**Cybersecurity Project**

It is a project about hacking a site similar to Facebook that contains 18 vulnerabilities from the OASP (which talks about the most famous vulnerabilities) and they were discovered, and this file contains them all and also contains recommendations to prevent these vulnerabilities from existing.

**Types of vulnerabilities in this project**

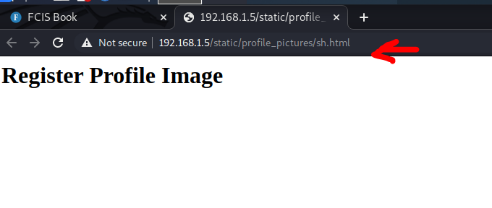
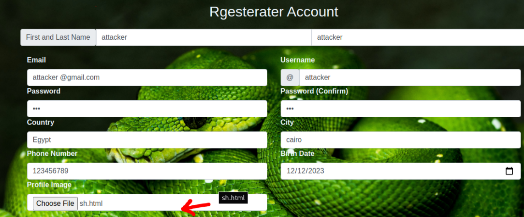
* ***SQL injection (2)***
* ***XSS (3)***
* ***File upload(2)***
* ***Broken Access controls(2)***
* ***SSRF(1)***
* ***Cryptographic Failure(1)***
* ***Method Tampering(1)***
* ***Directory Traversal(1)***
* ***Misconfiguration(1)***
* ***Brute Force(1)***
* ***Using Curl(1)***
* ***Using inspect(1)***
* ***Using Robots.txt(1)***

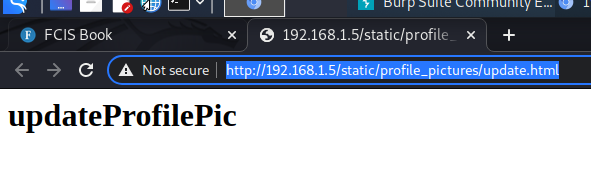
# **File upload vulnerability (2 Vuln.)**

**☞1\ Malicious File Execution.**

|  |  |
| --- | --- |
| Description: | File upload vulnerability occurs when a website or app lets users upload files without proper checks.  For example, during registration, if a system doesn't verify file types, an attacker might upload a harmful file, like an HTML file disguised as an image. This can lead to executing malicious code on the server, risking its security. |
| Recommendations | * File Type Validation * File Size Limit * Secure File Storage, Store uploaded files in a location isolated from the web root directory and apply strict permissions to prevent execution of uploaded files. |
| Affected Systems | http://192.168.1.5/static/profile\_pictures/sh.html |
| Threat Level | Medium (For us) 🡪 High (For System) |

**Screen Shots:**



A screenshot of a computer

Description automatically generated

**Steps:**

There is the same vulnerability in a lot of places.

**1.. Create HTML file “sh.html” contains <h1>profile pic</h1>**

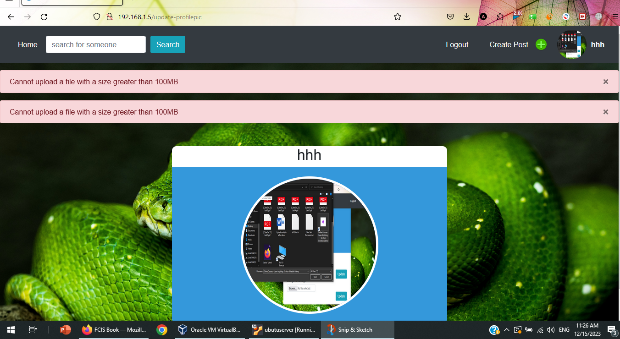
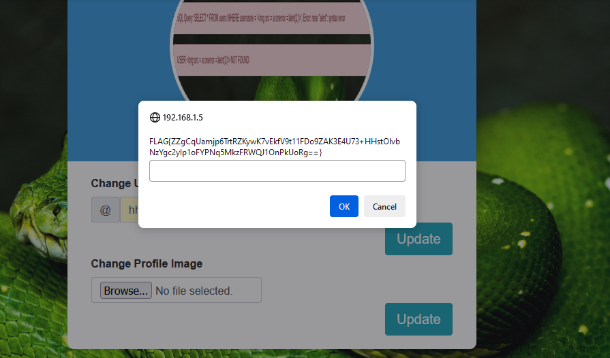
**2.. upload this file in Registration form.**

