AI-Powered Security Policy Report

Generated: 2025-11-07 17:17:36

Total Vulnerabilities Scanned: 26
• SAST: 10
• SCA: 8
• DAST: 8

LLM Models Used:

• LLaMA 3.3 70B (Groq) - SAST/SCA
• LLaMA 3.1 8B Instant (Groq) - DAST

Policy #1: SAST

Title: Explicit Unescape
Severity: HIGH
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 explicit unescaping of user input in web page generation. This vulnerability can lead to unauthorized access to sensitive data, theft of user sessions, and defacement of web pages. If exploited, this vulnerability can compromise the confidentiality, integrity, and availability of our web applications and user data.

COMPLIANCE MAPPING
This policy aligns with the following standards and guidelines:
- 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.8.28 (Secure coding)
- 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 properly sanitized and validated before being rendered in web pages.
- Secure coding principles must be applied to software development, including the use of templating engines that automatically escape user input.
- Regular security testing and code reviews must be performed to identify and remediate XSS

vulnerabilities.
Success criteria include:
- No XSS vulnerabilities detected in security scans and code reviews.
- All user input is properly sanitized and validated.
Validation methods include:
- Regular security testing and code reviews.
- Automated vulnerability scanning.

REMEDIATION PLAN
To remediate the identified vulnerability, the following technical actions are required:
- Review and fix the explicit unescaping of user input in the Mustache template (app/views/profile.js, line 42).
- Implement secure coding principles and validate user input.
Responsible party: Security Lead.
Timeline: 1 week.
Verification steps:
- Code review to ensure proper input validation and sanitization.
- Re-scan the application using automated vulnerability scanning tools.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following strategies must be implemented:
- Regular security testing and code reviews.
- Automated vulnerability scanning.
- Logging and alerting requirements:
+ Log all security-related events, including input validation failures.
+ Alert the Security Lead in case of suspected XSS attacks.
Continuous monitoring strategies:
- Regularly review and update the secure coding guidelines.
- Perform periodic security awareness training for developers.

Policy #2: SAST

Title: Express Check Csurf Middleware Missing
Severity: CRITICAL
LLM: LLaMA 3.3 70B

POLICY IDENTIFIER
SP-2024-001: Cross-Site Request Forgery (CSRF) Protection Policy

RISK STATEMENT
The absence of Express Check Csurf middleware in our application poses a significant risk to our business, as it exposes us to Cross-Site Request Forgery (CSRF) attacks. If exploited, this vulnerability could lead to unauthorized actions being performed on behalf of our users, resulting in potential financial loss, reputational damage, and compromised user data. The affected systems include our web application, and the impacted users are our customers and employees who interact with 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.8.26 (Application security requirements), A.8.29 (Security testing in development and acceptance)
- Industry standards: OWASP A5:2021 (Security Misconfiguration), CWE-352 (Cross-Site Request Forgery)

POLICY REQUIREMENTS
To mitigate the risk of CSRF attacks, the following security controls must be implemented:
- Enable Express Check Csurf middleware in our web application
- Conduct regular security testing and code reviews to identify vulnerabilities
- Implement a web application firewall (WAF) to filter incoming traffic
- Success criteria: Verification of Csurf middleware implementation through code review and security testing
- Validation methods: Regular security audits and penetration testing

REMEDIATION PLAN
To remediate this critical vulnerability, the following actions are required:
- Technical action: Enable Express Check Csurf middleware in the server.js file (line 15)
- Responsible party: CTO
- Timeline: 24 hours
- Verification steps: Code review and re-scan using a vulnerability scanner

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, we will:
- Implement logging and alerting mechanisms to detect potential CSRF attacks
- Conduct regular security audits and penetration testing
- Continuously monitor our web application for signs of vulnerabilities and update our security controls accordingly
- Utilize industry-standard tools and frameworks, such as OWASP ZAP, to identify and remediate vulnerabilities.

Policy #3: SAST

Title: Node Postgres Sqli
Severity: CRITICAL
LLM: LLaMA 3.3 70B

POLICY IDENTIFIER
SP-2024-001: SQL Injection Protection Policy

RISK STATEMENT
The SQL injection vulnerability poses a significant risk to our organization's data security. If exploited, an attacker could gain unauthorized access to sensitive user data, potentially leading to identity theft, financial loss, or reputational damage. The affected systems include our user database, and the impacted users are all registered customers. The potential consequences of a successful attack could result in significant financial and reputational losses.

