AI-Powered Security Policy Report

Generated: 2025-11-06 23:36:38

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 business impact, including reputational damage, financial loss, and regulatory non-compliance. The affected systems include the User Controller application, and the impacted users are all individuals with access to 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: 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 remediate 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 technical actions are required:
- Review and refactor the User Controller application to use parameterized queries or prepared statements.
- Perform a code review to ensure all user input is validated and sanitized.
- Conduct a security test to verify the remediation.
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 security testing and code reviews must be performed.
- Logging and alerting must be configured to detect anomalous database activity.
- Continuous monitoring strategies must include:
- Regularly updating and patching dependencies.
- Implementing a Web Application Firewall (WAF) to detect and prevent SQL injection attacks.
- Conducting regular security awareness training for developers to ensure they are aware of the risks and consequences of SQL injection attacks.

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
This policy is identified as SP-2024-001, titled "Cross-Site Scripting (XSS) Protection Policy," aimed at mitigating the risks associated with Cross-Site Scripting vulnerabilities.

2. RISK STATEMENT
The exploitation of Cross-Site Scripting (XSS) vulnerabilities poses a significant risk to our organization. If an attacker successfully injects malicious scripts into our web applications, it could lead to unauthorized access to sensitive user data, session hijacking, and potentially devastating financial and reputational consequences. All users interacting with our web applications are at risk, along with the systems and data those applications access.

3. COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), DE.AE-3 (Anomaly Detection)
- 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 XSS vulnerabilities, the following security controls must be implemented:
- **Input Validation:** All user input must be validated and sanitized before being reflected in the application's response.
- **Output Encoding:** User input must be properly encoded to prevent the injection of malicious scripts.
- **Content Security Policy (CSP):** Implement a robust CSP to define which sources of content are allowed to be executed within a web page.
Success criteria include regular security audits and penetration testing to validate the effectiveness of these controls.

5. REMEDIATION PLAN
For the identified "Var In Href" vulnerability:
- **Technical Actions:** Review and fix the template variable usage in the 'href' attribute in `src/views/profile.mustache` (line 23) to prevent the injection of malicious scripts.
- **Responsible Party:** Development Lead
- **Timeline:** 2 weeks
- **Verification Steps:** Conduct a code review and re-scan the application for XSS vulnerabilities after the fix is implemented.

6. MONITORING AND DETECTION
To detect similar vulnerabilities in the future:
- Regularly update and patch web applications and their components.
- Implement logging and alerting for suspicious web application activity.
- Conduct regular security audits and penetration testing.

- Utilize web application firewalls (WAFs) to monitor and filter HTTP traffic.

By adhering to this policy, we aim to significantly reduce the risk of XSS attacks, protecting our users, data, and systems from potential harm.

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 user information and business-critical files.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), PR.IP-1 (Identity and Access Control)
- ISO 27001 Annex A controls: A.7.1 (Physical security perimeters), A.8.28 (Secure coding), A.8.29 (Security testing in development and acceptance)
- 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:
- Input validation and sanitization for all user-provided file paths
- Implementation of a whitelist-based approach for allowed file paths
- Regular security testing and code reviews to identify and address potential vulnerabilities
- Success criteria: No unauthorized access to sensitive files or data; validation methods include regular 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 construct file paths
- Responsible party: Security Lead
- Timeline: 1 week
- Verification steps: Code review, re-scan using a vulnerability scanner, and penetration testing to ensure the vulnerability is fully addressed

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: Regularly review system logs, perform vulnerability scans, and conduct penetration testing to identify and address 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, 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 who access our web applications, 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 Annex A controls: A.8.1 (User endpoint devices), A.8.23 (Web filtering), 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 address 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 (line 12) 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 to ensure the 'httpOnly' flag is correctly implemented.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures must be taken:
- Regular security testing and code reviews must be conducted to identify potential vulnerabilities.
- Logging and alerting requirements: All security-related logs must be monitored for suspicious activity, and alerts must be triggered in case of potential security incidents.
- Continuous monitoring strategies: Implement a web application firewall (WAF) to detect and prevent XSS attacks, and regularly update security testing tools to ensure coverage of emerging threats.

