AI-Powered Security Policy Report

Generated: 2025-11-06 21:54:16

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 disruption and potential regulatory non-compliance. The affected systems include the 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 remediate potential vulnerabilities.
Success criteria include:
- All user input is validated and sanitized.
- Parameterized queries or prepared statements are used for all SQL queries.
- Security testing and code reviews are performed regularly.

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 remediate any similar vulnerabilities.
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 measures must be implemented:
- Regular security testing and code reviews.
- Logging and alerting for suspicious database activity.
- Continuous monitoring of application security using industry-standard tools and techniques.
By implementing these measures, the organization can reduce the risk of SQL injection attacks and protect sensitive user data.

Policy #2: SAST

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

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

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

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

POLICY REQUIREMENTS
To mitigate the risk of XSS attacks, the organization will implement the following security controls:
- Validate and sanitize all user input data
- Use a Content Security Policy (CSP) to define allowed sources of content
- Implement a Web Application Firewall (WAF) to detect and prevent XSS attacks
- Conduct regular security testing and code reviews to identify vulnerabilities
Success criteria will be measured by the absence of XSS vulnerabilities in security scans and penetration tests. Validation methods will include code reviews, security testing, and verification of CSP and WAF configurations.

REMEDIATION PLAN
To remediate the identified vulnerability, the following technical actions are required:
- Review and fix the template variable usage in the 'href' attribute (src/views/profile.mustache, line 23)
- Implement input validation and sanitization for all user input data
- Configure CSP and WAF to prevent XSS attacks
The responsible party for remediation is the Development Lead. The timeline for remediation is 2 weeks. Verification steps will include a code review, security testing, and verification of CSP and WAF configurations.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the organization will:
- Conduct regular security scans and penetration tests
- Monitor web application logs for suspicious activity
- Implement a threat intelligence program to stay informed about emerging threats
Continuous monitoring strategies will include regular code reviews, security testing, and verification of security controls. Logging and alerting requirements will be configured to detect and respond to potential XSS attacks.

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.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:
- All user input used to construct file paths must be properly sanitized and validated.
- Implement a whitelist approach to only allow access to intended directories and files.
- Conduct regular security testing and code reviews to identify and address similar vulnerabilities.
Success criteria include:
- Verification of input validation and sanitization mechanisms through code review and penetration testing.
- Confirmation of restricted access to sensitive files and directories.

REMEDIATION PLAN
To remediate the identified vulnerability:
- Technical actions: Review and fix the vulnerable code in src/routes/files.js (line 67) to properly sanitize user input and restrict file access.
- Responsible party: Security Lead
- Timeline: 1 week
- Verification steps: Conduct a code review and re-scan the application using a vulnerability scanner to ensure the fix is effective.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future:
- Implement logging and alerting mechanisms to monitor file access and user input.
- Conduct regular security audits and penetration testing to identify potential vulnerabilities.
- Utilize static application security testing (SAST) tools to analyze code for security weaknesses.
Continuous monitoring strategies include regular review of security logs and vulnerability scans to ensure the security controls are effective and up-to-date.

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.8.1 (User endpoint devices), A.8.5 (Secure authentication), A.8.23 (Web filtering)
- Industry standards: OWASP A2:2021 (Cryptographic Failures), 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 include verification of 'httpOnly' flag presence in all cookie sessions, and validation of secure authentication mechanisms.
- Technical requirements include updating the session.js file to include the 'httpOnly' flag, and implementing secure authentication protocols.

REMEDIATION PLAN
To remediate this vulnerability, the following actions are required:
- Technical action: Update the session.js file to include the 'httpOnly' flag (line 12).
- Responsible party: Dev Lead
- Timeline: 2 weeks
- Verification steps: Code review, re-scan, and penetration testing to ensure the 'httpOnly' flag is present and secure authentication mechanisms are in place.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures must be implemented:
- Regular security testing and code reviews to identify vulnerabilities in session management and authentication mechanisms.
- Logging and alerting requirements include monitoring for suspicious activity and unauthorized access attempts.
- Continuous monitoring strategies include regular vulnerability scans, penetration testing, and code reviews to ensure the security of our web application and sensitive data.

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 the security of our systems and data. If exploited, an attacker could gain unauthorized access to sensitive information, compromising the confidentiality, integrity, and availability of our systems. This could lead to financial losses, reputational damage, and legal liabilities. The affected systems include our production environment, and the impacted data includes sensitive user information and business-critical data.

