AI-Powered Security Policy Report

Generated: 2025-11-06 22:26:44

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 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 financial and reputational damage. The affected systems include the user management application, and the impacted data includes sensitive user 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:

- 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 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 address any similar vulnerabilities.

Responsible party: Security Lead.
Timeline: 1 week.
Verification steps:

- Code review by a senior developer.
- Automated vulnerability scanning using industry-standard tools.

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 using industry-standard tools.
- Logging and alerting requirements:

+ All SQL queries must be logged.
+ Alerts must be triggered in case of suspicious SQL query patterns.

Continuous monitoring strategies:

- Regularly review and update the application's security testing and code review procedures.
- Stay up-to-date with the latest industry standards and best practices for SQL injection protection.

Policy #2: SAST

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

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

1. POLICY IDENTIFIER
Policy ID: SP-2024-001
Policy Title: Cross-Site Scripting (XSS) Protection Policy

2. RISK STATEMENT
The organization is at risk of Cross-Site Scripting (XSS) attacks, which could lead to unauthorized access to sensitive user data, session hijacking, and malicious code execution. If exploited, this vulnerability could result in significant financial losses, reputational damage, and compromised customer trust. The affected systems include the web application, specifically the profile page, and the data/users at risk are all users accessing the profile page.

3. 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)

4. POLICY REQUIREMENTS
To mitigate the risk of XSS attacks, the following security controls must be implemented:
- Input validation and sanitization for all user-input data
- Output encoding for all data rendered in the browser
- Content Security Policy (CSP) implementation to define allowed sources of content
- Regular security testing and code reviews to identify and address potential vulnerabilities
Success criteria include:
- No detected XSS vulnerabilities in regular security scans
- Successful implementation of CSP and input validation mechanisms
Validation methods include code reviews, penetration testing, and regular security audits.

5. REMEDIATION PLAN
To address the identified vulnerability, the following 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)
- Implement input validation and sanitization for all user-input data
- Develop and implement a Content Security Policy (CSP)
Responsible party: Dev Lead
Timeline: 2 weeks
Verification steps: Code review, re-scan, and penetration test to ensure the vulnerability is resolved and no new vulnerabilities are introduced.

6. MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the organization will:
- Implement regular security scans and code reviews

- Monitor web application logs for suspicious activity
- Utilize threat intelligence to stay informed about emerging XSS threats

Continuous monitoring strategies include regular security audits, penetration testing, and code reviews to ensure the security controls are effective and up-to-date.

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 losses, 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), DE.CM-8 (Vulnerability Management)
- 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)
- Relevant 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 file path constructions
- Implementation of a whitelist-based approach for allowed file paths
- Regular security testing and code reviews to identify similar vulnerabilities
- Success criteria: No detectable path traversal vulnerabilities in code reviews and security scans
- Validation methods: Automated security scans, manual code reviews, and penetration testing

REMEDIATION PLAN
To remediate the identified vulnerability:
- Technical actions: Review and fix the vulnerable code in src/routes/files.js (line 67) by implementing input validation and sanitization
- Responsible party: Security Lead
- Timeline: 1 week
- Verification steps: Code review, re-scan using automated security tools, and penetration testing

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 security scanning tools to identify potential vulnerabilities
- Continuous monitoring strategies: Regularly review system logs, perform vulnerability scans, and conduct penetration testing to identify potential security weaknesses.

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 application, and the impacted users are our customers and employees who use 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.8.23 (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, 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 implemented:
- Regular security testing and code reviews must be conducted 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: Regular vulnerability scanning, penetration testing, and code reviews must be conducted to ensure the security of our web application.

Policy #5: SAST

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

POLICY IDENTIFIER
SP-2024-001: Secure Coding - Hardcoded Secrets 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 secrets.

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 performed regularly to detect and remove hardcoded secrets.
- Secure coding principles and guidelines must be followed during software development.
- Success criteria: No hardcoded secrets are present in the codebase, and all secrets are properly stored and managed.
- Validation methods: Regular code reviews, automated scanning, and penetration testing.

