Security Design Review (SDR) Document

# Project Information

Project Name: badstore.net

Reviewer(s): Dimitar Rebranin

Date: 11/14/24

Design Document Version: v1.0

Design Team Contact:

---

# Stage 1: Study

## 1. Documentation Review

**High-Level Understanding**  
The initial overview of the badstore.net shows that the website(application) was built vulnerable on purpose. Primarily designed for training and testing. It is a small e-commerce site with few security flaws such as SQL injection vulnerabilities, XSS, weak authentication and more.

**Threat-Aware Review**

* Potential exposure to SQL injection in forms or URLs due to insufficient input validation.
* Possible weak or unencrypted authentication, which could expose user credentials.
* Lacks role-based access control, allowing unauthorized users to access sensitive information.
* Potential for sensitive data (e.g., user credentials, personal information) to be exposed in plaintext, especially if stored without encryption.

## 2. Notes and Observations

* Observations of insufficient CSRF protection.
* Forms and other user inputs appear to lack input sanitation, increasing injection risks.
* Noted lack of secure transport layers, making data susceptible to interception.

# Stage 2: Inquire

Team’s Security Concerns:

* Concerns about ensuring real-world relevance in vulnerabilities while maintaining safe sandboxed environments.
* Questions around implementing automated scans for real-time feedback during testing exercises.

Follow-up questions:

* **Session Management**: Are there controls to prevent session hijacking?
* **Data Handling**: Is there sensitive information handled in a controlled manner?

# Stage 3: Identify

## Focus Areas

* **Interfaces**: Form and input interfaces are prime areas to examine, especially for sanitization and validation mechanisms to prevent injection attacks.
* **Storage**: Assess if sensitive information is encrypted and stored securely in the database.
* **Communications**: Determine whether all client-server communications use secure channels (e.g., HTTPS).

## Degree of Security Focus

The design is intentionally insecure to allow for real-world vulnerability testing. However, clear distinctions in the documentation between simulated and real-world vulnerabilities should be made to ensure safe operation within its training context. Key protections like HTTPS enforcement, encrypted database fields, and session security need attention.

# Stage 4: Collaborate

## Risk Discussion

**Identified Risks and Mitigations**

* **SQL Injection**: Mitigation through parameterized queries and input sanitization.
* **Cross-Site Scripting (XSS)**: Mitigation with strict content security policies and HTML escaping.
* **Session Management**: Implementation of session cookies with secure and HTTP-only flags.

## Solution Exploration

**Alternative Solutions**

1. **SQL Injection**:
   * *Parameterized Queries*: Strong protection but requires consistent application across the site.
   * *Stored Procedures*: Could limit injection attacks, though they may add complexity to the system.
2. **XSS**:
   * *Content Security Policy (CSP)*: Prevents inline scripts but requires detailed setup.
   * *Input Escaping*: Simple but may not cover all cases.

## Decision Documentation

**Final Design Decisions**

* Prioritize CSP implementation for XSS.
* Implement HTTPS across the application, particularly for login and sensitive data exchange.
* Ensure all sensitive data in storage is encrypted.

# Stage 5: Write

## Report Organization

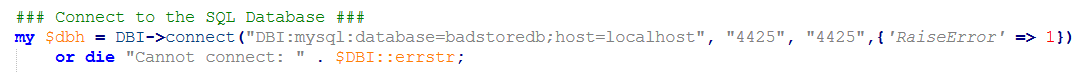
**High-Priority Issues**

* SQL Injection vulnerabilities in input fields – lack of input sanitization, it could lead to user input of more SQL statements/commands

A white background with black text

Description automatically generated

* Hardcoded database credentials



* Lack of Access Controls – there is no authentication or access control in place to ensure only authorized users can access the admin portal

A screenshot of a computer

Description automatically generated

**Lower-Priority Issues**

* Lack of CSRF protection.
* Clarification of data retention policies in the documentation.

## Suggestions and Alternatives

**Recommended Changes**

* Implement HTTPS across all pages.
* Enforce stricter authentication and session management controls – ensure user is authenticated and authorized as admin

A screenshot of a computer

Description automatically generated

* Incorporate parameterized queries for all database interactions.
  + Connect to the SQL database securely

A computer screen with text

Description automatically generated

* + Execute SQL queries securely with parameters

A screen shot of a computer code

Description automatically generated

* Implement a CSRF token

A screenshot of a computer screen

Description automatically generated

## Priority Rankings

* **Must**: Enforce HTTPS, secure session management, encrypt sensitive storage.
* **Ought**: Use CSP for XSS and improve access controls.
* **Should**: Add CSRF tokens on all forms to protect against CSRF attacks.

# Stage 6: Follow Up

## Change Verification

**Resolution Confirmation**

* Review implementation of HTTPS, secure session flags, and encrypted storage for confirmed changes.

## Open Issues

**Unresolved Points**

* No agreement yet on session timeouts for inactive sessions.
* Pending decisions on user role-based access controls.

## Final Sign-Off

**Review Completion**  
After verifying that high-priority changes are implemented, confirm sign-off on the security design review.

# Conclusion

**Overall Security Assessment**

The design review identifies badstore.net as a purposefully vulnerable training tool with controlled security gaps. However, improvements are recommended to prevent unintentional exploitation outside of its sandboxed environment.

**Next Steps**

* Implement agreed-upon changes and conduct a secondary review.
* Plan periodic reviews of any new or modified vulnerabilities introduced in the training environment.