

# **DOM XSS using web messages and JSON.parse(DOM-based)**

## **Executive Summary**

This report outlines the discovery and successful exploitation of a **DOM-based Cross-Site Scripting (XSS)** vulnerability involving web messages and insecure usage of `JSON.parse()` on the client side. By injecting a crafted message into a trusted origin, an attacker can execute arbitrary JavaScript in the victim's browser. The vulnerability was used to trigger a browser `print()` function as proof-of-concept. This issue presents a **high severity risk** due to its client-side control and potential for full session hijacking.

## **Introduction**

The purpose of this security assessment was to evaluate the security posture of the target application, specifically identifying vulnerabilities in client-side code execution and message handling. The application utilizes web messaging (postMessage API) and insecurely parses incoming messages without verifying their origin or content.

## **Methodology**

1. Analyzed the application's homepage JavaScript to identify usage of `window.addEventListener("message", ...)`.
2. Confirmed use of `JSON.parse()` on user-supplied message content without validation or origin checks.
3. Identified a logic flow where a `load-channel` message type sets an iframe's `src` attribute based on attacker-controlled input.
4. Crafted a malicious iframe on the exploit server to deliver the payload to the application.
5. Verified JavaScript execution using the browser's `print()` function as evidence of successful exploitation.

## **Vulnerability Findings**

- **Type:** DOM-Based Cross-Site Scripting (XSS) via `postMessage`
- **Location:** Homepage JavaScript message handler
- **Severity:** High

## **Description**

The application contains a `message` event listener that:

- Accepts any origin (`targetOrigin: '\*'`)
- Parses message content using `JSON.parse()` without validation
- Acts on message properties without sanitization

A specially crafted JSON string with a `url` property set to `javascript:print()` triggers arbitrary script execution when injected into the iframe `src` attribute.

## **Proof of Concept (PoC)**

html

CopyEdit

```
<iframe src="https://website.ID.net/" onload='this.contentWindow.postMessage(
{"type":"load-channel","url":"javascript:print()"}', '*')></iframe>
```

This payload was stored and served from the exploit server. When visited by the victim, the script successfully triggered the `print()` function.

## **Impact Assessment**

- Full control over DOM via iframe injection
- Arbitrary JavaScript execution in the victim's browser
- Potential for session hijacking, data theft, CSRF token exposure, and more
- No origin validation means any site can exploit this vulnerability

## **Recommendations**

1. Always validate the origin of incoming `postMessage` events using `event.origin`.
2. Avoid directly using `JSON.parse()` on untrusted inputs; instead, implement robust input validation.
3. Never allow `javascript:` URIs to be set as the `src` of iframe or other elements.
4. Sanitize all user-controlled data before applying it to the DOM.
5. Use a Content Security Policy (CSP) to mitigate the impact of DOM XSS attacks.

## **Conclusion**

The exploitation of this DOM-based XSS vulnerability via the `postMessage` API demonstrates the significant risk posed by unvalidated cross-origin communication and unsafe JSON parsing. Fixing this vulnerability and enforcing strict message handling rules are essential to maintaining application security and user trust.

Kama Gajah