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.8.8 (Management of technical vulnerabilities), A.8.16 (Monitoring activities), A.8.29 (Security testing in development and acceptance)
- Industry standards: OWASP A03:2021 (Injection), CWE-89 (Improper Neutralization of Special Elements used in an SQL Command)

POLICY REQUIREMENTS
To mitigate SQL injection vulnerabilities, the following security controls must be implemented:
- Use parameterized queries or prepared statements for all database interactions
- Validate and sanitize all user input data
- Implement regular security testing and code reviews for all development projects
- Success criteria: All database interactions use secure query methods, and no SQL injection vulnerabilities are detected during security testing
- Validation methods: Code reviews, penetration testing, and vulnerability scanning

REMEDIATION PLAN
To remediate the detected SQL injection vulnerability:
- Specific technical actions: Review and refactor the affected code in app/data/user-dao.js (line 87) to use parameterized queries
- Responsible party: CTO (due to critical severity)
- Timeline: Completion within 24-48 hours
- Verification steps: Code review, re-scan using vulnerability scanning tools, and penetration testing

MONITORING AND DETECTION
To detect similar vulnerabilities in the future:
- Implement regular security testing and code reviews for all development projects
- Configure logging and alerting to detect anomalous database activity
- Use continuous monitoring strategies, including vulnerability scanning and penetration testing, to identify potential SQL injection vulnerabilities
- Review and update this policy annually, or as needed, to ensure compliance with evolving industry standards and best practices.

Policy #4: SAST

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

POLICY IDENTIFIER
SP-2024-001: Hardcoded Secret Policy

RISK STATEMENT
The presence of hardcoded secrets in our codebase poses a significant risk to our organization's security. If exploited, an attacker could gain unauthorized access to sensitive information, compromising the confidentiality, integrity, and availability of our systems and data. This vulnerability affects our development, testing, and production environments, as well as our users' sensitive information. The potential consequences of exploitation include data breaches, financial loss, and reputational damage.

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.8.12 (Data leakage prevention), A.8.28 (Secure coding), A.8.1 (User endpoint devices), A.8.31 (Separation of development, test and production environments), A.7.14 (Secure disposal or re-use of equipment)
- 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 secret management systems.
- Code reviews must be conducted regularly to detect and remove hardcoded secrets.
- Secure coding principles must be applied to software development.
- Success criteria: All hardcoded secrets must be removed from the codebase, and code reviews must be conducted quarterly to ensure compliance.
- Validation methods: Code reviews, vulnerability scans, and penetration testing.

REMEDIATION PLAN
To remediate this vulnerability, the following technical actions are required:
- Review and fix the hardcoded secret in config/env.js (line 23).
- Implement a secret management system to store sensitive information.
- Conduct a code review to detect and remove any additional hardcoded secrets.
- Responsible party: Security Lead
- Timeline: 1 week
- Verification steps: Code review, vulnerability scan, and penetration test.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures must be implemented:
- Regular code reviews and vulnerability scans.
- Logging and alerting requirements: All code changes must be logged, and alerts must be triggered for any suspicious activity.
- Continuous monitoring strategies: Regularly review and update our secure coding principles and secret management systems to ensure they are aligned with industry best practices.

Policy #5: SAST

Title: Express Path Join
Severity: HIGH
LLM: LLaMA 3.3 70B

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

RISK STATEMENT
The path traversal vulnerability poses a significant risk to our organization's data security. If exploited, an attacker could potentially access sensitive files and data outside the intended directory, leading to unauthorized data disclosure, modification, or deletion. This vulnerability affects our web application, specifically the contributions module, and could impact all users who interact with the application. The potential consequences of exploitation include reputational damage, financial loss, and legal liabilities.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), DE.CM-8 (Vulnerability Management)
- ISO 27001 Annex A controls: A.8.29 (Security testing in development and acceptance), A.8.8 (Management of technical vulnerabilities)
- Relevant industry standards: OWASP A5:2021 (Security Misconfiguration), CWE-22 (Improper Limitation of a Pathname to a Restricted Directory)

