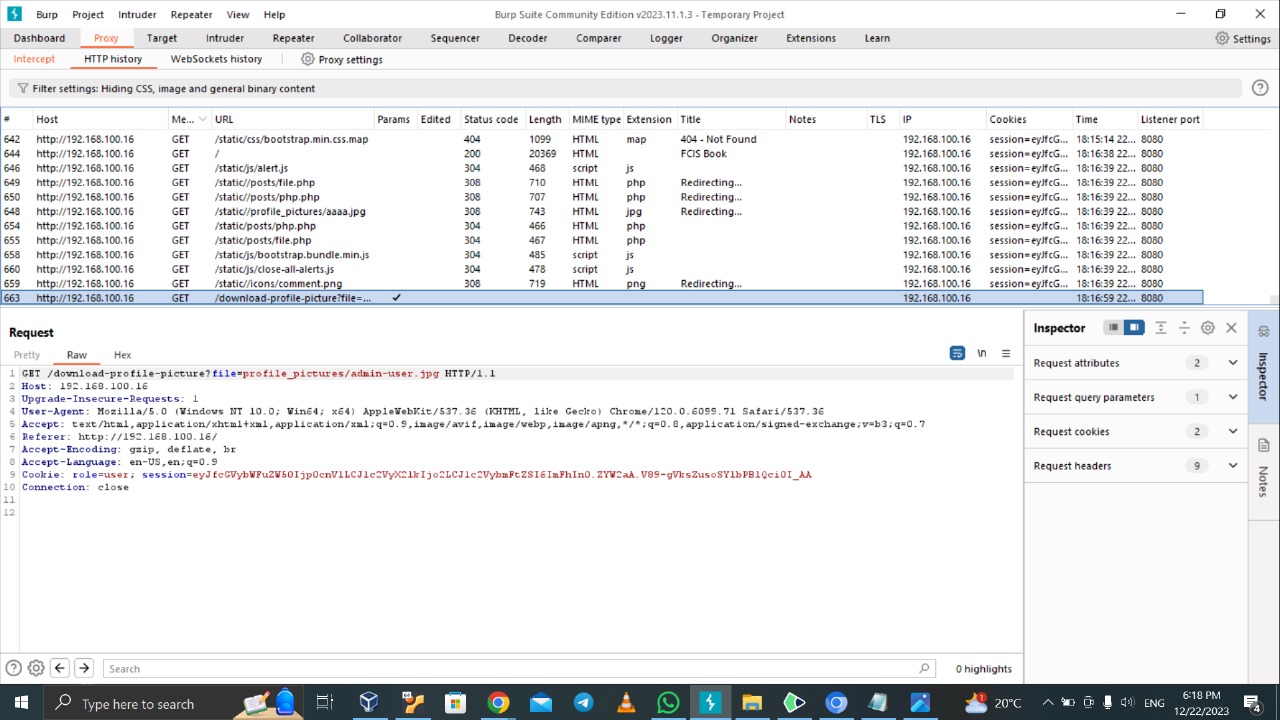
**8- Robots**

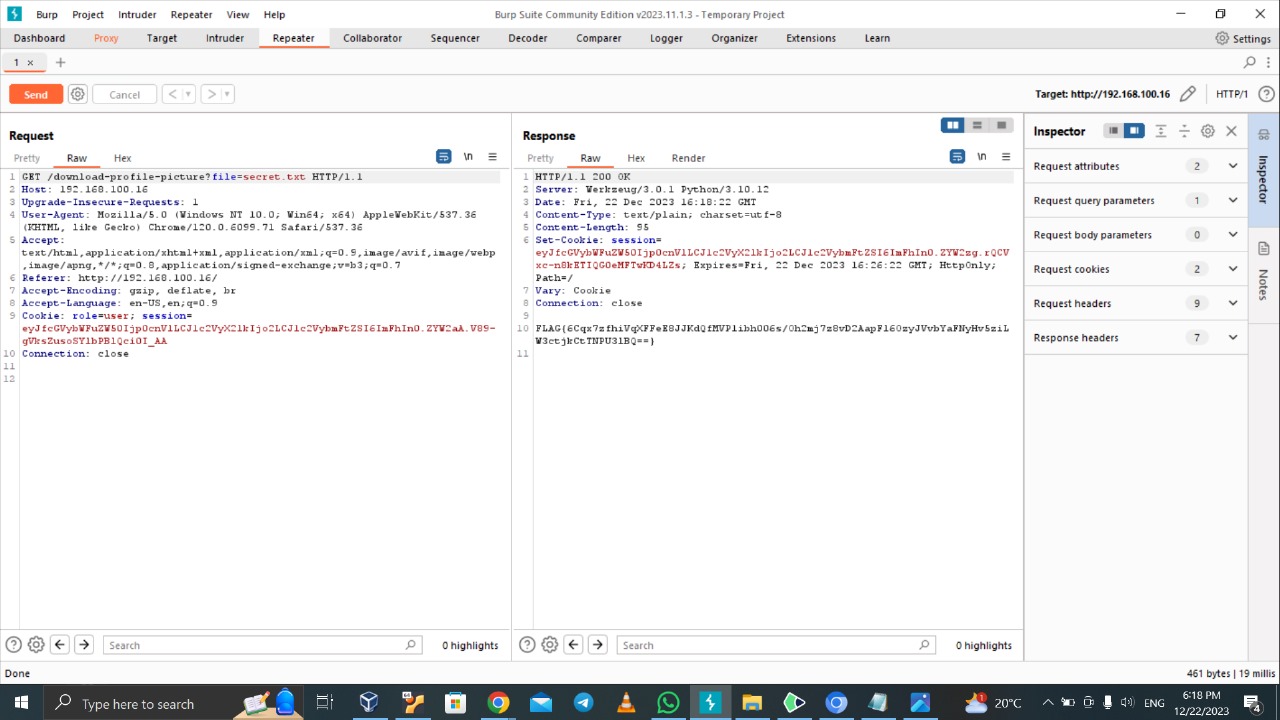
|  |  |
| --- | --- |
| **Description** | * The file robots.txt is used to give instructions to web robots, such as search engine crawlers, about locations within the web site that robots are allowed, or not allowed, to crawl and index. * The information in the file may therefore help an attacker to map out the site's contents, especially if some of the locations identified are not linked from elsewhere in the site. * We try to check if we could see robot.txt so after we login we insert on URL after our IP robots.txt and we show un allowed files |
| **Recommendation** | * enforce proper access control on robots.txt |
| **Affect systems** | * search engine indexing and web crawlers |
| **Threat level** | Low |



