AI-Powered Security Policy Report

Generated: 2025-11-07 15:08:03

Total Vulnerabilities Scanned: 26
• SAST: 8
• SCA: 10
• DAST: 8

LLM Models Used:

• LLaMA 3.3 70B (Groq) - SAST/SCA
• LLaMA 3.1 8B Instant (Groq) - DAST

Policy #1: SAST

Title: Node Sqli
Severity: HIGH
LLM: LLaMA 3.3 70B

POLICY IDENTIFIER
SP-2024-001: SQL Injection Prevention Policy

RISK STATEMENT
The organization is at risk of a SQL injection attack, which could compromise the confidentiality, integrity, and availability of sensitive user data. If exploited, this vulnerability could lead to unauthorized access, modification, or deletion of data, resulting in significant business impact, including reputational damage, financial loss, and regulatory non-compliance. The affected systems include the user management application, and the impacted users are all individuals with access to the application.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), DE.CM-8 (Vulnerability Management)
- ISO 27001: A.8.8 (Management of technical vulnerabilities), A.5.37 (Documented operating procedures), A.5.26 (Response to information security incidents), A.8.16 (Monitoring activities), A.8.26 (Application security requirements)
- Industry standards: OWASP Top 10 (A03:2021-Injection), CWE-89 (Improper Neutralization of Special Elements used in an SQL Command)

POLICY REQUIREMENTS
To prevent SQL injection attacks, the following security controls must be implemented:
- All user input must be validated and sanitized using parameterized queries or prepared statements.
- Regular security testing and code reviews must be performed to identify and remediate vulnerabilities.

- Success criteria include:
- All user input is validated and sanitized.
- No SQL injection vulnerabilities are detected during security testing.
- Code reviews are performed regularly to ensure compliance with this policy.

REMEDIATION PLAN
To remediate the identified vulnerability, the following actions must be taken:
- Technical actions: Review and fix the vulnerable code in src/controllers/UserController.js (line 45) by implementing parameterized queries or prepared statements.
- Responsible party: Security Lead
- Timeline: 1 week
- Verification steps: Code review, re-scan, and penetration testing to ensure the vulnerability is fully remediated.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures must be implemented:
- Regular security testing and code reviews.
- Logging and alerting requirements: All security-related events must be logged and alerts must be generated in case of suspicious activity.
- Continuous monitoring strategies: Regularly review and update this policy to ensure it remains effective in preventing SQL injection attacks.

Policy #2: SAST

Title: Var In Href
Severity: MEDIUM
LLM: LLaMA 3.3 70B

POLICY IDENTIFIER
SP-2024-001: Cross-Site Scripting (XSS) Protection Policy

RISK STATEMENT
The organization is at risk of Cross-Site Scripting (XSS) attacks due to the use of template variables in anchor tags with the 'href' attribute. This vulnerability could allow malicious actors to inject malicious code, potentially leading to unauthorized access, data theft, or system compromise. The affected systems include the web application, and the impacted users are all individuals accessing the application. If exploited, this vulnerability could result in significant reputational damage, financial loss, and compromised customer trust.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), DE.CM-8 (Vulnerability Management)
- ISO 27001: A.8.23 (Web filtering), A.8.8 (Management of technical vulnerabilities), A.5.7 (Threat intelligence)
- Industry standards: OWASP Top 10 (A07:2021 - Cross-Site Scripting), CWE-79 (Improper Neutralization of Input During Web Page Generation)

POLICY REQUIREMENTS
To mitigate the risk of XSS attacks, the following security controls must be implemented:
- All user-input data must be validated and sanitized before being used in web page generation.
- Template variables used in anchor tags must be properly encoded to prevent code injection.
- Success criteria: All web application code must pass a security review and vulnerability scan without detecting any XSS vulnerabilities.
- Validation methods: Code review, vulnerability scanning, and penetration testing will be used to verify compliance with this policy.

REMEDICATION PLAN
To remediate the identified vulnerability:
- Specific technical actions: Review and fix the template variable usage in the 'profile.mustache' file (line 23) to ensure proper encoding and validation of user-input data.
- Responsible party: Dev Lead
- Timeline: 2 weeks
- Verification steps: Code review, re-scan with a vulnerability scanner, and penetration testing will be performed to verify the fix.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future:
- Regular code reviews and vulnerability scans will be performed to identify potential XSS vulnerabilities.
- Logging and alerting requirements: All security-related logs will be monitored, and alerts will be triggered in case of suspected XSS attacks.
- Continuous monitoring strategies: The organization will implement a web application firewall (WAF) and regularly update its vulnerability management program to stay informed about new XSS vulnerabilities and attack vectors.

