Name: Jennifer Omphile Mogami

Student No. : ST10119337

Pathway: APDS7311

Module: Application Development Security

Lecturer: Isaac Leshaba

POE Part 1

Table of Contents

Proposal for Secure Inter-Departmental Bulletin Board System

Introduction

We provide a thorough strategy for the creation of an Inter-Departmental Bulletin Board System in response to the National Government's urgent need for a safe and private platform to enable inter-departmental collaboration on delicate subjects. This system's main goal is to make it possible for government agencies to work together on highly sensitive issues that call for the expertise and resources of several agencies. This proposal outlines a comprehensive strategy to guarantee the highest level of security for our bulletin board system, with a focus on the login procedure.

The National Government's confidence in our capacity to provide a safe platform for this vital function emphasizes the seriousness of our responsibilities. As a crucial tool for the government, this bulletin board system will require strong security measures to protect sensitive data and guarantee that only authorized users may access the system.

1. Registration of New Users and Login Process

a. HTTP Requests and Traffic Security

HTTPS Implementation

The Secure Hypertext Transfer Protocol (HTTPS), also known as an Internet standard protocol, is used on the Internet to encrypt and protect the secrecy of the regular HTTP protocol. It oversees applying different high-security cryptographic methods to user data to prevent data theft and leakage of sensitive information, including passwords and other financial data pertaining to credit cards and debit cards. With the use of the TLS and SSL layer certificates, HTTPS enables the web browser to identify and flag websites that have an additional degree of security. To offer secrecy and authenticity in communication, it provides an additional layer of encryption (awmankit, 2022). To ensure the security of HTTP requests and traffic, we will implement strong encryption using HTTPS:

- **SSL/TLS Encryption**: Utilize SSL/TLS encryption to secure all communication between clients and the server.
- **Certificate Management**: Carefully manage SSL/TLS certificates, ensuring they are up-to-date and correctly configured.

- **Strict HTTPS Policy**: Enforce a strict "HTTPS-only" policy, rejecting any non-secure HTTP connections.
- **Action method and HTTP security:** Implementing proper action methods and enforcing HTTPS ensures secure transmission of data.

Security Headers

Browsers are required to abide by security header instructions that are transmitted in the HTTP header response. A web server's response to a browser attempting to access a web page is known as an HTTP header. When a web page is unavailable, for example, the header response (400 response header), indicates this. Such a security instruction will prevent a browser from downloading harmful data from other websites (Montti, 2022). To enhance security, we will implement essential security headers:

- **Content Security Policy (CSP)**: Define and enforce a CSP to restrict the sources of executable scripts, styles, and other content.
- **HTTP Strict Transport Security (HSTS)**: Set HSTS headers to instruct browsers to interact with our application exclusively through HTTPS.
- **X-Content-Type-Options**: Implement this header to prevent browsers from interpreting files as something other than declared by the Content-Type header.

b. Input Validation

Client-Side Validation

Client-side validations help the user submit accurate information into form fields or other locations on a GUI where information is needed. When a user clicks out of the input field after entering an email address but forgetting the "@" sign, a notice may show alerting them to their error. Additionally, the user could be prevented from pressing "Submit" until they have corrected the problem; as a result, no HTTP request is sent (Rex, 2022). To ensure data integrity and prevent injection attacks, we will implement comprehensive input validation:

- **Client-Side Validation Libraries**: Utilize JavaScript libraries or frameworks to perform real-time input validation.
- **Immediate Error Feedback**: Provide users with immediate feedback on input errors to prevent unnecessary server requests.
- **Empty fields not allowed:** Enforcing that empty fields are not allowed helps prevent submission of incomplete or invalid data.

Server-Side Validation

User experience is less important than how incoming data will really be handled and stored when it comes to server-side validation (Rex, 2022). Server-side validation is essential to reject malicious or malformed input effectively:

- **Data Validation Libraries**: Use server-side libraries and frameworks with built-in validation capabilities.
- **Input Sanitization**: Sanitize input data to remove potentially harmful characters or content.
- **Regular Expression Validation**: Employ regular expressions to validate complex data types, such as email addresses or usernames.
- **Input validation using RegEx/other framework:** Utilizing regular expressions or a validation framework is a strong practice for input validation.

c. Storing and Hashing of Passwords

Password Hashing

In order to protect against the risk of password breaches, the technique of "password hashing" entails algorithmically converting a password into ciphertext or an irreversibly obfuscated version of itself. A mathematical hashing algorithm essentially changes one string of characters in a password into a whole new string (Team, 2022). To securely store user passwords, we will adopt industry best practices:

- **Cryptographic Hashing**: Implement strong cryptographic hashing algorithms like bcrypt.
- **Unique Salt for Each User**: Generate a random and unique salt for each user to prevent rainbow table attacks.