Policy #5: SAST

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

POLICY IDENTIFIER
SP-2024-001: Hardcoded Secret Policy

RISK STATEMENT
The presence of hardcoded secrets in our codebase poses a significant risk to our organization's security posture. If exploited, an attacker could gain unauthorized access to sensitive information, compromising the confidentiality, integrity, and availability of our systems and data. This vulnerability affects our authentication mechanisms, potentially impacting all users and systems that rely on these 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 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 A6:2021 (Security Misconfiguration), 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 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 managed and secured.
- Validation methods: Regular code reviews, automated scanning, 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.
- Conduct a thorough code review to detect and remove any other hardcoded secrets.
- Responsible party: Security Lead
- Timeline: 1 week
- Verification steps: Code review, automated scanning, and penetration testing.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures must be implemented:
- Regular code reviews and automated scanning to detect hardcoded secrets.
- Logging and alerting mechanisms to detect and respond to potential security incidents.
- Continuous monitoring strategies, including regular penetration testing and vulnerability assessments, to identify and remediate vulnerabilities.
- Training and awareness programs for developers to educate them on secure coding practices and the risks associated with hardcoded secrets.

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 could have significant business impacts, including unauthorized modification of sensitive data, disruption of critical services, and potential intellectual property theft. If left unaddressed, this vulnerability could allow attackers to manipulate the prototype chain of JavaScript objects, potentially leading to remote code execution, data tampering, or other malicious activities. This vulnerability affects all systems and applications utilizing the lodash library, potentially exposing sensitive user data and compromising the integrity of our services.

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 (Weaknesses in Prototype Pollution)

POLICY REQUIREMENTS
To mitigate the risks associated with the lodash prototype pollution vulnerability, the following security controls must be implemented:
- Utilize a patched version of the lodash library (>= 4.17.21) in all applications and services.
- Implement Content Security Policy (CSP) to restrict the execution of malicious scripts.
- Conduct regular code reviews to identify and address potential vulnerabilities in custom code.
- Success criteria: Verification of patched library versions, CSP implementation, and code review results.
- Validation methods: Automated vulnerability scanning, manual code review, and penetration testing.

REMEDIATION PLAN
To remediate this vulnerability, the following technical actions are required:
- Update all instances of the lodash library to the latest patched version.
- Implement CSP in all web applications.
- Conduct a thorough code review to identify and address potential vulnerabilities.
- Responsible party: Security Lead
- Timeline: 1 week
- Verification steps: Automated vulnerability scanning, code review, and penetration testing.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures must be implemented:
- Regular automated vulnerability scanning of all applications and services.
- Logging and alerting mechanisms to detect potential exploitation attempts.
- Continuous monitoring of code repositories and dependency updates to identify potential vulnerabilities.
- Regular security awareness training for developers to ensure secure coding practices.

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, potentially allowing attackers to redirect users to malicious websites, resulting in phishing attacks, malware distribution, or other types of cyber threats. If exploited, this vulnerability could compromise sensitive user data, damage our reputation, and lead to financial losses. The affected systems include all web applications built using the Express framework, and the impacted users are all individuals 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), A.7.2 (Physical entry), A.8.1 (User endpoint devices)
- 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
- Regularly update and patch the Express framework to ensure the latest security fixes are applied
- Conduct regular security audits and penetration testing to identify potential vulnerabilities

Success criteria include:
- No detected instances of open redirect attacks
- All user-inputted URLs are validated and authorized
- WAF rules are regularly updated to reflect emerging threats

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 authorization mechanisms
- Configure the WAF to detect and prevent suspicious redirect requests

The responsible party for this remediation is the Development Lead. The timeline for completion is 2 weeks. Verification steps include:
- Code review to ensure URL validation and authorization mechanisms are properly implemented
- Re-scanning the application using a vulnerability scanner to ensure the patch is applied correctly
- Penetration testing to simulate open redirect attacks and verify the WAF rules are effective

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following monitoring and detection strategies will be implemented:
- Regular security audits and penetration testing

- Continuous logging and alerting of suspicious redirect requests
- Regular review of WAF rules to ensure they are up-to-date and effective
- Implementation of an incident response plan to quickly respond to detected vulnerabilities or attacks.

