Sri Lanka Institute of Information Technology

Bug Bounty Report 09

WS Assignment

IE2062 – Web Security

IT23159730

W.H.M.S.R.Bandara

Vulnerability Title

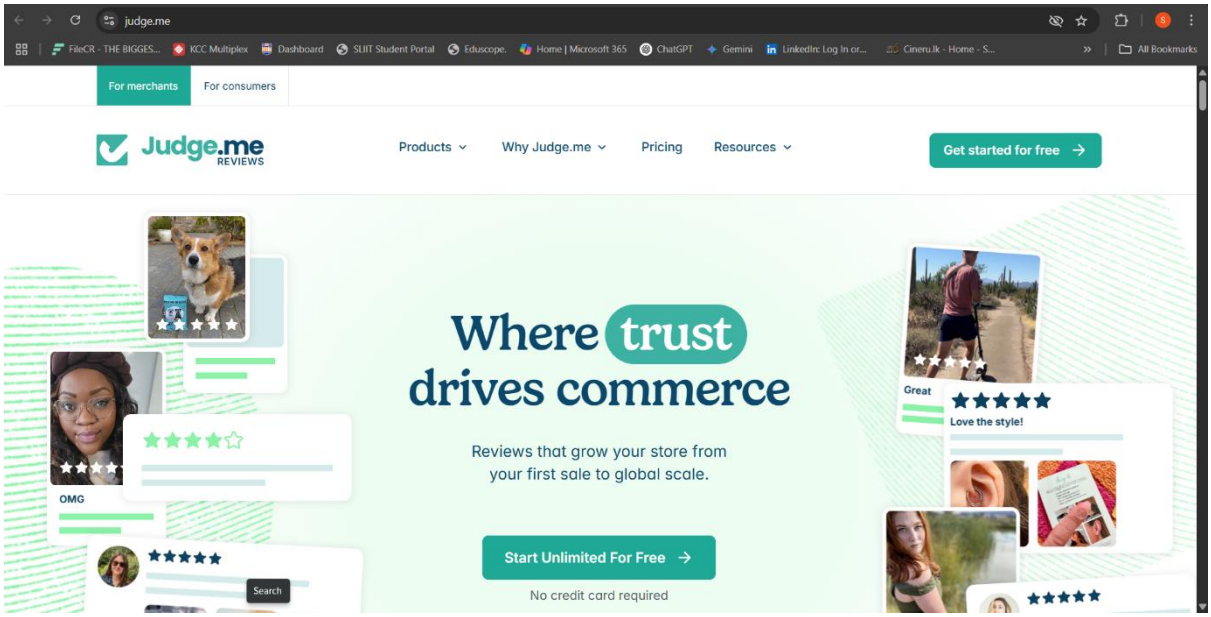
Cross-Origin Resource Sharing (CORS) Misconfiguration on https://judge.me

Vulnerability Description

Cross-Origin Resource Sharing (CORS) is a significant security feature that web browsers use to restrict how web applications access resources from other origins (domains, protocols, or ports). It prevents malicious websites from making unauthorized requests to servers on behalf of users.During a security audit of https://judge.me, it was discovered that the server returns an overly permissive CORS policy by responding with:

Access-Control-Allow-Origin: *

This wildcard allows any website to access judge.me-hosted resources. While the server does not seem to set Access-Control-Allow-Credentials: true in this specific case (which would increase the severity), the existence of the wildcard still significantly increases the risk surface, especially if any endpoints expose sensitive or personalized data. If origin restriction is not done correctly, attackers could create malicious websites for connecting to your application's API endpoints, which could deceive browsers to send requests and reveal sensitive information.



Other scans

Firewall Detection.

Before the automated scan with OWASP ZAP (2.16.0), Identify the Firewall by using Wafw00f Tool.

Open Ports Detection

Done the nmap scan for detect open ports.

Scanning for common Issues

These scans done by with Nikto.

Detect and exploit SQL injection flaws

Detect and exploit SQL injection flaws

Affected Components

- **Domain** https://judge.me/
- **Affected Endpoint** /reviews/verified_review_follow_up
- **Affected Header** Access-Control-Allow-Origin: *
- **Risk Level** Medium
- **Confidence** Medium
- **CWE-ID** 264 (Permissions, Privileges, and Access Controls)
- **WASC-ID** 14 (Cross-Domain Resource Sharing)

Impact Assessment

If the application inadvertently exposes sensitive data (e.g., user reviews, private API responses), an attacker can:

- **Steal Private Data:**Attackers may craft a malicious web page that steals information from the authenticated user's session.
- **Get Around Same-Origin Policy:**Attackers use CORS misconfigurations to bypass browsers' in-built protection mechanisms, like Same-Origin Policy (SOP).
- **Credential Stealing (Conditional):**If credentials or tokens would be exposed on such endpoints, attackers might hijack user sessions (even Access-Control-Allow-Credentials is not configured currently).
- **Leverage in Chained Attacks:** Combined with other weaknesses (weak tokens, IDORs), CORS misconfiguration could result in full account takeovers or broader system compromise.
- **Regulatory Non-Compliance:**Personal identifiable information (PII) exposure could constitute GDPR, CCPA, or other regulatory environment non-compliance, leading to fines and litigation risk.

