AI-Powered Security Policy Report

Generated: 2025-11-06 16:19:27

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 management application, and the impacted users are all registered users of 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 potential vulnerabilities.
Success criteria include:
- All user input is validated and sanitized.
- Parameterized queries or prepared statements are used for all SQL queries.
- Regular security testing and code reviews are performed.

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 a Cross-Site Scripting (XSS) attack, which could allow malicious actors to inject malicious code into our web application, potentially leading to unauthorized access to sensitive data, theft of user credentials, or disruption of business operations. This vulnerability affects our web application, specifically the profile page, and could impact all users who access this page. If exploited, this vulnerability could result in significant reputational damage, financial loss, and legal liability.

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

POLICY REQUIREMENTS
To mitigate the risk of XSS attacks, the following security controls must be implemented:
- All user input must be validated and sanitized before being used in web page generation.
- Template variables used in anchor tags with the 'href' attribute must be properly encoded to prevent injection of malicious code.
- Success criteria: All user input is validated and sanitized, and no XSS vulnerabilities are detected during regular security testing.
- Validation methods: Regular security testing, code reviews, and penetration testing.

REMEDICATION PLAN
To remediate this vulnerability, the following technical actions are required:
- Review and fix the template variable used in the anchor tag with the 'href' attribute in the profile.mustache file (line 23).
- Responsible party: Dev Lead
- Timeline: 2 weeks
- Verification steps: Code review, re-scan, and penetration testing.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures must be implemented:
- Regular security testing and code reviews to identify potential XSS vulnerabilities.
- Logging and alerting requirements: All security-related logs must be monitored and alerts must be generated in case of suspicious activity.
- Continuous monitoring strategies: Regularly review and update security policies and procedures to ensure they are aligned with industry best practices and compliance requirements.

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 and data stored within the application.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), DE.CM-1 (Anomaly Detection)
- 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), A.7.2 (Physical entry), A.8.31 (Separation of development, test and production environments)
- 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 paths
- Implementation of a whitelist 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: Code reviews, security scans, and penetration testing

REMEDIATION PLAN
To remediate the identified vulnerability:
- Specific technical actions: Review and fix the vulnerable code in src/routes/files.js (line 67) to properly sanitize user input and implement a whitelist approach for allowed file paths
- 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:
- Implement logging and alerting for suspicious file access attempts
- Conduct regular security scans and code reviews
- Utilize continuous monitoring strategies, such as anomaly detection and periodic vulnerability assessments
- Perform regular security testing, including penetration testing and code reviews, to identify and remediate vulnerabilities before they can be exploited.

Policy #4: SAST

Title: Express Cookie Session No Httponly
Severity: MEDIUM
LLM: LLaMA 3.3 70B

POLICY IDENTIFIER
SP-2024-001: Secure Session Management Policy

RISK STATEMENT
The absence of the 'httpOnly' flag in cookie sessions poses a significant risk to our organization's data security. If exploited, this vulnerability could lead to session hijacking via cross-site scripting (XSS) attacks, allowing unauthorized access to sensitive information. This could result in financial loss, reputational damage, and compromised customer data. The affected systems include our web applications, and the impacted users are our customers and employees who use these applications.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), DE.CM-8 (Vulnerability Management)
- ISO 27001: A.8.1 (User endpoint devices), A.8.23 (Web filtering), A.8.29 (Security testing in development and acceptance)
- Industry standards: OWASP A2:2021 (Broken Authentication), CWE-1004 (Sensitive Cookie Without 'HttpOnly' Flag)

POLICY REQUIREMENTS
To mitigate this risk, the following security controls must be implemented:
- All cookie sessions must be configured with the 'httpOnly' flag to prevent JavaScript access.
- Regular security testing and code reviews must be conducted to identify and remediate similar vulnerabilities.
- Success criteria: All cookie sessions are configured with the 'httpOnly' flag, and no similar vulnerabilities are detected during security testing.
- Validation methods: Code reviews, vulnerability scans, and penetration testing.

REMEDIATION PLAN
To remediate this vulnerability, the following technical actions are required:
- Review and update the session.js file to include the 'httpOnly' flag in cookie sessions.
- 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: Regularly review and update session management configurations, and conduct vulnerability scans and penetration testing to identify potential vulnerabilities.

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 the confidentiality, integrity, and availability of our systems and data. If exploited, an attacker could gain unauthorized access to sensitive information, compromising our business operations and reputation. This vulnerability affects our development, testing, and production environments, as well as our customers' and employees' personal data.

