AI-Generated Security Policies Summary

Generated on 2025-11-13 16:18:23

This document summarizes AI-generated security policies derived from SAST, SCA, and DAST analyses. Each policy includes its metadata, executive summary, key risks, and security controls for remediation.

Dynamic Application Security Testing (DAST) Policy

Policy ID: POL-DAST-2025-001
Status: Active
Created: 2025-11-05T00:00:00Z
Last Updated: 2025-11-05T00:00:00Z
Author: Security Policy Generator

Executive Summary

Current runtime scanning has uncovered three medium■severity findings that expose the organization to data leakage, content spoofing, and speculative execution attacks. Immediate remediation and hardened scanning processes are required to reduce the attack surface and align with NIST CSF and ISO 27001 controls.

Purpose

To identify and remediate runtime security vulnerabilities through dynamic testing

Description

This policy defines requirements for DAST implementation, runtime security monitoring, and production security hardening

Applicability

All web applications, APIs, and internet■facing services

Enforcement

Pre■production DAST scans required before deployment to production

Exceptions

Internal■only applications may use reduced scan frequency with documented approval

Objectives

- Identify runtime vulnerabilities before code reaches production
- Remediate identified findings within defined timeframes
- Maintain continuous compliance with NIST CSF and ISO 27001 security controls

Risk Assessment

Overall Risk Level: Medium
Critical Count: 0
High Count: 0
Medium Count: 3
Low Count: 0
Business Impact: Exploitation could lead to information disclosure, cross■site scripting, and reduced confidence in service integrity, potentially impacting customer trust and regulatory compliance.
Likelihood: Medium

Security Controls

SC-DAST-001: Security Headers Implementation
Enforce mandatory HTTP security headers to protect against content sniffing and click■jacking.

- Update web server configuration to include X-Content-Type-Options: nosniff
- Add X-Frame-Options: SAMEORIGIN and Content■Security■Policy headers
- Automate header verification in CI/CD pipeline using a linting tool

SC-DAST-002: Cache Control and Content Hardening
Prevent sensitive or dynamic content from being stored or cached unintentionally.

- Set Cache-Control: no-store, no-cache for all dynamic endpoints
- Review and update response headers for static assets to include appropriate max■age values
- Integrate automated DAST rule to flag cacheable responses lacking proper directives

SC-DAST-003: Spectre Mitigation and Site Isolation
Apply mitigations to reduce risk from speculative execution side■channel attacks.

- Enable site■isolation features in modern browsers (e.g., Chrome's Site Isolation flag) for all public pages
- Patch underlying OS and runtime libraries to latest versions containing Spectre mitigations
- Validate mitigation effectiveness with vendor■provided test suites after each release

Remediation Actions

P2 - Add Missing X-Content-Type-Options Header
Owner: Web Operations Team | Timeline: Within 14 days

- All web front■ends
- /api/*

Success Criteria: All HTTP responses contain X-Content-Type-Options: nosniff header verified by automated scan

P2 - Restrict Storable and Cacheable Content
Owner: Application Development Team | Timeline: Within 21 days

- /static/*
- /api/public/*

Success Criteria: No DAST findings for cacheable content without Cache-Control or Pragma headers

P2 - Mitigate Spectre via Site Isolation
Owner: Infrastructure Security Team | Timeline: Within 30 days

- All public web applications

Success Criteria: Site Isolation enabled in browsers for all domains and OS patches applied; spectre test suite passes

Compliance Mapping

NIST CSF Categories:

- PR.IP-1
- PR.DS-2
- DE.CM-1

ISO 27001 Controls:

- A.9.1.1
- A.12.6.1
- A.13.1.1
- A.14.2.9

Monitoring Requirements

- Continuous runtime security monitoring via WAF logs
- Weekly automated DAST scans on staging and production environments

- Monthly review of security■header compliance reports
- Quarterly update of Spectre mitigation status

Review Schedule

Monthly policy review and quarterly full penetration testing

Static Application Security Testing (SAST) Policy

Policy ID: POL-SAST-2025-001
Status: Active
Created: 2025-11-05T00:00:00
Last Updated: 2025-11-05T00:00:00
Author: Security Policy Generator

Executive Summary

The organization adopts a risk■based SAST program aligned with NIST CSF and ISO 27001 to continuously detect code■level weaknesses. Current findings are limited to a single low■severity issue, indicating an overall healthy code security posture while highlighting the need for ongoing review of unknown findings.

