AI-Powered Security Policy Report

Generated: 2025-11-06 15:46:07

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 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-1 (Anomaly Detection)
- 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 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 actions must be taken:
- The Security Lead is responsible for ensuring the vulnerability is remediated.
- The developer must review and fix the vulnerable code in the UserController.js file (line 45) within 1 week.
- Verification steps include a code review, re-scan, and penetration test to ensure the vulnerability is fully remediated.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following strategies must be implemented:
- Regular security testing and code reviews must be performed.
- Logging and alerting must be configured to detect potential SQL injection attacks.
- Continuous monitoring of the application and its components must be performed to identify potential vulnerabilities.
- The security team must review logs and alerts regularly to detect and respond to potential security incidents.

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 compromise the security of our web application and potentially lead to unauthorized access to sensitive user data. If exploited, this vulnerability could result in financial loss, reputational damage, and legal liabilities. The affected systems include our web application, and the impacted users are our customers and employees who use the application.

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:
- Input validation and sanitization for all user-input data
- Output encoding to prevent malicious code injection
- Regular security testing and vulnerability assessments
- Implementation of a Web Application Firewall (WAF) to detect and prevent XSS attacks
Success criteria: No XSS vulnerabilities detected during security testing and vulnerability assessments.
Validation methods: Regular security testing, code reviews, and penetration testing.

REMEDICATION PLAN
To remediate the identified vulnerability, the following technical actions are required:
- Review and fix the template variable used in the anchor tag with the 'href' attribute in the src/views/profile.mustache file (line 23)
- Implement input validation and sanitization for all user-input data
Responsible party: Dev Lead
Timeline: 2 weeks
Verification steps: Code review, re-scan, and penetration testing to ensure the vulnerability is fully remediated.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures will be implemented:
- Regular security testing and vulnerability assessments
- Logging and alerting requirements: All security-related logs will be monitored and alerts will be triggered in case of suspicious activity
- Continuous monitoring strategies: Regular code reviews, security testing, and penetration testing will be performed to identify and remediate vulnerabilities.

Policy #3: SAST

Title: Path Join Resolve Traversal
Severity: HIGH
LLM: LLaMA 3.3 70B

POLICY IDENTIFIER
SP-2024-001: Path Traversal Protection Policy

RISK STATEMENT
The Path Traversal vulnerability poses a significant risk to our organization's data security. If exploited, an attacker could access sensitive files outside the intended directory, potentially leading to data breaches, intellectual property theft, or disruption of critical business operations. This vulnerability affects our web application, specifically the file handling functionality, and could impact all users who interact with the system.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), DE.CM-8 (Vulnerability Management)
- ISO 27001 Annex A controls: A.8.28 (Secure coding), A.8.29 (Security testing in development and acceptance)
- Relevant industry standards: OWASP 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:
- Validate and sanitize all user input used to construct file paths
- Implement a whitelist approach to restrict file access to authorized directories only
- Conduct regular security testing and code reviews to identify and address similar vulnerabilities
Success criteria include:
- All user input is validated and sanitized
- File access is restricted to authorized directories
- Security testing and code reviews are conducted quarterly

REMEDIATION PLAN
To remediate the Path Traversal vulnerability, the following technical actions are required:
- Review and fix the vulnerable code in src/routes/files.js (line 67)
- Implement input validation and sanitization for all file paths
- Conduct a thorough code review to identify and address similar vulnerabilities
The Security Lead is responsible for overseeing the remediation efforts, which must be completed within 1 week. Verification steps include:
- Code review by a senior developer
- Re-scanning the application using a vulnerability scanner
- Conducting a penetration test to validate the fixes

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 file access attempts
- Continuous monitoring of the application using a Web Application Firewall (WAF) and a vulnerability scanner
The Security Lead is responsible for ensuring that these measures are in place and that the policy is

reviewed and updated annually.

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 of our web application, 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.8.29 (Security testing in development and acceptance)
- Industry standards: OWASP A2:2017 (Broken Authentication), CWE-1004 (Sensitive Cookie Without 'HttpOnly' Flag)

POLICY REQUIREMENTS
To mitigate this risk, the following security controls must be implemented:
- All cookie sessions must be configured with the 'httpOnly' flag to prevent JavaScript access.
- Regular security testing and code reviews must be 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, penetration testing, and vulnerability scanning.

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 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 for suspicious activity, and alerts must be triggered in case of potential security incidents.
- Continuous monitoring strategies: Regularly review and update security policies, conduct vulnerability scanning, and perform penetration testing to ensure the security of our web application.

