AI-Powered Security Policy Report

Generated: 2025-11-06 22:36:24

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 Protection Policy

RISK STATEMENT
The organization is at risk of a SQL injection attack, which could lead to unauthorized access to sensitive data, modification of database structures, or disruption of critical business operations. If exploited, this vulnerability could result in significant financial losses, reputational damage, and compromise of customer trust. 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 mitigate the risk of 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 to separate code from user input.

- Regular security testing and code reviews must be performed to identify and address potential vulnerabilities.
- Success criteria include:
- All user input is validated and sanitized.
- No SQL injection vulnerabilities are detected during security testing.
- Code reviews confirm the use of parameterized queries or prepared statements.

REMEDIATION PLAN
To remediate the identified vulnerability, the following actions must be taken:
- Technical action: Review and modify the `src/controllers/UserController.js` file to use parameterized queries or prepared statements.
- Responsible party: Security Lead
- Timeline: 1 week
- 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 performed.
- Logging and alerting must be configured to detect potential SQL injection attacks.
- Continuous monitoring strategies include:
- Regularly updating and patching dependencies and libraries.
- Implementing a Web Application Firewall (WAF) to detect and prevent SQL injection attacks.
- Providing security training to developers to ensure they are aware of the risks and best practices for 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 a Cross-Site Scripting (XSS) attack, which could allow malicious actors to inject malicious code into our web application, potentially leading to unauthorized access to sensitive data, theft of user credentials, or disruption of business operations. This vulnerability affects our web application, specifically the profile page, and could impact all users who access this page. If exploited, this vulnerability could result in significant financial and reputational damage to the organization.

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 injection of malicious code.
- Success criteria: All user input is validated and sanitized, and no XSS vulnerabilities are detected in regular security scans.
- Validation methods: Regular security scans, 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 profile.mustache file (line 23).
- Responsible party: Dev Lead
- Timeline: 2 weeks
- Verification steps: Code review, re-scan, and penetration test 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 scans and code reviews to identify potential XSS vulnerabilities.
- Logging and alerting requirements: All security-related events must be logged and alerts must be sent to the security team in case of potential security incidents.
- Continuous monitoring strategies: Regularly review and update the web application to ensure it remains secure and up-to-date with the latest security patches and best practices.

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 access sensitive files outside the intended directory, potentially leading to data breaches, intellectual property theft, or disruption of critical business operations. This vulnerability affects our web application, specifically the file handling functionality, and could impact all users who interact with the system.

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.8.28 (Secure coding), A.8.29 (Security testing in development and acceptance)
- Relevant industry standards: OWASP Top 10 (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:
- All user input used to construct file paths must be properly sanitized and validated.
- Implement a whitelist approach to only allow access to authorized directories and files.
- Conduct regular security testing and code reviews to identify and address potential vulnerabilities.

Success criteria include:
- No unauthorized access to sensitive files or directories.
- All file handling functionality is properly validated and sanitized.

Validation methods include code reviews, penetration testing, and vulnerability scanning.

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.
- Responsible party: Security Lead
- Timeline: 1 week
- Verification steps: Conduct a code review, re-scan the application for vulnerabilities, and perform a penetration test to ensure the fix is effective.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future:
- Implement logging and alerting for suspicious file access attempts.
- Conduct regular security audits and code reviews.
- Utilize automated vulnerability scanning tools to identify potential weaknesses.

Continuous monitoring strategies include:
- Regularly reviewing system logs for signs of unauthorized access.
- Performing periodic penetration testing and vulnerability assessments.
- Maintaining up-to-date secure coding practices and security testing processes.

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 use 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, penetration testing, and vulnerability scanning.

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, re-scan using a vulnerability scanner, and penetration testing.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures must be taken:
- Regular security testing and code reviews must be performed.
- Logging and alerting requirements: All security-related logs must be monitored, and alerts must be triggered in case of suspicious activity.
- Continuous monitoring strategies: Implement a web application firewall (WAF) and a vulnerability management program to continuously monitor and detect vulnerabilities.

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. 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 Management)
- 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 code analysis tools.
- Success criteria: All hardcoded secrets must be removed from the codebase, and secrets must be stored securely.
- Validation methods: Regular code reviews, automated code analysis, and penetration testing.