**9- Directory traversal**

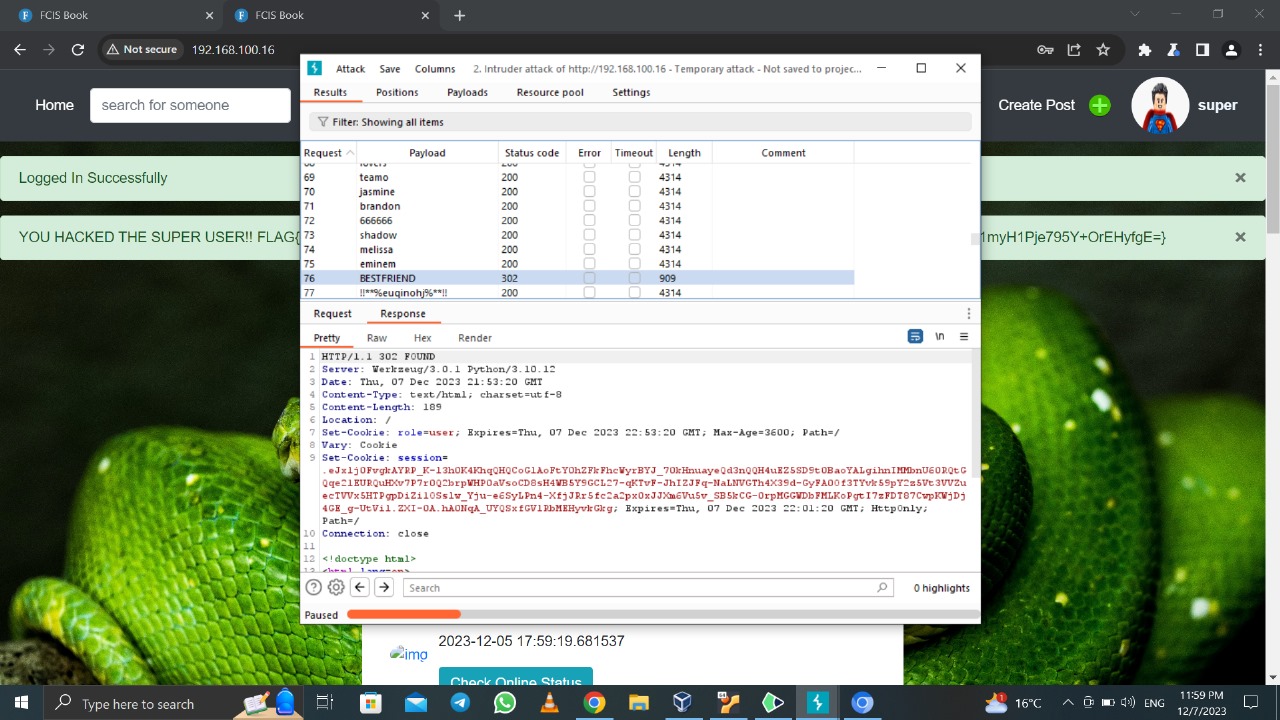
|  |  |
| --- | --- |
| **Description** | * a type of security vulnerability that occurs when a web application allows unauthorized access to files and directories outside the intended or authorized directory structure. * We apply it when we press on photos on posts we notice that the picture is downloaded so we open intercept on burb suit and we click on image to be download we notice on the request on file name the file we download so we apply directory traversal (../) several times and add secret.txt and we forward the request so secret.txt is downloaded instead of the picture |
| **Recommendation** | * avoid passing user-supplied input to file system APIs. * Validate user input by comparing it to an allow list of permitted values. If that’s not possible, ensure that the input only contains alphanumeric characters. * After validating the user supplied input, use file system APIs to canonicalize the path and verify that it starts with the expected base directory. |
| **Affect systems** | * The systems hosting the file download functionality * **it effects the Confidentiality: as we have un authorized access to sensitive file** * also effect on **integrity** as we can manipulate on file |
| **Threat level** | High |

