AI-Powered Security Policy Report

Generated: 2025-11-06 15:54:54

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 faces a significant risk of data breaches and unauthorized access to sensitive information due to potential SQL injection vulnerabilities in our applications. If exploited, these vulnerabilities could lead to the theft of confidential data, disruption of business operations, and damage to our reputation. The affected systems include our user management application, and the impacted data includes user credentials and personal information.

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 before being used in SQL queries.
- Parameterized queries or prepared statements must be used instead of concatenating user input into

SQL queries.
- Regular security testing and code reviews must be performed to identify and remediate potential SQL injection vulnerabilities.
Success criteria include:
- No SQL injection vulnerabilities detected during security testing and code reviews.
- All user input is validated and sanitized.
Validation methods include:
- Regular security testing and code reviews.
- Automated vulnerability scanning.

REMEDIATION PLAN
To remediate the identified SQL injection vulnerability, the following technical actions are required:
- Review and refactor the affected code in src/controllers/UserController.js to use parameterized queries or prepared statements.
- Perform a thorough code review to identify and remediate any similar vulnerabilities.
The responsible party for this remediation is the Security Lead.
The timeline for remediation is 1 week.
Verification steps include:
- Code review by a senior developer.
- Re-scanning the application for SQL injection vulnerabilities.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures must be implemented:
- Regular security testing and code reviews.
- Automated vulnerability scanning.
- Logging and alerting requirements:
- All SQL queries must be logged.
- Alerts must be triggered for suspicious SQL queries.
Continuous monitoring strategies include:
- Regularly reviewing logs for suspicious activity.
- Performing periodic security testing and code reviews.
By following this policy, we can prevent SQL injection attacks and protect our sensitive data from unauthorized access.

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 improper neutralization of input during web page generation. This vulnerability could allow malicious actors to inject malicious scripts into our web application, potentially leading to unauthorized access, data theft, or disruption of services. The affected systems include our web application, and the impacted users are our customers and employees who interact with the application. If exploited, this vulnerability could result in significant reputational damage, financial loss, and legal liabilities.

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.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 must be validated and sanitized before being used in web page generation.
- Template variables used in anchor tags with the 'href' attribute must be properly encoded to prevent malicious input.
- Regular security testing and code reviews must be conducted to identify and remediate potential XSS vulnerabilities.
Success criteria include:
- No detected XSS vulnerabilities in security scans and code reviews.
- All user input is validated and sanitized according to industry standards.
Validation methods include regular security testing, code reviews, and penetration testing.

REMEDIATION PLAN
To remediate the identified vulnerability, the following technical actions are required:
- Review and fix the template variable used in the anchor tag with the 'href' attribute in the src/views/profile.mustache file (line 23).
- Implement input validation and sanitization for all user input used in web page generation.
The responsible party for this remediation is the Development Lead. The timeline for completion is 2 weeks. Verification steps include a code review and re-scan of the application to ensure the vulnerability is fully remediated.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the organization will implement the following monitoring and detection strategies:
- Regular security testing and code reviews to identify potential XSS vulnerabilities.
- Logging and alerting requirements will be established to detect and respond to potential XSS attacks.
- Continuous monitoring strategies will include regular updates to security testing tools and

methodologies to stay current with emerging threats.

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, modification, or deletion. This could result in reputational damage, financial loss, and legal liabilities. The affected systems include our web application servers, and the impacted data includes confidential business information and customer data.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), PR.PT-3 (Access Control)
- ISO 27001 Annex A controls: A.7.1 (Physical security perimeters), 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:
- Validate and sanitize all user input used to construct file paths
- Implement a whitelist approach to restrict file access to authorized directories
- Conduct regular security testing and code reviews to identify and remediate similar vulnerabilities
- Success criteria: No unauthorized file access or data breaches occur due to Path Traversal attacks
- Validation methods: Regular security audits, penetration testing, and code reviews

REMEDIATION PLAN
To remediate the identified vulnerability:
- Technical actions: Review and fix the vulnerable code in src/routes/files.js (line 67) to properly sanitize user input and restrict file access
- 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:
- Implement logging and alerting mechanisms to monitor file access patterns and detect potential Path Traversal attempts
- Conduct regular security testing and code reviews to identify and remediate vulnerabilities
- Utilize automated security scanning tools to identify potential vulnerabilities
- Continuous monitoring strategies: Regularly review and update security policies, conduct security awareness training, and perform vulnerability assessments to ensure the security of our systems and data.

