Executive Summary

This penetration testing exercise focused on evaluating the security posture of a simulated web application through the **Basic Challenges** provided on HackThisSite. The objective was to identify vulnerabilities commonly found in web applications, such as weak authentication mechanisms, improper input validation, insecure storage of sensitive data, and poor session management practices.

Through the completion of challenges (Levels 1-11), several critical vulnerabilities were discovered, including exposed credentials, directory traversal, and improper handling of user inputs. These vulnerabilities highlight gaps in web application security that, if exploited in a real-world scenario, could lead to unauthorized access, data breaches, and significant reputational damage.

The findings and recommendations presented in this report aim to improve the security of web applications by addressing identified weaknesses and implementing best practices in web development and security management.

Scope

The scope of this penetration test was limited to the **Basic Challenges** provided on HackThisSite, a legal and controlled environment designed for cybersecurity training. The test simulated real-world scenarios by identifying and exploiting vulnerabilities in the application's design and implementation.

The following activities were performed within the testing scope:

- Analysis of client-side and server-side vulnerabilities.
- Inspection of web application components, including forms, cookies, and URL parameters.
- Testing of authentication and session management mechanisms.
- Review of application error handling and data exposure.

The testing adhered to ethical guidelines and HackThisSite's terms of service, ensuring that all activities remained within the designated environment.

Findings

Level 1 – Exposed Credentials in HTML Source Code

Challenge Description:
The challenge involved discovering a password embedded within the web application. The objective was to locate and use the password to gain access to the next level.

Vulnerability:
Exposed credentials in the HTML source code. The application stored sensitive information, specifically the password, directly in the source code of the webpage.

Exploitation:
By right-clicking on the webpage and selecting "View Page Source," the password was found within an HTML comment.

Impact:
Exposing credentials in the source code allows attackers to retrieve sensitive information without requiring advanced technical skills. In a real-world scenario, this could lead to unauthorized access to sensitive resources or systems.

Remediation:
To mitigate this vulnerability:

1. Never hardcode sensitive information, such as passwords, in the front-end code.
2. Store sensitive data securely on the server and use secure authentication mechanisms.
3. Implement a robust access control policy to prevent unauthorized access.

Level 2 - Exposed Login Credentials

Challenge Description:
The challenge required accessing a login-protected page by finding the correct credentials.

Vulnerability:
Insecure cookie storage. The username and password were stored in plaintext cookies.

Exploitation:
Used the browser's developer tools to inspect cookies and retrieve the login information.

Impact:
Storing sensitive information in plaintext cookies can lead to unauthorized access if the cookies are intercepted or inspected.

Remediation:
Use session tokens instead of storing sensitive data in cookies. Ensure cookies are encrypted and have attributes like `Secure` and `HttpOnly` enabled.

Level 3 - Bypassing Login Authentication

Challenge Description:
The login form could be bypassed by modifying the HTML form input values.

Vulnerability:
Client-side authentication. The application relied on client-side validation, allowing attackers to manipulate input fields.

Exploitation:
Edited the HTML form using browser developer tools and bypassed the authentication mechanism.

Impact:
Weak client-side validation allows attackers to gain unauthorized access.

Remediation:
Implement server-side validation and ensure sensitive processes like authentication are handled securely on the server.

Level 4 - Basic Directory Traversal

Challenge Description:
The challenge required navigating directories to find sensitive files.

Vulnerability:
Improper handling of user input in directory paths allowed directory traversal attacks.

Exploitation:
Used `../` in the URL to navigate to a file containing the required information.

Impact:
Attackers could access unauthorized files, exposing sensitive information.

Remediation:
Validate and sanitize user inputs to prevent directory traversal attacks. Use access controls to restrict sensitive directories.

Level 5 - Hidden Form Field Manipulation

Challenge Description:
A hidden form field contained sensitive information that could be altered to bypass restrictions.

Vulnerability:
Insecure use of hidden form fields.

Exploitation:
Viewed and modified the hidden field’s value using browser developer tools to bypass the intended functionality.

Impact:
Hidden fields are easily visible and manipulable, leading to unauthorized actions.

Remediation:
Avoid relying on hidden fields for critical data. Use server-side validation to verify user actions.

Level 6 - Simple CAPTCHA Bypass

Challenge Description:
The challenge required bypassing a CAPTCHA implemented on the form.

