CORS Configuration Report

juice-shop.herokuapp.com_#_cors_2025-09-08_00-22-11

Executive Summary

This report details the Cross-Origin Resource Sharing (CORS) configuration analysis for https://juice-shop.herokuapp.com/#, conducted on 2025-09-08 at 00:22:11. The analysis revealed 3 potential vulnerabilities related to overly permissive CORS policies. Addressing these vulnerabilities is crucial to mitigate the risk of unauthorized access and potential data breaches. The report provides detailed findings and actionable mitigation strategies.

CORS Configuration Report

Target: https://juice-shop.herokuapp.com/# | Generated: 2025-09-08_00-22-11

Total findings: 3 (vulnerabilities: 3) | Elapsed: 4.50s

Detailed Findings

1. Wildcard Access-Control-Allow-Origin

Description: The server is configured to allow any origin to access its resources through the use of a wildcard (*) in the Access-Control-Allow-Origin header.

Risk: Medium. This configuration poses a security risk as it allows any website to make cross-origin requests to the server, potentially leading to unauthorized access to sensitive data if coupled with other vulnerabilities.

Evidence: The server responded with Access-Control-Allow-Origin: * when a request was made from a random origin (e.g., http://bzqtifak.attacker.site).

Mitigation: Replace the wildcard (*) with a specific list of trusted origins. Regularly review and update this list to ensure only authorized domains are permitted to access the server's resources.

2. Overly Permissive Methods

Description: The server allows a wide range of HTTP methods through the Access-Control-Allow-Methods header.

Risk: Medium. Allowing unnecessary methods (e.g., PUT, PATCH, DELETE) increases the attack surface and may enable attackers to perform unintended operations on the server if proper authorization checks are not in place.

Evidence: The Access-Control-Allow-Methods header includes GET, HEAD, PUT, PATCH, POST, and DELETE.

Mitigation: Restrict the Access-Control-Allow-Methods header to only include the methods that are actually required for cross-origin requests. For example, if only GET and POST are needed, remove the other methods.

3. Sensitive Headers Exposed

Description: The server allows sensitive headers, such as Authorization and X-Api-Key, to be exposed through the Access-Control-Allow-Headers header.

Risk: Medium. Exposing sensitive headers can lead to the leakage of authentication credentials or API keys, potentially allowing attackers to impersonate legitimate users or access protected resources.

Evidence: The Access-Control-Allow-Headers header includes Authorization and X-Api-Key.

Mitigation: Remove sensitive headers like Authorization and X-Api-Key from the Access-Control-Allow-Headers header. Only include the necessary custom headers that the client needs to send.