Penetration Testing Report

Full Name: Syed Junaid Ahmad Andrabi

Program: HCPT

Date : 08/08/2024

Introduction

This report document hereby describes the proceedings and results of a Black Box security assessment conducted against the **Week {3} Labs**. The report hereby lists the findings and corresponding best practice mitigation actions and recommendations.

I. Objective

The objective of the assessment was to uncover vulnerabilities in the **Week {3} Labs** and provide a final security assessment report comprising vulnerabilities, remediation strategy and recommendation guidelines to help mitigate the identified vulnerabilities and risks during the activity.

II. Scope

The scope of the penetration testing project by Hacktify Cyber Security aims to identify and address vulnerabilities related to SQL Injection and Cross Site Request Frogery within the specified web application..

SQL Injection (SQLi) allows attackers to manipulate databases, leading to data theft, unauthorized access, and system compromise. Cross-Site Request Forgery (CSRF) exploits user trust, making them perform unintended actions on a web application, potentially resulting in unauthorized changes, data breaches, and account hijacking. Both pose significant security risks.

Application , Name	{Lab 1 – SQL Injection} {Lab 2 – Cross-Site Request Frogery}

III. Summary

Outlined is a Black Box Application Security assessment for the **Week {3} Labs**.

Total number of Sub-labs: 13(10-SQL Injection, 3-Cross-Site Request Frogery)

High	Medium	Low

3	4	6

High - **3 Sub-lab with high difficulty level**

Medium - **4 Sub-labs with medium difficulty level**

Low - **6 Sub-labs with low difficulty level**

1. SQL Injection

1.1. Strings and Errors Part 1!

Reference	**Risk Rating**
Sub-lab-1: Strings and Errors Part 1!	**Low**
Tools Used	
Burp-Suite	
Vulnerability Description	
The SQL Injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. Attackers insert malicious SQL code into input fields (like login forms) that are not properly sanitized.	
How It Was Discovered	
Automated Tools –Burp-suite	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_1/lab_1.php	
Consequences of not Fixing the Issue	
Data Breaches: Unauthorized access to sensitive data.	
Suggested Countermeasures	

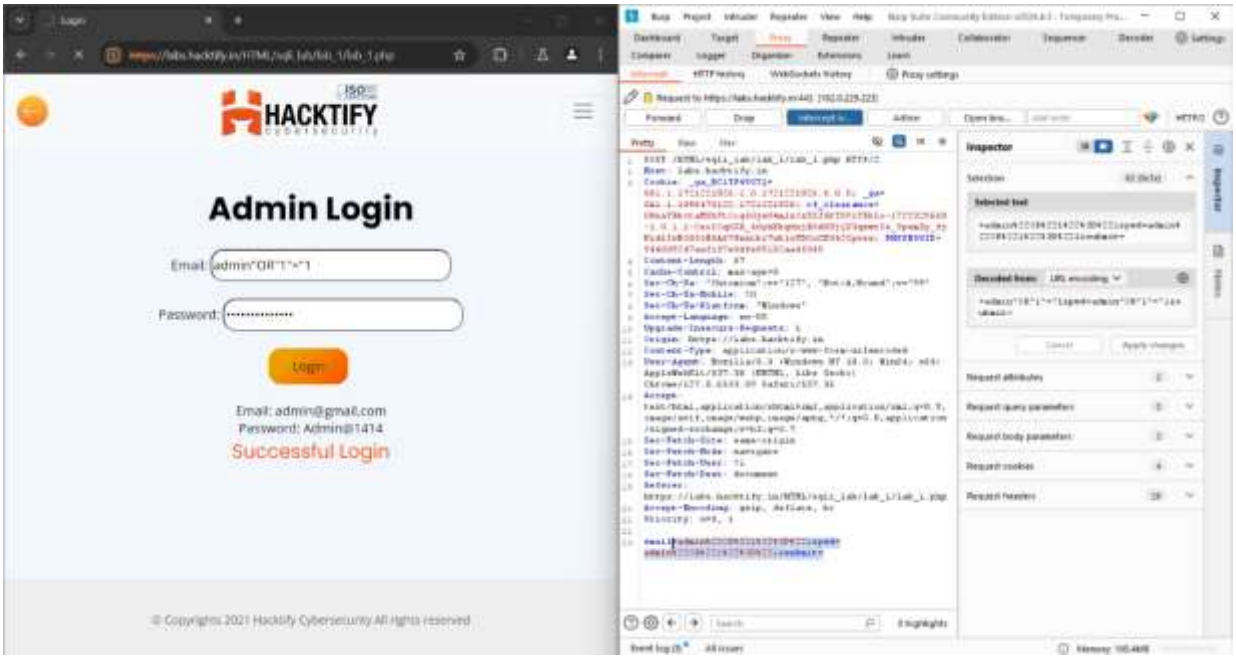
Proof of Concept

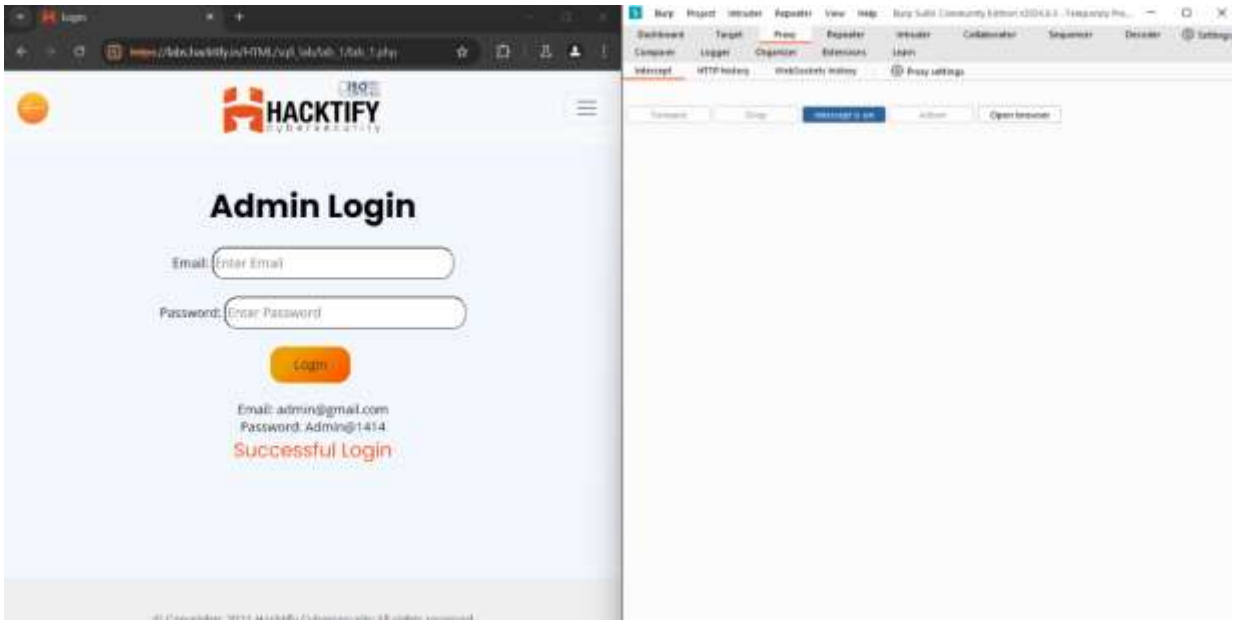
- **Input Validation**: Ensure all user inputs are properly validated and sanitized.
- **Parameterized Queries**: Use parameterized queries or prepared statements to prevent SQL code injection.

References

https://owasp.org/www-community/attacks/SQL_Injection
https://portswigger.net/web-security/sql-injection

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab





1.2. Strings and Errors Part 2!

Reference	Risk Rating
Sub-lab-1:Strings and Errors Part 2!	**Low**
Tools Used	
Bur	
Vulnerability Description	
The SQL Injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. Attackers insert malicious SQL code into input fields (like login forms) that are not properly sanitized.	
How It Was Discovered	
Automated Tools – Burp-Suite	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_2/lab_2.php	
Consequences of not Fixing the Issue	

Proof of Concept

- **Data Breaches**: Unauthorized access to sensitive data.
- **Data Manipulation**: Alteration or deletion of data.
- **System Compromise**: In some cases, attackers can gain control over the entire database server.

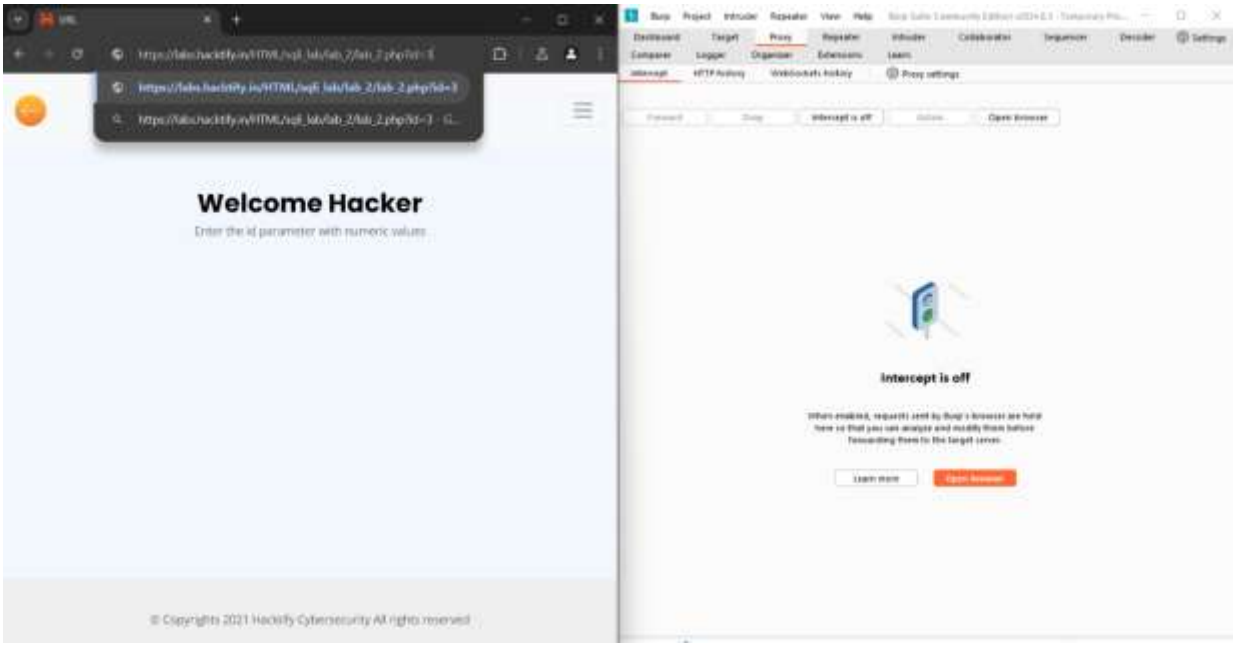
Suggested Countermeasures

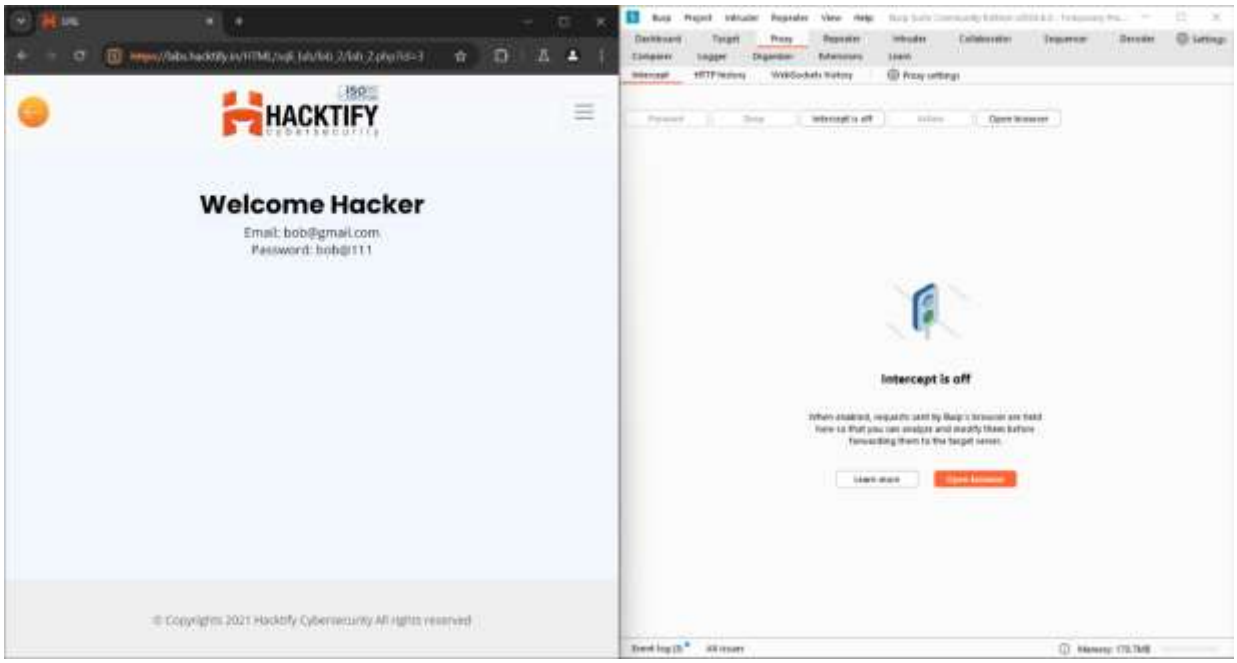
- **Input Validation**: Ensure all user inputs are properly validated and sanitized.
- **Parameterized Queries**: Use parameterized queries or prepared statements to prevent SQL code injection.
- **Stored Procedures**: Utilize stored procedures to handle database operations.

References

https://www.geeksforgeeks.org/sql-injection/
https://owasp.org/www-community/attacks/SQL_Injection
https://portswigger.net/web-security/sql-injection

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab.





1.3. Strings and Errors Part 3!

Reference	Risk Rating
Sub-lab-3: Strings and Errors Part 3!	Low
Tools Used	
Burp-Suite	
Vulnerability Description	
The SQL Injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. Attackers insert malicious SQL code into input fields (like login forms) that are not properly sanitized. Unauthorized Data Access: This can lead to unauthorized access to sensitive data, such as user credentials, personal information, and financial records.	
How It Was Discovered	
Automated Tools –Burp-suite	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_3/lab_3.php	
Consequences of not Fixing the Issue	

Proof of Concept

- **Data Breaches**: Unauthorized access to sensitive data.
- **Data Manipulation**: Alteration or deletion of data.
- **System Compromise**: In some cases, attackers can gain control over the entire database server.

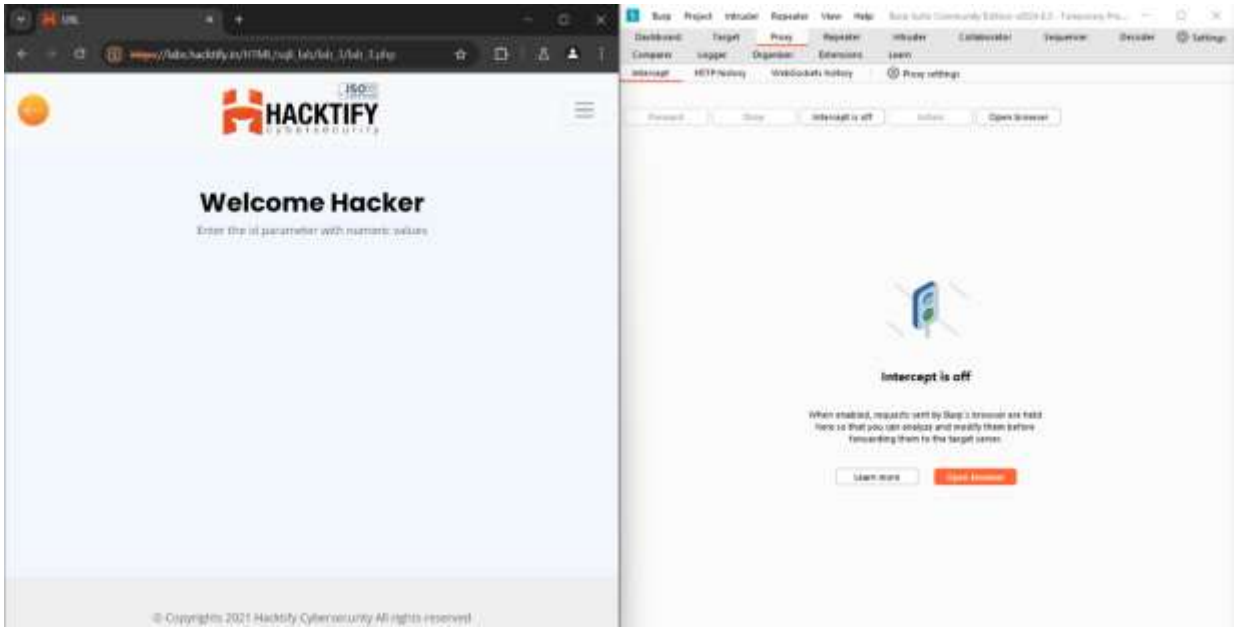
Suggested Countermeasures

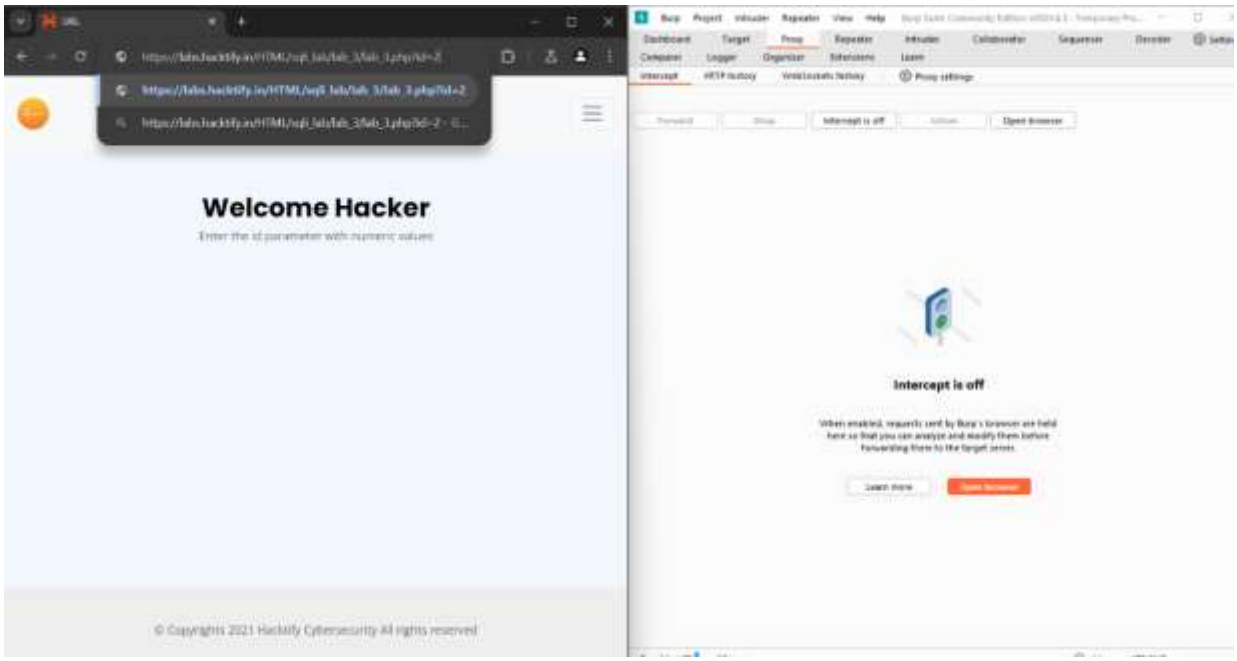
- **Input Validation**: Ensure all user inputs are properly validated and sanitized.
- **Parameterized Queries**: Use parameterized queries or prepared statements to prevent SQL code injection.
- **Stored Procedures**: Utilize stored procedures to handle database operations.