REMEDIATION PLAN
To remediate the existing vulnerability:
- Technical action: Review and fix the hardcoded secret in src/config/auth.js (line 8) by storing it in an environment variable or a secure secret management system.
- Responsible party: Security Lead
- Timeline: 1 week
- 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:
- Regular code reviews and automated scanning must be performed to identify hardcoded secrets.
- Logging and alerting requirements: Any detection of hardcoded secrets must be logged and alerted to the Security Lead.
- Continuous monitoring strategies: Implement a continuous integration and continuous deployment (CI/CD) pipeline to automate code reviews and scanning.

By following this policy, we can ensure the secure storage and management of secrets, reducing the risk of unauthorized access and data breaches.

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 exploitation of the lodash prototype pollution vulnerability poses a significant risk to our organization's intellectual property and sensitive data. If left unaddressed, this vulnerability could allow attackers to manipulate the prototype chain of JavaScript objects, potentially leading to arbitrary code execution, data tampering, or unauthorized access to sensitive information. This could compromise our web applications, affecting both internal users and external customers, and resulting in reputational damage, financial losses, and legal liabilities.

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 A03:2021 (Injection), CWE-1321 (Prototype Pollution)

POLICY REQUIREMENTS
To mitigate the risks associated with lodash prototype pollution, the following security controls must be implemented:
- Use a secure version of lodash that is not vulnerable to prototype pollution.
- Implement input validation and sanitization for all user-provided data.
- Utilize a Content Security Policy (CSP) to define which sources of content are allowed to be executed within a web page.
- Conduct regular security audits and penetration testing to identify and address potential vulnerabilities.

Success criteria include the successful implementation of these controls, as validated through code reviews, security testing, and compliance audits.

REMEDIATION PLAN
To remediate this vulnerability, the following technical actions are required:
- Update all instances of lodash to a secure version.
- Implement input validation and sanitization for all user-provided data.

The Security Lead is responsible for overseeing the remediation efforts, given the High severity of this vulnerability. The timeline for completion is 1 week. Verification steps include code reviews, re-scanning for vulnerabilities, and penetration testing to ensure the updates have been successfully applied and are effective.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, we will implement the following monitoring and detection strategies:
- Regularly review and update dependencies to ensure we are using secure versions of libraries like lodash.
- Utilize logging and alerting tools to detect potential security incidents.

- Conduct continuous monitoring through regular security audits, penetration testing, and vulnerability scanning.

This proactive approach will help identify and mitigate potential vulnerabilities before they can be exploited, ensuring the security and integrity of our 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 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 unauthorized access to sensitive information. If exploited, this vulnerability could compromise the confidentiality, integrity, and availability of our systems and data, affecting all users who interact with our web applications. The potential consequences include financial loss, reputational damage, and legal liabilities.

COMPLIANCE MAPPING
This policy is aligned 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)
- Relevant 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 sanitization using a reputable library
- 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 monitoring and detection strategies will be implemented:
- Regular security scans and code reviews
- Logging and alerting on suspicious URL redirections
- Continuous monitoring of web application traffic for anomalies
- Annual penetration testing and security audits to identify and address 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 execute malicious code on our servers. The potential consequences include unauthorized access to sensitive information, disruption of business operations, and damage to our reputation. The affected systems include all web applications using the axios library, and the impacted users are both internal employees and external customers.

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 used in axios requests
- Implement web filtering controls to restrict access to unauthorized external websites
- Conduct regular security audits and penetration testing to identify potential vulnerabilities
- Ensure that all developers undergo training on secure coding practices, including input validation and sanitization

Success criteria include:
- No unauthorized access to sensitive data or systems
- No successful exploitation of SSRF vulnerabilities
- Regular security audits and penetration testing results indicate no high-severity vulnerabilities

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 used in axios requests
- Configure web filtering controls to restrict access to unauthorized external websites

Responsible party: Security Lead
Timeline: 1 week
Verification steps: Code review, re-scan using vulnerability scanning tools, and penetration testing

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures must be implemented:
- Regular security audits and penetration testing
- Continuous monitoring of web application logs for suspicious activity
- Alerting and incident response procedures for potential security incidents

Logging and alerting requirements include:
- Log all axios requests and responses
- Alert on suspicious activity, such as unauthorized access attempts or unusual request patterns

Continuous monitoring strategies include:
- Regularly review web application logs for suspicious activity
- Conduct periodic security audits and penetration testing to identify potential 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 access 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, potentially leading to unauthorized access, data breaches, and reputational damage.