Policy #3: SAST

Title: Path Join Resolve Traversal
Severity: HIGH
LLM: LLaMA 3.3 70B

POLICY IDENTIFIER
SP-2024-001: Path Traversal Protection Policy

RISK STATEMENT
The Path Traversal vulnerability poses a significant risk to our organization's data security. If exploited, an attacker could potentially access sensitive files outside the intended directory, leading to unauthorized data disclosure, tampering, or destruction. This could result in reputational damage, financial losses, and legal liabilities. The affected systems include our web application servers, and the impacted data includes sensitive user information and business-critical files.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), DE.CM-8 (Vulnerability Management)
- ISO 27001: A.8.28 (Secure Coding), A.8.29 (Security Testing in Development and Acceptance)
- Industry Standards: OWASP A5:2021 (Security Misconfiguration), CWE-22 (Improper Limitation of a Pathname to a Restricted Directory)

POLICY REQUIREMENTS
To mitigate the Path Traversal vulnerability, the following security controls must be implemented:
- Input validation and sanitization for all user-provided file paths
- Canonicalization of file paths to prevent directory traversal
- Implementation of a web application firewall (WAF) to detect and prevent path traversal attacks
- Regular security testing and code reviews to identify and remediate similar vulnerabilities
Success criteria include:
- No detectable path traversal vulnerabilities in code reviews and security scans
- Successful implementation of input validation and sanitization mechanisms
Validation methods include:
- Code reviews
- Penetration testing
- Automated security scanning

REMEDIATION PLAN
To remediate the identified vulnerability, the following technical actions are required:
- Review and fix the vulnerable code in src/routes/files.js (line 67)
- Implement input validation and sanitization for all user-provided file paths
- Configure the WAF to detect and prevent path traversal attacks
The responsible party for remediation is the Security Lead, given the High severity of the vulnerability. The timeline for remediation is 1 week. Verification steps include:
- Code review
- Re-scanning with automated security tools
- Penetration testing

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures will be implemented:
- Regular code reviews and security testing
- Automated security scanning and logging

- Continuous monitoring of web application logs for suspicious activity
Logging and alerting requirements include:
- Logging all access attempts to sensitive files and directories
- Alerting the Security Lead in case of suspected path traversal attacks
Continuous monitoring strategies include:
- Regularly reviewing web application logs for suspicious activity
- Performing periodic security testing and code reviews to identify and remediate vulnerabilities.

Policy #4: SAST

Title: Express Cookie Session No Httponly
Severity: MEDIUM
LLM: LLaMA 3.3 70B

POLICY IDENTIFIER
SP-2024-001: Secure Session Management Policy

RISK STATEMENT
The absence of the 'httpOnly' flag in cookie sessions poses a significant risk to our organization's data security. If exploited, this vulnerability could lead to session hijacking via cross-site scripting (XSS) attacks, allowing unauthorized access to sensitive information. This could result in financial loss, reputational damage, and compromised customer data. The affected systems include our web applications, and the impacted users are our customers and employees who access these applications.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), DE.CM-8 (Vulnerability Management)
- ISO 27001 Annex A controls: A.8.1 (User endpoint devices), A.8.23 (Web filtering), A.8.29 (Security testing in development and acceptance)
- Industry standards: OWASP A2:2017 (Broken Authentication), CWE-1004 (Sensitive Cookie Without 'HttpOnly' Flag)

POLICY REQUIREMENTS
To mitigate this risk, the following security controls must be implemented:
- All cookie sessions must be configured with the 'httpOnly' flag to prevent JavaScript access.
- Regular security testing and code reviews must be performed to identify and remediate similar vulnerabilities.
- Success criteria: All cookie sessions are configured with the 'httpOnly' flag, and no similar vulnerabilities are detected during security testing.
- Validation methods: Code reviews, security scans, and penetration testing.

REMEDIATION PLAN
To remediate this vulnerability, the following technical actions are required:
- Review and update the session.js file to include the 'httpOnly' flag in cookie sessions.
- Responsible party: Dev Lead
- Timeline: 2 weeks
- Verification steps: Code review, security scan, and penetration testing to ensure the 'httpOnly' flag is correctly implemented and no similar vulnerabilities exist.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures must be implemented:
- Regular security testing and code reviews to identify vulnerabilities in session management.
- Logging and alerting requirements: All security-related logs must be monitored, and alerts must be triggered in case of suspicious activity.
- Continuous monitoring strategies: Regularly review and update security policies, perform security awareness training, and conduct penetration testing to ensure the security of our web applications.