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

POLICY REQUIREMENTS
To mitigate the risk of hardcoded secrets, the following security controls must be implemented:
- All secrets must be stored in environment variables or a secure secret management system.
- Code reviews must be performed to detect and remove hardcoded secrets.
- Secure coding principles must be applied to software development.
- Success criteria: All secrets are stored securely, and code reviews are performed regularly.
- Validation methods: Regular code reviews, security audits, and vulnerability scans.

REMEDIATION PLAN
To remediate the identified 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 resolved.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future:
- Regular code reviews must be performed to detect hardcoded secrets.
- Logging and alerting requirements: All code changes must be logged, and alerts must be triggered for suspicious activity.
- Continuous monitoring strategies: Regular security audits, vulnerability scans, and penetration tests must be performed to identify and remediate vulnerabilities.
- The Security Lead is responsible for ensuring that these measures are implemented and effective.

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, leading to unintended behavior and security breaches. This vulnerability affects all systems and applications utilizing the lodash library, potentially exposing sensitive user data and intellectual property.

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

POLICY REQUIREMENTS
To mitigate the risks associated with the lodash prototype pollution vulnerability, the following security controls must be implemented:
- Utilize a secure version of the lodash library that has addressed the prototype pollution vulnerability.
- Implement input validation and sanitization for all user-provided data to prevent malicious input.
- Conduct regular security audits and vulnerability assessments to identify and remediate similar issues.
- Success criteria: Verification of the secure lodash version, validation of input validation and sanitization controls, and successful completion of security audits.
- Validation methods: Code review, vulnerability scanning, and penetration testing.

REMEDIATION PLAN
To remediate this vulnerability, the following technical actions are required:
- Update the lodash library to a secure version.
- Implement input validation and sanitization controls.
- Conduct a thorough security audit to identify similar vulnerabilities.
- 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 logging and alerting for suspicious activity related to the lodash library.
- Continuous monitoring of vulnerability feeds and security advisories for updates on the lodash library.
- Quarterly security audits and vulnerability assessments to identify and remediate similar issues.
- Implementation of a Web Application Firewall (WAF) to detect and prevent malicious input.

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, compromising sensitive information and damaging our reputation. If exploited, this vulnerability could lead to phishing attacks, malware distribution, and unauthorized access to our systems. The affected systems include all 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-3 (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 malicious redirects.
- Conduct regular security audits and penetration testing to identify potential vulnerabilities.
Success criteria include:
- No detected open redirect vulnerabilities in our web applications.
- All user-inputted URLs are validated and authorized.
Validation methods include:
- Code reviews to ensure proper URL validation.
- Penetration testing to simulate attacks and identify vulnerabilities.

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.
The responsible party is the Development Lead, and the timeline for remediation is 2 weeks.
Verification steps include:
- Code review to ensure proper implementation of URL validation and authorization.
- Penetration testing to simulate attacks and identify vulnerabilities.

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.
- Logging and alerting mechanisms to detect suspicious activity.

- Continuous monitoring of our web applications for potential vulnerabilities.
Logging requirements include:
- All user-inputted URLs must be logged and monitored.
- Any suspicious activity must be alerted to the security team.
Continuous monitoring strategies include:
- Regular code reviews and security audits.
- Penetration testing and vulnerability scanning.

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 access sensitive internal systems, data, and services, potentially leading to unauthorized data breaches, lateral movement, and disruption of critical business operations. The affected systems include all web applications utilizing the axios library, and the impacted data includes sensitive user information and internal business 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.8.23 (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-inputted URLs to prevent unauthorized requests.
- Implement web filtering to restrict access to internal systems and services.
- Conduct regular vulnerability scans and penetration testing to identify potential SSRF vulnerabilities.
- Success criteria: No SSRF vulnerabilities detected during vulnerability scans and penetration testing.
- Validation methods: Regular code reviews, vulnerability scans, and penetration testing.

REMEDIATION PLAN
To remediate the identified SSRF vulnerability, the following technical actions are required:
- Update axios to the latest version.
- Implement URL validation and sanitization for all user-inputted URLs.
- Configure web filtering to restrict access to internal systems and services.
- 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 monitoring and detection strategies will be implemented:
- Regular logging and alerting for suspicious URL requests.
- Continuous monitoring of web application logs for potential SSRF attacks.
- Regular vulnerability scans and penetration testing to identify potential SSRF vulnerabilities.
- Implementation of a Web Application Firewall (WAF) to detect and prevent SSRF attacks.