POLICY REQUIREMENTS
To mitigate the path traversal vulnerability, the following security controls must be implemented:
- Input validation and sanitization for all user-input data used in file path construction
- Implementation of a whitelist approach for allowed file paths and directories
- Regular security testing and code reviews to identify and address similar vulnerabilities
- Success criteria: Verification of input validation and sanitization through code review and penetration testing
- Validation methods: Automated testing, manual code review, and vulnerability scanning

REMEDIATION PLAN
To remediate the identified vulnerability, the following actions are required:
- Specific technical actions: Review and fix the vulnerable code in app/routes/contributions.js (line 156) to implement input validation and sanitization
- Responsible party: Security Lead
- Timeline: 1 week
- Verification steps: Code review, re-scan, and penetration testing to ensure the vulnerability is fully addressed

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures will be implemented:
- Regular security testing and code reviews
- Logging and alerting requirements: All security-related events, including input validation failures, will be logged and alerted to the Security Lead
- Continuous monitoring strategies: Automated vulnerability scanning, regular code reviews, and penetration testing will be conducted to identify and address potential vulnerabilities.

Policy #6: SCA

Title: lodash - N/A
Severity: HIGH
LLM: LLaMA 3.3 70B

POLICY IDENTIFIER
SP-2024-001: Lodash Security Policy

RISK STATEMENT
The exploitation of the lodash vulnerability poses a significant risk to our organization's data integrity and confidentiality. If exploited, this vulnerability could lead to unauthorized access, modification, or deletion of sensitive data, resulting in financial losses, reputational damage, and potential legal liabilities. The affected systems include our web applications, APIs, and data storage systems, which could impact all users and 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.5.10 (Acceptable use of information and other associated assets), A.8.6 (Capacity management), A.5.37 (Documented operating procedures), A.5.31 (Legal, statutory, regulatory and contractual requirements), A.7.3 (Securing offices, rooms and facilities)
- Industry standards: OWASP A9 (Using Components with Known Vulnerabilities), CWE-79 (Improper Neutralization of Input During Web Page Generation)

POLICY REQUIREMENTS
To mitigate the risk associated with the lodash vulnerability, the following security controls must be implemented:
- Regularly update and patch lodash dependencies to ensure the latest security fixes are applied.
- Implement a vulnerability management program to identify, classify, and remediate vulnerabilities in a timely manner.
- Develop and maintain documented operating procedures for handling and updating dependencies.
- Conduct regular code reviews to ensure secure coding practices are followed.
Success criteria include:
- All lodash dependencies are updated to the latest version within 1 week.
- A vulnerability management program is established and operational within 2 weeks.
Validation methods include code reviews, dependency audits, and vulnerability scans.

REMEDIATION PLAN
To remediate the lodash vulnerability, the following technical actions are required:
- Update all lodash dependencies to the latest version.
- Conduct a thorough code review to ensure secure coding practices are followed.
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, and penetration test to ensure the vulnerability is fully remediated.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures will be implemented:
- Regular vulnerability scans and dependency audits.
- Logging and alerting requirements will be established to detect potential security incidents.
- Continuous monitoring strategies will be implemented to identify and remediate vulnerabilities in a timely manner.

This will ensure the organization's systems and data remain secure and protected against known and unknown vulnerabilities.

Policy #7: SCA

Title: express - N/A
Severity: CRITICAL
LLM: LLaMA 3.3 70B

POLICY IDENTIFIER
SP-2024-001 - Express Security Policy

RISK STATEMENT
The lack of proper security controls in our Express application poses a significant risk to our organization's data and systems. If exploited, this vulnerability could lead to unauthorized access, data breaches, and disruption of critical services. The affected systems include our web application servers, databases, and all users who interact with our online services. The potential consequences include financial loss, reputational damage, and legal liabilities.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), DE.CM-8 (Vulnerability Management)
- ISO 27001: A.5.10 (Acceptable use of information and other associated assets), A.8.6 (Capacity management), A.5.37 (Documented operating procedures), A.5.31 (Legal, statutory, regulatory and contractual requirements), A.7.3 (Securing offices, rooms and facilities)
- Industry standards: OWASP Top 10, CWE-200 (Information Exposure)

POLICY REQUIREMENTS
To mitigate this risk, the following security controls must be implemented:
- Conduct regular security audits and vulnerability assessments on our Express application
- Implement proper input validation and sanitization to prevent common web attacks
- Ensure all sensitive data is encrypted in transit and at rest
- Develop and maintain up-to-date documentation of our application's architecture and security controls
- Establish a capacity management plan to monitor and adjust resource utilization
- Define and enforce acceptable use policies for our information and associated assets

