



**QRB Platform**

Web based Learning platform

QRB Platform | Graduation Project | 2024



Department of Computer Science

Faculty Information Technology and Computer Science

Yarmouk University

Irbid, Jordan

A picture containing shape

Description automatically generated

**QRB**

**Web-based Learning platform**

**QASIM YAHYA TAWALBEH 2021904133**

**RAYA IBRAHIM BAWATI 2021904132**

**BATOOL MOHAMMAD QAZAN 2021904139**

Department of Information Technology/ Cybersecurity program

Faculty Information Technology and Computer Science

Yarmouk University

Irbid, Jordan

***Contact Information***

*This project report is submitted to the Department of Information Technology/ Cybersecurity program at Yarmouk University in partial fulfillment of the requirements for the degree of Bachelor of Information Technology in Cybersecurity.*

***Author(s):***

*Qasim Tawalbeh ,2021904133*

*Address: Irbid*

*E-mail*: kasemtawalbeh56@gmail.com

*Raya Bawati ,2021904132*

*Address: Irbid*

*E-mail*: ibrahimraya263@gmail.com

*Batool Qazan ,2021904139*

*Address: Irbid*

*E-mail*: batoolqazan11111@gmail.com

***University supervisor(s):***

*Eng. Mohammad Jaafrah*

*Department of Information Technology/* Cybersecurity *program Internet: http://yu.edu.jo* Faculty *of Information Technology Phone:*  *and Computer Sciences*

*Yarmouk University Fax: +962 27211111*

*Jordan*

***Intellectual Property Right Declaration***

*This is to declare that the work under the supervision of Eng.* Mohammad Jaafrah *having title “QRB* platform*” carried out in partial fulfillment of the requirements of Bachelor of Cybersecurity, is the sole property of the Yarmouk University and the respective supervisor and is protected under the intellectual property right laws and conventions. It can only be considered/ used for purposes like extension for further enhancement, product development, adoption for commercial/organizational usage, etc., with the permission of the University and respective supervisor.*

*Date: 2/06/2024*

**Author(s):**

Name: QASIM TAWALBEH *Signature:*  Qasim

Name: RAYA IBRAHIM *Signature:*  Raya

Name: BATOOL QAZAN *Signature: Batool*

***Supervisor(s):***

*Eng. Mohammad Jaafrah*

*Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**Anti-Plagiarism Declaration**

This is to declare that the above publication produced under the supervision of Eng. **MOHAMMAD JAFFRAH** having the title “QRB Platform” is the sole contribution of the author(s) and no part hereof has been reproduced illegally (cut and paste) which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. I/We will be responsible and liable for any consequence if a violation of this declaration is proven.

Date:

2/06/2024

**Author(s):**

Name: QASIM TAWALBEH *Signature:*  Qasim

Name: RAYA IBRAHIM *Signature:*  Raya

Name: BATOOL QAZAN *Signature: Batool*

**ACKNOWLEDGMENT**

In the name of Allah, the Most Gracious, the Most Merciful

All praise and greatness to Allah, Lord of the Two Worlds and may Allah's prayers and peace be upon our Prophet Muhammad and his companions.

We would like to thank to our honorable Eng-Mohammad Rasool AlJaafrah, for his support and guidance in completing our project on the topic (Web Based Platform Learning). It was a great learning experience.

In addition, we extend Special thanks to my advisor, friends, and family

for their unwavering support and encouragement.

I am also grateful to the QRB Platform team for them

valuable insights and expertise. Your contributions

have been key to the success of my project.

Thank you all for being a part of this incredible journey!

**ABSTRACT**

QRB Platform presents a comprehensive web-based learning platform designed

to educate users about various web security vulnerabilities. The platform

includes detailed pages on common vulnerabilities like SQL Injection,

Cross-Site Scripting (XSS), File Upload vulnerabilities, and File Inclusion vulnerabilities.

Each page provides thorough descriptions, and prevention techniques, also web pages intentionally introduced with vulnerability To Simulate real-world examples (Attack).

By offering interactive content and practical exercises,

This platform aims to enhance users' understanding and awareness of web security issues, ultimately contributing to safer web development practices.

Table of Figure

1. Figure 1: XSS page ……………………………………………………………………………………………………………………16
2. Figure 2: SQL Injection Figure………………………………………………………………………………………………….17
3. Figure 3: File INCLUSION ………………………………………………………………………………………………………….18
4. Figure 4: File Upload Figure ………………………………………………………………………………………………………19
5. Figure 5: Application Design …………………………………………………………………………………………………….22
6. Figure 6: Account Creation ……………………………………………………………………………………………………….24
7. Figure 7: login process …………………………………………………………………………………………………………….25
8. Figure 8: Module Selection ……………………………………………………………………………………………………….26
9. Figure 8.1: Module Selection…………………………………………………………………………………………………… 27
10. Figure 9: Interactive Labs …………………………………………………………………………………………………………28
11. Figure 10: database Figure ………………………………………………………………………………………………………30
12. Figure 11: SQL Malicious input………………………………………………………………………………………………… 32
13. Figure 12: SQL Prevention Techniques ………………………………………………………………………………………32
14. Figure 13: XSS Vulnerable code: ………………………………………………………………………………………………33
15. Figure 14: XSS Malicious Input: ………………………………………………………………………………………………33
16. Figure 15: XSS Reflected result …………………………………………………………………………………………………34
17. Figure 16: XSS Stored section …………………………………………………………………………………………………34
18. Figure 17 XSS Stored Malicious Comment: ………………………………………………………………………………35
19. Figure 18: XSS Stored Resulting: ………………………………………………………………………………………………35
20. Figure 19: SQL Injection Lab Interface ………………………………………………………………………………………41
21. Figure 20: SQL Results: …………………………………………………………………………………………………………… 41
22. Figure21: Reflected XSS Lab Interface ………………………………………………………………………………………43
23. Figure22: Stored XSS Lab Interface …………………………………………………………………………………………43
24. Figure 23: File Inclusion Lab Interface ………………………………………………………………………………………45
25. Figure 24: File Inclusion Lab test ………………………………………………………………………………………………46
26. Figure 25: File Inclusion Lab query ……………………………………………………………………………………………46
27. Figure 26: File Inclusion Lab result ……………………………………………………………………………………………47
28. Figure 27: File Upload Lab interface …………………………………………………………………………………………48
29. Figure 28: File Upload Lab Test ………………………………………………………………………………………………48
30. Figure 29: File Upload Lab result ………………………………………………………………………………………………49
31. Figure 29.1: File Upload Lab result ……………………………………………………………………………………………49
32. Figure 30: File Upload Lab Leve 2 Interface ………………………………………………………………………………50
33. Figure 31: File Upload Lab Leve 2 Result …………………………………………………………………………………