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

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures must be implemented:
- Regular code reviews and automated code analysis using tools such as static application security testing (SAST) tools.
- Logging and alerting requirements: All code changes must be logged, and alerts must be triggered for any suspicious activity.
- Continuous monitoring strategies: Regular penetration testing, vulnerability scanning, and code reviews must be performed to detect and remediate vulnerabilities.

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. If exploited, this vulnerability could lead to unauthorized access and modification of sensitive data, resulting in financial loss, reputational damage, and legal liabilities. The affected systems include all web applications utilizing the lodash library, potentially impacting all users and data stored within these systems.

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.8.31 (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 to prevent malicious input.
- Conduct regular code reviews to identify and address potential vulnerabilities in custom code.
- Success criteria: Verification of the secure lodash version and input validation through code review and automated testing.
- Validation methods: Automated testing using tools like OWASP ZAP, manual code review, and penetration testing.

REMEDIATION PLAN
To remediate the Prototype Pollution vulnerability, the following actions are required:
- Technical Action: Update the lodash library to a secure version (>= 4.17.21) and implement input validation and sanitization.
- Responsible Party: Security Lead
- Timeline: 1 week
- Verification Steps: Code review, automated testing using OWASP ZAP, and re-scanning for vulnerabilities.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the organization will:
- Implement continuous monitoring of dependencies and libraries for known vulnerabilities.
- Conduct regular code reviews and automated testing for security vulnerabilities.
- Log and alert on suspicious activity related to user input and data access.
- Perform periodic penetration testing to identify and address potential vulnerabilities.
By following this policy, the organization can effectively mitigate the risks associated with the Prototype

Pollution vulnerability in lodash and ensure the security and integrity of its systems and data.

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's web applications, allowing attackers to redirect users to malicious websites, potentially leading to phishing, malware distribution, or other cyber threats. If exploited, this vulnerability could compromise sensitive user data, damage our reputation, and result in financial losses. The affected systems include our web applications built using the Express framework, and the impacted users are our customers and employees who interact 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 authorized.
- Implement a web application firewall (WAF) to detect and prevent suspicious redirect requests.
- Conduct regular security testing and code reviews to identify potential vulnerabilities.
- Success criteria: No open redirect vulnerabilities detected during security testing and code reviews.
- Validation methods: Regular security testing, code reviews, and WAF logs analysis.

REMEDIATION PLAN
To remediate the Express Open Redirect vulnerability:
- Technical actions: Update Express to the latest version, implement URL validation, and configure the WAF.
- Responsible party: Dev Lead
- Timeline: 2 weeks
- Verification steps: Code review, security testing, and WAF logs analysis.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future:
- Implement logging and alerting for suspicious redirect requests.
- Conduct regular security testing and code reviews.
- Use automated vulnerability scanning tools to identify potential vulnerabilities.
- Continuous monitoring strategies: Regularly review WAF logs, security test results, and code reviews to identify potential vulnerabilities and improve the overall security posture of our web applications.

Policy #8: SCA

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

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

RISK STATEMENT
The axios Server-Side Request Forgery (SSRF) vulnerability poses a significant risk to our organization's security posture. If exploited, an attacker could potentially access sensitive internal systems, data, and services, leading to unauthorized data breaches, lateral movement, and disruption of business operations. This vulnerability affects all systems and users that utilize the axios library, including developers, administrators, and end-users.

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 in axios, 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 internal systems and services
- Configure axios to use a whitelist of allowed domains and protocols
- Regularly review and update axios dependencies to ensure the latest security patches are applied
- Success criteria: Verification of input validation and sanitization through code review and penetration testing
- Validation methods: Automated testing, code review, and regular security audits

REMEDIATION PLAN
To remediate the SSRF vulnerability in axios, the following technical actions are required:
- Update axios dependencies to the latest version
- Implement input validation and sanitization for all user-input data used in axios requests
- Configure web filtering to restrict access to internal systems and services
- Responsible party: Security Lead
- Timeline: 1 week
- Verification steps: Code review, re-scan, and penetration testing

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following monitoring and detection strategies will be implemented:
- Regular security audits and code reviews
- Automated vulnerability scanning and penetration testing
- Logging and alerting for suspicious axios requests
- Continuous monitoring of axios dependencies for security updates and patches

- Regular review of OWASP and CWE guidelines for emerging threats and vulnerabilities

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 our organization's data security. If exploited, an attacker could potentially gain unauthorized access to sensitive information, compromising the confidentiality and integrity of our systems and data. This vulnerability affects all users and systems that rely on JWT for authentication and authorization.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Security), 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 bits for RSA, 128 bits for AES)
- Implement secure key management practices, including key rotation and revocation
- Conduct regular security audits and penetration testing to identify vulnerabilities

Success criteria: Successful implementation of secure JWT configuration, verified through code review and security testing.