Policy #5: SAST

Title: Hardcoded Secret
Severity: HIGH
LLM: LLaMA 3.3 70B

POLICY IDENTIFIER
SP-2024-001: Hardcoded Secret Policy

RISK STATEMENT
The presence of hardcoded secrets in our codebase poses a significant risk to our organization's security posture. If exploited, an attacker could gain unauthorized access to sensitive information, compromising the confidentiality, integrity, and availability of our systems and data. This vulnerability affects our authentication mechanisms, potentially impacting all users and systems that rely on these credentials.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), PR.IP-1 (Identity and Access Control)
- ISO 27001 Annex A controls: A.8.28 (Secure coding), A.8.12 (Data leakage prevention), A.8.27 (Secure system architecture and engineering principles), A.8.31 (Separation of development, test and production environments), A.8.24 (Use of cryptography)
- Industry standards: OWASP, CWE-798 (Use of Hard-coded Credentials)

POLICY REQUIREMENTS
To mitigate the risk of hardcoded secrets, the following security controls must be implemented:
- All secrets must be stored in environment variables or a secure secret management system.
- Code reviews must be performed regularly to detect and remove hardcoded secrets.
- Secure coding principles must be applied to software development, including the use of secure coding guidelines and secure coding practices.
- Success criteria: All hardcoded secrets must be removed from the codebase, and a secure secret management system must be implemented.
- Validation methods: Regular code reviews, automated scanning, and penetration testing.

REMEDIATION PLAN
To remediate this vulnerability, the following technical actions are required:
- Remove the hardcoded secret from src/config/auth.js (line 8).
- Implement a secure secret management system to store and manage secrets.
- Perform a code review to detect and remove any other hardcoded secrets.
- Responsible party: Security Lead
- Timeline: 1 week
- Verification steps: Code review, automated scanning, and penetration testing.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures must be implemented:
- Regular code reviews to detect hardcoded secrets.
- Automated scanning to detect vulnerabilities.
- Continuous monitoring of logs and alerts to detect suspicious activity.
- Logging and alerting requirements: All security-related events must be logged and alerted to the Security Lead.
- Continuous monitoring strategies: Regular security audits, penetration testing, and vulnerability scanning.

Policy #6: SCA

Title: lodash - CWE-1321
Severity: HIGH
LLM: LLaMA 3.3 70B

POLICY IDENTIFIER
SP-2024-001: Lodash Prototype Pollution Protection Policy

RISK STATEMENT
The organization is at risk of intellectual property theft and data corruption due to the Prototype Pollution vulnerability in the lodash library. This vulnerability can be exploited by attackers to inject malicious code, potentially leading to unauthorized access, data breaches, or disruption of critical systems. The affected systems include all web applications utilizing the lodash library, which may compromise sensitive user data and intellectual property.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), DE.CM-8 (Vulnerability Management)
- ISO 27001: A.5.32 (Intellectual Property Rights), A.8.23 (Web Filtering), A.8.31 (Separation of Development, Test, and Production Environments), A.8.30 (Outsourced Development), A.5.10 (Acceptable Use of Information and Other Associated Assets)
- Industry Standards: OWASP A3:2021-Injection, CWE-1321: Prototype Pollution

POLICY REQUIREMENTS
To mitigate the Prototype Pollution vulnerability in lodash, the following security controls must be implemented:
- Utilize a secure version of the lodash library (>= 4.17.21) that addresses the Prototype Pollution vulnerability.
- Implement input validation and sanitization for all user-provided data.
- Conduct regular code reviews to ensure the secure use of the lodash library.
- Success criteria: Verification of the updated lodash library version and validation of input validation and sanitization mechanisms through code reviews and penetration testing.

REMEDIATION PLAN
To remediate the Prototype Pollution vulnerability, the following actions are required:
- Update the lodash library to a secure version (>= 4.17.21).
- Conduct a thorough code review to ensure the secure use of the lodash library.
- Implement input validation and sanitization for all user-provided data.
- Responsible Party: Security Lead
- Timeline: 1 week
- Verification Steps: Code review, re-scan, and penetration testing to validate the updated library version and input validation mechanisms.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the organization will:
- Conduct regular vulnerability scans and code reviews.
- Implement logging and alerting mechanisms to detect potential exploitation attempts.
- Utilize continuous monitoring strategies, including periodic penetration testing and code reviews, to identify and address potential vulnerabilities.
- The Security Lead will be responsible for monitoring and detecting similar vulnerabilities, with support from the Development Team for code reviews and penetration testing.

