A

Project Report

on

Vulnerability Assessment and Penetration Testing (VAPT) on a Web Application

Submitted in fulfilment of the requirements

for the award of the

IBM's PBEL Certificate

in

Cybersecurity

by

Anuj Kumar Arya

from

Bachelors of Technology

in

Computer Science and Engineering (AIML)

Galgotias College of Engineering and Technology,

Greater Noida, Uttar Pradesh

India-201310

Under the Supervision of

Mr. Rushikesh Dinkar

July, 2025

ABSTRACT

This project focuses on conducting a **Vulnerability Assessment and Penetration Testing (VAPT) on a web application** using Burp Suite, a leading tool in the field of web application security. The main objective was to identify potential vulnerabilities that could be exploited by attackers and assess the overall security posture of the target application.

The methodology followed a standard penetration testing lifecycle, beginning with reconnaissance and scanning, followed by enumeration, exploitation, and finally reporting. Using Burp Suite's suite of tools—such as **the Proxy, Repeater, Intruder**, and Scanner—we conducted both manual and automated testing of the application. Special attention was given to common vulnerabilities listed in the **OWASP** Top 10 such **as Cross-Site Scripting (XSS), SQL Injection, Broken Authentication, and CSRF.**

Key findings include the detection of medium to high severity vulnerabilities, some of which could allow unauthorized access or data leakage. Screenshots and technical details have been included to support the findings. The project also addresses some challenges faced during the testing process, including false positives and encrypted sessions.

This report concludes with mitigation recommendations and future suggestions for improving the security of the web application.

CONTENTS

LIST OF TABLES

Table Title

LIST OF FIGURES

Figure Title

ABBREVIATIONS

Abbreviation	Full Form
VAPT	Vulnerability Assessment and Penetration Testing
OWASP	Open Web Application Security Project
PoC	Proof of Concept
XSS	Cross-Site Scripting
SQLi	Structured Query Language Injection
CSRF	Cross-Site Request Forgery
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
DVWA	Damn Vulnerable Web Application
IP	Internet Protocol
URL	Uniform Resource Locator
GUI	Graphical User Interface
API	Application Programming Interface
MITM	Man-in-the-Middle
CVE	Common Vulnerabilities and Exposures
WAF	Web Application Firewall
PII	Personally Identifiable Information

CHAPTER 1

INTRODUCTION

In the era of digital transformation, web applications serve as the backbone of countless online services—ranging from e-commerce platforms and banking portals to government services and healthcare systems. These applications handle vast amounts of sensitive data, including personally identifiable information (PII), authentication credentials, and financial records. However, with this increased reliance on web technologies comes a significantly expanded **attack surface** for malicious actors to exploit.

Web applications often suffer from security misconfigurations, weak input validation, outdated libraries, and flawed authentication mechanisms. These issues can lead to critical vulnerabilities such as **SQL Injection**, **Cross-Site Scripting (XSS)**, **Cross-Site Request Forgery (CSRF)**, and **Insecure Direct Object References (IDOR)**. If left unaddressed, such vulnerabilities can result in unauthorized data access, session hijacking, denial of service, or even full system compromise.

Vulnerability Assessment and Penetration Testing (VAPT) is a systematic process used to detect and evaluate these security flaws. While **vulnerability assessment** focuses on identifying known issues and configuration problems, **penetration testing** goes one step further by simulating actual attacks to exploit discovered weaknesses and assess the impact. The combined approach gives organizations a comprehensive understanding of their security posture.

This project specifically utilizes **Burp Suite**, a powerful web security testing framework, to perform black-box testing of a web application. Burp Suite allows both automated and manual testing through its integrated modules such as:

- **Proxy**: Captures and modifies HTTP/HTTPS requests.
- **Spider**: Crawls the application to map all accessible endpoints.
- **Scanner**: Automatically tests for common vulnerabilities (Pro version).
- **Intruder**: Performs fuzzing and brute-force attacks.
- **Repeater**: Replays and manipulates requests for manual exploitation.
- **Decoder** and **Comparer**: Aid in analysing and comparing encoded data.

The purpose of this project is to **simulate real-world attack scenarios** in a controlled environment and provide a detailed analysis of discovered vulnerabilities, including risk ratings, proof-of-concept exploits, and actionable remediation recommendations. By doing so, the project demonstrates the importance of proactive security testing in modern web development and provides hands-on experience in offensive cybersecurity practices.