REMEDIATION PLAN
To remediate this vulnerability, the following technical actions are required:
- Update JWT library to the latest version
- Configure secure key sizes and management practices

Responsible party: Dev Lead
Timeline: 2 weeks
Verification steps: Code review, security testing, and verification of secure key sizes and management practices.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures will be implemented:
- Regular security audits and penetration testing
- Logging and alerting for suspicious activity related to JWT authentication and authorization
- Continuous monitoring of system and application logs for potential security incidents

By following this policy, we can ensure the secure configuration and use 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
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 integrity. If exploited, this vulnerability could lead to unauthorized access, modification, or theft of sensitive information, resulting in financial losses, reputational damage, 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 A03:2021 (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 assessments
- 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 technical actions are required:
- Update minimist to the latest version
- Implement input validation and sanitization
- Configure the WAF to detect and prevent prototype pollution attacks

The responsible party is the CTO, given the critical severity of the vulnerability. The timeline for remediation is 24-48 hours. Verification steps include:
- Code review
- Vulnerability scan
- 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 data integrity, and maintain compliance with relevant industry standards and regulations.

Policy #11: DAST

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

POLICY IDENTIFIER
SP-2023-001: SQL Injection Prevention Policy

RISK STATEMENT
The SQL injection vulnerability in the `http://testapp.local:3000/api/users/search` endpoint poses a medium-risk threat to the organization's data integrity and confidentiality. If exploited, this vulnerability could allow an attacker to manipulate database queries, potentially leading to unauthorized data access, modification, or deletion. This could result in financial losses, reputational damage, and non-compliance with regulatory requirements. 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)
- 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)

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 a whitelist approach.
2. **Parameterized Queries**: Prepared statements with parameterized queries must be used to build SQL queries instead of string concatenation.
3. **Error Handling**: Error messages must be sanitized to prevent information disclosure.
4. **Code Review**: Regular code reviews must be conducted to ensure compliance with this policy.
5. **Security Training**: Developers must undergo security training to understand the risks associated with SQL injection and how to prevent it.

Success criteria:

* All user input is validated and sanitized on the server-side.
* Prepared statements with parameterized queries are used to build SQL queries.
* Error messages are sanitized to prevent information disclosure.
* Code reviews are conducted quarterly to ensure compliance.
* Developers receive regular security training.

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 compliance with this policy
+ Re-scan the application using a web application scanner to verify the vulnerability has been fixed

+ Conduct a penetration test to simulate a real-world attack

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

* Regular web application scanning using a web application scanner.
* Conduct regular code reviews to ensure compliance with this policy.
* Implement logging and alerting mechanisms to detect potential SQL injection attacks.
* Continuously monitor the application's logs for suspicious activity.

By implementing these measures, we can ensure the prevention of SQL injection attacks and maintain the confidentiality, integrity, and availability of our data.

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

Risk Statement:

Cross-site scripting (XSS) is a type of cyber attack that involves injecting malicious code into a user's browser, potentially leading to unauthorized access, data theft, or system compromise. If exploited, this vulnerability could result in financial losses, reputational damage, and compromised sensitive data. The affected systems and data include the web application at http://testapp.local:3000/search, which processes user input and displays search results.

Compliance Mapping:

* NIST Cybersecurity Framework (CSF):
+ PR.DS-5: Implement secure coding practices to prevent vulnerabilities.
+ PR.DS-6: Validate and sanitize user input to prevent attacks.
* ISO 27001 Annex A controls:
+ A.8.23: Web filtering to manage access to external websites.
+ A.8.28: Secure coding principles to prevent vulnerabilities.
+ A.8.29: Security testing in development and acceptance to identify vulnerabilities.
+ A.8.8: Management of technical vulnerabilities to assess and mitigate risks.
* Industry standards:
+ CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting').
+ OWASP: Cross-Site Scripting (XSS) Prevention Cheat Sheet.