Policy #5: SAST

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

POLICY IDENTIFIER
Policy ID: SP-2024-001
Policy Title: Secure Storage of Sensitive Information

RISK STATEMENT
The presence of hardcoded secrets in our codebase poses a significant risk to our organization's security. If exploited, this vulnerability could lead to unauthorized access to sensitive information, compromising the confidentiality, integrity, and availability of our systems and data. This could result in financial losses, reputational damage, and legal liabilities. All systems, data, and users that rely on the affected code are at risk.

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.8.28 (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 sensitive information, including secrets and credentials, 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 include:
- No hardcoded secrets are detected in code reviews or vulnerability scans.
- All sensitive information is stored securely, using approved secret management systems.

REMEDIATION PLAN
To remediate the identified vulnerability:
- Remove the hardcoded secret from the affected code (src/config/auth.js, line 8).
- Implement a secure secret management system to store sensitive information.
- Perform a thorough code review to detect and remove any additional hardcoded secrets.
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:
- Regular code reviews must be performed using automated tools and manual inspection.
- Vulnerability scanning and penetration testing must be conducted regularly.
- Logging and alerting mechanisms must be implemented to detect and respond to potential security incidents.
Continuous monitoring strategies include:
- Regularly updating and patching dependencies and libraries.

- Implementing a secure development lifecycle (SDLC) that includes secure coding principles and guidelines.

Policy #6: SCA

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

POLICY IDENTIFIER
Policy ID: SP-2024-001
Policy Title: Prevention of Prototype Pollution in Lodash

RISK STATEMENT
The exploitation of the Prototype Pollution vulnerability in lodash could have significant business impacts, including the potential for unauthorized modification of sensitive data, disruption of critical systems, and compromise of intellectual property. If exploited, this vulnerability could lead to financial losses, reputational damage, and legal liabilities. The affected systems include all applications utilizing the lodash library, with potential consequences affecting both internal users and external customers.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), DE.CM-8 (Vulnerability Management)
- ISO 27001: 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)
- Industry Standards: OWASP A3:2021-Injection, CWE-1321: Prototype Pollution

POLICY REQUIREMENTS
To mitigate the Prototype Pollution vulnerability in lodash, the following security controls must be implemented:
- Regularly update and patch the lodash library to the latest version.
- Implement input validation and sanitization for all user-provided data.
- Utilize a Web Application Firewall (WAF) to detect and prevent malicious requests.
- Conduct regular security audits and penetration testing to identify potential vulnerabilities.
Success criteria include the successful implementation of these controls, as validated through code reviews, vulnerability scans, and penetration testing.

REMEDIATION PLAN
To remediate this vulnerability, the following technical actions are required:
- Update the lodash library to the latest version.
- Review and refactor code to prevent prototype pollution.
The responsible party for this remediation is the Security Lead, given the High severity of the vulnerability. The timeline for completion is 1 week. Verification steps include a code review, re-scan for vulnerabilities, and penetration testing to ensure the vulnerability has been successfully mitigated.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures must be implemented:
- Regular logging and monitoring of application security events.
- Configuration of alerts for suspicious activity indicative of prototype pollution attempts.
- Continuous monitoring of the application through regular security audits and penetration testing.
This proactive approach will enable the timely identification and mitigation of potential vulnerabilities, ensuring the security and integrity of our systems and data.

Policy #7: SCA

Title: express - CWE-601
Severity: MEDIUM
LLM: LLaMA 3.3 70B

POLICY IDENTIFIER
SP-2024-001: Express Open Redirect Policy

RISK STATEMENT
The Express Open Redirect vulnerability poses a significant risk to our organization, as it could allow attackers to redirect users to malicious websites, potentially leading to phishing attacks, malware infections, or unauthorized access to sensitive data. 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 scans, code reviews, and penetration testing

REMEDIATION PLAN
To remediate the Express Open Redirect vulnerability, the following technical actions are required:
- Update Express to the latest version
- Implement URL validation and filtering
- Configure the WAF to detect and prevent malicious redirects
- Responsible party: Dev Lead
- Timeline: 2 weeks
- Verification steps: Code review, security 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
- Logging and alerting requirements: All security-related events must be logged and alerted to the security team
- Continuous monitoring strategies: Regularly review security logs and conduct security testing to identify potential vulnerabilities
- Implement a web application security scanner to detect vulnerabilities and alert the security team.