Success criteria include:
- Completion of a thorough vulnerability assessment within the next 30 days
- Implementation of recommended security controls within 60 days
- Quarterly review and update of our security documentation and policies

REMEDIATION PLAN
To address this critical vulnerability, the following technical actions are required:
- Conduct an immediate code review to identify and remediate any security flaws
- Implement a web application firewall (WAF) to detect and prevent common web attacks
- Configure logging and alerting to detect potential security incidents

The CTO is responsible for overseeing the remediation efforts, given the critical severity of this vulnerability. The timeline for completion is within the next 24-48 hours. Verification steps include a code review, re-scan, and penetration test to ensure the vulnerability has been properly addressed.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, we will:
- Implement continuous monitoring of our application's security logs
- Conduct regular vulnerability assessments and penetration tests

- Establish a bug bounty program to encourage responsible disclosure of security flaws
- Require all developers to undergo security training and follow secure coding practices

Logging and alerting requirements include:
- Configuring our WAF to alert on potential security incidents
- Implementing a security information and event management (SIEM) system to monitor and analyze security logs
- Establishing a incident response plan to quickly respond to security incidents.

Policy #8: SCA

Title: jsonwebtoken - N/A
Severity: HIGH
LLM: LLaMA 3.3 70B

POLICY IDENTIFIER
SP-2024-001: JSON Web Token Security Policy

RISK STATEMENT
The exploitation of vulnerabilities in JSON Web Tokens (JWT) could have significant business impacts, including unauthorized access to sensitive data and systems. If exploited, this vulnerability could lead to data breaches, financial losses, and reputational damage. The affected systems include all applications and services that utilize JWT for authentication and authorization.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), DE.AE-3 (Anomaly Detection)
- ISO 27001 Annex A controls: A.5.10 (Acceptable use of information and other associated assets), A.8.6 (Capacity management), A.5.37 (Documented operating procedures), A.5.31 (Legal, statutory, regulatory and contractual requirements), A.7.3 (Securing offices, rooms and facilities)
- Relevant industry standards: OWASP JSON Web Token Cheat Sheet, CWE-287 (Improper Authentication)

POLICY REQUIREMENTS
To mitigate the risks associated with JWT vulnerabilities, the following security controls must be implemented:
- Validate all JWTs on each request to prevent token tampering and replay attacks.
- Implement secure key management practices for signing and verifying JWTs.
- Use secure algorithms and protocols for JWT encryption and decryption.
- Establish procedures for handling and storing sensitive data transmitted via JWTs.
Success criteria include:
- All JWTs are validated on each request.
- Secure key management practices are in place and reviewed quarterly.
Validation methods include regular security audits and penetration testing.

REMEDIATION PLAN
To address the identified vulnerability, the following technical actions are required:
- Update all JWT libraries and dependencies to the latest versions.
- Implement secure key management practices for signing and verifying JWTs.
The responsible party for this remediation is the Security Lead. The timeline for completion is 1 week. Verification steps include code review and re-scanning for vulnerabilities.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures must be taken:
- Implement logging and alerting for all JWT-related errors and anomalies.
- Conduct regular security audits and penetration testing to identify potential vulnerabilities.
- Establish a continuous monitoring strategy to detect and respond to security incidents in a timely manner.
Logging requirements include capturing all JWT validation errors and anomalies, with alerts sent to the Security Lead for review and response.

Policy #9: SCA

Title: mongoose - N/A
Severity: MEDIUM
LLM: LLaMA 3.3 70B

POLICY IDENTIFIER
SP-2024-001: Mongoose Security Policy

RISK STATEMENT
The lack of description for the mongoose vulnerability poses a moderate risk to our organization's data security. In non-technical terms, this vulnerability could allow unauthorized access or manipulation of sensitive data, potentially leading to data breaches, financial losses, or reputational damage. The affected systems include our database management systems, and the impacted users are our customers and internal stakeholders who rely on the security of our data.