Purpose

To establish standards for identifying and remediating source code vulnerabilities through static analysis.

Description

This policy defines requirements for SAST implementation, vulnerability management, and secure code development practices across the organization.

Applicability

All development teams and applications that contain custom source code, including micro■services, libraries, and scripts.

Enforcement

Mandatory SAST scans must be integrated into every CI/CD pipeline with defined quality gates; builds failing the gate are blocked from promotion.

Exceptions

Legacy applications may request temporary exemptions through the Security Review Board, subject to a documented risk acceptance and remediation timeline.

Objectives

- Integrate automated static analysis into every code commit and build process.
- Ensure identified vulnerabilities are triaged, prioritized, and remediated within defined timeframes.
- Maintain compliance with NIST CSF Protect function and ISO 27001 A.14 System Development lifecycle controls.

Risk Assessment

Overall Risk Level: Low
Critical Count: 0
High Count: 0
Medium Count: 0

Low Count: 1
Business Impact: A low■severity, unknown vulnerability could lead to minor information disclosure if left unaddressed, but the impact on core business operations is limited.
Likelihood: Low

Security Controls

SC-001: Input Validation Framework
Implement a centralized input validation library that enforces whitelist■based checks for all external data.

- Select a vetted validation library (e.g., OWASP ESAPI).
- Integrate the library into all new code modules.
- Refactor existing modules to use the library within 90 days.
- Document validation rules per data source in the code repository.

SC-002: Automated SAST Integration
Configure CI/CD pipelines to run approved SAST tools on every pull request and nightly builds.

- Deploy SAST tool (e.g., SonarQube, Checkmarx) on the build server.
- Create pipeline step that fails the build if findings exceed the defined threshold.
- Generate a SARIF report and store it in the artifact repository.
- Notify the development team via Slack/Email on each scan result.

SC-003: Vulnerability Triage and Response Process
Establish a formal process to triage SAST findings, assign remediation owners, and track closure.

- Log each finding in the ticketing system with severity tags.
- Assign tickets to the responsible component owner within 1 business day.
- Review tickets in weekly security stand■ups.
- Close tickets only after successful re■scan and peer review.

Remediation Actions

P3 - Review and fix unknown security report analysis finding
Owner: Backend Development Team | Timeline: Within 14 days

- Application code base (global scope)
- CI/CD pipeline configuration

Success Criteria: The unknown finding is either resolved or documented with a risk acceptance; subsequent SAST scan shows no recurrence of the same issue.

Compliance Mapping

NIST CSF Categories:

- PR.DS-5
- PR.IP-1
- DE.CM-4

ISO 27001 Controls:

- A.14.2.1
- A.14.2.5
- A.12.6.1

Monitoring Requirements

- Automated SAST scans on every code commit (CI) and nightly full scans (CD).
- Weekly reporting of new findings to the Security Steering Committee.
- Quarterly security code reviews by peer developers and the Application Security Team.

Review Schedule

Quarterly policy review and annual audit

Software Composition Analysis (SCA) Policy

Policy ID: POL-SCA-2025-001
Status: Active
Created: 2025-11-05T00:00:00Z
Last Updated: 2025-11-05T00:00:00Z
Author: Security Policy Generator

Executive Summary

Our current software supply chain contains 33 known vulnerable dependencies, including 16 high■severity findings. Immediate remediation and continuous monitoring are required to reduce exposure and comply with regulatory and contractual obligations.

Purpose

To manage third■party dependency security risks and ensure supply■chain integrity for all software assets.

Description

This policy establishes mandatory requirements for automated dependency scanning, vulnerability triage, remediation, and the creation and maintenance of a Software Bill of Materials (SBOM).

Applicability

All applications, services, containers, and infrastructure■as■code artifacts that incorporate third■party libraries, packages, or open■source components.

Enforcement

Automated SCA scans must run on every code commit and build pipeline; builds containing high■severity (CVSS ≥7.0) vulnerable dependencies are blocked until remediation or approved exception.