Vulnerability:
Weak CAPTCHA implementation and validation.

Exploitation:
Used browser developer tools to inspect the CAPTCHA and input the correct value directly from the page source.

Impact:
Attackers can bypass poorly implemented CAPTCHAs to automate malicious activities.

Remediation:
Implement CAPTCHA solutions with backend validation and complexity to deter bots.

Level 7 - Brute Force Protection

Challenge Description:
A brute force attack was used to guess the login password.

Vulnerability:
Lack of rate-limiting and account lockout mechanisms.

Exploitation:
Used a simple script to try multiple passwords until the correct one was found.

Impact:
Attackers can exploit weak passwords or test multiple combinations to gain unauthorized access.

Remediation:
Implement rate-limiting, account lockouts, and password strength requirements.

Level 8 - Case Sensitivity Exploit

Challenge Description:
The application failed to handle case sensitivity correctly in password validation.

Vulnerability:
Weak authentication logic not enforcing strict case sensitivity.

Exploitation:
Tried variations of the password with different cases until successful login.

Impact:
Failure to enforce strict password policies weakens authentication mechanisms.

Remediation:
Ensure consistent handling of password inputs with strict case sensitivity checks.

Level 9 - Source Code Review

Challenge Description:
The password was stored in plaintext in the source code of the application.

Vulnerability:
Exposed sensitive data in application code.

Exploitation:
Viewed the source code using "View Page Source" to retrieve the password.

Impact:
Exposing sensitive information in source code leads to unauthorized access.

Remediation:
Never hardcode sensitive data in source files. Use environment variables or secure key management practices.

Level 10 - Authentication Bypass via URL Manipulation

Challenge Description:
The challenge involved accessing restricted pages by manipulating the URL parameters.

Vulnerability:
Insecure direct object references (IDOR).

Exploitation:
Modified the URL to access restricted resources directly.

Impact:
Attackers can bypass access controls to access unauthorized data.

Remediation:
Implement proper authorization checks on the server-side for all resources.

Level 11 - Improper Error Handling

Challenge Description:
Error messages disclosed sensitive information useful for exploitation.

Vulnerability:
Verbose error messages revealed application logic and sensitive data.

Exploitation:
Triggered an error and extracted useful information from the response to guess the correct input.

Impact:
Verbose error messages can give attackers valuable insights into the application's internals.

Remediation:
Implement generic error messages for end users while logging detailed messages securely on the server.

Recommendations

Based on the vulnerabilities identified during the penetration test, the following recommendations are proposed to enhance the security of web applications:

1. **Authentication and Authorization**:
    - Enforce strong password policies, including complexity and length requirements.
    - Implement server-side validation for all authentication mechanisms.
    - Use multi-factor authentication (MFA) to add an additional layer of security.
2. **Input Validation and Output Encoding**:
    - Sanitize and validate all user inputs to prevent injection attacks and directory traversal.
    - Use parameterized queries to mitigate SQL injection risks.
3. **Session Management**:
    - Use secure cookies with attributes like `HttpOnly` and `Secure`.
    - Implement session expiration and invalidation mechanisms for inactive users.
4. **Error Handling**:
    - Replace verbose error messages with generic messages for end users.
    - Log detailed errors securely on the server side for debugging purposes.
5. **Sensitive Data Protection**:
    - Avoid storing sensitive information, such as credentials, in plaintext in cookies or source code.
    - Use encryption to protect sensitive data both in transit and at rest.

6. **Rate Limiting and Brute Force Protection**:
    - Implement rate-limiting mechanisms to prevent brute force attacks.
    - Lock accounts temporarily after multiple failed login attempts.
7. **Regular Security Audits**:
    - Conduct regular penetration testing and code reviews to identify vulnerabilities.
    - Update the application regularly to address emerging threats and vulnerabilities.

Conclusion

The penetration testing exercise revealed critical and high-risk vulnerabilities in the simulated web application environment. Addressing these vulnerabilities is crucial for improving the overall security posture and protecting sensitive information.

By implementing the recommended security measures, organizations can mitigate the risks posed by common web application vulnerabilities and reduce their attack surface. Continuous improvement, regular security assessments, and adherence to secure coding practices will ensure that web applications remain resilient against evolving cyber threats.

This exercise provided valuable insights into identifying and addressing web application vulnerabilities, emphasizing the importance of proactive security practices in modern software development.