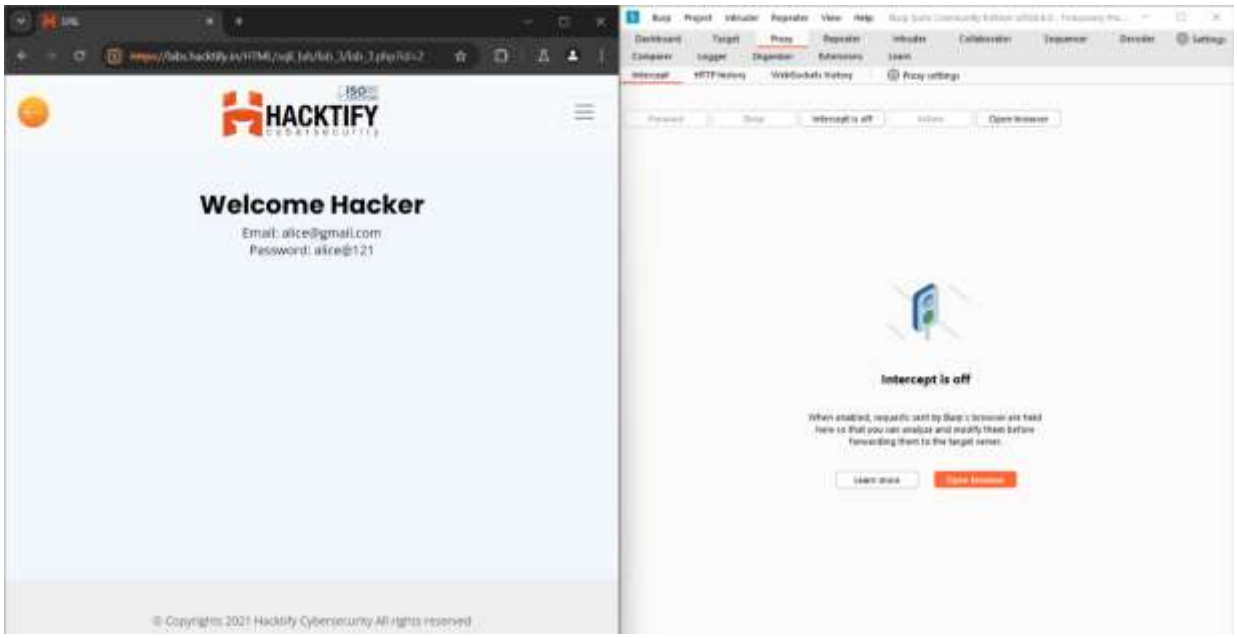
References

https://www.geeksforgeeks.org/sql-injection/
https://owasp.org/www-community/attacks/SQL_Injection
https://portswigger.net/web-security/sql-injection

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab





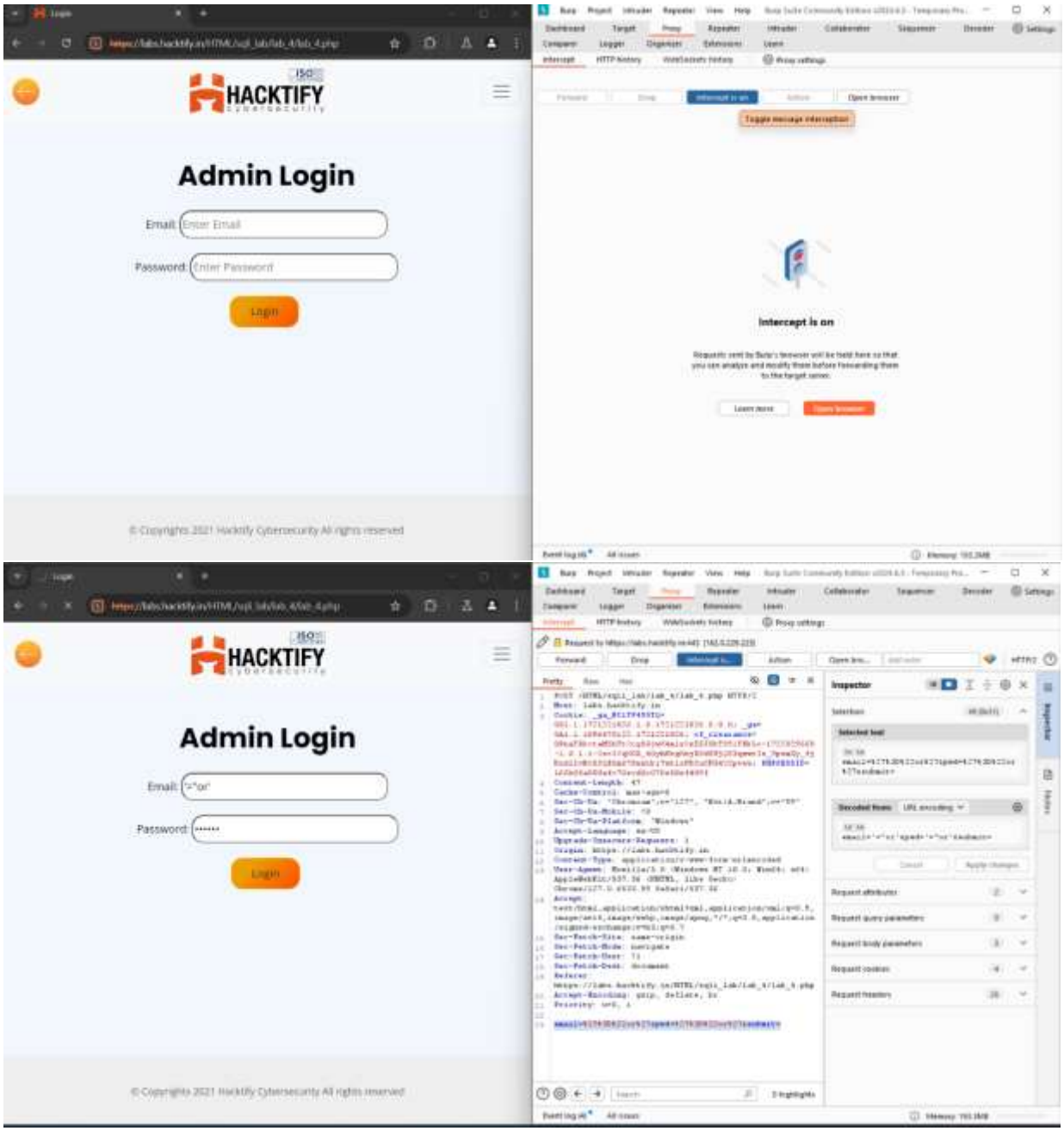


1.4. Let’s Trick ‘em!

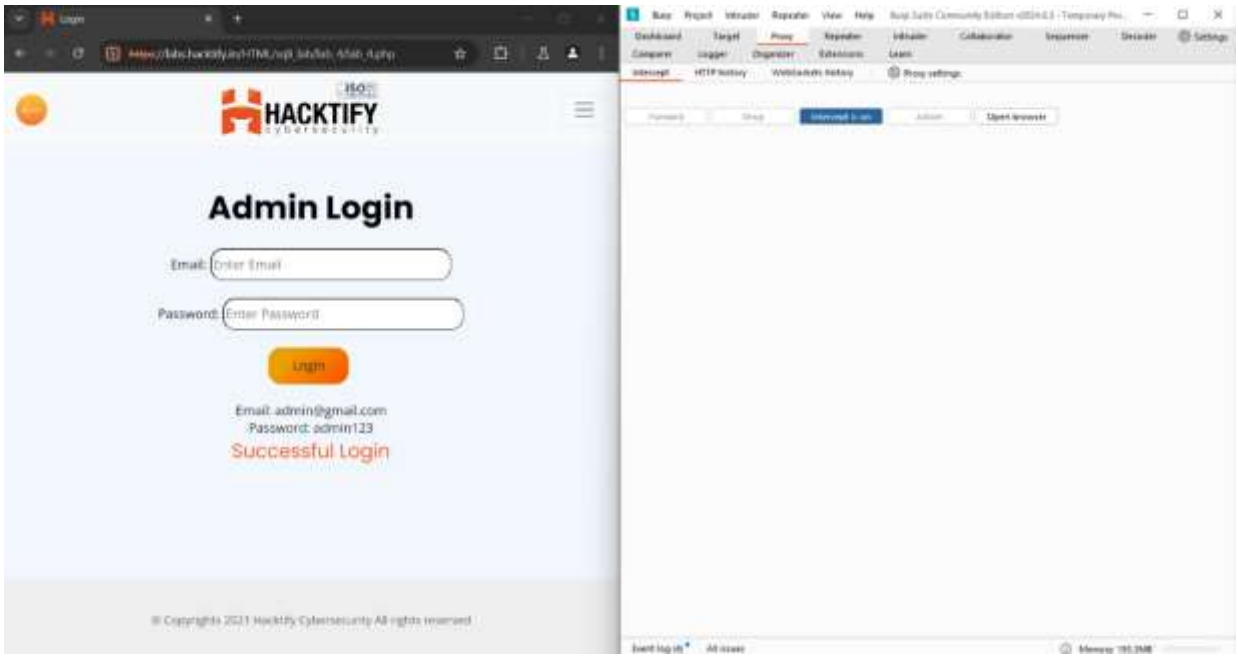
Proof of Concept

Reference	Risk Rating
Sub-lab-4: Let's Trick 'em!	Medium
Tools Used	
Burp-Suite	
Vulnerability Description	
The SQL Injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. Attackers insert malicious SQL code into input fields (like login forms) that are not properly sanitized. Unauthorized Data Access: This can lead to unauthorized access to sensitive data, such as user credentials, personal information, and financial records.	
How It Was Discovered	
Automated Tools – Burp-Suite	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_4/lab_4.php	
Consequences of not Fixing the Issue	
• **Data Breaches**: Unauthorized access to sensitive data. • **Data Manipulation**: Alteration or deletion of data. • **System Compromise**: In some cases, attackers can gain control over the entire database server.	
Suggested Countermeasures	
• **Input Validation**: Ensure all user inputs are properly validated and sanitized. • **Parameterized Queries**: Use parameterized queries or prepared statements to prevent SQL code injection. • **Stored Procedures**: Utilize stored procedures to handle database operations	
References	
https://www.geeksforgeeks.org/sql-injection/ https://owasp.org/www-community/attacks/SQL_Injection https://portswigger.net/web-security/sql-injection	

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab


HACKTIFY
Admin Login
Email
Enter Email
Password
Enter Password
Login
© Copyrights 2021 Hacktify Cybersecurity All rights reserved
Intercept is on
HACKTIFY
Admin Login
Email
="or"
Password
Login
© Copyrights 2021 Hacktify Cybersecurity All rights reserved

Proof of Concept

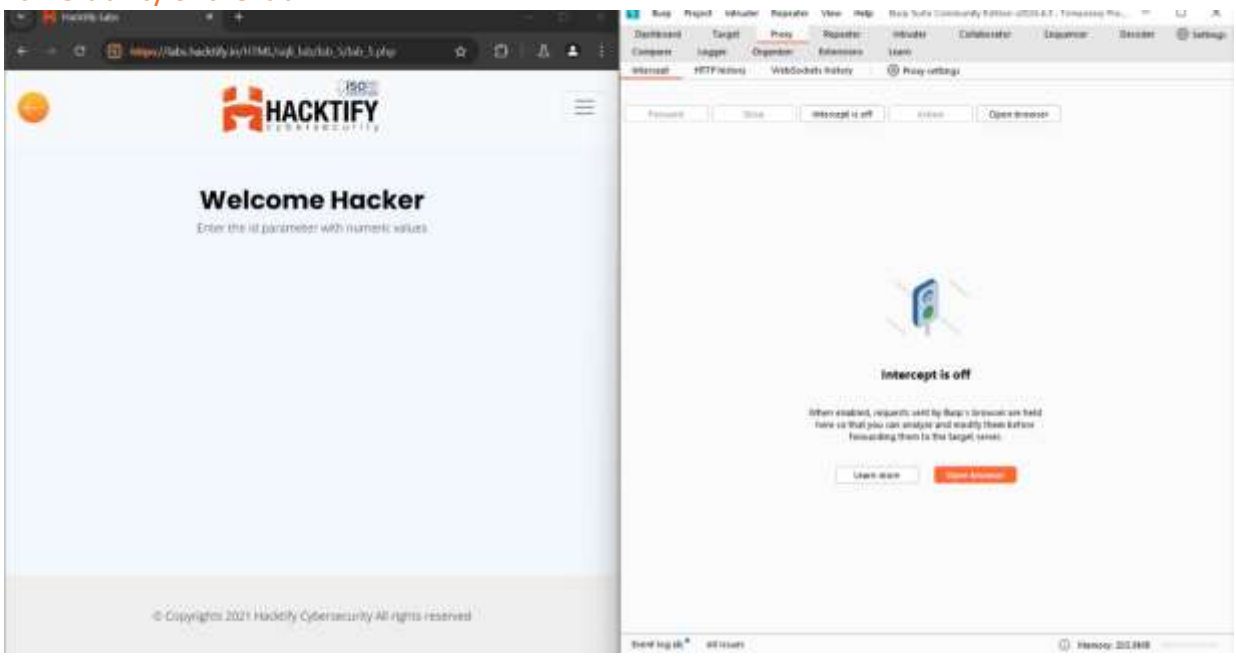


1.5. Booleans and Blind!

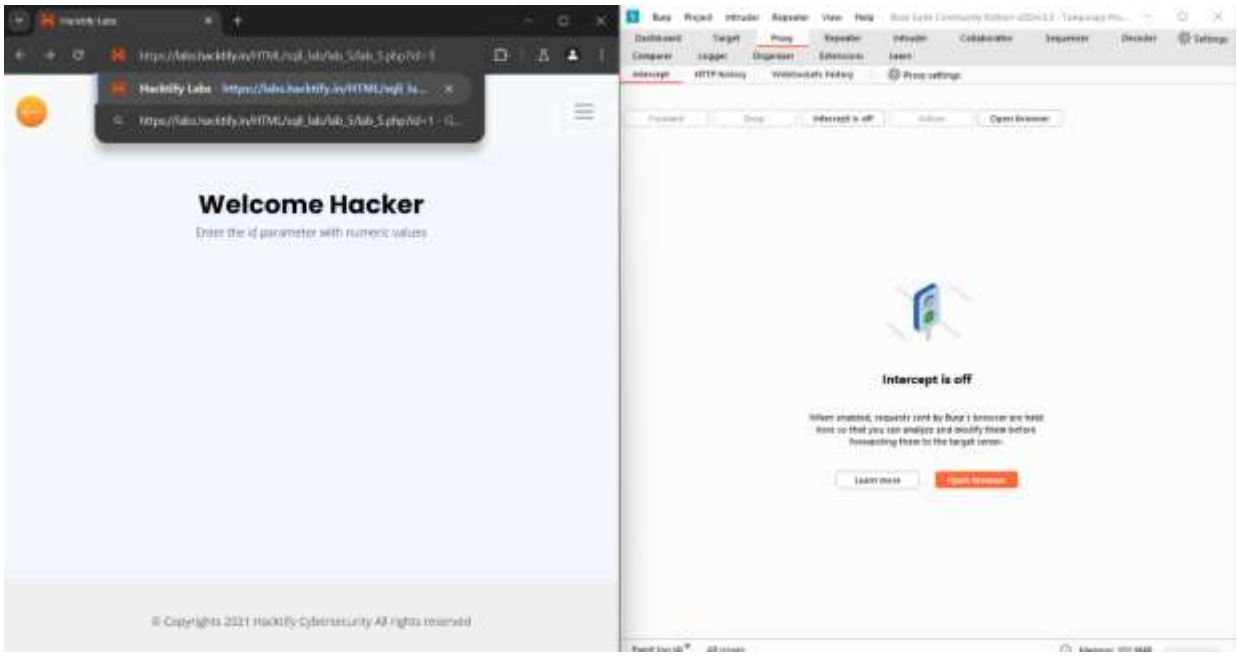
Reference	Risk Rating
Sub-lab-5: Booleans and Blind!	**High**
Tools Used	
Burp- Suite	
Vulnerability Description	
The SQL Injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. Attackers insert malicious SQL code into input fields (like login forms) that are not properly sanitized. Unauthorized Data Access: This can lead to unauthorized access to sensitive data, such as user credentials, personal information, and financial records.	
How It Was Discovered	
Automated Tools – Burp-Suite	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_5/lab_5.php	