Policy #7: SCA

Title: express - CWE-601
Severity: MEDIUM
LLM: LLaMA 3.3 70B

POLICY IDENTIFIER
SP-2024-001: Express Open Redirect Protection Policy

RISK STATEMENT
The express open redirect vulnerability poses a significant risk to our organization, as it could allow attackers to redirect users to malicious websites, potentially leading to phishing attacks, malware infections, or other types of cyber threats. If exploited, this vulnerability could compromise the security of our web applications, affecting our customers, employees, and partners who rely on these systems. The potential consequences include reputational damage, financial loss, and legal liabilities.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), DE.AE-2 (Anomaly Detection)
- ISO 27001 Annex A controls: A.8.23 (Web filtering), A.5.30 (ICT readiness for business continuity)
- Industry standards: OWASP A10:2021 (Server-Side Request Forgery), CWE-601 (URL Redirection to Untrusted Site)

POLICY REQUIREMENTS
To mitigate the express open redirect vulnerability, the following security controls must be implemented:
- Validate all user-inputted URLs to ensure they are legitimate and authorized.
- Implement a web application firewall (WAF) to detect and prevent suspicious URL redirections.
- Conduct regular security testing and code reviews to identify and address potential vulnerabilities.
Success criteria: No open redirect vulnerabilities detected during security testing and code reviews.
Validation methods: Regular security scans, code reviews, and penetration testing.

REMEDIATION PLAN
To remediate the express open redirect vulnerability, the following technical actions are required:
- Update the express framework to the latest version.
- Implement URL validation and filtering mechanisms.
- Configure the WAF to detect and prevent suspicious URL redirections.
Responsible party: Dev Lead. Timeline: 2 weeks. Verification steps: Code review, security scan, and penetration testing.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures must be implemented:
- Regular security testing and code reviews.
- Logging and alerting mechanisms to detect suspicious URL redirections.
- Continuous monitoring of web application traffic to identify potential security threats.
Logging requirements: All URL redirections must be logged and monitored for suspicious activity.
Alerting requirements: Alerts must be triggered when suspicious URL redirections are detected.
Continuous monitoring strategies: Regular security scans, code reviews, and penetration testing must be conducted to identify potential vulnerabilities.

Policy #8: SCA

Title: axios - CWE-918
Severity: HIGH
LLM: LLaMA 3.3 70B

POLICY IDENTIFIER
SP-2024-001: Server-Side Request Forgery (SSRF) Protection Policy

RISK STATEMENT
The Server-Side Request Forgery (SSRF) vulnerability in axios poses a significant risk to our organization's security posture. If exploited, this vulnerability could allow attackers to bypass security controls, access sensitive data, and potentially lead to unauthorized access to our systems and data. The affected systems include our web applications, APIs, and related infrastructure. The potential consequences of an SSRF attack include data breaches, unauthorized access to sensitive information, and disruption of business operations.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), DE.CM-8 (Vulnerability Management)
- ISO 27001 Annex A controls: A.8.23 (Web filtering), A.8.1 (User endpoint devices), A.5.10 (Acceptable use of information and other associated assets), A.5.33 (Protection of records), A.8.16 (Monitoring activities)
- Relevant industry standards: OWASP A10:2021 (Server-Side Request Forgery), CWE-918 (Server-Side Request Forgery)

POLICY REQUIREMENTS
To mitigate the SSRF vulnerability, the following security controls must be implemented:
- Validate and sanitize all user-input data
- Implement web application firewalls (WAFs) to detect and prevent SSRF attacks
- Configure axios to use a whitelist of allowed URLs
- Regularly review and update dependencies to ensure the latest security patches are applied
- Success criteria: Conduct regular security audits and penetration testing to validate the effectiveness of these controls.

REMEDIATION PLAN
To remediate the SSRF vulnerability, the following technical actions are required:
- Update axios to the latest version
- Implement URL validation and sanitization
- Configure WAF rules to detect and prevent SSRF attacks
- Responsible party: Security Lead
- Timeline: 1 week
- Verification steps: Conduct a code review, re-scan the application using a vulnerability scanner, and perform a penetration test to validate the remediation.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following monitoring and detection strategies will be implemented:
- Regularly review logs for suspicious activity
- Implement alerting rules to detect potential SSRF attacks
- Conduct regular security audits and penetration testing to identify vulnerabilities
- Continuously monitor dependencies for security updates and apply patches promptly.