1. Problem Definition

As cyberattacks continue to rise, many web applications are deployed without proper security validation, making them vulnerable to threats like SQL Injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF). Such vulnerabilities can lead to data breaches, unauthorized access, or even full application compromise. The goal of this project is to identify these weaknesses before an attacker can exploit them.

2. Objectives

The primary goal of this project is to evaluate the security posture of a web application through comprehensive black-box testing. Specific objectives include:

1. **Identify security vulnerabilities** in the web application using both automated and manual techniques aligned with the OWASP Top 10 standards.
2. **Use Burp Suite's tools** (Proxy, Repeater, Intruder, Scanner, etc.) to intercept, analyse, and manipulate web traffic in real-time.
3. **Simulate realistic attack vectors** such as injection attacks, broken authentication, and session management flaws.
4. **Document findings** with technical details, impact analysis, and reproduction steps for each vulnerability.
5. **Recommend mitigations** based on industry best practices to improve the application's overall security posture.
6. **Demonstrate ethical hacking practices** in a safe and legal testing environment to gain hands-on experience in cybersecurity.

3. Scope

• The testing is restricted to the **application layer** (OSI Layer 7), with focus on HTTP and HTTPS traffic.

• All tests were conducted using **black-box testing methodology**, assuming no internal access to the source code or server-side logic.

• The assessment targeted **common web vulnerabilities**, such as:

- Input validation flaws (e.g., SQL Injection, XSS)
- Session management issues
- Broken access controls
- Misconfigured security headers

• A vulnerable test environment was used (e.g., **DVWA**, **bWAPP**, or a custom web app) to simulate real-world scenarios without legal risk.

4. Limitations

- Burp Suite Community Edition lacks the automated vulnerability scanner, which is only available in the Professional version.
- Testing was limited to the frontend-exposed functionality and did not include API or mobile application endpoints.
- Denial-of-Service (DoS) and destructive tests were avoided to preserve application availability.
- Third-party services or cloud infrastructure associated with the web app were not assessed.

5. Motivations

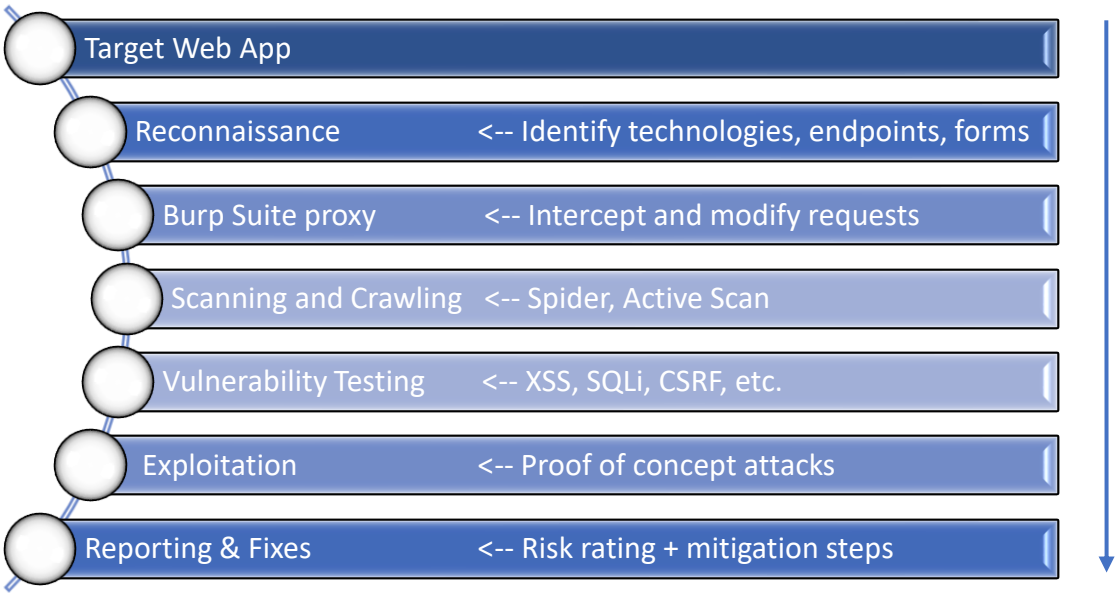
The motivation behind this project is to understand and apply ethical hacking practices to proactively discover and remediate vulnerabilities in a web application. By conducting VAPT, organizations can avoid potential financial and reputational damage and stay compliant with security standards such as OWASP Top 10, PCI-DSS, and GDPR.