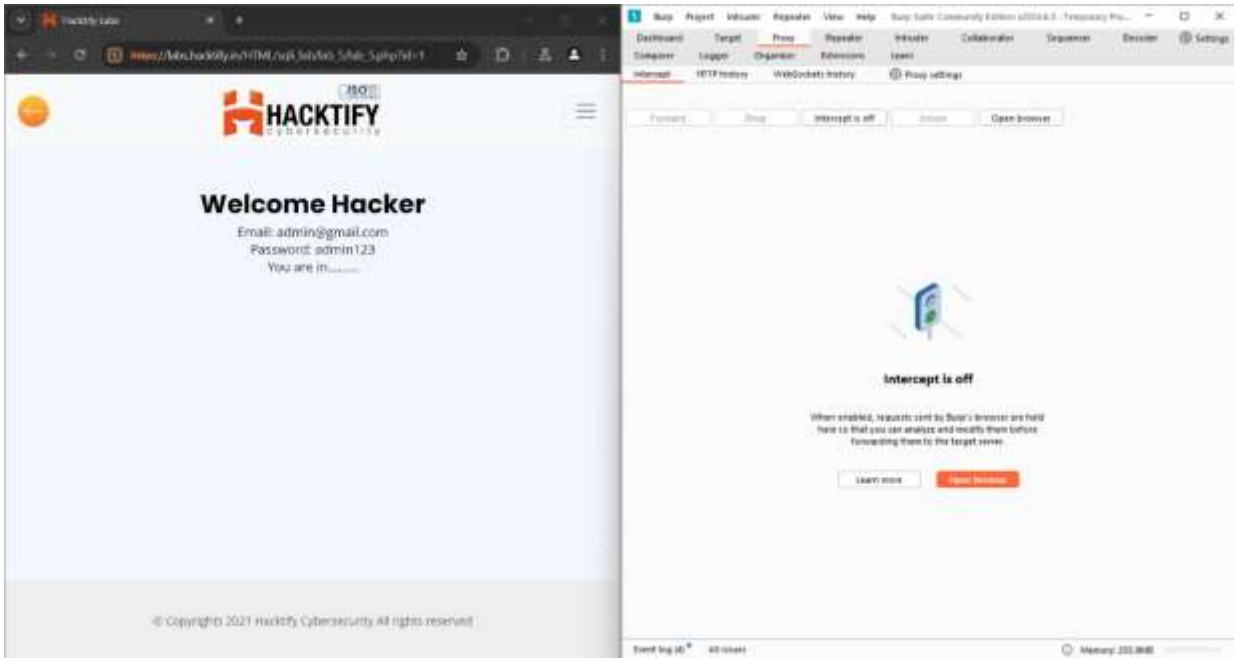
Consequences of not Fixing the Issue
• Attackers can alter, delete, or insert data within the database. This can compromise the integrity of the data, leading to incorrect information being stored and used. • A successful SQL Injection attack can severely damage an organization's reputation. Customers and partners may lose trust in the organization's ability to protect their data, leading to long-term reputational harm.
Suggested Countermeasures
• **Sanitize Inputs**: Ensure all user inputs are properly sanitized to remove any potentially harmful characters. • **Whitelist Validation**: Use whitelisting to allow only expected input formats and values. • **Prepared Statements**: Use parameterized queries or prepared statements to ensure that user inputs are treated as data, not executable code.
References
https://www.varonis.com/blog/what-is-sql-injection https://www.geeksforgeeks.org/sql-injection/ https://owasp.org/www-community/attacks/SQL_Injection https://portswigger.net/web-security/sql-injection

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab



Proof of Concept





1.6. Error Based : Tricked

Reference	Risk Rating
Sub-lab-6: Error Based : Tricked	**Medium**

Tools Used

Burp Suite

Vulnerability Description

The SQL Injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database.
Attackers insert malicious SQL code into input fields (like login forms) that are not properly sanitized.
Unauthorized Data Access: This can lead to unauthorized access to sensitive data, such as user credentials, personal information, and financial records.

How It Was Discovered

Automated Tool----Burp-suite

Vulnerable URLs

https://labs.hacktify.in/HTML/sqli_lab/lab_6/lab_6.php

Consequences of not Fixing the Issue

- Attackers can gain unauthorized access to sensitive data, such as personal information, financial records, and intellectual property. This can lead to significant privacy violations and legal repercussions.
- Attackers can alter, delete, or insert data within the database. This can compromise the integrity of the data, leading to incorrect information being stored and used.

Suggested Countermeasures

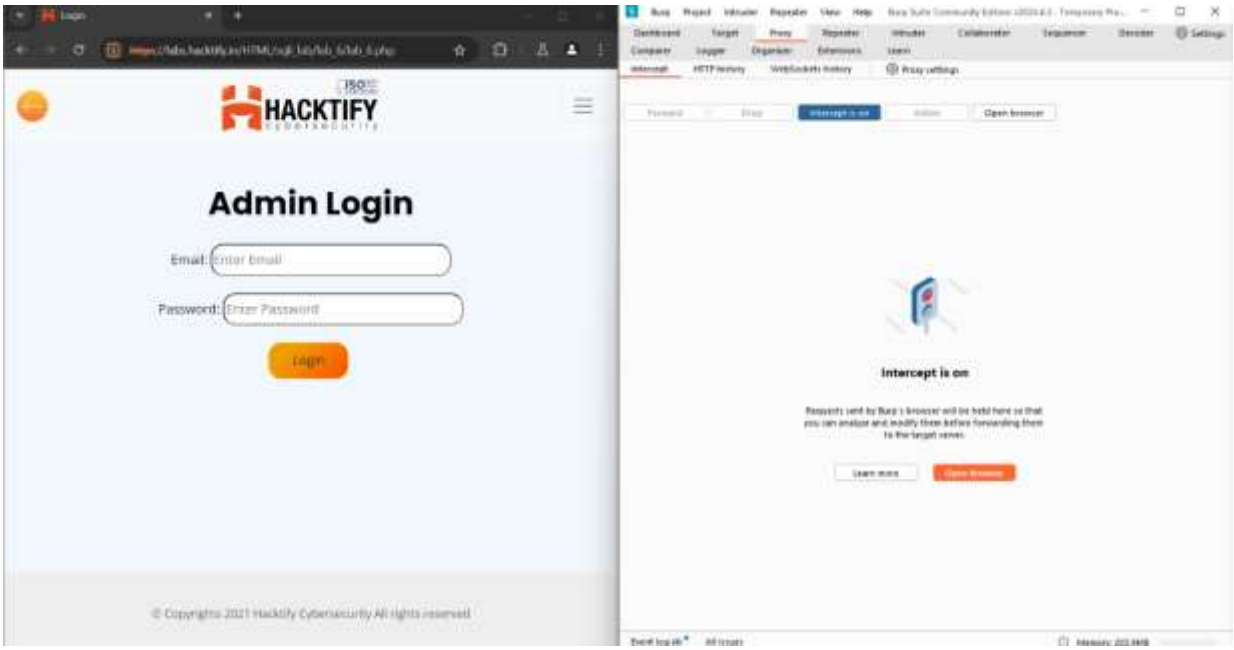
- **Sanitize Inputs**: Ensure all user inputs are properly sanitized to remove any potentially harmful characters.
- **Whitelist Validation**: Use whitelisting to allow only expected input formats and values.
- **Prepared Statements**: Use parameterized queries or prepared statements to ensure that user inputs are treated as data, not executable code.
- **Traffic Filtering**: Deploy a WAF to filter and monitor incoming traffic for malicious SQLi patterns.

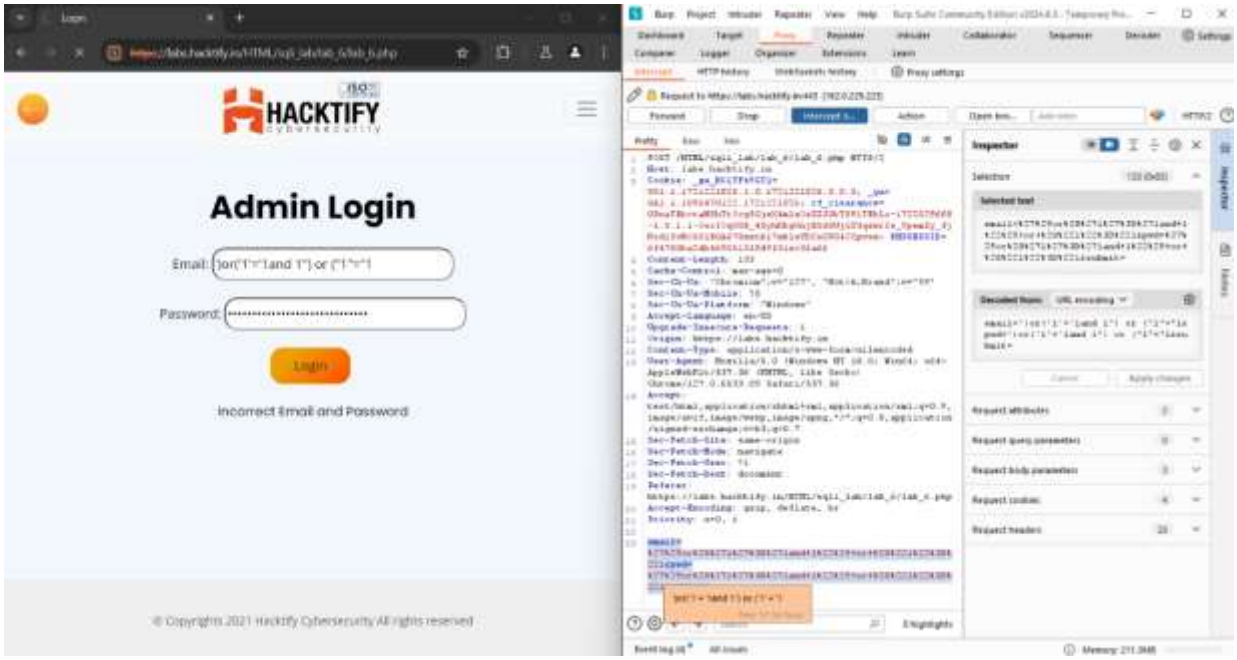
References

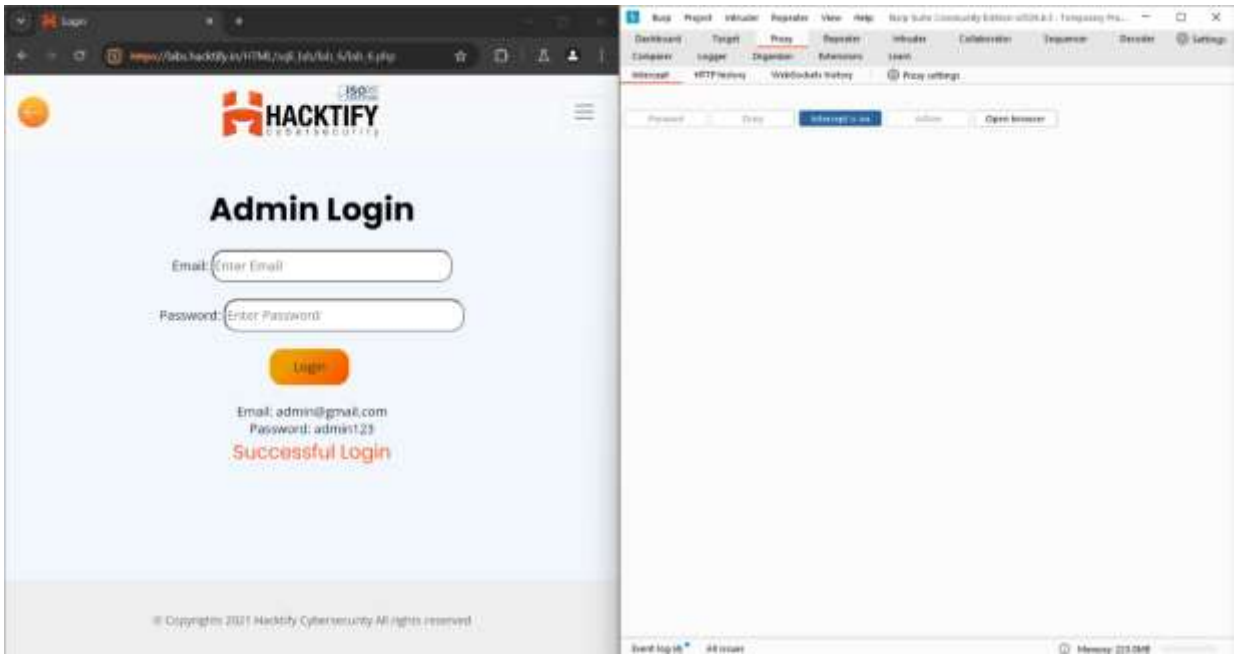
https://owasp.org/www-community/attacks/SQL_Injection
https://portswigger.net/web-security/sql-injection
https://www.varonis.com/blog/what-is-sql-injection

Proof of Concept

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab







1.7. Errors and Post!

Reference	Risk Rating
Sub-lab-7: Errors and Post!	**Low**
Tools Used	
Burp Suite	
Vulnerability Description	
The SQL Injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. Attackers insert malicious SQL code into input fields (like login forms) that are not properly sanitized. Unauthorized Data Access: This can lead to unauthorized access to sensitive data, such as user credentials, personal information, and financial records.	
How It Was Discovered	
Automated Tool----Burp-suite	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_7/lab_7.php	
Consequences of not Fixing the Issue	

Proof of Concept

- Attackers can gain unauthorized access to sensitive data, such as personal information, financial records, and intellectual property. This can lead to significant privacy violations and legal repercussions.
- Attackers can alter, delete, or insert data within the database. This can compromise the integrity of the data, leading to incorrect information being stored and used.