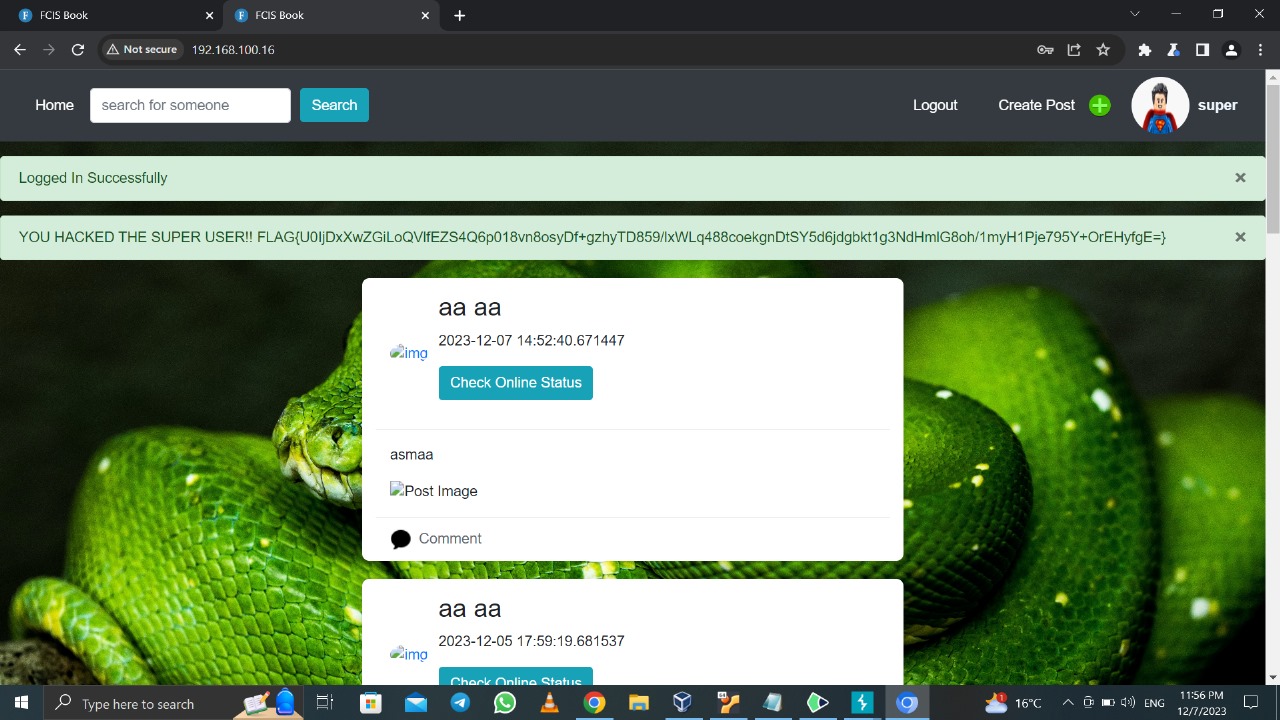




**10-Authentaction**

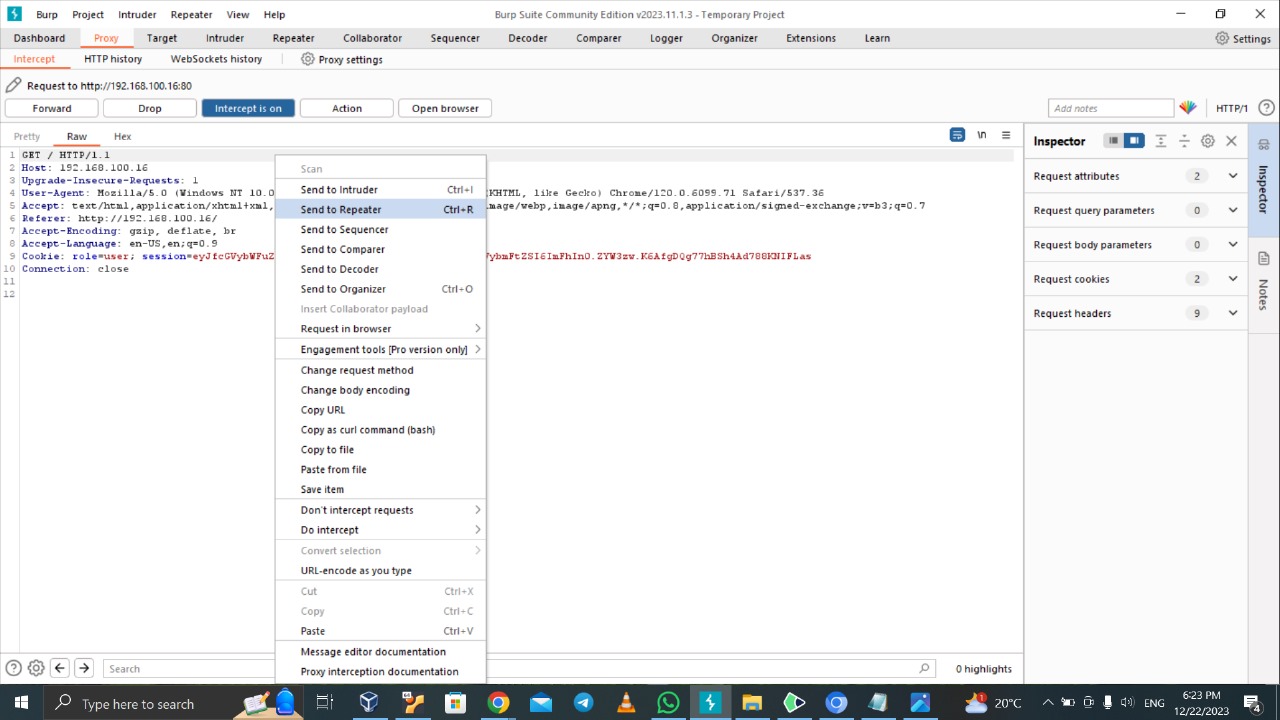
|  |  |
| --- | --- |
| **Description** | 1. Authentication vulnerability is critical vulnerability as we could login as specific user **depend on**:  * information we know about user and we use it to guess the password * we already know password or password is weak * Apply brute force attack   2- we use information we get from **IDOR Vulnerability** as we know the users on the website so we have username so we apply brute force attack on each user using list of passwords provided and we login as user (super). |
| **Recommendation** | * Prevent applying brute force by making specific number of trials to login with a wrong username or password * We can also make a strong password and does not contain personal information people know it and could use it to know password |
| **Affect systems** | * Users specifically super user and system * it effects **confidentiality**: attacker can view user data * also effects **integrity**: as attacker can update user’s data * it effects on **availability**: could delete the account |
| **Threat level** | Critical |