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, integrity, and authenticity of sensitive data. If exploited, an attacker could potentially decode and manipulate JWTs, leading to unauthorized access to protected resources and data. This vulnerability affects all systems, data, and users that rely on JWT for authentication and authorization.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), PR.IP-1 (Identity and Access Control)
- ISO 27001: A.8.24 (Use of Cryptography), A.8.26 (Application Security Requirements), A.8.27 (Secure System Architecture and Engineering Principles)
- Industry Standards: OWASP A03:2021 (Injection), CWE-327 (Use of a Broken or Risky Cryptographic Algorithm)

POLICY REQUIREMENTS
To mitigate this 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
- Validate JWTs on each request to prevent token manipulation
Success criteria: All JWTs must be generated with a secure key size, and token validation must be performed on each request. Validation methods: Regular code reviews, penetration testing, and vulnerability scanning.

REMEDIATION PLAN
To remediate this vulnerability, the following technical actions are required:
- Update the JSON Web Token library to use a secure default key size
- Review and update all JWT-related code to ensure secure key management and token validation
Responsible party: Dev Lead
Timeline: 2 weeks
Verification steps: Code review, re-scan with vulnerability scanning tools, and penetration testing.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures must be implemented:
- Regular vulnerability scanning and penetration testing
- Logging and alerting on all JWT-related errors and exceptions
- Continuous monitoring of application security and cryptography practices
Logging requirements: All JWT-related errors and exceptions must be logged and alerted to the security team. Continuous monitoring strategies: Regular code reviews, security audits, and compliance checks.

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 legal liabilities. The affected systems include our web applications and APIs that utilize the minimist library, potentially impacting all users and data stored within these systems.

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:
- Regularly update and patch the minimist library to the latest version.
- Implement input validation and sanitization for all user-provided data.
- Conduct regular code reviews and security audits to identify potential vulnerabilities.
- Success criteria: No prototype pollution vulnerabilities detected during code reviews and security audits.
- Validation methods: Automated code scanning tools and manual code reviews.

REMEDIATION PLAN
To remediate the minimist prototype pollution vulnerability:
- Technical actions: Update the minimist library to the latest version, implement input validation and sanitization, and conduct a thorough code review.
- Responsible party: CTO (due to critical severity)
- Timeline: Completion within 24-48 hours
- Verification steps: Conduct a code review, re-scan the application using automated tools, and perform a penetration test to ensure the vulnerability is fully remediated.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future:
- Implement continuous monitoring and logging of application security events.
- Configure alerting systems to notify the security team of potential security incidents.
- Conduct regular security audits and code reviews to identify potential vulnerabilities.
- Utilize automated code scanning tools to detect prototype pollution vulnerabilities and other security weaknesses.
By following this policy, we can ensure the security and integrity of our systems and data, protecting against the minimist prototype pollution vulnerability and similar threats.

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 application, which may compromise sensitive data and lead to unauthorized access. If exploited, this vulnerability could result in data breaches, financial losses, and reputational damage. The affected systems include our web application, databases, and user accounts.

COMPLIANCE MAPPING

This policy aligns with the following standards and controls:

* NIST Cybersecurity Framework (CSF):
+ PR.DS-5: Protect against unauthorized access to or use of the information system that could be facilitated by malicious insiders, or external entities with or without malicious intentions, including unauthorized external users.
+ ID.AM-1: Identify the organization's information system and the related mission, business functions, or processes.
* ISO 27001 Annex A controls:
+ A.8.8: Management of technical vulnerabilities
+ A.8.16: Monitoring activities
+ A.8.23: Web filtering
+ A.5.33: Protection of records
+ A.6.1: Screening
* Industry standards:
+ CWE-89: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
+ OWASP: SQL Injection Prevention Cheat Sheet

POLICY REQUIREMENTS

To mitigate the risk of SQL injection attacks, the following security controls must be implemented:

1. **Input Validation and Sanitization**: All user input must be validated and sanitized on the server-side to prevent malicious data from being injected into SQL queries.
2. **Parameterized Queries**: Prepared statements with parameterized queries must be used instead of string concatenation to build SQL queries.
3. **Regular Code Reviews**: Regular code reviews must be conducted to identify and address potential vulnerabilities.
4. **Logging and Alerting**: Logging and alerting mechanisms must be implemented to detect and respond to potential SQL injection attacks.

Success Criteria and Validation Methods

* Input validation and sanitization must be implemented for all user input.
* Parameterized queries must be used for all database interactions.

* Regular code reviews must be conducted quarterly.
* Logging and alerting mechanisms must be implemented and tested quarterly.