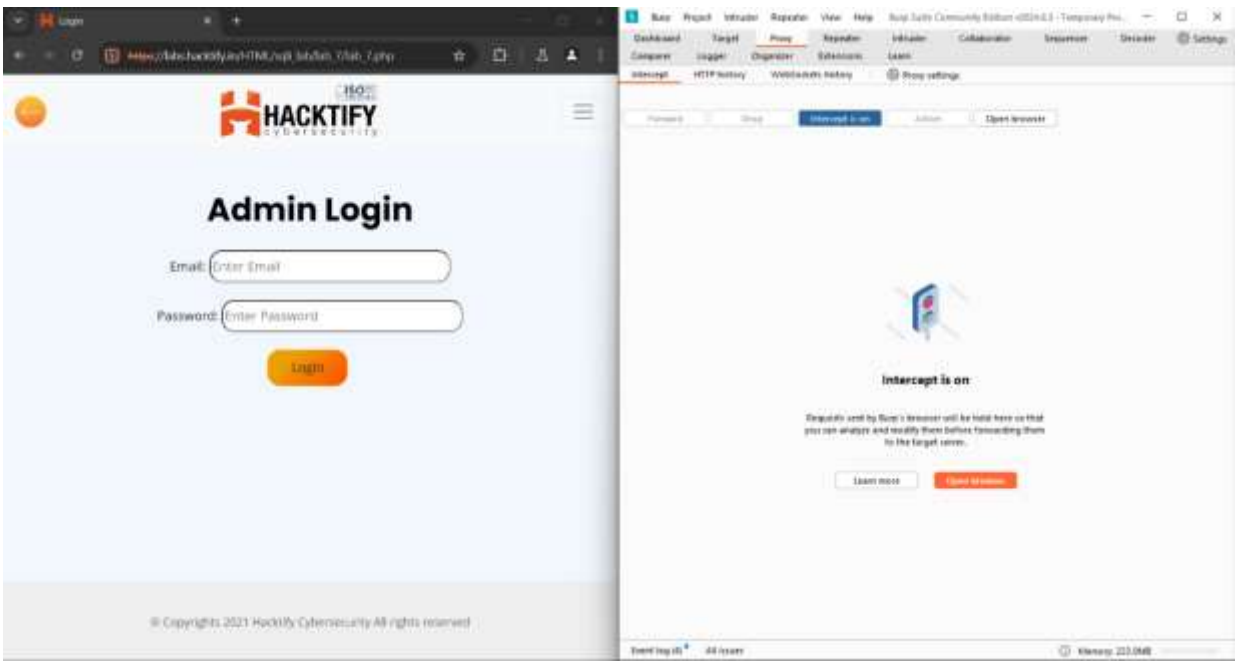
Suggested Countermeasures

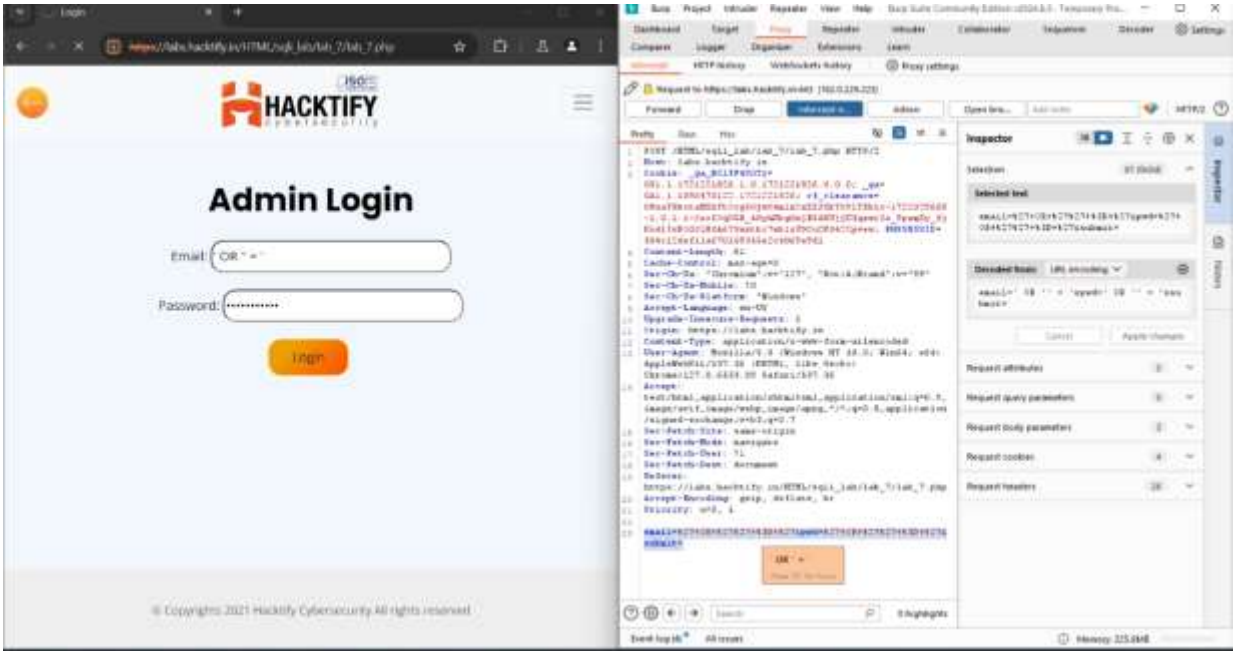
- **Sanitize Inputs**: Ensure all user inputs are properly sanitized to remove any potentially harmful characters.
- **Whitelist Validation**: Use whitelisting to allow only expected input formats and values.
- **Prepared Statements**: Use parameterized queries or prepared statements to ensure that user inputs are treated as data, not executable code.
- **Traffic Filtering**: Deploy a WAF to filter and monitor incoming traffic for malicious SQLi patterns.

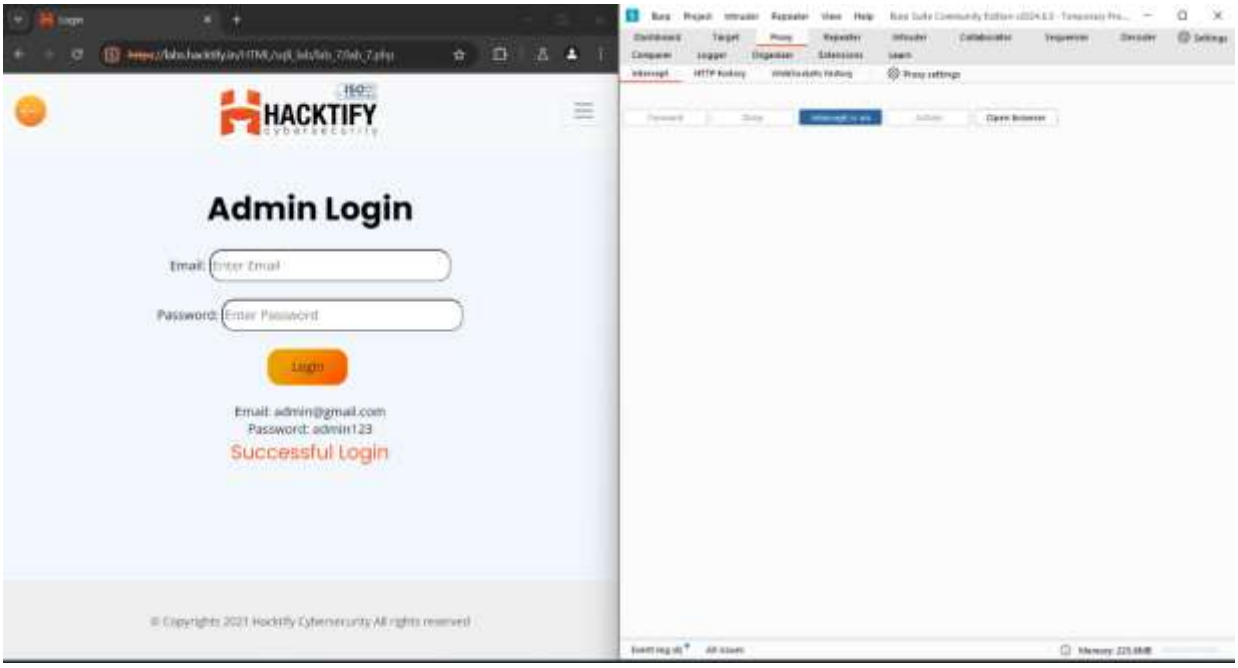
References

https://owasp.org/www-community/attacks/SQL_Injection
https://portswigger.net/web-security/sql-injection
https://www.varonis.com/blog/what-is-sql-injection

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab




HACKTIFY
Admin Login
Email: OR " = "
Password:
Login
© Copyrights 2021 Hacktify Cybersecurity All rights reserved

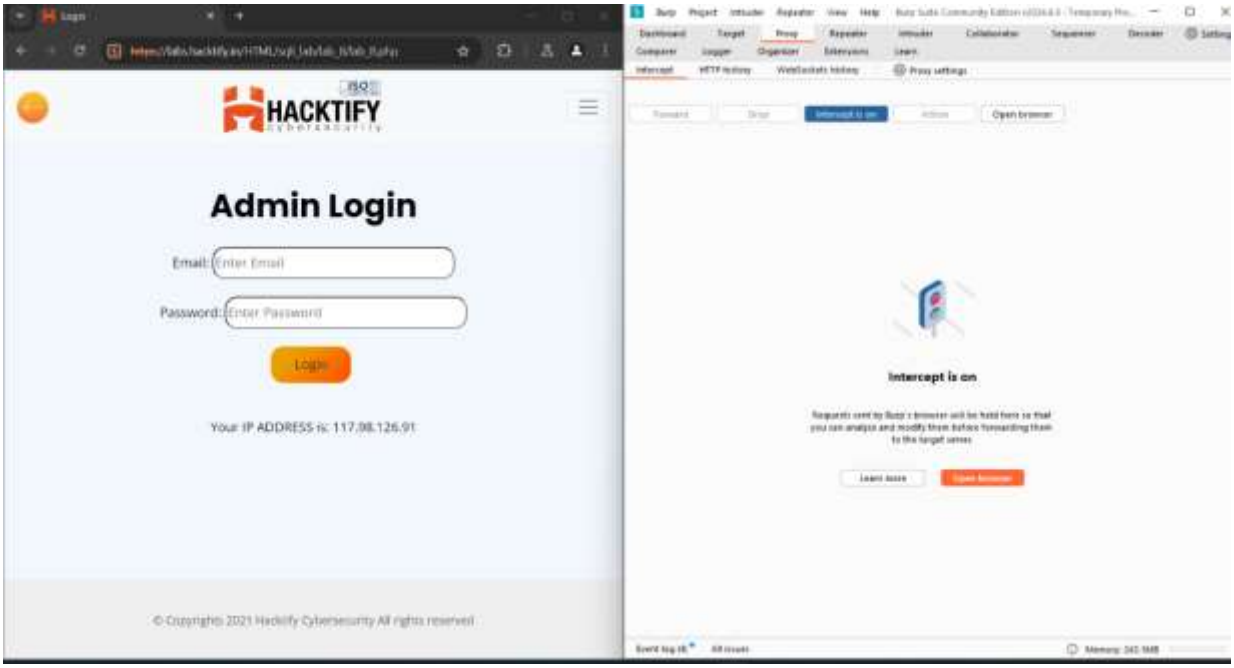

HACKTIFY
Admin Login
Email: Enter Email
Password: Enter Password
Login
Email: admin@gmail.com
Password: admin123
Successful Login
© Copyrights 2021 Hacktify Cybersecurity All rights reserved

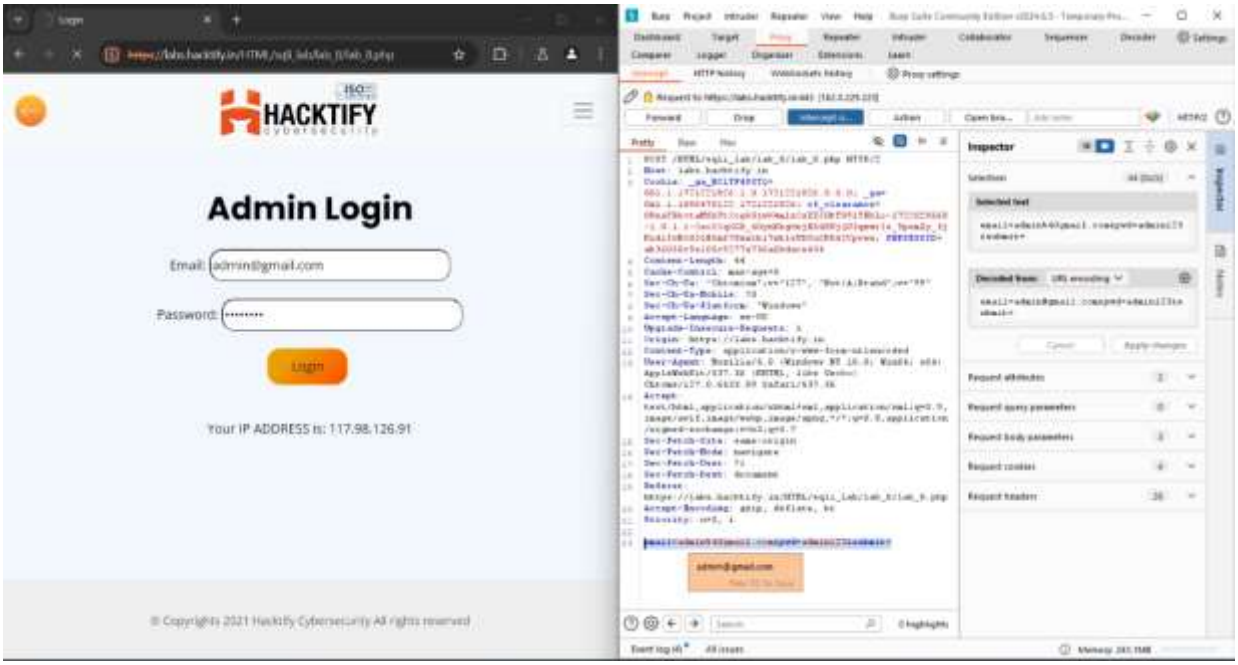
Proof of Concept

1.8. User Agents lead us!

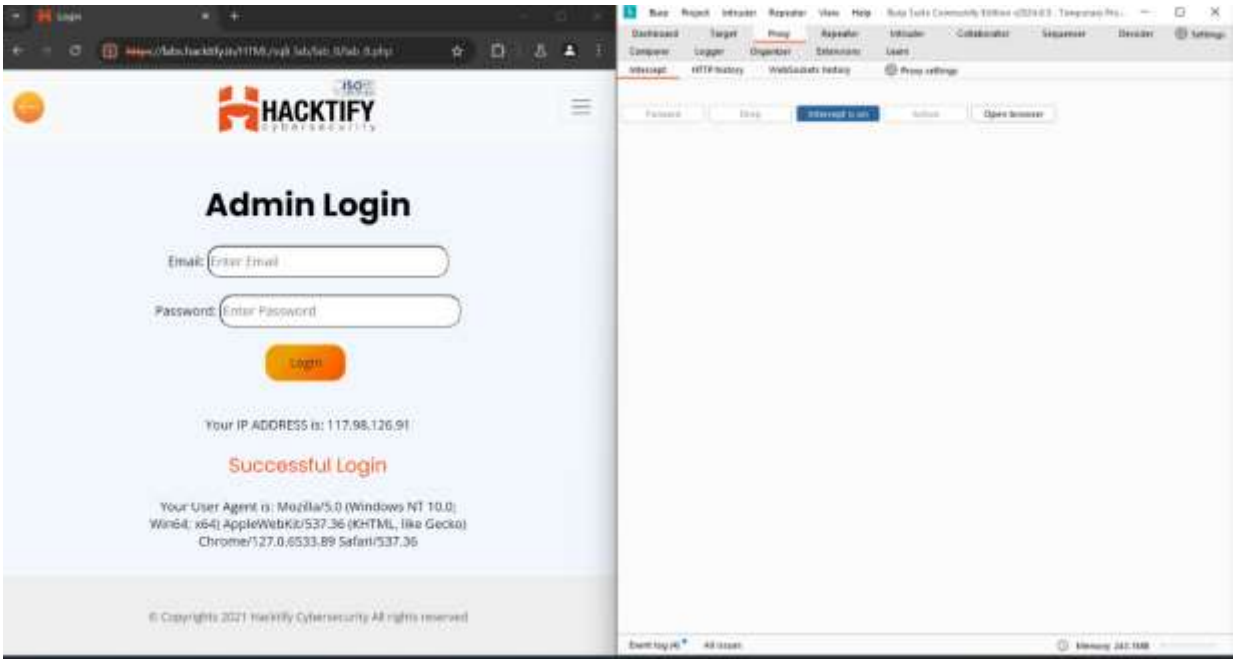
Reference	Risk Rating
Sub-lab-8: User Agents lead us!	**High**
Tools Used	
Burp Suite	
Vulnerability Description	
Attackers modify the “User-Agent” header in their HTTP requests to include malicious SQL code. If the application logs or processes the “User-Agent” header without proper sanitization, the malicious code can be executed as part of an SQL query. This can lead to unauthorized access, data manipulation, or even full control over the database.	
How It Was Discovered	
Automated Tool----Burp-suite	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_8/lab_8.php	
Consequences of not Fixing the Issue	
• **Data Breaches**: Unauthorized access to sensitive data. • **Data Manipulation**: Alteration or deletion of data. • **System Compromise**: Potential control over the entire database server	
Suggested Countermeasures	
1. **Input Validation**: Sanitize and validate all inputs, including HTTP headers. 2. **Parameterized Queries**: Use parameterized queries or prepared statements to handle inputs safely. 3. **Least Privilege**: Apply the principle of least privilege to database accounts. 4. **Web Application Firewalls (WAF)**: Deploy a WAF to filter and monitor incoming traffic for malicious patterns.	
References	
https://portswigger.net/kb/issues/00100200_sql-injection https://owasp.org/www-community/attacks/SQL_Injection https://unsql.ai/sql-security/injection-in-sql/	

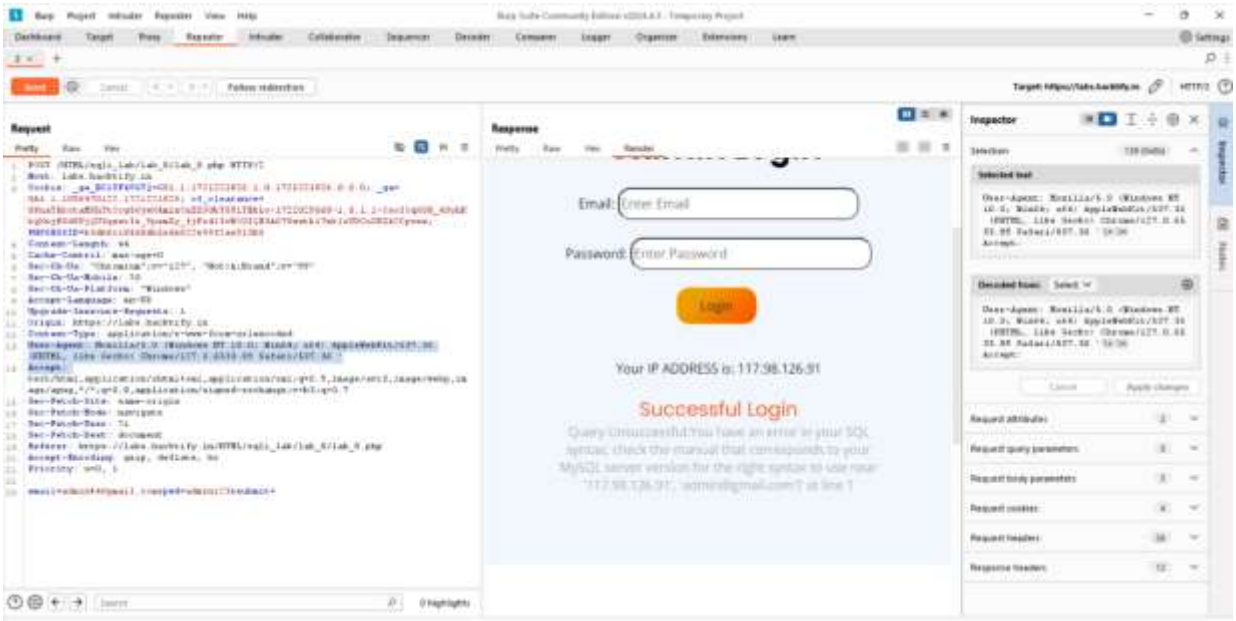
This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab

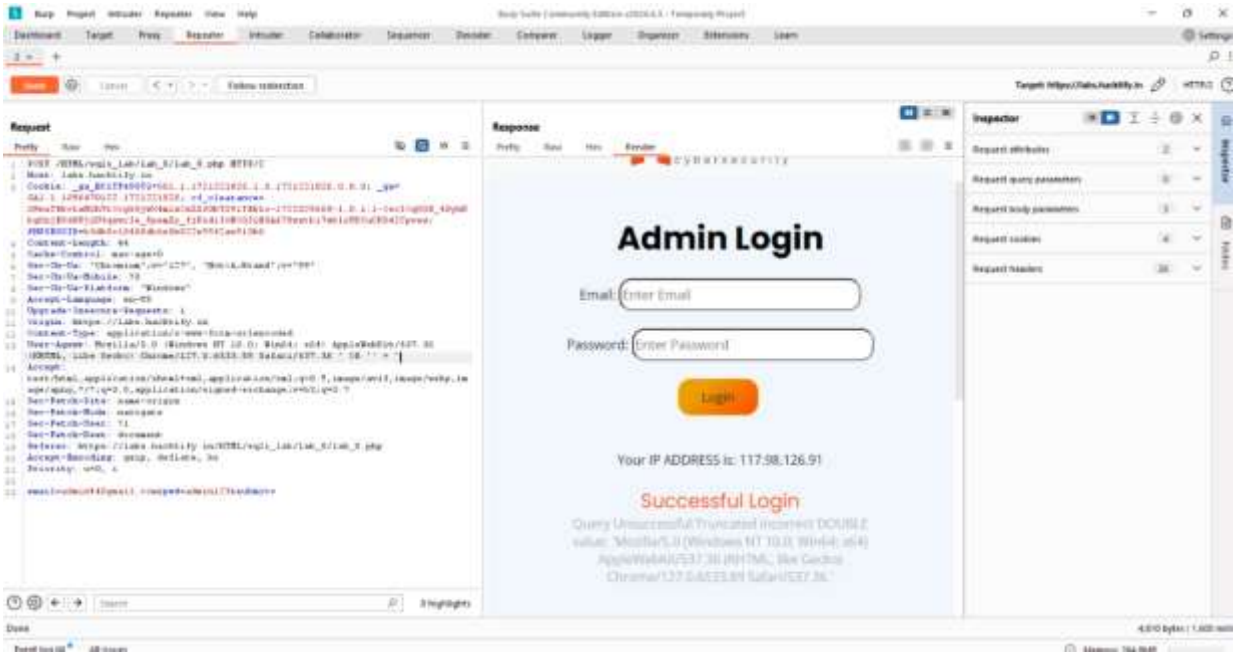




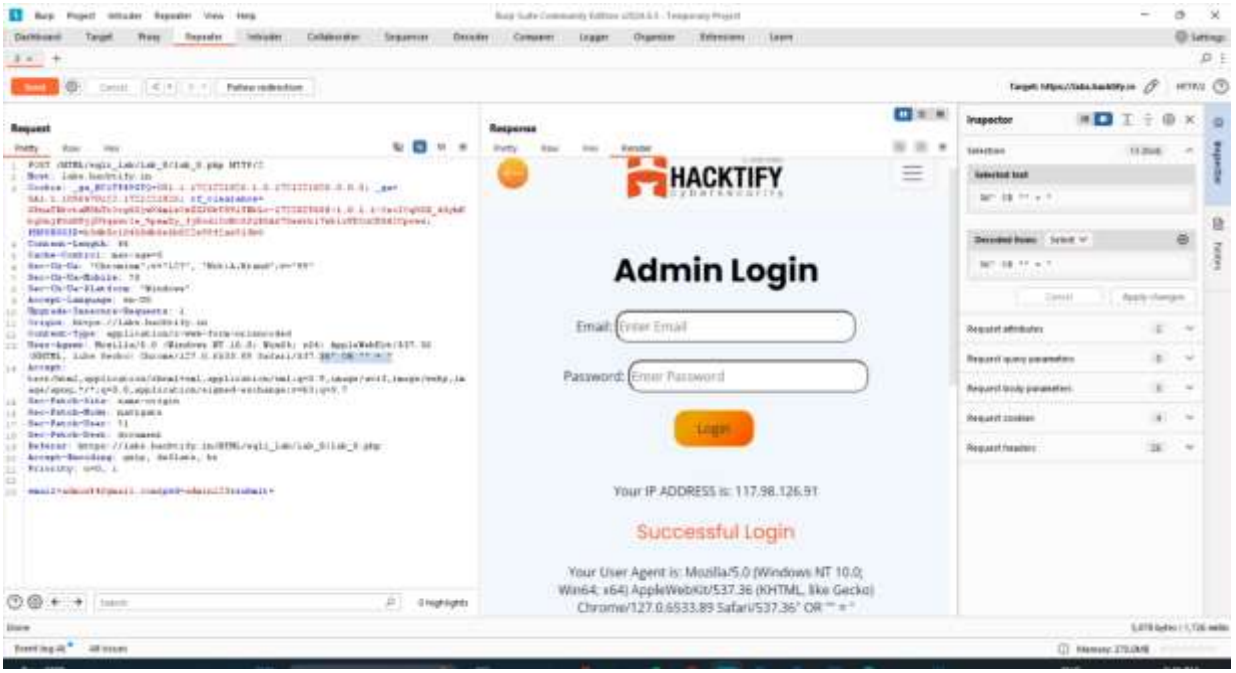
Proof of Concept







=>Vulnerable to SQl Injection



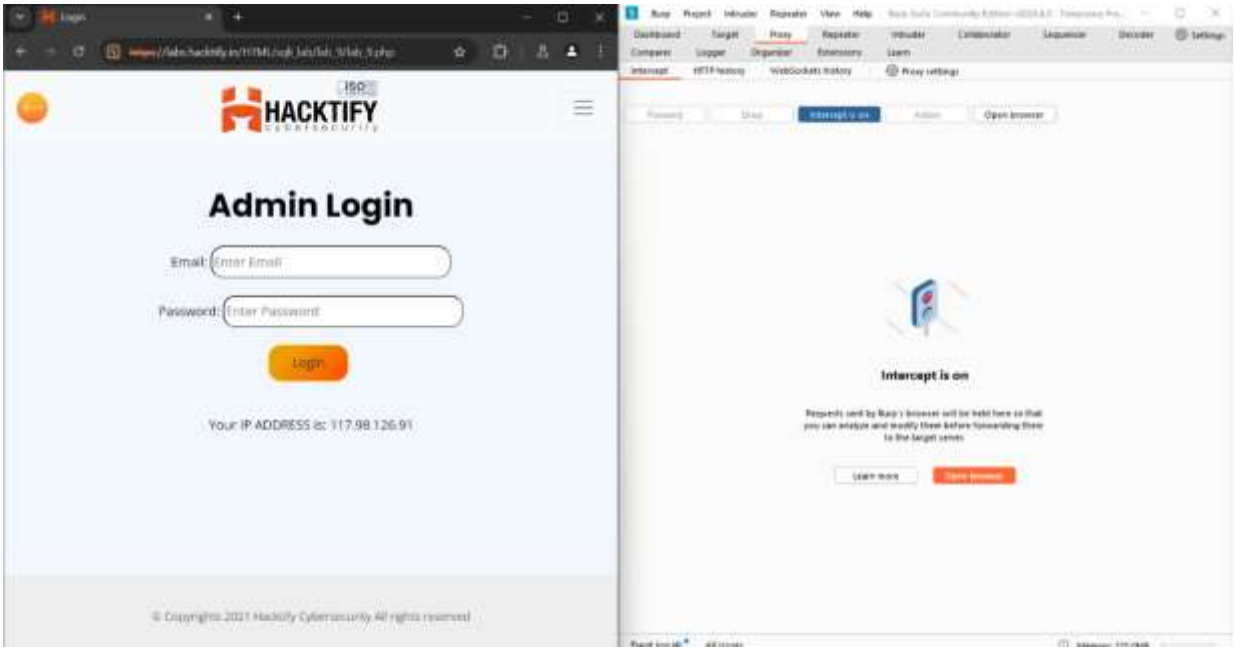
Proof of Concept

1.9. Referer lead us!

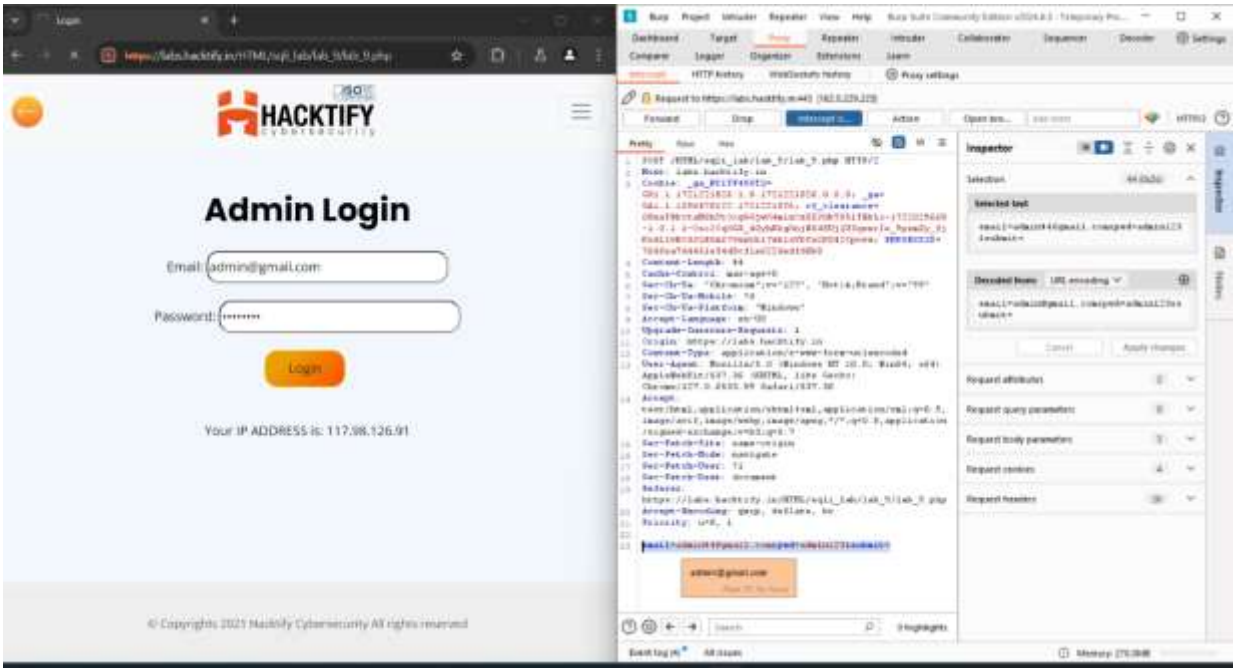
Reference	Risk Rating
Sub-lab-9: Referer lead us!	**Medium**
Tools Used	
Burp Suite	
Vulnerability Description	
Attackers modify the “User-Agent” header in their HTTP requests to include malicious SQL code. If the application logs or processes the “User-Agent” header without proper sanitization, the malicious code can be executed as part of an SQL query. This can lead to unauthorized access, data manipulation, or even full control over the database.	
How It Was Discovered	
Automated Tool----Burp-suite	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_9/lab_9.php	
Consequences of not Fixing the Issue	
• **Data Breaches**: Unauthorized access to sensitive data. • **Data Manipulation**: Alteration or deletion of data. • **System Compromise**: Potential control over the entire database server	
Suggested Countermeasures	
• **Input Validation**: Sanitize and validate all inputs, including HTTP headers. • **Parameterized Queries**: Use parameterized queries or prepared statements to handle inputs safely. • **Least Privilege**: Apply the principle of least privilege to database accounts. • **Web Application Firewalls (WAF)**: Deploy a WAF to filter and monitor incoming traffic for malicious patterns.	
References	

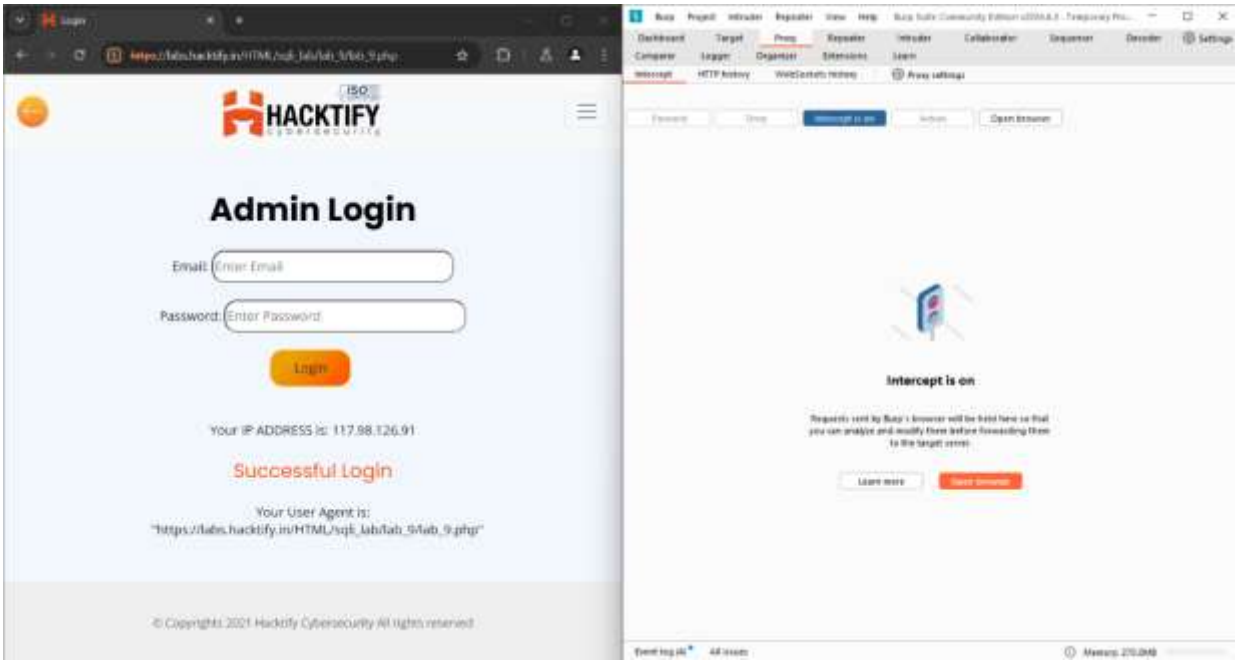
https://portswigger.net/kb/issues/00100200_sql-injection
https://owasp.org/www-community/attacks/SQL_Injection
https://unsql.ai/sql-security/injection-in-sql/

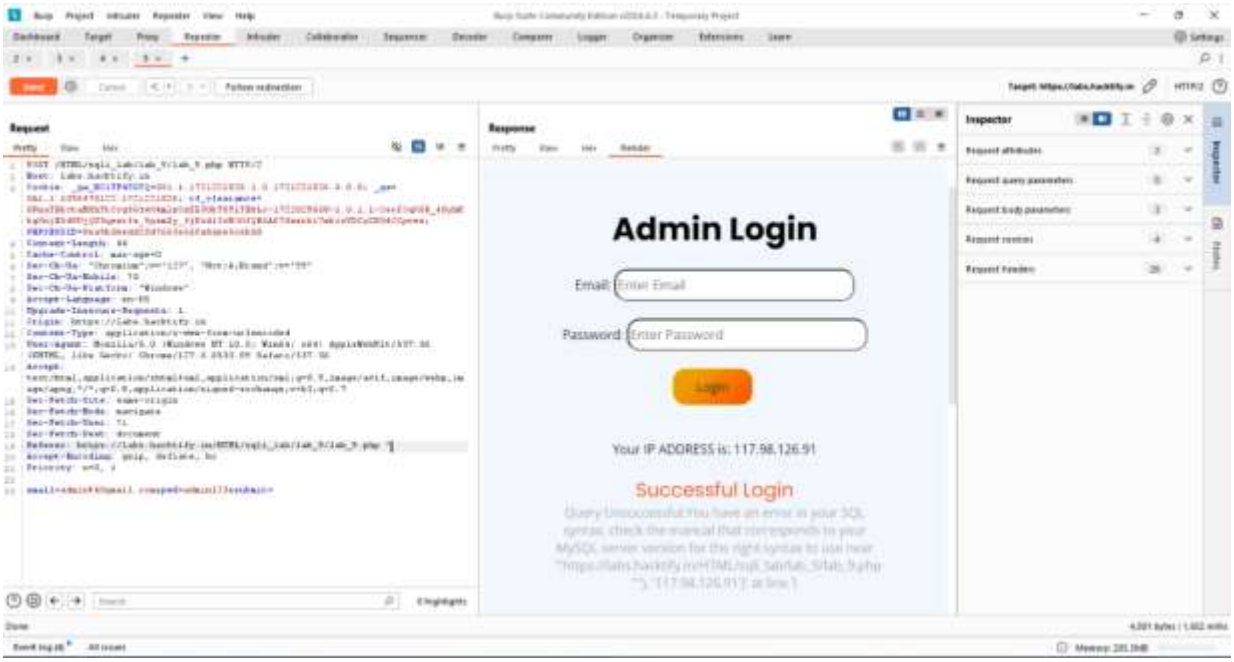
This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab

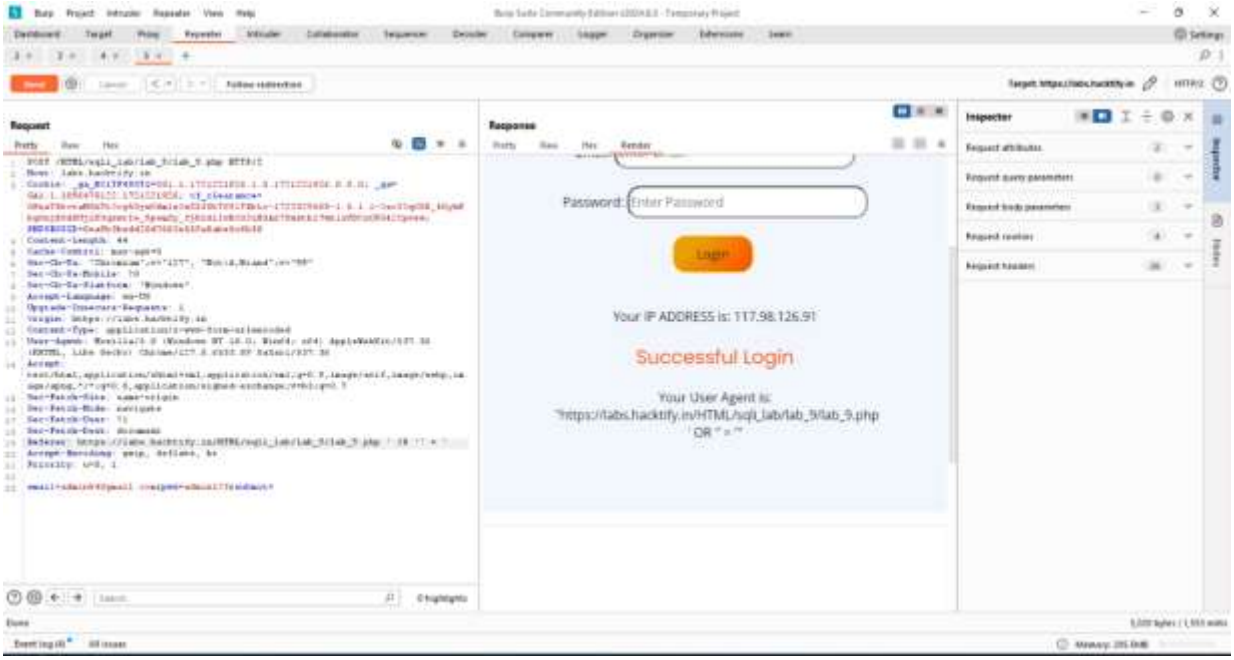


Proof of Concept






Admin Login
Email: Enter Email
Password: Enter Password
Login
Your IP ADDRESS is: 117.98.126.91
Successful Login

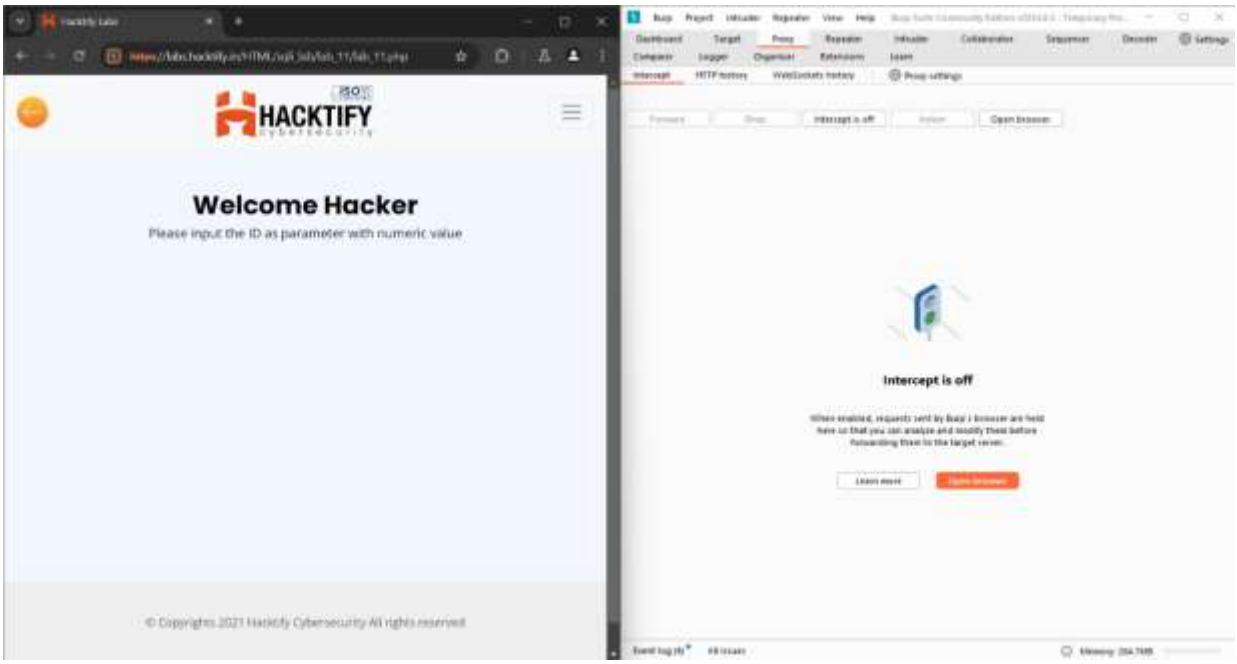

Password: Enter Password
Login
Your IP ADDRESS is: 117.98.126.91
Successful Login
Your User Agent is:
"https://labs.hacktify.in/HTML/sqli_lab/lab_9/lab_9.php
' OR " = "'

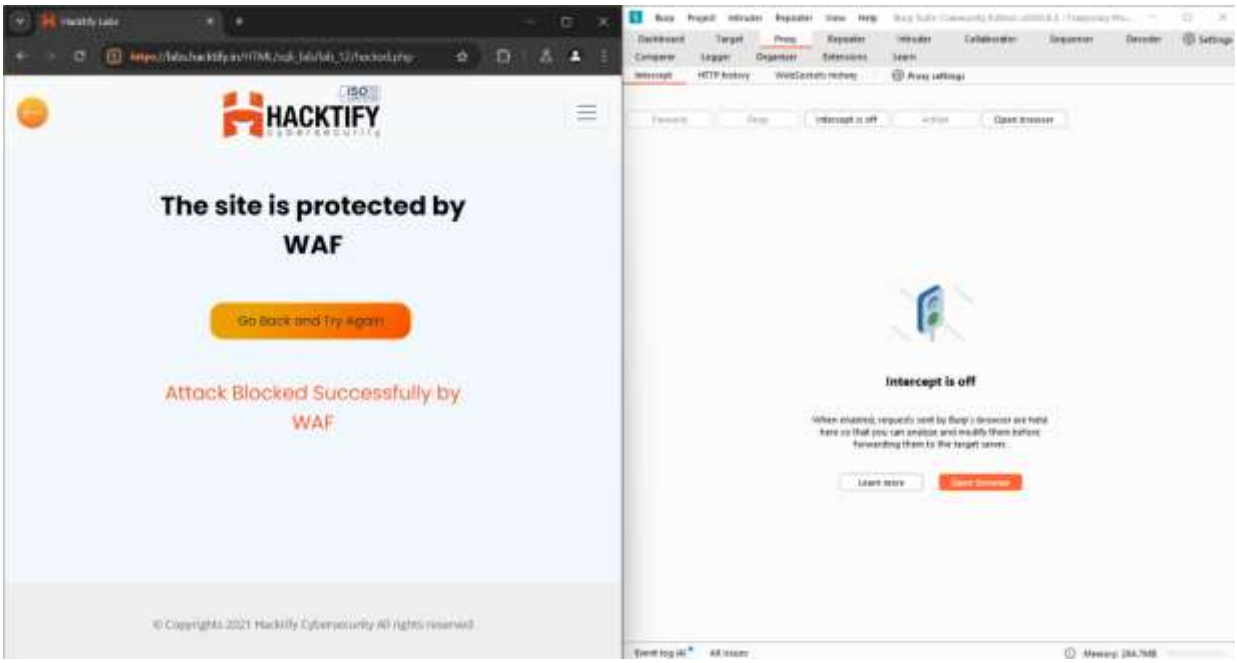
Proof of Concept

1.10 WAF's are injected!

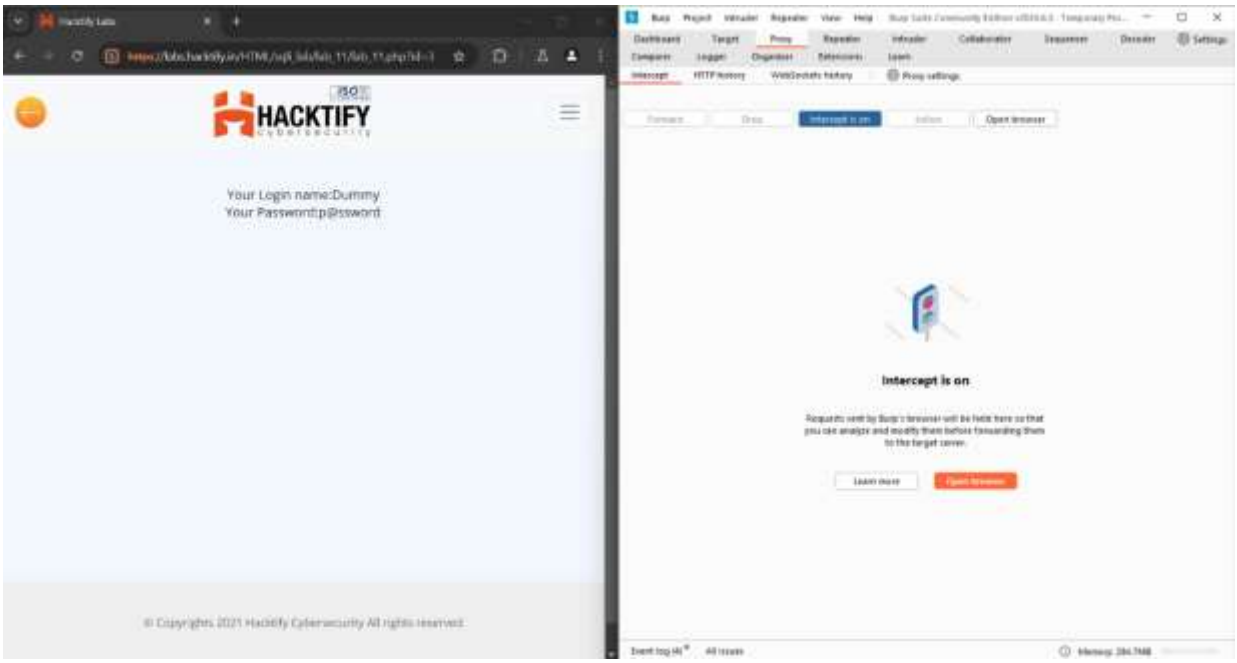
Reference	Risk Rating
Sub-lab-11: WAF's are injected	**High**
Tools Used	
Burp Suite	
Vulnerability Description	
Attackers use sophisticated evasion techniques to manipulate SQL queries in a way that bypasses WAF filters. This can include encoding, obfuscation, and altering the structure of the SQL payload.	
How It Was Discovered	
Automated Tool----Burp-suite	
Vulnerable URLs	
https://labs.hacktify.in/HTML/sqli_lab/lab_12/hacked.php	
Consequences of not Fixing the Issue	
• **Data Breaches**: Unauthorized access to sensitive data. • **Data Manipulation**: Alteration or deletion of data. • **System Compromise**: Potential control over the entire database server	
Suggested Countermeasures	
1. **Advanced WAF Configuration**: Regularly update and configure WAF rules to handle new evasion techniques. 2. **Input Validation**: Implement robust input validation and sanitization on the server side. 3. **Parameterized Queries**: Use parameterized queries or prepared statements to handle inputs safely. 4. **Regular Security Audits**: Conduct regular security audits and penetration testing to identify and fix potential vulnerabilities.	
References	
https://portswigger.net/web-security/sql-injection https://www.varonis.com/blog/what-is-sql-injection https://owasp.org/www-community/attacks/SQL_Injection	

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab





Proof of Concept



2. Cross-Site Request Forgery

2.1. Eassyy CSRF

Reference	Risk Rating
Sub-lab-1: Eassyy CSRF	**Low**
Tools Used	
Burp-Suite .Poc Generator	
Vulnerability Description	
• CSRF exploits the trust that a web application has in the user's browser. When a user is authenticated to a site, their browser automatically includes their session cookies with every request. • An attacker crafts a malicious request and tricks the user into executing it. This can be done through social engineering techniques, such as sending a link via email or embedding it in a malicious website. • The malicious request inherits the user's identity and privileges, causing the web application to execute actions on behalf of the user without their consent.	
How It Was Discovered	
Automated Tools – Browser Inspect /Proxy /Burp-Suite	

Vulnerable URLs
https://labs.hacktify.in/HTML/csrf_lab/lab_1/login.php

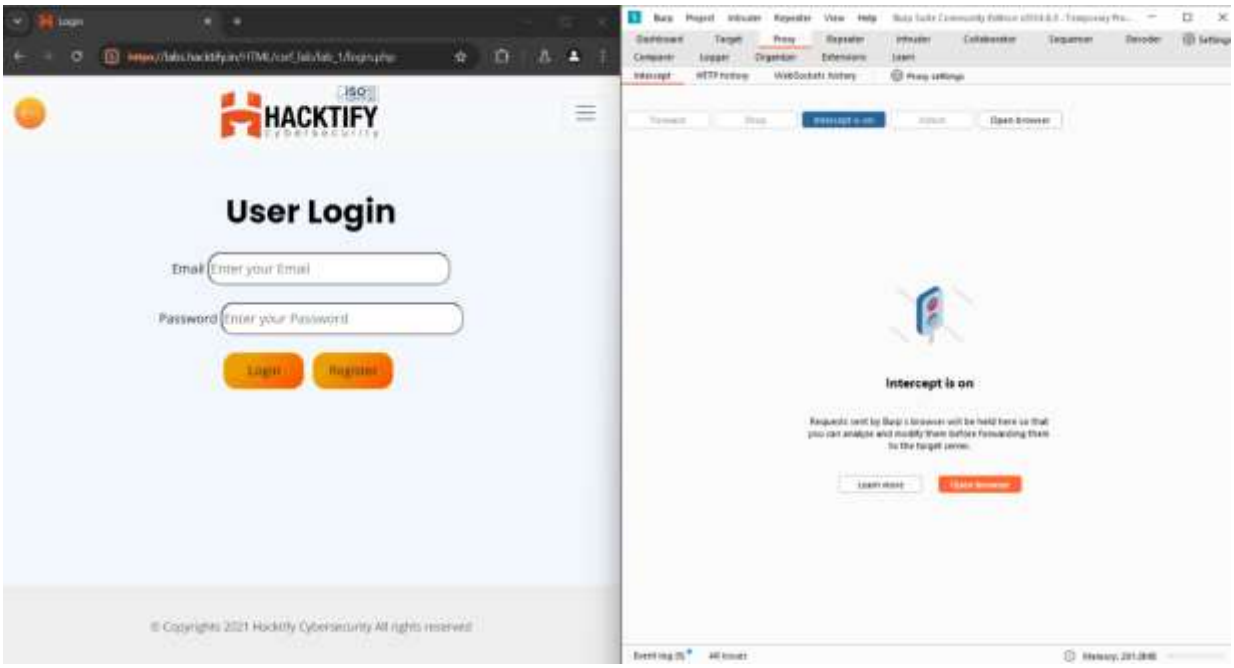
Consequences of not Fixing the Issue
• **Unauthorized Transactions**: Attackers can perform unauthorized transactions, such as transferring funds or making purchases. • **Data Manipulation**: Attackers can change user settings, such as email addresses or passwords. • **Compromise of Entire Application**: If the victim is an administrative user, the attacker can potentially compromise the entire web application.