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, an attacker could hijack user sessions via cross-site scripting (XSS) attacks, potentially leading to unauthorized access to sensitive information. This vulnerability affects all users of our web application, compromising the confidentiality and integrity of their data.

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.8.1 (User endpoint devices), A.8.29 (Security testing in development and acceptance), A.8.5 (Secure authentication)
- 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 conducted 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, vulnerability 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.
- Conduct a code review to ensure all cookie sessions are properly configured.
- Responsible party: Dev Lead
- Timeline: 2 weeks
- Verification steps: Code review, re-scan using a vulnerability scanner, 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 must be conducted to identify vulnerabilities in session management.
- Logging and alerting requirements: All security-related events, including login attempts and session management errors, must be logged and alerted to the security team.
- Continuous monitoring strategies: Regularly review and update security policies and procedures to ensure alignment with industry best practices and compliance requirements.

Policy #5: SAST

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

SP-2024-001: Secure Coding Policy for Hardcoded Secrets
==========

POLICY IDENTIFIER
This policy aims to address the vulnerability of hardcoded secrets in our software development process.

RISK STATEMENT
The presence of hardcoded secrets in our codebase poses a significant risk to our organization's security. If exploited, an attacker could gain unauthorized access to sensitive information, compromising our systems, data, and users. This could lead to financial loss, reputational damage, and legal liabilities. The affected systems include our production environment, and the impacted users are our customers and employees.

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: 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 conducted to identify and remove hardcoded secrets.
* Secure coding principles must be applied to software development.
* Success criteria: All secrets are stored securely, and code reviews are conducted regularly.
* Validation methods: Code reviews, security audits, and vulnerability scans.

REMEDIATION PLAN
To address the identified vulnerability, the following technical actions are required:
* Remove the hardcoded secret from the `src/config/auth.js` file.
* Store the secret in an environment variable or a secure secret management system.
* Conduct a code review to identify and remove any other hardcoded secrets.
* Responsible party: Security Lead
* Timeline: 1 week
* Verification steps: Code review, re-scan, and penetration test.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures must be implemented:
* Regular code reviews and security audits.
* Vulnerability scans and penetration tests.
* Logging and alerting requirements: All security-related events must be logged and alerted to the Security Lead.
* Continuous monitoring strategies: Regularly monitor the codebase for hardcoded secrets and other

security vulnerabilities.
* Verification steps: Code review, re-scan, and penetration test.

Policy #6: SCA

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

POLICY IDENTIFIER
Policy ID: SP-2024-001
Policy Title: Prevention of Prototype Pollution in Lodash

RISK STATEMENT
The organization faces a significant risk due to the presence of a Prototype Pollution vulnerability in the lodash library, categorized as CWE-1321. This vulnerability could allow an attacker to manipulate the prototype chain of JavaScript objects, potentially leading to arbitrary code execution, data tampering, or unauthorized access to sensitive information. If exploited, this vulnerability could compromise the integrity and confidentiality of our systems, data, and user information, ultimately affecting our reputation and business operations.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), DE.CM-1 (Anomaly Detection)
- 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 A03:2021 (Injection), CWE-1321 (Prototype Pollution)

POLICY REQUIREMENTS
To mitigate the risk of Prototype Pollution in lodash, the following security controls must be implemented:
- Use a secure version of lodash that has patched the Prototype Pollution vulnerability.
- Implement input validation and sanitization for all user-input data.
- Utilize a Web Application Firewall (WAF) to detect and prevent malicious traffic.
- Success criteria: Successful implementation of the above controls will be validated through regular security audits, penetration testing, and code reviews.

