FCIS Book
Penetration Testing Report

FCIS book

Team members

ID	Name
20201701163	يوسف شعبان السيد
20201701036	يوسف محمد منصور عبدالحفيظ
20201700478	عبدالرحمن عمرو محمد
20201701218	مريم رضا علي
20201700799	مروه سامح علي
20201701063	ايمان ممدوح عوض

Table of Contents

1 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

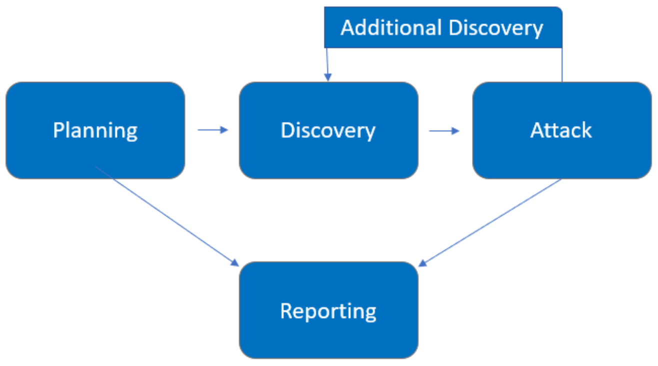
And ***FIVE*** Bonus vulnerabilities

2 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.



2.1 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.

2.2 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.

2.3 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.

2.4 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.

3 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.

4 Scope of Engagement

The scope of this engagement includes performing Internal Penetration Testing for FCIS book.

4.1 Targets

Assessment	Details
Internal Penetration Test	Local host

4.2 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.

5 Findings Table

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Outdated Software version at	Medium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

6 Technical Findings

6.1 Authentication

Description:	The vulnerability discovered is related to weak authentication on the system's 'super' username. By utilizing Burp Suite and performing a brute force attack against a list of passwords, unauthorized access was gained to the 'super' account and it is password is ‘BESTFRIEND’.
Recommendations	• Implement multi-factor authentication (MFA): Adding an extra layer of authentication will enhance security significantly. • use an encrypted channel and connection (https) when sending user credentials. • Implement account lockout mechanisms: After a certain number of failed login attempts, lock the account temporarily to prevent brute force attacks. • Implement stronger password policies: Enforce longer, complex passwords, and periodic password changes. • Use only (POST) requests should be used to transmit credentials to the server. • Stored passwords should be hashed and salted using cryptographically secure algorithms to make it hard.
Affected Systems	192.168.100.34 (Login page)
Threat Level	High

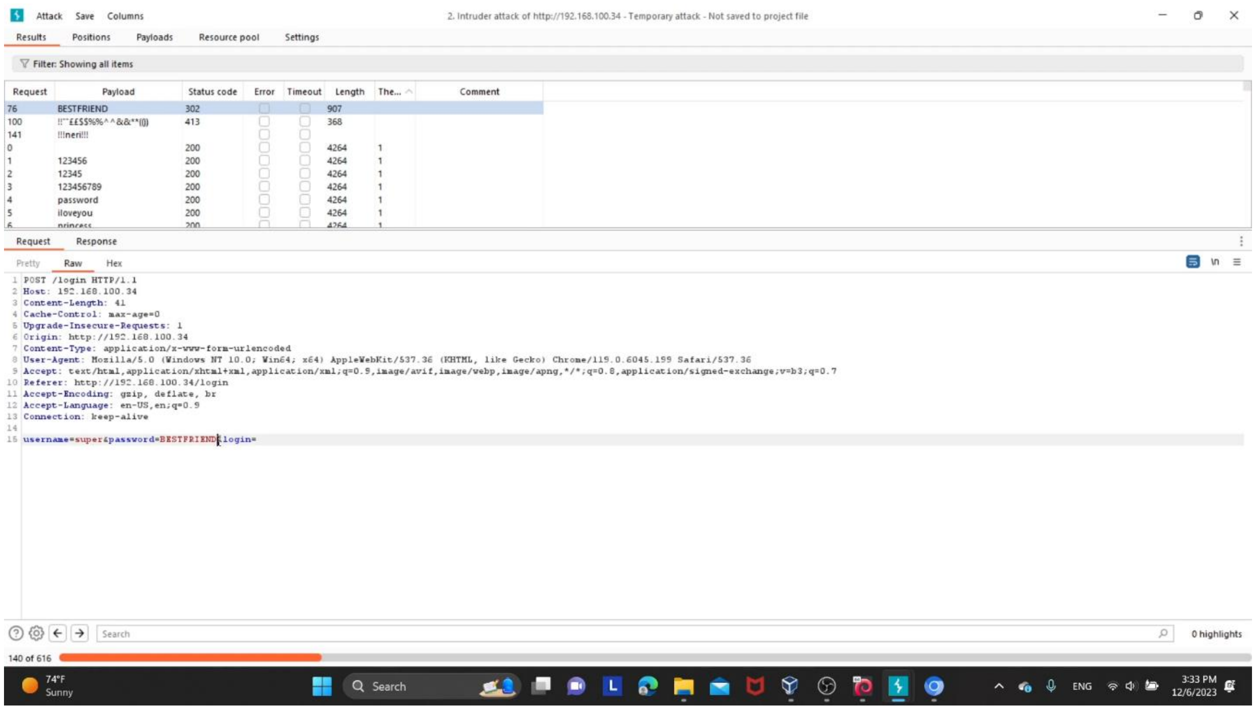
***Flag*:**

FLAG{aUcB8VO2bIHFEM9upBgOY3oFEOrBLwatezZ5sDlorxLFE9ORFkWFoviNcRRhriIDDnXJ
6+bOZUdul6pLnEvJTAIMYFWkuY0K8uJKQTrrOeE=}

Steps to reproduce: -

1- First, after reading the manual of the system we got that there is a ‘super’ username.
2- Go to login page of the system and try to login with username ‘super’ and any password.
3- Then open Burp Suite and send the request of login to ‘Intruder’ and add payload on password.
4- Going to Payloads section and paste the list of passwords and in settings paste the incorrect message of login.
5- Then, Start the brute force attack.

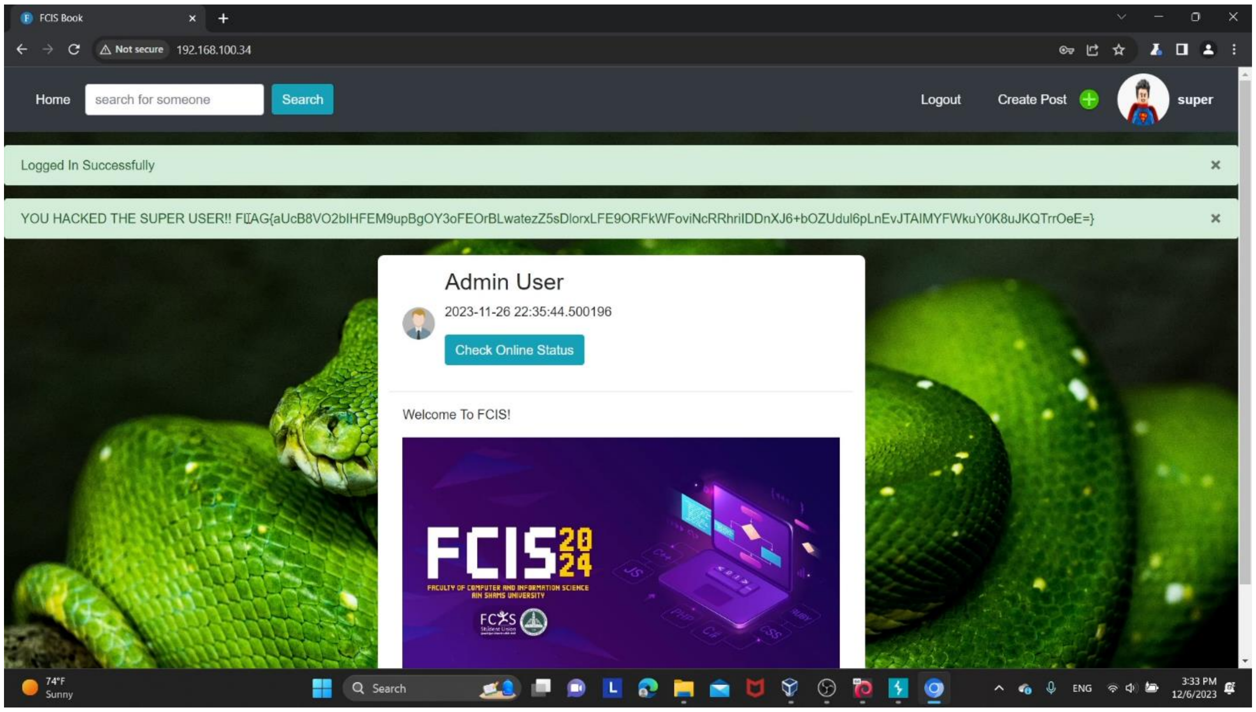
Screenshots



Request	Payload	Status code	Error	Timeout	Length	The...	Comment
76	BESTFRIEND	302	☐	☐	907		
100	!!"£££$$%%^^&&**(())	413	☐	☐	368		
141	!!!neri!!!		☐	☐			
0		200	☐	☐	4264	1	
1	123456	200	☐	☐	4264	1	
2	12345	200	☐	☐	4264	1	
3	123456789	200	☐	☐	4264	1	
4	password	200	☐	☐	4264	1	
5	iloveyou	200	☐	☐	4264	1	
6	princess	200	☐	☐	4264	1	

```
POST /login HTTP/1.1
Host: 192.168.100.34
Content-Length: 41
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://192.168.100.34
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.199 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://192.168.100.34/login
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: keep-alive

username=super&password=BESTFRIEND&login=
```



Logged In Successfully

YOU HACKED THE SUPER USER!! FLAG{aUcB8VO2bIHFEM9upBgOY3oFEOrBLwatezZ5sDlorxLFE9ORFkWFoviNcRRhriIDDnXJ6+bOZUdul6pLnEvJTAIMYFWkuY0K8uJKQTrrOeE=}

Admin User

2023-11-26 22:35:44.500196

Welcome To FCIS!

6.2 Reflected Cross-Site Scripting (Reflected XSS)

Description:	The discovered vulnerability is a Reflected Cross-Site Scripting (Reflected XSS) issue found within the search bar functionality. By inputting a crafted script <script>alert(1)</script> into the search bar, the website reflects and executes the script, get an alert. This indicates that the website is vulnerable to reflected XSS attacks.
Recommendations	• Input validation and output encoding: Validate and sanitize user inputs to prevent malicious scripts from being executed, implement, use well-known secure library. • Only accept input based on what is expected or valid to enter in the section. • use the Content-Type and X-Content-Type-Options headers to ensure that browsers interpret the responses in the way you intend. • Use security headers: Implement Content Security Policy (CSP) to restrict the sources from which content can be loaded.
Affected Systems	192.168.100.34 (Search bar)
Threat Level	Moderate

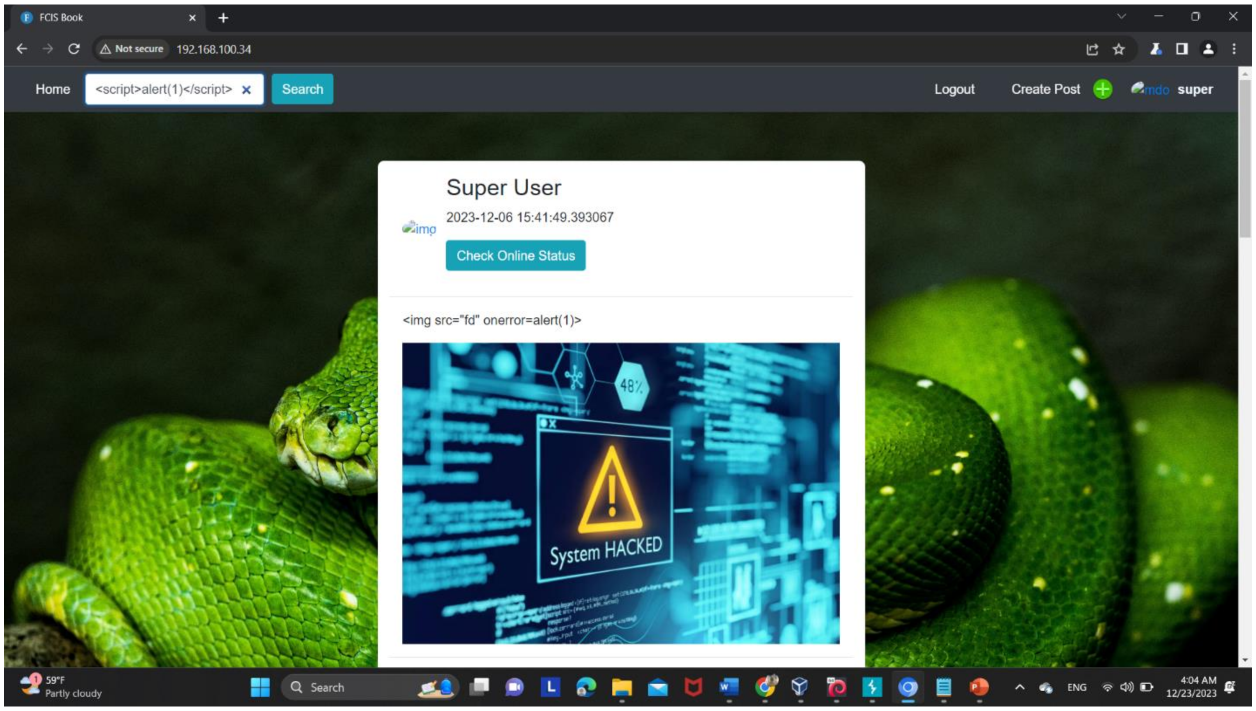
***Flag*:**

FLAG{0rOGPxtv4biaiAcIF/NQOeFECOKC8VUZmqtammSSfcetmavV7tD9Ur/IByrR5n9cewuIQn
/3IWRsotHHbsMVsw==}

Steps to reproduce: -

1- In search bar in website, we try to enter this script <script>alert(1)</script> and click on search button.
2- The payload is executed.
3- There is an immediate reflected response (alert) get on the website.