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, gain unauthorized access to internal systems, and potentially exfiltrate sensitive data. The affected systems include all web 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, intellectual property theft, and reputational damage.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), DE.CM-8 (Vulnerability Management)
- ISO 27001 Annex A controls: A.8.23 (Web filtering), A.8.1 (User endpoint devices), A.5.10 (Acceptable use of information and other associated assets), A.5.33 (Protection of records), A.8.16 (Monitoring activities)
- Relevant industry standards: OWASP A10:2021 (Server-Side Request Forgery), CWE-918 (Server-Side Request Forgery)

POLICY REQUIREMENTS
To mitigate the SSRF vulnerability, the following security controls must be implemented:
- Validate and sanitize all user-input data
- Implement web application firewalls (WAFs) to detect and prevent SSRF attacks
- Configure axios to only allow requests to trusted domains
- Regularly review and update dependencies to ensure the latest security patches are applied
Success criteria include:
- No SSRF vulnerabilities detected in regular security scans
- All user-input data is validated and sanitized
- WAFs are configured and actively monitoring traffic

REMEDIATION PLAN
To remediate the SSRF vulnerability, the following technical actions are required:
- Update axios to the latest version
- Configure WAFs to detect and prevent SSRF attacks
- Implement input validation and sanitization for all user-input data
The responsible party for this remediation is the Security Lead. The timeline for completion is 1 week.
Verification steps include:
- Code review to ensure input validation and sanitization are implemented correctly
- Re-scanning the application with security tools to ensure no SSRF vulnerabilities are detected

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following monitoring and detection strategies will be implemented:
- Regular security scans using industry-recognized tools
- Logging and alerting for suspicious traffic patterns

- Continuous monitoring of application dependencies for known vulnerabilities
- Annual penetration testing to identify and exploit potential vulnerabilities.

Policy #9: SCA

Title: jsonwebtoken - CWE-327
Severity: MEDIUM
LLM: LLaMA 3.3 70B

POLICY IDENTIFIER
SP-2024-001: Secure JSON Web Token Configuration

RISK STATEMENT
The use of insecure default key sizes in JSON Web Tokens (JWT) poses a significant risk to the confidentiality and integrity of our systems and data. If exploited, an attacker could potentially decrypt sensitive information, leading to unauthorized access and data breaches. This vulnerability affects all systems 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.8.26 (Application security requirements), A.8.28 (Secure coding), A.8.27 (Secure system architecture and engineering principles), A.8.24 (Use of cryptography)
- Industry standards: OWASP A03:2021 (Injection), CWE-327 (Use of a Broken or Risky Cryptographic Algorithm)

POLICY REQUIREMENTS
To mitigate this risk, the following security controls must be implemented:
- Use a secure key size for JWT (minimum 2048 bits for RSA, 128 bits for AES)
- Implement secure key management practices, including key rotation and revocation
- Validate and verify JWT signatures on each request
- Use a secure random number generator for token generation
Success criteria: Successful implementation will be measured by regular security audits and penetration testing. Validation methods include code reviews, vulnerability scans, and monitoring of system logs for suspicious activity.

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

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures will be implemented:
- Regular security audits and vulnerability scans
- Monitoring of system logs for suspicious activity related to JWT
- Implementation of a Web Application Firewall (WAF) to detect and prevent common web attacks
Logging and alerting requirements: All security-related logs will be monitored and alerts will be sent to the security team in case of suspicious activity. Continuous monitoring strategies include regular code reviews, security testing, and participation in bug bounty programs to identify and address vulnerabilities proactively.

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 sensitive data. If exploited, this vulnerability could lead to unauthorized access, modification, or theft of sensitive information, resulting in reputational damage, financial loss, and potential legal liabilities. The affected systems include our web applications, development environments, and production servers, which could impact all users, including employees, customers, and partners.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
* NIST CSF: PR.DS-5 (Data Protection), DE.CM-8 (Vulnerability Management)
* ISO 27001: A.5.32 (Intellectual Property Rights), A.8.23 (Web Filtering), A.8.30 (Outsourced Development), A.8.31 (Separation of Development, Test, and Production Environments), A.8.4 (Access to Source Code)
* Industry standards: OWASP A9 (Using Components with Known Vulnerabilities), CWE-1321 (Prototype Pollution)

POLICY REQUIREMENTS
To mitigate the minimist prototype pollution vulnerability, the following security controls must be implemented:
* Regularly update and patch dependencies, including minimist, to ensure the latest security fixes are applied.
* Implement web filtering to restrict access to malicious content and reduce the risk of vulnerability exploitation.
* Enforce separation of development, testing, and production environments to prevent unauthorized access and modifications.
* Limit access to source code, development tools, and software libraries to authorized personnel only.
* Conduct regular code reviews and vulnerability scans to detect and remediate similar vulnerabilities.