REMEDIATION PLAN
To address the identified vulnerability, the following remediation plan is established:
- Technical actions: Update lodash to the latest secure version, implement input validation and sanitization, and configure the WAF to detect and prevent malicious traffic.
- Responsible party: Security Lead
- Timeline: Completion within 1 week (High severity)
- Verification steps: Code review, re-scan using vulnerability scanning tools, and penetration testing to ensure the vulnerability is fully remediated.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the organization will:
- Implement continuous monitoring of dependencies and libraries for known vulnerabilities.
- Utilize logging and alerting mechanisms to detect potential security incidents.
- Conduct regular security audits and penetration testing to identify and address vulnerabilities before they can be exploited.
- Maintain up-to-date knowledge of industry standards and best practices, such as OWASP and CWE, to ensure the organization's security posture remains robust and effective.

Policy #7: SCA

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

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

RISK STATEMENT
The Express Open Redirect vulnerability poses a significant risk to our organization's web applications, allowing attackers to redirect users to malicious websites, potentially leading to phishing, malware distribution, or other malicious activities. This vulnerability could compromise sensitive user data, damage our reputation, and disrupt business operations. The affected systems include all web applications utilizing the Express framework, and the impacted users are all customers and employees interacting with these applications.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), DE.AE-2 (Anomaly Detection)
- ISO 27001: A.8.23 (Web filtering), A.5.30 (ICT readiness for business continuity), A.5.26 (Response to information security incidents)
- 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 within the expected domain.
- Implement a whitelist of allowed redirect URLs.
- Use the `express-validator` library to sanitize and validate user input.
- Conduct regular code reviews to ensure compliance with this policy.
Success criteria include:
- All user-inputted URLs are validated and sanitized.
- No unauthorized redirects are allowed.
Validation methods include code reviews, penetration testing, and vulnerability scanning.

REMEDIATION PLAN
To remediate this vulnerability, the following technical actions are required:
- Update the Express framework to the latest version.
- Implement URL validation and sanitization using `express-validator`.
- Configure a whitelist of allowed redirect URLs.
The responsible party for this remediation is the Development Lead. The timeline for completion is 2 weeks. Verification steps include code review, re-scanning for vulnerabilities, and penetration testing.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures will be implemented:
- Regular vulnerability scanning and penetration testing.
- Logging and alerting for suspicious URL redirection attempts.
- Continuous monitoring of web application traffic for anomalies.
- Annual code reviews to ensure compliance with this policy.
By following this policy, we can effectively mitigate the Express Open Redirect vulnerability and protect

our web applications from similar threats.

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 vulnerability in axios poses a significant risk to our organization, as it could allow attackers to bypass security controls and access sensitive data. If exploited, this vulnerability could lead to unauthorized access to internal systems, data breaches, and disruption of business operations. The affected systems include all applications using the axios library, and the impacted users are both internal employees and external customers.

COMPLIANCE MAPPING
This policy is aligned with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), DE.CM-1 (Anomalies and Events)
- ISO 27001 Annex A controls: A.8.23 (Web filtering), A.8.1 (User endpoint devices), A.5.10 (Acceptable use of information), 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 used in axios requests
- Implement web filtering to restrict access to unauthorized external websites
- Configure axios to use a proxy server or a whitelist of allowed domains
- Conduct regular security audits and penetration testing to identify potential vulnerabilities
Success criteria include:
- No unauthorized access to internal systems or sensitive data
- All axios requests are validated and sanitized
- Web filtering is enabled and configured correctly
Validation methods include code reviews, security audits, and penetration testing.