Policy Requirements:

To prevent cross-site scripting (XSS) attacks, the following security controls must be implemented:

1. **Secure Coding Practices:** Use a vetted library or framework that prevents XSS vulnerabilities or provides constructs to avoid them.
2. **Input Validation and Sanitization:** Validate, filter, and sanitize all user input to prevent malicious code injection.
3. **Code Review:** Conduct regular code reviews to identify and address potential vulnerabilities.
4. **Security Testing:** Perform security testing in the development life cycle to identify and remediate vulnerabilities.

Success Criteria:

* All user input is validated, filtered, and sanitized to prevent malicious code injection.
* The web application at http://testapp.local:3000/search is free from XSS vulnerabilities.
* Regular code reviews and security testing are conducted to identify and address potential vulnerabilities.

Remediation Plan:

* **Responsible Party:** Development Lead (Medium severity)
* **Timeline:** 2 weeks
* **Technical Actions:**
1. Review and update the web application's code to use a vetted library or framework that prevents XSS vulnerabilities.
2. Implement input validation and sanitization mechanisms to prevent malicious code injection.
3. Conduct regular code reviews to identify and address potential vulnerabilities.
* **Verification Steps:**
1. Conduct a code review to ensure secure coding practices are implemented.
2. Perform a re-scan of the web application to verify the absence of XSS vulnerabilities.
3. Conduct a penetration test to simulate real-world attacks and identify potential vulnerabilities.

Monitoring and Detection:

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

* Regularly scan the web application for vulnerabilities using a reputable vulnerability scanner.
* Conduct regular code reviews and security testing to identify and address potential vulnerabilities.
* Implement logging and alerting mechanisms to detect and respond to potential security incidents.
* Continuously monitor the web application for suspicious activity and adjust security controls as needed.

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) Protection Policy

RISK STATEMENT

The absence of Anti-CSRF tokens in our web application's HTML submission forms poses a medium-severity risk to our organization. If exploited, this vulnerability could allow an attacker to trick 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, sensitive data stored in our database, and our organization's online reputation.

COMPLIANCE MAPPING

This policy addresses the following compliance requirements:

* ISO 27001: A.8.23 (Web filtering), A.8.8 (Management of technical vulnerabilities), A.5.14 (Information transfer), and A.6.4 (Disciplinary process)
* NIST CSF: PR.DS-5 (Protecting Information and Systems), ID.RA-3 (Risk Management), and RC.CO-3 (Recovery Activities)
* Industry standards: OWASP (Cross-Site Request Forgery), CWE (CWE-352)

POLICY REQUIREMENTS

To mitigate the risk of CSRF attacks, we require the following security controls to be implemented:

1. **Technical Requirements:**
* Implement a vetted library or framework that provides Anti-CSRF tokens, such as OWASP CSRFGuard.
* Ensure that the application is free of cross-site scripting issues.
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 HTML submission forms include Anti-CSRF tokens.
* The application's codebase is free of cross-site scripting issues.
* Logging and alerting mechanisms are implemented and functioning correctly.

REMEDIATION PLAN

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

* **Responsible Party:** Development Lead
* **Timeline:** 2 weeks
* **Verification Steps:**
+ Code review to ensure Anti-CSRF tokens are implemented correctly.
+ Re-scan the application using a web application scanner to identify potential CSRF vulnerabilities.
+ Conduct penetration testing to simulate CSRF attacks and verify the effectiveness of the

implemented controls.

MONITORING AND DETECTION

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

* **Logging and Alerting:** Implement logging and alerting mechanisms to detect and respond to potential CSRF attacks.
* **Continuous Monitoring:** Conduct regular code reviews and penetration testing to identify and address potential CSRF vulnerabilities.
* **Vulnerability Scanning:** Regularly scan the application using a web application scanner to identify 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

This policy addresses the risk of unauthorized access to sensitive information due to the use of HTTP basic or digest authentication over an unsecured connection. If exploited, this vulnerability could lead to unauthorized access to user credentials, potentially resulting in data breaches, financial loss, and reputational damage. The affected systems, data, and users include all users accessing the testapp.local:3000/api/login endpoint.