**3.. execute this file when clicking on image in profile.**

**☞ 2\ DOS attack.**

|  |  |
| --- | --- |
| Description: | File upload vulnerability occurs when a website or app lets users upload files without proper checks.  For example, during registration, if a system doesn't verify file types, an attacker might upload a harmful file, like an HTML file disguised as an image. This can lead to executing malicious code on the server, risking its security. |
| Recommendations | * File Type Validation * File Size Limit * Secure File Storage , Store uploaded files in a location isolated from the web root directory and apply strict permissions to prevent execution of uploaded files. |
| Affected Systems | <http://192.168.1.5/static/profile_pictures/sh.html> -->profile page |
| Threat Level | High (For us) 🡪 High (For System) |

**Screen Shots:**

**Steps:**

**1.. Try to make Server Faild by Two ways First way🡪 by put JS code in HTML code contains infinite loop to make sever busy always and not able to get any request else.**

**2.. Second way🡪 Make DOS attack by upload large file “9.5 GB”.**

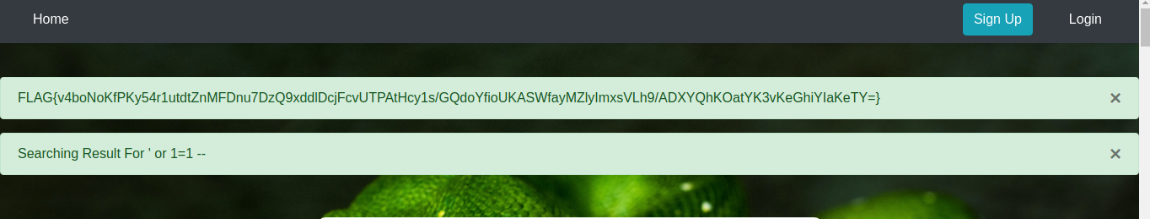
**3.. Server size will be full and can’t be able to store the remain size, so Server was down**

**SQL injection (2 Vuln.)**

**☞1\Error-Based SQL injection in Search field**

|  |  |
| --- | --- |
| Description: | SQL injection in a search field happens when attackers input harmful SQL code into the search bar. This manipulation can trick the system into running unintended database commands, potentially granting unauthorized access to sensitive information. |
| Recommendations | * Parameterized Queries * Input Validation * Least Privilege Principle * Web Application Firewall (WAF) * Error Handling |
| Affected Systems | http://192.168.1.5/search?username=%27+or+1%3D1+--&search= |
| Threat Level | Medium (For us) 🡪 High (For System) |

**Screen Shots:**



**Steps:**

**1.. Focus on Search field, write a malicious code here.**

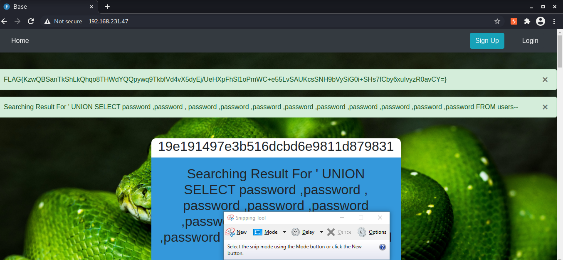
**2.. try error-based SQL injection.**

**☞ 2\Union-Based SQL injection in Search field**

|  |  |
| --- | --- |
| Description: | SQL injection in a search field happens when attackers input harmful SQL code into the search bar. This manipulation can trick the system into running unintended database commands, potentially granting unauthorized access to sensitive information. |
| Recommendations | * Parameterized Queries * Input Validation * Least Privilege Principle * Web Application Firewall (WAF) * Error Handling |
| Affected Systems | http://192.168.1.5/search?username=%27+or+1%3D1+--&search= |
| Threat Level | High (For us) 🡪 High (For System) |

**Screen Shots:**

A screenshot of a computer

Description automatically generated

**Steps:**

**1.. Because in search field I am retrieving data from database, I did a brute force Union-Based SQL injection to know the number of columns.**

**2.. Then I did a brute force SQL injection to know the passwords from DB by add ‘password’ in all columns to get the match.**