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 an attacker to bypass security controls, access sensitive data, and potentially lead to unauthorized access to our systems and data. The affected systems include all applications utilizing the axios library, and the impacted users are all individuals with access to these applications. The potential consequences of an SSRF attack include data breaches, unauthorized access to sensitive information, and reputational damage.

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 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 vulnerability scans and penetration tests to identify potential SSRF vulnerabilities
- Success criteria: No SSRF vulnerabilities detected during vulnerability scans and penetration tests
- Validation methods: Regular code reviews, vulnerability scans, and penetration tests

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 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, and penetration test

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following monitoring and detection strategies will be implemented:
- Regular vulnerability scans and penetration tests
- Logging and alerting for suspicious axios requests
- Continuous monitoring of application logs for signs of SSRF attacks
- Annual review and update of this policy to ensure compliance with evolving industry standards and best practices.

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 gain unauthorized access to sensitive information, compromising our users' trust and potentially leading to financial losses. The affected systems include all applications utilizing JWT for authentication and authorization, impacting both internal and external users.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), PR.IP-1 (Identity Management)
- ISO 27001 Annex A controls: A.8.24 (Use of cryptography), A.8.26 (Application security requirements), A.8.27 (Secure system architecture and engineering principles)
- Industry standards: OWASP A03:2021 (Injection), CWE-327 (Use of a Broken or Risky Cryptographic Algorithm)

POLICY REQUIREMENTS
To mitigate the risk associated with insecure default key sizes in JWT, the following security controls must be implemented:
- Use a secure key size of at least 2048 bits for RSA algorithms or equivalent for other algorithms.
- Implement key rotation and revocation mechanisms to ensure timely updates and minimize the impact of compromised keys.
- Conduct regular security audits and penetration testing to identify and address potential vulnerabilities.

Success criteria include the successful implementation of secure key sizes and the absence of vulnerabilities in regular security audits. Validation methods include code reviews, security scans, and penetration testing.

REMEDIATION PLAN
To remediate this vulnerability, the following technical actions are required:
- Update the JSON Web Token library to the latest version.
- Configure the library to use a secure key size.
- Implement key rotation and revocation mechanisms.

The responsible party for this remediation is the Development Lead. The timeline for completion is 2 weeks. Verification steps include a code review and a re-scan using a vulnerability scanner to ensure the insecure default key size has been addressed.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures must be taken:
- Regularly review and update dependencies to ensure the use of the latest secure versions.
- Implement logging and alerting for any suspicious activity related to authentication and authorization.
- Conduct continuous monitoring through regular security audits and penetration testing to identify and address potential vulnerabilities before they can be exploited.

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 losses, reputational damage, and potential legal liabilities. The affected systems include our web applications, development environments, and production servers, which could impact all users and stakeholders.

COMPLIANCE MAPPING
This policy 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:
- Conduct regular code reviews and vulnerability assessments to identify and remediate similar issues
- Implement web filtering and access controls to restrict access to external websites and minimize exposure to malicious content
- Enforce separation of development, test, and production environments to prevent unauthorized access and data breaches
- Manage access to source code, development tools, and software libraries to prevent unauthorized modifications
- Success criteria: Successful implementation of security controls, verified through code reviews, vulnerability assessments, and penetration testing

REMEDIATION PLAN
To remediate the minimist prototype pollution vulnerability, the following technical actions are required:
- Update the minimist library to the latest version
- Conduct a thorough code review to identify and remediate similar issues
- Implement additional security controls, such as input validation and sanitization
- Responsible party: CTO (due to critical severity)
- Timeline: 24-48 hours
- 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 code reviews and vulnerability assessments
- Logging and alerting mechanisms to detect suspicious activity
- Continuous monitoring of web applications and development environments

- Annual penetration testing and security audits to identify and remediate vulnerabilities
- The security team will review logs and alerts regularly to detect potential security incidents and respond accordingly.