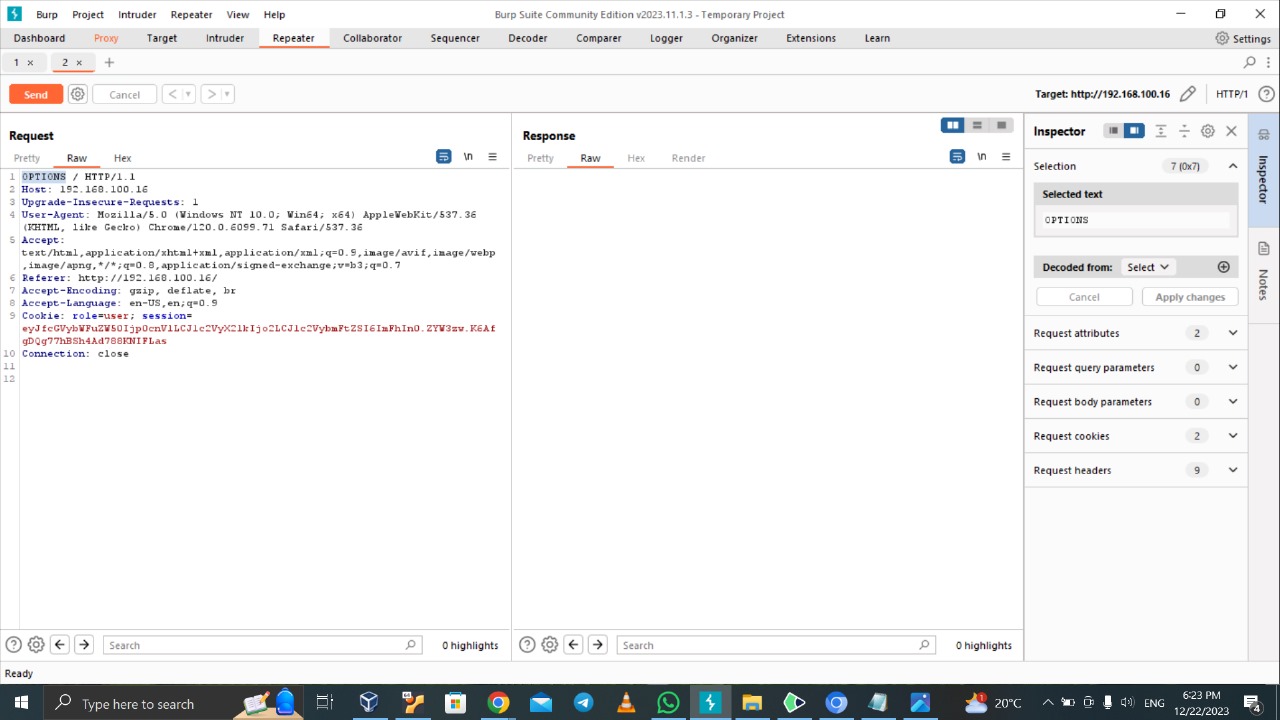


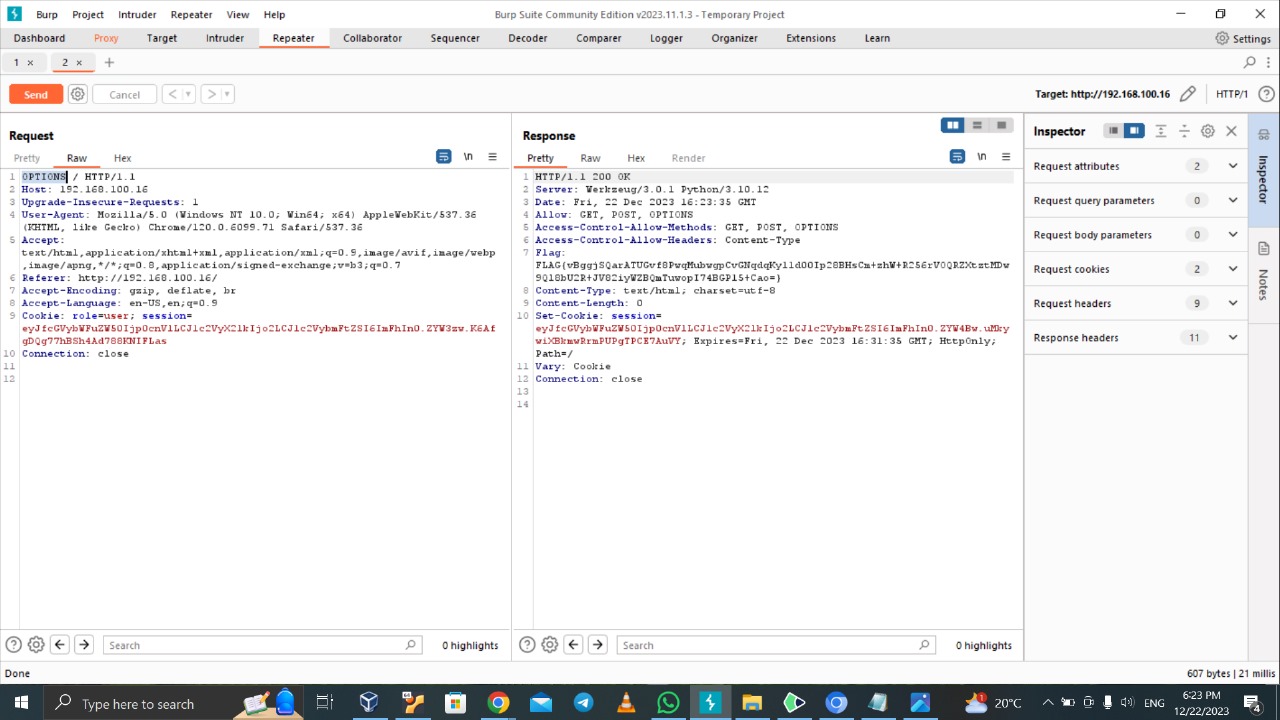


**11-CRUL**

|  |  |
| --- | --- |
| **Description** | * It is a way to change the methods of request * We try to click on button on home page and before it we open intercept on burb suit we send the request to repeater and we change get methods to options |
| **Recommendation** | * Handle different HTTP methods |
| **Affect systems** | * web application's server |
| **Threat level** | Low |

