Security Audit Report

Application Under Test: Swag Labs (https://www.saucedemo.com/)

Date: 10th September 2025

Auditor: Collins Kwasi Adu

1. Executive Summary

The security audit was conducted on the Swag Labs web application to assess its resilience against common web-based threats and industry standard vulnerabilities.
The assessment focused on key user-facing features such as login, product search, product details, cart, and checkout.
The objective of this audit was to:

- Identify vulnerabilities that could be exploited by attackers.
- Assess the application's security posture in relation to secure coding practices and best practices for web applications.
- Provide actionable recommendations for strengthening the application's defense.

Key Findings:

- Authentication and session handling were functional but lacked strong protections against brute-force attacks and automated login attempts.
- Multiple missing or misconfigured HTTP security headers were detected, including the absence of **Content Security Policy (CSP)**, **Anti-Clickjacking headers**, and **Strict-Transport-Security (HSTS)**. These weaknesses increase the risk of **cross-site scripting (XSS)**, **clickjacking**, and **SSL stripping attacks**.
- Cross-Domain Misconfigurations were noted, potentially allowing untrusted origins to interact with the application in unintended way.
- Information disclosure was observed through JavaScript comments, caching policies, and header configurations, which may aid attackers in reconnaissance
- No critical vulnerabilities such as SQL Injection or Remote Code Execution were identified.

Overall Posture:
The application demonstrates a **moderate security level**, which is acceptable for a demo or training environment but would not meet enterprise-grade security standards in production. While the core functionality works as expected, the reliance on default or missing security configurations leaves the system vulnerable to common web attacks.

High-priority improvements include enforcing strict HTTP headers (CSP, HSTS, X-Frame-Options), strengthening authentication controls, and tightening cross-domain access. Addressing these issues will significantly improve the resilience of the application and reduce its exposure to common exploitation techniques.

2. Scope

In-Scope:

- Frontend application at https://www.saucedemo.com/
- Features tested: User login/logout, Product catalog & search, Product details page, Shopping cart, Checkout workflow

Out-of-Scope:

- Backend APIs (not directly accessible)
- Payment gateway integrations (simulated only)
- Administrative interfaces (not available in demo app)

3. Methodology

The security audit was performed using a **structured approach** that combined manual inspection, automated scanning tools, CI/CD pipeline integration, and AI-assisted analysis. The process followed these phases:

1. Environment Setup

- The Swag Labs application (https://www.saucedemo.com/) was selected as the target for assessment.
- Initial scans were executed locally to validate tool configurations and confirm that security reports were generated correctly.
- Once validated, the testing workflow was containerized and integrated into a **GitHub Actions CI/CD pipeline** to ensure repeatability and automation.

2. Tools and Frameworks

- **OWASP ZAP (Zed Attack Proxy):** Used for automated **Dynamic Application Security Testing (DAST)**, including active and passive scans against the web application.
- **AI Assistants (ChatGPT & Claude):** Utilized to interpret scan findings, explain vulnerabilities in plain language, and support report drafting.
- **GitHub Actions:** Configured to run ZAP inside a Docker container on every push/PR, automatically generating HTML reports.
- **GitHub Pages:** Configured to host the generated ZAP report publicly for review.

3. Testing Techniques

- **Automated Scanning:**
  - **Passive Scans** to detect misconfigured HTTP headers, information leaks, caching issues, and SSL/TLS weaknesses.
  - **Active Scans** (limited scope) to probe for XSS, SQL injection, CSRF, and authentication flaws.
- **Pipeline Integration:**
  - After successful local validation, scans were executed automatically as part of a GitHub Actions workflow.
  - Reports were exported and published to GitHub Pages:
    👉 ZAP Report

4. Test Categories

- Input validation and injection attacks (XSS, SQLi).
- Authentication and session management (brute force attempts, cookie security).
- Authorization controls (basic checks on user-level access).
- Transport security (HTTPS enforcement, HSTS, TLS configuration).
- Error handling and information disclosure.