Password Recovery

- **Secure Password Reset**: Implement a secure password reset mechanism, involving sending reset links via email with limited time validity.
- **Multi-Factor Authentication (MFA)**: Offer MFA as an option for account recovery to enhance security.

d. Maintaining Authentication State

To maintain the authentication state of users securely, we will use the following mechanisms:

- **Session Management**: Implement robust session management techniques, including assigning unique session tokens for each user session. Session management refers to a series of requests and answers connected to a certain user and is used to support secure interactions between a user and a service or application (Packetlabs, 2022).
- **Secure Cookies**: Use secure HTTP cookies with HttpOnly and Secure flags to prevent cookie theft through JavaScript and enforce cookie transmission over secure channels. Small data packets known as HTTP cookies are kept in your browser. This data is vulnerable to assaults since it can include private information like passwords or user details. You can 'secure' your cookies to reduce susceptibility by adding properties to the set cookies, making it more difficult for outsiders to change.

  Cookies are made more secure by Really Simple SSL using the HttpOnly, secure, and use_only_cookies settings. Making these adjustments to the plugin, in our opinion, is an easy way for us to contribute to the overall security of your website, especially because Really Simple SSL assists you in protecting your website by converting your site to SSL. (Hulsebos, 2022).
- **Session Timeout**: Configure session timeouts to automatically log out users after a defined period of inactivity.

e. Credential Security

Adversaries continue to primarily enter an organization's IT environment through compromising the credentials of Active Directory accounts. They employ a variety of strategies, including brute-force assaults, phishing, credential stuffing, and password spraying (Dibley, 2023).

Credential security is critical to prevent unauthorized access:

- **Password Policies**: Enforce strong password policies, including minimum length, complexity requirements (e.g., uppercase, lowercase, digits, special characters), and password expiration.
- **Account Lockout**: Implement account lockout mechanisms to protect against brute force attacks, locking an account after a specified number of failed login attempts.
- **Rate Limiting**: Apply rate limiting to thwart rapid and repeated login attempts.
- **Use Multifactor Authentication (MFA)**
- **Avoid Reusing Passwords:** Include a notification to warn users from entering the same passwords.

f. Overall Flow of Login Process (Salinkar, 2021)

Our login process will follow this secure flow:

1. User provides credentials (username and password).
2. Client-side validation checks for common errors before sending data to the server.
3. The server performs thorough server-side validation to reject any malicious or malformed input.
4. The password is securely hashed and compared to the stored hashed password.
5. If authentication is successful, a secure session or JWT token is generated and sent to the user.
6. The user's authentication state is maintained for the duration of their session.
7. After a successful login, the user gains access to the bulletin board.

2. Protection Against Common Security Threats

a. Username Harvesting (Overby, 2022)

To protect against username harvesting:

- **Rate Limiting**: Implement rate limiting on login attempts to slow down attackers attempting to harvest usernames.
- **Username Enumeration**: Ensure that error messages do not reveal whether a provided username is valid.
- **Account Lockout**: Implement account lockout mechanisms to prevent excessive login attempts, which can assist in username harvesting.
- Make sure to return a generic “No such username or password” message when a login failure occurs.
- Make sure the HTTP response, and the time taken to respond are no different when a username does not exist, and an incorrect password is entered.
- Ensure that “forgotten password” page does not reveal usernames.
- If the password reset process involves sending an email, have the user enter their email address. Then send an email with a password reset link if the account exists.

b. Brute Force Attacks

In a brute force attack, a hacker makes automated server requests that iterate over a list of possible responses to a problem statement in attempt to obtain sensitive online application data and accounts (DATADOME, 2023).