REMEDIATION PLAN
To remediate the SSRF vulnerability, the following technical actions are required:
- Update axios to the latest version
- Implement input validation and sanitization for all user-input data
- Configure web filtering and proxy servers
The responsible party for this remediation is the Security Lead, given the High severity of the vulnerability. The timeline for remediation is 1 week. Verification steps include code reviews, re-scanning for vulnerabilities, and penetration testing.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures must be taken:
- Implement logging and alerting for all axios requests
- Conduct regular security audits and penetration testing
- Continuously monitor application logs for anomalies and suspicious activity
Logging requirements include:
- Log all axios requests and responses

- Alert on suspicious or unauthorized requests

Continuous monitoring strategies include:

- Regularly review application logs for anomalies
- Perform periodic security audits and penetration testing
- Stay up-to-date with the latest security patches and updates for axios and other dependencies.

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 decipher sensitive information, leading to unauthorized access, data breaches, or malicious activities. This vulnerability affects all systems and users that rely on JWT for authentication and authorization.

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: A.8.24 (Use of Cryptography), A.8.26 (Application Security Requirements), A.8.27 (Secure System Architecture and Engineering Principles)
- Industry Standards: OWASP A03:2021 (Injection), CWE-327 (Use of a Broken or Risky Cryptographic Algorithm)

POLICY REQUIREMENTS
To mitigate this risk, the following security controls must be implemented:
- Use a secure key size for JWT (minimum 2048-bit RSA or equivalent)
- Implement secure key management practices, including key rotation and revocation
- Conduct regular security audits and penetration testing to identify vulnerabilities
Success criteria include:
- Verification of secure key sizes through code review and cryptographic analysis
- Successful penetration testing without exploiting the JWT vulnerability

REMEDIATION PLAN
To remediate this vulnerability, the following technical actions are required:
- Update the JSON Web Token library to use a secure default key size
- Review and update all JWT-related code to ensure secure key management practices
The responsible party for this remediation is the Development Lead. The timeline for completion is 2 weeks. Verification steps include:
- Code review to ensure secure key sizes and management practices
- Re-scanning the system for vulnerabilities using automated tools

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures will be implemented:
- Regular security audits and penetration testing
- Continuous monitoring of system logs for suspicious activity related to JWT
- Alerting and incident response plans will be established to quickly respond to potential security incidents
Logging requirements include:
- Detailed logs of all JWT-related activities
- Regular log review and analysis to identify potential security issues

By following this policy, we can ensure the secure configuration of JSON Web Tokens, protecting our

systems and data from potential security threats.

Policy #10: SCA

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

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

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

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.8.30 (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 vulnerability scans
- 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 vulnerability scans and code reviews
- Compliance with ISO 27001 and NIST CSF requirements

REMEDIATION PLAN
To remediate the minimist prototype pollution vulnerability, the following actions are required:
- Update the minimist library to the latest version
- Conduct a thorough code review to identify and fix any potential vulnerabilities
- Implement a WAF to detect and prevent malicious traffic
Responsible party: CTO (due to critical severity)
Timeline: 24-48 hours
Verification steps: Code review, re-scan, and penetration test

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures must be implemented:
- Regular logging and alerting for suspicious activity
- Continuous monitoring of web applications and development environments
- Quarterly vulnerability scans and code reviews

- Annual penetration testing and security audits

By following this policy, we can ensure the protection of our intellectual property and sensitive data, and maintain compliance with relevant industry standards and regulations.

Policy #11: DAST

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

Security Policy Section: SP-2023-001 - SQL Injection Prevention

Policy Identifier: SP-2023-001

Policy Title: SQL Injection Prevention and Response

Risk Statement:

The SQL injection vulnerability detected in the `http://testapp.local:3000/api/users/search` endpoint poses a medium-risk threat to the organization. If exploited, this vulnerability could allow unauthorized access to sensitive user data, potentially leading to data breaches, identity theft, and reputational damage. The affected systems, data, and users include the web application's user database, API endpoints, and all users who interact with the application.

