Vulnerability Report

Roman Janssens

Date of Assessment: 31/12/2024

Contents

Executive Summary

This report explains the results of penetration testing conducted on ShopMore. The purpose of this was to identify potential vulnerabilities and provide recommendations to mitigate risks. Key findings include:

- **Critical Vulnerabilities:** Anti-forgery token on login page not responsive, Unauthenticated Access to /admin/create
- **High-Risk Vulnerabilities:** Cross-Site Scripting (XSS) in product creation page
- **Medium and Low-Risk Vulnerabilities:** SQL exception due to missing username, clickjacking vulnerability on the Categories button

Recommendations:

1. Implement secure coding practices.
2. Validate and sanitize user inputs.
3. Enable proper error handling and suppress sensitive information disclosure.

This report is intended to help decision-makers understand the identified risks and prioritize actions to address them.

Scoping

Assets in Scope:

- **Web Application:** ShopMore
- **Endpoints:** Local IP address (10.0.2.100).
- **Infrastructure:** Virtual Machine environment, including snapshots for testing.
- **User Accounts:** Test accounts with different roles: user:user, admin:admin, and test:test.

Exclusions from Scope:

- **Third-party services and external APIs.**
- **Physical network devices (routers, firewalls).**
- **Denial of Service (DoS) or DDoS testing.**

Methodology

The testing approach adhered to industry-standard methodologies, including:

- **OWASP Top 10** and **WSTG Guidelines**.
- Manual and automated testing using tools such as Burp Suite
- Risk assessment using CVSS (Common Vulnerability Scoring System).

Testing Process

1. Reconnaissance and information gathering.
2. Vulnerability identification using automated scans and manual testing.
3. Exploitation attempts to assess vulnerability impact.
4. Reporting and documentation of findings.

Difficulty Levels of Risks

Severity Level	Description	CVSS Score Range
Critical	Immediate action required	9.0 - 10.0
High	High-priority vulnerabilities	7.0 - 8.9
Medium	Moderate impact vulnerabilities	4.0 - 6.9
Low	Minor issues with limited impact	0.1 - 3.9

Findings

Finding 1: Anti-Forgery Token not responding on Login Page

Severity / Risk Rating: Critical (CVSS score: 9.1)

Description: The login page does not implement the anti-forgery token, leaving it vulnerable to Cross-Site Request Forgery (CSRF) attacks.

Proof of Concept / Steps for Reproduction:

1. Intercept the login request using a proxy tool like burp suite .
2. Observe that no anti-forgery token is used in the form submission.
3. Craft a malicious request to exploit CSRF vulnerabilities.

Impact / Risk: An attacker can perform unauthorized actions on behalf of authenticated users, compromising integrity and confidentiality.

Recommendation / Mitigation:

1. Use CSRF protection.
2. Use server-side frameworks that support anti-forgery tokens.

Finding 2: Cross-Site Scripting (XSS) in Product Creation and "New Orders" Tab

Severity / Risk Rating: High (CVSS score: 8.3)

Description: The application is vulnerable to Cross-Site Scripting (XSS) when creating new products. User inputs are not properly checked, allowing malicious JavaScript to execute when accessing the "New Orders" tab. This happens after a product is created with an unfiltered input. The XSS is shown on the New Orders page.

Proof of Concept / Steps for Reproduction:

1. Navigate to the product creation form.
2. Enter </script><script>alert('hello')</script> in the text field product name.
3. Create the product.
4. Navigate to the "New Orders" tab.
5. Observe the JavaScript alert triggered by the XSS that was injected during product creation.

Impact / Risk: Attackers can inject and execute malicious JavaScript, potentially stealing session cookies and performing unauthorized actions. The XSS vulnerability could be exploited to target users interacting with the New Orders tab.

Recommendation / Mitigation:

1. **Server-Side Input Validation**: Validate and check all user inputs on the server before saving them to the database, particularly inputs from the product creation form.
2. **Client-Side Input Sanitization**: Use a client-side library like **DOMPurify** to filter inputs and prevent the execution of malicious scripts.
3. **Content Security Policy (CSP)**: Use a robust CSP to restrict the execution of untrusted JavaScript.

Finding 3: Unauthenticated Access to /admin/create (Access Control)

Severity / Risk Rating: Critical (CVSS score: 9.8)

Description: Unauthenticated users can access the /admin/create page. This means that attackers can make users with high privileges like admin.

Proof of Concept:

1. Navigate to the /admin/create page without logging in.
2. Fill in the user creation form and assign the "admin" role to the new user.
3. Submit the form and log in with the newly created admin account.

Impact / Risk: This vulnerability allows unauthenticated users to escalate their privileges, potentially leading to unauthorized access and data manipulation.

Recommendation / Mitigation:

1. Restrict access to the /admin/create page to authenticated users with admin privileges.
2. Use server-side role validation to prevent unauthorized role assignments.
3. Use role-based access control (RBAC) to enforce proper authorization across the application.

Finding 4: SQL Exception Due to Missing Username on Login page

Severity / Risk Rating: Medium (CVSS score: 6.4)

Description: When you leave the username field empty and enter a long password. 4001 characters exactly. The system throws a SqlException because the @username parameter is not supplied in the query. Same kind of error appears when filling in no username and password.

Proof of Concept:

1. Leave the username field empty.
2. Enter a password with 4001 characters.
3. Submit the form and observe the exception.

Impact / Risk: This leads to an unhandled exception, causing system instability and potentially opening the door for SQL injection or denial of service.

Recommendation / Mitigation:

1. Validate both username and password inputs before executing the query.
2. Add error handling to catch exceptions and return user-friendly messages.
3. Limit input lengths to avoid performance or security issues.

Finding 5: Clickjacking Vulnerability

Severity / Risk Rating: Medium (CVSS score: 6.5)

Description:
Clickjacking vulnerabilities were found on multiple interactive elements across the website, including the "Categories" button. Using Burp Suite's ClickBandit tool, an attacker could overlay a malicious iframe to hijack user clicks.

Impact / Risk:
Clickjacking can lead to unauthorized actions, such as modifying user settings doing un intended actions, posing a risk to user security and application integrity.

Recommendation / Mitigation:

1. Add **X-Frame-Options** headers (DENY or SAMEORIGIN) to prevent iframe embedding.
2. Implement a **Content Security Policy (CSP)** to restrict iframe usage:
    - “Content-Security-Policy: frame-ancestors 'self'; “
    - Test regularly for clickjacking vulnerabilities with tools like Burp Suite.

Recommendations

1. **Implement Secure Coding Practices:** Use input validation, statements, and follow OWASP secure coding guidelines.
2. **Utilize Modern Security Libraries:** Libraries for sanitization and validation, such as OWASP ESAPI or DOMPurify.
3. **Regular Security Testing:** Schedule vulnerability assessments to identify new threats.
4. **Training:** Provide training to developers on secure development practices.

Appendix

Tools Used: Burp Suite, OWASP ZAP

References
OWASP Top 10: **https://owasp.org/www-project-top-ten/**
WSTG: **https://owasp.org/www-project-web-security-testing-guide/**
CVSS Calculator: **https://www.first.org/cvss/calculator/3.1**