Policy #9: SCA

Title: jsonwebtoken - CWE-327
Severity: MEDIUM
LLM: LLaMA 3.3 70B

POLICY IDENTIFIER
SP-2024-001: Secure JSON Web Token Configuration

RISK STATEMENT
The use of insecure default key sizes in JSON Web Tokens (JWT) poses a significant risk to the confidentiality and integrity of our systems and data. If exploited, an attacker could potentially access sensitive information, compromising user data and trust. The affected systems include all applications utilizing JWT for authentication and authorization, impacting both internal and external users.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), PR.IP-1 (Identity and Access Control)
- ISO 27001 Annex A controls: A.8.24 (Use of cryptography), A.8.26 (Application security requirements), A.8.27 (Secure system architecture and engineering principles)
- Industry standards: OWASP Secure Coding Practices, CWE-327 (Use of a Broken or Risky Cryptographic Algorithm)

POLICY REQUIREMENTS
To mitigate the risk associated with insecure default key sizes in JWT, the following security controls must be implemented:
- Use a secure key size of at least 2048 bits for RSA algorithms or equivalent for other algorithms.
- Implement secure key management practices, including key rotation and revocation.
- Conduct regular security audits and penetration testing to identify vulnerabilities.
Success criteria include the successful implementation of secure key sizes and management practices, validated through internal audits and external penetration testing.

REMEDIATION PLAN
To remediate this vulnerability, the following technical actions are required:
- Update the JSON Web Token library to the latest version.
- Configure the application to use a secure key size.
The responsible party for this remediation is the Development Lead. The timeline for completion is 2 weeks. Verification steps include code review and re-scanning for vulnerabilities.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following strategies will be implemented:
- Regular security audits and code reviews.
- Continuous monitoring of application logs for suspicious activity.
- Implementation of a Web Application Firewall (WAF) to detect and prevent common web attacks.
Logging and alerting requirements include logging all authentication and authorization attempts, with alerts triggered for suspicious activity. Continuous monitoring will be performed through regular security scans and penetration testing.

Policy #10: SCA

Title: minimist - CWE-1321
Severity: CRITICAL
LLM: LLaMA 3.3 70B

POLICY IDENTIFIER
SP-2024-001: Minimist Prototype Pollution Protection Policy

RISK STATEMENT
The minimist prototype pollution vulnerability poses a significant risk to our organization's intellectual property and data security. If exploited, this vulnerability could lead to unauthorized access, modification, or theft of sensitive data, resulting in financial losses, reputational damage, and potential legal liabilities. The affected systems include our web applications, development environments, and production servers, which could impact all users and stakeholders.

COMPLIANCE MAPPING
This policy is aligned with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), DE.CM-8 (Vulnerability Management)
- ISO 27001: A.5.32 (Intellectual Property Rights), A.8.23 (Web Filtering), A.8.30 (Outsourced Development), A.8.31 (Separation of Development, Test, and Production Environments), A.8.4 (Access to Source Code)
- Industry standards: OWASP A3:2021 (Injection), CWE-1321 (Prototype Pollution)

POLICY REQUIREMENTS
To mitigate the minimist prototype pollution vulnerability, the following security controls must be implemented:
- Validate and sanitize all user input data
- Implement a web application firewall (WAF) to detect and prevent malicious traffic
- Conduct regular code reviews and security testing
- Ensure separation of development, testing, and production environments
- Limit access to source code and development tools
Success criteria include:
- No detected instances of prototype pollution
- Successful completion of security testing and code reviews
- Verification of WAF configuration and logging

REMEDIATION PLAN
To remediate the minimist prototype pollution vulnerability, the following technical actions are required:
- Update minimist to the latest version
- Implement input validation and sanitization
- Configure WAF rules to detect and prevent prototype pollution attacks
The responsible party is the CTO, given the critical severity of this vulnerability. The timeline for remediation is 24-48 hours. Verification steps include code review, re-scanning, and penetration testing.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures will be implemented:
- Regular security testing and code reviews
- Continuous monitoring of WAF logs and alerts
- Implementation of a vulnerability management program
Logging and alerting requirements include:

- WAF logs will be monitored and alerts will be triggered for suspicious activity
- Security testing and code review results will be documented and tracked

Continuous monitoring strategies include:

- Regularly updating dependencies and libraries
- Conducting periodic security audits and risk assessments

Policy #11: DAST

Title: SQL Injection
Severity: MEDIUM
LLM: LLaMA 3.1 8B Instant

SP-2023-001: SQL Injection Prevention Policy

RISK STATEMENT

This policy addresses the risk of SQL injection attacks on our web application, which may compromise sensitive data and disrupt business operations. If exploited, an attacker could gain unauthorized access to our database, leading to potential data breaches, financial losses, and reputational damage. The affected systems, data, and users include our web application, database, and all users who interact with the application.