Steps to Reproduce

1. Configure OWASP ZAP

- Launch OWASP ZAP and set the target https://judge.me/ .
- Add the site to the context and ensure it's in scope.

2. Perform an Automated Scan

    - Spider the site with the AJAX Spider and detect dynamic content.
    - Start an automated scan by clicking the "Attack" button in the Automated Scan window.
    - Wait until the scan finishes.

3. Analyze Alerts

    - Alert: "Cross-Domain Misconfiguration" (CWE-284).
    - URL: https://judge.me/reviews/reviews_verified_review_follow_up .
    - Evidence: Access-Control-Allow-Origin: *.
    - Risk: Medium, Confidence: Medium.
    - Source: Passive scan (10098 - Cross-Domain Misconfiguration).

4. Inspect HTTP Responses

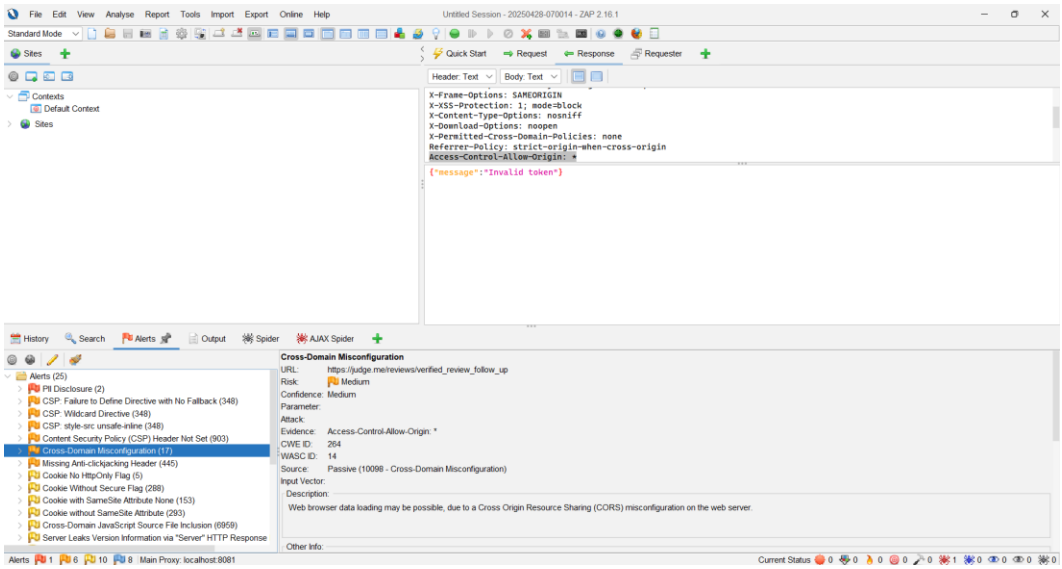
- Check the HTTP response headers for the target URL in the ZAP interface:

    - Observed: Access-Control-Allow-Origin: *.

Example response header (simplified):
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff

- The presence of Access-Control-Allow-Origin: * confirms the vulnerability, as it allows any origin to access the resource.

Proof of Concept Screenshot



Proposed Mitigation or Fix

To prevent unwanted cross-origin access due to relaxed CORS settings, the following are recommended:

- **Restrict Origins Strictly:** Substitute * with a whitelist of trusted domains:

  Access-Control-Allow-Origin: https://yourtrustedsite.com

- **Disable Wildcard When Credentials Needed:**
  When *Access-Control-Allow-Credentials: true* must be used, never use a wildcard. CORS spec forbids this combination.
- **Use Dynamic Origin Validation:** Dynamically reflect the Origin header only if it's on a trusted list of domains.
- **Limit Sensitive Endpoints:** Restrict sensitive operations (e.g., reading personal user info) to authenticated sessions and apply strict CORS checks.
- **Regular Security Audits:** Audit the CORS policy regularly during penetration testing cycles to prevent configuration drift.
- **Set Proper Pre-flight Controls:** Secure OPTIONS requests with strong server-side validation to reject invalid cross-origin requests at preflight.

Conclusion.

The https://judge.me/ OWASP ZAP scan identified a cross-domain misconfiguration by virtue of an overly liberal *Access-Control-Allow-Origin:* * header exposing the application to unwanted cross-origin access. This is a medium-severity bug that is concerning for a review website where trust and user data are everything since it could lead to data leaks or unauthorized access to review data. The absence of the other security controls, such as a CSP header, cookies without the **HttpOnly** attribute (also found by ZAP), increases the overall risk as it allows the respective attacks such as XSS to be more potent. Restricting the *Access-Control-Allow-Origin* header, sanitizing request origins, and the use of complementary security controls will significantly restrict the unauthorized access risk and enhance the security posture of the platform. This report contains succinct steps to reproduce the vulnerability and actionable mitigation advice. Regular caution, safe coding, and regular auditing are critical in making certain that the platform remains secure against future attacks, especially in the e-commerce review arena where data integrity is fundamental.