6. Tools and Technologies

- **Burp Suite Community Edition** – for intercepting, scanning, and exploiting vulnerabilities.
- **OWASP Top 10 Guidelines** – for aligning the testing with known security risks.
- **Web Browser with Burp Proxy Configuration** – to simulate normal user interaction.
- **Test Environment / DVWA (Damn Vulnerable Web App)** – optionally used as a vulnerable target platform for testing purposes.

7. Flowchart: VAPT Lifecycle Using Burp Suite

Below is a simplified flowchart that describes the major steps performed during this project:


Target Web App
Reconnaissance <-- Identify technologies, endpoints, forms
Burp Suite proxy <-- Intercept and modify requests
Scanning and Crawling <-- Spider, Active Scan
Vulnerability Testing <-- XSS, SQLi, CSRF, etc.
Exploitation <-- Proof of concept attacks
Reporting & Fixes <-- Risk rating + mitigation steps

CHAPTER 2

LITERATURE REVIEW

1. Introduction to Web Application Security

Over the last decade, the rise in web-based services has led to increased attention on web application security. According to the Verizon Data Breach Investigations Report, web application attacks are among the top causes of confirmed data breaches. This highlights the critical need for regular security audits of public-facing applications.

2. OWASP Top 10 Framework

The OWASP (Open Web Application Security Project) provides a widely accepted framework for assessing and mitigating common security risks in web applications. The OWASP Top 10 categorizes the most critical security vulnerabilities, including:

- A1: Broken Access Control
- A2: Cryptographic Failures
- A3: Injection (e.g., SQL, NoSQL, Command)
- A7: Identification and Authentication Failures
- A8: Software and Data Integrity Failures

This project aligns with OWASP's methodology, as it forms the foundation for most VAPT procedures.

3. Penetration Testing Methodologies

Several VAPT methodologies are recognized in the cybersecurity community:

- PTES (Penetration Testing Execution Standard)
- NIST SP 800-115 (Technical Guide to Information Security Testing)
- OWASP Web Security Testing Guide (WSTG)

These methodologies outline phases such as information gathering, threat modelling, vulnerability analysis, exploitation, and post-exploitation, which are reflected in this project's structure.

4. Burp Suite in Research and Practice

Burp Suite, developed by PortSwigger, is widely recognized as a premier tool for web application security testing. Its modular design (Proxy, Repeater, Intruder, Scanner, etc.) allows penetration testers to:

- Intercept and manipulate live HTTP/S traffic
- Perform automated and semi-automated scans
- Conduct fuzzing attacks and authentication bypass testing

In a study by S. A. Akinyelu and A. O. Adewumi (2021), Burp Suite was evaluated against tools like OWASP ZAP and Nikto. The findings showed that Burp Suite provided higher flexibility for manual exploitation and better request inspection tools for discovering advanced logic flaws.

5. Deliberately Vulnerable Web Applications (DVWAs)

Applications like DVWA (Damn Vulnerable Web App), bWAPP, and Mutillidae are used in academia and training labs to simulate real-world vulnerabilities. These applications allow testers to:

- Practice injection and XSS attacks in a safe environment
- Learn how broken access control manifests in insecure code
- Understand how Burp Suite can be used to construct Proof-of-Concept (PoC) exploits

6. Importance of Ethical Hacking in Academia

According to a paper published in the International Journal of Cyber-Security and Digital Forensics (IJCSDF), integrating ethical hacking into computer science and information technology curricula significantly enhances practical security understanding among students. VAPT projects like this help bridge the gap between theoretical learning and real-world application.

Conclusion

This literature review has explored the foundational theories, frameworks, and tools supporting the project. The relevance of OWASP guidelines, industry methodologies, and the capabilities of Burp Suite reinforce the project's approach and help validate the tools and techniques used.

CHAPTER 3

METHODOLOGY

1. Testing Approach

This project follows a structured **Black-Box Penetration Testing** methodology. In this approach, the tester does not have access to the internal source code or architecture of the web application. The entire assessment is conducted from an external attacker's perspective, using only publicly accessible interfaces and user-level accounts.