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 vulnerability, 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 potential vulnerabilities

Success criteria include:
- Verification of secure key sizes through code review and security testing
- Implementation of secure key management practices
- Regular security audits and penetration testing to identify potential vulnerabilities

REMEDIATION PLAN
To remediate this vulnerability, the following technical actions are required:
- Update JWT configuration to use a secure key size
- Implement 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 verify secure key sizes
- Security testing to validate the implementation of secure key 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 requirements for suspicious activity related to JWT authentication and authorization
- Continuous monitoring of system and application logs for potential security incidents

By implementing these measures, we can ensure the secure use of JSON Web Tokens and protect 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 security. If exploited, this vulnerability could lead to unauthorized access and modification of sensitive data, resulting in financial loss, 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 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 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 completion of regular code reviews and vulnerability assessments
- Implementation of a WAF and logging/alerting mechanisms

REMEDIATION PLAN
To remediate the minimist prototype pollution vulnerability, the following technical actions are required:
- Update the minimist library to the latest version
- Implement input validation and sanitization for all user-input data
- Configure the WAF to detect and prevent prototype pollution attacks

The responsible party for this remediation is the CTO, given the critical severity of the vulnerability. The timeline for completion is within 24-48 hours.
Verification steps include:
- Code review of the updated minimist library and input validation/sanitization implementation
- Re-scanning of the application for vulnerabilities
- Penetration testing to ensure the WAF is configured correctly

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

- Regular logging and alerting for suspicious activity
- Continuous integration and continuous deployment (CI/CD) pipeline monitoring
- Regular code reviews and vulnerability assessments
- Implementation of a security information and event management (SIEM) system to detect and respond to potential security incidents.

Policy #11: DAST

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

SP-2023-001: SQL Injection Prevention Policy

RISK STATEMENT

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

COMPLIANCE MAPPING

* NIST Cybersecurity Framework (CSF):
+ PR.DS-5: Implement secure coding practices
+ PR.DS-6: Implement secure software development life cycle (SDLC) practices
* 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:
+ OWASP: SQL Injection Prevention Cheat Sheet
+ 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 a whitelist approach. This includes type checking, length checking, and escaping of special characters.
2. **Parameterized Queries**: All database queries must use parameterized queries with prepared statements to prevent SQL injection.
3. **Secure Coding Practices**: Developers must adhere to secure coding practices, including the use of ORM (Object-Relational Mapping) tools and secure coding guidelines.
4. **Code Reviews**: Regular code reviews must be conducted to ensure that all code changes meet the security requirements.
5. **Logging and Monitoring**: All database queries and user input must be logged and monitored for suspicious activity.

Success Criteria:

* All user input is validated and sanitized on the server-side.
* All database queries use parameterized queries with prepared statements.
* Secure coding practices are followed by all developers.

* Regular code reviews are conducted to ensure security requirements are met.

REMEDIATION PLAN

* **Responsible Party:** Dev Lead
* **Timeline:** 2 weeks
* **Technical Actions:**
1. Update the `http://testapp.local:3000/api/users/search` endpoint to use parameterized queries with prepared statements.
2. Implement input validation and sanitization on the server-side using a whitelist approach.
3. Conduct a code review to ensure all code changes meet the security requirements.
* **Verification Steps:**
1. Code review to ensure secure coding practices are followed.
2. Re-scan the application for SQL injection vulnerabilities.
3. Conduct a penetration test to ensure the vulnerability is not present.

MONITORING AND DETECTION

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

* **Logging and Alerting:** All database queries and user input must be logged and monitored for suspicious activity.
* **Continuous Monitoring:** Regular security scans and penetration tests must be conducted to ensure the application is secure.
* **Code Reviews:** Regular code reviews must be conducted to ensure all code changes meet the security requirements.

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) 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. If exploited, this vulnerability could allow an attacker to inject malicious code into a user's browser, potentially leading to unauthorized access, data theft, or other malicious activities. This vulnerability affects our web application's search function, which is used by various users across the organization. If left unaddressed, this vulnerability could lead to reputational damage, financial losses, and compromised user data.