Exceptions

Critical business■critical dependencies with no viable alternatives may be approved by the Change Advisory Board (CAB) provided compensating controls such as runtime monitoring and additional penetration testing are implemented.

Objectives

- Establish a definitive, version■controlled inventory of all third■party components.
- Detect and remediate high■severity vulnerabilities within 7 days of discovery.
- Maintain an up■to■date SBOM and ensure continuous compliance with licensing and security policies.

Risk Assessment

Overall Risk Level: High
Critical Count: 0
High Count: 16
Medium Count: 14

Low Count: 3
Business Impact: Exploitation of high■severity vulnerable libraries could lead to data breach, service disruption, or loss of intellectual property, impacting customer trust and regulatory compliance.
Likelihood: High

Security Controls

SC-SCA-001: Dependency Inventory Management

Maintain a centralized, version■controlled Software Bill of Materials (SBOM) for every application repository.

- Integrate sbom■generation tools (e.g., CycloneDX, Syft) into CI pipelines.
- Store SBOM files in a read■only artifact repository linked to the source code.
- Review and reconcile SBOM against approved vendor list quarterly.

SC-SCA-002: Automated Vulnerability Scanning

Run SCA scans on every pull request and nightly builds; block merges when high■severity findings are present.

- Configure SCA tool (e.g., Snyk, Dependabot, OWASP Dependency■Check) to fail builds on CVSS ≥7.0.
- Publish scan results to a centralized dashboard for visibility.
- Enforce policy as code using GitHub Actions or Azure Pipelines gate.

SC-SCA-003: Continuous Threat Intelligence Integration

Subscribe to CVE feeds and vulnerability databases to automatically update scanning signatures.

- Enable daily sync with NVD, GitHub Advisory Database, and vendor security bulletins.
- Map incoming CVEs to affected components in the SBOM.
- Trigger alerts in the Security Operations Center (SOC) when new high■severity CVEs are identified.

SC-SCA-004: Vulnerability Remediation Workflow

Standardize a ticket■based process for triaging, fixing, and verifying vulnerable dependencies.

- Create a remediation ticket automatically for each high■severity finding.
- Assign tickets to the owning development team with a 7■day SLA.
- Require peer■review and automated regression testing before merge.

SC-SCA-005: Rollback and Patch Deployment

Ensure rapid rollback or patch deployment mechanisms are in place for vulnerable components.

- Maintain version■controlled rollback scripts in the repository.
- Test rollback procedures in a staging environment quarterly.
- Document recovery steps in the incident response playbook.

Remediation Actions

P0 - Patch all high■severity vulnerable dependencies

Owner: DevOps Team | Timeline: Immediate (0■7 days)

- lodash@4.17.20
- express@4.16.0
- moment@2.24.0
- axios@0.19.2
- react@16.8.6

Success Criteria: All high■severity CVEs resolved; no build failures due to blocked dependencies.

P1 - Upgrade medium■severity dependencies with known exploits

Owner: Application Development Leads | Timeline: Within 30 days

- chalk@2.4.2
- debug@3.2.7
- xml2js@0.4.19

Success Criteria: Medium■severity CVEs reduced by 80%; regression tests pass.

P2 - Review low■severity and license■risk dependencies

Owner: Legal & Compliance | Timeline: Within 60 days

- left-pad@1.3.0
- underscore@1.9.1

Success Criteria: All license conflicts resolved; low■severity CVEs documented with risk acceptance.

Compliance Mapping

NIST CSF Categories:

- ID.AM-2
- PR.IP-12
- DE.CM-8
- RS.RP-1
- RC.RP-1

ISO 27001 Controls:

- A.15.1.1
- A.15.1.2
- A.12.6.1
- A.12.4.1
- A.16.1.2
- A.17.1.2

Monitoring Requirements

- Daily automated SCA scans on all repositories.
- Continuous CVE feed ingestion and correlation with SBOM.
- Weekly license compliance reports.
- Monthly KPI dashboard (mean time to remediate, scan pass rate).

Review Schedule

Monthly dependency review meetings; quarterly formal audit of SCA process and compliance.