REMEDIATION PLAN
To address the critical severity of this vulnerability, the following remediation plan is required:
* Technical actions: Update minimist to the latest version, implement web filtering, and enforce environment separation.
* Responsible party: CTO
* Timeline: 24-48 hours
* Verification steps: Conduct a code review, re-scan for vulnerabilities, and perform a penetration test to validate the remediation.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following monitoring and detection strategies will be implemented:
* Regularly log and monitor web application activity for suspicious behavior.
* Configure alerting systems to notify security teams of potential vulnerabilities.

* Conduct continuous monitoring and vulnerability scanning to identify and remediate vulnerabilities in a timely manner.
* Perform regular code reviews and security audits to ensure compliance with this policy and industry standards.

Policy #11: DAST

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

SP-2024-001: SQL Injection Prevention Policy

Policy Identifier: SP-2024-001
Policy Title: SQL Injection Prevention

Risk Statement:

The SQL injection vulnerability detected in the `http://testapp.local:3000/api/users/search` endpoint poses a medium-severity risk to the organization. If exploited, this vulnerability could allow an attacker to manipulate database queries, potentially leading to unauthorized access to sensitive user data, disruption of business operations, and reputational damage. The affected systems/data/users include the web application's user database and all users who interact with the application.

Compliance Mapping:

* NIST Cybersecurity Framework (CSF):
+ PR.DS-5: Implement secure coding practices
+ ID.BE-4: Implement a vulnerability management program
* 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 prevent SQL injection attacks, the following security controls must be implemented:

1. **Input Validation and Sanitization**: All user input must be validated and sanitized on the server-side using a whitelist approach. This includes type checking, length checking, and escaping of special characters.
2. **Parameterized Queries**: Prepared statements with parameterized queries must be used instead of string concatenation to build SQL queries.
3. **Code Review**: Regular code reviews must be conducted to ensure that all code changes adhere to secure coding practices.
4. **Vulnerability Scanning**: Regular vulnerability scans must be performed to detect and remediate potential SQL injection vulnerabilities.

Success criteria:

* All user input is validated and sanitized on the server-side.
* Prepared statements with parameterized queries are used for all database queries.

* Code reviews are conducted quarterly.
* Vulnerability scans are performed monthly.

Remediation Plan:

* **Responsible Party:** Dev Lead
* **Timeline:** 2 weeks
* **Technical Actions:**
1. Update the `http://testapp.local:3000/api/users/search` endpoint to use parameterized queries.
2. Implement input validation and sanitization on the server-side.
3. Conduct code reviews to ensure adherence to secure coding practices.
* **Verification Steps:**
1. Code review to ensure secure coding practices.
2. Re-scan the application for SQL injection vulnerabilities.
3. Conduct a penetration test to simulate a real-world attack.

Monitoring and Detection:

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

* **Logging and Alerting**: Log all database queries and alert the security team to any suspicious activity.
* **Continuous Monitoring**: Regularly scan the application for SQL injection vulnerabilities using a vulnerability scanner.
* **Code Analysis**: Conduct regular code analysis to detect potential security vulnerabilities.

By implementing these security controls and monitoring strategies, the organization can effectively 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

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

RISK STATEMENT:

The Cross-Site Scripting (XSS) vulnerability in the `/search` endpoint of our web application poses a medium-risk threat to our organization. If exploited, this vulnerability could allow an attacker to inject malicious code into a user's browser, potentially leading to unauthorized access, data theft, or other malicious activities. This vulnerability affects our web application, specifically the `/search` endpoint, and could impact users who interact with this feature.