****

****

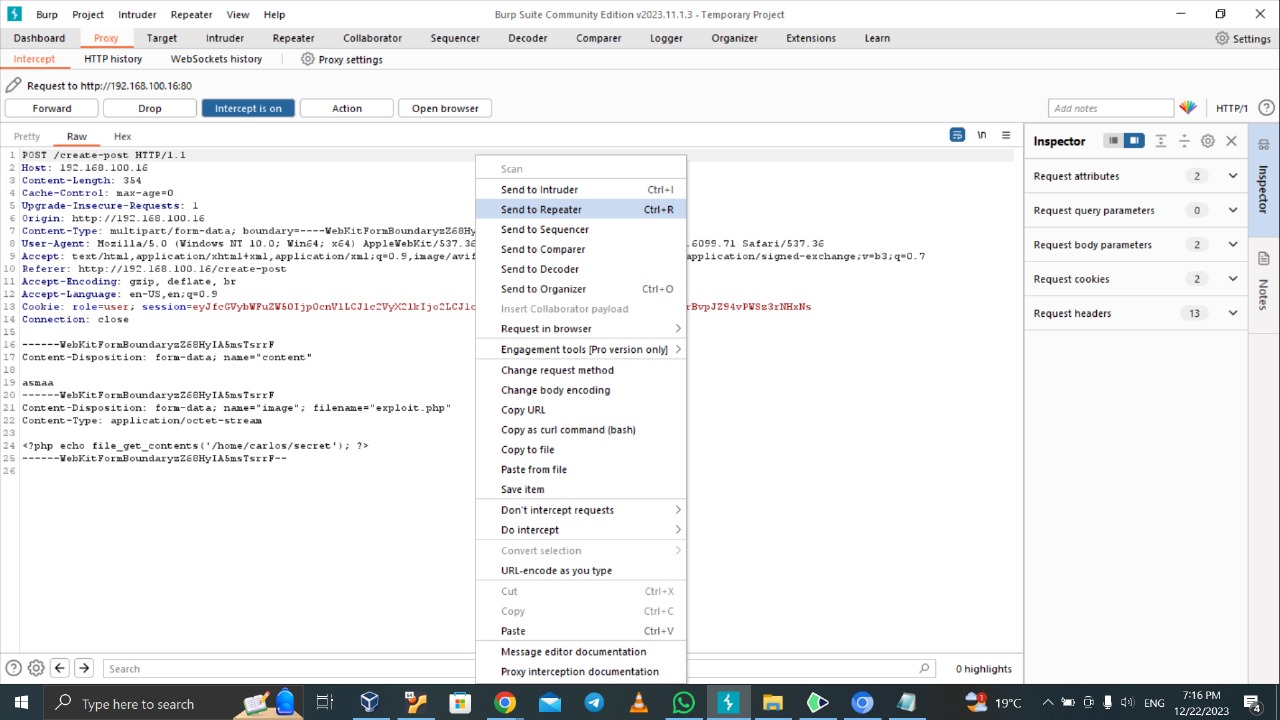
****

**12- file upload**

|  |  |
| --- | --- |
| **Description** | * This vulnerability allow attacker to upload and execute malicious files on a web server. * This can lead to unauthorized access to sensitive data, server compromise, and even complete system control (RCE). * We open create post then we upload php file before that we open intercept then resend to repeater then we change content type to image / jbg |
| **Recommendation** | Make restriction and validations and make a white list on files allowed |
| **Affect systems** | The website and file upload |
| **Threat level** | Critical |