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, 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 detected in code reviews and vulnerability scans.
- Validation methods: Regular code reviews, vulnerability scans, and penetration testing.

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 testing to ensure the vulnerability is resolved.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future:
- Regular code reviews and vulnerability scans must be performed.
- Logging and alerting must be implemented to detect potential security incidents.
- Continuous monitoring strategies must be employed to identify and remediate vulnerabilities in a timely manner.
- Code analysis tools must be used to detect hardcoded secrets and other security weaknesses.

Policy #6: SCA

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

POLICY IDENTIFIER
SP-2024-001: Lodash Prototype Pollution Protection Policy

RISK STATEMENT
The organization faces a significant risk due to the presence of a Prototype Pollution vulnerability in the lodash library, categorized as CWE-1321. This vulnerability can lead to unauthorized modification of application behavior, potentially resulting in data breaches, system compromise, or disruption of critical services. The affected systems include all applications utilizing the vulnerable lodash version, impacting both internal users and external customers. If exploited, this vulnerability could lead to severe business consequences, including reputational damage, financial loss, and regulatory non-compliance.

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 risk associated with the lodash Prototype Pollution vulnerability, the following security controls must be implemented:
- Utilize a secure version of the lodash library, ensuring it is updated to the latest patch level.
- Implement 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 remediate similar vulnerabilities.
- Success criteria include the successful update of the lodash library and the implementation of CSP, validated through code reviews and security testing.

REMEDIATION PLAN
To address the identified vulnerability, the following remediation steps are required:
- Technical Action: Update the lodash library to the latest secure version and implement CSP.
- Responsible Party: Security Lead
- Timeline: Completion within 1 week due to the High severity of the vulnerability.
- Verification Steps: Conduct a code review to verify the update of the lodash library and the implementation of CSP. Perform a re-scan using a vulnerability scanner to ensure the vulnerability is remediated.

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

to ensure the organization's security posture remains robust against evolving threats.

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, as it could allow attackers to redirect users to malicious websites, potentially leading to phishing attacks, malware infections, or other types of cyber threats. 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 malicious redirects
- 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 testing, code reviews, and WAF logs analysis

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
- Configure the WAF to detect and prevent malicious redirects
- Responsible party: Dev Lead
- Timeline: 2 weeks
- Verification steps: Code review, security testing, and WAF logs analysis

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures must be implemented:
- Regular security testing and code reviews
- Logging and alerting: Configure the WAF and web application logs to detect and alert on potential open redirect attacks
- Continuous monitoring strategies: Implement a continuous integration and continuous deployment (CI/CD) pipeline to ensure timely updates and security patches
- Regularly review and update the policy to ensure it remains effective and aligned with industry standards and best practices.

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, gain unauthorized access to internal systems, and potentially exfiltrate sensitive data. The affected systems include all applications utilizing the axios library, and the impacted users are all internal stakeholders and external customers interacting with these applications. The potential consequences of an SSRF attack include data breaches, intellectual property theft, and reputational damage.

COMPLIANCE MAPPING
This policy aligns 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.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 to restrict access to internal systems and sensitive data
- Configure axios to use a whitelist of allowed domains and protocols
- Conduct regular security audits and penetration testing to identify potential SSRF vulnerabilities

Success criteria include:
- All axios requests are validated and sanitized
- Web filtering is enabled and configured correctly
- No SSRF vulnerabilities are detected during security audits and penetration testing

Validation methods include code reviews, security audits, and penetration testing.

REMEDIATION PLAN
To remediate the SSRF vulnerability, the following technical actions are required:
- Update axios to the latest version
- Implement input validation and sanitization for all axios requests
- Configure web filtering to restrict access to internal systems and sensitive data

The responsible party for this remediation is the Security Lead. The timeline for completion is 1 week. Verification steps include code reviews, security audits, and penetration testing.

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

- Continuous monitoring of application logs for signs of SSRF attacks

Logging requirements include:

- Log all axios requests
- Log all errors and exceptions related to axios requests
- Alert on suspicious axios requests

Continuous monitoring strategies include:

- Regularly review application logs for signs of SSRF attacks
- Conduct regular security audits and penetration testing to identify potential SSRF 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 Policy

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, data breaches, and reputational damage. This vulnerability affects all systems, data, and users that rely on JWTs for authentication and authorization.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), PR.IP-2 (Inventory of Authorized and Unauthorized Devices)
- ISO 27001: A.8.24 (Use of Cryptography), A.8.26 (Application Security Requirements), A.8.27 (Secure System Architecture and Engineering Principles), A.8.28 (Secure Coding)
- 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 JWTs (minimum 2048-bit RSA or equivalent)
- Implement secure key management practices, including key rotation and revocation
- Validate and verify JWTs on each request to prevent token manipulation
- Use a secure algorithm for signing and verifying JWTs (e.g., RS256 or ES256)