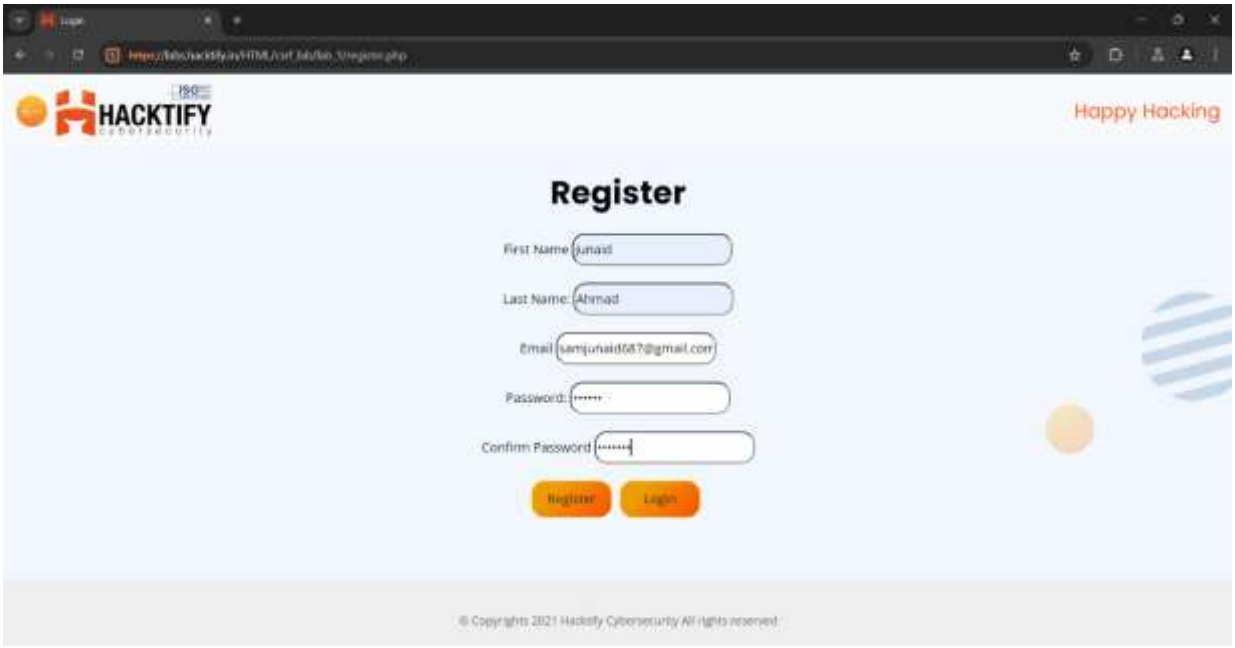
Suggested Countermeasures
• **Anti-CSRF Tokens**: Use unique tokens for each session or request to verify the legitimacy of requests. • **SameSite Cookies**: Set cookies with the SameSite attribute to prevent them from being sent with cross-site requests. • **Referer Header Validation**: Check the Referer header to ensure requests are coming from trusted sources. • **User Interaction**: Require user interaction (e.g., CAPTCHA) for critical actions to ensure they are intentional. •

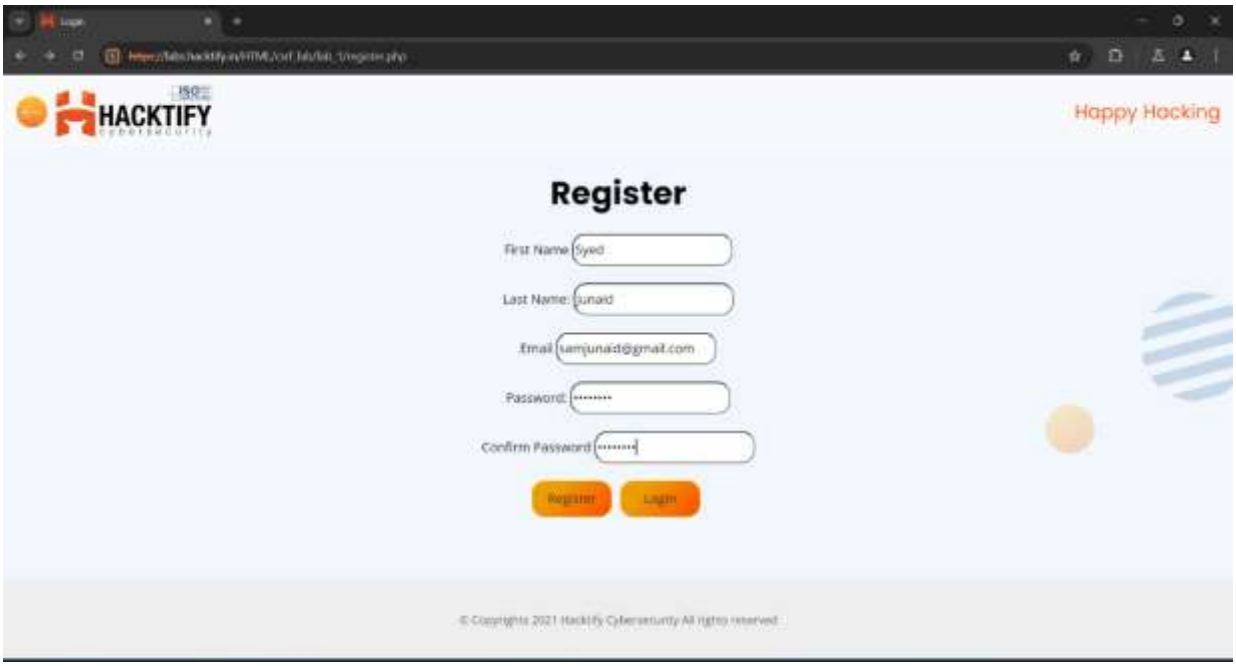
References
https://www.rapid7.com/fundamentals/cross-site-request-forgery/ https://owasp.org/www-community/attacks/csrf

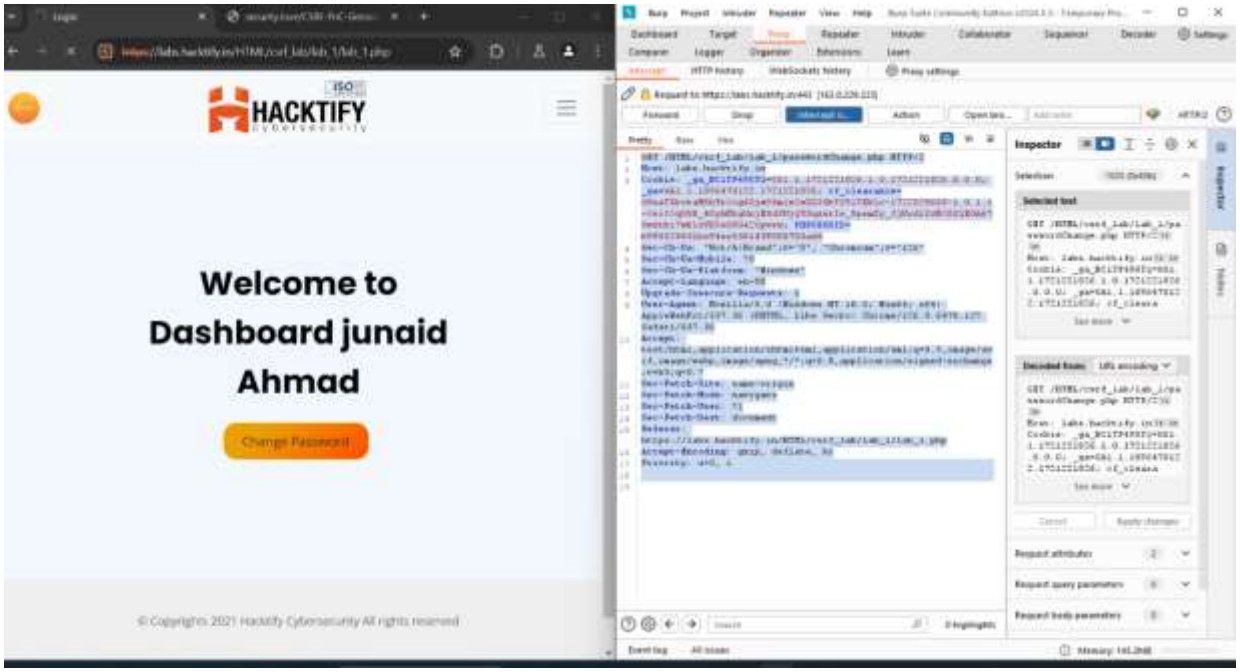
This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab

Proof of Concept

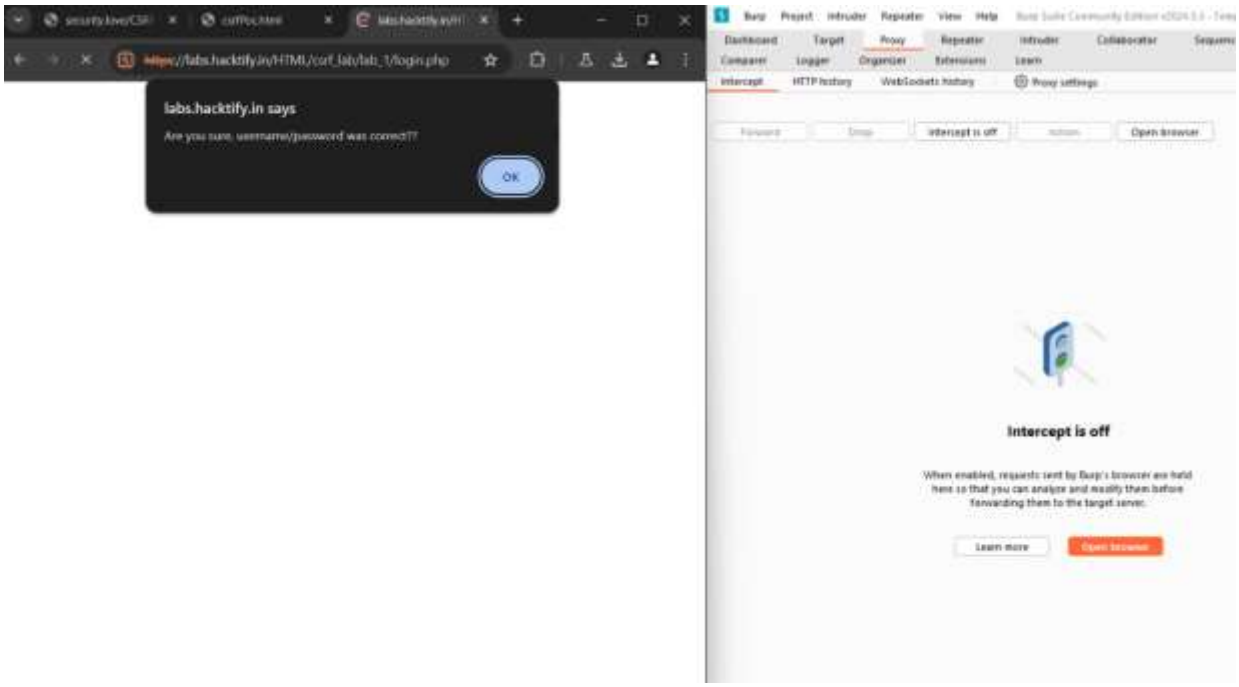


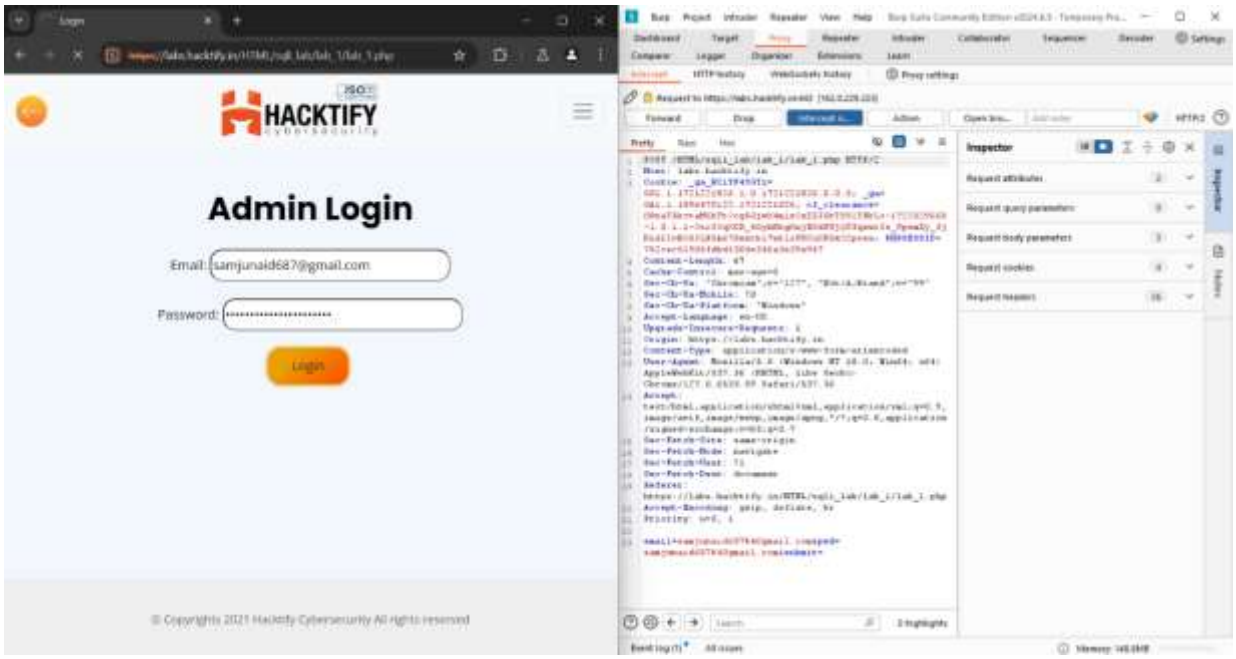


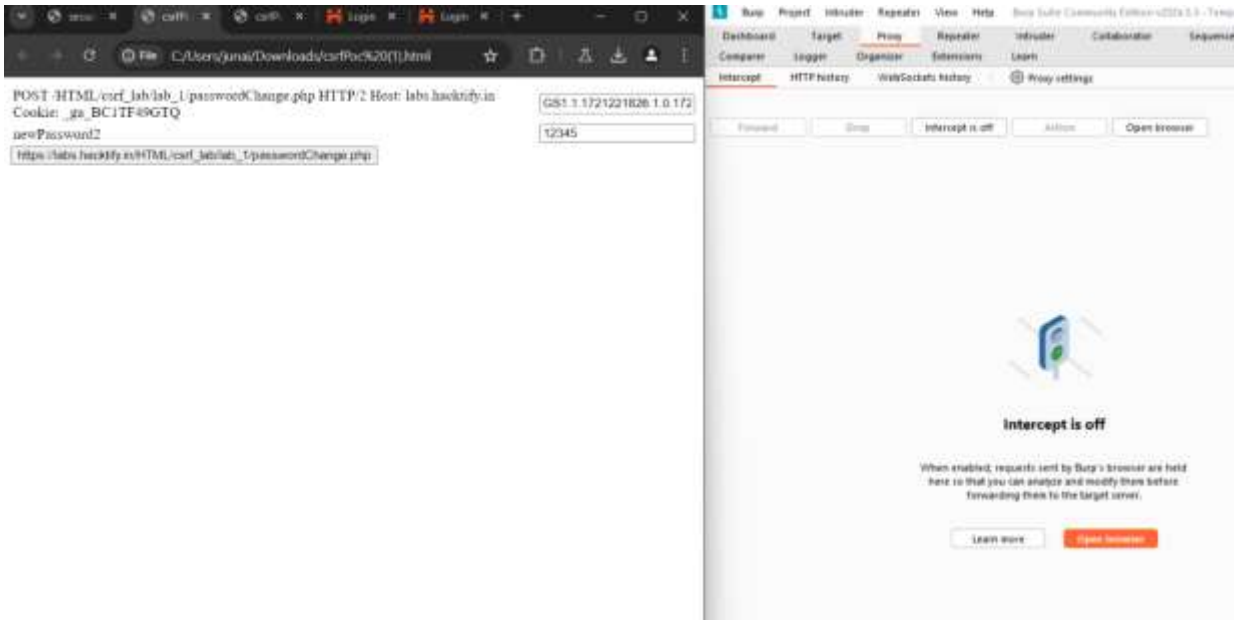

HACKTIFY
Happy Hacking
Register
First Name: Syed
Last Name: Junaid
Email: samjunaid@gmail.com
Password:
Confirm Password:
Register
Login
© Copyrights 2021 Hacktify Cybersecurity All rights reserved

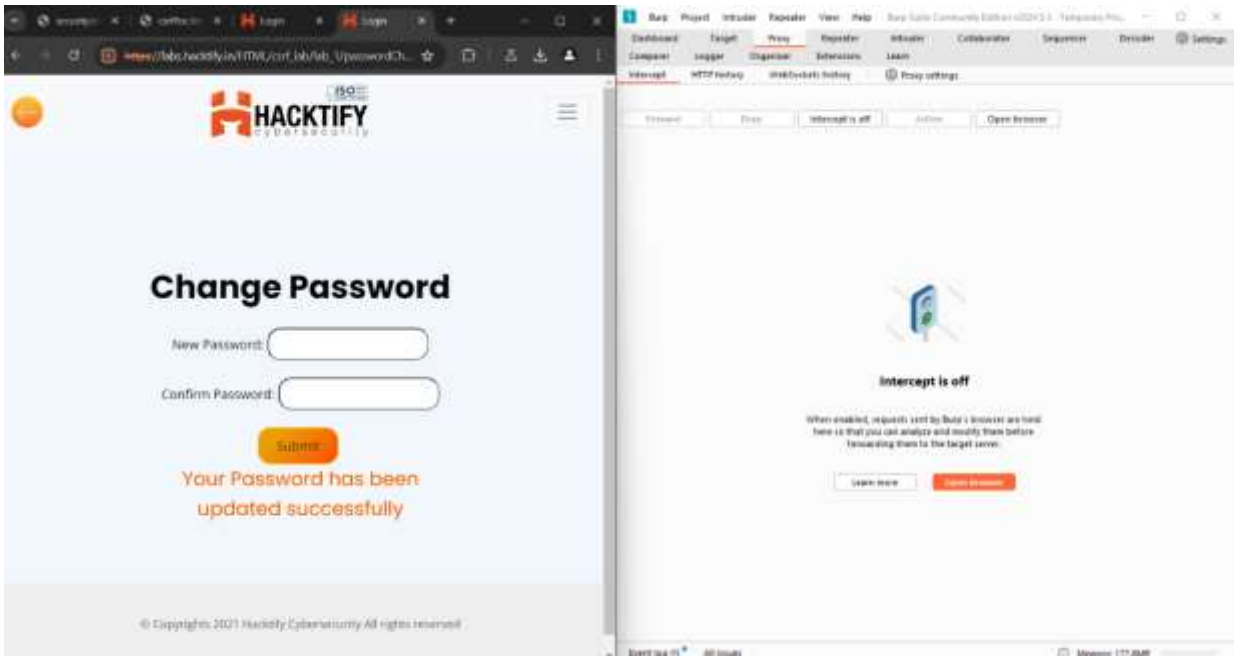

HACKTIFY
Welcome to
Dashboard junaid
Ahmad
Change Password
© Copyrights 2021 Hacktify Cybersecurity All rights reserved
Inspector

Proof of Concept






POST /HTML/csrf_lab/lab_1/passwordChange.php HTTP/2 Host: labs.hacktify.in
Cookie: _ga_BC1TF49GTQ
newPassword2
12345
Intercept is off


HACKTIFY
Change Password
New Password:
Confirm Password:
Submit
Your Password has been updated successfully
Intercept is off