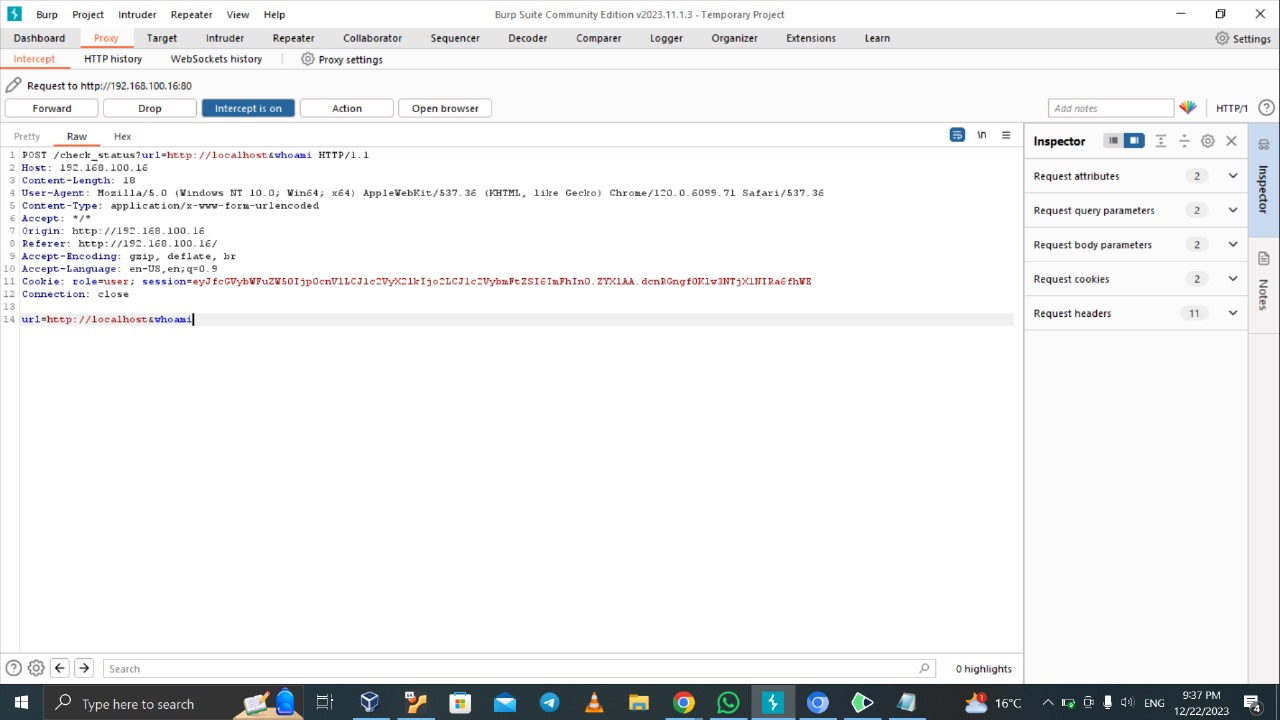
****

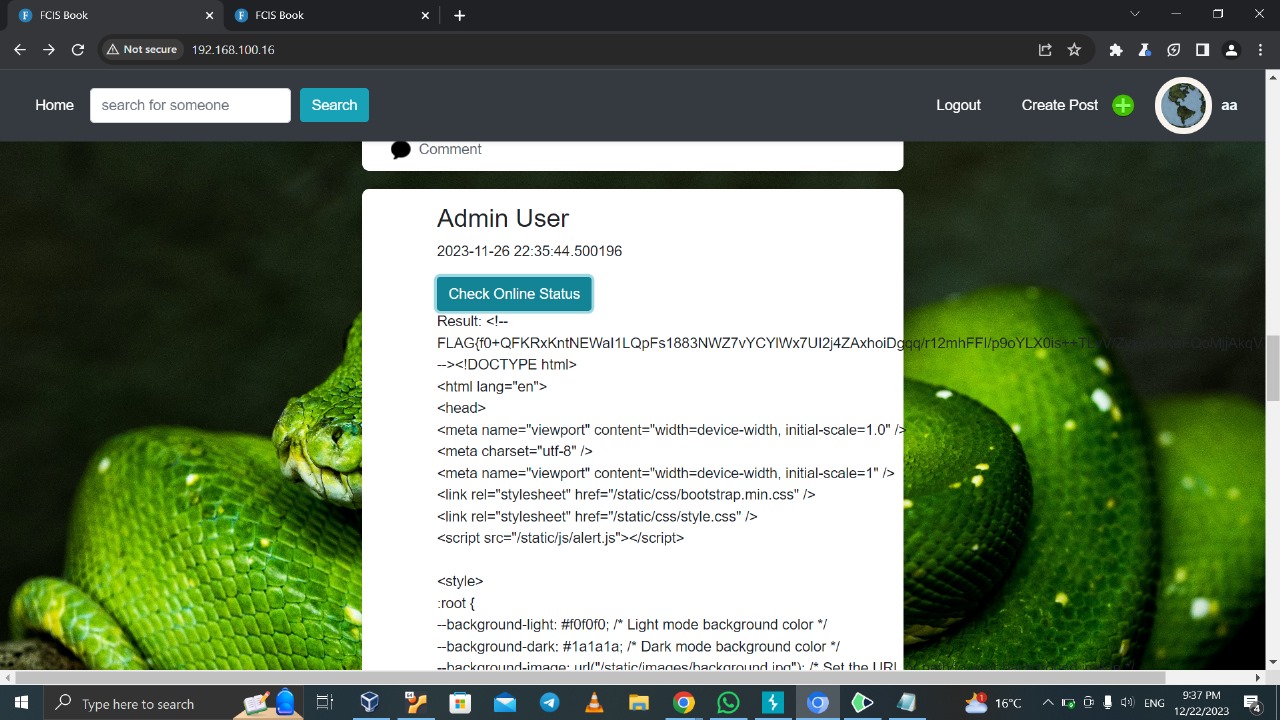
****

****

**13-SSRF**

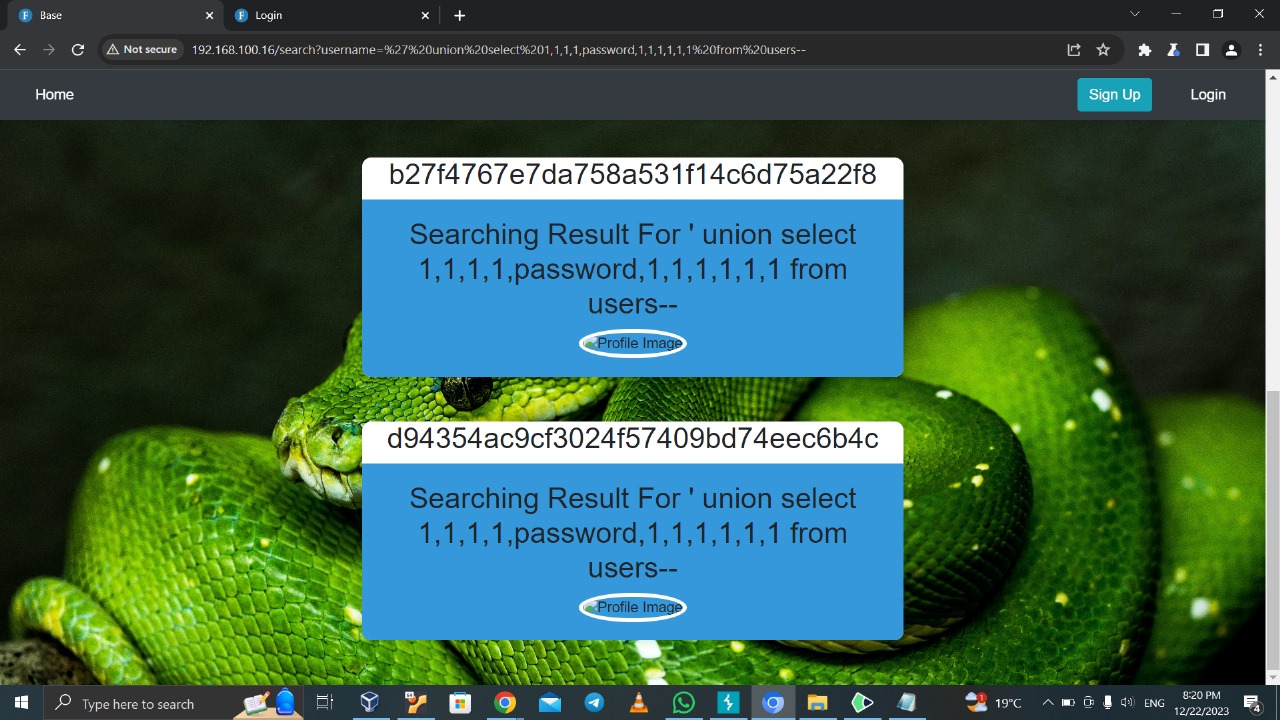
|  |  |
| --- | --- |
| **Description** | * It is important vulnerability , we open intercept and click check status then we enter on URL and write localhost and who am I and forward it |
| **Recommendation** | * **Implement Access Controls** |
| **Affect systems** | * Potentially the local server or application running on localhost. |
| **Threat level** | * High to Critical |