# QRB Platform Documentation

# Table of Contents

1. **Chapter One: Introduction**
   * Overview……………………………………………………………………………………………………………………11
   * Objectives……………………………………………………………………………………………………………………
   * Scope…………………………………………………………………………………………………………………………
   * Methodology………………………………………………………………………………………………………………
2. **Chapter Two Context and Review**…………………………………………………………………………………………14
   1. Historical………………………………………………………………………………………………………………
   2. Background……………………………………………………………………………………………………………………
      1. Early Web Security………………………………………………………………………………………………………………
      2. Rise of Web Applications……………………………………………………………………………………
      3. Modern Web Security…………………………………………………………………………………………
   3. Types of Cyber Attacks…………………………………………………………………………………………………
      1. SQL Injection………………………………………………………………………………………………………
      2. Cross Site Scripting (XSS) ……………………………………………………………………………………
      3. File Inclusion……………………………………………………………………………………………………….
      4. File Upload………………………………………………………………………………………………………….
   4. Related Works………………………………………………………………………………………………………………
3. **Chapter Three Development Approach………………………………………………………………………………21**
   1. Application Design………………………………………………………………………………………………………….
   2. Application Usage Guide…………………………………………………………………………………………………
   3. Database Structure……………………………………………………………………………………………………….
   4. Web Attacks…………………………………………………………………………………………………………….
      1. SQL Injection (SQLi) ……………………………………………………………………………………………
      2. Cross-Site Scripting (XSS) ……………………………………………………………………………………
      3. File Inclusion………………………………………………………………………………………………………
      4. File Upload…………………………………………………………………………………………………………
4. **Chapter Four: Architectural Design……………………………………………………………………………………….37**
   1. Major Modules………………………………………………………………………………………………………………
   2. Detailed Component Description……………………………………………………………………………………
5. **Chapter Five Implementation and Testing**………………………………………………………………………………39
   1. SQL Injection………………………………………………………………………………………………………………
   2. Cross Site Scripting (XSS) …………………………………………………………………………………………………
   3. File Inclusion………………………………………………………………………………………………………………
   4. File Upload………………………………………………………………………………………………………………
6. **Chapter Six: Conclusion**…………………………………………………………………………………………………………49
   1. Future Work………………………………………………………………………………………………………………
   2. Conclusion………………………………………………………………………………………………………………

**Chapter Seven: References**………………………………………………………………………………………………………50

### CHAPTER ONE: INTRODUCTION

#### Overview

In today’s digital age, web security has become a critical concern for businesses and individuals alike. The increasing frequency and sophistication of cyber-attacks pose significant risks to data integrity, privacy, and financial security. Understanding and mitigating these risks is essential for maintaining trust and ensuring the safe operation of web applications.

In this Documentation, we outline a project to create a Learning web application about web security vulnerabilities. The primary objective is to provide a realistic explanation and labs for learning and understanding web application security vulnerabilities and mitigation strategies.

The QRB Platform aims to address these concerns by providing a comprehensive and interactive learning environment focused on web security vulnerabilities. By simulating real-world attacks and offering hands-on experience, the platform empowers users to understand common web threats.

#### 1.2 Objectives

* Develop a platform to improve the understanding of web security vulnerabilities among developers, students, and professionals.
* Educate users about common web security vulnerabilities through detailed explanations and demonstrations.
* Offer comprehensive explanations and practical demonstrations of effective mitigation strategies for each vulnerability covered.
* Provide an interactive environment where users can practice identifying and mitigating these vulnerabilities.
* Design the platform to be accessible and engaging for users of all levels, utilizing tutorials and interactive content.
* Enhance users’ understanding of secure coding practices and security measures through continuous learning and assessment.

#### 1.3 Scope

The QRB Platform covers a range of web security vulnerabilities, including SQL Injection, Cross-Site Scripting (XSS), File Inclusion, and File Upload vulnerabilities. The platform is designed for students, developers, and cybersecurity professionals who wish to deepen their knowledge and skills in web security.

#### 1.4 Methodology

The development of the QRB Platform involved extensive research, design, implementation, and testing phases. The methodology included:

* Conducting a literature review to identify key vulnerabilities and educational approaches.
* Designing an intuitive and user-friendly interface.
* Testing the platform with real-world scenarios.

### CHAPTER TWO: CONTEXT AND REVIEW

#### 2. Background and Literature Review

##### 2.1 Historical Background

###### 2.1.1 Early Web Security­­­

Web security has evolved significantly since the early days of the internet. Initially, the focus was on securing physical networks and implementing basic security protocols. Early web applications were often developed with little regard for security, leading to numerous vulnerabilities that could be easily exploited by attackers.

###### 2.1.2 Rise of Web Applications

As web applications became more prevalent, the scope of security expanded to address more sophisticated threats specifically targeting web-based systems. The advent of e-commerce and online banking in the late 1990s and early 2000s highlighted the need for robust security measures, as financial data became a prime target for cybercriminals.

###### 2.1.3 Modern Web Security

Today, web security encompasses a wide range of practices and technologies aimed at protecting web applications from various threats. This includes secure coding practices, regular security assessments, penetration testing, and the adoption of frameworks and tools designed to identify and mitigate vulnerabilities early in the development process.

**Vulnerabilities**: These are weaknesses inherent in a system's design or implementation that can be exploited by attackers to execute malicious actions, gain unauthorized access to data, or perpetrate various forms of denial-of-service attacks. These vulnerabilities encompass a wide range of techniques, including SQL injection, cross-site scripting, and others. The primary objective behind exploiting these vulnerabilities is to achieve unauthorized access, exfiltrate sensitive information, disrupt services, or inflict other forms of harm. The repercussions of cyber vulnerabilities extend far beyond mere financial losses. Organizations often face severe reputational damage, loss of customer trust, and potential legal ramifications following a successful breach. Moreover, critical infrastructure systems such as power grids, healthcare facilities, and communication networks are also susceptible, posing significant risks to public safety and national security. Therefore, addressing and mitigating vulnerabilities are paramount to safeguarding both digital assets and societal well-being.

**Threat**: A threat encompasses any conceivable event, whether malicious or not, that has the potential to jeopardize an asset. In simpler terms, a threat refers to any adverse occurrence that could impact or harm one's possessions.

