Security Audit of Internee.pk's Website

1. Introduction

Web applications are prone to cyber attacks because they involve user data and allow public access through the internet. Flaws in web applications can cause serious problems like data breaches, unauthorized access, and loss of service. Hence, security audits are required periodically to scan vulnerabilities and enhance the security status of a website.
This report highlights the findings of a security audit, which is a part of an internship assignment. The aim of this security audit is to examine the security status of a web application environment and learn about typical vulnerabilities like SQL Injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF). To practice in a safe and legal manner, a vulnerable test application is used instead of a live website.

2. Objective

The main objective of this task is to identify potential security vulnerabilities in a web application and suggest appropriate security improvements. The specific objectives are:

- To perform security testing on critical areas such as login pages, user profiles, and input forms
- To detect common web vulnerabilities including SQL Injection, XSS, and CSRF
- To use automated security tools to scan and analyse the application
- To evaluate the risk level of identified vulnerabilities
- To provide recommendations for improving website security

This task helps in understanding how security audits are conducted in real-world environments.

3. Tools Used

The following tools and platforms were used to perform the security audit:

- **OWASP ZAP** – for automated vulnerability scanning
- **Burp Suite (Community Edition)** – for manual request inspection and testing
- **OWASP Juice Shop** – a deliberately vulnerable web application used as a testing environment.
- **Web Browser** – for manual interaction with the application
- **GitHub** – for documentation and reporting

These tools are widely used in the cybersecurity industry for web application security testing.

4. Test Environment

For ethical and legal reasons, testing was conducted on a controlled and safe environment. Instead of testing a real production website, the OWASP Juice Shop application was deployed locally as a test environment.

The test environment included:

- A locally hosted vulnerable web application
- A web browser for interacting with the application.
- Security tools for scanning and analysis

This setup ensured that all testing activities were safe, legal, and focused only on learning and evaluation.

5. Scan Results

Automated scanning was performed using OWASP ZAP to analyze the web application for common vulnerabilities. The tool crawled the application and tested various endpoints including login forms and user input fields.

The scan results indicated the presence of multiple security issues such as:

- Lack of proper input validation
- Insecure handling of user inputs
- Missing or weak security headers
- Potential SQL Injection and XSS points

These findings highlight how automated tools can quickly identify common security weaknesses in web applications.

6. Vulnerabilities Found

The following key vulnerabilities were identified during the audit:

SQL Injection

User input was not properly validated before being processed by the database. This could allow an attacker to manipulate database queries and gain unauthorized access to sensitive information.

Cross-Site Scripting (XSS)

Some input fields accepted and displayed user input without proper sanitization. This could allow an attacker to inject malicious scripts into the browser of another user.

Cross-Site Request Forgery (CSRF)

The application did not verify the authenticity of some user requests. This could allow an attacker to trick users into performing unwanted actions while they are logged in.

Security Misconfiguration

Certain security headers were missing or not properly configured, increasing the risk of browser-based attacks.

7. Risk Analysis

Each vulnerability was analysed based on its potential impact:

- **SQL Injection (High Risk):** Can lead to full database compromise, data theft, or data deletion
- **XSS (Medium Risk):** Can be used to steal session cookies or execute malicious scripts in user browsers.
- **CSRF (Medium Risk):** Can force users to perform unintended actions.
- **Security Misconfiguration (Low to Medium Risk):** Can increase exposure to browser-based attacks.

Risk analysis helps prioritize which vulnerabilities should be fixed first to reduce overall security risk.

8. Recommendations

To improve the security of the website, the following recommendations are suggested:

- Implement strict input validation and sanitization for all user inputs.
- Use prepared statements or parameterized queries for database access.
- Add CSRF protection tokens to all forms and state-changing requests.
- Configure proper security headers such as Content-Security-Policy and X-Frame-Options
- Enforce HTTPS to protect data transmitted between users and the server.
- Regularly update and patch the web application and its dependencies.

Applying these recommendations will significantly reduce the risk of common web attacks.

9. Conclusion

This security audit assignment has given me hands-on experience with security testing of web applications. Using automated and semi-automatic security testing tools, a number of common vulnerabilities have been identified and examined. The findings have shown the need for regular security audits to be carried out in order to identify vulnerabilities before they can be exploited by attackers.

This assignment has also shown the role of tools such as OWASP ZAP and Burp Suite in evaluating the security of applications.