The testing was guided by:

- **OWASP Web Security Testing Guide (WSTG)**
- **OWASP Top 10 Vulnerabilities**
- **Burp Suite's best practices**

2. Lab Setup

Component	Description
Host Machine	Windows 11 / Kali Linux (local machine)
Testing Tools	Burp Suite Community Edition, Firefox (with Burp Proxy), DVWA (localhost)
Target Application	DVWA (Damn Vulnerable Web Application) running on XAMPP (localhost)
Network	Localhost (127.0.0.1) with proxy enabled
Browser Config	Manual proxy set to 127.0.0.1:8080 in Firefox

3. Tools Used and Configuration

Burp Suite Modules Used:

Module	Purpose
Proxy	Intercepts HTTP/HTTPS requests/responses between browser and server.
Spider	Automatically maps web app content and links.
Repeater	Resends and manipulates HTTP requests manually for deeper testing.
Intruder	Performs brute force, fuzzing, and parameter testing.

Module	Purpose
Decoder	Encodes/decodes Base64, URL, HTML, etc., to inspect payloads.
Comparer	Compares responses to identify behavioural differences or anomalies.

4. Step-by-Step Testing Process

Step 1: Target Configuration

- Launched DVWA on XAMPP at http://127.0.0.1/dvwa/.
- Configured Firefox to use Burp Suite's proxy (127.0.0.1:8080).
- Intercepted HTTP requests via Burp Proxy to validate connectivity.

Step 2: Mapping and Crawling

- Used **Spider** to map out all reachable endpoints (GET and POST).
- Identified input fields, login pages, and functionality modules.

Step 3: Vulnerability Identification

- Tested for common vulnerabilities manually and using Intruder:
  - **SQL Injection** on login, search, and feedback fields.
  - **XSS (Reflected & Stored)** on comment and profile update pages.
  - **CSRF** on password change forms by crafting forged HTML forms.
  - **Broken Authentication** by testing session tokens and cookies.

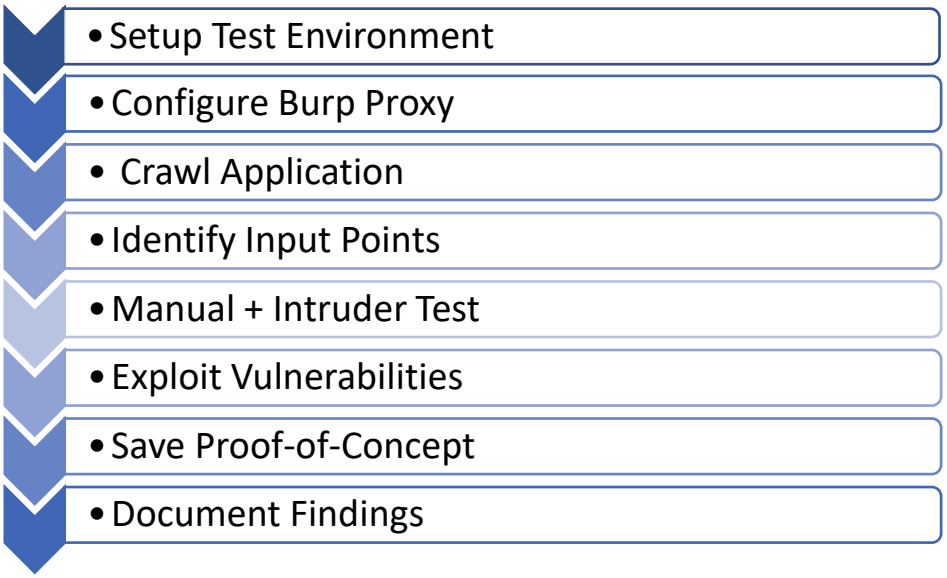
Step 4: Manual Exploitation

- Used **Repeater** to test parameter tampering and authentication bypass.
- Captured session tokens and attempted session hijacking.
- Injected XSS payloads like <script> alert('XSS') </script> and verified execution.

Step 5: PoC Documentation

- Captured screenshots of all findings.
- Logged payloads, response codes, behaviour, and risk rating for each vulnerability.
- Saved CSRF PoC forms for demonstration.

