Client-side prototype pollution in third-party libraries

Executive Summary

This assessment uncovered a **client-side prototype pollution vulnerability** in the target application, made exploitable via a gadget in a third-party JavaScript library. The vulnerability led to DOM-based Cross-Site Scripting (XSS), allowing attackers to execute arbitrary JavaScript in the context of another user. Exploitation required no user interaction beyond visiting a malicious link. This issue presents a **high severity risk** due to the potential for session hijacking, credential theft, and user impersonation.

Introduction

The goal of this test was to assess the security of client-side JavaScript code and its integration with third-party libraries. Specifically, the focus was on identifying DOM-based vulnerabilities introduced through prototype pollution a class of attacks increasingly affecting modern JavaScript applications.

Methodology

1. Used Burp Suite's DOM Invader for automated client-side vulnerability analysis.
2. Enabled prototype pollution detection to identify sources and sinks in the DOM.
3. Scanned the application for gadget chains linked to vulnerable sources.
4. Generated and tested proof-of-concept XSS payloads using built-in DOM Invader features.
5. Crafted and delivered a malicious link via the provided exploit server.

Vulnerability Findings

- **Type:** Client-Side Prototype Pollution → DOM-Based XSS
- **Location:** Hash property (URL fragment string)
- **Severity:** High
- **Source:** `location.hash`
- **Sink:** `setTimeout()`
- **Gadget:** `hitCallback` in a minified third-party JavaScript library

Description

Prototype pollution occurs when attackers manipulate the `__proto__` property to inject properties into global objects. In this case, the application's reliance on a vulnerable third-party library allowed an attacker to inject a function into the global prototype, which was later executed via a gadget (`hitCallback`) reaching a JavaScript sink (`setTimeout()`).

Proof of Concept (PoC)

1. Enabled **DOM Invader** in Burp Suite's browser and visited the lab.
2. Detected prototype pollution via the URL fragment:

   markdown

   CopyEdit

   `#__proto__[hitCallback]=alert(1)`
3. DOM Invader identified `setTimeout()` as the sink.
4. Verified successful execution of `alert(1)` via auto-generated PoC.
5. Created and tested a malicious exploit URL:

   html

   CopyEdit
6. Delivered the payload via the exploit server. The victim's browser executed `alert(document.cookie)` as expected.

Impact Assessment

This vulnerability allows:

- DOM XSS, enabling full JavaScript execution in the victim's browser.
- Session hijacking through `document.cookie` theft.
- Credential theft, defacement, and data manipulation.
- Abuse of trust in third-party libraries, often overlooked in audits.

Due to the widespread use of third-party JavaScript, this vulnerability poses a **high risk** in real-world environments.

Recommendations

1. Sanitize user-controlled properties like `__proto__` before merging or processing objects.
2. Use Object.create(null) to create objects without a prototype chain.
3. Regularly audit and update third-party libraries to patch known gadget vulnerabilities.
4. Implement Content Security Policy (CSP) to mitigate the impact of DOM XSS.
5. Utilize tools like DOM Invader or linters during development to flag insecure patterns.

Conclusion

The discovery of a prototype pollution vulnerability chained to a gadget in a third-party library illustrates the dangers of unvalidated client-side inputs and unsafe library integration. Immediate remediation and proactive dependency management are critical to prevent exploitation and maintain user trust.