COMPLIANCE MAPPING

* **NIST CSF**: PR.DS-5 (Protect Data) - Implement secure coding practices to prevent data breaches.
* **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), A.8.8 (Management of technical vulnerabilities)
* **OWASP**: A3:2017 - Broken Access Control (XSS is a subset of Broken Access Control)
* **CWE**: CWE-79 - Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

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 attacks or provides constructs that make it easier to avoid this weakness.
2. **Input Validation and Sanitization**: Ensure all input is validated, filtered, and sanitized to prevent malicious code injection.
3. **Code Review**: Conduct regular code reviews to identify and address potential XSS vulnerabilities.
4. **Security Testing**: Implement security testing processes in the development life cycle to detect and prevent XSS attacks.

REMEDIATION PLAN

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

* **Responsible Party**: Development Lead
* **Timeline**: 2 weeks
* **Verification Steps**:
1. Code review to ensure input validation and sanitization.
2. Re-scan the web application using a vulnerability scanner to detect any remaining XSS vulnerabilities.
3. Conduct a penetration test to simulate a real-world attack scenario.

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 and alerting mechanisms to detect potential XSS attacks.
2. **Continuous Monitoring**: Conduct regular security testing and code reviews to identify and address potential XSS vulnerabilities.
3. **Vulnerability Scanning**: Regularly scan the web application using a vulnerability scanner to detect any remaining XSS vulnerabilities.

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

This policy addresses the risk of Cross-Site Request Forgery (CSRF) attacks, which can compromise the security of our web applications and expose sensitive user data. If exploited, a CSRF attack can lead to unauthorized changes to user profiles, financial transactions, or other sensitive operations. This vulnerability affects our web application's API, specifically the `/api/profile/update` endpoint, which is accessible to all users.

COMPLIANCE MAPPING

- NIST Cybersecurity Framework (CSF): PR.DS-5 (Protecting Data at Rest)
- ISO 27001 Annex A controls: A.8.23 (Web filtering), A.8.8 (Management of technical vulnerabilities), A.5.14 (Information transfer), and A.6.4 (Disciplinary process)
- Industry standards: OWASP (Cross-Site Request Forgery (CSRF)), CWE (CWE-352: Cross-Site Request Forgery (CSRF))

POLICY REQUIREMENTS

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

1. **Implementation of Anti-CSRF Tokens**: Our web application must use a vetted library or framework that provides anti-CSRF tokens, such as OWASP CSRFGuard.
2. **Proper HTML Form Handling**: All HTML forms must include anti-CSRF tokens to prevent CSRF attacks.
3. **Regular Code Reviews**: Our development team must conduct regular code reviews to ensure that our application is free of cross-site scripting issues, which can bypass CSRF defenses.
4. **Logging and Alerting**: Our logging and alerting systems must detect and alert on suspicious activity related to CSRF attacks.

Success Criteria:

- The web application's API has implemented anti-CSRF tokens.
- All HTML forms include anti-CSRF tokens.
- The development team has conducted regular code reviews to ensure CSRF defenses are in place.

REMEDIATION PLAN

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

- **Responsible Party**: Dev Lead
- **Timeline**: 2 weeks
- **Technical Actions**:
1. Implement OWASP CSRFGuard in our web application.
2. Update all HTML forms to include anti-CSRF tokens.

3. Conduct regular code reviews to ensure CSRF defenses are in place.
- **Verification Steps**:
1. Code review to ensure anti-CSRF tokens are implemented correctly.
2. Re-scan the web application for CSRF vulnerabilities.
3. Conduct penetration testing to verify CSRF defenses.

MONITORING AND DETECTION

To detect similar vulnerabilities in the future, we require:

- **Logging and Alerting**: Our logging and alerting systems must detect and alert on suspicious activity related to CSRF attacks.
- **Continuous Monitoring**: Our security team must conduct regular security assessments to identify and address potential CSRF vulnerabilities.
- **Code Reviews**: Our development team must conduct regular code reviews to ensure CSRF defenses are in place.

By implementing this policy, we can mitigate the risk of CSRF attacks and protect our web application's security and user data.