**Attack**: An attack refers to a deliberate and malicious action taken by an adversary with the intent to exploit vulnerabilities, compromise security measures, or cause harm to a system, network, or organization. Attacks can manifest in various forms, such as unauthorized access, data breaches, malware infections, denial-of-service (DoS) disruptions, or social engineering manipulations. The primary objective of an attack is typically to achieve some form of unauthorized gain, whether it be access to sensitive information, financial theft, or disruption of services.

##### 2.1.4 Types of Cyber Attacks

###### 2.1.4.1 Cross Site Scripting (XSS)

XSS involves injecting malicious scripts into content from otherwise trusted websites. This can compromise the interaction between users and the site, leading to stolen session tokens, user credentials, or the spread of malware. XSS attacks can be classified into stored, reflected, and DOM-based, each with unique methods and impacts.

**Example:** In 2014, eBay experienced a significant XSS vulnerability that allowed attackers to inject malicious code into listings, potentially stealing user information.

**Prevention:** Validate and sanitize all user inputs and use Content Security Policy (CSP) to restrict the execution of scripts.

A person sitting at a computer

Description automatically generated

*Figure 1 XSS*

###### 2.1.4.2 SQL Injection

SQL Injection (SQLi) is a common attack vector where malicious SQL statements are inserted into an entry field for execution. This can result in unauthorized access to database information, data manipulation, and even complete database compromise. It remains one of the most critical web security risks, with numerous high-profile breaches attributed to SQLi vulnerabilities.

**Example:** In 2008, Heartland Payment Systems suffered a massive data breach due to an SQL Injection attack, compromising over 130 million credit card numbers.

**Prevention:** Use prepared statements and parameterized queries to ensure that SQL code is not executed directly from user input.

A computer screen shot of a computer

Description automatically generated­­

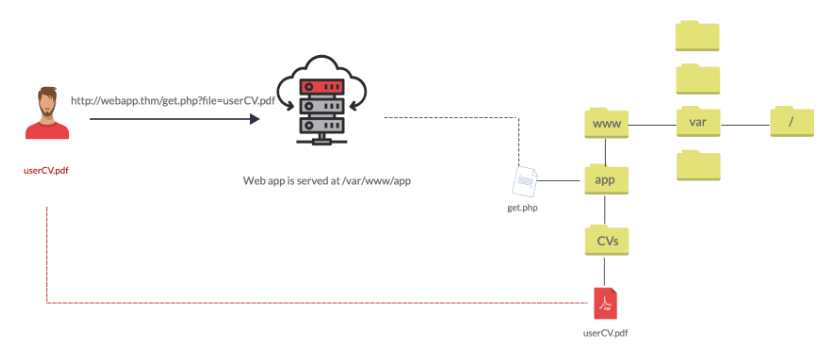
*Figure 2: SQL Injection Figure*

###### 2.1.4.3 File Inclusion

File inclusion vulnerabilities arise when a web application allows external files to be included or executed on the server. This can lead to remote code execution, data leakage, and full system compromise. There are two main types of file inclusion attacks: Local File Inclusion (LFI) and Remote File Inclusion (RFI).

**Example:** In 2011, a vulnerability in the Joomla! content management system allowed remote attackers to include arbitrary files, leading to potential remote code execution.

**Prevention:** Restrict file paths to known and trusted directories and validate all file inputs.



*Figure 3: File inclusion*

###### 2.1.4.4 File Upload

File upload vulnerabilities occur when an application fails to validate or sanitize user-uploaded files, potentially allowing malicious files to be uploaded and executed. These vulnerabilities can be exploited to upload web shells, modify server configurations, or distribute malware.

**Example:** In 2020, a file upload vulnerability in the WordPress plugin File Manager allowed attackers to upload malicious files, leading to widespread exploitation.

**Prevention:** Implement strict file validation and sanitization and restrict executable file types.

A diagram of a computer

Description automatically generated

*Figure 4: File Upload*

##### 2.2 Related Works

Previous studies and platforms have explored various aspects of web security education. Notable among these are projects focusing on simulated environments to teach SQL Injection (SQLi), Cross Site Scripting (XSS), File Inclusion, and File Upload vulnerabilities. These works provide a foundation upon which the QRB Platform builds, aiming to offer an integrated and interactive learning experience.

#### 2.2.1Existing Platforms

**OWASP Web Goat**: An intentionally insecure web application designed to teach web application security lessons. Users can practice exploiting vulnerabilities in a controlled environment.

**DVWA (Damn Vulnerable Web Application)**: A PHP/MySQL web application that is vulnerable by design, providing a practice environment for security enthusiasts to test their skills.

#### 2.2.2Comparison with QRB Platform

Unlike these platforms, the QRB Platform integrates a structured curriculum with interactive tutorials, and progress tracking, offering a more comprehensive learning experience.

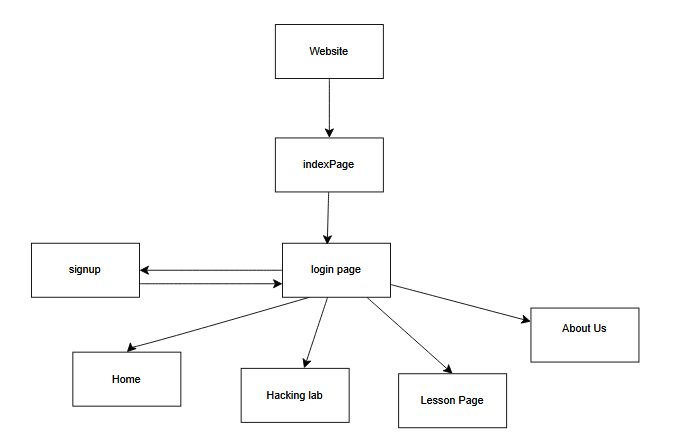
The QRB Platform includes dedicated labs for the following four attack types, each providing hands-on experience and prevention strategies:

1. **Cross Site Scripting (XSS)**
   * **Description**: XSS involves injecting malicious scripts into content from otherwise trusted websites. This can compromise the interaction between users and the site, leading to stolen session tokens, user credentials, or the spread of malware. XSS attacks can be classified into stored, reflected, each with unique methods and impacts.
   * **Example**: In 2014, eBay experienced a significant XSS vulnerability that allowed attackers to inject malicious code into listings, potentially stealing user information.
   * **Prevention**: Validate and sanitize all user inputs and use Content Security Policy (CSP) to restrict the execution of scripts.
2. **SQL Injection (SQLi)**
   * **Description**: SQL Injection is a common attack vector where malicious SQL statements are inserted into an entry field for execution. This can result in unauthorized access to database information, data manipulation, and even complete database compromise. It remains one of the most critical web security risks, with numerous high-profile breaches attributed to SQLi vulnerabilities.
   * **Example**: In 2008, Heartland Payment Systems suffered a massive data breach due to an SQL Injection attack, compromising over 130 million credit card numbers.
   * **Prevention**: Use prepared statements and parameterized queries to ensure that SQL code is not executed directly from user input.
3. **File Inclusion**
   * **Description**: File inclusion vulnerabilities arise when a web application allows external files to be included or executed on the server. This can lead to remote code execution, data leakage, and full system compromise. There are two main types of file inclusion attacks: Local File Inclusion (LFI) and Remote File Inclusion (RFI).
   * **Example**: In 2011, a vulnerability in the Joomla! content management system allowed remote attackers to include arbitrary files, leading to potential remote code execution.
   * **Prevention**: Restrict file paths to known and trusted directories, and validate all file inputs.
4. **File Upload**
   * **Description**: File upload vulnerabilities occur when an application fails to validate or sanitize user-uploaded files, potentially allowing malicious files to be uploaded and executed. These vulnerabilities can be exploited to upload web shells, modify server configurations, or distribute malware.
   * **Example**: In 2020, a file upload vulnerability in the WordPress plugin File Manager allowed attackers to upload malicious files, leading to widespread exploitation.
   * **Prevention**: Implement strict file validation and sanitization and restrict executable file types.

### CHAPTER THREE: DEVELOPMENT APPROACH

#### 3.1 Application Design

The QRB Platform is designed with user experience in mind. The interface is intuitive, ensuring users can easily navigate through different modules and understand the content without unnecessary complexity. The design process involved creating wireframes and prototypes, followed by user testing to refine the layout and functionality.



*Figure 5:* Application Design

**Design Principles:**

* **Simplicity:** The interface is kept clean and straightforward, allowing users to focus on the learning content.
* **Accessibility:** The platform adheres to accessibility standards, ensuring it is usable by people with disabilities.
* **Responsiveness:** The layout adapts seamlessly to different devices and screen sizes, providing a consistent user experience.

**User Interface:**

* **Tutorial Page:** Lists all available modules with brief descriptions and progress indicators.
* **labs Page:** Contains detailed Hints, interactive labs, for each vulnerability.

#### 3.2 Application Usage Guide

The usage guide provides a step-by-step walkthrough of the platform. Users begin by registering on the platform, followed by exploring the various educational modules dedicated to different web security vulnerabilities. Each module includes interactive tutorials, theoretical explanations, and practical exercises.

**Registration and Login:**

* **Account Creation:** Users create an account with basic information and secure authentication mechanisms.

A screenshot of a phone

Description automatically generated

Figure 6 **Account Creation**

* **Login Process:** Users log in using their credentials.

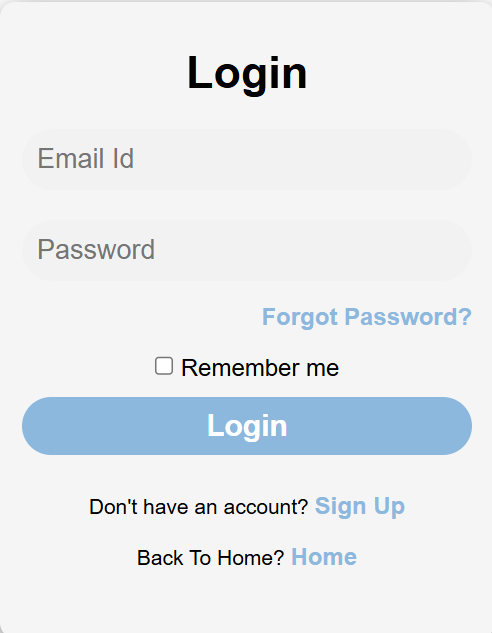


Figure 7 **login process**

**Navigating Modules:**

* **Module Selection:** Users can select a module to view detailed content and start interactive labs.

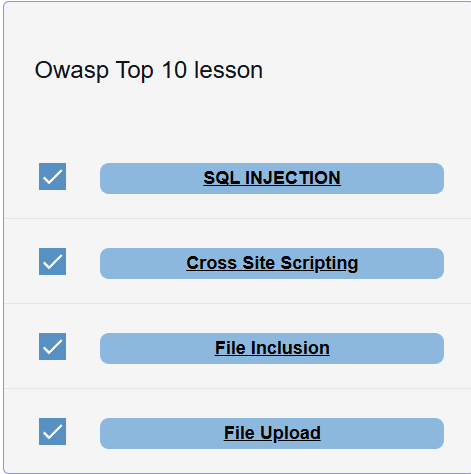


Figure 8: **Module Selection**

A screenshot of a computer security system

Description automatically generated

Figure 8.1: **Module Selection**

* **Interactive Labs:** Hands-on labs allow users to exploit vulnerabilities in a controlled environment, reinforcing theoretical knowledge with practical skills.

A screenshot of a computer program

Description automatically generated

Figure9: **Interactive Labs**

#### 3.3 Database Structure

The platform’s backend is powered by a robust database structure designed to handle user data, module content, and interaction logs securely. The database schema includes tables for users, modules, progress tracking, and logs.

**Database Schema:**

* **Users Table:** Stores user information, including hashed passwords and authentication tokens.
* **Modules Table:** Contains details about each educational module, including content and associated vulnerabilities.
* **Progress Table:** Tracks user progress through modules, recording completion status and quiz scores.
* **Logs Table:** Maintains logs of user interactions, ensuring auditability and accountability.

**Database Technologies:**

* **MySQL:** Chosen for its reliability and support for complex queries.
* **ORM (Object-Relational Mapping):** Utilized to simplify database interactions and improve maintainability.