COMPLIANCE MAPPING:

* NIST CSF: PR.DS-5 (Protect Critical Data), ID.BE-5 (Identify and Authenticate Users)
* ISO 27001 Annex A:
+ A.8.23 (Web filtering)
+ A.8.28 (Secure coding)
+ A.8.29 (Security testing in development and acceptance)
+ A.7.1 (Physical security perimeters)
+ A.8.8 (Management of technical vulnerabilities)
* Industry standards: CWE-79 (Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')), OWASP (Cross-Site Scripting (XSS))

POLICY REQUIREMENTS:

1. **Input Validation and Sanitization**: All user input will be validated, filtered, and sanitized to prevent XSS attacks. This includes converting special characters to HTML entities.
2. **Secure Coding Practices**: Our development team will adhere to secure coding principles, ensuring that all code is reviewed and tested for vulnerabilities.
3. **Security Testing**: Regular security testing will be conducted during the development life cycle to identify and address potential vulnerabilities.
4. **Web Application Firewall (WAF) Configuration**: Our WAF will be configured to detect and prevent XSS attacks.

Success Criteria:

* All user input will be validated and sanitized.
* Secure coding practices will be followed by all developers.
* Regular security testing will be conducted during the development life cycle.
* WAF configuration will be updated to detect and prevent XSS attacks.

REMEDIATION PLAN:

* **Responsible Party:** Dev Lead
* **Timeline:** 2 weeks
* **Technical Actions:**
1. Review and update code to validate and sanitize user input.

2. Configure WAF to detect and prevent XSS attacks.
3. Conduct security testing during the development life cycle.
* **Verification Steps:**
1. Code review to ensure secure coding practices.
2. Re-scan the web application for vulnerabilities.
3. Conduct penetration testing to verify the effectiveness of the WAF configuration.

MONITORING AND DETECTION:

* **Logging and Alerting:** Our web application will be configured to log all security-related events, including attempts to exploit the XSS vulnerability.
* **Continuous Monitoring:** Regular security testing and vulnerability scanning will be conducted to detect similar vulnerabilities in the future.
* **Incident Response:** An incident response plan will be in place to respond to any security incidents related to the XSS vulnerability.

Policy #13: DAST

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

Security Policy Section: SP-2023-001 - Cross-Site Request Forgery (CSRF) Protection

Policy Identifier: SP-2023-001

Policy Title: Protection against Cross-Site Request Forgery (CSRF) Attacks

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, a CSRF attack could allow an attacker to manipulate user actions without their knowledge or consent, potentially leading to unauthorized changes to user profiles or other sensitive data. This vulnerability could compromise the confidentiality, integrity, and availability of our systems and data, resulting in reputational damage, financial losses, and compromised user trust.

Compliance Mapping:

* NIST CSF: PR.DS-5 (Protect against malicious code)
* ISO 27001: A.8.23 (Web filtering), A.8.8 (Management of technical vulnerabilities), A.5.14 (Information transfer), A.6.4 (Disciplinary process)
* Industry standards: OWASP (Cross-Site Request Forgery (CSRF)), CWE (CWE-352: Cross-Site Request Forgery (CSRF))

Policy Requirements:

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

1. **Technical Requirements:**
* Utilize a vetted library or framework that provides Anti-CSRF token protection, such as OWASP CSRFGuard.
* Ensure the application is free of cross-site scripting (XSS) issues, as most CSRF defenses can be bypassed using attacker-controlled script.
2. **Procedural Requirements:**
* Conduct regular code reviews to identify and address potential CSRF vulnerabilities.
* Implement logging and alerting mechanisms to detect and respond to potential CSRF attacks.
3. **Success Criteria:**
* The application's HTML submission forms will include Anti-CSRF tokens.
* Regular code reviews will identify and address potential CSRF vulnerabilities.
* Logging and alerting mechanisms will detect and respond to potential CSRF attacks.

Remediation Plan:

To address this vulnerability, the following technical actions are required:

* Responsible Party: Dev Lead
* Timeline: 2 weeks

* Verification Steps:
+ Code review to ensure Anti-CSRF tokens are implemented correctly.
+ Re-scan the application for potential CSRF vulnerabilities.
+ Conduct a penetration test to verify the effectiveness of the CSRF protection measures.

Monitoring and Detection:

To detect similar vulnerabilities in the future, we will:

* Regularly scan the application for potential CSRF vulnerabilities using industry-standard tools.
* Conduct regular code reviews to identify and address potential CSRF vulnerabilities.
* Implement logging and alerting mechanisms to detect and respond to potential CSRF attacks.

By implementing these security controls and procedures, we can effectively mitigate the risk of CSRF attacks and protect the confidentiality, integrity, and availability of our systems and data.

Policy #14: DAST

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

SP-2023-001: Secure Authentication over Unsecured Connections

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

Risk Statement:

The use of HTTP basic or digest authentication over an unsecured connection poses a medium risk to the organization. If exploited, an attacker could intercept and reuse user credentials, potentially leading to unauthorized access to sensitive data and systems. This vulnerability affects all users who authenticate to the affected system, including employees, customers, and partners.