Success criteria include:
- All JWTs use a secure key size and algorithm
- Key rotation and revocation processes are in place and documented
- JWT validation and verification are performed on each request

REMEDIATION PLAN
To remediate this vulnerability, the following technical actions are required:
- Update the JSON Web Token library to use a secure key size and algorithm
- Implement secure key management practices
- Update the application code to validate and verify JWTs on each request

The responsible party for this remediation is the Development Lead. The timeline for completion is 2 weeks. Verification steps include a code review, re-scan, and penetration test to ensure the vulnerability has been fully addressed.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures will be implemented:
- Regular security audits and code reviews
- Logging and alerting for suspicious JWT activity
- Continuous monitoring of the application and its dependencies for known vulnerabilities
- Regular updates to the JSON Web Token library and other dependencies to ensure the latest security patches are applied.

Policy #10: SCA

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

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

RISK STATEMENT
The minimist prototype pollution vulnerability poses a significant risk to our organization's intellectual property and data integrity. If exploited, this vulnerability could allow unauthorized access to sensitive data, compromise our software development lifecycle, and potentially lead to malicious code execution. 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 to authorized personnel

Success criteria will be measured by:
- Successful implementation of input validation and sanitization
- Deployment of a WAF with configured rules to detect prototype pollution attacks
- Regular code reviews and vulnerability assessments with no critical findings

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 CTO is responsible for overseeing the remediation efforts, given the critical severity of the vulnerability. The timeline for remediation is 24-48 hours.

Verification steps include:
- Code review to ensure input validation and sanitization are properly implemented
- Re-scanning the application with a vulnerability scanner to confirm the vulnerability is resolved
- Conducting a penetration test to validate the WAF configuration

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures will be implemented:
- Regular logging and monitoring of web application traffic
- Alerting and notification systems for suspicious activity
- Continuous monitoring of vulnerability scanners and penetration testing results

The security team will review logs and alerts regularly to identify 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 Prevention Policy

RISK STATEMENT

The SQL injection vulnerability in the `http://testapp.local:3000/api/users/search` endpoint poses a medium-risk threat to the organization. If exploited, this vulnerability could allow unauthorized access to sensitive user data, potentially leading to identity theft, financial loss, or reputational damage. The affected systems, data, and users include the web application's user database, which contains sensitive information such as user credentials and personal details.

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 (Protecting Data)
* CWE: CWE-89 (Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting'))
* OWASP: A3 (Broken Authentication and Session Management)

POLICY REQUIREMENTS

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

1. **Input Validation and Sanitization**: All user input must be validated and sanitized on the server-side to prevent malicious SQL code from being injected.
2. **Parameterized Queries**: Prepared statements with parameterized queries must be used instead of building SQL queries using string concatenation.
3. **Regular Code Reviews**: Regular code reviews must be conducted to identify and address potential vulnerabilities.
4. **Logging and Alerting**: Log all SQL queries and alert the security team in case of suspicious activity.

REMEDIATION PLAN

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

1. **Update Web Application Code**: Update the web application code to use parameterized queries and input validation.
2. **Conduct Code Review**: Conduct a thorough code review to identify and address potential vulnerabilities.
3. **Implement Logging and Alerting**: Implement logging and alerting mechanisms to detect suspicious activity.

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 monitoring and detection strategies must be implemented:

1. **Regular Vulnerability Scans**: Conduct regular vulnerability scans to identify potential vulnerabilities.
2. **Logging and Alerting**: Log all SQL queries and alert the security team in case of suspicious activity.
3. **Continuous Monitoring**: Continuously monitor the web application for anomalous behavior and potential security incidents.

By implementing these security controls and remediation actions, we can prevent SQL injection attacks and protect sensitive user data.

Policy #12: DAST

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

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

Policy Identifier:
- Unique policy ID: SP-2023-001
- Policy title: Cross-Site Scripting (XSS) Vulnerability Prevention

Risk Statement:
The Cross-Site Scripting (XSS) vulnerability poses a medium-risk threat to our organization's web applications. If exploited, this vulnerability could allow attackers to inject malicious code into our users' browsers, potentially leading to unauthorized access, data theft, or system compromise. This vulnerability affects our web application's search functionality, which is used by employees and customers alike. Successful exploitation could result in reputational damage, financial loss, and compromised sensitive data.