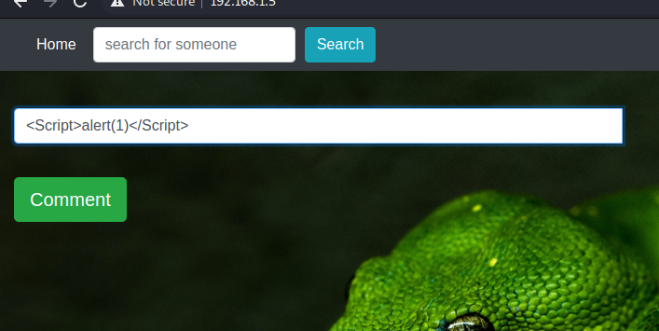
**3.. Then the query executed, and some hash values retrieved.**

Xss vulnerabilities (3 Vuln.)

**☞ 1\ Stored XSS**

|  |  |
| --- | --- |
| Description: | Cross-Site Scripting (XSS) is a web security flaw where attackers inject malicious scripts into web pages viewed by other users. This happens when the application fails to properly validate or sanitize user input. XSS can lead to executing harmful scripts, compromising user data, or performing unauthorized actions on affected websites. Prevention involves validating and encoding user input and implementing security measures like Content Security Policy (CSP) to mitigate this risk. |
| Recommendations | * Input Validation * Educate Developers and Users * Regular Security Testing |
| Affected Systems | http://192.168.1.5/comments?post\_id=14&user= |
| Threat Level | Medium (For us) 🡪 High (For System) |

**Screen shot:**



**Steps:**

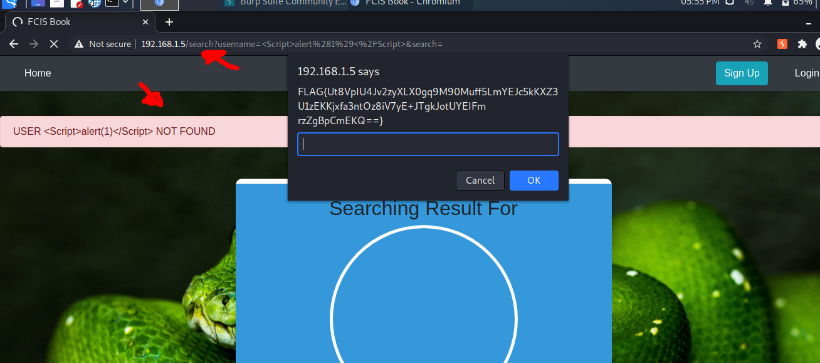
**1.. We search about any input field to execute a malicious code.**

**2.. Go to comment field and write a JS code that make a malicious action.  
 3.. The malicious action(alert) executed and stored in database.**

|  |  |
| --- | --- |
| Description: | Cross-Site Scripting (XSS) is a web security flaw where attackers inject malicious scripts into web pages viewed by other users. This happens when the application fails to properly validate or sanitize user input. XSS can lead to executing harmful scripts, compromising user data, or performing unauthorized actions on affected websites. Prevention involves validating and encoding user input and implementing security measures like Content Security Policy (CSP) to mitigate this risk. |
| Recommendations | * Input Validation * Educate Developers and Users * Regular Security Testing |
| Affected Systems | http://192.168.1.5/search?username= |
| Threat Level | Medium (For us) 🡪 High (For System) |

**☞**2\ **Reflected XSS**

**Screen Shot:**



**Steps:**

**1.. We search about any input field to execute a malicious code.**

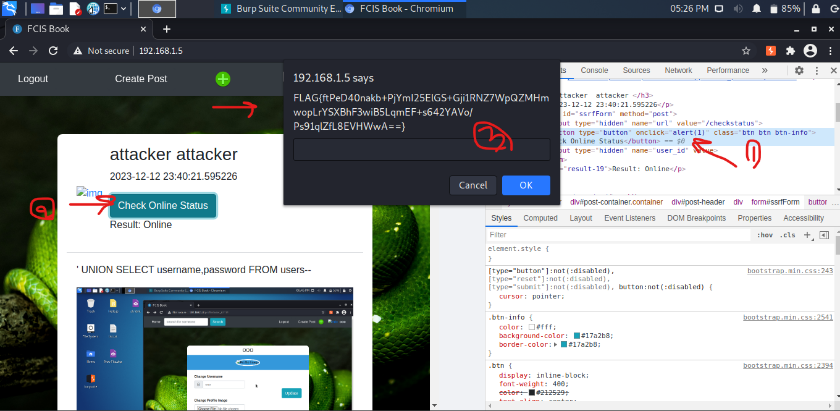
**2.. Go to Search field and write Reflected xss code.**