5. Testing Methodology Flowchart


• Setup Test Environment
• Configure Burp Proxy
• Crawl Application
• Identify Input Points
• Manual + Intruder Test
• Exploit Vulnerabilities
• Save Proof-of-Concept
• Document Findings

CHAPTER 4

RESULTS ANALYSIS

1. Summary of Vulnerabilities Found

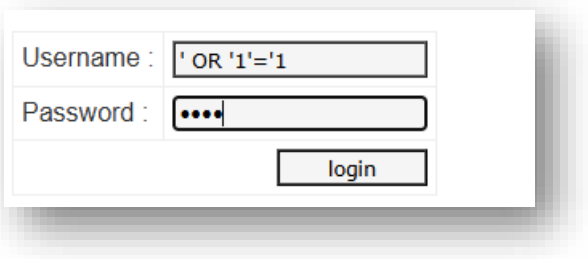
The table below summarizes the vulnerabilities discovered during testing, categorized by type and severity according to the OWASP Risk Rating Methodology:

Vulnerability	Description	Risk Level	Affected Component	PoC Available
SQL Injection	Unsanitized input in login form allows SQL code execution	High	Login form	✅
Reflected XSS	JavaScript injection executes when input is reflected unsanitized	Medium	Search box / URL parameter	✅
Stored XSS	Malicious script stored in user profile executes on page load	High	Profile comment section	✅
CSRF (Password Change)	Attacker can trick users to submit a request that changes their password	High	Account Settings Form	✅
Insecure Cookie Flags	Session cookies lack HttpOnly and Secure flags	Medium	Authentication tokens	✅
Open Redirect	Redirection URL can be manipulated by attacker	Low	Logout or redirect page	✅ (partial)

2. Screenshots (Proof-of-Concept)

Figure 1: SQL Injection in Login Form

- Payload Used: ' OR '1'='1



- Result: Authenticated as admin without valid credentials.

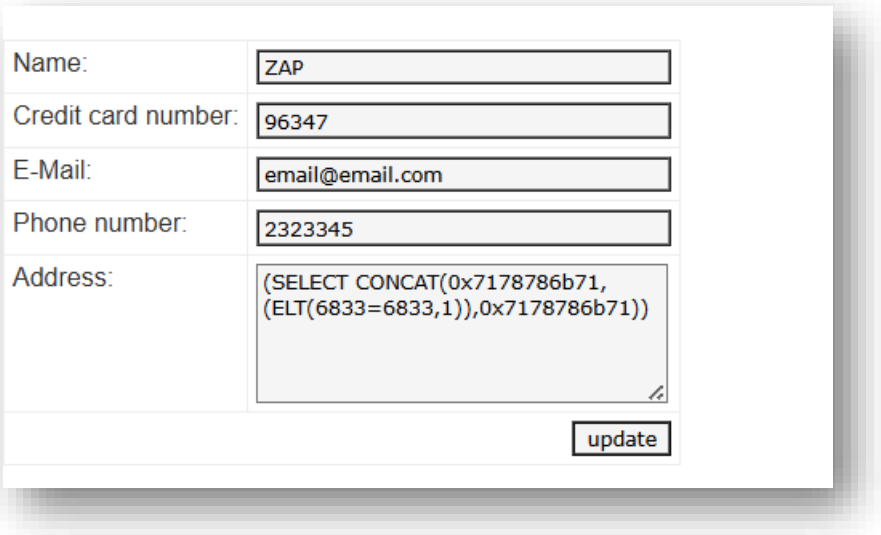


Figure 2: Stored XSS in Profile Comment Section

- Payload Used: <script> alert('XSS') </script>
- Result: Alert box triggered for every user viewing the page.

Figure 3: Reflected XSS in Profile Search Section

- Payload Used: <script> alert('XSS_Reflected') </script>
- Result: Alert box triggered for every user viewing the page.

