Security Assessment Report on Restaurant Website Application

Couse Elec s425f
Group member :
13140568 Yu Ching Ting
13327704 WU Fei
13559724 Peng Haichen

github:
https://github.com/s350fgroup25/Food_ordering/blob/main/Food_Group25.zip
https://github.com/s350fgroup25/s425f/tree/main

Introduction

This report presents an in-depth security review of the Restaurant Website application. Leveraging Burp Suite Professional HTTP analysis, vulnerabilities were identified across multiple functional and administrative paths, including the main site, administrative interfaces, stylesheet directories, and configuration endpoints. The findings include critical SQL injection flaws, Cross-Site Request Forgery (CSRF) risks, clickjacking susceptibility, and several configuration and injection weaknesses. This report aims to document these issues comprehensively and provide actionable mitigation strategies to enhance the application's overall security posture.

Scope and Methodology

The assessment utilized Burp Suite Professional to perform comprehensive scanning, manual verification of automated findings, and detailed HTTP traffic inspection. The scope covered the following paths:

- /admin (Administrative login and management pages)
- /css (Stylesheet and UI resources)
- /config (Potential configuration files or endpoints)
- Application public paths serving data including food items, orders, and categories.

Testing techniques included injection payloads for SQLi, exploitation attempts for CSRF, header analysis for clickjacking, and resource loading inspections for mixed content and path-relative CSS vulnerabilities. Results were critically reviewed for validation to exclude false positives

Vulnerability Analysis and Findings

A01:2021 – Broken Access Control

Broken Access Control remains the most critical and prevalent security risk in modern web applications. In the Restaurant Website application, multiple instances of broken access control were detected, particularly in the administrative and configuration endpoints.

- Nature of Issue: The `/admin/login.php` endpoint was found vulnerable to SQL Injection, effectively allowing unauthorized bypass of authentication controls. Furthermore, several administrative functions lacked proper role or session verification, enabling attackers to access privileged resources without necessary permissions.
- Evidence: Burp Suite scans revealed that crafted input into the `username` parameter on admin login produced authenticated responses despite invalid credentials. This was validated through explicit payload injections that altered backend SQL queries.
- Impact: Exploitation permits unauthorized users to assume admin privileges, modify backend data such as menu items, orders, and configuration settings, and potentially manipulate or disrupt business operations.
- Mitigation:
    - Enforce strict server-side role-based access control (RBAC) checks for every sensitive API or UI resource.
    - Harden session management with secure tokens linked to user identity and roles.
    - Deny default access and implement whitelist-only access for admin paths.
    - Periodically audit access controls and permissions to avoid privilege creep.
    - Utilize logging and alerting mechanisms to detect unauthorized access attempts promptly.

A02:2021 – Cryptographic Failures

Cryptographic Failures refer to improper or missing encryption that undermines confidentiality and integrity.

- Observed Issues: Multiple pages loaded insecure content over HTTP while the main site was served over HTTPS — creating mixed content issues that facilitate man-in-the-middle (MITM) attacks. Moreover, the lack of HTTP Strict Transport Security (HSTS) enabled downgrade attacks allowing attackers to force sessions to insecure HTTP.
- Impact: Attackers can intercept sensitive login credentials, session cookies, and personal user data, exposing the system to session hijacking and data breaches.
- Examples: Browser security warnings during testing indicated mixed content load for images and style sheets. Requests to external domains were made without encryption.
- Mitigation:
  - Deploy the entire application and static resources over HTTPS exclusively.
  - Implement and enforce HSTS with preload lists for modern browsers.
  - Avoid caching sensitive data on client or intermediate proxies.
  - Use adaptive cryptographic algorithms for password storage (e.g., bcrypt, Argon2).
  - Encrypt sensitive data at rest with strong and regularly rotated keys.

A03:2021 – Injection (SQL Injection)

Injection attacks remain one of the most dangerous threats, allowing attackers to manipulate backend databases and execute arbitrary commands.