Compliance Mapping:

* NIST Cybersecurity Framework (CSF):
+ PR.DS-5: Protect data in use
+ PR.DS-6: Protect data in transit
+ PR.DS-7: Protect data at rest
* ISO 27001 Annex A controls:
+ A.8.8: Management of technical vulnerabilities
+ A.8.16: Monitoring activities
+ A.8.23: Web filtering
+ A.5.33: Protection of records
+ A.6.1: Screening
* Industry standards:
+ CWE-89: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
+ OWASP: SQL Injection Prevention Cheat Sheet

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 to prevent malicious SQL code from being injected.
2. **Parameterized Queries:** Prepared statements with parameterized queries must be used instead of string concatenation to build SQL queries.
3. **Data Encryption:** Sensitive user data must be encrypted both in transit and at rest.
4. **Regular Security Audits:** Regular security audits and vulnerability scans must be performed to detect and address potential security weaknesses.
5. **Secure Coding Practices:** Developers must adhere to secure coding practices, including input validation, error handling, and secure data storage.

Remediation Plan:

To remediate this vulnerability, the following technical actions must be taken:

1. **Responsible Party:** Dev Lead
2. **Timeline:** 2 weeks
3. **Verification Steps:**
* Code review
* Re-scan with a web application scanner
* Penetration testing

Monitoring and Detection:

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

1. **Logging and Alerting:** Log all API requests and errors, and set up alerts for suspicious activity.
2. **Continuous Monitoring:** Regularly scan the web application for vulnerabilities and perform penetration testing.
3. **Security Information and Event Management (SIEM) System:** Implement a SIEM system to monitor and analyze security-related logs and events.

By implementing these security controls and monitoring strategies, the organization can effectively prevent SQL injection attacks and protect sensitive user data.

Policy #12: DAST

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

POLICY IDENTIFIER

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

RISK STATEMENT

The Cross-Site Scripting (XSS) vulnerability in the search function of our web application poses a medium-risk threat to our organization's security. If exploited, this vulnerability could allow an attacker to inject malicious code into a user's browser, potentially leading to unauthorized access to sensitive data, session hijacking, or even malware distribution. This vulnerability affects all users interacting with the search function, including employees, customers, and partners. The potential consequences of a successful attack could include financial loss, reputational damage, and compromised sensitive data.

COMPLIANCE MAPPING

This policy is aligned 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 (Protecting Information and Systems), PR.DS-7 (Protecting Against Malicious Code), and PR.DS-8 (Protecting Against Denial of Service)
* OWASP: A3:2017 - Broken Access Control, A5:2017 - Security Misconfiguration, and A9:2017 - Using Known Vulnerable Components
* CWE: CWE-79 (Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting'))

POLICY REQUIREMENTS

To prevent Cross-Site Scripting (XSS) vulnerabilities, the following security controls must be implemented:

1. **Input Validation and Sanitization**: All user input must be validated, filtered, and sanitized to prevent malicious code injection. This includes converting special characters to HTML entities.
2. **Secure Coding Practices**: Developers must adhere to secure coding principles, including the use of vetted libraries and frameworks that prevent XSS vulnerabilities.
3. **Regular Security Testing**: Security testing processes must be defined and implemented in the development life cycle to detect and prevent XSS vulnerabilities.
4. **Web Application Firewall (WAF)**: A WAF must be configured to detect and block suspicious traffic patterns indicative of XSS attacks.
5. **Logging and Monitoring**: Logs must be configured to detect and alert on potential XSS attacks, and continuous monitoring strategies must be implemented to identify and address vulnerabilities.

REMEDIATION PLAN

To remediate this vulnerability, the following technical actions must be taken:

* Responsible Party: Dev Lead
* Timeline: 2 weeks
* Verification Steps:
+ Code review to ensure input validation and sanitization
+ Re-scan of the web application using a vulnerability scanner
+ Penetration testing to simulate XSS attacks

MONITORING AND DETECTION

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

* Regular security testing and code reviews
* Continuous monitoring of logs and alerts for suspicious traffic patterns
* Implementation of a WAF to detect and block potential XSS attacks
* Regular updates and patches to ensure the web application and its dependencies are up-to-date and secure.