Compliance Mapping:

- NIST Cybersecurity Framework (CSF):
- PR.DS-5: Implement secure coding practices to prevent vulnerabilities.
- PR.DS-6: Use secure coding libraries and frameworks.
- ISO 27001 Annex A controls:
- A.8.23: Web filtering to reduce exposure to malicious content.
- A.8.28: Secure coding principles to prevent vulnerabilities.
- A.8.29: Security testing in development and acceptance to identify vulnerabilities.
- A.8.8: Management of technical vulnerabilities to identify and remediate vulnerabilities.
- Industry standards:
- CWE-79: Cross-Site Scripting (XSS)
- OWASP: Top 10 Web Application Security Risks

Policy Requirements:

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

1. **Secure Coding Practices:** All developers must adhere to secure coding principles, including input validation, filtering, and sanitization of user input.
2. **Vetted Libraries and Frameworks:** Only vetted libraries and frameworks that prevent XSS vulnerabilities must be used in web application development.
3. **Security Testing:** Regular security testing must be performed during the development life cycle to identify and remediate vulnerabilities.
4. **Web Filtering:** Web filtering must be implemented to reduce exposure to malicious content.

Remediation Plan:

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

1. **Responsible Party:** Development Lead

2. **Timeline:** 2 weeks
3. **Technical Actions:**
* Review and update the web application's search functionality to prevent XSS vulnerabilities.
* Implement input validation, filtering, and sanitization of user input.
* Use a vetted library or framework that prevents XSS vulnerabilities.
4. **Verification Steps:**
* Code review to ensure secure coding practices are followed.
* Re-scan the web application for vulnerabilities after remediation.
* Perform penetration testing to verify the effectiveness of remediation efforts.

Monitoring and Detection:

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

1. **Logging and Alerting:** Regular logging and alerting must be performed to detect suspicious activity.
2. **Continuous Monitoring:** Continuous monitoring of web application vulnerabilities must be performed using automated tools and manual testing.
3. **Security Testing:** Regular security testing must be performed during the development life cycle to identify and remediate vulnerabilities.

Policy #13: DAST

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

POLICY IDENTIFIER:
SP-2023-001 - Cross-Site Request Forgery (CSRF) Protection Policy

RISK STATEMENT:
The absence of Anti-CSRF tokens in our web application's HTML submission forms poses a medium-severity risk to our organization. If exploited, this vulnerability could allow attackers to trick users into performing unintended actions, potentially leading to financial losses, data breaches, or reputational damage. The affected systems include our web application's API endpoint (http://testapp.local:3000/api/profile/update), which handles user profile updates. This vulnerability could impact users who interact with our web application, compromising their sensitive information and trust in our services.

COMPLIANCE MAPPING:
This policy aligns with the following compliance requirements:

* NIST Cybersecurity Framework (CSF):
+ PR.DS-5: Implement secure coding practices to prevent vulnerabilities.
+ RC.CO: Communicate recovery activities to internal and external stakeholders.
* ISO 27001 Annex A controls:
+ A.8.23: Web filtering to manage access to external websites and reduce exposure to malicious content.
+ A.8.8: Management of technical vulnerabilities to identify and address potential weaknesses.
+ A.5.14: Information transfer to ensure secure data exchange between systems and parties.
+ A.6.4: Disciplinary process to address information security policy violations.
* Industry standards:
+ OWASP: Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet.
+ CWE: CWE-352: Cross-Site Request Forgery (CSRF).

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

1. **Technical Requirements:**
* Integrate a vetted library or framework that provides Anti-CSRF tokens, such as OWASP CSRFGuard.
* Ensure the application is free of cross-site scripting (XSS) issues to prevent CSRF defenses from being bypassed.
2. **Procedural Requirements:**
* Conduct regular code reviews to identify and address potential vulnerabilities.
* Implement logging and alerting mechanisms to detect and respond to CSRF attacks.
3. **Success Criteria:**
* The web application's API endpoint (http://testapp.local:3000/api/profile/update) is protected with Anti-CSRF tokens.
* Regular code reviews and penetration testing identify no CSRF vulnerabilities.

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

1. **Responsible Party:** Development Lead
2. **Timeline:** 2 weeks
3. **Verification Steps:**
* Code review to ensure Anti-CSRF tokens are implemented correctly.
* Re-scan the application for CSRF vulnerabilities using a reputable tool.
* Conduct a penetration test to simulate a CSRF attack.