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 with a web application scanner
+ Penetration testing

MONITORING AND DETECTION

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

* Regular web application scanning
* Regular code reviews
* Logging and alerting mechanisms
* Continuous monitoring of system logs and network traffic

By implementing these security controls and monitoring strategies, we can mitigate the risk of SQL injection attacks and protect our sensitive data and systems.

Policy #12: DAST

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

POLICY IDENTIFIER:
SP-2023-001: Cross-Site Scripting (XSS) Vulnerability Prevention and Mitigation

RISK STATEMENT:
The Cross-Site Scripting (XSS) vulnerability poses a medium-risk threat to the organization's web applications, potentially allowing attackers to inject malicious code into user browsers. If exploited, this vulnerability could lead to unauthorized access, data theft, or reputational damage. The affected systems include web applications, user data, and sensitive information stored within these applications.

COMPLIANCE MAPPING:

- **NIST Cybersecurity Framework (CSF):**
- PR.DS-5: Implement secure coding practices to prevent vulnerabilities.
- PR.DS-6: Use secure coding techniques to prevent vulnerabilities.
- **ISO 27001 Annex A:**
- A.8.23: Web filtering to reduce exposure to malicious content.
- A.8.28: Secure coding principles to be applied in software development.
- A.8.29: Security testing processes to be defined and implemented in the development life cycle.
- A.8.8: Management of technical vulnerabilities to be performed.
- **Industry Standards:**
- CWE-79: Cross-Site Scripting (XSS) vulnerability.
- OWASP: Secure Coding Practices.

POLICY REQUIREMENTS:

1. **Input Validation and Sanitization:** Ensure all user input is validated, filtered, and sanitized to prevent XSS attacks. This includes converting special characters to HTML entities.
2. **Secure Coding Practices:** Implement secure coding practices, such as using vetted libraries and frameworks, to prevent XSS vulnerabilities.
3. **Security Testing:** Define and implement security testing processes in the development life cycle to detect and prevent XSS vulnerabilities.
4. **Web Filtering:** Implement web filtering to reduce exposure to malicious content.

Success Criteria:

- All web applications will be free from XSS vulnerabilities.
- Secure coding practices will be implemented and enforced.
- Regular security testing will be performed to detect and prevent XSS vulnerabilities.

REMEDIATION PLAN:

1. **Responsible Party:** Development Lead
2. **Timeline:** 2 weeks
3. **Technical Actions:**
- Review and update web application code to implement input validation and sanitization.
- Implement secure coding practices, such as using vetted libraries and frameworks.

- Perform security testing to detect and prevent XSS vulnerabilities.
4. **Verification Steps:**
- Code review to ensure secure coding practices are implemented.
- Re-scan web applications for XSS vulnerabilities.
- Perform penetration testing to verify the effectiveness of security measures.

MONITORING AND DETECTION:

1. **Logging and Alerting:** Implement logging and alerting mechanisms to detect XSS attacks and vulnerabilities.
2. **Continuous Monitoring:** Perform regular security testing and vulnerability scanning to detect and prevent XSS vulnerabilities.
3. **Incident Response:** Establish an incident response plan to respond to XSS attacks and 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 information. This vulnerability affects our web application's API, specifically the `http://testapp.local:3000/api/profile/update` endpoint, and poses a medium risk to our organization.

COMPLIANCE MAPPING

This policy aligns with 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 Cybersecurity Framework (CSF): PR.DS-5 (Protect the confidentiality, integrity, and availability of information), and RC.CO (Recovery activities)
* Industry standards: OWASP (OWASP CSRFGuard), CWE (CWE-352)

POLICY REQUIREMENTS

To mitigate the risk of CSRF attacks, the following security controls must be implemented:

1. **Technical Requirements**: Implement a vetted library or framework (e.g., OWASP CSRFGuard) that prevents CSRF attacks. 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.
3. **Success Criteria**: The application must pass a security audit and penetration testing to ensure that CSRF attacks are mitigated.

REMEDIATION PLAN

To remediate 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 testing

MONITORING AND DETECTION

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

1. **Logging and Alerting**: Configure logging and alerting mechanisms to detect potential CSRF attacks.
2. **Continuous Monitoring**: Conduct regular security audits and penetration testing to identify and

address potential vulnerabilities.
3. **Code Review**: Conduct regular code reviews to ensure that the application is free of CSRF vulnerabilities.

DISCIPLINARY PROCESS

In the event of a CSRF attack, a disciplinary process will be followed to take actions against personnel and other relevant interested parties who have committed an information security policy violation.