**3.. click on search button, code executed.**

**☞**3**\ DOM XSS**

|  |  |
| --- | --- |
| Description: | Cross-Site Scripting (XSS) is a web security flaw where attackers inject malicious scripts into web pages viewed by other users. This happens when the application fails to properly validate or sanitize user input. XSS can lead to executing harmful scripts, compromising user data, or performing unauthorized actions on affected websites. Prevention involves validating and encoding user input and implementing security measures like Content Security Policy (CSP) to mitigate this risk. |
| Recommendations | * Educate Developers and Users * Regular Security Testing |
| Affected Systems | <http://192.168.1.5/> 🡪Home |
| Threat Level | Medium (For us) 🡪 High (For System) |

**Screen Shot:**



**Steps:**

**1.. Open inspect in Home page.**

**2.. Get the check status button from inspect and go to the onclick pram.**

**3.. Change the method to alert (1).**

**4.. in the home page click on the button the alert will executed and the flag shown**.

**HTTP Method Tampering**

|  |  |
| --- | --- |
| Description: | "HTTP Method Tampering" is vulnerability you're describing involves manipulating the HTTP method from GET to HEAD using Burp Suite’s Intercept feature. This action may lead to unexpected behavior in the web application, potentially exposing sensitive information, bypassing security controls, or triggering unintended functionalities due to the altered request method. |
| Recommendations | * Strictly Validate and Enforce Allowed HTTP Methods: Design your application to only accept and process legitimate HTTP methods (GET, POST , etc..) Validate incoming requests to ensure they adhere to expected and allowed methods. * Utilize HTTP Strict Transport Security (HSTS): Preventing attackers from intercepting and altering traffic between the client and the server. * Security Testing |
| Affected Systems | <http://192.168.1.5> 🡪Home |
| Threat Level | Medium (For us) 🡪 High (For System) |

**Screen Shot**

A screenshot of a computer

Description automatically generated

**Steps:**

**1.. Login and go to the home page.**

**2.. Open the intercept in home page.**

**3.. Change request method from GET to HEAD and send the request.**

**4.. The request executed and the flag shown.**

**Curl command (Send Request to sever)**

|  |  |
| --- | --- |
| Description: | The command `curl -I http://192.168.1.5` sends a request to the specified IP address, asking for only the HTTP headers in response. This action could potentially reveal sensitive information present in the headers, especially if the server is misconfigured or vulnerable. This command might be used for reconnaissance by attackers to gather information about the server, potentially aiding in further exploitation. |
| Recommendations | * Monitoring and Detection of Unauthorized Activities * Proper Server Configuration: Ensure servers are configured properly to minimize the exposure of sensitive information. * Access Control and Permissions: Set access policies and rules to prevent unauthorized access to sensitive information. |
| Affected Systems | CMD executed on Ip 192.168.1.5 |
| Threat Level | Medium (For us) 🡪 High (For System) |

**Screen Shot:**

A computer screen with white text

Description automatically generated with medium confidence

**Steps:**

**1.. Open CMD and try to send a request to server.**

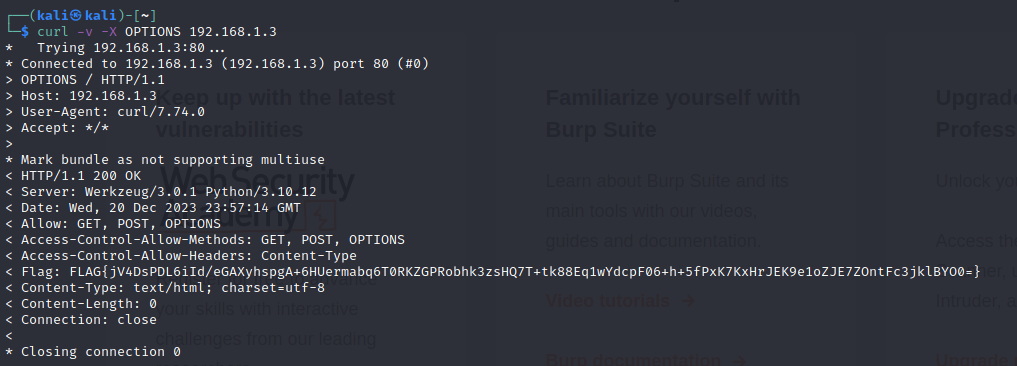
**2.. Write a Curl command and execute it to know what will happen.**

