Penetration Testing Report

Executive Summary

This penetration testing report documents the findings and recommendations from a security assessment conducted on the web application "Basic Hack The Site." The objective was to identify security vulnerabilities, assess their potential impact, and recommend remediation strategies to enhance the security posture of the application. The assessment uncovered multiple vulnerabilities across different levels, ranging from basic HTML source code viewing to advanced server-side attacks.

Scope

The scope of this penetration test included the following levels and resources:

1. **Level 1, 2 & 3:** Viewing HTML Source Code
2. **Level 4 & 5:** Editing Web Pages in Browser
3. **Level 6:** ASCII Character Codes
4. **Level 7:** Linux `cal` Command
5. **Level 8 & 9:** Server-Side Includes (SSI) Injection
6. **Level 10:** Cookies Manipulation
7. **Level 11:** Apache .htaccess File

Vulnerability Descriptions, Key Findings, and Security Recommendations

Level 1, 2 & 3: Viewing HTML Source Code

Description:

The application exposes sensitive information in the HTML source code which can be easily viewed by users.

Key Findings:

- Users can view sensitive data like comments, hidden fields, or configuration details by inspecting the HTML source.

Recommendations:

- Remove any sensitive information from the client-side HTML.
- Ensure comments and hidden fields do not expose critical data.
- Implement server-side validation to prevent reliance on client-side security.

Level 4 & 5: Editing Web Pages in Browser

Description:

Users can edit the web page's content directly in the browser, which can lead to unauthorized actions or content manipulation.

Key Findings:

- Users can manipulate form fields or hidden inputs to bypass client-side validations.
- Possible to alter displayed content to mislead or phish other users.

Recommendations:

- Implement strict server-side validation to ensure integrity and authenticity of user inputs.
- Use Content Security Policy (CSP) to restrict the types of content that can be loaded and executed.

Level 6: ASCII Character Codes

Description:

Vulnerabilities related to improper handling of ASCII character codes can lead to input validation issues or injection attacks.

Key Findings:

- Special characters can be exploited to perform injections or bypass input filters.

Recommendations:

- Implement robust input validation and sanitization to handle special characters correctly.
- Encode all user inputs before processing.

Level 7: Linux `cal` Command

Description:

Exploitation of the `cal` command can reveal system information or be used to execute unintended commands.

Key Findings:

- Users can inject commands through improperly validated input parameters.

Recommendations:

- Validate and sanitize all inputs passed to system commands.
- Use parameterized queries or secure API calls to interact with the system.

Level 8 & 9: Server-Side Includes (SSI) Injection

Description:

SSI injection allows attackers to execute server-side scripts by injecting SSI directives into input fields.

Key Findings:

- Application is vulnerable to SSI injection due to lack of input sanitization.

Recommendations:

- Disable SSI on the server if not required.
- Sanitize all inputs to prevent the inclusion of SSI directives.
- Implement input validation and encoding.

Level 10: Cookies Manipulation

Description:

Manipulating cookies can lead to unauthorized access or session hijacking.

Key Findings:

- Users can edit cookies to escalate privileges or impersonate other users.

Recommendations:

- Implement secure cookie attributes (HttpOnly, Secure, SameSite).
- Use server-side session management and validation.
- Encrypt sensitive cookie data.

Level 11: Apache .htaccess File

Description:

Improper configuration of `.htaccess` files can lead to unauthorized access or directory traversal attacks.

Key Findings:

- Misconfigured `.htaccess` files expose sensitive directories or files.

Recommendations:

- Review and restrict `.htaccess` file configurations.
- Implement proper access controls and permissions on sensitive directories.
- Regularly audit and monitor `.htaccess` files for unauthorized changes.

Conclusion

The penetration test identified several vulnerabilities across different levels of the web application. Each finding has been documented with corresponding security recommendations. Implementing these recommendations will significantly enhance the security posture of the application, protecting it against potential attacks and unauthorized access. Regular security assessments and adherence to best practices are recommended to maintain a robust security framework.