Screenshots

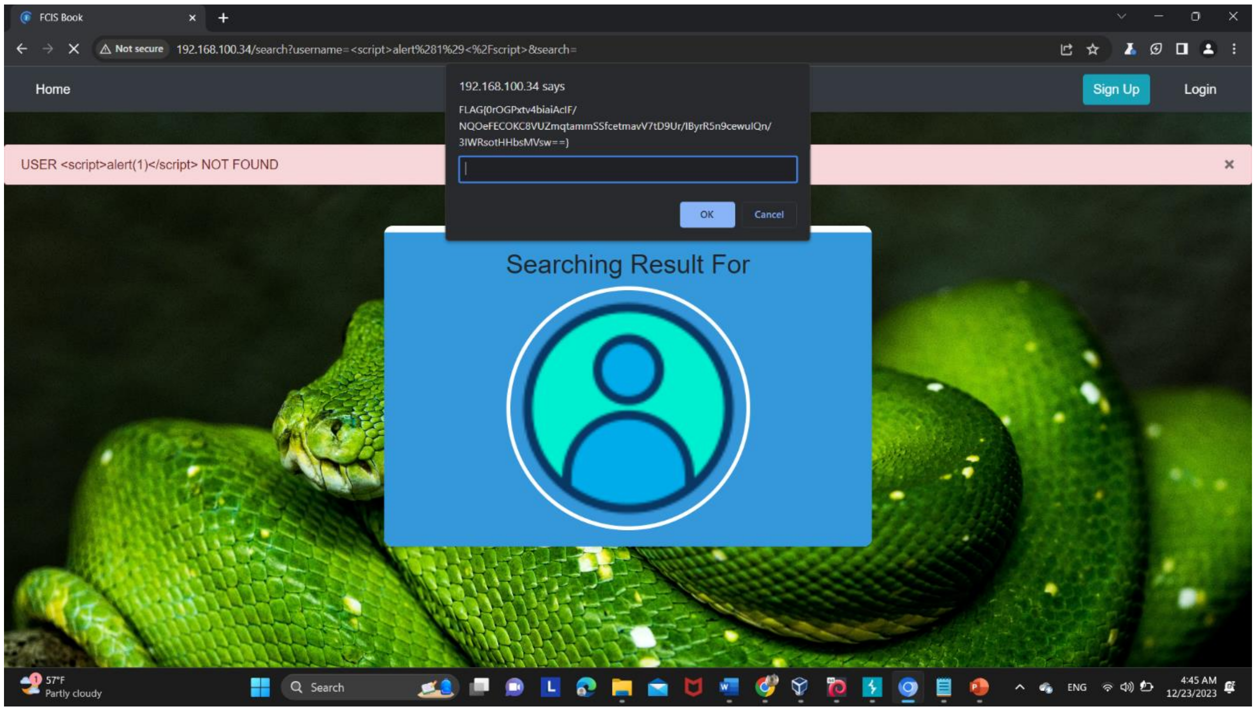


Super User

2023-12-06 15:41:49.393067

Check Online Status

<img src="fd" onerror=alert(1)>


FCIS Book
Not secure
192.168.100.34/search?username=<script>alert%281%29<%2Fscript>&search=
Home
Sign Up
Login
OK
Cancel
Searching Result For

192.168.100.34 says

```
FLAG{0rOGPxtv4biaiAclF/
NQOeFECOKC8VUZmqtammSSfcetmavV7tD9Ur/IByrR5n9cewuIQn/
3IWRsotHHbsMVsw==}
```

USER <script>alert(1)</script> NOT FOUND

6.3 Stored Cross-Site Scripting (Stored XSS)

Description:	This vulnerability is a Stored Cross-Site Scripting (Stored XSS) found within the comments section of the system. By inputting a crafted script <script>alert(1)</script> into the comment field and subsequently viewing the comment or comments page, the website reflects and executes the script. This signifies that the system is vulnerable to stored XSS attacks, where he injected script persists and gets executed whenever the affected content is displayed.
Recommendations	• Input validation and output encoding: Implement strict input validation and sanitize user inputs to prevent malicious scripts from being stored in the database. • Only accept input based on what is expected or valid to enter in the section. • use the Content-Type and X-Content-Type-Options headers to ensure that browsers interpret the responses in the way you intend. • Content security policy (CSP): Utilize CSP headers to restrict which sources are allowed to be loaded and executed.
Affected Systems	192.168.100.34 (comments)
Threat Level	High

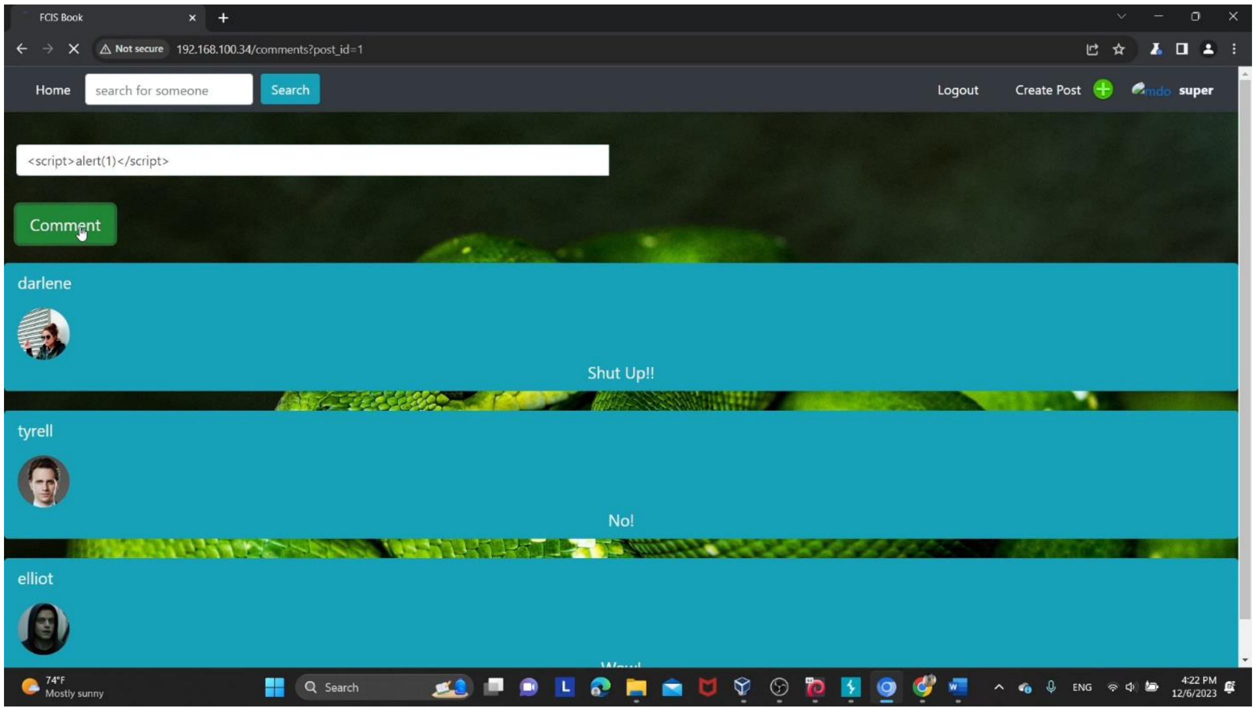
***Flag*:**

FLAG{4LQXjd1ATGb1i/8Gqs28LJN9h/vHxV9wmFp07dqAwnHRKht2U4Mvp5psy6QYji5NYUu8
NWpWcOrliUUmITw4qw==}

Steps to reproduce: -

1- In comments page in website when we try to add a comment on post and enter this script in comment bar <script>alert(1)</script> and click on comment button.
2- Going to open the comments page and when just you visit this comment.
3- This script is executed and it shows an alert in the page.

Screenshots



192.168.100.34/comments?post_id=1

```
<script>alert(1)</script>
```

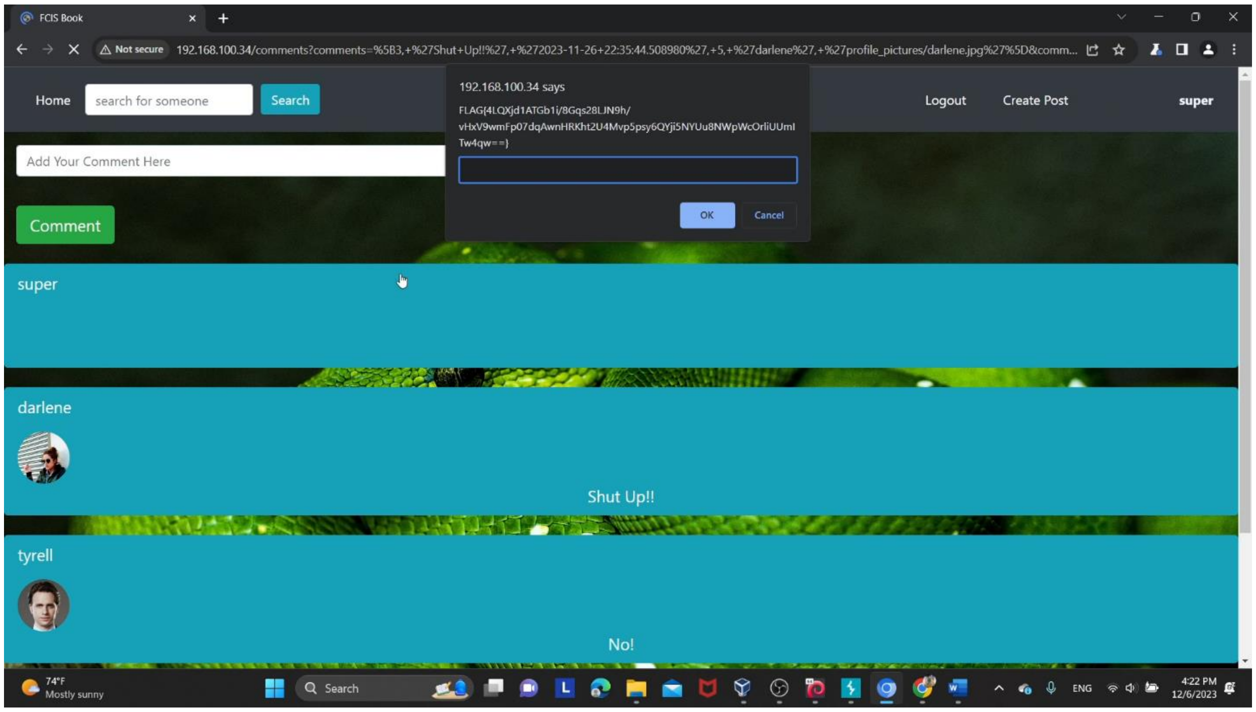
darlene

Shut Up!!

tyrell

No!

elliot



192.168.100.34/comments?comments=%5B3,+%27Shut+Up!!%27,+%272023-11-26+22:35:44.508980%27,+5,+%27darlene%27,+%27profile_pictures/darlene.jpg%27%5D&comm...

192.168.100.34 says

```
FLAG{4LQXjd1ATGb1i/8Gqs28LJN9h/
vHxV9wmFp07dqAwnHRKht2U4Mvp5psy6QYji5NYUu8NWpWcOrliUUml
Tw4qw==}
```

super

darlene

Shut Up!!

tyrell

No!

6.4 DOM-based Cross-Site Scripting (Dom XSS)

Description:	This vulnerability is a DOM Cross-Site Scripting (DOM XSS) found within the functionality of the 'Check Online Status' button on website page. By modifying the onclick attribute using the browser's developer tools 'inspect' to onclick="alert(1)", the attacker can manipulate the button button's behaviour to execute arbitrary JavaScript code. This signifies that the system is vulnerable to DOM XSS attacks, where the manipulation of the Document Object Model (DOM) leads to script execution.
Recommendations	• Sanitize and validate user inputs: Implement strict input validation and ensure that user-controlled data is not directly used to manipulate the DOM.Only accept input based on what is expected or valid to enter in the section. • Use safe JavaScript libraries: Utilize secure JavaScript frameworks that automatically handle DOM manipulation securely. • Avoid using user inputs in client-side scripting: Refrain from directly incorporating user-controlled data into client-side scripts or DOM manipulations without proper validation.
Affected Systems	192.168.100.34 (Check Online Status Button)
Threat Level	Moderate

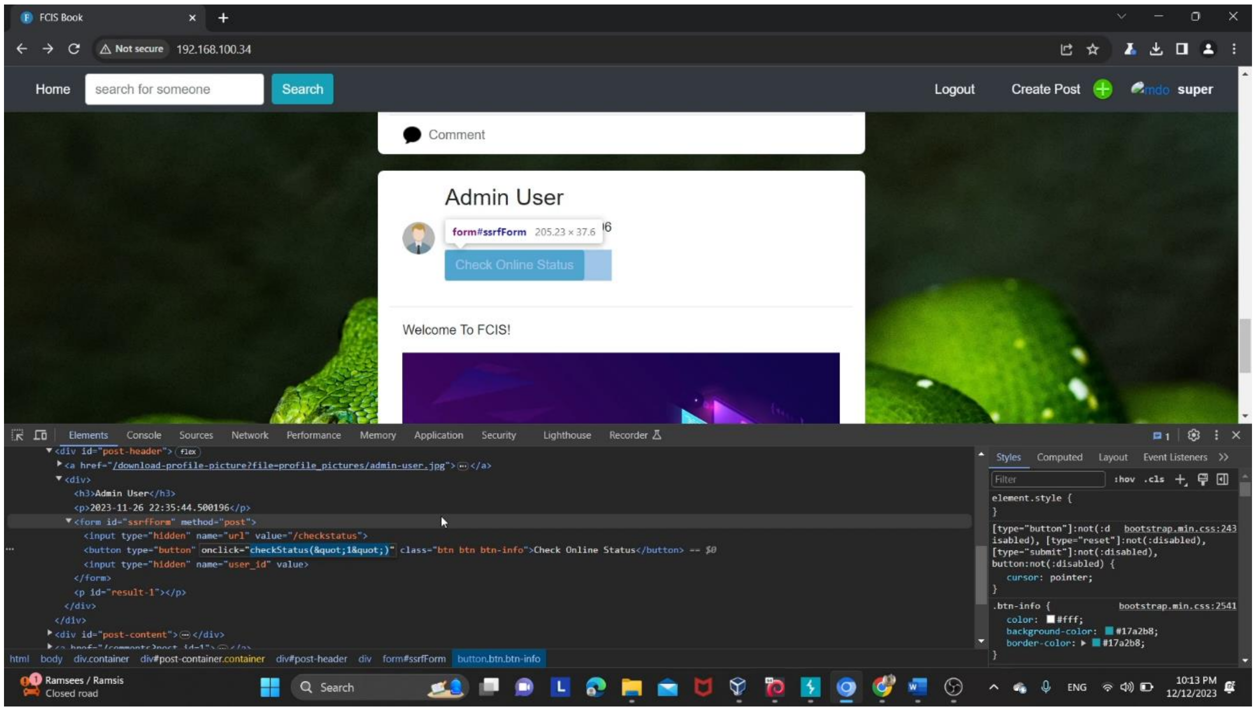
Flag:

FLAG{7FIeNV0M/tjxB0vWOCaZERE2zpbgbKx5bG6YbBWlaNClqORku1g5JnrVY35F2wCO
RdY6/BGOLmIut83MRNggVQ==}

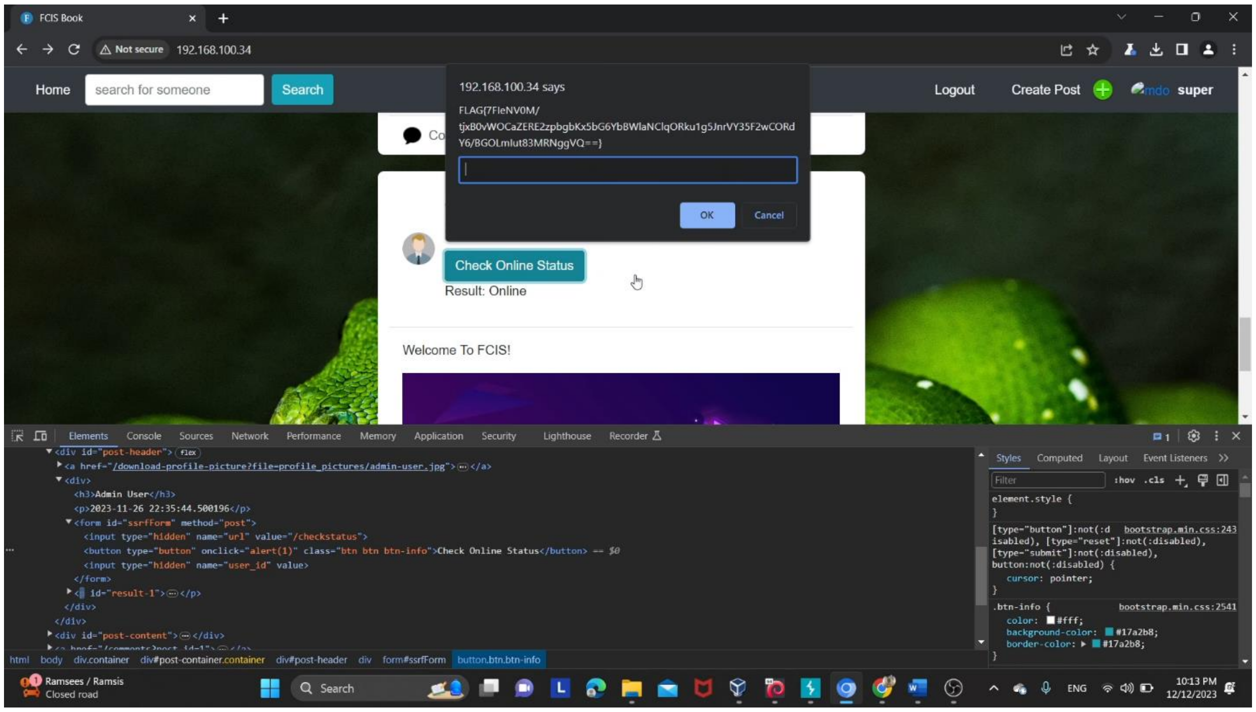
Steps to reproduce: -

1- In home page, right click on 'Check Online Status' button.
2- Then, click on inspect.
3- In html code in tag button, modify the behaviour or functionality of this button.
4- Replace the attribute of onclick from onclick="checkStatus(&quot;1&quot;)" to onclick="alert(1)".
5- Then, when we click on 'Check Online Status' button it executed the JavaScript code and shows me an alert in page instead of return me the user status.