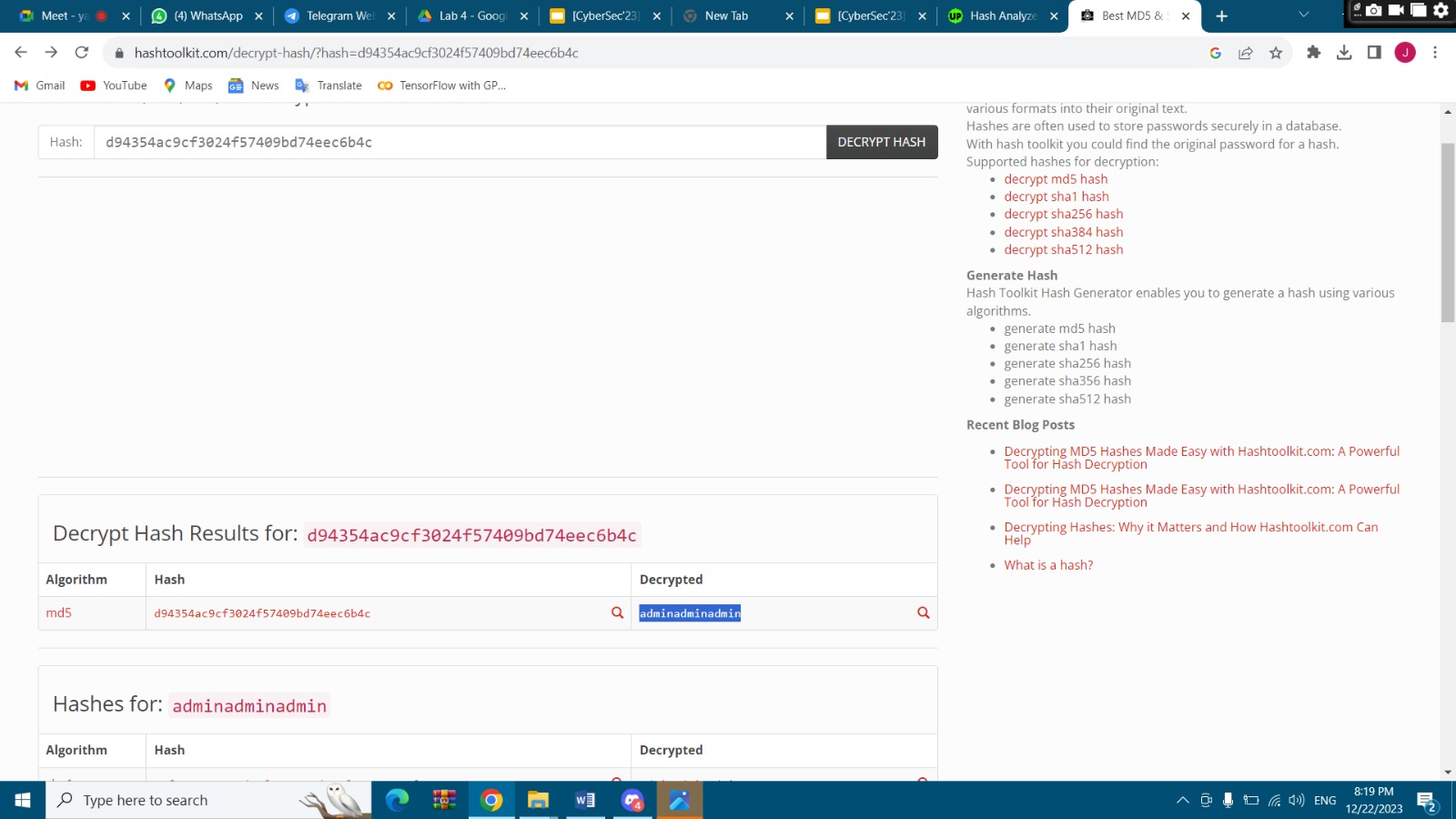
****

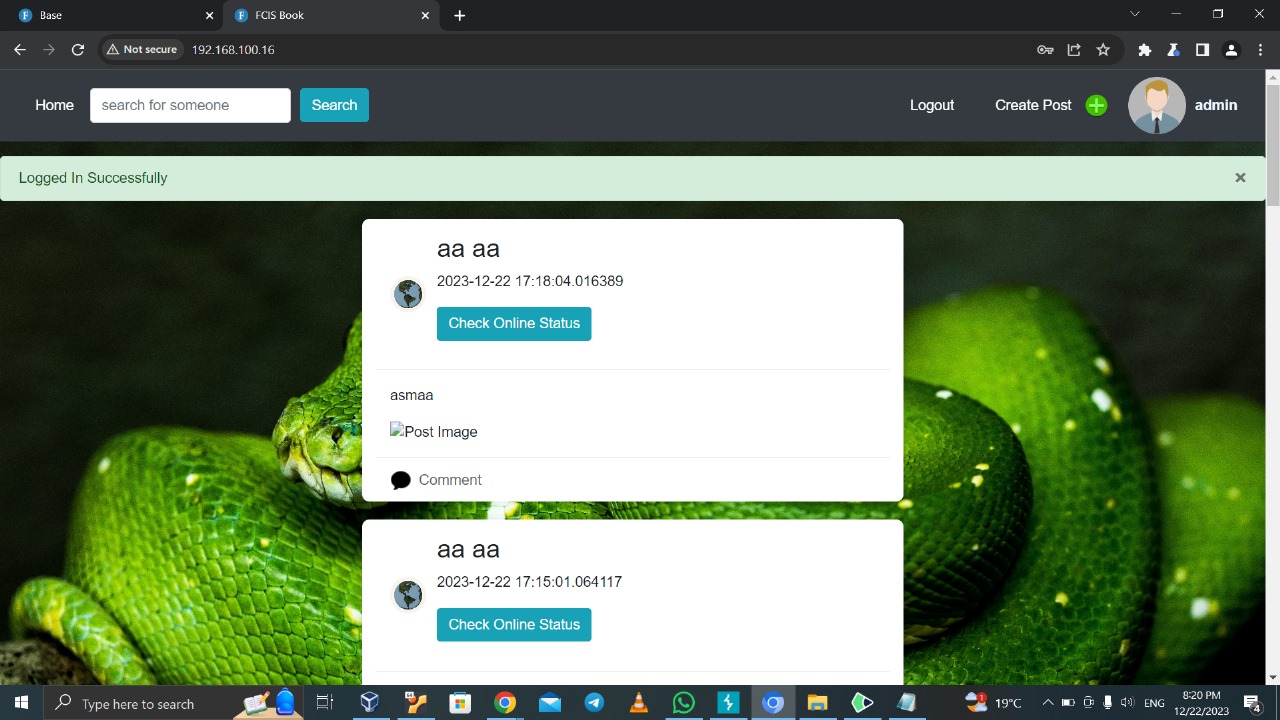
****

**14-Crtptograpy**

|  |  |
| --- | --- |
| **Description** | * Is one of important vulnerabilities * We use SQL injection (union) to show all passwords on database, so we see hash passwords and we copy it to hash analyzer so we know its MD5/MD4 so we open MD5/MD4 decode and we decode all passwords and we know the password of admin so we log in with it |
| **Recommendation** | * Use strong, salted hashing algorithms for storing passwords to increase the difficulty of password decryption. |
| **Affect systems** | * web application's database and user’s accounts |
| **Threat level** | **Critical** |

