DOM XSS in AngularJS expression with angle brackets and double quotes HTML-encoded

Executive Summary

This report outlines the findings of a security assessment conducted on the search functionality of the target application. A DOM-based Cross-Site Scripting (DOM XSS) vulnerability was identified within an AngularJS expression. This vulnerability enables the execution of arbitrary JavaScript in the context of the user's browser, which could lead to session hijacking, defacement, or other malicious activities. The severity of this vulnerability is **high**, posing a significant risk to the integrity and security of user data.

Introduction

The purpose of this security assessment was to identify client-side vulnerabilities in the target application, specifically within the search feature. The assessment focused on evaluating the impact of DOM-based XSS when AngularJS is improperly handled.

Methodology

1. Used a browser and developer tools to analyze client-side scripts and DOM behavior.
2. Performed manual testing of input handling in the search feature.
3. Reviewed source code to detect AngularJS directives and expressions.
4. Tested crafted AngularJS payloads to exploit the XSS vulnerability.
5. Confirmed successful JavaScript execution via alert box as proof of concept.

Vulnerability Findings

- **Type:** DOM-Based Cross-Site Scripting (XSS)
- **Location**: Search input field within AngularJS expression
- **Severity:** High

Description

The application improperly embeds user input within an AngularJS expression handled by the `ng-app` directive. Since AngularJS scans HTML for expressions in double curly braces (`{{ }}`), this allows attackers to inject malicious expressions that get executed in the browser.

Proof of Concept (PoC)

1. Navigated to the search functionality and entered a test string.

2. Observed in the page source that the string was placed within an AngularJS directive.

3. Injected the payload:

   javascript

   CopyEdit

   `{{$on.constructor('alert(1)')()}}`

4. Upon clicking "Search", an alert box was triggered, confirming successful code execution.

Impact Assessment

Exploitation of this vulnerability allowed attackers to run arbitrary JavaScript in a victim's browser. This led to:

- Session hijacking

- Credential theft

- Unauthorized actions on behalf of the user

- Complete compromise of user accounts

Recommendations

1. Sanitize and escape all user input placed within AngularJS bindings.

2. Use AngularJS's strict contextual escaping (SCE) to prevent dangerous expressions.

3. Avoid using user-controlled data in AngularJS expressions directly.

4. Perform regular security testing focused on client-side JavaScript vulnerabilities.

Conclusion

The discovery of a DOM-based XSS vulnerability in the AngularJS-based search functionality highlights the importance of secure coding practices in front-end frameworks. Remediating this vulnerability and applying best practices in input handling will help prevent future exploitation and improve overall application security.