AI-Powered Security Policy Report

Generated: 2025-11-07 23:47:53

Total Vulnerabilities Scanned: 10
• SAST: 10
• SCA: 0
• DAST: 0

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

Security Policy: Explicit Unescape

Executive Summary
The Explicit Unescape vulnerability, detected in the `app/views/profile.js` file, poses a significant risk to our web application's security. This vulnerability, categorized as HIGH severity, can lead to Cross-Site Scripting (XSS) attacks, compromising user data and system integrity. To mitigate this risk, we must implement a comprehensive security policy that adheres to international standards, including NIST Cybersecurity Framework and ISO/IEC 27001:2022. This policy outlines the necessary steps to prevent, detect, and respond to Explicit Unescape vulnerabilities, ensuring the confidentiality, integrity, and availability of our systems and data.

Risk Analysis

CVSS v3.1 Score Breakdown:
- **Base Score**: 8.8
- **Attack Vector**: Network
- **Attack Complexity**: Low
- **Privileges Required**: None
- **User Interaction**: Required
- **Scope**: Unchanged
- **Confidentiality Impact**: High
- **Integrity Impact**: High
- **Availability Impact**: Low

Threat Intelligence:
- **Known Exploits**: Yes, public exploits are available
- **MITRE ATT&CK; Techniques**: T1055 (Social Engineering), T1061 (Web Session Cookie)
- **Attack Chain**: An attacker can exploit this vulnerability by injecting malicious code into user input, which is then executed by the application, potentially leading to unauthorized access, data theft, or system compromise

Compliance Framework Mapping

NIST Cybersecurity Framework:
- **Control ID**: PR.DS-5
- **Function**: Protect
- **Category**: Data Security
- **Implementation Requirements**: Implement secure coding practices, input validation, and output encoding to prevent XSS attacks
- **Evidence Required**: Code reviews, security testing reports, and compliance documentation

- **Control ID**: DE.CM-1
- **Function**: Detect
- **Category**: Anomaly Detection
- **Implementation Requirements**: Implement SIEM correlation rules, WAF signatures, and IDS/IPS patterns to detect potential XSS attacks
- **Evidence Required**: SIEM logs, WAF logs, and IDS/IPS alerts

ISO/IEC 27001:2022 Annex A:
- **Control**: A.8.23
- **Objective**: Manage access to external websites to reduce exposure to malicious content
- **Implementation Guidance**: Implement web filtering, monitor user activity, and enforce secure browsing practices
- **Audit Trail**: Web filtering logs, user activity logs, and security incident reports

- **Control**: A.8.8
- **Objective**: Manage technical vulnerabilities of information systems
- **Implementation Guidance**: Implement vulnerability management, patch management, and secure coding practices
- **Audit Trail**: Vulnerability scan reports, patch management logs, and code review records

Additional Frameworks:
- **OWASP ASVS**: Verify that the application implements secure coding practices, input validation, and output encoding to prevent XSS attacks
- **PCI DSS**: Implement secure coding practices, input validation, and output encoding to prevent XSS attacks, and ensure compliance with PCI DSS requirements

Defense-in-Depth Strategy

Layer 1: Prevention
1. **Code Level**: Implement secure coding practices, input validation, and output encoding to prevent XSS attacks
2. **Framework Level**: Enable security features in the web framework, such as CSRF protection and XSS filtering
3. **Infrastructure Level**: Implement WAF, network controls, and web filtering to prevent malicious

traffic

Layer 2: Detection
1. **SIEM Correlation Rules**:
```sql
rule ExplicitUnescape {
description = "Detect Explicit Unescape attacks"
condition = [
"http.request.uri.query_string" =~ /|<\/script>/<br/> ]<br/> action = [<br/> "alert": "Explicit Unescape attack detected"<br/> ]<br/>}<br/>```<br/><br/>2. **WAF Signatures**:<br/>```xml<br/><rule><br/> <name>ExplicitUnescape</name><br/> <pattern>&amp;lt;script&amp;gt;|&amp;lt;/script&amp;gt;</pattern><br/> <action>block</action><br/></rule><br/>```<br/><br/>3. **IDS/IPS Patterns**:<br/>```snort<br/>alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"Explicit Unescape attack"; content:"<script>|"; sid:1000001;)
```

4. **Log Monitoring**:
Monitor web server logs for suspicious activity, such as unusual user agent strings or query parameters

Layer 3: Response

If Exploitation Detected:
1. **Immediate Actions** (0-15 minutes):
- Contain the incident by blocking malicious traffic
- Preserve evidence for further analysis

2. **Short-term Response** (15 min - 4 hours):
- Investigate the incident to determine the root cause and scope
- Assess the impact on the organization and its assets

3. **Long-term Response** (4+ hours):
- Conduct a root cause analysis to identify vulnerabilities and weaknesses
- Develop and implement a remediation plan to prevent similar incidents

Remediation Strategies

Strategy A: Secure Framework Migration
- **Approach**: Migrate to a secure web framework that provides built-in security features, such as CSRF protection and XSS filtering
- **Security**: ■■■■■
- **Performance**: ■■■■
- **Complexity**: ■■■
- **Cost**: $10,000
- **Timeline**: 6 weeks

Strategy B: Manual Validation Layer
- **Approach**: Implement a custom validation layer to validate user input and prevent XSS attacks
- **Security**: ■■■■
- **Performance**: ■■■
- **Complexity**: ■■
- **Cost**: $5,000
- **Timeline**: 3 weeks

Strategy C: Web Application Firewall
- **Approach**: Implement a WAF to detect and prevent XSS attacks
- **Security**: ■■■
- **Performance**: ■■■■■
- **Complexity**: ■
- **Cost**: $2,000
- **Timeline**: Immediate
- **Note**: Temporary solution, implement A or B for permanent fix

Recommended Approach: Strategy A with justification
Strategy A provides the highest level of security and is a long-term solution. While it may require more resources and time, it is the most effective way to prevent XSS attacks and ensure the security of our web application.

Security Tool Integration

SAST Configuration:
```yml
rules:
- id: ExplicitUnescape
pattern: /|<\/script>/<br/> languages:<br/> - javascript<br/>```<br/><br/>#### DAST Testing:<br/>```python<br/>import requests<br/><br/>def test_explicit_unescape():<br/> url = "https://example.com/profile"<br/> payload = {"name": "<script>alert('XSS')"}
response = requests.post(url, data=payload)
assert "XSS" not in response.text
```

SCA Monitoring:
Monitor dependencies for known vulnerabilities and ensure that all dependencies are