COMPLIANCE MAPPING

* NIST Cybersecurity Framework (CSF):
+ PR.DS-5: Implement secure coding practices to prevent common web application vulnerabilities.
+ PR.DS-6: Implement input validation and sanitization to prevent common web application vulnerabilities.
* ISO 27001 Annex A controls:
+ A.8.8: Management of technical vulnerabilities - Obtain information about technical vulnerabilities, evaluate exposure, and take appropriate measures.
+ A.8.16: Monitoring activities - Monitor networks, systems, and applications for anomalous behavior and take appropriate actions.
+ A.8.23: Web filtering - Manage access to external websites to reduce exposure to malicious content.
+ A.5.33: Protection of records - Protect records from loss, destruction, falsification, unauthorized access, and unauthorized release.
* Industry standards:
+ OWASP: Preventing SQL Injection (A5:2017)
+ CWE: CWE-89: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

POLICY REQUIREMENTS

To prevent SQL injection attacks, the following security controls must be implemented:

1. **Input Validation and Sanitization**: All user input must be validated and sanitized on the server-side using prepared statements with parameterized queries.
2. **Secure Coding Practices**: Developers must follow secure coding practices, including type checking and escaping all data received from clients.
3. **Regular Code Reviews**: Regular code reviews must be conducted to ensure that all code changes are secure and compliant with this policy.
4. **Logging and Alerting**: All database queries must be logged, and alerts must be configured to detect suspicious activity.

Success Criteria and Validation Methods

* All user input is validated and sanitized on the server-side.
* Secure coding practices are followed by all developers.
* Regular code reviews are conducted to ensure code changes are secure and compliant with this

policy.
* Logging and alerting are configured to detect suspicious activity.

REMEDIATION PLAN

* **Responsible Party**: Dev Lead
* **Timeline**: 2 weeks
* **Technical Actions**:
1. Update the web application to use prepared statements with parameterized queries.
2. Conduct regular code reviews to ensure secure coding practices are followed.
3. Configure logging and alerting to detect suspicious activity.
* **Verification Steps**:
1. Code review to ensure secure coding practices are followed.
2. Re-scan the web application for vulnerabilities.
3. Conduct penetration testing to ensure the vulnerability is mitigated.

MONITORING AND DETECTION

To detect similar vulnerabilities in the future, the following strategies must be implemented:

* Regularly scan the web application for vulnerabilities using a web application scanner.
* Conduct regular code reviews to ensure secure coding practices are followed.
* Configure logging and alerting to detect suspicious activity.
* Continuously monitor the web application for anomalous behavior.

Policy #12: DAST

Title: Cross Site Scripting (Reflected)
Severity: MEDIUM
LLM: LLaMA 3.1 8B Instant

SP-2023-001: Cross-Site Scripting (XSS) Prevention Policy

Policy Title: Prevent Cross-Site Scripting (XSS) Attacks on Web Applications

Risk Statement:

The Cross-Site Scripting (XSS) vulnerability poses a medium-level risk to our organization's web applications. If exploited, this vulnerability could allow attackers to inject malicious code into user browsers, potentially leading to unauthorized access, data theft, or other malicious activities. This could compromise sensitive data, disrupt business operations, and damage our reputation. The affected systems, data, and users include:

* Web applications hosted on the testapp.local:3000 domain
* Users interacting with these web applications
* Sensitive data stored and processed by these applications

Compliance Mapping:

This policy aligns with the following compliance requirements:

* ISO 27001: A.8.23 (Web filtering), A.8.28 (Secure coding), A.8.29 (Security testing in development and acceptance), A.7.1 (Physical security perimeters), and A.8.8 (Management of technical vulnerabilities)
* NIST Cybersecurity Framework: PR.DS-5 (Protect the confidentiality, integrity, and availability of information)
* CWE: CWE-79 (Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting'))
* OWASP: A1:2017 (Injection)

Policy Requirements:

To prevent Cross-Site Scripting (XSS) attacks on our web applications, the following security controls must be implemented:

1. **Input Validation and Sanitization:** Ensure all user input is validated, filtered, and sanitized to prevent malicious code injection. Convert special characters to HTML entities using a vetted library or framework.
2. **Secure Coding Practices:** Implement secure coding principles, such as using prepared statements and parameterized queries, to prevent SQL injection and other vulnerabilities.
3. **Security Testing:** Define and implement security testing processes in the development life cycle to identify and address vulnerabilities early.
4. **Code Reviews:** Conduct regular code reviews to ensure adherence to secure coding practices and identify potential vulnerabilities.