Policy #11: DAST

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

SP-2023-001: SQL Injection Protection Policy

RISK STATEMENT

This policy aims to mitigate the risk of SQL injection attacks on our web applications. SQL injection is a type of cyber attack where an attacker injects malicious SQL code into a web application's database, potentially leading to unauthorized data access or modification. If exploited, this vulnerability could result in the theft of sensitive user data, disruption of business operations, and reputational damage. The affected systems/data/users include our web application's database, user accounts, and sensitive business data.

COMPLIANCE MAPPING

This policy aligns with the following compliance requirements:

* ISO 27001: A.8.8 (Management of technical vulnerabilities), A.8.16 (Monitoring activities), A.8.23 (Web filtering), A.5.33 (Protection of records), and A.6.1 (Screening)
* NIST Cybersecurity Framework: PR.DS-5 (Protect the confidentiality, integrity, and availability of data)
* CWE: CWE-89 (Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting'))
* OWASP: A1:2017 (Injection)

POLICY REQUIREMENTS

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

1. **Input Validation**: All user input must be validated on the server-side using a whitelist approach to ensure that only expected input is processed.
2. **Parameterized Queries**: Prepared statements with parameterized queries must be used instead of building SQL queries using string concatenation.
3. **Error Handling**: Error messages must be sanitized to prevent the disclosure of sensitive information.
4. **Logging and Monitoring**: All database queries and errors must be logged and monitored for suspicious activity.
5. **Code Review**: Regular code reviews must be conducted to ensure that all code adheres to the policy requirements.

Success criteria:

* All user input is validated on the server-side.
* Prepared statements with parameterized queries are used for all database queries.
* Error messages are sanitized to prevent sensitive information disclosure.
* All database queries and errors are logged and monitored.

REMEDIATION PLAN

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

* Responsible party: Dev Lead
* Timeline: 2 weeks
* Verification steps: Code review, re-scan, and penetration testing

MONITORING AND DETECTION

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

* Regular code reviews and security audits must be conducted.
* Logging and monitoring must be implemented to detect suspicious activity.
* Continuous integration and continuous deployment (CI/CD) pipelines must be used to ensure that all code changes are thoroughly tested and validated.

By implementing these measures, we can effectively mitigate the risk of SQL injection attacks and protect our web applications, user data, and business operations.

Policy #12: DAST

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

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

RISK STATEMENT

The Cross-Site Scripting (XSS) vulnerability in the `http://testapp.local:3000/search` endpoint poses a medium-severity risk to the 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 system compromise. This vulnerability affects the `testapp.local` system, which handles sensitive user data and provides critical functionality to customers.

COMPLIANCE MAPPING

This policy aligns with the following compliance requirements:

* ISO 27001 Annex A controls:
+ A.8.23: Web filtering
+ A.8.28: Secure coding
+ A.8.29: Security testing in development and acceptance
+ A.8.8: Management of technical vulnerabilities
* NIST Cybersecurity Framework:
+ PR.DS-5: Implement secure coding practices
+ PR.DS-6: Implement secure coding standards
* Industry standards:
+ CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
+ OWASP Top 10: Cross-Site Scripting (XSS)

POLICY REQUIREMENTS

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

1. **Input Validation and Sanitization**: Ensure that all user input is validated, filtered, and sanitized to prevent malicious code injection.
2. **Secure Coding Practices**: Implement secure coding practices, such as using vetted libraries and frameworks, and following secure coding standards.
3. **Security Testing**: Conduct regular security testing, including penetration testing and code reviews, to identify and remediate vulnerabilities.
4. **Logging and Alerting**: Implement logging and alerting mechanisms to detect and respond to potential XSS attacks.

REMEDIATION PLAN

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

1. **Validate and Sanitize User Input**: Update the `http://testapp.local:3000/search` endpoint to validate and sanitize user input.

