CodeShuriken

Security Assessment Report

Repository: https://github.com/SarthakShieldersoft/TestVWA.git

Generated: 2025-07-31T07:23:32.725Z

Report ID: 20250731072332_fee07cb7_SBOM_1ff28aad

Table of Contents

Preface

About This Report

This automated security assessment report has been generated to provide comprehensive insights into the security posture of your software repository. The analysis includes dependency vulnerabilities, code quality metrics, and actionable security recommendations.

Methodology

Our security assessment employs multiple scanning techniques including:

- *Static code analysis for vulnerability detection*
- *Dependency vulnerability scanning against known CVE databases*
- *Software Bill of Materials (SBOM) generation and analysis*
- *Code quality and security best practices evaluation*

Scope and Limitations

This report covers static analysis findings and known vulnerabilities in dependencies. It does not include dynamic testing, manual code review findings, or infrastructure security assessments. The analysis is based on the repository state at the time of scanning.

How to Use This Report

Review the Executive Summary for high-level findings, then proceed to detailed sections. Prioritize critical and high-severity vulnerabilities for immediate remediation. Use the recommendations section to improve your overall security posture.

Executive Summary

This report was prepared by automated security scanning to review aspects of the security and integrity of the repository **https://github.com/SarthakShieldersoft/TestVWA.git**.

This report identifies potential security weaknesses and vulnerabilities found through static code review and searches of public vulnerability sources. The analysis focused particularly on dependency vulnerabilities that could be exploited to alter system behavior, access critical data, or conduct denial of service attacks.

Scan Type:	sbom	**Status:**	
Repository:	https://github.com/SarthakShieldersoft/TestVWA.git		
Branch:	main	**Completed:**	2025-07-31T07:23:47.430Z

Key Findings Summary:

- Repository contains 1 files with 1 successfully processed
- Scan completion rate: 100%
- **Total vulnerabilities found: 0**
  - Critical: 0
  - High: 0
  - Medium: 0
  - Low: 0
- Primary languages identified: CSS (14.26%), XML (11.93%), Java (72.21%), PowerShell (1.61%)

Last update:	2025-07-31T07:23:32.725Z	**Status:**	
Version:	1.0	**Repository:**	https://github.com/SarthakShieldersoft/TestVWA.git

1 Repository Information

1.1 Language Distribution

Language	Percentage
CSS	14.26%
XML	11.93%
Java	72.21%
PowerShell	1.61%

1.2 File Types Analysis

File Extension	Count
.md	1
.bat	2
.css	1
.jsp	7
.ps1	1
.xml	2
.java	18
.properties	3
no_extension	1

2 Security Analysis & Vulnerability Assessment

2.0 pom.xml Analysis

File: pom.xml

Type: SBOM

Security Analysis Report for `pom.xml`

This report analyzes the provided `pom.xml` file for security vulnerabilities and outdated dependencies. The analysis focuses on known vulnerabilities and outdated versions of commonly exploited libraries.

Vulnerabilities and Outdated Packages:

The `pom.xml` file reveals several critical security vulnerabilities and outdated packages:

1. Spring Framework (Outdated):

Dependency:: `org.springframework:spring-webmvc`, `org.springframework:spring-web`, `org.springframework:spring-context`, `org.springframework:spring-orm`

Version:: 4.3.9.RELEASE

Severity:: High

Vulnerability:: Spring Framework 4.3.9 is extremely outdated. Numerous critical vulnerabilities have been patched in later versions. These vulnerabilities could lead to Remote Code Execution (RCE), information disclosure, and other serious attacks.

Remediation:: Upgrade to the latest stable Spring Framework version (check the official Spring website for the latest). Consider using a supported Long Term Support (LTS) version.

2. Spring Security (Vulnerable):

Dependency:: `org.springframework.security:spring-security-web`, `org.springframework.security:spring-security-config`

Version:: 4.2.3.RELEASE

Severity:: Critical

Vulnerability:: Similar to the Spring Framework, this version is severely outdated and contains numerous critical vulnerabilities that could be exploited for RCE, authentication bypass, and other serious security breaches.

Remediation:: Upgrade to the latest stable Spring Security version.

3. Jackson Databind (Vulnerable to Deserialization):

Dependency:: `com.fasterxml.jackson.core:jackson-databind`

Version:: 2.8.11

Severity:: High

Vulnerability:: Older versions of Jackson Databind are vulnerable to deserialization attacks, allowing attackers to execute arbitrary code by crafting malicious JSON payloads.

Remediation:: Upgrade to the latest stable version of Jackson Databind. Consider using a more secure deserialization mechanism if possible.