Success Criteria:

* All web applications hosted on testapp.local:3000 domain are validated and sanitized for user input.

* Secure coding practices are implemented and enforced throughout the development process.
* Regular security testing and code reviews are conducted to identify and address vulnerabilities.

Remediation Plan:

To address this vulnerability, the following technical actions are required:

* Responsible Party: Dev Lead (Medium severity)
* Timeline: 2 weeks
* Verification Steps:
+ Code review to ensure input validation and sanitization
+ Re-scan for vulnerabilities using a web application scanner
+ Penetration testing to simulate real-world attacks

Monitoring and Detection:

To detect similar vulnerabilities in the future, the following strategies will be implemented:

* Regular web application scanning using a vulnerability scanner
* Continuous monitoring of application logs for suspicious activity
* Implementation of a web application firewall (WAF) to detect and prevent malicious traffic

By following this policy, we can prevent Cross-Site Scripting (XSS) attacks on our web applications, protect sensitive data, and maintain a secure online presence.

Policy #13: DAST

Title: Absence of Anti-CSRF Tokens
Severity: MEDIUM
LLM: LLaMA 3.1 8B Instant

POLICY IDENTIFIER
SP-2024-001: Cross-Site Request Forgery (CSRF) Prevention Policy

RISK STATEMENT

The lack of Anti-CSRF tokens in our web application's HTML submission form poses a medium-severity risk to our organization. If exploited, a cross-site request forgery attack could allow an attacker to manipulate our users into performing unintended actions, potentially leading to financial losses, data breaches, or reputational damage. The affected systems, data, and users include our web application's users, customer data, and sensitive business information.

COMPLIANCE MAPPING

This policy is aligned with the following compliance requirements:

* ISO 27001 - A.8.23: Web filtering
* ISO 27001 - A.8.8: Management of technical vulnerabilities
* ISO 27001 - A.5.14: Information transfer
* ISO 27001 - A.6.4: Disciplinary process
* NIST CSF - RC.CO: Recovery activities are communicated to internal and external stakeholders
* CWE-352: Cross-Site Request Forgery (CSRF)

POLICY REQUIREMENTS

To mitigate the risk of cross-site request forgery attacks, the following security controls must be implemented:

1. **Technical Requirements**: Implement a vetted library or framework that prevents CSRF attacks, such as OWASP CSRFGuard.
2. **Procedural Requirements**: Ensure that all web application developers and security personnel are trained on CSRF prevention best practices.
3. **Success Criteria**: Conduct regular code reviews and penetration testing to ensure the absence of CSRF vulnerabilities.
4. **Validation Methods**: Use automated tools, such as OWASP ZAP, to scan for CSRF vulnerabilities.

REMEDIATION PLAN

To address this vulnerability, the following technical actions are required:

1. **Responsible Party**: Dev Lead
2. **Timeline**: 2 weeks
3. **Verification Steps**: Code review, re-scan with OWASP ZAP, and penetration testing

MONITORING AND DETECTION

To detect similar vulnerabilities in the future, the following strategies will be implemented:

1. **Logging and Alerting**: Configure logging and alerting mechanisms to detect suspicious activity.
2. **Continuous Monitoring**: Regularly scan the web application for CSRF vulnerabilities using automated tools.
3. **Code Review**: Conduct regular code reviews to ensure adherence to CSRF prevention best practices.

By implementing this policy, we will ensure the security and integrity of our web application, protect our users' data, and maintain compliance with relevant industry standards and regulations.

Policy #14: DAST

Title: Weak Authentication Method
Severity: MEDIUM
LLM: LLaMA 3.1 8B Instant

SP-2024-001: Secure Authentication over Unsecured Connections

Risk Statement

The use of HTTP basic or digest authentication over an unsecured connection poses a medium-risk vulnerability to our organization. If exploited, an attacker could intercept and reuse user credentials, compromising the confidentiality and integrity of our systems and data. This vulnerability affects all users accessing our application via the `http://testapp.local:3000/api/login` endpoint, potentially leading to unauthorized access, data breaches, and reputational damage.