COMPLIANCE MAPPING
This policy aligns with the following compliance requirements:
- NIST CSF: PR.DS-5 (Data Protection), DE.AE-3 (Anomaly Detection)
- ISO 27001 Annex A controls: A.5.10 (Acceptable use of information and other associated assets), A.8.6 (Capacity management), A.5.37 (Documented operating procedures), A.5.31 (Legal, statutory, regulatory and contractual requirements), A.7.3 (Securing offices, rooms and facilities)
- Relevant industry standards: OWASP A10:2021 (Security Logging and Monitoring Failures), CWE-200 (Information Exposure)

POLICY REQUIREMENTS
To mitigate this vulnerability, the following security controls must be implemented:
- Develop and maintain detailed documentation for mongoose, including acceptable use policies and procedures for handling information and associated assets.
- Conduct regular capacity management reviews to ensure resources are utilized efficiently and adjusted according to current and expected capacity requirements.
- Establish and document operating procedures for information processing facilities, making them available to relevant personnel.
- Identify, document, and keep up-to-date legal, statutory, regulatory, and contractual requirements relevant to information security.
- Implement physical security measures for offices, rooms, and facilities to protect against unauthorized access.
Success criteria include the completion of these documentation and implementation tasks, validated through internal audits and compliance reviews.

REMEDIATION PLAN
To remediate this vulnerability, the following technical actions are required:
- Develop a comprehensive documentation set for mongoose, including user manuals, technical guides, and security policies.
- Conduct a capacity management review to identify areas for optimization.
- Implement physical security measures for sensitive areas.
The responsible party for this remediation is the Development Lead, given the medium severity of the vulnerability. The timeline for completion is two weeks. Verification steps include a code review, documentation audit, and a physical security assessment.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, we will implement the following monitoring and detection

strategies:
- Regular security audits and compliance reviews to ensure adherence to established policies and procedures.
- Logging and alerting mechanisms to detect anomalies in system behavior, aligned with OWASP and NIST guidelines.
- Continuous monitoring of system performance and capacity to identify potential security risks early.
By following these strategies, we aim to proactively identify and mitigate security vulnerabilities, ensuring the integrity and security of our data and systems.

Policy #10: SCA

Title: axios - N/A
Severity: HIGH
LLM: LLaMA 3.3 70B

POLICY IDENTIFIER
SP-2024-001: Axios Security Policy

RISK STATEMENT
The lack of description for the axios vulnerability poses a significant risk to our organization's information security. In non-technical terms, this vulnerability could allow unauthorized access or manipulation of sensitive data, potentially leading to financial loss, reputational damage, or legal liabilities. If exploited, this vulnerability could affect our web applications, compromising user data and disrupting business operations.

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.5.10 (Acceptable use of information and other associated assets), A.8.6 (Capacity management), A.5.37 (Documented operating procedures), A.5.31 (Legal, statutory, regulatory and contractual requirements), A.7.3 (Securing offices, rooms and facilities)
- Relevant industry standards: OWASP A09:2021 (Security Logging and Monitoring Failures), CWE-79 (Improper Neutralization of Input During Web Page Generation)

POLICY REQUIREMENTS
To mitigate this vulnerability, the following security controls must be implemented:
- Conduct regular security audits and vulnerability assessments for all web applications using axios.
- Implement input validation and sanitization for all user-input data.
- Develop and maintain up-to-date documentation for axios usage and security best practices.
- Ensure that all developers and personnel handling axios are aware of and adhere to the acceptable use policy.
- Success criteria: Successful implementation of security controls will be measured by regular security audits and vulnerability assessments, with a target of zero critical vulnerabilities.

REMEDIATION PLAN
To remediate this vulnerability, the following technical actions are required:
- Conduct a thorough review of axios usage in all web applications.
- Implement security patches and updates for axios.
- Develop and deploy input validation and sanitization mechanisms.
Responsible party: Security Lead.
Timeline: 1 week.
Verification steps: Code review, re-scan, and penetration testing will be conducted to verify the effectiveness of the remediation efforts.

MONITORING AND DETECTION
To detect similar vulnerabilities in the future, the following measures will be implemented:
- Regular security logging and monitoring of web applications using axios.
- Alerting mechanisms will be set up to notify the Security Lead of potential security incidents.
- Continuous monitoring strategies, including regular vulnerability assessments and penetration testing, will be employed to identify and remediate vulnerabilities in a timely manner.
By following this policy, we aim to minimize the risk of axios-related vulnerabilities and ensure the

security and integrity of our web applications and user data.