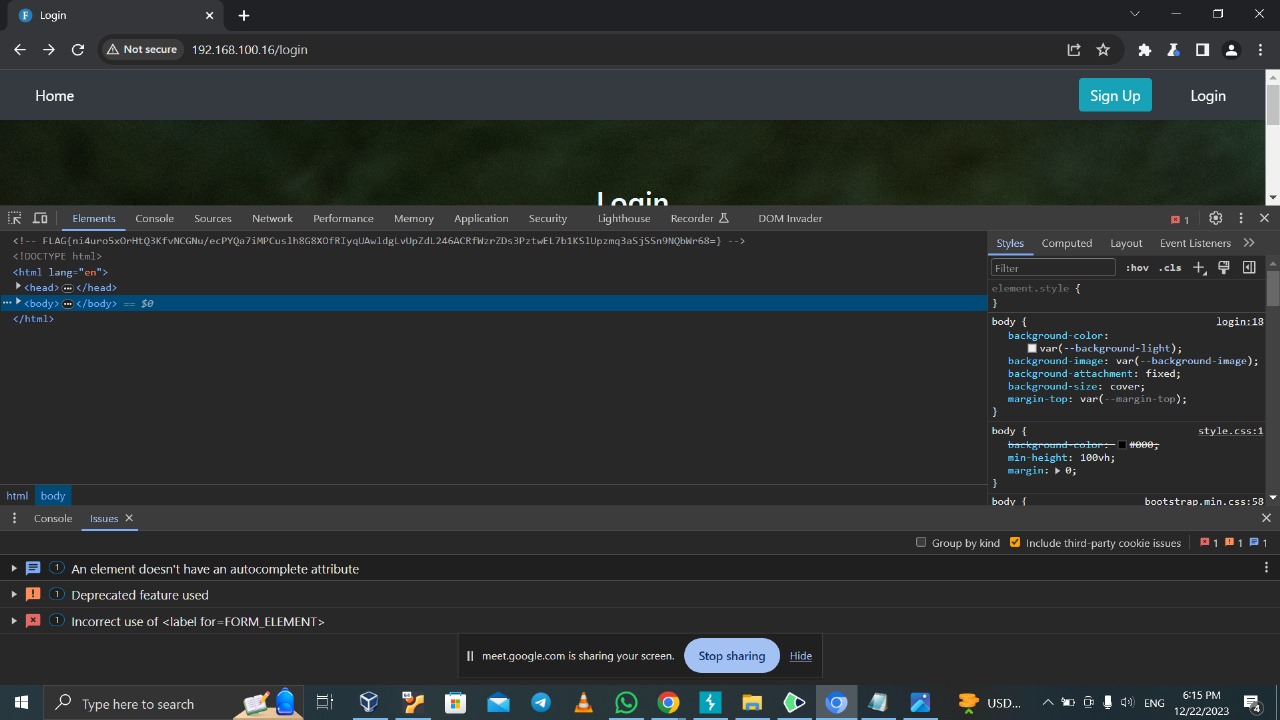
****

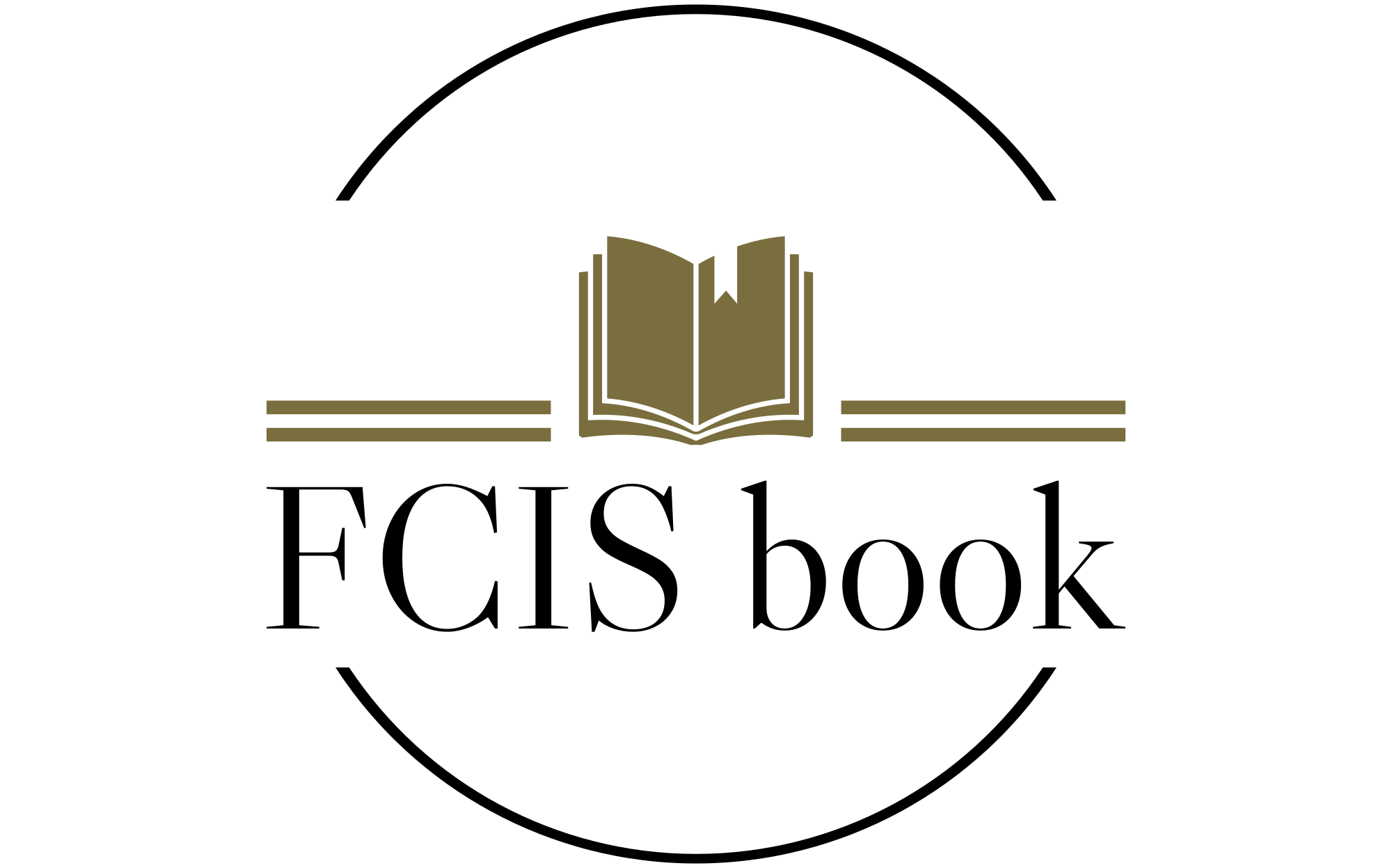
****

****

**15- flag on login (informational)**

|  |  |
| --- | --- |
| **Description** | * We open on login page the inspect to show if there any useful information and we see a flag |
| **Recommendation** | * Not show any useful information on inspect |
| **Affect systems** | It is only informational but we can say login page and users |
| **Threat level** | informational |

****



Last Page