Compliance Mapping

* NIST Cybersecurity Framework (CSF):
+ PR.DS-5: Protect the confidentiality, integrity, and availability of information
+ ID.AM-1: Identify and manage information system risks
* ISO 27001 Annex A controls:
+ A.8.5: Secure authentication
+ A.7.4: Physical security monitoring
+ A.8.21: Security of network services
+ A.8.1: User endpoint devices
+ A.7.14: Secure disposal or re-use of equipment
* Industry standards:
+ CWE-287: Improper Authentication
+ OWASP: Authentication and Session Management

Policy Requirements

To mitigate this vulnerability, we require the following security controls to be implemented:

1. **Technical Requirements**: Protect the connection using HTTPS (TLS 1.2 or later) or implement a stronger authentication mechanism, such as OAuth or JWT-based authentication.
2. **Procedural Requirements**: Ensure that all authentication requests are made over a secure connection (HTTPS) and that sensitive data is transmitted using encryption.
3. **Success Criteria**: Validate the implementation of HTTPS or a stronger authentication mechanism through code review, re-scan, and penetration testing.
4. **Logging and Alerting**: Implement logging and alerting mechanisms to detect and respond to potential authentication-related incidents.

Remediation Plan

* **Responsible Party**: Dev Lead (Medium severity)
* **Timeline**: 2 weeks
* **Technical Actions**:
1. Update the application to use HTTPS (TLS 1.2 or later) for all authentication requests.
2. Implement a stronger authentication mechanism, such as OAuth or JWT-based authentication.
* **Verification Steps**:

1. Code review to ensure HTTPS or stronger authentication mechanism is implemented correctly.
2. Re-scan the application using a vulnerability scanner to detect any remaining vulnerabilities.
3. Conduct penetration testing to verify the effectiveness of the implemented security controls.

Monitoring and Detection

To detect similar vulnerabilities in the future, we will:

1. **Regularly Review Application Code**: Conduct regular code reviews to identify potential security vulnerabilities.
2. **Implement Logging and Alerting**: Implement logging and alerting mechanisms to detect and respond to potential authentication-related incidents.
3. **Continuously Monitor Application**: Continuously monitor the application for any signs of unauthorized access or data breaches.

By implementing these security controls and procedures, we can mitigate the risk associated with HTTP basic or digest authentication over unsecured connections and ensure the confidentiality, integrity, and availability of our systems and data.

Policy #15: DAST

Title: Cookie Without Secure Flag
Severity: MEDIUM
LLM: LLaMA 3.1 8B Instant

SP-2023-001: Secure Cookie Flag Policy

RISK STATEMENT

The absence of the secure flag on sensitive cookies poses a medium-severity risk to our organization. If exploited, this vulnerability could allow unauthorized access to sensitive information, compromising user confidentiality and potentially leading to data breaches. Affected systems include web applications, user endpoint devices, and network services. The potential consequences of exploitation include unauthorized access to sensitive data, reputational damage, and financial losses.

COMPLIANCE MAPPING

* NIST Cybersecurity Framework:
+ PR.DS-5: Implement secure protocols for data transmission
+ ID.RA-2: Identify and authenticate users
* ISO 27001 Annex A controls:
+ A.8.5: Secure authentication
+ A.8.21: Security of network services
+ A.8.27: Secure system architecture and engineering principles
+ A.8.28: Secure coding
+ A.8.1: User endpoint devices
* Industry standards:
+ CWE-614: Information Exposure Through Browser Side Scripting
+ OWASP: Secure Cookies

POLICY REQUIREMENTS

To mitigate this risk, the following security controls must be implemented:

1. **Technical Requirements**: Ensure that all cookies containing sensitive information or session tokens are passed using an encrypted channel. Set the secure flag for these cookies to prevent unauthorized access.
2. **Procedural Requirements**: Develop and maintain secure coding practices, including the use of secure protocols for data transmission and authentication.
3. **Success Criteria**: Verify that all sensitive cookies are passed using an encrypted channel and the secure flag is set.
4. **Validation Methods**: Conduct regular code reviews, penetration testing, and vulnerability scanning to ensure compliance.

REMEDIATION PLAN

* **Responsible Party**: Dev Lead
* **Timeline**: 2 weeks
* **Technical Actions**:
1. Review and update the web application's cookie handling code to set the secure flag for sensitive cookies.

2. Implement encryption for sensitive cookie transmission.
* **Verification Steps**:
1. Conduct a code review to ensure compliance.
2. Perform a penetration test to simulate attacks on the web application.
3. Re-scan the web application for vulnerabilities.

MONITORING AND DETECTION

To detect similar vulnerabilities in the future:

1. **Logging and Alerting**: Configure logging and alerting mechanisms to detect unauthorized access attempts.
2. **Continuous Monitoring**: Regularly scan the web application for vulnerabilities and conduct penetration testing.
3. **Code Review**: Conduct regular code reviews to ensure compliance with secure coding practices.

By implementing this policy, we can ensure the secure handling of sensitive cookies and prevent unauthorized access to sensitive information.