Policy #11: DAST

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

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

Policy Title: Protection against Cross-Site Scripting (XSS) Attacks

Risk Statement:

The Cross-Site Scripting (XSS) vulnerability poses a medium-risk threat to our organization's web applications. If exploited, an attacker could inject malicious code into our users' browsers, potentially leading to unauthorized access, data theft, or system compromise. This vulnerability affects our web-based systems, including the search function at http://localhost:4000/search, and may impact user data and system integrity.

Compliance Mapping:

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

Policy Requirements:

To mitigate the XSS vulnerability, we require the following security controls:

1. **Secure Coding Practices:** Implement a vetted library or framework that prevents XSS attacks or provides constructs to avoid this weakness.
2. **Security Testing:** Define and implement security testing processes in the development life cycle to detect and prevent XSS vulnerabilities.
3. **Code Review:** Conduct regular code reviews to ensure secure coding practices are followed.
4. **Logging and Alerting:** Implement logging and alerting mechanisms to detect similar vulnerabilities in the future.

Success Criteria:

* All web applications are secured against XSS attacks.
* Secure coding practices are followed in all software development projects.
* Regular security testing and code reviews are conducted to detect and prevent XSS vulnerabilities.

Remediation Plan:

* **Responsible Party:** Development Lead

* **Timeline:** 2 weeks
* **Technical Actions:**
1. Implement a vetted library or framework that prevents XSS attacks.
2. Define and implement security testing processes in the development life cycle.
3. Conduct regular code reviews to ensure secure coding practices are followed.
* **Verification Steps:**
1. Code review to ensure secure coding practices are followed.
2. Re-scan the web application for XSS vulnerabilities.
3. Conduct penetration testing to verify the effectiveness of the remediation.

Monitoring and Detection:

To detect similar vulnerabilities in the future, we will:

* Implement logging and alerting mechanisms to detect XSS attacks.
* Conduct regular security testing and code reviews.
* Continuously monitor web application logs for suspicious activity.

By implementing these security controls and following this policy, we can mitigate the XSS vulnerability and protect our web applications from potential attacks.

Policy #12: DAST

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

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

Policy Identifier: SP-2023-001

Policy Title: Preventing Cross-Site Scripting (XSS) Vulnerabilities

Risk Statement:

The Cross-Site Scripting (XSS) vulnerability poses a medium-risk threat to the organization's online presence and user data. If exploited, this vulnerability could allow attackers to inject malicious scripts into user sessions, potentially leading to unauthorized access, data theft, or reputational damage. The affected systems and data include user profiles, session cookies, and sensitive information stored on user endpoint devices.

Compliance Mapping:

- **NIST Cybersecurity Framework (CSF):** PR.DS-5 (Protect Data)
- **ISO 27001 Annex A Controls:**
- A.8.23 (Web filtering)
- A.8.1 (User endpoint devices)
- A.8.15 (Logging)
- A.8.4 (Access to source code)
- A.5.37 (Documented operating procedures)
- **Industry Standards:** CWE-79 (Cross-Site Scripting), OWASP Top 10 (2017)

Policy Requirements:

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

1. **Input Validation and Sanitization:** All user input must be validated and sanitized to prevent malicious scripts from being injected into the application.
2. **Output Encoding:** All user-generated output must be properly encoded to prevent XSS attacks.
3. **Secure Coding Practices:** Developers must adhere to secure coding practices, including the use of prepared statements and parameterized queries.
4. **Regular Code Reviews:** Regular code reviews must be conducted to identify and address potential XSS vulnerabilities.
5. **Logging and Monitoring:** Logs must be produced, stored, and analyzed to detect and respond to potential XSS attacks.

Success Criteria:

- All user input is validated and sanitized.
- All user-generated output is properly encoded.
- Secure coding practices are followed by all developers.
- Regular code reviews are conducted to identify and address potential XSS vulnerabilities.

- Logs are produced, stored, and analyzed to detect and respond to potential XSS attacks.

Remediation Plan:

- **Responsible Party:** Development Lead
- **Timeline:** 2 weeks
- **Technical Actions:**
1. Implement input validation and sanitization using a reputable library (e.g., OWASP ESAPI).
2. Update the application to use prepared statements and parameterized queries.
3. Conduct regular code reviews to identify and address potential XSS vulnerabilities.
- **Verification Steps:**
1. Code review to ensure input validation and sanitization are implemented correctly.
2. Re-scan the application for XSS vulnerabilities using a reputable tool (e.g., OWASP ZAP).
3. Conduct penetration testing to simulate real-world attacks.