Screenshots



```
▼<div id="post-header"> flex
  ▶<a href="/download-profile-picture?file=profile_pictures/admin-user.jpg">⋯</a>
  ▼<div>
     <h3>Admin User</h3>
     <p>2023-11-26 22:35:44.500196</p>
    ▼<form id="ssrfForm" method="post">
       <input type="hidden" name="url" value="/checkstatus">
       <button type="button" onclick="checkStatus(&quot;1&quot;)" class="btn btn btn-info">Check Online Status</button> == $0
       <input type="hidden" name="user_id" value>
     </form>
     <p id="result-1"></p>
   </div>
 </div>
▶<div id="post-content">⋯</div>
```



```
192.168.100.34 says
FLAG{7FleNV0M/
tjxB0vWOCaZERE2zpbgbKx5bG6YbBWlaNClqORku1g5JnrVY35F2wCORd
Y6/BGOLmlut83MRNggVQ==}
```

```
▼<div id="post-header"> flex
  ▶<a href="/download-profile-picture?file=profile_pictures/admin-user.jpg">⋯</a>
  ▼<div>
     <h3>Admin User</h3>
     <p>2023-11-26 22:35:44.500196</p>
    ▼<form id="ssrfForm" method="post">
       <input type="hidden" name="url" value="/checkstatus">
       <button type="button" onclick="alert(1)" class="btn btn btn-info">Check Online Status</button> == $0
       <input type="hidden" name="user_id" value>
     </form>
    ▶<p id="result-1">⋯</p>
   </div>
 </div>
▶<div id="post-content">⋯</div>
```

6.5 SQL Injection

Description:	The vulnerability discovered is a SQL Injection flaw within the search functionality of the system. By inputting the query ' or 1=1 -- into the search bar and clicking the search button, the system executes the query and returns user data. This behaviour indicates that the system is vulnerable to SQL Injection attacks, where malicious SQL code is injected into the input fields, altering the intended SQL query and allowing unauthorized access to the database.
Recommendations	• Use parameterized queries: Utilize parameterized queries or prepared statements to separate SQL code from user inputs, preventing direct concatenation of user inputs into SQL queries. • Whitelist Input Validation: Defining what values are authorized. Everything else is considered unauthorized. Useful for values that cannot be specified as parameter placeholders, such as the table name. • Input validation and sanitization: Implement strict input validation and sanitize user inputs to remove or escape potentially malicious characters.
Affected Systems	192.168.100.34 (Search bar)
Threat Level	Critical

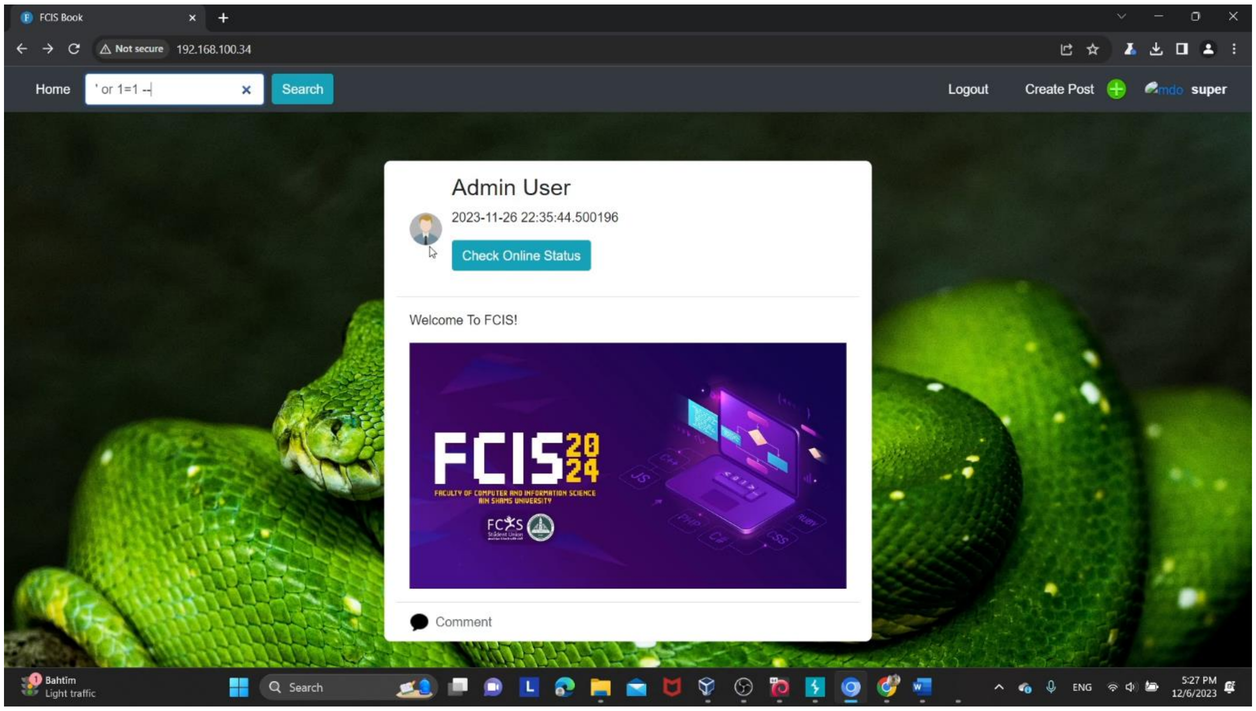
***Flag*:**

FLAG{18fePeyjXBlCFNd5OUCYdsTsOOtVAukV1uaZyXjTkfg7V417JjPBFru+hPDW5aNfXLf
FIVniYEGxKnorly4QippeVL3fdHe8pq7cjTivbtA=}

Steps to reproduce: -

1- In search bar, enter this statement in search ' 1=1 or --.
2- Then, click on search button.
3- The system will retrieve a data about users which there are in the system.

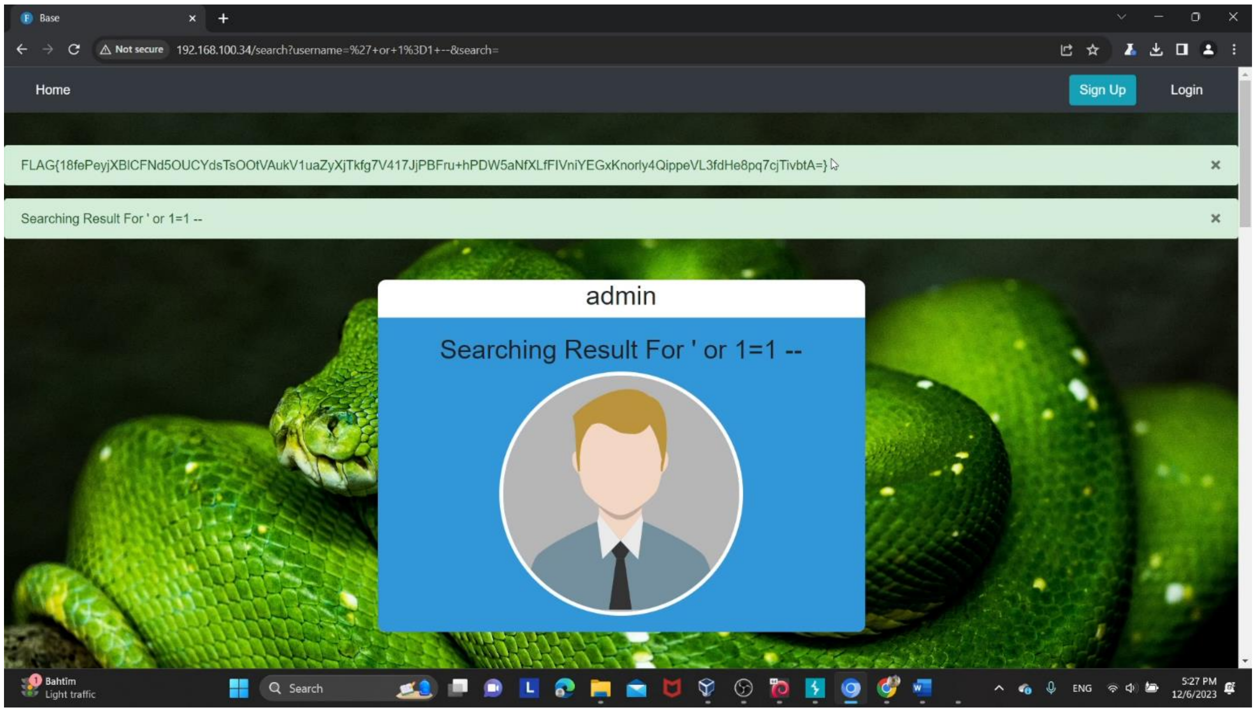
Screenshots



Admin User

2023-11-26 22:35:44.500196

Welcome To FCIS!



```
192.168.100.34/search?username=%27+or+1%3D1+--&search=
```

```
FLAG{18fePeyjXBlCFNd5OUCYdsTsOOtVAukV1uaZyXjTkfg7V417JjPBFru+hPDW5aNfXLfFIVniYEGxKnorly4QippeVL3fdHe8pq7cjTivbtA=}
```

Searching Result For ' or 1=1 --

admin

Searching Result For ' or 1=1 --

6.6 SQL Injection (Union)

Description:	This vulnerability is a SQL Injection flaw discovered within the search bar functionality of the system. By inputting the SQL query ' union select password, password, password, password, password, password, password, password, password, password, password, password from users -- into the search bar and clicking the search button, the system executes the query and returns user's hashed passwords from the database. This behaviour indicates that the system is vulnerable to Union SQL Injection attacks, allowing an attacker to manipulate the SQL query to extract sensitive data, in this case, the hashed passwords.
Recommendations	• Use parameterized queries: Utilize parameterized queries or prepared statements to separate SQL code from user inputs, preventing direct concatenation of user inputs into SQL queries. • Whitelist Input Validation: Defining what values are authorized. Everything else is considered unauthorized. Useful for values that cannot be specified as parameter placeholders, such as the table name. • Input validation and sanitization: Implement strict input validation and sanitize user inputs to remove or escape potentially malicious characters.
Affected Systems	192.168.10.34 (Search bar)
Threat Level	Critical

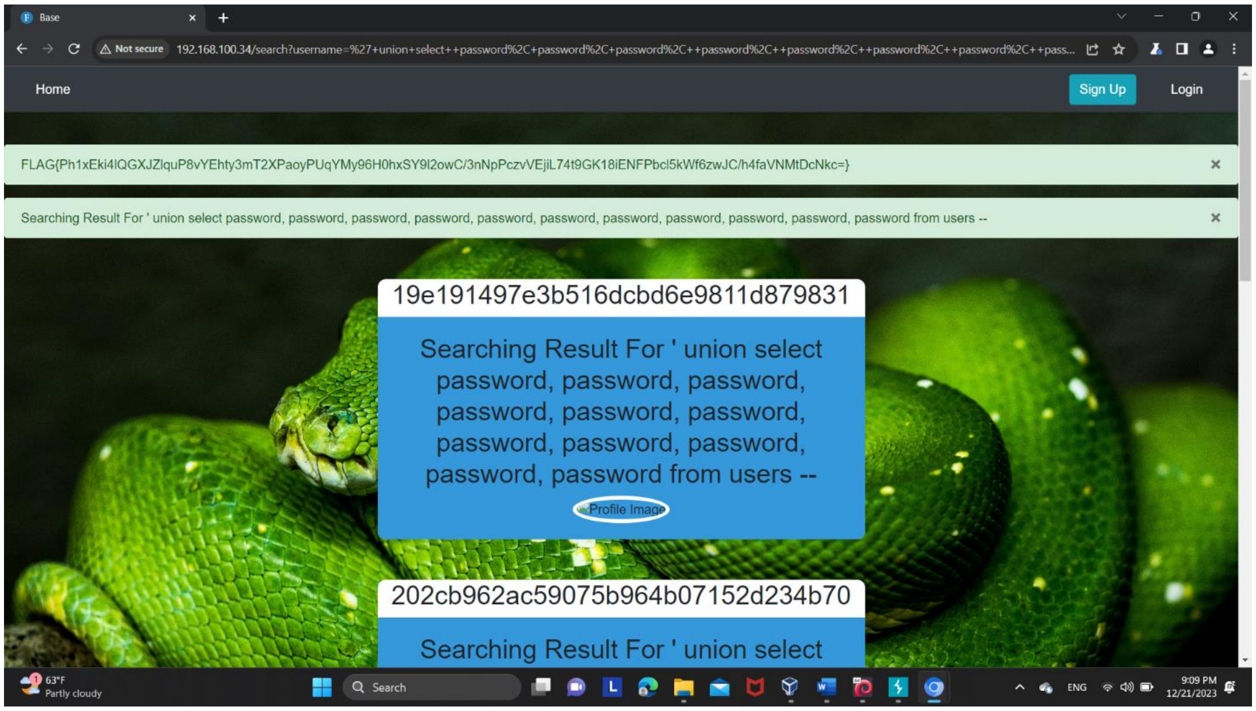
***Flag*:**

FLAG{Ph1xEki4lQGXJZlquP8vYEhty3mT2XPaoyPUqYMy96H0hxSY9l2owC/3nNpPczvVEji
L74t9GK18iENFPbcl5kWf6zwJC/h4faVNMtDcNkc=}

Steps to reproduce: -

1- In search bar, enter this statement in search ' union select password, password, password, password, password, password, password, password, password, password, password, password from users --.
2- Then, click on search button.
3- The system will retrieve a data of with the user's hashed passwords which there are in the system.