To protect against brute force attacks:

- **Account Lockout**: Implement account lockout mechanisms after a certain number of failed logins attempts within a specified timeframe.
- **CAPTCHA**: Introduce CAPTCHA challenges after a certain number of failed logins attempts to deter automated brute force attacks.
- **Enforcing strong, unique passwords.**
- **Limited login attempts:** Limiting login attempts is an effective strategy against brute force attacks.

c. Session Jacking

By secretly getting the session ID and impersonating the authorized user, a technique called session hijacking, often referred to as TCP session hijacking, allows one to take control of an online user session. After gaining access to the user's session ID, the attacker can pretend to be that user and do whatever action that user is permitted to perform on the network.

This kind of attack results in the ability to access a server without having to login to it. If the communication session is ongoing, the attacker may stop worrying about server authentication after hijacking a session (Arampatzis, 2023).

To prevent session jacking:

- **HTTPS**: Utilize HTTPS to encrypt session data during transmission, making it difficult for attackers to intercept.
- **Session Timeout**: Implement session timeouts to automatically log out users after a period of inactivity.
- **Session Regeneration**: Generate a new session ID upon login to invalidate any existing session IDs.

d. Session Fixation

A web-based attack method called session fixation involves tricking the user into viewing a URL that has a predetermined session identifier. A victim of a session fixation assault may be

able to transfer money, steal sensitive information, or entirely take over the victim's session. Find out the causes of session obsession and how to avoid it (Banach, 2021).

To mitigate session fixation attacks:

- **Session Regeneration**: Generate a new session ID upon login to invalidate any existing session IDs.
- **Session Rotation**: Periodically rotate session IDs during a user's session to minimize the window of opportunity for fixation attacks.
- **Multi-factor Authentication (MFA)**: Offer MFA as an additional layer of security to mitigate session fixation risks.

Conclusion

In conclusion, our proposed security measures aim to create a highly secure inter-departmental bulletin board system. This system will ensure that only authorized users can access the system while protecting against common security threats. We are committed to maintaining the confidentiality of sensitive government information and continuously monitoring and adapting our security practices to address emerging threats effectively. Our approach prioritizes data security and user authentication, enabling government departments to collaborate securely on resolving critical issues.

References

Arampatzis, A. (2023, July 20). *Venafi*. Retrieved from What Is Session Hijacking?: https://venafi.com/blog/what-session-hijacking/

awmankit. (2022, March 27). *Geeks for Geeks*. Retrieved from How to secure HTTP requests ?: https://www.geeksforgeeks.org/how-to-secure-http-requests/

Banach, Z. (2021, July 16). *Invicti*. Retrieved from Understanding session fixation attacks: https://www.invicti.com/blog/web-security/session-fixation-attacks/

DATADOME. (2023, March 01). *DATADOME*. Retrieved from How to Prevent Brute Force Attack with 9 Advanced Strategies: https://datadome.co/bot-management-protection/how-to-prevent-brute-force-attacks/

Dibley, J. (2023, April 21). *Netwrix*. Retrieved from Securing Account Credentials to Protect Your Organization: https://blog.netwrix.com/2023/04/21/secure-credentials-and-protection/

Hulsebos, A. (2022, September 30). *Really Simple SSL*. Retrieved from What are Secure Cookies?: https://really-simple-ssl.com/definition/what-are-secure-cookies/

Montti, R. (2022, February 23). *Search Engine Journal*. Retrieved from 5 HTTP Security Headers You Need To Know For SEO: https://www.searchenginejournal.com/http-security-headers/415404/#close

Overby, S. (2022, November 28). *Mimecast*. Retrieved from 7 Ways to Protect Against Credential Theft: https://www.mimecast.com/blog/7-ways-to-protect-against-credential-theft/

Packetlabs. (2022, September 09). *Packetlabs*. Retrieved from Session Management in HTTP: How does it work?: https://www.packetlabs.net/posts/session-management/#:~:text=Session%20management%20is%20used%20to,associated%20with%20that%20particular%20user.

Rex, N. (2022, June 07). *Medium*. Retrieved from Client side validation Vs. Server side validation: https://medium.com/@rexnino/client-side-validation-vs-server-side-validation-2ef116a27bcf

Salinkar, S. (2021, April 13). *Medium*. Retrieved from A guide to designing successful Login experiences: https://medium.muz.li/a-guide-to-designing-successful-login-experiences-d9cdb81877ec

Team, S. (2022, July 26). *STYTCH*. Retrieved from What is password hashing?: https://stytch.com/blog/what-is-password-hashing/#:~:text=Password%20hashing%20is%20the%20practice,the%20threat%20of%20password%20breaches.