DevSecOps SAST Report: OWASP Juice Shop Vulnerable App

**Repository:** https://github.com/AdeolaMercy/devsecops\_proj  
**Branch:** master  
**Pipeline:** GitHub Actions + Semgrep Cloud

# 1. Executive Summary

This report summarizes the security posture of the forked OWASP Juice Shop repository. It focuses on both the CI/CD pipeline setup and the security findings from automated SAST scans using Semgrep Cloud.

Key Highlights:  
- Automated SAST integrated via GitHub Actions  
- Critical security findings detected, requiring immediate attention  
- High and medium severity issues present, affecting application security best practices

# 2. Project Overview

**Repository:** https://github.com/AdeolaMercy/devsecops\_proj  
**Branch:** master  
**SAST Tool:** Semgrep Cloud  
**Pipeline Triggers:** push, pull request, workflow dispatch  
**Scan Scope:** Full JavaScript/Node.js codebase excluding node\_modules

# 3.1 Pipeline Setup

GitHub Actions Workflow (.github/workflows/semgrep\_sast.yml):

name: Semgrep SAST Scan  
  
on:  
 push:  
 branches: [ "main" ]  
 pull\_request:  
 branches: [ "main" ]  
 workflow\_dispatch:  
  
jobs:  
 semgrep:  
 runs-on: ubuntu-latest  
  
 steps:  
 - uses: actions/checkout@v4  
  
 - name: Set up Python  
 uses: actions/setup-python@v5  
 with:  
 python-version: "3.x"  
  
 - name: Run Semgrep Cloud Scan  
 env:  
 SEMGREP\_APP\_TOKEN: ${{ secrets.SEMGREP\_APP\_TOKEN }}  
 run: semgrep ci

# 3.2 SAST Workflow Diagram

A diagram of a website

AI-generated content may be incorrect.

# 4. Security Findings

## Critical Findings

|  |  |  |  |
| --- | --- | --- | --- |
| Finding | Description | Impact | Recommendation |
| express-mongo-nosqli | Potential NoSQL injection via Express + MongoDB | Remote code/data compromise | Sanitize all inputs for MongoDB and Sequelize queries |
| sequelize-express | SQL injection risk in Sequelize queries through Express endpoints | Database compromise |
| vm-express | Unsafe usage of Node.js vm module in Express context | Arbitrary code execution | Replace eval() and vm usage with safe alternatives |
| eval-express | Usage of eval() on user-controlled input | Arbitrary code execution |

## High Findings

|  |  |  |  |
| --- | --- | --- | --- |
| Finding | Description | Impact | Recommendation |
| express-sequelize-injection | Potential SQL injection in Sequelize ORM | Database compromise | Sanitize all inputs and Sequelize queries |
| detected-jwt-token | JWT tokens hardcoded or exposed in code | Credential leakage | Remove hardcoded JWT secrets; use environment variables existing tokens |
| detected-generic-secret | Generic secrets detected in code or config | Sensitive data exposure | Rotate exposed secrets and invalidate. Store secrets securely |

## Medium Findings

|  |  |  |  |
| --- | --- | --- | --- |
| Finding | Description | Impact | Recommendation |
| session-fixation | Session ID not rotated after authentication | Session hijacking | Enable secure session handling and rotate session IDs |
| cookies-default-express | Default cookie settings may allow theft | Client-side risk | Configure cookies with HttpOnly and Secure flags |
| crypto-weak-algorithm | Weak or outdated crypto algorithm detected | Data compromise | Replace weak crypto algorithms with modern standards (e.g., AES-256) |
| detected-private-key | Private keys exposed in repository | Credential leakage / impersonation | Store private keys securely |

# 5. Risk Analysis

- Critical Issues: Immediate exploitation risk (code injection, NoSQL/SQL injection).  
- High Issues: Potential credential or sensitive data exposure.  
- Medium Issues: Moderate security misconfigurations that could be chained by attackers.  
  
Overall Risk Level: High. It is recommended that all critical and high findings be remediated, and mitigations can be put in place for the medium findings, if they can’t be remediated.

# 6. Recommendations

## 6.1 Code Remediation

Implement code remediation as set out in the findings table above

## 6.2 Pipeline Enhancements

- Fail builds on critical/high findings with --error flag  
- Upload JSON scan artifacts for offline review  
- Schedule nightly automated scans for continuous monitoring

## 6.3 Policy & Governance

- Enforce pre-merge SAST scans for pull requests  
- Integrate Semgrep Cloud dashboards into developer workflow  
- Maintain rule whitelist/ignore lists carefully to avoid false negatives

# 7. Next Steps

1. Patch all critical vulnerabilities immediately  
2. Remediate high and medium severity findings  
3. Expand CI/CD security checks to include:  
- Dependency scanning  
- Container scanning (if using Docker)  
- DAST scans using OWASP ZAP after deployment  
4. Train developers on secure coding practices aligned with OWASP Top 10

# 8. References

- Semgrep Documentation: https://semgrep.dev/docs/  
- OWASP Juice Shop: https://owasp.org/www-project-juice-shop/  
- OWASP Top 10 Vulnerabilities: https://owasp.org/Top10/