Policy #14: DAST

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

SP-2023-001: Secure Authentication over Unsecured Connections

Policy Identifier: SP-2023-001
Policy Title: Secure Authentication over Unsecured Connections

Risk Statement:

The use of HTTP basic or digest authentication over an unsecured connection poses a medium-severity risk to the organization. If exploited, this vulnerability could allow unauthorized access to sensitive information, leading to potential data breaches and reputational damage. The affected systems, data, and users include:

* All users accessing the application via the affected API endpoint
* Sensitive information stored on user endpoint devices
* Network services and premises used to access the application

Compliance Mapping:

* NIST CSF: PR.DS-5 (Protect Data in Transit)
* 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)
* Industry standards: CWE-287 (Improper Authentication), OWASP (Authentication and Session Management)

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)
* Implement a stronger authentication mechanism, such as OAuth or JWT
2. **Procedural Requirements:**
* Conduct regular security audits and vulnerability scans to detect similar vulnerabilities
* Implement logging and alerting mechanisms to detect unauthorized access attempts
* Continuously monitor network services and premises for unauthorized physical access
3. **Success Criteria:**
* HTTPS connection established on the affected API endpoint
* Stronger authentication mechanism implemented and tested
* Regular security audits and vulnerability scans conducted
* Logging and alerting mechanisms implemented and tested

Remediation Plan:

* **Responsible Party:** Dev Lead
* **Timeline:** 2 weeks
* **Technical Actions:**

+ Update the API endpoint to use HTTPS
+ Implement a stronger authentication mechanism
+ Conduct code review and re-scan to ensure vulnerability is resolved
* **Verification Steps:**
+ Conduct penetration testing to verify vulnerability is resolved
+ Review logging and alerting mechanisms to ensure they are functioning correctly

Monitoring and Detection:

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

* Regular security audits and vulnerability scans
* Continuous monitoring of network services and premises
* Logging and alerting mechanisms to detect unauthorized access attempts
* Code review and re-scan to ensure vulnerability is resolved

By implementing these security controls and procedures, we can mitigate the risk of unauthorized access to sensitive information and protect the organization's reputation.

Policy #15: DAST

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

Security Policy Section: SP-2023-001 - Secure Cookie Flag

Policy Identifier: SP-2023-001
Policy Title: Secure Cookie Flag Implementation

Risk Statement:

The absence of the secure flag on sensitive cookies poses a medium-severity risk to the organization. If exploited, this vulnerability could allow unauthorized access to sensitive information, compromising user data and potentially leading to financial losses or reputational damage. The affected systems include the web application serving cookies without the secure flag, specifically the endpoint at http://testapp.local:3000/. This vulnerability affects users who interact with the application, potentially exposing their sensitive information to unauthorized parties.

Compliance Mapping:

* NIST Cybersecurity Framework (CSF):
+ PR.DS-5: Protect data in transit
+ ID.BE-1: 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
* Industry standards:
+ CWE-614: Use of Hard-coded Passwords or Credentials
+ OWASP: Secure Cookies

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 a secure cookie flag policy to enforce this requirement.
2. **Procedural Requirements:**
* Conduct regular code reviews to identify and address insecure cookie usage.
* Develop and maintain a secure coding policy to ensure adherence to secure coding principles.

Success Criteria and Validation Methods:

* Success criteria: All cookies containing sensitive information or session tokens are set with the secure flag.
* Validation methods: Regular code reviews, manual verification of cookie flags, and automated scanning tools.

Remediation Plan:

* **Responsible Party:** Dev Lead
* **Timeline:** 2 weeks
* **Technical Actions:**
1. Update the web application to set cookies with the secure flag.
2. Implement a secure cookie flag policy.
3. Conduct regular code reviews to identify and address insecure cookie usage.
* **Verification Steps:**
1. Code review to ensure secure cookie flag implementation.
2. Automated scanning to verify secure cookie flag usage.
3. Manual verification of cookie flags.

Monitoring and Detection:

To detect similar vulnerabilities in the future:

1. **Logging and Alerting Requirements:** Implement logging and alerting mechanisms to detect insecure cookie usage.
2. **Continuous Monitoring Strategies:** Regularly scan the web application for insecure cookie usage and conduct code reviews to identify and address potential vulnerabilities.

By implementing these security controls and procedures, the organization can mitigate the risk associated with insecure cookie usage and ensure the secure transmission of sensitive information.