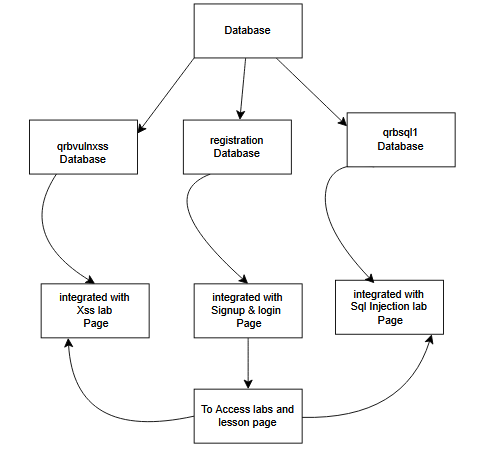


Figure10: database Figure

#### 3.4 Web Attacks

##### 3.4.1 SQL Injection (SQLi)

###### 3.4.1.1 How SQL Injection Works

SQL Injection involves inserting malicious SQL code into a query, manipulating the database to execute unintended commands. Attackers can extract, modify, or delete data by exploiting improperly sanitized input fields.

**Example Code:**



Malicious input

A close up of words

Description automatically generated

**Figure 11:** Malicious input

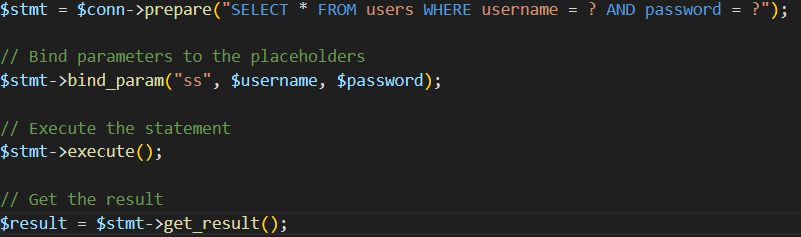
**Resulting Query:**

SQL

SELECT \* FROM users WHERE username = '' OR '1'='1' AND password = '' OR '1'='1';

**Prevention Techniques:**

* **Prepared Statements:** Ensure SQL code and data are kept separate
* **Parameterized Queries:** Use parameters instead of directly embedding user inputs into SQL queries.



**Figure 12: SQL Prevention Techniques**

##### 3.4.2 Cross-Site Scripting (XSS)

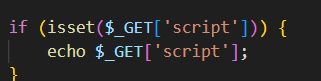
###### 3.4.2.1 How XSS (Reflected) Attack Works

Reflected XSS occurs when malicious scripts are injected into the input fields of a web application, which are then reflected back to the user. This can happen when input from a user is included in the web page’s output without proper validation.

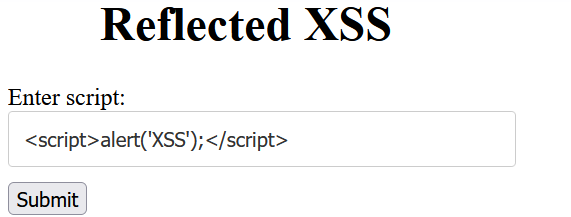
**Example:**

html

-- Vulnerable code



**Figure13: XSS Vulnerable code:**



**Figure14: XSS Malicious Input:**

**Resulting Page:**

**A close up of a blue surface

Description automatically generated**

**Figure 15: XSS Reflected result**

**Prevention Techniques:**

* **Input Validation:** Check and sanitize user inputs.
* **Output Encoding:** Encode data before rendering it in the browser.

###### 3.4.2.2 How XSS (Stored) Attack Works

Stored XSS involves injecting malicious scripts into a web application’s stored data, which is then executed when other users access the affected content.

**Example:**

**A screenshot of a computer

Description automatically generated**

Figure 16: XSS Stored section 34

**Malicious Comment:**

A screenshot of a computer

Description automatically generated

Figure:17 XSS Stored **Malicious Comment:**

**Resulting Page:**

**A close up of a blue surface

Description automatically generated**

**Figure 18: XSS Stored Resulting:**

**Prevention Techniques:**

* **Sanitization:** Remove or encode harmful characters from user inputs.
* **Content Security Policy (CSP):** Use CSP to restrict script execution.

##### 3.4.3 File Inclusion

###### 3.4.3.1 How File Inclusion Works

File inclusion vulnerabilities allow an attacker to include files, which can lead to remote code execution or data exposure. Local File Inclusion (LFI) involves including files from the local server, while Remote File Inclusion (RFI) involves including files from a remote server.

**Example (LFI):**

PHP

-- Vulnerable code

include($\_GET['page'] . '.PHP);

**Malicious Input:**

PHP

?page=../../../../etc/passwd

**Resulting Inclusion:**

php

include('../../../../etc/passwd.php');

**Prevention Techniques:**

* **Whitelisting:** Allow only specific, known files to be included.
* **Input Validation:** Restrict user inputs to prevent directory traversal.

##### 3.4.4 File Upload

###### 3.4.4.1 How File Upload Works

File upload vulnerabilities occur when an application does not properly validate or sanitize uploaded files. This can lead to the execution of malicious scripts or the upload of unwanted content.

**Example:**

php

-- Vulnerable code

move\_uploaded\_file($\_FILES['file']['tmp\_name'], 'uploads/' . $\_FILES['file']['name']);

**Malicious File:**

php

file.php

**Resulting File Upload:**

php

uploads/file.php

**Prevention Techniques:**

* **File Validation:** Check file types and contents.
* **Restrict File Types:** Allow only specific file types (e.g., images, documents).
* **Sanitize File Names:** Remove harmful characters from file names.

### CHAPTER FOUR: ARCHITECTURAL DESIGN

#### 4.1 Major Modules

The QRB Platform is structured into several key modules, each responsible for a specific aspect of the platform’s functionality. These modules work together to provide a seamless user experience and ensure comprehensive coverage of web security topics.

**Modules:**

* **User Management:** Handles user registration, authentication.
* **Content Management:** Manages educational content, including tutorials, and interactive labs.
* **Interactive Labs:** Provides a sandbox environment for practicing web security techniques.

#### 4.2 Detailed Component Description

Each module comprises several components that interact to deliver the desired functionality. Detailed descriptions of these components include their responsibilities, interactions, and security considerations.

**User Management:**

* **Registration:** Handles user sign-ups and account creation.
* **Authentication:** Manages login, logout, and session management.
* **Delivery:** Ensures content is delivered to users in an accessible and organized manner.

**Interactive Labs:**

* **Sandbox Environment:** Provides a controlled environment for practicing web security techniques.
* **Lab Scenarios:** Offers pre-configured scenarios to simulate real-world attacks and defenses.
* **Real-Time Feedback:** Provides instant feedback on user actions within the labs.

**Reporting and Analytics:**

* **Usage Analytics:** Analyzes platform usage to identify trends and areas for improvement.
* **Security Audits:** Conducts regular security audits to ensure the platform remains secure.