**3.. The command executed and got the flag.**

**Curl command (Send OPTIONS Request to sever)**

|  |  |
| --- | --- |
| Description: | The command `curl -I http://192.168.1.5` sends a request to the specified IP address, asking for only the HTTP headers in response. This action could potentially reveal sensitive information present in the headers, especially if the server is misconfigured or vulnerable. This command might be used for reconnaissance by attackers to gather information about the server, potentially aiding in further exploitation. |
| Recommendations | * Monitoring and Detection of Unauthorized Activities * Proper Server Configuration: Ensure servers are configured properly to minimize the exposure of sensitive information. * Access Control and Permissions: Set access policies and rules to prevent unauthorized access to sensitive information. |
| Affected Systems | CMD executed on Ip 192.168.1.5 |
| Threat Level | Medium (For us) 🡪 High (For System) |

**Screen Shots:**



**Steps:**

**1.. Open CMD and try to send an OPTIONS method and request a server.**

**2.. Write a Curl command and execute it to know what will happen.**

**3.. The command executed and got the flag.**

**Cryptographic Failures.**

|  |  |
| --- | --- |
| Description: | Cryptographic Failures, or previously known as Sensitive data exposure vulnerabilities refer to security flaws in a system that lead to the unintended exposure of confidential or sensitive information.  Cryptographic Failures refer to the failure to properly implement and manage cryptographic practices in a way that ensures the confidentiality and integrity of sensitive data. |
| Recommendations | * Use Established Algorithms: Employ widely accepted and reviewed cryptographic algorithms. Avoid creating custom algorithms or using outdated or weak encryption methods. * Strong Key Management: Ensure secure key storage and management practices. Use strong, randomly generated keys and rotate them periodically. Protect keys from unauthorized access. * Implement Secure Protocols 🡪 HTTPS |
| Affected Systems | http://192.168.1.5/search?username=%27+or+1%3D1+--&search= |
| Threat Level | High (For us) 🡪 High (For System) |

**Screen Shot:**

A screen shot of a computer

Description automatically generated

**Steps:**

**1.. After the query executed, and some hash values retrieved.**

**2.. I used hash analyzer to know the kind of hash algorithm used MD5**

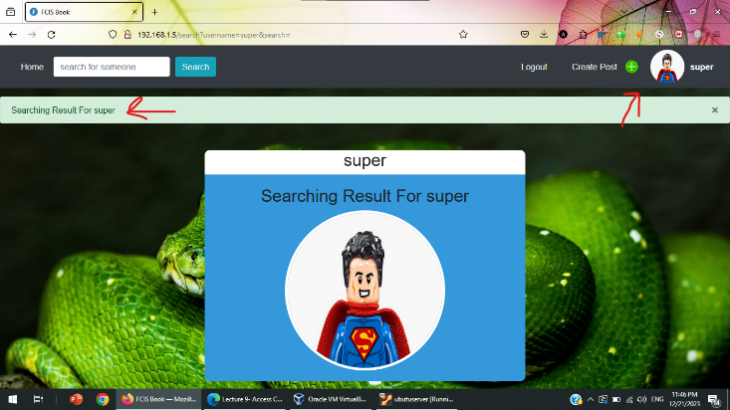
**3.. I used MD5 translator, and the value became “adminadminadmin”**

**4.. From Error Based got the Username” admin”, and the password that I got it and login successfully.**

**Misconfiguration Vulnerability**

|  |  |
| --- | --- |
| Description: | Misconfiguration vulnerabilities occur when a system or software is set up incorrectly, allowing unintended access, exposure, or mismanagement of resources. For example, if a system's access permissions are improperly configured, it might mistakenly provide access to sensitive user profiles when queried, even if those profiles should be restricted or non-existent for certain users or queries. |
| Recommendations | * Monitor and Log Configuration Changes: Implement robust monitoring and logging mechanisms to track changes in configurations. * Regularly Test and Validate Configurations: Perform regular security testing, including penetration testing and vulnerability assessments, to identify and rectify misconfigurations before they are exploited by attackers. * Regularly Test and Validate Configurations: Perform regular security testing, including penetration testing and vulnerability assessments, to identify and rectify misconfigurations before they are exploited by attackers. |
| Affected Systems | <http://192.168.1.3> 🡪 Home |
| Threat Level | Meduim (For us) 🡪 High (For System) |