Screenshots



```
192.168.100.34/search?username=%27+union+select++password%2C+password%2C+password%2C++password%2C++password%2C++password%2C++password%2C++pass...
```

FLAG{Ph1xEki4lQGXJZlquP8vYEhty3mT2XPaoyPUqYMy96H0hxSY9l2owC/3nNpPczvVEjiL74t9GK18iENFPbcl5kWf6zwJC/h4faVNMtDcNkc=}

Searching Result For ' union select password, password, password, password, password, password, password, password, password, password, password from users --

19e191497e3b516dcbd6e9811d879831

Searching Result For ' union select password, password, password, password, password, password, password, password, password, password, password from users --

202cb962ac59075b964b07152d234b70

Searching Result For ' union select

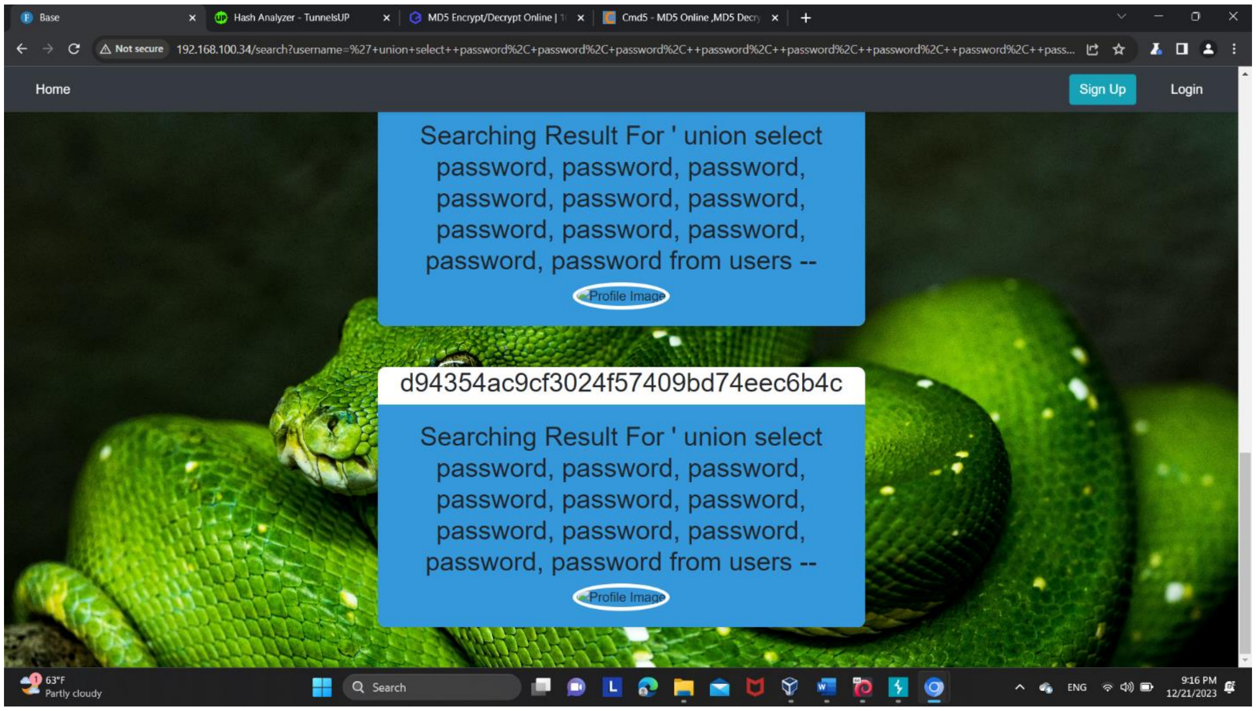
6.7 Cryptographic

Description:	This vulnerability involves weak or inappropriate hashing practices. When a hashed password is easily reversible back to its original plaintext form, it indicates a vulnerability in the hashing algorithm used or in the storage of the hashed passwords. Attackers can take advantage of this vulnerability by obtaining the hashed passwords and decoding them to retrieve the original passwords.
Recommendations	• Use strong hashing algorithms: Employ robust, cryptographic hashing algorithms for hashing passwords. • Salting: Apply a unique salt for each password before hashing to add an additional layer of security against precomputed hash attacks. • Keep passwords hashed securely: Ensure that hashed passwords are stored in a secure and protected environment, ideally separate from other user information.
Affected Systems	192.168.100.34
Threat Level	Critical

Steps to reproduce: -

1- In above vulnerability (union SQL injection) after we got user's hashed passwords.
2- Then, take this hashed value of password same 'd94354ac9cf3024f57409bd74eec6b4c'.
3- Going to any hash analyser website to know which type of hashing used.
4- We get that the algorithm that used in hashing passwords is MD5.
5- Then we go to any website that decodes the hashed values which hashed by MD5.
6- We successfully get the password of the user and it is value is 'adminadminadmin'.

Screenshots

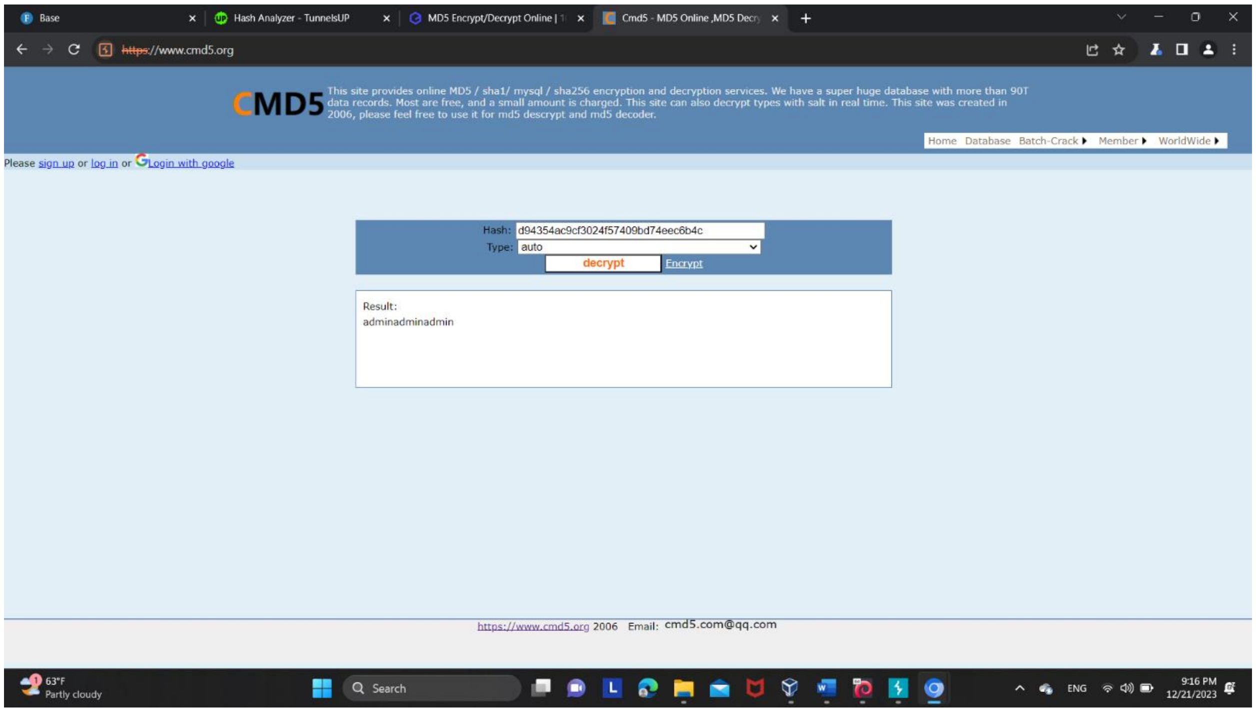


192.168.100.34/search?username=%27+union+select++password%2C+password%2C+password%2C++password%2C++password%2C++password%2C++password%2C++pass...

Searching Result For ' union select password, password, password, password, password, password, password, password, password, password, password from users --

d94354ac9cf3024f57409bd74eec6b4c

Searching Result For ' union select password, password, password, password, password, password, password, password, password, password, password from users --



https://www.cmd5.org

This site provides online MD5 / sha1/ mysql / sha256 encryption and decryption services. We have a super huge database with more than 90T data records. Most are free, and a small amount is charged. This site can also decrypt types with salt in real time. This site was created in 2006, please feel free to use it for md5 descrypt and md5 decoder.

Hash: d94354ac9cf3024f57409bd74eec6b4c
Type: auto

Result:
adminadminadmin

https://www.cmd5.org 2006 Email: cmd5.com@qq.com

6.8 Broken Access Control

Description:	This vulnerability represents a Broken Access Control issue within the website's functionality. By accessing the browser's developer tools (cookies), inspecting the application, and modifying the value of the 'role' attribute from 'user' to 'admin' within the cookies, the user can elevate their privileges and gain unauthorized access to higher privileged roles. This indicates a flaw in the access control mechanisms where user roles can be manipulated on the client-side, bypassing proper authorization checks.
Recommendations	• Except for public resources, deny access by default. • Principle of least privilege: Assign users the minimum permissions necessary for their functionalities and roles. • Access control checks should always be performed on the server side.
Affected Systems	192.168.100.34 (cookies)
Threat Level	High

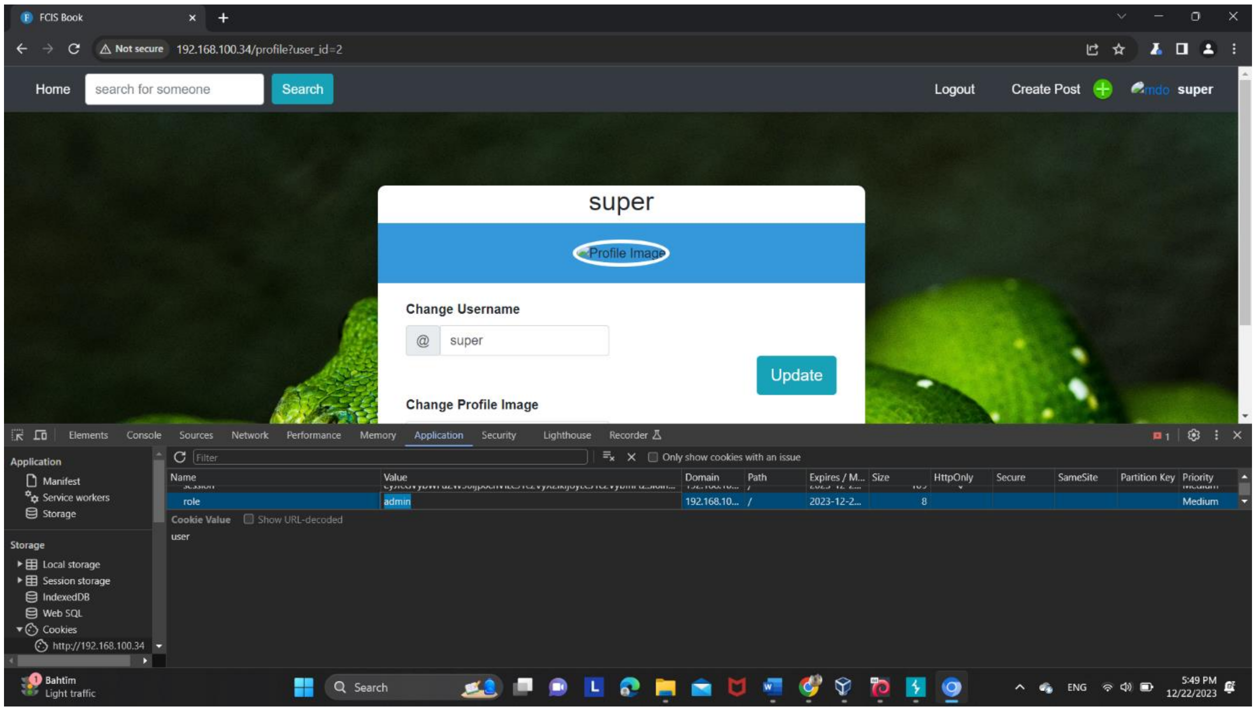
Flag:

FLAG{ctFzJaktJPI1aBHn7Kp9XUm/iH1FybOOO/VeIBoACUBHm6/xzpxeyUArplG+VwGnXsXjx
mhanq+EpjQfURVwelcwrRshkjCTjsEZFy9qc3U=}

Steps to reproduce: -

1- After logged in as a user in a website.
2- Right click on a website, then click on inspect.
3- Choose cookies section.
4- Change the value of role attribute from user to admin.
5- Refresh the website, then you get the admin privileges and you can pretend now as an admin.