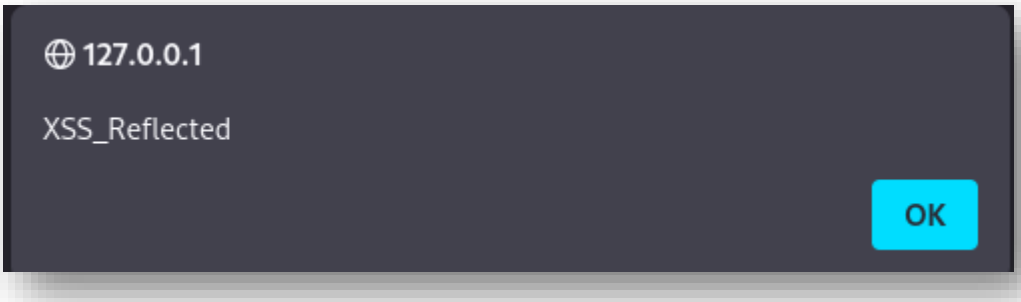


Figure 4: DOM-Based XSS in URL Section

- Go to: /vulnerabilities/xss_d/
- Use URL: ?default=<script>alert('XSS_DOM') </script>
- Alert box fires

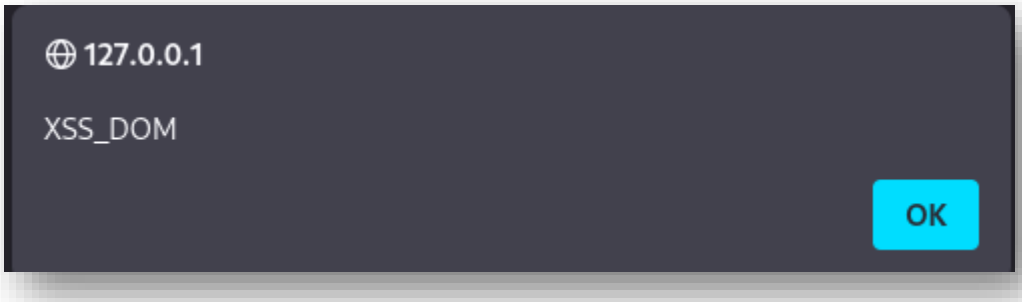


Figure 5: CSRF Exploit PoC

- A simple HTML form was crafted and auto-submitted to change the victim's password silently.

```
<!-- File: csrf-poc.html -->
<!DOCTYPE html>
<html>
  <body>
    <h2>This is a gift card offer!</h2>

    <form action="http://127.0.0.1/dvwa/vulnerabilities/csrf/" method="POST">
      <input type="hidden" name="password_new" value="hacked123">
      <input type="hidden" name="password_conf" value="hacked123">
      <input type="hidden" name="Change" value="Change">
      <input type="submit" value="Click here to claim">
    </form>

    <script>
      // Automatically submit the form when the page loads
      document.forms[0].submit();
    </script>
  </body>
</html>
```

- Double-click to open csrf-poc.html in your browser (while still logged into DVWA).
- Try logging in again with:
    - Username: `admin`
    - Password: `hacked123`

  You can log in with the new password, the **CSRF attack was successful**.

3. Discussion of Results

The results show that the application is **highly vulnerable** to critical web attacks. The **SQL Injection** flaw directly affects authentication and could allow database compromise. **Stored XSS** could be used to steal session tokens or conduct phishing attacks. The **CSRF vulnerability** allows full account takeover if the victim is logged in while visiting a malicious page.

Less severe but still significant issues like **missing cookie security flags** increase the risk of session hijacking, especially in unencrypted (HTTP) environments.

4. Challenges Faced

Challenge	How It Was Addressed
False Positives from Intruder fuzzing	Manually verified each result using Repeater and browser interaction
Encoding Issues	Used Burp Decoder to transform payloads (e.g., URL encode/HTML encode)
Session Expiry During Testing	Logged in repeatedly using Burp macros (optional in Pro version)
Limited Burp Scanner Access (Community)	Relied on manual testing for logic flaws and parameter manipulation
CSRF Tokens Changing Dynamically	Used intercepts and dynamic CSRF token extraction manually

CHAPTER 5

CONCLUSION

1. Project Summary

This project successfully demonstrated the process of conducting a comprehensive **Vulnerability Assessment and Penetration Testing (VAPT)** on a web application using **Burp Suite**. The approach simulated real-world attacks in a controlled environment to uncover critical security flaws in the target application, which was a deliberately vulnerable web app (DVWA).