2. **Implement Secure Coding Practices**: Review and update the codebase to ensure secure coding practices are followed.
3. **Conduct Security Testing**: Conduct a penetration test and code review to identify and remediate any remaining vulnerabilities.

Responsible party: Dev Lead
Timeline: 2 weeks
Verification steps: Code review, re-scan, penetration test

MONITORING AND DETECTION

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

1. **Regular Security Scans**: Conduct regular security scans to identify potential vulnerabilities.
2. **Logging and Alerting**: Implement logging and alerting mechanisms to detect and respond to potential XSS attacks.
3. **Continuous Monitoring**: Continuously monitor the system for signs of malicious activity.

By implementing these security controls and remediation actions, the organization can prevent Cross-Site Scripting (XSS) vulnerabilities and ensure the security and integrity of sensitive user data.

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

Policy Identifier: SP-2023-001
Policy Title: Cross-Site Request Forgery (CSRF) Protection Policy

Risk Statement:

This policy aims to mitigate the risk of Cross-Site Request Forgery (CSRF) attacks, which can compromise the security of our web applications and potentially lead to unauthorized data modifications or financial losses. If exploited, a CSRF attack can result in the theft of sensitive user data, disruption of business operations, and damage to our reputation. The affected systems, data, and users include all web applications, user accounts, and sensitive data stored within our database.

Compliance Mapping:

* NIST CSF: PR.DS-5, ID.RA-5, RC.CO-3
* ISO 27001: A.8.23, A.8.8, A.5.14, A.6.4
* Industry Standards: OWASP, CWE-352

Policy Requirements:

To ensure the protection of our web applications from CSRF attacks, the following security controls must be implemented:

1. **Technical Requirements:**
* Use a vetted library or framework that provides anti-CSRF tokens, such as OWASP CSRFGuard.
* Ensure that all HTML submission forms include anti-CSRF tokens.
* Regularly review and update our web application code to prevent cross-site scripting issues.
2. **Procedural Requirements:**
* Conduct regular security audits and vulnerability assessments to identify potential CSRF vulnerabilities.
* Implement a secure coding practice that includes input validation, output encoding, and secure cookie handling.
* Establish a disciplinary process to address personnel who commit information security policy violations.

Success Criteria and Validation Methods:

* Regular security audits and vulnerability assessments will be conducted quarterly to identify potential CSRF vulnerabilities.
* The implementation of anti-CSRF tokens and secure coding practices will be validated through code reviews and penetration testing.
* The effectiveness of our CSRF protection measures will be measured through regular security monitoring and incident response exercises.

Remediation Plan:

* **Responsible Party:** Dev Lead
* **Timeline:** 2 weeks
* **Technical Actions:**
+ Update our web application code to include anti-CSRF tokens.
+ Implement secure coding practices to prevent cross-site scripting issues.
+ Conduct regular security audits and vulnerability assessments.
* **Verification Steps:**
+ Code review and re-scan for CSRF vulnerabilities.
+ Penetration testing to validate CSRF protection measures.

Monitoring and Detection:

To detect similar vulnerabilities in the future, we will:

* Regularly scan our web applications for CSRF vulnerabilities using automated tools.
* Conduct regular security audits and vulnerability assessments.
* Implement logging and alerting requirements to detect potential CSRF attacks.
* Continuously monitor our web applications for signs of CSRF attacks and respond promptly to any incidents.

By implementing this policy, we aim to protect our web applications from CSRF attacks and ensure the confidentiality, integrity, and availability of our sensitive data.

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-severity risk to the organization. If exploited, an attacker could intercept and reuse user credentials, compromising the confidentiality and integrity of sensitive information. This vulnerability affects all users accessing the application via the `http://testapp.local:3000/api/login` endpoint, potentially leading to unauthorized access to sensitive data and systems.