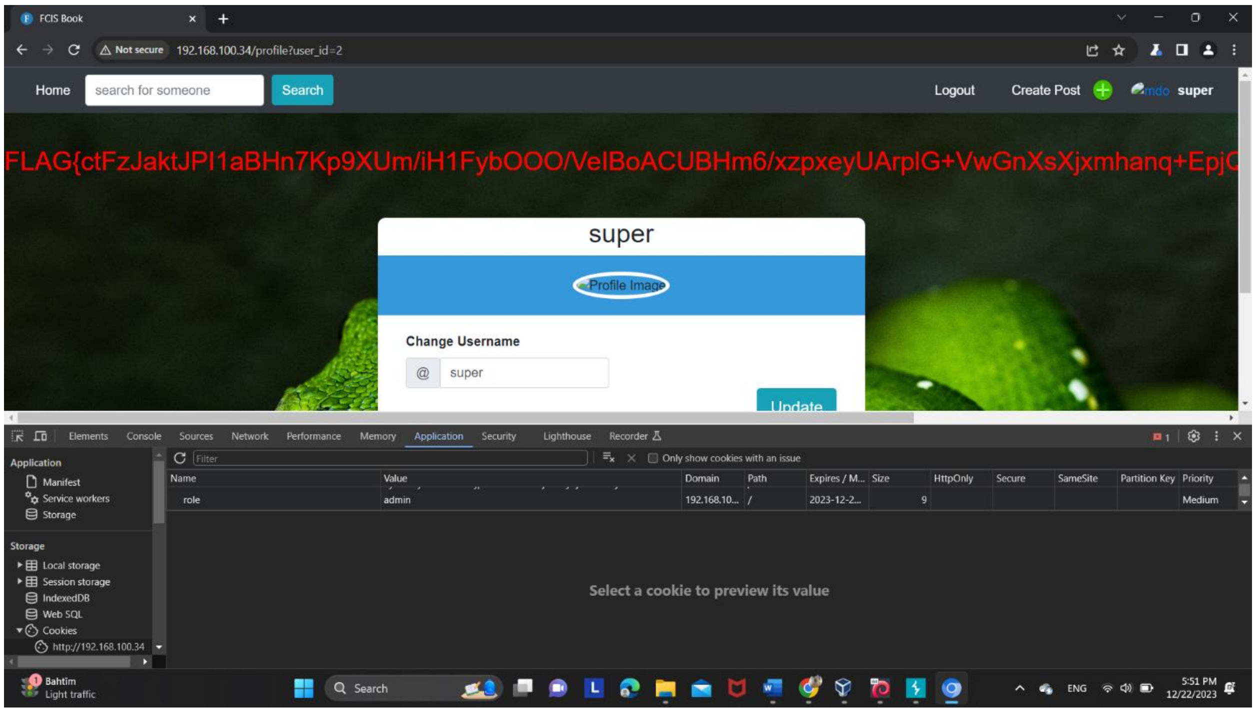
Screenshots


192.168.100.34/profile?user_id=2

Name	Value	Domain	Path	Expires / M...	Size	HttpOnly	Secure	SameSite	Partition Key	Priority
role	admin	192.168.10...	/	2023-12-2...	8					Medium

Cookie Value

user


192.168.100.34/profile?user_id=2

FLAG{ctFzJaktJPI1aBHn7Kp9XUm/iH1FybOOO/VelBoACUBHm6/xzpxeyUArplG+VwGnXsXjxmhanq+EpjC

Name	Value	Domain	Path	Expires / M...	Size	HttpOnly	Secure	SameSite	Partition Key	Priority
role	admin	192.168.10...	/	2023-12-2...	9					Medium

Select a cookie to preview its value

6.9 IDOR

Description:	This vulnerability is an Insecure Direct Object Reference (IDOR) flaw within the website's functionality. By manipulating the 'user_id' parameter in the URL while accessing the profile page (changing from 'user_id=2' to 'user_id=3' or another number), an unauthorized user can view the profile of other users. This vulnerability arises from insufficient or improper authorization checks that allow direct referencing of internal objects (such as user IDs) by users who shouldn't have access to them.
Recommendations	• Implement proper authorization checks: Ensure that user access to resources is verified and authorized on the server-side, not solely relying on client-side controls or direct object references in URLs. • Principle of least privilege: Assign users the minimum permissions necessary for their functionalities and roles.
Affected Systems	192.168.100.34 (user_id in URL)
Threat Level	High

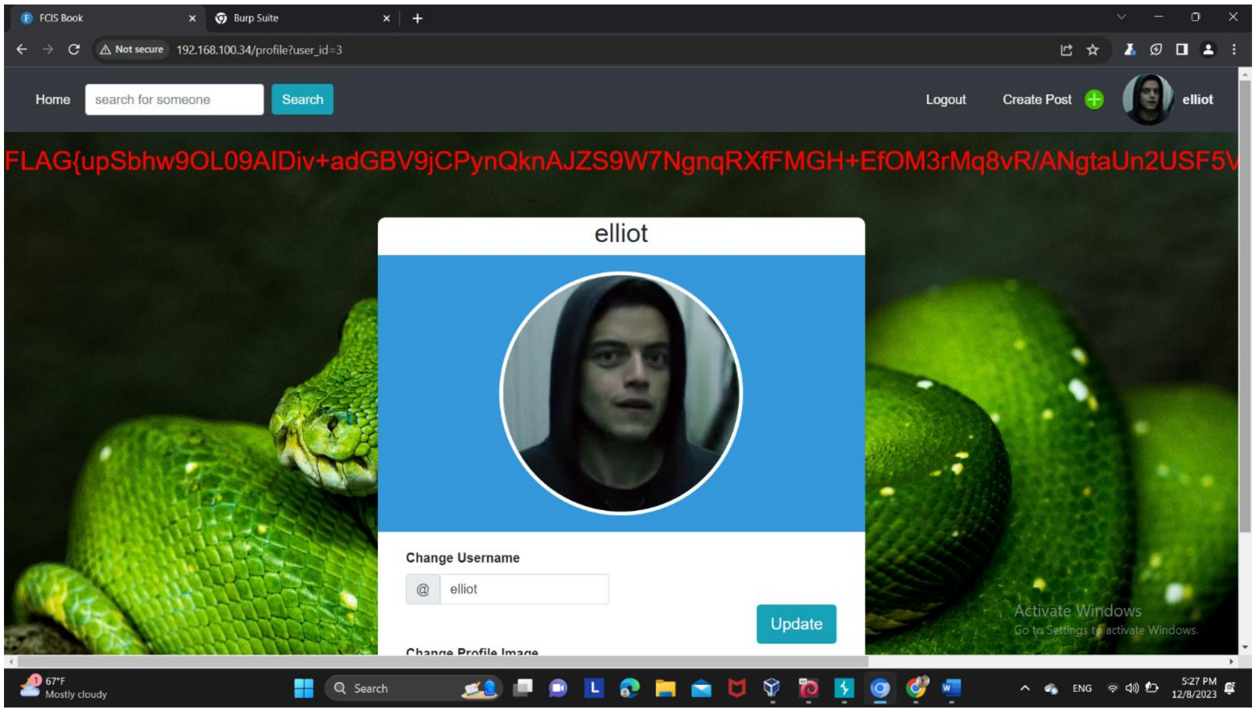
***Flag*:**

FLAG{upSbhw9OL09AIDiv+adGBV9jCPynQknAJZS9W7NgnqRXfFMGH+EfOM3rMq8vR/ANgt aUn2USF5VjiMG8isVhT78WODs7Fsqwu66hoEmwpnk=}

Steps to reproduce: -

1- After logged in with your email in a website.
2- Go to your profile page.
3- In the URL: http://192.168.100.34/profile?user_id=2 .
4- Change the value that takes by user_id parameter from user_id=2 to user_id=3 or any other number.
5- Go to this modified URL, then you get the access of this user page.

Screenshots



```
192.168.100.34/profile?user_id=3
FLAG{upSbhw9OL09AIDiv+adGBV9jCPynQknAJZS9W7NgnqRXfFMGH+EfOM3rMq8vR/ANgtaUn2USF5V
```

6.10 Server-Side Request Forgery (SSRF)

Description:	The vulnerability identified is a Server-Side Request Forgery (SSRF) within the website. By intercepting the request made when clicking the 'Check Online Status' button using Burp Suite, an attacker modifies the URL within the intercepted request to point to url=http://localhost. Upon forwarding this modified request, the website displays the content of the localhost site. This behaviour indicates that the website is vulnerable to SSRF attacks, allowing attackers to make unauthorized requests to internal or external systems accessible by the server.
Recommendations	• Input validation: Validate and sanitize user-provided URLs to ensure they point only to safe and authorized locations. • Use network-level protections: Implement network-level protections like firewalls or proxies to restrict access to internal resources. • Enforce the URL schema, port, and destination with a positive allow list. • Do not send raw responses to clients.
Affected Systems	192.168.1.1 (Check Online Status Button)
Threat Level	High

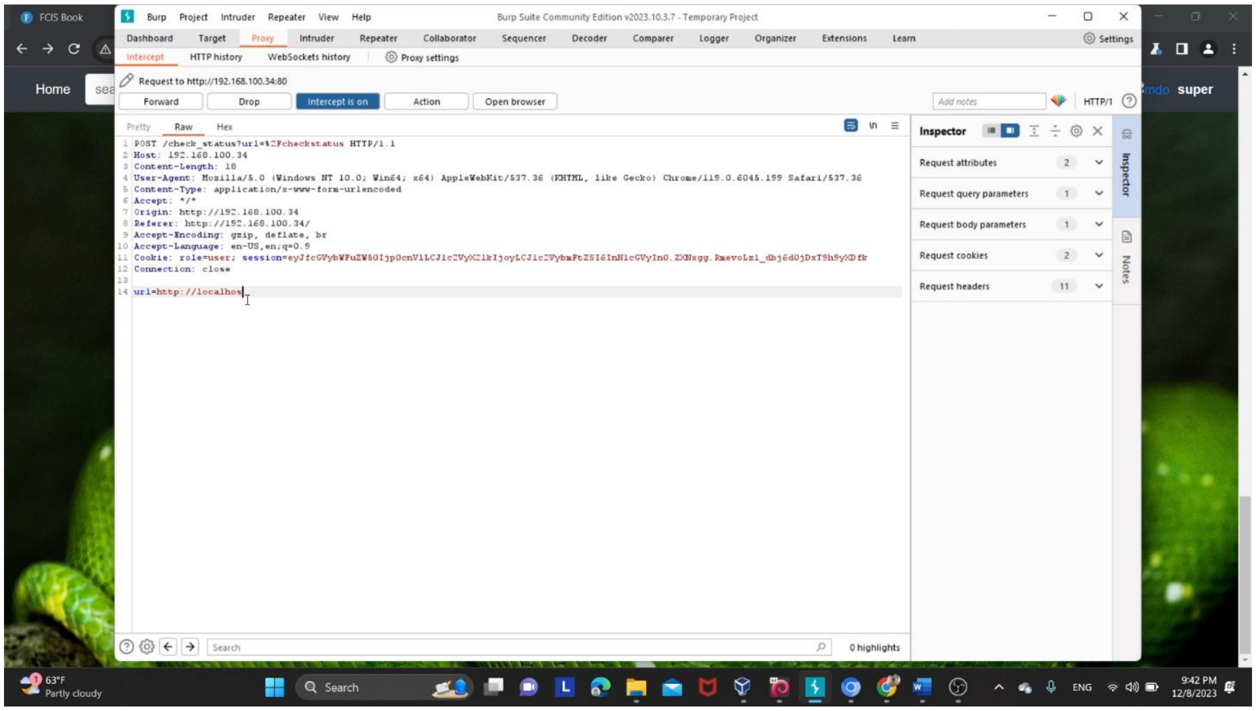
***Flag*:**

FLAG{YRbr7NNEdpvi9OlnV1Oc7BgXF8ibVWScr5j8izuOCOaL6l9r7y9atu9ymnt0Uci38oXTTOV
x1tuFEHW2piRVRTiabxcQQBuG+sLNn+mFTbU=}

Steps to reproduce: -

1- Open website home page.
2- Go to open Burp Suite, then open the intercept to intercept the request.
3- Click on the ‘Check Online Status’ button on the website.
4- Return to the Burp suite to the intercepted request, then change the URL.
5- Change the URL from url=%2Fcheckstatus to url=http://localhost
6- Then, forward the request and go to website you get that it return to you the content of localhost site instead show the status.

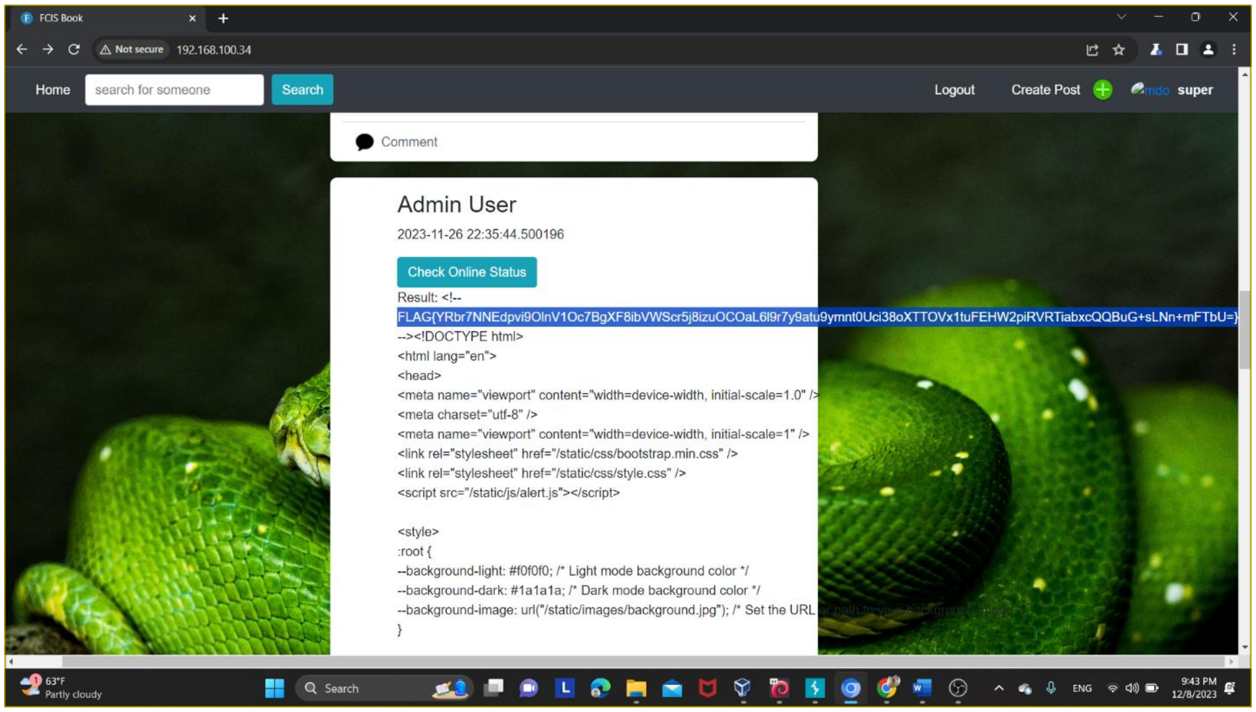
Screenshots



Request to http://192.168.100.34:80

```
POST /check_status?url=%2Fcheckstatus HTTP/1.1
Host: 192.168.100.34
Content-Length: 18
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.199 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: */*
Origin: http://192.168.100.34
Referer: http://192.168.100.34/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: role=user; session=eyJfcGVybWFuZW50IjpOcnVlLCJ1c2VyX2lkIjoyLCJ1c2VybmFtZSI6InNlcGVyIn0.ZXNxgg.RmevoLzl_dbj6d0jDxT9h9yXDfk
Connection: close

url=http://localhos
```



Admin User

2023-11-26 22:35:44.500196

Check Online Status

```
Result: <!--
FLAG{YRbr7NNEdpvi9OlnV1Oc7BgXF8ibVWScr5j8izuOCOaL6l9r7y9atu9ymnt0Uci38oXTTOVx1tuFEHW2piRVRTiabxcQQBuG+sLNn+mFTbU=}
--><!DOCTYPE html>
<html lang="en">
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<link rel="stylesheet" href="/static/css/bootstrap.min.css" />
<link rel="stylesheet" href="/static/css/style.css" />
<script src="/static/js/alert.js"></script>

<style>
:root {
--background-light: #f0f0f0; /* Light mode background color */
--background-dark: #1a1a1a; /* Dark mode background color */
--background-image: url("/static/images/background.jpg"); /* Set the URL
}
```