### CHAPTER FIVE: IMPLEMENTATION AND TESTING

#### 5.1 SQL Injection

The SQL Injection module was tested using **Manual Testing** and methodologies to ensure it effectively simulates real-world vulnerabilities and teaches users how to mitigate them.

**Testing Tools:**

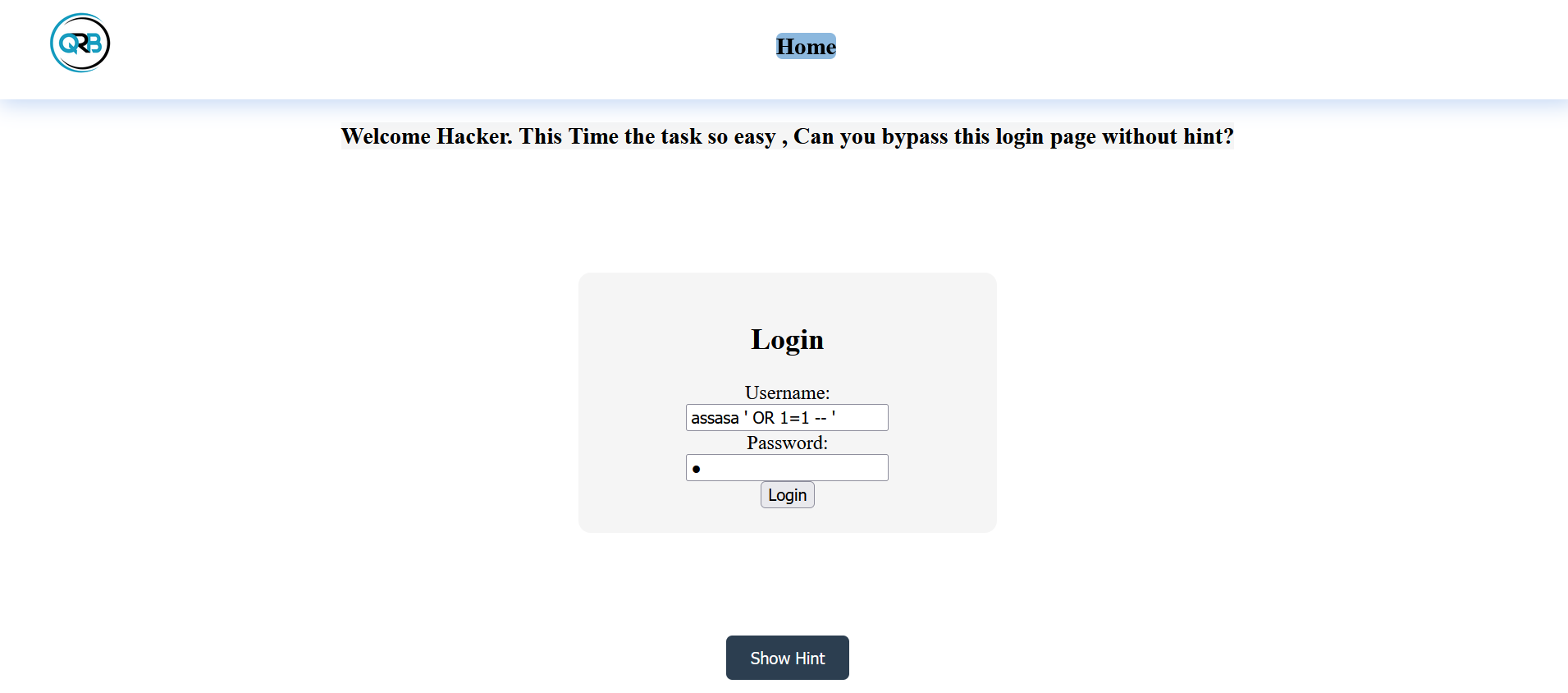
* **Manual Testing:** Conducted by security experts to identify and exploit SQL Injection vulnerabilities.

**Testing Scenarios:**

* **Basic SQL Injection:** Users learn to identify and exploit basic SQL Injection vulnerabilities.
* **Advanced SQL Injection:** Users practice exploiting more complex SQL Injection scenarios, such as blind SQL Injection.