Policy #13: DAST

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

SP-2023-001: Cross-Site Request Forgery (CSRF) Prevention Policy

Policy Identifier: SP-2023-001

Policy Title: CSRF Prevention Policy

Risk Statement:

The absence of Anti-CSRF tokens in HTML submission forms poses a medium-severity risk to our organization. If exploited, this vulnerability could allow an attacker to perform unauthorized actions on our application, potentially leading to data tampering, account takeovers, or other malicious activities. This vulnerability affects our web application's security, specifically the `/api/profile/update` endpoint, and could impact our users' sensitive information.

Compliance Mapping:

* NIST Cybersecurity Framework (CSF):
+ PR.DS-5: Implement secure coding practices to prevent common web application vulnerabilities.
+ RC.CO: Communicate recovery activities to internal and external stakeholders.
* ISO 27001 Annex A controls:
+ A.8.23: Web filtering to manage access to external websites and reduce exposure to malicious content.
+ A.8.8: Management of technical vulnerabilities to obtain, evaluate, and address vulnerabilities.
+ A.5.14: Information transfer to establish secure rules and procedures for transferring information.
+ A.6.4: Disciplinary process to formalize and communicate actions against personnel for information security policy violations.
* Industry standards:
+ OWASP: CSRF is a common web application vulnerability.
+ CWE: CWE-352: Cross-Site Request Forgery.

Policy Requirements:

To prevent CSRF attacks, we require the following security controls to be implemented:

1. **Technical Requirements:**
* Use a vetted library or framework that provides anti-CSRF tokens, such as OWASP CSRFGuard.
* Ensure the application is free of cross-site scripting (XSS) issues, as most CSRF defenses can be bypassed using attacker-controlled script.
2. **Procedural Requirements:**
* Conduct regular code reviews to identify and address potential CSRF vulnerabilities.
* Implement logging and alerting mechanisms to detect and respond to potential CSRF attacks.
3. **Success Criteria:**
* The application's `/api/profile/update` endpoint is protected with anti-CSRF tokens.
* Regular code reviews and security testing identify and address potential CSRF vulnerabilities.
* Logging and alerting mechanisms detect and respond to potential CSRF attacks.

Remediation Plan:

To address this vulnerability, we require the following technical actions:

1. **Responsible Party:** Dev Lead
2. **Timeline:** 2 weeks
3. **Technical Actions:**
* Integrate OWASP CSRFGuard into the application.
* Conduct a thorough code review to identify and address potential CSRF vulnerabilities.
* Implement logging and alerting mechanisms to detect and respond to potential CSRF attacks.
4. **Verification Steps:**
* Conduct a code review to ensure anti-CSRF tokens are properly implemented.
* Perform a re-scan using a vulnerability scanner to verify the fix.
* Conduct a penetration test to simulate a CSRF attack and verify the application's defenses.

Monitoring and Detection:

To detect similar vulnerabilities in the future, we will:

1. **Regularly Scan for Vulnerabilities:** Use a vulnerability scanner to identify potential CSRF vulnerabilities.
2. **Conduct Regular Code Reviews:** Perform regular code reviews to identify and address potential CSRF vulnerabilities.
3. **Implement Logging and Alerting:** Implement logging and alerting mechanisms to detect and respond to potential CSRF attacks.
4. **Continuous Monitoring:** Continuously monitor the application's security posture to identify and address potential CSRF vulnerabilities.

Policy #14: DAST

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

SP-2023-001: 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 the organization. If exploited, this vulnerability could allow unauthorized access to sensitive information, compromising the confidentiality and integrity of our systems and data. This could lead to financial losses, reputational damage, and potential non-compliance with regulatory requirements. The affected systems, data, and users include all users accessing the application via the `http://testapp.local:3000/api/login` endpoint.