**Screen Shot:**

****

**Steps:**

1.. Click on Search field and search for any user**, like Super.**

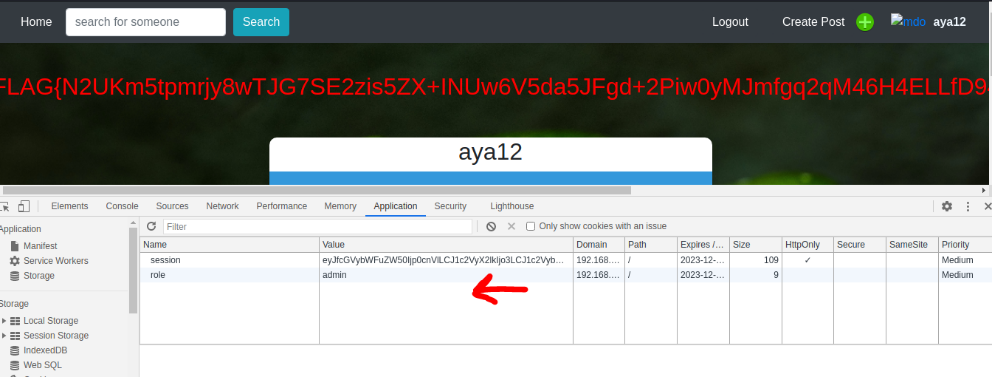
2.. If a user exists, the system automatically **the system will automatically open this user page.**

**Broken access control (2 Vuln)**

**☞** 1\ Vertical Broken Access

|  |  |
| --- | --- |
| Description: | **Vertical broken access control.** Attackers exploit weak access controls in a website to change their role from a regular user to an admin using tools like browser inspect elements or by modifying cookies. Preventing this requires robust server-side authorization checks and not relying solely on client-side data for user privileges. |
| Recommendations | * Least Privilege Principle * Sensitive Data Protection: Avoid exposing sensitive data like user roles or permissions directly in client-side storage mechanisms like cookies. If necessary, encrypt sensitive information stored on the client-side and validate it on the server. * Implement Proper Access Controls: Enforce strict access controls on the server-side. Validate user permissions and roles with each request to ensure users only access resources they are authorized to access. |
| Affected Systems | http://192.168.1.5/profile |
| Threat Level | Medium (For us) 🡪 High (For System) |

**Screen Shot:**



**Steps:**

**1.. Right click, open inspect in the web site.**

**2.. Go to Application 🡪 Cookies 🡪http://192.168.1.5**

**3.. change role from user to admin.**

**☞**2\broken access vulnerability **(IDOR)**

|  |  |
| --- | --- |
| Description: | Insecure Direct Object Reference (IDOR) is a security flaw where users can access other users' data by changing parameters in the URL. For example, altering the user ID in the URL might allow access to another user's profile. This occurs due to inadequate access controls or authorization checks in the application. Preventing it involves implementing strict access controls and verifying user permissions for each request. |
| Recommendations | * Implement Proper Access Controls: Ensure that the application enforces robust access controls. Authenticate and authorize users for specific actions or data access. Check the user's permissions against the requested action or resource on the server-side. * Use Unique Identifiers: Avoid using easily guessable or sequential identifiers in URLs or parameters that could expose sensitive information. Employ random or unique identifiers to access resources or perform actions. * Limit Access Scope: Apply strict access controls to ensure that users can only access resources they are explicitly authorized to access. Avoid granting unnecessary permissions to users. |
| Affected Systems | http://192.168.1.5/profile?id=5 |
| Threat Level | Medium (For us) 🡪 High (For System) |