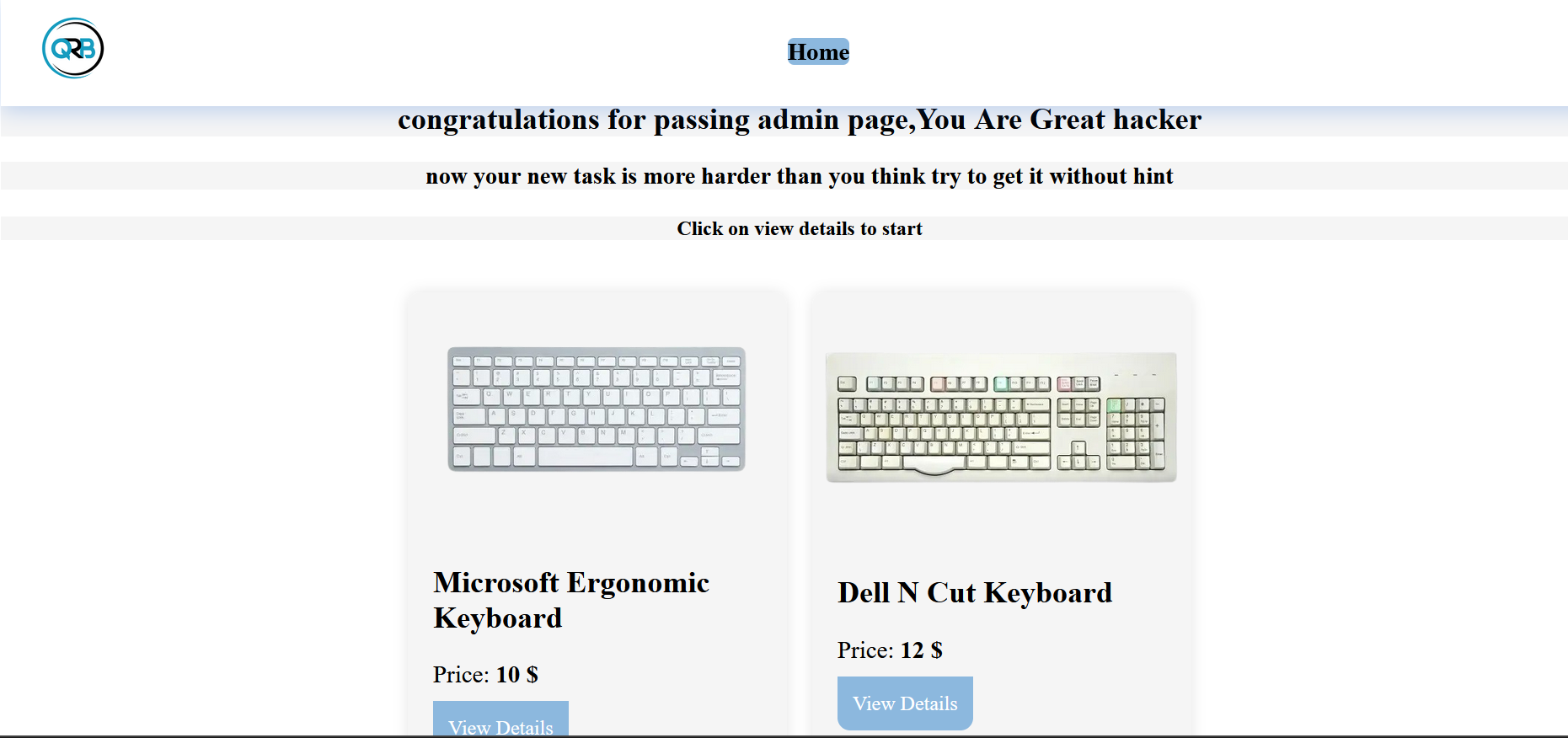
**Results:**

* **Effectiveness:** Users demonstrated a significant improvement in their ability to identify and mitigate SQL Injection vulnerabilities.
* **Feedback:** User feedback was overwhelmingly positive, highlighting the practical nature of the tutorials and labs.

****

**Figure 19: SQL Injection Lab Interface**

* **SQL Injection Lab Interface:** Screenshot showing the SQL Injection lab interface, with examples of user input and resulting queries.

****

**Figure 20: SQL Results:**

* **SQL Results:** Screenshot of SQL output, demonstrating the identification and exploitation of SQL Injection vulnerabilities.

#### 5.2 Cross Site Scripting (XSS)

The XSS module was thoroughly tested to ensure it provides comprehensive coverage of both reflected and stored XSS vulnerabilities.

**Testing Tools:**

* **Manual Testing:** Conducted by security experts to ensure thorough coverage of XSS scenarios.

**Testing Scenarios:**

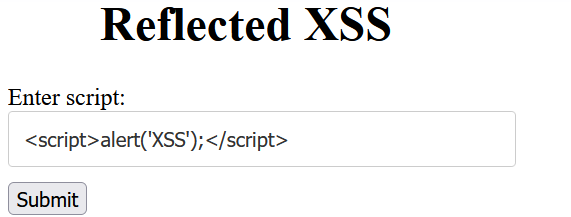
* **Reflected XSS:** Users practice identifying and exploiting reflected XSS vulnerabilities in various input fields.
* **Stored XSS:** Users learn to identify and mitigate stored XSS vulnerabilities in user-generated content.

**Results:**

* **Effectiveness:** Users reported a significant increase in their understanding of XSS vulnerabilities and mitigation techniques.
* **Feedback:** Users appreciated the interactive nature of the labs and the detailed explanations provided.

**Screenshots:**

* **XSS Lab Interface:** Screenshot showing the XSS lab interface, with examples of user input and resulting page content.



**Figure21:** Reflected **XSS Lab Interface**

* **A close up of a blue surface

  Description automatically generated**



**Figure22 Stored XSS Lab Interface**

A close up of a blue surface

Description automatically generated

#### 5.3 File Inclusion

The File Inclusion module was tested to ensure it accurately simulates Local File Inclusion (LFI) vulnerabilities.

**Testing Tools:**

* **Manual Testing:** Conducted by security experts to exploit file inclusion vulnerabilities.

**Testing Scenarios:**

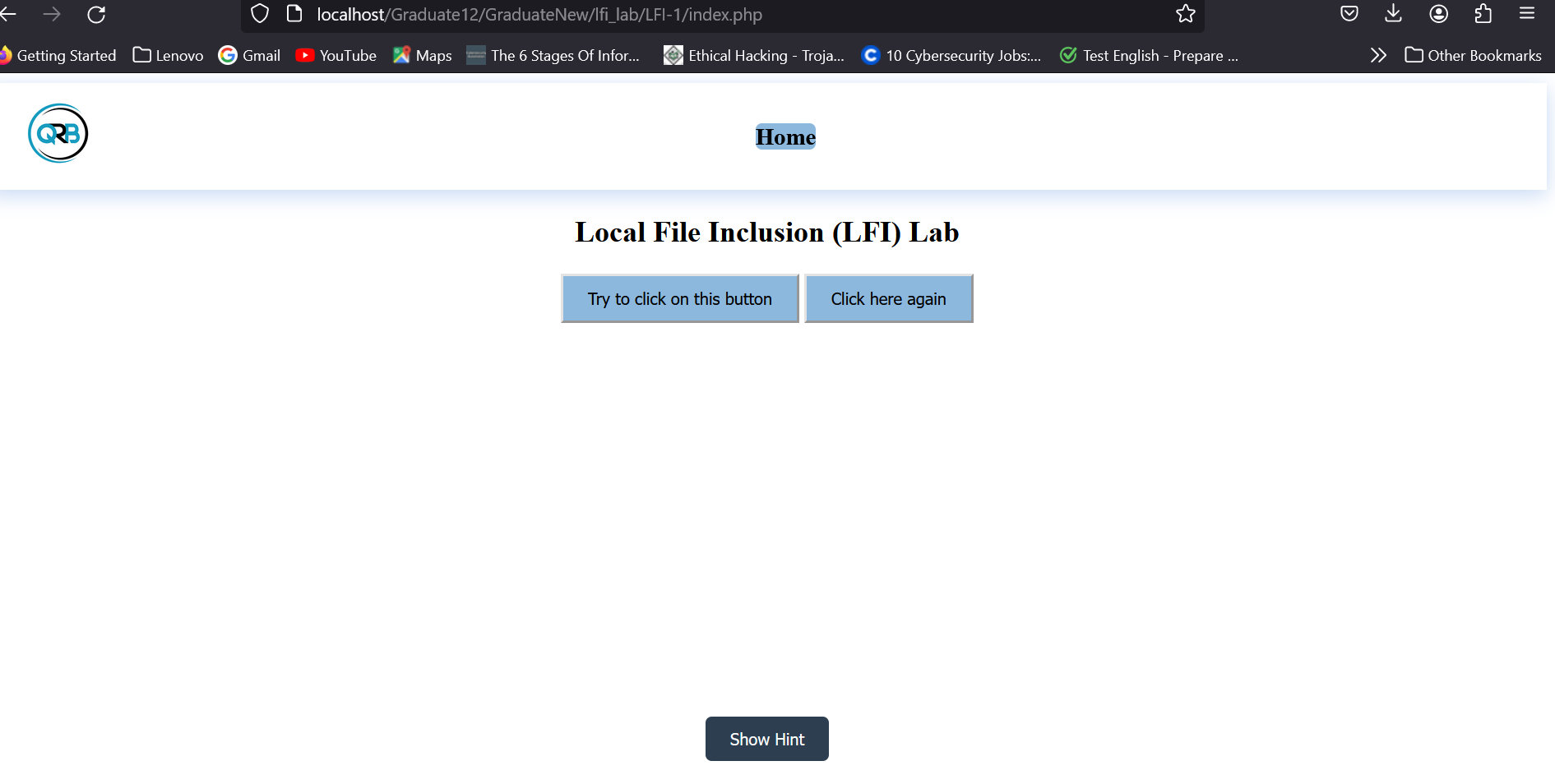
* **Local File Inclusion (LFI):** Users practice identifying and exploiting LFI vulnerabilities to read sensitive files.