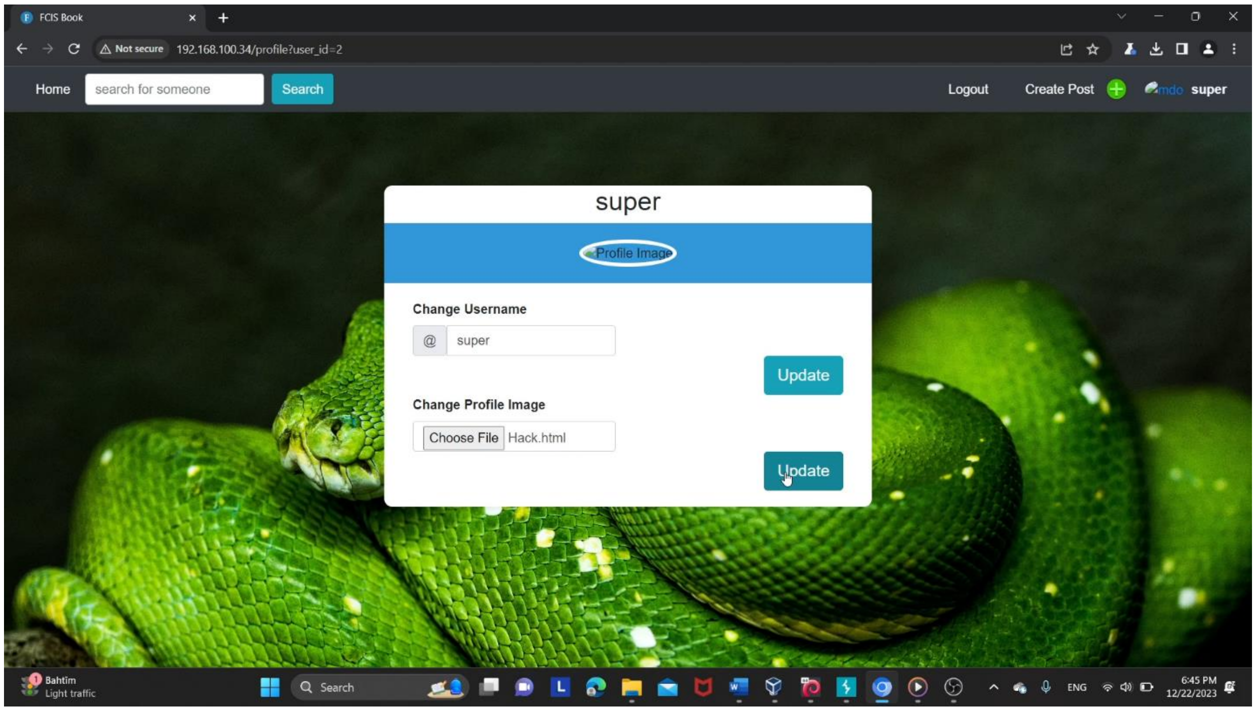
6.11 File Upload

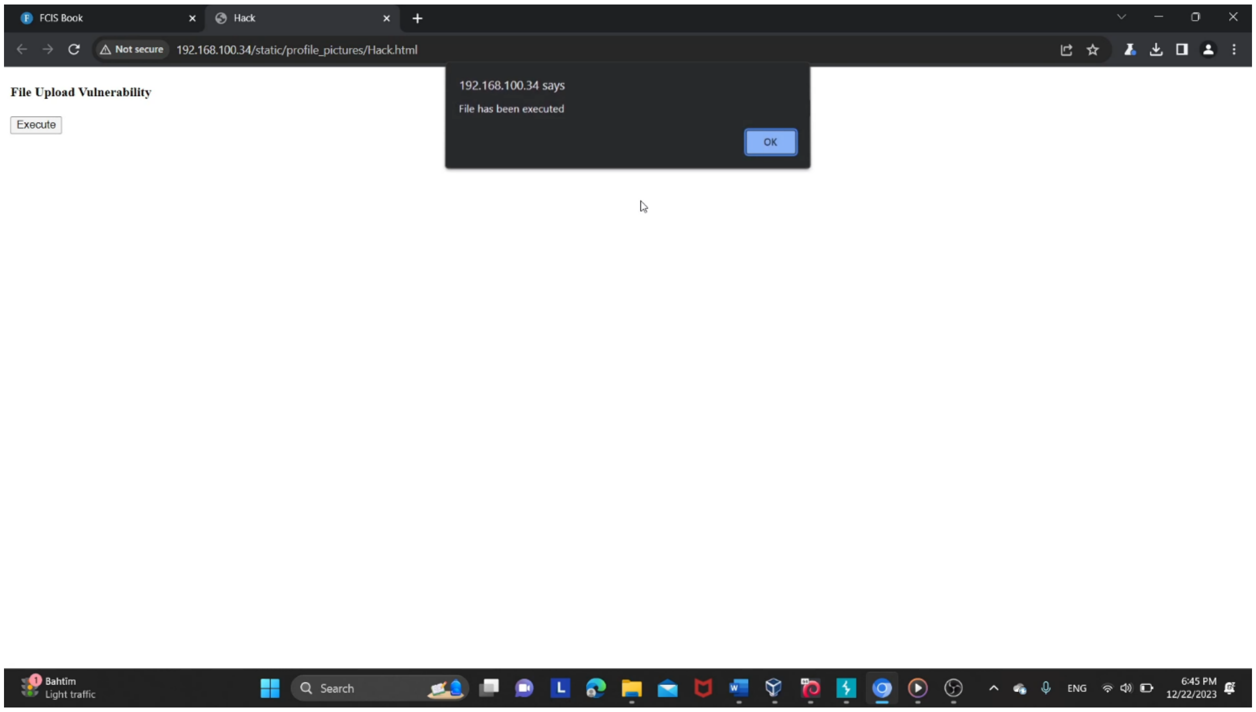
Description:	This vulnerability occurs when a web application fails to properly validate or sanitize the file type and content uploaded by users. Attackers can exploit this vulnerability by uploading malicious files instead of uploading their images (such as scripts or executables) disguised as harmless files (like images), leading to potential security risks.
Recommendations	• whitelist of permitted extensions rather than a blacklist of prohibited ones. • Restrict Upload Locations: Store uploaded files in a separate directory and prevent execution permissions on that directory to limit the potential impact of uploaded malicious files. • Makes sure the filename doesn't contain any substrings that may be interpreted as a directory or a traversal sequence (../). • As much as possible, use an established framework for preprocessing file uploads rather than attempting to write your own validation mechanisms.
Affected Systems	192.168.100.34 (profile image)
Threat Level	Critical

Steps to reproduce: -

1- First, I create a html file that have a button when I click on it, it execute a JavaScript code.
2- After logged in a website.
3- Go to your profile page.
4- Click on chose file in section change profile image.
5- Choose your malicious file you want to upload to the system.
6- Then, click on update to upload the file.
7- When you right click on profile image. Then, click on open image in new tab.
8- Now you can access your file and execute it.

Screenshots




192.168.100.34/static/profile_pictures/Hack.html

File Upload Vulnerability

Execute

192.168.100.34 says

File has been executed

OK

6.12 Interacting with HTTP requests & responses Curl (OPTIONS)

Description:	HTTP OPTIONS Method Information Disclosure: The server's response to an HTTP OPTIONS request includes information about the supported HTTP methods (GET, POST, PUT, DELETE) and certain headers like Access-Control-Allow-Methods, Access-Control-Allow-Headers, and Content-type. Revealing this information might provide insights into the server's configuration and capabilities, which could be leveraged by attackers to plan potential attacks.
Recommendations	• Minimize Information Disclosure: Configure the server to restrict the details disclosed in response to an OPTIONS request to minimize exposure of sensitive information. • Access Control: Implement proper access controls and permissions to limit access to sensitive information based on the principle of least privilege. • HTTP Header Configuration: Review and modify server configurations to limit the information exposed through HTTP headers.
Affected Systems	192.168.100.34
Threat Level	Informational

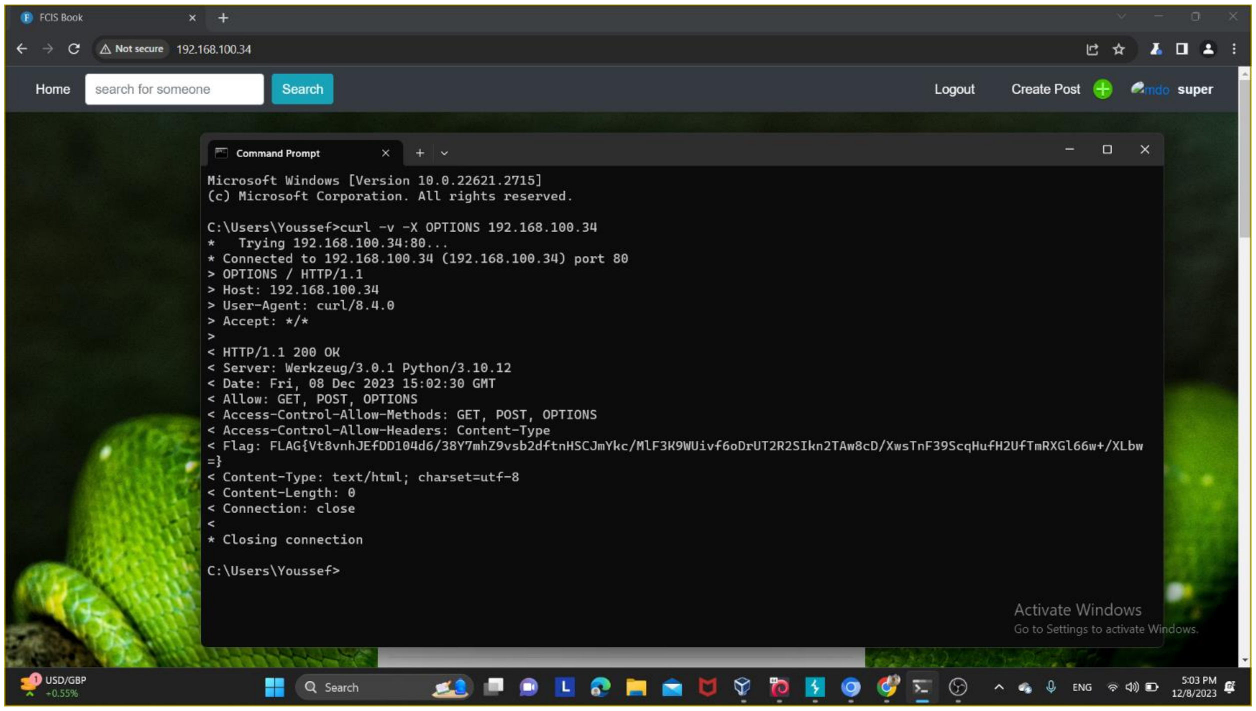
Flag:

FLAG{Vt8vnhJEfDD104d6/38Y7mhZ9vsb2dftnHSCJmYkc/MlF3K9WUivf6oDrUT2R2SIkn2TAw
8cD/XwsTnF39ScqHufH2UfTmRXGl66w+/XLbw=}

Steps to reproduce: -

1- First, open the command prompt or anything to write the command.
2- Write curl command: curl -v -X OPTIONS IP same curl -v -X OPTIONS 192.168.100.34
3- Then, click enter to execute the command.
4- It will return you the Allowed-methods, Access-Control-Allow-Methods, Access-Control-Allow-Headers, Content-Type and some information about system.

Screenshots



```
Microsoft Windows [Version 10.0.22621.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Youssef>curl -v -X OPTIONS 192.168.100.34
*   Trying 192.168.100.34:80...
* Connected to 192.168.100.34 (192.168.100.34) port 80
> OPTIONS / HTTP/1.1
> Host: 192.168.100.34
> User-Agent: curl/8.4.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: Werkzeug/3.0.1 Python/3.10.12
< Date: Fri, 08 Dec 2023 15:02:30 GMT
< Allow: GET, POST, OPTIONS
< Access-Control-Allow-Methods: GET, POST, OPTIONS
< Access-Control-Allow-Headers: Content-Type
< Flag: FLAG{Vt8vnhJEfDD104d6/38Y7mhZ9vsb2dftnHSCJmYkc/MlF3K9WUivf6oDrUT2R2SIkn2TAw8cD/XwsTnF39ScqHufH2UfTmRXGl66w+/XLbw
=}
< Content-Type: text/html; charset=utf-8
< Content-Length: 0
< Connection: close
<
* Closing connection

C:\Users\Youssef>
```

6.13 HTTP Method Tampering

Description:	This vulnerability occurs when an attacker can modify the HTTP request method from its original state (changing from POST to GET or another) using tools like Burp Suite. Applications often rely on specific HTTP methods to perform various actions (POST for submitting data, GET for retrieving data). By tampering with the method, an attacker may attempt to bypass security controls, access unauthorized resources, or manipulate the application's behaviour.
Recommendations	• Use Correct HTTP Methods: Ensure that the application uses appropriate HTTP methods for specific actions (POST for data submission, GET for data retrieval) and enforce these methods at the server-side. • Input Validation and Access Controls: Implement strong input validation and access controls to prevent unauthorized access or manipulation of data. • HTTP Method Restrictions: Configure the server or application to reject unexpected or unauthorized HTTP methods.
Affected Systems	192.168.100.34 (Login page)
Threat Level	High

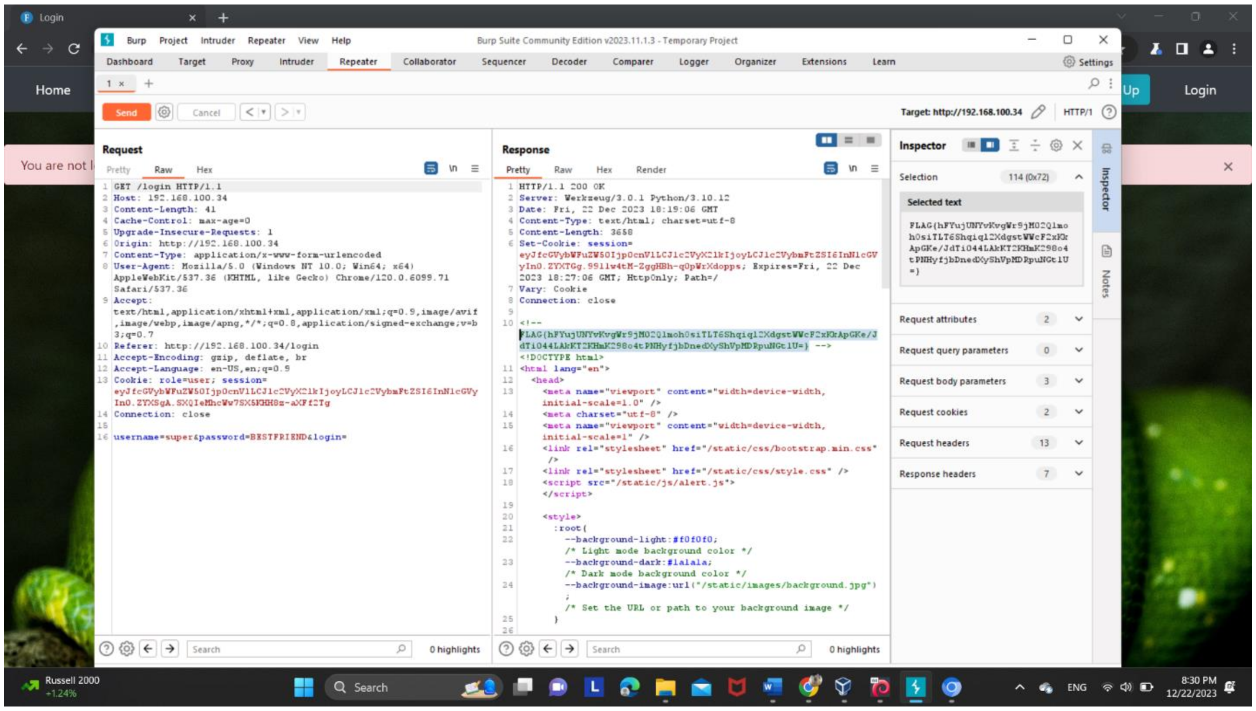
Flag:

FLAG{hFYujUNYvKvgWr9jM02Q1mohOsiTLT6Shqiql2XdgstWWcF2xKkApGKe/JdTi044LAkKT
2KHmK298o4tPNHyfjbDnedXyShVpMDRpuNGt1U=}

Steps to reproduce: -

1- First, Open Burp suite to show requests and responses.
2- When you login to account go to Burp Suite to login request.
3- Then, send this request to repeater to modify on it.
4- Change POST method to GET method and send the request.