**Screen Shot:**

A screenshot of a computer

Description automatically generated

**Steps:**

**1.. Go to profile page.**

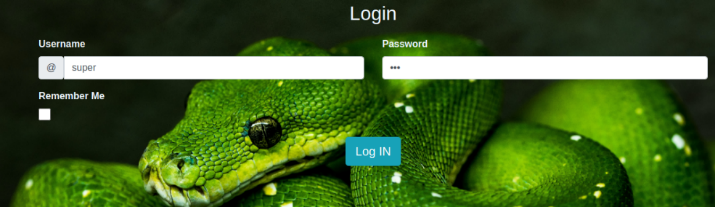
**2.. Change id in URL to any other id.**

**3.. User profile is opened.**

**Password Guessing vulnerability.**

|  |  |
| --- | --- |
| Description: | This vulnerability, commonly known as "Password Guessing," occurs when an attacker, armed with a known username, uses tools like Burp Suite to try a list of common passwords systematically to guess the correct password and gain unauthorized access to an account. |
| Recommendations | * Strong Password Policies * Multi-factor Authentication * Rate Limiting * Brute Force Protection: Deploy intrusion detection systems or tools that detect and block repeated login attempts from suspicious sources. |
| Affected Systems | 192.168.1.5/login |
| Threat Level | Medium (For us) 🡪 High (For System) |

**Screen Shots:**

** A screenshot of a computer

Description automatically generated**

**Steps:**

**1.. Form error-based SQL injection in Search field, we get some user’s name.**

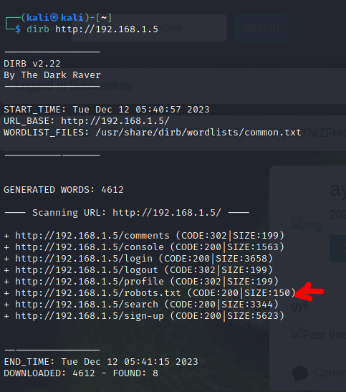
**2.. Choose name “Super” and use list to do brute force attack to get the password.**

**3.. After 70 trails, the password exists “BESTFRIEND”, and we could login in site.**

**Sensitive Information Exposure through robots.txt**

|  |  |
| --- | --- |
| Description: | "Sensitive Information Exposure through robots.txt." It occurs when sensitive or private information is unintentionally disclosed through the robots.txt file. This file is meant to guide search engine crawlers on what to index or not index on a website. However, if sensitive directories or files are listed in robots.txt, it could potentially expose private information to anyone viewing the file. |
| Recommendations | * Secure Configuration: ensure that sensitive information isn't stored in easily guessable or predictable URLs. Avoid using default or common naming conventions for sensitive files or directories. * Review and Restrict Content: Regularly review the robots.txt file to ensure it does not expose sensitive directories, files, or information. Remove any sensitive entries that shouldn't be indexed by search engines. * Authentication and Authorization: Implement proper authentication and authorization mechanisms for sensitive areas of the website. Ensure that even if they are listed in robots.txt, unauthorized access is prevented. |
| Affected Systems | 192.168.1.5/robots.txt |
| Threat Level | Low (For us) 🡪 High (For System) |

**Screen Shots:**

A screenshot of a computer

Description automatically generated

**Steps:**

**1.. open CMD and write dirb to know what the files in the site are.**

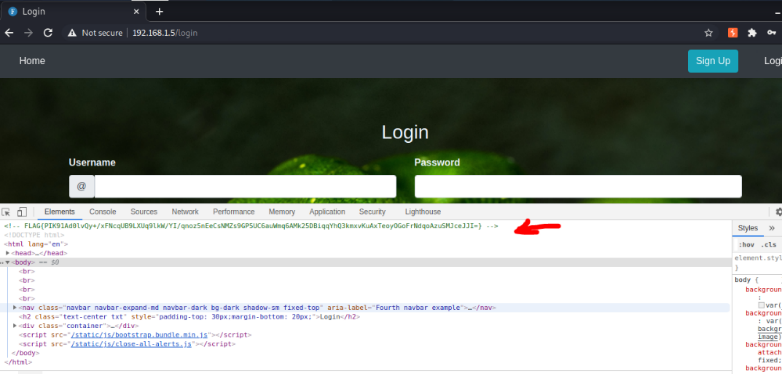
**2.. No exist a sensitive file but robot.txt exists.**