REVIEW AND UPDATE

This policy will be reviewed and updated annually or as needed to ensure that it remains effective in mitigating the risk of CSRF attacks.

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 our organization. If exploited, an unauthorized party could intercept and reuse user credentials, compromising the confidentiality and integrity of our systems and data. This vulnerability affects our web application's login functionality, potentially impacting all users who access the system through the affected endpoint.

Compliance Mapping:

* NIST Cybersecurity Framework (CSF):
+ PR.DS-5: Implement secure protocols for authentication and key exchange
+ PR.DS-6: Implement secure protocols for data transmission
* ISO 27001 Annex A controls:
+ A.8.5: Secure authentication
+ A.8.21: Security of network services
+ A.8.1: User endpoint devices
+ A.7.14: Secure disposal or re-use of equipment
* Industry standards:
+ CWE-287: Improper Authentication
+ OWASP: Authentication and Session Management

Policy Requirements:

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

1. **Technical Requirements:**
* Update the login endpoint to use HTTPS (TLS 1.2 or later) for all connections.
* Implement a stronger authentication mechanism, such as OAuth or JWT-based authentication.
2. **Procedural Requirements:**
* Conduct regular security audits to identify and address potential vulnerabilities.
* Develop and maintain a secure coding standard for all development teams.
3. **Success Criteria:**
* The login endpoint is updated to use HTTPS within 2 weeks.
* A stronger authentication mechanism is implemented within 4 weeks.
* Regular security audits are conducted quarterly.

Remediation Plan:

* **Responsible Party:** Development Lead
* **Timeline:** 2 weeks
* **Technical Actions:**
+ Update the login endpoint to use HTTPS.
+ Implement a stronger authentication mechanism.

* **Verification Steps:**
+ Code review by the development team.
+ Re-scan the application using a web application scanner (e.g., OWASP ZAP).
+ Conduct a penetration test to verify the effectiveness of the remediation.

Monitoring and Detection:

To detect similar vulnerabilities in the future, we will:

1. **Implement Logging and Alerting:**
* Configure the web application firewall (WAF) to log and alert on potential authentication-related attacks.
* Implement a security information and event management (SIEM) system to monitor and alert on suspicious activity.
2. **Conduct Continuous Monitoring:**
* Regularly scan the application for vulnerabilities using a web application scanner.
* Conduct penetration tests to verify the effectiveness of our security controls.
* Review and update our secure coding standard regularly.

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 Policy

Risk Statement:

The Secure Cookie Flag policy aims to mitigate the risk of sensitive information exposure due to insecure cookie transmission. If exploited, this vulnerability can lead to unauthorized access to user sessions, compromising confidentiality and integrity. Potential consequences include:

* Unauthorized access to sensitive data
* Compromised user sessions
* Reputation damage due to data breaches

Affected systems/data/users:

* All web applications using cookies for session management
* Users with sensitive information stored in cookies

Compliance Mapping:

* NIST Cybersecurity Framework (CSF):
+ PR.DS-5: Protect data in use
+ PR.DS-6: Protect data in transit
* 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 Protocol
+ OWASP: Secure Cookies

Policy Requirements:

To ensure secure cookie transmission, the following requirements must be implemented:

1. **Technical Requirements:**
* Set the secure flag for cookies containing sensitive information or session tokens.
* Ensure that cookies are transmitted over encrypted channels (HTTPS).
2. **Procedural Requirements:**
* Review and update cookie-related code to ensure secure flag is set.
* Conduct regular security audits to detect similar vulnerabilities.

3. **Success Criteria:**
* 100% of cookies containing sensitive information or session tokens have the secure flag set.
* No instances of insecure cookie transmission are detected during security audits.

Remediation Plan:

* **Responsible Party:** Dev Lead
* **Timeline:** 2 weeks
* **Technical Actions:**
1. Review and update cookie-related code to set the secure flag.
2. Conduct a code review to ensure secure flag is set for all relevant cookies.
3. Re-scan the application for similar vulnerabilities.
* **Verification Steps:**
1. Conduct a code review to ensure secure flag is set.
2. Re-scan the application for similar vulnerabilities.

Monitoring and Detection:

To detect similar vulnerabilities in the future:

* Regularly conduct security audits and vulnerability scans.
* Implement logging and alerting mechanisms to detect insecure cookie transmission.
* Continuously monitor application code and configuration for changes that may introduce similar vulnerabilities.

By implementing this policy, we ensure the secure transmission of sensitive information and protect user sessions from unauthorized access.