**Results:**

* **Effectiveness:** Users demonstrated a marked improvement in their ability to identify and prevent file inclusion vulnerabilities.
* **Feedback:** Users found the labs to be highly informative and practical, with real-world relevance.

**Screenshots:**

* **File Inclusion Lab Interface:** Screenshot showing the file inclusion lab interface, with examples of user input and resulting file inclusions.



**Figure 23: File Inclusion Lab Interface**

A screenshot of a computer

Description automatically generated

Figure 24: File Inclusion Lab test

A screenshot of a computer

Description automatically generated

Figure 25: File Inclusion Lab query

* **Acunetix Results:** Screenshot of Acunetix output, demonstrating the identification and exploitation of file inclusion vulnerabilities.

A screenshot of a computer

Description automatically generated

Figure 26: File Inclusion Lab result

#### 5.4 File Upload

The File Upload module was tested to ensure it effectively teaches users to identify and mitigate file upload vulnerabilities.

**Testing Tools:**

* **Manual Testing:** Conducted by security experts to exploit file upload vulnerabilities.

**Testing Scenarios:**

* **Basic File Upload:** Users learn to identify and exploit basic file upload vulnerabilities.

**Results:**

* **Effectiveness:** Users reported a significant increase in their ability to identify and mitigate file upload vulnerabilities.
* **Feedback:** Users appreciated the detailed explanations and practical examples provided in the labs.

**Screenshots:**

* **File Upload Lab Interface:** Screenshot showing the file upload lab interface.

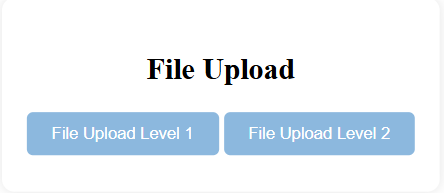


Figure 27: File Upload Lab interface

* **File Upload** Level 1**:** Screenshot showing the file upload lab interface, with examples of user input and resulting file uploads.

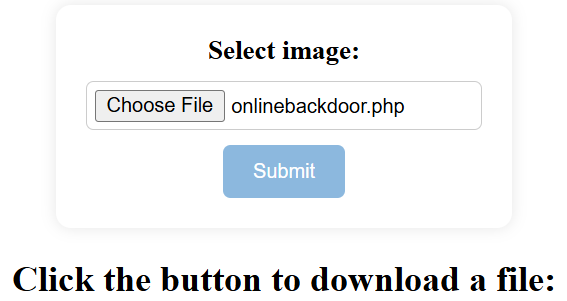


Figure 28: File Upload Lab Test

Result

A close up of a sign

Description automatically generated

Figure 29: File Upload Lab result

A black background with white text

Description automatically generated

Figure 29.1: File Upload Lab Result

* **File Upload** Level 2**:** Screenshot showing the file upload lab interface, with examples of user input and resulting file uploads.

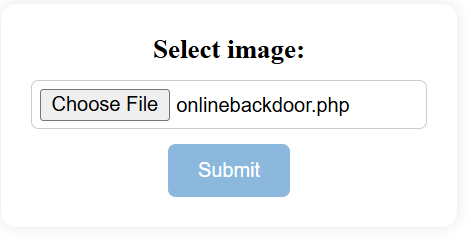


Figure 30: File Upload Lab Leve 2 Interface

Result

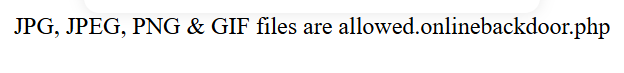


Figure 31: File Upload Lab Leve 2 Result

### CHAPTER SIX: CONCLUSION

#### 6.1 Future Work

The QRB Platform has laid a solid foundation for web security education, but there is always room for improvement and expansion. Future work will focus on:

* **Expanding the Curriculum:** Adding more modules covering additional web security topics such as CSRF, clickjacking, and session management.
* **Advanced Labs:** Developing more advanced lab scenarios to challenge users and enhance their skills.
* **User Feedback Integration:** Continuously improving the platform based on user feedback and emerging web security trends.
* **Community Engagement:** Building a community of learners and professionals to share knowledge and collaborate on web security challenges.

#### 6.2 Conclusion

The QRB Platform represents a significant step forward in web security education, providing users with an interactive and comprehensive learning environment. By simulating real-world vulnerabilities and offering hands-on experience, the platform empowers users to understand and defend against common web threats. The positive feedback and demonstrated effectiveness of the platform underscore its value as a vital tool in the fight against cybercrime.

### CHAPTER SEVEN: REFERENCES

1. OWASP. (2021). OWASP Top Ten Project. https://owasp.org/www-project-top-ten/
2. Acunetix. (2021). What is SQL Injection? https://www.acunetix.com/websitesecurity/sql-injection/
3. PortSwigger. (2021). Cross-Site Scripting (XSS). https://portswigger.net/web-security/cross-site-scripting
4. CWE. (2021). CWE-98: Improper Control of Filename for Include/Require Statement in PHP Program ('PHP Remote File Inclusion'). https://cwe.mitre.org/data/definitions/98.html
5. OWASP ZAP. (2021). https://owasp.org/www-project-zap/