MONITORING AND DETECTION:
To detect similar vulnerabilities in the future, we will:

1. **Regularly Scan the Application:** Use a reputable vulnerability scanner to identify potential CSRF vulnerabilities.
2. **Implement Logging and Alerting:** Configure logging and alerting mechanisms to detect and respond to CSRF attacks.
3. **Conduct Regular Code Reviews:** Perform regular code reviews to identify and address potential vulnerabilities.

Policy #14: DAST

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

SP-2023-001: Secure Authentication over Unsecured Connections

Risk Statement:

The use of HTTP basic or digest authentication over an unsecured connection poses a significant risk to the organization's sensitive data and user credentials. If exploited, this vulnerability could allow unauthorized access to our systems, leading to potential data breaches, financial losses, and reputational damage. The affected systems, data, and users include all employees, contractors, and customers who use our web application for authentication purposes.

Compliance Mapping:

* NIST CSF: PR.DS-5 (Protect Information and Systems)
* ISO 27001: A.8.5 (Secure authentication), A.7.4 (Physical security monitoring), A.8.21 (Security of network services), A.8.1 (User endpoint devices), A.7.14 (Secure disposal or re-use of equipment)
* CWE: CWE-287 (Improper Authentication)
* OWASP: A9 (Using Components with Known Vulnerabilities)

Policy Requirements:

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

1. **Technical Requirements:**
* Protect the connection using HTTPS (TLS 1.2 or later) for all authentication requests.
* Implement a stronger authentication mechanism, such as OAuth or JWT, for all web application authentication.
2. **Procedural Requirements:**
* Conduct regular security audits and vulnerability scans to identify and remediate similar vulnerabilities.
* Develop and enforce a secure coding standard for all web application development.
* Provide regular security awareness training to all employees and contractors.

Success Criteria and Validation Methods:

* The web application uses HTTPS for all authentication requests.
* A stronger authentication mechanism is implemented for all web application authentication.
* Regular security audits and vulnerability scans are conducted to identify and remediate similar vulnerabilities.
* Secure coding standards are enforced for all web application development.

Remediation Plan:

* **Technical Actions:**
+ Update the web application to use HTTPS for all authentication requests.
+ Implement a stronger authentication mechanism, such as OAuth or JWT.
* **Responsible Party:** Dev Lead

* **Timeline:** 2 weeks
* **Verification Steps:**
+ Conduct a code review to ensure HTTPS is implemented correctly.
+ Perform a vulnerability scan to verify the absence of similar vulnerabilities.

Monitoring and Detection:

* Regular security audits and vulnerability scans will be conducted to detect similar vulnerabilities in the future.
* Logging and alerting requirements will be implemented to detect and respond to potential security incidents.
* Continuous monitoring strategies will be implemented to ensure the security controls are effective and up-to-date.

Policy #15: DAST

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

SP-2023-001: Secure Cookie Flag Policy

Policy Title: Secure Cookie Flag Implementation

Risk Statement:

The absence of the secure flag on sensitive cookies poses a medium-severity risk to our organization. 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):
+ PR.DS-5: Implement secure protocols for sensitive data transmission
+ ID.AM-1: Identify and authenticate users
+ RA.AE-1: Analyze and respond to security incidents
* 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 error message
+ OWASP: Secure Cookies

Policy Requirements:

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

1. **Technical Requirement:** Set the secure flag for all cookies containing sensitive information or session tokens.
2. **Procedural Requirement:** Ensure that all developers and engineers follow secure coding practices, including the use of the secure flag for sensitive cookies.
3. **Success Criteria:** Verify that all cookies containing sensitive information or session tokens have the secure flag set.
4. **Validation Methods:** Conduct regular code reviews, vulnerability scans, and penetration testing to ensure compliance.

Remediation Plan:

* **Responsible Party:** Development Lead
* **Timeline:** 2 weeks
* **Technical Actions:**
1. Update the web application's cookie settings to include the secure flag for sensitive cookies.

2. Conduct a code review to ensure compliance with secure coding practices.
3. Perform a vulnerability scan to verify the absence of the vulnerability.
* **Verification Steps:** Code review, re-scan, and penetration testing.

Monitoring and Detection:

To detect similar vulnerabilities in the future, we will:

1. **Logging and Alerting:** Implement logging and alerting mechanisms to detect unauthorized access to sensitive cookies.
2. **Continuous Monitoring:** Conduct regular vulnerability scans and penetration testing to identify potential security risks.
3. **Code Review:** Perform regular code reviews to ensure compliance with secure coding practices.

By implementing this policy, we ensure the secure transmission of sensitive information and protect our users' confidentiality.