Screenshots



Request

```
GET /login HTTP/1.1
Host: 192.168.100.34
Content-Length: 41
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://192.168.100.34
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.71
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif
,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b
3;q=0.7
Referer: http://192.168.100.34/login
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: role=user; session=
eyJfcGVybWFuZW50IjpOcnVlLCJlc2VyX2lkIjoyLCJlc2VybmFtZSI6InNlcGVy
In0.ZYXSgA.SXQIeMhcWw7SX5KHH8z-aXFf2Tg
Connection: close

username=super&password=BESTFRIEND&login=
```

Response

```
HTTP/1.1 200 OK
Server: Werkzeug/3.0.1 Python/3.10.12
Date: Fri, 22 Dec 2023 18:19:06 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 3658
Set-Cookie: session=
eyJfcGVybWFuZW50IjpOcnVlLCJlc2VyX2lkIjoyLCJlc2VybmFtZSI6InNlcGV
yIn0.ZYXTGg.991lw4tM-ZggHBh-q0pWrXdopps; Expires=Fri, 22 Dec
2023 18:27:06 GMT; HttpOnly; Path=/
Vary: Cookie
Connection: close

<!--
FLAG(hFYujUNYvKvgWr9jM02Q1moh0siTLT6Shqiql2XdgstWWcF2xKrApGKe/J
dTi044LAkKT2KHmK298o4tPNHyfjbDnedXyShVpMDRpuNGt1U=) -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0" />
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,
    initial-scale=1" />
    <link rel="stylesheet" href="/static/css/bootstrap.min.css"
     />
    <link rel="stylesheet" href="/static/css/style.css" />
    <script src="/static/js/alert.js">
    </script>

    <style>
      :root{
        --background-light:#f0f0f0;
        /* Light mode background color */
        --background-dark:#1a1a1a;
        /* Dark mode background color */
        --background-image:url("/static/images/background.jpg")
        ;
        /* Set the URL or path to your background image */
      }
```

Inspector

Selection 114 (0x72)

Selected text

```
FLAG(hFYujUNYvKvgWr9jM02Q1mo
h0siTLT6Shqiql2XdgstWWcF2xKr
ApGKe/JdTi044LAkKT2KHmK298o4
tPNHyfjbDnedXyShVpMDRpuNGt1U
=)
```

Request attributes	2
Request query parameters	0
Request body parameters	3
Request cookies	2
Request headers	13
Response headers	7

6.14 Directory/Path Traversal

Description:	This vulnerability allows an attacker to navigate in the intended directory structure by manipulating input, typically by appending "../" or similar sequences to the file path in a URL or parameter. In this case, changing the file parameter from admin-user.jpg to secret.txt allowed access to a file outside the designated directory, potentially revealing sensitive information.
Recommendations	• he best way to prevent directory traversal vulnerabilities is to avoid passing user-supplied input to filesystem APIs. • Input Validation and Sanitization: Implement strict input validation and sanitization for file paths or any user-supplied input to prevent directory traversal attempts. • File Access Controls: Ensure that the application's access controls and permissions restrict access to sensitive files or directories.
Affected Systems	192.168.100.34
Threat Level	High

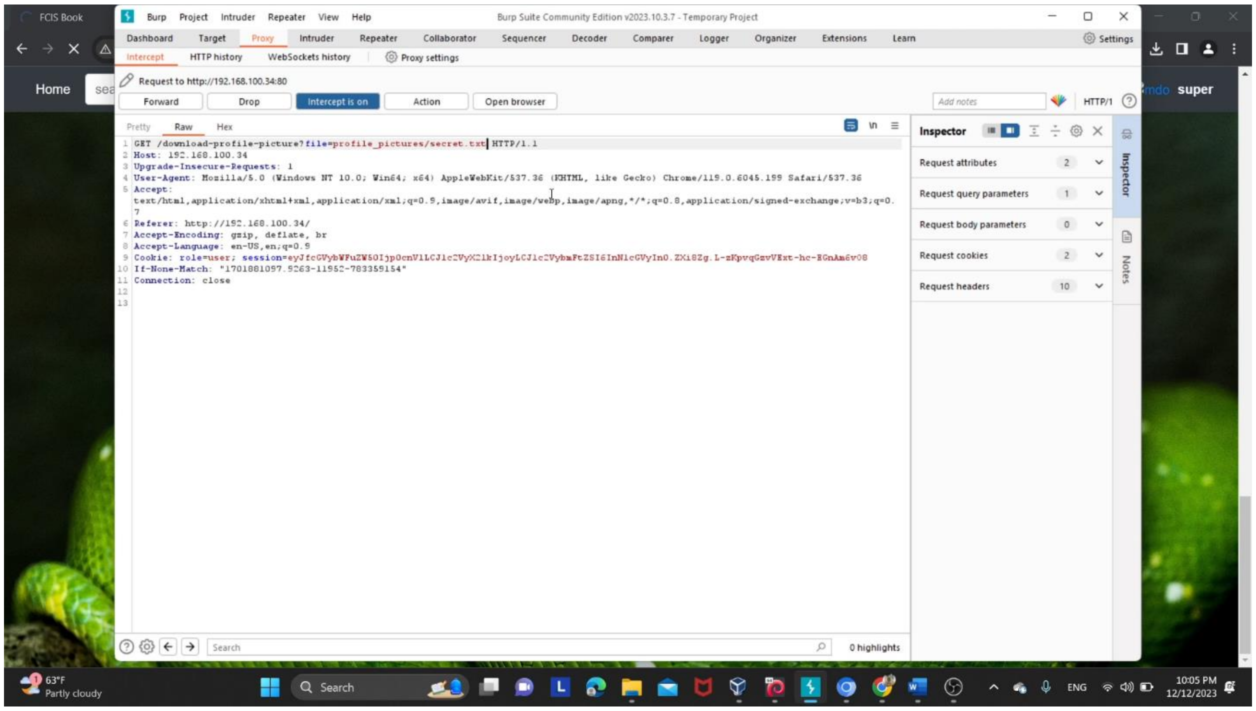
Flag:

FLAG{aCkp/QKD3taNiv36OukKPL3R7jrqkn5ymXuXBf1Pff32QrwjV3xbgnuoX8vBCInKozQdHMU9GI/zeFKu3LVf6Q==}

Steps to reproduce: -

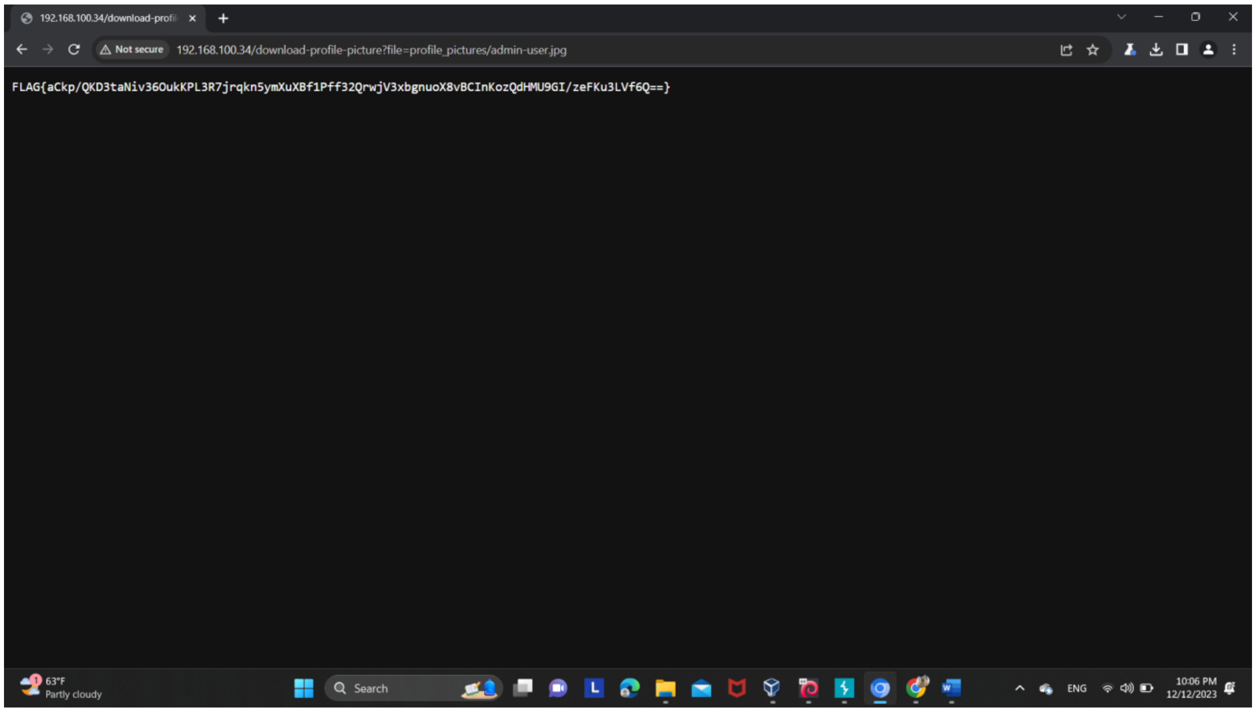
1- From reading the manual of the system we got that there is a file its name is “secret.txt”.
2- First, Open Burp suite to intercept request.
3- login to account go to home page.
4- Open intercept in Burp suite to intercept the request.
5- Then, click on admin image to access(download) it, Then, go to Burp suite to the intercept.
6- Change the GET request from GET /download-profile-picture?file=profile_pictures/admin-user.jpg to GET /download-profile-picture?file=profile_pictures/secret.txt.
7- Then, forward the request so you will get access to this file.

Screenshots


Burp Suite Community Edition v2023.10.3.7 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Proxy settings
Forward Drop Intercept is on Action Open browser
Inspector

Request to http://192.168.100.34:80

```
GET /download-profile-picture?file=profile_pictures/secret.txt HTTP/1.1
Host: 192.168.100.34
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.199 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.
7
Referer: http://192.168.100.34/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: role=user; session=eyJfcGVybWFuZW50IjpOcnVlLCJlc2VyX2lkIjoyLCJlc2VybmFtZSI6InNlcGVyIn0.ZXi8Zg.L-zKpvqGzvVExt-hc-EGnAm6v08
If-None-Match: "1701881097.9263-11952-783359154"
Connection: close
```



192.168.100.34/download-profile-picture?file=profile_pictures/admin-user.jpg

```
FLAG{aCkp/QKD3taNiv36OukKPL3R7jrqkn5ymXuXBf1Pff32QrwjV3xbgnuoX8vBCInKozQdHMU9GI/zeFKu3LVf6Q==}
```

6.15 Robots.txt File Exposure

Description:	The robots.txt file is a text file placed in the root directory of a website, specifying which parts of the site they are allowed or disallowed to access. Accessing this file directly through the browser by appending /robots.txt to the website's IP or domain reveals the contents of this file.
Recommendations	• Review Content of robots.txt: Ensure that the information in the robots.txt file doesn't expose sensitive directories, files, or information that could aid attackers. • Security: Relying solely on robots.txt to protect sensitive information is not secure. Implement proper access controls and security measures to protect critical data. • Regular Review: Periodically review and update the robots.txt file as the website's content and structure change.
Affected Systems	192.168.100.34
Threat Level	Informational

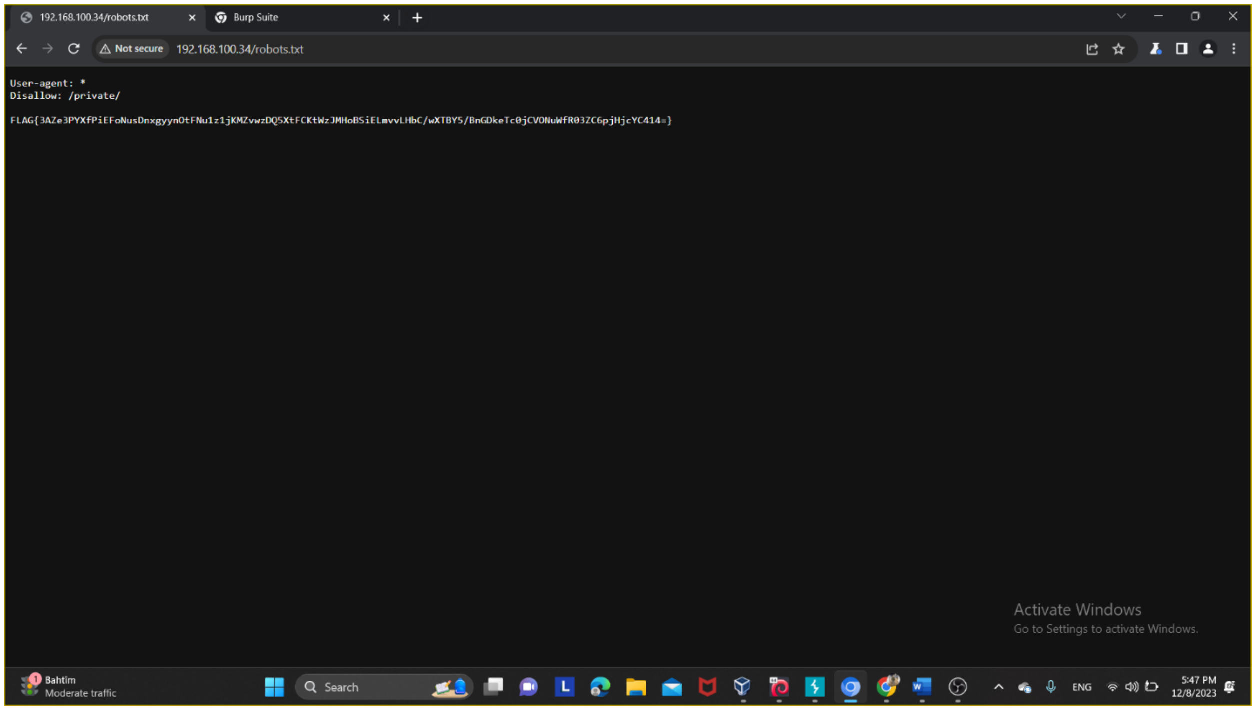
Flag:

FLAG{3AZe3PYXfPiEFoNusDnxgyynOtFNu1z1jKMZvwzDQ5XtFCKtWzJMHoBSiELmvvLHbC/wXTBY5/BnGDkeTc0jCVONuWfR03ZC6pjHjcYC414=}

Steps to reproduce: -

1- After IP address of the website append file robots.txt
2- Then, go to this directory.
3- It opens the robots.txt file.