5. Documentation & Analysis

- Findings were compiled from both local and pipeline-based ZAP scans.
- AI tools (ChatGPT, Claude) were leveraged to summarize vulnerabilities, prioritize risks, and provide remediation recommendations.
- Final results were documented in this structured security audit report for supervisor review

4. Vulnerabilities

ID	VULNERABILITY	DESCRIPTION	EVIDENCE	IMPACT
V-01	Content Security Policy (CSP) Header Not Set	The application does not set a CSP header, increasing exposure to XSS and data injection.	ZAP Alert: CSP Header Not Set (GET https://www.saucedemo.com/)	Medium – Allows potential XSS and data injection.
V-02	Missing Anti-Clickjacking Header	The app does not set X-Frame-Options or CSP frame-ancestors directive.	ZAP Alert: Missing Anti-clickjacking Header (GET https://www.saucedemo.com/)	Medium – Enables clickjacking attacks.
V-03	Cross-Domain Misconfiguration	Overly permissive CORS policy allows access from untrusted origins.	ZAP Alert: Cross-Domain Misconfiguration	Medium – May allow cross-site exploitation of app functionality.

V-04	Strict-Transport-Security (HSTS) Header Not Set	The application does not enforce HSTS, exposing users to SSL stripping.	ZAP Alert: HSTS Header Not Set	Low – Weakens transport security
V-05	X-Content-Type-Options Header Missing	The app does not prevent MIME type sniffing by browsers.	ZAP Alert: X-Content-Type-Options Header Missing	Low – Could allow content-type spoofing
V-06	Information Disclosure – Suspicious Comments	Suspicious comments in JavaScript files could reveal internal logic.	ZAP Alert: Suspicious Comments (static/js/...chunk.js)	Informational – May help attackers understand app behavior.
V-07	Retrieved from Cache	Content retrieved from shared cache may expose sensitive data.	ZAP Alert: Retrieved from Cache	Informational – Risk of data leakage in shared environments.
V-08	Modern Web Application	ZAP identified the app as a modern SPA (Single Page Application).	ZAP Alert: Modern Web Application	Informational – No direct risk, but requires AJAX spider for crawling.
V-09	User Agent Fuzzer	Responses differ based on user agent header, may expose fingerprinting issues.	ZAP Alert: User Agent Fuzzer	Informational – May allow attackers to fingerprint users/systems

5. Risk Assessment and Prioritization

Test Results and Risk Summary

- ✓ **Medium Severity (3 findings)**
  - V-01: Content Security Policy (CSP) Header Not Set
  - V-02: Missing Anti-Clickjacking Header
  - V-03: Cross-Domain Misconfiguration
- ✓ **Low Severity (2 findings)**
  - V-04: Strict-Transport-Security (HSTS) Header Not Set
  - V-05: X-Content-Type-Options Header Missing
- ✓ **Informational (4 findings)**
  - V-06: Information Disclosure – Suspicious Comments
  - V-07: Retrieved from Cache
  - V-08: Modern Web Application
  - V-09: User Agent Fuzzer.

Remediation Priorities

1. **High Priority (Fix Immediately)**

    - Implement CSP headers (V-01) to reduce risk of XSS and data injection.
    - Add Anti-Clickjacking protections (V-02) to prevent UI redress attacks.
    - Review and tighten CORS configuration (V-03) to prevent cross-site abuse.
2. **Medium Priority (Next 2-3 sprints)**
    - Enforce HSTS headers (V-04) to mitigate SSL stripping risks.
    - Add X-Content-Type-Options headers (V-05) to prevent MIME-type spoofing.
3. **Low Priority (Ongoing monitoring)**
    - Review JavaScript comments and code disclosures (V-06).
    - Control caching (V-07) for sensitive endpoints.
    - Consider spider tuning for SPA crawling (V-08).
    - Mitigate user-agent fingerprinting issues (V-09).

Overall Risk Posture

The Swag Labs demo app demonstrates moderate security with weaknesses mainly in HTTP response headers and cross-domain configuration. While no critical vulnerabilities were found, remediation should be prioritized around CSP, anti-clickjacking, and CORS settings to reduce exposure to common attacks.