Compliance Mapping:

* NIST CSF: PR.DS-5 (Protect the confidentiality, integrity, and availability of information)
* ISO 27001: 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)
* CWE: CWE-287 (Improper Authentication)
* OWASP: A04:2017 - Insecure Authentication

Policy Requirements:

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

1. **Technical Requirements:**
* Protect the connection using HTTPS (TLS 1.2 or later) for all authentication requests.
* Implement a stronger authentication mechanism, such as OAuth or JWT-based authentication.
2. **Procedural Requirements:**
* Conduct regular security audits and vulnerability assessments to identify and remediate similar vulnerabilities.
* Develop and enforce a secure coding standard for all development teams.
3. **Success Criteria:**
* All authentication requests are made over HTTPS.
* A stronger authentication mechanism is implemented and used for all authentication requests.
* Regular security audits and vulnerability assessments are conducted.

Remediation Plan:

* **Responsible Party:** Dev Lead
* **Timeline:** 2 weeks
* **Technical Actions:**
+ Update the application to use HTTPS for all authentication requests.
+ Implement a stronger authentication mechanism (OAuth or JWT-based).
+ Conduct code reviews to ensure secure coding practices.
* **Verification Steps:**
+ Conduct a re-scan of the application using a vulnerability scanner.
+ Perform a penetration test to verify the effectiveness of the remediated controls.

Monitoring and Detection:

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

* Regular security audits and vulnerability assessments will be conducted.
* Logging and alerting requirements will be implemented to detect unauthorized access attempts.
* Continuous monitoring strategies will be implemented to detect potential security incidents.

By implementing these security controls and monitoring strategies, we can mitigate the risk associated with this vulnerability 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 cookies poses a medium-severity risk to our organization's data confidentiality and integrity. If exploited, this vulnerability could allow unauthorized access to sensitive information, potentially leading to data breaches, financial losses, and reputational damage. The affected systems, data, and users include all web applications and users who interact with them, particularly those handling sensitive information.

COMPLIANCE MAPPING

- NIST Cybersecurity Framework (CSF):
- PR.DS-5: Implement secure protocols for sensitive information transmission
- ID.AM-1: Identify and authenticate users and devices
- 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 an Insecure Interface
- OWASP: Secure Cookies (A3)

POLICY REQUIREMENTS

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

1. **Technical Requirements:**
- Ensure that all cookies containing sensitive information or session tokens are set with the secure flag.
- Implement HTTPS (TLS) encryption for all web applications.
2. **Procedural Requirements:**
- Conduct regular code reviews to identify and address insecure cookie usage.
- Develop and maintain secure coding guidelines for web application development.
3. **Success Criteria:**
- All web applications use HTTPS (TLS) encryption.
- All cookies containing sensitive information or session tokens have the secure flag set.
- Regular code reviews identify and address insecure cookie usage.

REMEDIATION PLAN

- **Responsible Party:** Dev Lead
- **Timeline:** 2 weeks
- **Technical Actions:**
1. Update web application code to set the secure flag for sensitive cookies.

2. Configure HTTPS (TLS) encryption for all web applications.
- **Verification Steps:**
1. Conduct code reviews to ensure secure cookie usage.
2. Perform re-scans to verify vulnerability remediation.

MONITORING AND DETECTION

To detect similar vulnerabilities in the future:

1. **Logging and Alerting Requirements:**
- Configure web application logs to monitor cookie usage and secure flag settings.
- Set up alerts for suspicious cookie activity.
2. **Continuous Monitoring Strategies:**
- Regularly scan web applications for vulnerabilities.
- Conduct regular code reviews to identify and address insecure coding practices.

By implementing this policy, we ensure the secure transmission of sensitive information and protect our organization's data confidentiality and integrity.