Screenshots



```
User-agent: *
Disallow: /private/

FLAG{3AZe3PYXfPiEFoNusDnxgyynOtFNu1z1jKMZvwzDQ5XtFCKtWzJMHoBSiELmvvLHbC/wXTBY5/BnGDkeTc0jCVONuWfR03ZC6pjHjcYC414=}
```

6.16 Sensitive Information Disclosure (in Html code)

Description:	When inspecting the HTML code of the login page, sensitive information, in this case, a content flag, is directly visible within the code, indicating a successful action or secret data. However, in a production environment, the exposure of such sensitive information within HTML code can pose a security risk.
Recommendations	• Avoid Hard-Coded Secrets: Ensure that sensitive information, such as flags, credentials, or tokens, is not hard-coded within the HTML code, as this could potentially be accessed by unauthorized individuals. • Secure Storage: Store sensitive data in secure locations such as server-side configurations, encrypted databases, or secure vaults, rather than directly embedding them within HTML or client-side scripts.
Affected Systems	192.168.100.34 (Login page)
Threat Level	Moderate

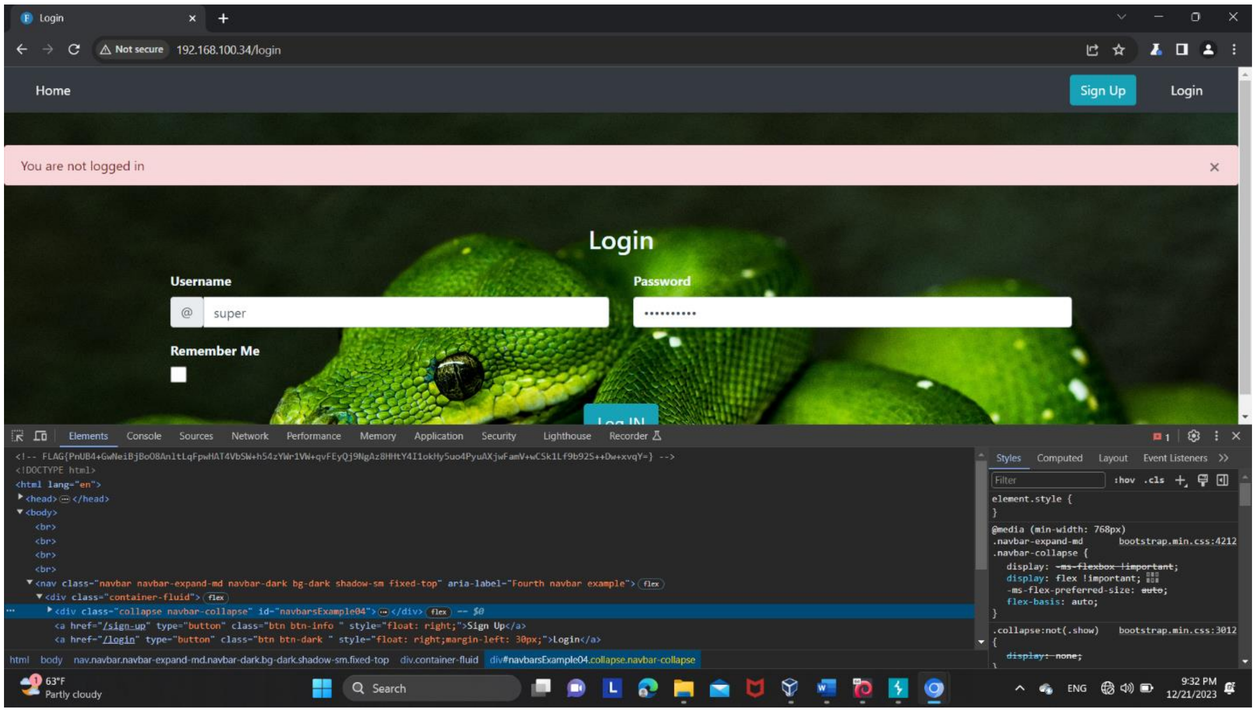
***Flag*:**

FLAG{PnUB4+GwNeiBjBoO8AnltLqFpwHAT4VbSW+h54zYWr1VW+qvFEyQj9NgAz8HHtY4I1o
kHy5uo4PyuAXjwFamV+wCSk1Lf9b92S++Dw+xvqY=}

Steps to reproduce: -

1- Open login page.
2- Then, right click on the page, Then, click on inspect.
3- We get a Flag in html code.

Screenshots


Login
Not secure 192.168.100.34/login
Home
Sign Up
Login
Username
Password
Remember Me
Log IN
Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder
Styles Computed Layout Event Listeners

You are not logged in

Login

Username: super

```
<!-- FLAG{PnUB4+GwNeiBjBoO8AnltLqFpwHAT4VbSW+h54zYWr1VW+qvFEyQj9NgAz8HHtY4I1okHy5uo4PyuAXjwFamV+wCSk1Lf9b92S++Dw+xvqY=} -->
<!DOCTYPE html>
<html lang="en">
 ▶<head> ⋯ </head>
 ▼<body>
   <br>
   <br>
   <br>
   <br>
  ▼<nav class="navbar navbar-expand-md navbar-dark bg-dark shadow-sm fixed-top" aria-label="Fourth navbar example"> flex
   ▼<div class="container-fluid"> flex
     ▶<div class="collapse navbar-collapse" id="navbarsExample04"> ⋯ </div> flex == $0
      <a href="/sign-up" type="button" class="btn btn-info " style="float: right;">Sign Up</a>
      <a href="/login" type="button" class="btn btn-dark " style="float: right;margin-left: 30px;">Login</a>
html  body  nav.navbar.navbar-expand-md.navbar-dark.bg-dark.shadow-sm.fixed-top  div.container-fluid  div#navbarsExample04.collapse.navbar-collapse
```

```
element.style {
}
@media (min-width: 768px)
.navbar-expand-md        bootstrap.min.css:4212
.navbar-collapse {
   display: -ms-flexbox !important;
   display: flex !important;
   -ms-flex-preferred-size: auto;
   flex-basis: auto;
}
.collapse:not(.show)     bootstrap.min.css:3012
{
   display: none;
```

63°F Partly cloudy Search ENG 9:32 PM 12/21/2023

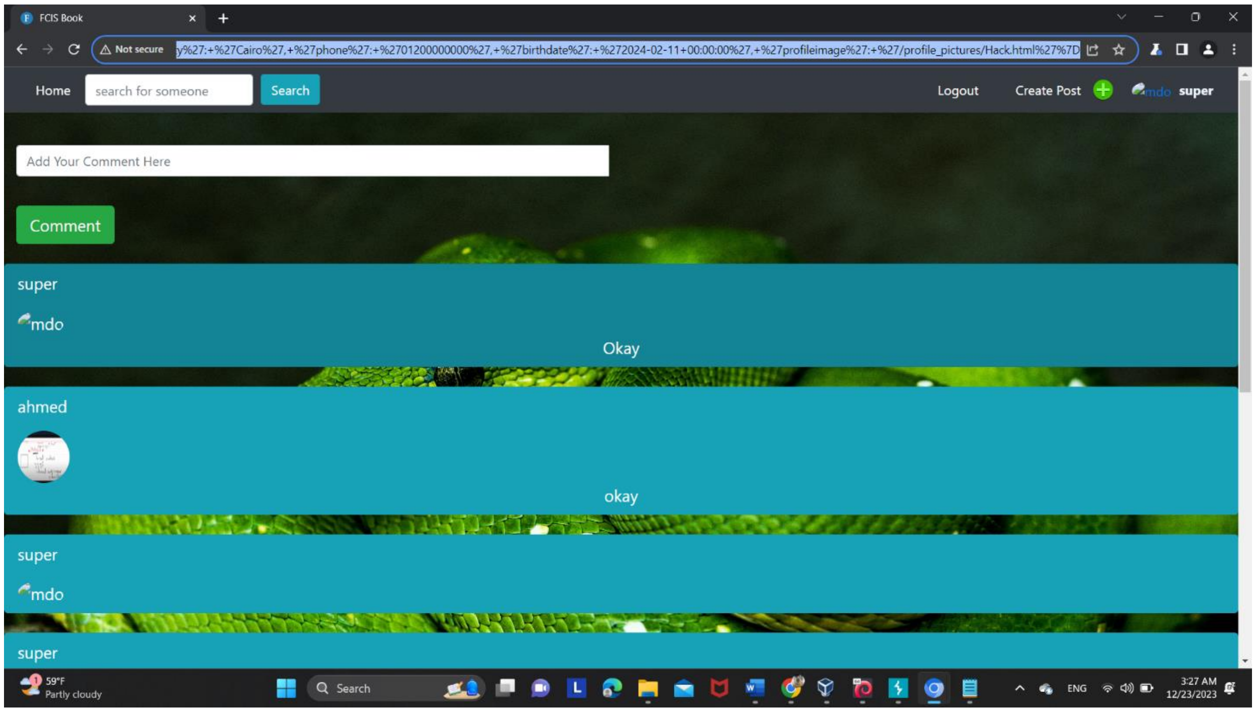
6.17 User Information Disclosure via URL

Description:	The URL seems to indicating that the application is vulnerable to that URL includes detailed information about users who have interacted with the site, revealing user IDs, usernames, email addresses, passwords (hashed), and profile image paths.
Recommendations	• Output Encoding: Properly encode user input before displaying it in URLs or web pages to prevent script execution. • Minimize Data Exposure: Avoid exposing sensitive user details through URLs or any client-side operations. Only share necessary information and limit the exposure of sensitive data.
Affected Systems	192.168.100.34 (Comments URL)
Threat Level	Critical

Steps to reproduce: -

1- Login to your email on the website.
2- Then, go to make a comment on a post.
3- When we add a comment we get that this information about user in URL.

Screenshots



y%27:+%27Cairo%27,+%27phone%27:+%2701200000000%27,+%27birthdate%27:+%272024-02-11+00:00:00%27,+%27profileimage%27:+%27/profile_pictures/Hack.html%27%7D

super
mdo
Okay

ahmed
okay

super
mdo

super

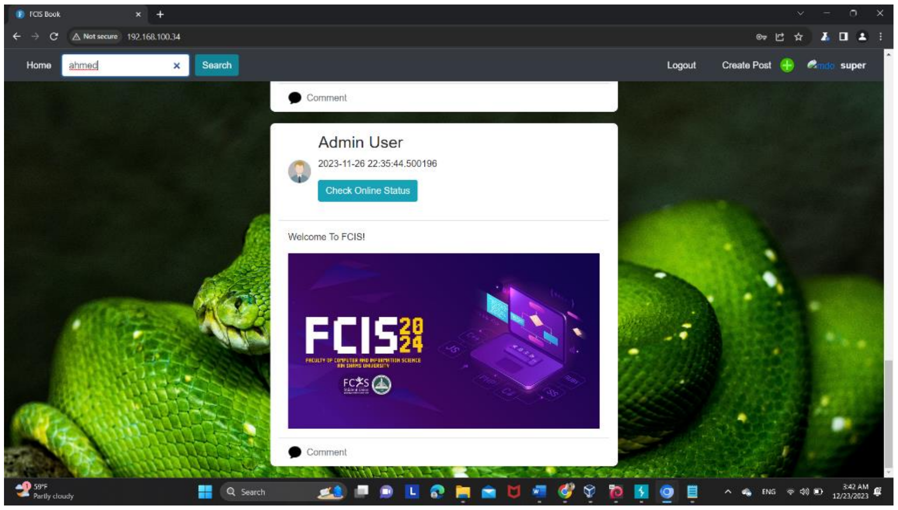
6.18 Access Controls on User Profiles

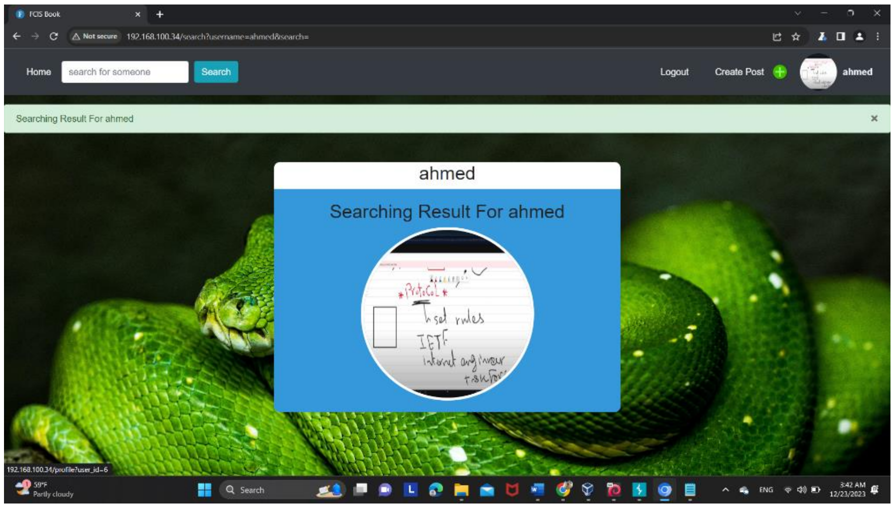
Description:	When performing a search for a user (same Ahmed) and accessing their profile, The application fails to enforce proper access controls. This leads to unauthorized access to Ahmed's profile, allowing actions such as modifying Ahmed's username or photo. This issue allows users to access and manipulate profiles of other users without proper authorization.
Recommendations	• Authentication and Authorization Checks: Ensure that each user action is validated against their authentication and authorization levels before granting access to sensitive functionalities, such as profile modification. • Implement robust session management to ensure that users can only access and modify their own profile unless specifically authorized otherwise.
Affected Systems	192.168.100.34
Threat Level	High

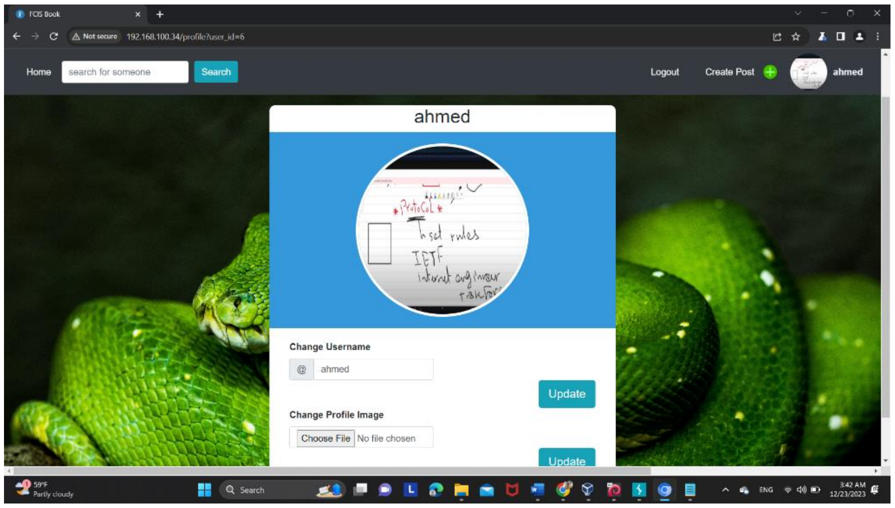
Steps to reproduce: -

1- Login to your email on the website.
2- Then, go to search bar and search about any user.
3- When it returns you the user that you search about, now you can access his profile and change his name and photo.

Screenshots




FCIS Book
Not secure 192.168.100.34/search?username=ahmed&search=
Home
search for someone
Search
Logout
Create Post
ahmed
Searching Result For ahmed
ahmed
Searching Result For ahmed
192.168.100.34/profile?user_id=6



Last Page