Monitoring and Detection:

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

- **Logging and Alerting:** Logs must be produced, stored, and analyzed to detect and respond to potential XSS attacks.
- **Continuous Monitoring:** The application must be continuously monitored for XSS vulnerabilities using reputable tools (e.g., OWASP ZAP).
- **Regular Code Reviews:** Regular code reviews must be conducted to identify and address potential XSS vulnerabilities.

Policy #13: DAST

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

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

Policy Identifier: SP-2023-001
Policy Title: CSRF Protection Policy

Risk Statement:

The absence of Anti-CSRF tokens in HTML submission forms poses a medium-severity risk to the organization. If exploited, this vulnerability could lead to unauthorized state changes, compromising the integrity and confidentiality of user data. Potential consequences include financial loss, reputational damage, and compromised user trust. Affected systems, data, and users include:

* Web applications with state-changing operations (e.g., user contributions, account management)
* User data stored in databases
* All users interacting with affected web applications

Compliance Mapping:

* 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
+ A.8.8: Management of technical vulnerabilities
+ A.5.14: Information transfer
+ A.6.4: Disciplinary process
* Industry standards:
+ CWE-352: Cross-Site Request Forgery
+ OWASP: Cross-Site Request Forgery (CSRF)

Policy Requirements:

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

1. **Technical Requirements:**
* Implement Anti-CSRF tokens for all state-changing operations in web applications.
* Use a secure token generation mechanism (e.g., cryptographically secure pseudo-random number generator).
* Validate tokens on server-side to prevent token tampering.
2. **Procedural Requirements:**
* Develop and maintain a secure coding standard for web application development.
* Conduct regular code reviews to ensure adherence to the secure coding standard.
* Provide training to developers on secure coding practices and CSRF prevention.

Success Criteria:

* All web applications with state-changing operations implement Anti-CSRF tokens.
* Secure coding standard is developed and maintained.
* Regular code reviews are conducted to ensure adherence to the secure coding standard.

Remediation Plan:

* **Responsible Party:** Dev Lead
* **Timeline:** 2 weeks
* **Technical Actions:**
+ Implement Anti-CSRF tokens in affected web applications.
+ Update secure coding standard to include CSRF prevention guidelines.
+ Conduct code reviews to ensure adherence to the updated secure coding standard.
* **Verification Steps:**
+ Code review to ensure Anti-CSRF tokens are implemented correctly.
+ Re-scan web applications for CSRF vulnerabilities after remediation.

Monitoring and Detection:

* Regularly scan web applications for CSRF vulnerabilities using automated tools (e.g., OWASP ZAP).
* Implement logging and alerting mechanisms to detect potential CSRF attacks.
* Conduct continuous monitoring of web application logs to identify potential security incidents.

By implementing this policy, the organization will mitigate the risk of CSRF attacks and ensure the confidentiality, integrity, and availability of user data.

Policy #14: DAST

Title: Cross-Domain Misconfiguration
Severity: MEDIUM
LLM: LLaMA 3.1 8B Instant

SP-2023-001: Cross-Origin Resource Sharing (CORS) Misconfiguration Policy

RISK STATEMENT

The Cross-Origin Resource Sharing (CORS) misconfiguration on our web server may allow unauthorized access to sensitive data, potentially leading to data breaches and reputational damage. If exploited, this vulnerability could compromise the confidentiality, integrity, and availability of our user data, exposing sensitive information to unauthorized parties. This vulnerability affects our web application, specifically the `/api/users` endpoint, and may impact our users' trust and loyalty.

COMPLIANCE MAPPING

This policy aligns with the following compliance requirements:

* ISO 27001: A.8.23 (Web filtering), A.8.1 (User endpoint devices), A.8.6 (Capacity management), A.8.3 (Information access restriction), and A.8.16 (Monitoring activities)
* NIST Cybersecurity Framework (CSF): PR.DS-5 (Protect Data)
* CWE: CWE-264 (Cross-Site Scripting (XSS))
* OWASP: A3:2013 (Broken Access Control)

POLICY REQUIREMENTS

To mitigate this vulnerability, we will implement the following security controls:

1. **Configure CORS headers properly**: Ensure that sensitive data is not available in an unauthenticated manner by implementing CORS headers that restrict access to authorized domains.
2. **Implement web application firewall (WAF) rules**: Configure WAF rules to detect and prevent unauthorized access to sensitive data.
3. **Regularly review and update CORS configurations**: Schedule regular reviews of CORS configurations to ensure they remain secure and up-to-date.