Compliance Mapping:

* NIST Cybersecurity Framework (CSF):
+ PR.DS-5: Implement secure protocols for sensitive data transmission
+ ID.AM-1: Identify and authenticate users
* ISO 27001 Annex A controls:
+ A.8.5: Secure authentication
+ A.7.4: Physical security monitoring
+ A.8.21: Security of network services
+ A.8.1: User endpoint devices
+ A.7.14: Secure disposal or re-use of equipment
* Industry standards:
+ CWE-287: Improper Authentication
+ OWASP: Authentication and Session Management (A5)

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 2.0 or JWT-based authentication.
2. **Procedural Requirements:**
* Conduct regular security audits to identify and remediate vulnerabilities.
* Implement a secure coding practice to prevent similar vulnerabilities in the future.
3. **Success Criteria:**
* All authentication requests are made over HTTPS.
* A stronger authentication mechanism is implemented and configured correctly.
* Regular security audits identify and remediate vulnerabilities.

Remediation Plan:

* **Responsible Party:** Dev Lead
* **Timeline:** 2 weeks
* **Technical Actions:**
1. Update the application to use HTTPS for all authentication requests.

2. Implement a stronger authentication mechanism (OAuth 2.0 or JWT-based).
3. Conduct code reviews to ensure secure coding practices.
* **Verification Steps:**
1. Conduct a code review to ensure secure coding practices.
2. Perform a re-scan using a vulnerability scanner to verify the remediation.
3. Conduct a penetration test to simulate an attack and verify the effectiveness of the remediation.

Monitoring and Detection:

To detect similar vulnerabilities in the future:

1. **Logging and Alerting Requirements:**
* Configure logging to capture authentication requests and responses.
* Set up alerts for suspicious authentication activity.
2. **Continuous Monitoring Strategies:**
* Conduct regular security audits to identify and remediate vulnerabilities.
* Implement a secure coding practice to prevent similar vulnerabilities in the future.

By implementing these security controls and following this remediation plan, we can mitigate the risk of unauthorized access to sensitive data and systems.

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 Implementation

Risk Statement:

The absence of the secure flag on cookies poses a medium-severity risk to our organization's security. If exploited, this vulnerability could allow unauthorized access to sensitive information, compromising user confidentiality and potentially leading to data breaches. This vulnerability affects all users interacting with our web application, particularly those using unencrypted connections.

Compliance Mapping:

* NIST Cybersecurity Framework (CSF):
+ Identify (ID): PR.DS-5 (Protect Data)
+ Protect (PR): PR.DS-5 (Protect Data)
+ Detect (DE): DE.AE-1 (Anomalous Activity Detection)
+ Respond (RS): RS.MI-2 (Incident Management)
* 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)

Policy Requirements:

1. **Secure Cookie Flag Implementation:** Ensure that all cookies containing sensitive information or session tokens are passed using an encrypted channel. This includes setting the secure flag for such cookies.
2. **Technical Requirements:**
* Update the web application to set the secure flag for cookies containing sensitive information or session tokens.
* Implement HTTPS (TLS) for all connections to the web application.
3. **Procedural Requirements:**
* Review and update the web application's cookie management policy to reflect the secure flag implementation.
* Conduct regular security audits to ensure compliance with this policy.

Success Criteria and Validation Methods:

* The secure flag is set for all cookies containing sensitive information or session tokens.

* HTTPS (TLS) is enabled for all connections to the web application.
* Regular security audits demonstrate compliance with this policy.

Remediation Plan:

* **Responsible Party:** Development Lead
* **Timeline:** 2 weeks
* **Technical Actions:**
1. Update the web application to set the secure flag for cookies containing sensitive information or session tokens.
2. Implement HTTPS (TLS) for all connections to the web application.
* **Verification Steps:**
1. Code review to ensure the secure flag is set correctly.
2. Re-scan the web application for vulnerabilities to ensure compliance.

Monitoring and Detection:

* Regularly review web application logs for suspicious activity related to cookie management.
* Implement logging and alerting mechanisms to detect similar vulnerabilities in the future.
* Conduct continuous monitoring of the web application's security posture to ensure compliance with this policy.