COMPLIANCE MAPPING

This policy is compliant with the following standards and regulations:

* ISO 27001:2022 - A.8.5 (Secure authentication), A.7.4 (Physical security monitoring), A.8.21 (Security of network services), A.8.1 (User endpoint devices), and A.7.14 (Secure disposal or re-use of equipment)
* NIST Cybersecurity Framework (CSF) - PR.DS-5 (Protect the confidentiality, integrity, and availability of sensitive information)
* CWE-287 (Improper Authentication)

POLICY REQUIREMENTS

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

1. **Technical Requirements**: Protect the connection using HTTPS (TLS 1.2 or higher) or use a stronger authentication mechanism (e.g., OAuth, JWT).
2. **Procedural Requirements**: Ensure that all developers and administrators are trained on secure authentication practices and protocols.
3. **Success Criteria**: Validate the implementation of HTTPS or a stronger authentication mechanism by:
* Verifying the presence of a valid SSL/TLS certificate
* Conducting a code review to ensure secure authentication practices
* Performing a penetration test to simulate unauthorized access attempts

REMEDIATION PLAN

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

1. **Responsible Party**: Development Lead
2. **Timeline**: 2 weeks
3. **Verification Steps**: Code review, re-scan, and penetration test

MONITORING AND DETECTION

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

1. **Logging and Alerting**: Configure logging to capture authentication attempts and alert on suspicious activity.
2. **Continuous Monitoring**: Regularly scan for vulnerabilities and perform penetration testing to identify potential security risks.
3. **Code Review**: Conduct regular code reviews to ensure secure authentication practices and protocols are followed.

By implementing these security controls and monitoring strategies, we can mitigate the risk of unauthorized access to sensitive information and ensure compliance with relevant standards and regulations.

Policy #15: DAST

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

SP-2023-001: Secure Cookie Flag Policy

Policy Identifier: SP-2023-001

Policy Title: Secure Cookie Flag Policy (General Category)

Risk Statement:

The absence of the secure flag on sensitive cookies poses a medium-risk vulnerability, potentially allowing unauthorized access to sensitive information via unencrypted connections. If exploited, this vulnerability could lead to unauthorized access to user sessions, compromising the confidentiality and integrity of our systems. Affected systems include the test application (http://testapp.local:3000/), which stores sensitive user data.

Compliance Mapping:

- NIST CSF: PR.DS-5 (Protect Data at Rest)
- ISO 27001: 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)
- CWE: CWE-614 (Information Exposure Through an Insecure Protocol)
- OWASP: A2: Broken Authentication

Policy Requirements:

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

1. **Technical Requirement:** Ensure that all cookies containing sensitive information or session tokens are passed using an encrypted channel by setting the secure flag.
2. **Procedural Requirement:** Review and update the secure cookie flag policy to reflect the new requirements.
3. **Success Criteria:** Verify that all sensitive cookies have the secure flag set and are transmitted over HTTPS.
4. **Validation Methods:** Conduct regular code reviews, re-scans, and penetration testing to ensure compliance.

Remediation Plan:

- **Responsible Party:** Dev Lead
- **Timeline:** 2 weeks
- **Technical Actions:**
1. Update the test application to set the secure flag for sensitive cookies.
2. Review and update the secure cookie flag policy.
- **Verification Steps:**
1. Conduct a code review to ensure the secure flag is set correctly.
2. Re-scan the application for vulnerabilities.
3. Perform a penetration test to verify the secure flag's effectiveness.

Monitoring and Detection:

To detect similar vulnerabilities in the future:

1. **Logging and Alerting Requirements:** Implement logging and alerting mechanisms to detect any attempts to access sensitive cookies via unencrypted connections.
2. **Continuous Monitoring Strategies:** Regularly scan the application for vulnerabilities and conduct penetration testing to ensure the secure flag is set correctly.
3. **Code Review:** Conduct regular code reviews to ensure compliance with the secure cookie flag policy.

By implementing this policy, we can ensure the confidentiality and integrity of our systems and protect sensitive user data from unauthorized access.