4. Log4j (Log4Shell Vulnerability):

Dependency:: `log4j:log4j`

Version:: 1.2.17

Severity:: Critical

Vulnerability:: This is the infamous Log4Shell vulnerability (CVE-2021-44228). This vulnerability allows for Remote Code Execution (RCE) via specially crafted log messages.

Remediation:: **Immediately** remove this dependency and replace it with Log4j 2.x or a more modern logging framework like Logback or SLF4j.

5. Apache Commons Collections (Vulnerable):

Dependency:: `org.apache.commons:commons-collections4`

Version:: 4.0

Severity:: Medium

Vulnerability:: Older versions of Commons Collections have been implicated in deserialization attacks. While the impact might be less severe than Jackson Databind, it's still a potential vector for exploitation.

Remediation:: Upgrade to the latest stable version.

6. MySQL Connector (Outdated):

Dependency:: `mysql:mysql-connector-java`

Version:: 5.1.46

Severity:: Medium

Vulnerability:: Outdated database connectors may contain security vulnerabilities that have been addressed in later versions.

Remediation:: Upgrade to the latest stable version.

7. Other Outdated/Vulnerable Dependencies:

Hibernate:: Upgrade to a recent stable version.

Commons FileUpload:: Upgrade to a recent stable version.

xercesImpl:: Vulnerable to XXE (XML External Entities) attacks. Consider using a safer XML parser or disabling external entity processing.

jjwt:: Upgrade to a recent stable version (JWT implementation vulnerabilities are common).

Spring Boot Starter (1.5.9.RELEASE):: This is quite old and likely inherits vulnerabilities from its component dependencies. Upgrade to a more recent version.

8. Missing Dependency Management:

The `pom.xml` lacks explicit dependency management for many components. This makes it difficult to enforce consistent versions across the project and can lead to unexpected conflicts or security risks if transitive dependencies introduce conflicting or vulnerable versions. Adding a `<dependencyManagement>` section would greatly improve the management of versions.

Recommendations:

> **Remediation:** 1. **Immediate Remediation:** Address the Log4j vulnerability by removing the dependency and replacing it with a secure alternative.

> **Remediation:** 2. **Upgrade Dependencies:** Update *all* outdated dependencies to their latest stable versions, paying close attention to Spring Framework, Spring Security, Jackson Databind, and database connectors.

> **Remediation:** 3. **Dependency Management:** Implement proper dependency management in your `pom.xml` to control versions and avoid conflicts.

> **Remediation:** 4. **Regular Security Scanning:** Integrate automated security scanning into your development pipeline to regularly check for new vulnerabilities in your dependencies.

> **Remediation:** 5. **Secure Coding Practices:** Implement secure coding practices to minimize the risk of vulnerabilities in your application code.

> **Remediation:** 6. **Vulnerability Database:** Regularly check the CVE (Common Vulnerabilities and Exposures) database for known vulnerabilities in your used libraries.

This report highlights the most critical issues. A more comprehensive analysis would involve a full security scan of the application itself (using tools like OWASP ZAP) to identify further vulnerabilities. Simply updating dependencies doesn't guarantee complete security.

3 Code Metrics

Total Lines of Code:	3791
Files Processed:	1/1
Completion:	100%

4 Recommendations

Immediate Actions Required:

- Review and address all critical and high severity vulnerabilities
- Update all outdated dependencies to their latest stable versions
- Implement proper dependency management practices

Long-term Security Improvements:

- Integrate automated security scanning into CI/CD pipeline
- Regular security code reviews
- Monitor security advisories for used dependencies
- Implement secure coding practices

5 Summary

Assessment Results Overview

Scan Type: sbom | **Status:** | **Files: 1/1** | **Progress: 100%**

Total Vulnerabilities: 0

Vulnerability Breakdown

Critical: 0 | **High: 0** | **Medium: 0** | **Low: 0**

Key Statistics

Repository URL:	https://github.com/SarthakShieldersoft/TestVWA.git
Branch Analyzed:	main
Scan Started:	2025-07-31T07:23:32.725Z
Scan Completed:	2025-07-31T07:23:47.430Z
Total Lines of Code:	3791

Primary Languages

CSS: 14.26% | XML: 11.93% | Java: 72.21% | PowerShell: 1.61%

Next Steps

1. **Immediate:** Address any critical or high-severity vulnerabilities identified
2. **Short-term:** Review and update outdated dependencies
3. **Long-term:** Implement continuous security monitoring and automated scanning
4. **Process:** Integrate security practices into development workflow

Report generated on 2025-07-31T07:23:32.725Z

Automated Security Assessment Tool