Compliance Mapping:

* NIST CSF: PR.DS-5 (Protect Information and Data)
* ISO 27001: A.8.5 (Secure authentication), A.7.4 (Physical security monitoring), A.8.21 (Security of network services), A.8.1 (User endpoint devices), A.7.14 (Secure disposal or re-use of equipment)
* Industry Standards: CWE-287 (Improper Authentication), OWASP Top 10 (2017) - A6: Sensitive Data Exposure

Policy Requirements:

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

1. **Technical Requirements:** Protect all authentication connections using HTTPS (TLS 1.2 or later).
2. **Procedural Requirements:** Implement a stronger authentication mechanism, such as OAuth or OpenID Connect, for all authentication requests.
3. **Success Criteria:** Verify that all authentication connections are encrypted using HTTPS and that the stronger authentication mechanism is properly configured.
4. **Validation Methods:** Conduct regular code reviews, penetration testing, and vulnerability scanning to ensure compliance.

Remediation Plan:

* **Responsible Party:** Dev Lead
* **Timeline:** 2 weeks
* **Technical Actions:**
1. Update the authentication endpoint to use HTTPS.
2. Implement a stronger authentication mechanism.
3. Configure the authentication mechanism to use secure protocols (e.g., TLS 1.2 or later).
* **Verification Steps:**
1. Conduct a code review to ensure HTTPS is properly implemented.
2. Perform a penetration test to verify the stronger authentication mechanism.
3. Run a vulnerability scan to detect any remaining issues.

Monitoring and Detection:

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

* **Logging and Alerting:** Configure logging and alerting mechanisms to detect any attempts to intercept or reuse user credentials.
* **Continuous Monitoring:** Regularly conduct vulnerability scanning, penetration testing, and code reviews to ensure compliance.
* **Incident Response:** Establish an incident response plan to quickly respond to any detected vulnerabilities or security incidents.

By implementing these measures, the organization will ensure the secure authentication of users over unsecured connections, protecting sensitive data and systems from unauthorized access.

Policy #15: DAST

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

SP-2023-001: Secure Cookie Handling Policy

RISK STATEMENT

The absence of the secure flag on sensitive cookies poses a medium-severity risk to our organization. If exploited, this vulnerability could lead to unauthorized access to sensitive information, compromising user confidentiality and potentially resulting in financial losses or reputational damage. The affected systems, data, and users include all users accessing our web application, particularly those using unencrypted connections.

COMPLIANCE MAPPING

This policy aligns with the following standards and controls:

* NIST Cybersecurity Framework (CSF):
+ PR.DS-5: Implement secure protocols for data transmission
+ ID.AM-1: Identify and authenticate users
+ DE.CM-1: Implement secure configuration for web servers
* 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 mitigate this risk, the following security controls must be implemented:

1. **Secure Cookie Flag**: Ensure that all sensitive cookies are set with the secure flag, indicating that they should only be transmitted over encrypted connections.
2. **Cookie Validation**: Implement cookie validation to ensure that only authorized cookies are accepted by the web application.
3. **Secure Protocol**: Ensure that all web traffic is transmitted over a secure protocol (HTTPS).
4. **Secure Configuration**: Implement secure configuration for web servers, including the use of secure protocols and encryption.

Success criteria:

* All sensitive cookies are set with the secure flag.
* Cookie validation is implemented and functioning correctly.
* All web traffic is transmitted over a secure protocol (HTTPS).
* Web servers are configured securely.

REMEDIATION PLAN

To remediate this vulnerability, the following technical actions are required:

* Responsible party: Dev Lead (Medium severity)
* Timeline: 2 weeks
* Verification steps:
+ Code review to ensure the secure flag is set for all sensitive cookies.
+ Re-scan of the web application using a vulnerability scanner to ensure the issue is resolved.
+ Penetration testing to verify the secure configuration of web servers.

MONITORING AND DETECTION

To detect similar vulnerabilities in the future:

* Implement logging and alerting mechanisms to monitor web traffic and detect any suspicious activity.
* Conduct regular security audits and vulnerability scans to identify potential issues.
* Continuously monitor web server configuration and ensure that secure protocols are used.

By implementing this policy, we can ensure that our web application handles cookies securely, protecting sensitive information and reducing the risk of unauthorized access.