Success criteria:

* CORS headers are properly configured to restrict access to authorized domains.
* WAF rules are implemented and regularly reviewed to detect and prevent unauthorized access.
* Regular reviews of CORS configurations are conducted to ensure security and compliance.

REMEDIATION PLAN

To remediate this vulnerability, we will:

1. **Configure CORS headers**: Implement CORS headers to restrict access to authorized domains. (Responsible party: Dev Lead, Timeline: 2 weeks)
2. **Implement WAF rules**: Configure WAF rules to detect and prevent unauthorized access to sensitive data. (Responsible party: Dev Lead, Timeline: 2 weeks)
3. **Review and update CORS configurations**: Schedule regular reviews of CORS configurations to

ensure they remain secure and up-to-date. (Responsible party: Dev Lead, Timeline: Ongoing)

MONITORING AND DETECTION

To detect similar vulnerabilities in the future, we will:

1. **Regularly scan for vulnerabilities**: Schedule regular vulnerability scans to detect potential security issues.
2. **Implement logging and alerting**: Configure logging and alerting mechanisms to detect and respond to potential security incidents.
3. **Conduct continuous monitoring**: Conduct continuous monitoring of our web application to detect and respond to potential security incidents.

By implementing these security controls and monitoring strategies, we can mitigate the risk of this vulnerability and ensure the confidentiality, integrity, and availability of our user data.

Policy #15: DAST

Title: X-Content-Type-Options Header Missing
Severity: MEDIUM
LLM: LLaMA 3.1 8B Instant

POLICY IDENTIFIER:
SP-2023-001 - Anti-MIME-Sniffing Header Configuration Policy

RISK STATEMENT:
The Anti-MIME-Sniffing header X-Content-Type-Options not being set to 'nosniff' poses a medium-severity risk to our organization's web applications. If exploited, this vulnerability could allow attackers to execute malicious scripts or inject malicious content, potentially leading to data breaches, unauthorized access, or reputational damage. The affected systems and data include all web servers and applications hosted on our infrastructure, as well as sensitive user data stored on these systems.

COMPLIANCE MAPPING:
- NIST CSF: PR.DS-5 (Protect Data) and ID.RD-12 (Identify Resources)
- ISO 27001 Annex A: A.8.23 (Web filtering), A.8.3 (Information access restriction), A.8.10 (Information deletion), A.8.7 (Protection against malware), and A.8.33 (Test information)
- Industry standards: OWASP (OWASP Top 10 - 2021), CWE (CWE-693 - Improper Neutralization of Input During Web Page Generation)

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

1. **Technical Requirement:** Set the X-Content-Type-Options header to 'nosniff' for all responses on our web servers.
2. **Procedural Requirement:** Ensure that all web applications and servers are configured to include the X-Content-Type-Options header with the value 'nosniff' in their HTTP responses.
3. **Success Criteria:** Verify that the X-Content-Type-Options header is set to 'nosniff' for all web servers and applications using regular security audits and vulnerability scans.
4. **Validation Methods:** Conduct regular security audits and vulnerability scans to ensure compliance with this policy.

REMEDIATION PLAN:
- **Responsible Party:** Dev Lead
- **Timeline:** 2 weeks
- **Technical Actions:**
1. Update web server configurations to include the X-Content-Type-Options header with the value 'nosniff'.
2. Review and update web application code to ensure compliance with this policy.
- **Verification Steps:**
1. Conduct a code review to ensure that the X-Content-Type-Options header is set correctly.
2. Perform a re-scan of our web servers and applications using a vulnerability scanner to verify compliance.

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

1. **Logging and Alerting:** Configure our web servers and applications to log any attempts to modify the X-Content-Type-Options header.

2. **Continuous Monitoring:** Regularly perform security audits and vulnerability scans to identify any potential vulnerabilities.
3. **Penetration Testing:** Conduct regular penetration testing to simulate real-world attacks and identify potential vulnerabilities.