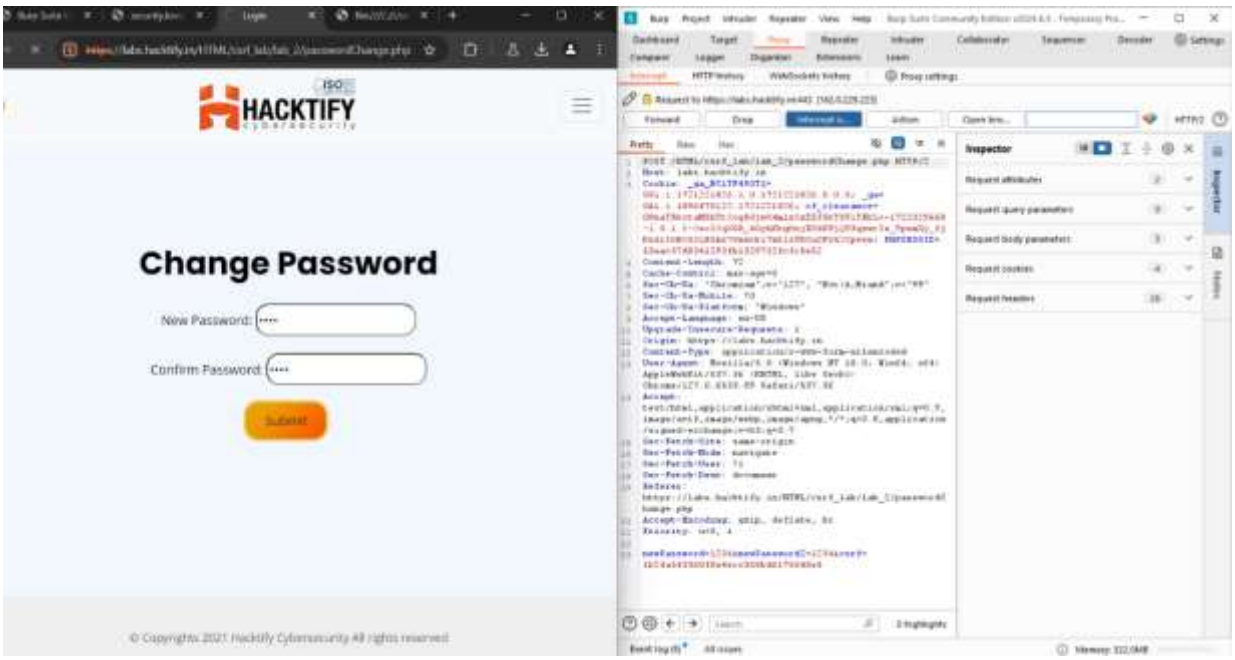
Proof of Concept

2.2. Always Validate Tokens

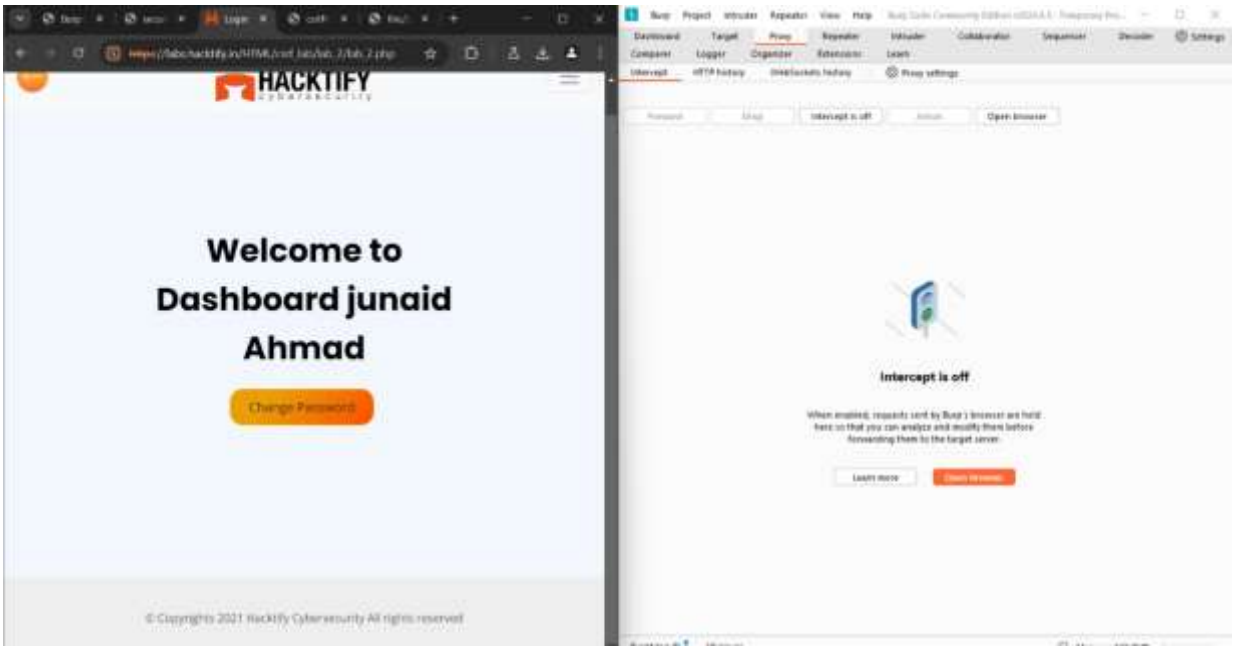
Reference	Risk Rating
Sub-lab-2: Always Validate Tokens	**Medium**
Tools Used	
Burp-Suite, Poc Generator	
Vulnerability Description	
• CSRF exploits the trust that a web application has in the user's browser. When a user is authenticated to a site, their browser automatically includes their session cookies with every request. • An attacker crafts a malicious request and tricks the user into executing it. This can be done through social engineering techniques, such as sending a link via email or embedding it in a malicious website. • The malicious request inherits the user's identity and privileges, causing the web application to execute actions on behalf of the user without their consent.	
How It Was Discovered	
Automated Tools – Proxy. (Burp-Suite)	
Vulnerable URLs	
https://labs.hacktify.in/HTML/csrf_lab/lab_2/login.php	
Consequences of not Fixing the Issue	
• **Unauthorized Transactions**: Attackers can perform unauthorized transactions, such as transferring funds or making purchases. • **Data Manipulation**: Attackers can change user settings, such as email addresses or passwords. • **Compromise of Entire Application**: If the victim is an administrative user, the attacker can potentially compromise the entire web application.	
Suggested Countermeasures	
• **Anti-CSRF Tokens**: Use unique tokens for each session or request to verify the legitimacy of requests. • **SameSite Cookies**: Set cookies with the SameSite attribute to prevent them from being sent with cross-site requests. • **Referer Header Validation**: Check the Referer header to ensure requests are coming from trusted sources. • **User Interaction**: Require user interaction (e.g., CAPTCHA) for critical actions to ensure they are intentional.	

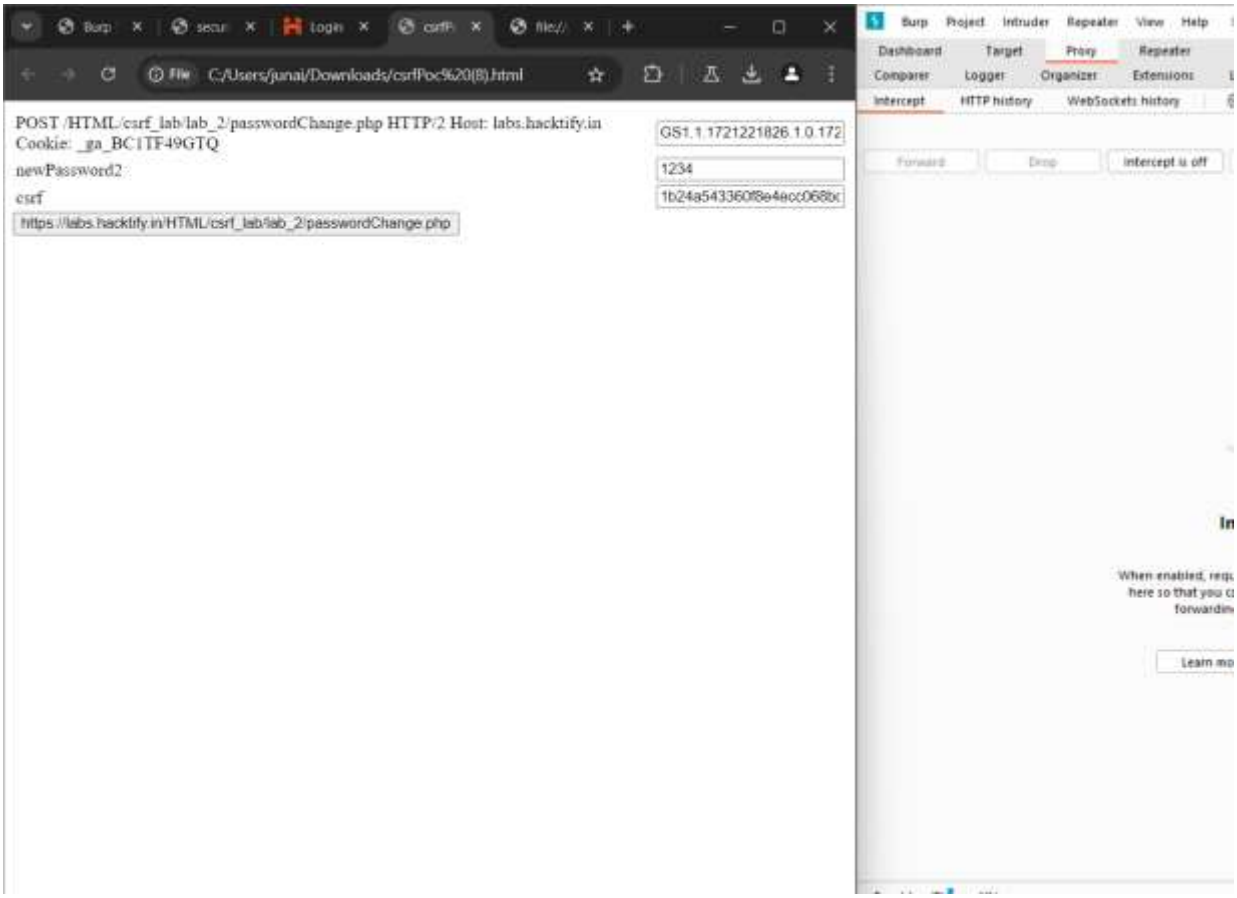
References
https://www.rapid7.com/fundamentals/cross-site-request-forgery/ https://owasp.org/www-community/attacks/csrf https://brightsec.com/blog/cross-site-request-forgery-csrf/

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab

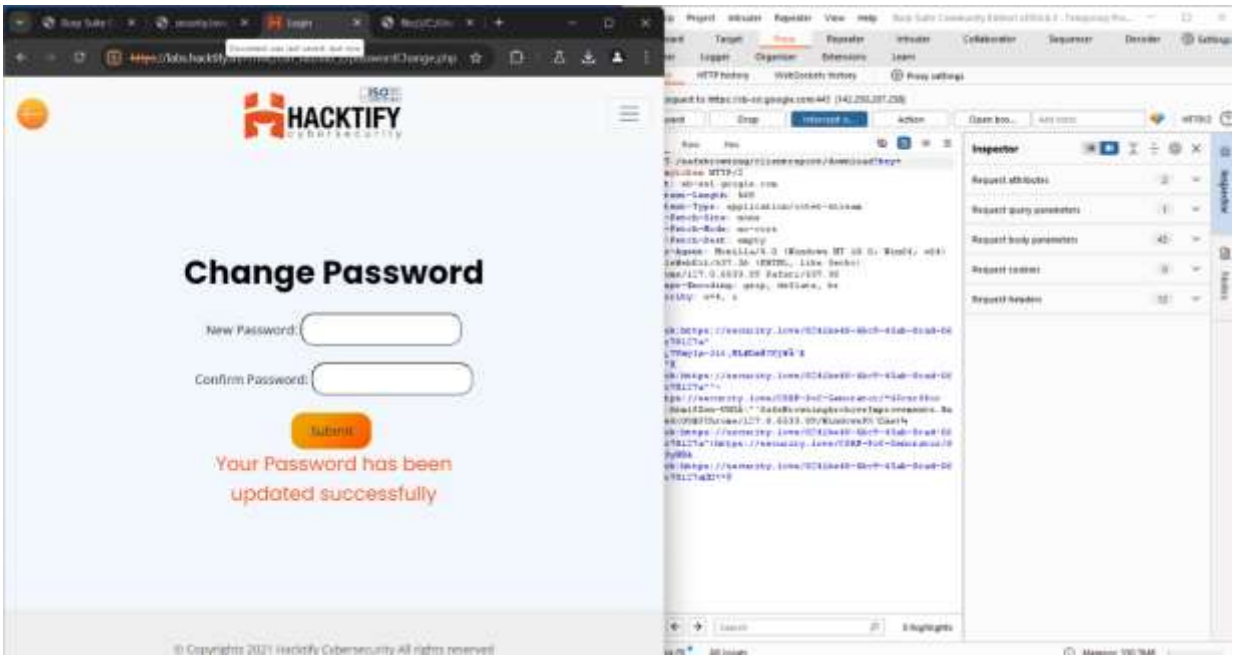


Proof of Concept





lla



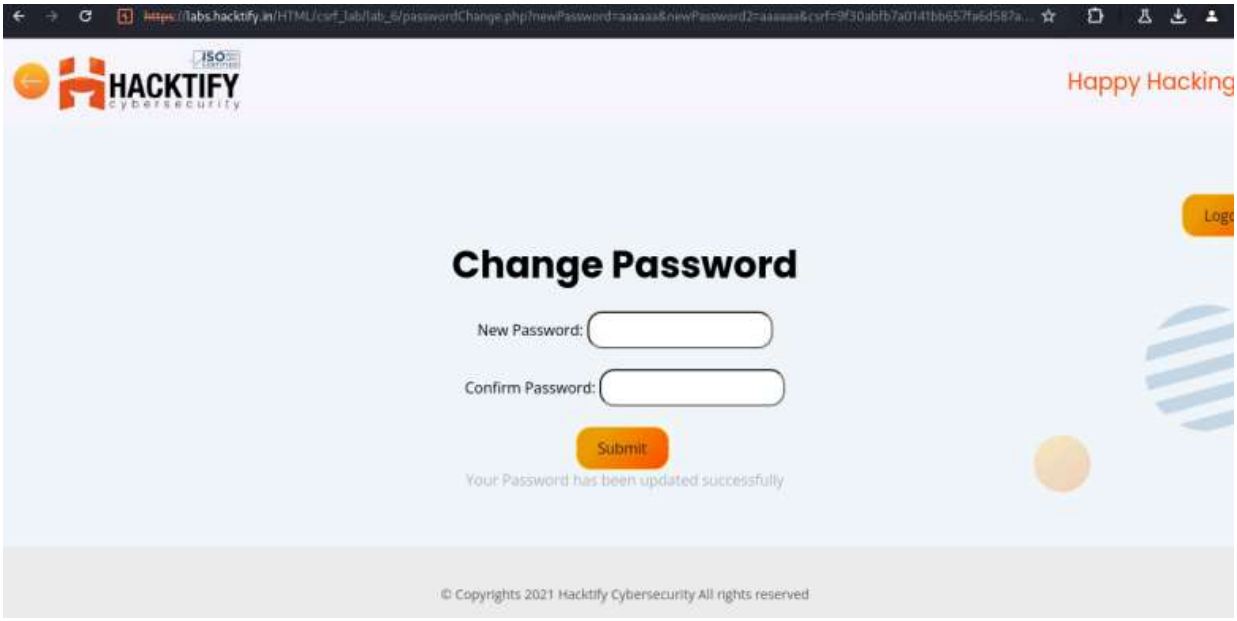
2.3. GET Me or POST ME

Reference	Risk Rating
Sub-lab-4: GET Me or POST ME	low
Tools Used	
Burp-Suite	

Proof of Concept

Vulnerability Description
• **Exploitation of Trust**: CSRF exploits the trust that a web application has in the user's browser. When a user is authenticated to a site, their browser automatically includes their session cookies with every request. • **Malicious Requests**: An attacker crafts a malicious request and tricks the user into executing it. This can be done through social engineering techniques, such as sending a link via email or embedding it in a malicious website. • **Unintended Actions**: The malicious request inherits the user's identity and privileges, causing the web application to execute actions on behalf of the user without their consent.
How It Was Discovered
Automated Tools –Burp-suite
Vulnerable URLs
https://labs.hacktify.in/HTML/csrf_lab/lab_6/login.php
Consequences of not Fixing the Issue
• **Unauthorized Transactions**: Attackers can perform unauthorized transactions, such as transferring funds or making purchases. • **Data Manipulation**: Attackers can change user settings, such as email addresses or passwords. • **Compromise of Entire Application**: If the victim is an administrative user, the attacker can potentially compromise the entire web application
Suggested Countermeasures
• **Anti-CSRF Tokens**: Use unique tokens for each session or request to verify the legitimacy of requests. • **SameSite Cookies**: Set cookies with the `SameSite` attribute to prevent them from being sent with cross-site requests. • **Referer Header Validation**: Check the `Referer` header to ensure requests are coming from trusted sources. • **User Interaction**: Require user interaction (e.g., CAPTCHA) for critical actions to ensure they are intentional.
References
https://brightsec.com/blog/cross-site-request-forgery-csrf/ https://www.cloudflare.com/learning/security/threats/cross-site-request-forgery/ https://owasp.org/www-community/attacks/csrf

This section contains the proof of the above vulnerabilities as the screenshot of the vulnerability of the lab


HACKTIFY
cybersecurity
Happy Hacking
Change Password
New Password:
Confirm Password:
Submit
Your Password has been updated successfully
© Copyrights 2021 Hacktify Cybersecurity All rights reserved