**3.. take the path of this file and put it into site.**

**Information Disclosure through Browser Inspection**

|  |  |
| --- | --- |
| Description: | "Information Disclosure through Browser Inspection" or "Information Leakage through Browser Developer Tools."  This vulnerability occurs when sensitive information within a website's code or resources, like passwords or API keys, is inadvertently exposed and accessible through browser inspection tools, posing a risk of unauthorized access or security breaches. |
| Recommendations | * Avoid Hardcoding Sensitive Information * secure Coding Practices * \*Content Security Policy (CSP): Implement CSP headers to control which resources a browser can load, limiting the risk of unauthorized scripts or content execution |
| Affected Systems7 | http://192.168.1.5/login |
| Threat Level | Low (For us) 🡪 High (For System) |

**Screen Shot:**



**Steps:**

**1.. In the login page, open the inspect to get any hint.**

**2.. The flag was the first line.**

**SSRF vulnerability.**

|  |  |
| --- | --- |
| Description: | SSRF is a vulnerability allowing attackers to manipulate a web server's outgoing requests by tricking it into accessing unauthorized URLs. This occurs when the application doesn't properly validate user input used as a URL, enabling attackers to redirect requests to internal resources, sensitive files, or external systems. This could lead to unauthorized access, data leakage, or exploitation of internal systems. Prevention involves strict input validation, restricting accessible URLs, and employing access controls to limit server access. |
| Recommendations | * Whitelisting Allowed URLs: Create a whitelist of approved URLs or domains that the application can access. This restricts the server from fetching data or making requests to unauthorized or sensitive endpoints. * URL Parsing and Redirection: Consider using URL parsing libraries or frameworks that provide secure methods for handling URLs, preventing the application from redirecting requests to internal or unauthorized resources. * se of Firewalls and Security Controls: Employ firewalls and network security measures to restrict outgoing requests from the server. Configure network settings to limit access to internal resources or sensitive information. |
| Affected Systems | <http://192.168.1.5> 🡪 comment |
| Threat Level | High (For us) 🡪 High (For System) |

**Screen shot:**

****

**Steps:** **1.. check status button in Create post in the home page, there is a URL when we open the intercept.**

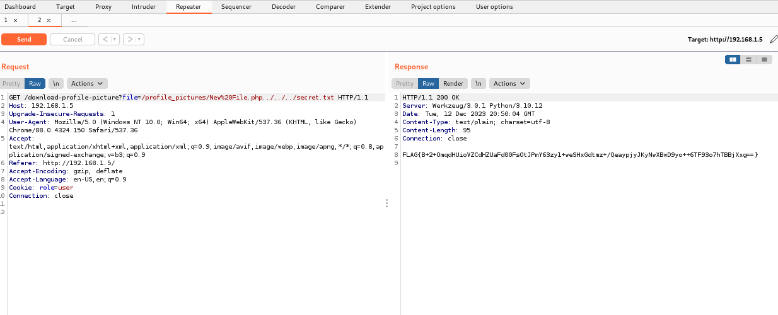
**2.. change the URL to go to another page or any URL other or by local host.**

**🡪 URL=http://localhost/**

**Directory Traversal vulnerability**

|  |  |
| --- | --- |
| Description: | Directory Traversal or Path Traversal is a security flaw in web applications. It allows attackers to access files or directories outside the intended scope by manipulating input parameters or URLs. Exploiting this flaw involves using " ../ " sequences to traverse up the directory tree and access restricted files or system-level resources. Mitigating this vulnerability requires robust input validation and sanitization to prevent unauthorized access to sensitive files or directories. |
| Recommendations | * Use Absolute Paths: Prefer using absolute paths rather than relative paths for file system operations. This ensures that files are accessed from known, fixed locations, reducing the risk of traversal attacks. * Access Controls: Implement proper access controls and permissions to limit the application's access to files and directories. Set strict permissions to only allow access to necessary resources and restrict access to sensitive areas. * Whitelisting: Use whitelists to specify allowed directories or file paths that the application can access. This restricts file operations to specific, authorized locations. |
| Affected Systems | http://192.168.1.5/download-profile-picture?file=/profile\_pictures/New%20File.php |
| Threat Level | High (For us) 🡪 High (For System) |

**Screen Shot:**



**Steps:**

**1.. In the comments page, open the intercept and click on uploaded image in comment.**

**2.. In the first line in the request there is a path of this image.**

**3.. Add some ../ in this path then add secret.txt file to get it.**

Thank You