- Vulnerable Parameters:
    - username in /admin/login.php
    - categoryid in /category-foods.php
    - foodid in /foodorder.php
- Testing Results:
    - Burp Suite demonstrated successful injection by using payloads including MySQL load_file() functions referencing an attacker-controlled DNS domain. DNS callbacks confirmed active execution.
    - Time-based blind SQL injection was also detected, with injected sleeps resulting in observable delays.
- Potential Attack Scenarios: Attackers can extract complete database contents, modify orders or transactions, manipulate prices, or disable security logs.
- Recommended Countermeasures:
    - Employ strictly parameterized queries or stored procedures throughout the application.
    - Sanitize and validate all user input, employing allowlists over blocklists.
    - Limit database user permissions to minimize damage scope.
    - Integrate Web Application Firewalls (WAF) to detect and block SQL injection patterns.
    - Conduct regular code reviews and security scans to identify injection risks early.

A05:2021 – Security Misconfiguration

Security misconfiguration was detected in multiple facets of the Restaurant Website application:

- Headers:
  - Absence of `X-Frame-Options` or `Content-Security-Policy` headers allowed the application to be embedded in external frames, exposing to clickjacking attacks.
  - Server response headers revealed PHP version and Apache server details, disclosing server fingerprint information aiding attackers.
- CSS Loading:
  - Use of path-relative URLs for stylesheets combined with missing `<!DOCTYPE html>` declarations exposed quirks mode rendering vulnerabilities that potentially facilitate CSS injection.
- Impact:
  - Attackers leverage fingerprinting information for targeted exploits.
  - Clickjacking attacks can trick users into unauthorized actions.
  - Confused rendering modes jeopardize UI security and user trust.
- Mitigation Steps:
  - Configure HTTP headers to include `X-Frame-Options: DENY` or `SAMEORIGIN`.
  - Add `X-Content-Type-Options: nosniff` and strict `Content-Security-Policy` directives.
  - Suppress or mask server and framework version info in headers.
  - Use absolute URLs for static resource loading and always define modern DOCTYPE in HTML pages.
  - Regularly scan for misconfiguration using automated tools and remediate immediately.

A07:2021 – Identification and Authentication Failures

Authentication mechanisms showed vulnerabilities that could undermine identity assurance:

- Observations: Absence of account lockout or rate limiting made brute forcing credentials feasible.
- Consequence: Attackers could guess credentials, escalating to administrative privileges.
- Recommendations:
    - Enforce MFA (Multi-Factor Authentication) on admin portals.
    - Implement account lockouts after successive failures with graduated delays.
    - Follow strong password policies and secure session management.

Other Notable Issues

Mixed Content

The usage of mixed HTTP and HTTPS undermines security guarantees, allowing man-in-the-middle attacks on resources.

Path-Relative CSS Vulnerability

Relative URLs and missing Doctype declaration place the application at risk for injection and inconsistent rendering, affecting security and usability.

Conclusion

The Restaurant Website application contains multiple critical security vulnerabilities predominantly driven by direct SQL injection flaws, absence of CSRF protections, insecure resource loading methods, and lack of frame and content security measures. These security issues threaten the integrity and safety of application data, user interactions, and administrative functions. Immediate remediation through parameterized queries, robust input validation, CSRF token implementation, strict security headers, and secure resource management is imperative. Only by addressing these weaknesses comprehensively can the application be considered secure for deployment in production environments.

References

- PortSwigger Web Security Academy: SQL Injection, CSRF, Clickjacking.
- OWASP Cheat Sheets: SQL Injection Prevention, CSRF Prevention, Clickjacking Defense.
- CWE-89 (SQL Injection), CWE-352 (CSRF), CWE-16 (Configuration).
- CAPEC-66 (SQL Injection), CAPEC-62 (CSRF), CAPEC-468 (Cross-Domain Theft).