The testing covered key OWASP Top 10 vulnerabilities, including **SQL Injection**, **Cross-Site Scripting (XSS)**, **Cross-Site Request Forgery (CSRF)**, and **Insecure Cookie Handling**. Each vulnerability was thoroughly examined using Burp Suite's tools, and proof-of-concept (PoC) exploits were documented with impact assessments and remediation strategies.

2. Learning Outcomes

Throughout this project, several valuable insights and skills were gained:

- Proficient use of **Burp Suite modules**: Proxy, Repeater, Intruder, Decoder, and Comparer.
- Deep understanding of how **client-server interactions** can be intercepted, analyzed, and exploited.
- Experience in identifying and exploiting **OWASP Top 10** vulnerabilities manually.
- Ability to develop and test **proof-of-concept exploits** safely and ethically.
- Familiarity with secure coding practices and mitigation strategies.

3. Future Work

If extended, the project can be improved by:

- Using **Burp Suite Professional** for automated vulnerability scanning.
- Expanding testing to include **REST APIs**, **mobile apps**, and **cloud services**.
- Integrating VAPT into a **CI/CD pipeline** for DevSecOps workflows.
- Exploring other tools like **OWASP ZAP**, **Nikto**, and **Nmap** for comparison and coverage.
- Applying **threat modelling** (e.g., STRIDE or DREAD) for more strategic assessments.

RECOMMENDATIONS

Based on the findings of the penetration test, the following security best practices and remediations are recommended to enhance the security of the target web application:

1. Prevent SQL Injection

- Use parameterized queries or prepared statements instead of concatenating SQL strings.
- Implement proper input validation and sanitization on both client and server sides.
- Use ORM frameworks to abstract direct SQL queries where possible.

2. Protect Against XSS

- Implement output encoding for any user-generated content displayed in the HTML, JavaScript, or URL.
- Use Content Security Policy (CSP) headers to limit script execution.
- Sanitize input using libraries like DOMPurify or server-side filters.

3. Defend Against CSRF

- Include unique, random CSRF tokens in all state-changing forms and validate them on the server.
- Use the SameSite cookie attribute set to Lax or Strict.
- Prompt users to re-authenticate before performing sensitive actions.

4. Secure Session Management

- Set HttpOnly and Secure flags on all authentication cookies.
- Implement session expiration and idle timeouts.
- Invalidate sessions after logout or abnormal activity.

5. Harden Web Application Configuration

- Disable detailed error messages on production servers.

- Restrict access to admin panels and sensitive endpoints using role-based access control (RBAC).
- Regularly update and patch all frameworks, libraries, and dependencies.

6. Implement Secure Development Lifecycle

- Conduct periodic code reviews and threat modelling.
- Automate security checks using static and dynamic testing tools (SAST/DAST).
- Train developers on secure coding practices and OWASP Top 10 awareness.

7. Run Regular VAPT and Compliance Audits

- Perform periodic internal and external penetration tests.
- Align testing practices with compliance standards like ISO 27001, PCI-DSS, or GDPR as applicable.

REFERENCES

[1] OWASP Foundation. (2023). *OWASP Top Ten Project*. Retrieved from https://owasp.org/www-project-top-ten/

[2] PortSwigger Ltd. (2024). *Burp Suite Community Edition Documentation*. Retrieved from https://portswigger.net/burp/documentation

[3] Damn Vulnerable Web Application (DVWA). (2023). *DVWA GitHub Repository*. Retrieved from https://github.com/digininja/DVWA

[4] National Institute of Standards and Technology. (2008). *Technical Guide to Information Security Testing and Assessment* (NIST SP 800-115). Retrieved from https://csrc.nist.gov/publications/detail/sp/800-115/final

[5] SANS Institute. (2022). *Web Application Penetration Testing Cheat Sheet*. Retrieved from https://www.sans.org/white-papers/

[6] Akinyelu, S. A., & Adewumi, A. O. (2021). *Comparative Analysis of Burp Suite, OWASP ZAP, and Nikto in Detecting Web Vulnerabilities*. International Journal of Cyber-Security and Digital Forensics, 10(3), 183–195.

[7] Verizon. (2023). *Data Breach Investigations Report (DBIR)*. Retrieved from https://www.verizon.com/business/resources/reports/dbir/

[8] OWASP. (2023). *Web Security Testing Guide (WSTG) v4.2*. Retrieved from https://owasp.org/www-project-web-security-testing-guide/