# DOM-based cookie manipulation in an E-commerce webapp

Ref: CVE-2026-24389

## Executive Summary

This report outlines the identification and exploitation of a DOM-based Cross-Site Scripting vulnerability arising from client-side cookie manipulation. The vulnerability enables an attacker to inject a malicious script into a browser cookie (`lastViewedProduct`) that is later processed by client-side JavaScript without proper sanitization. This leads to the execution of arbitrary JavaScript in the victim’s browser, confirming a high-severity issue.

## Introduction

The objective of this assessment was to evaluate the handling of client-side data, particularly browser cookies, for potential DOM-based injection flaws. This test focused on whether user-controlled data could be stored in cookies and later interpreted unsafely by client-side scripts.

## Methodology

1. Analyzed application behavior by browsing product pages and monitoring cookie changes.
2. Identified that `lastViewedProduct` is set client-side based on the last visited product page.
3. Crafted a product URL with embedded JavaScript to poison the `lastViewedProduct` cookie.
4. Verified that the cookie is later parsed and executed by client-side code on the homepage.
5. Delivered an exploit using an iframe that both sets the cookie and redirects to the homepage.

## Vulnerability Findings

- **Type:** DOM-Based Cross-Site Scripting via Cookie Injection
- **Location**: Homepage JavaScript (uses `lastViewedProduct` cookie)
- **Severity**: High

## Description

The application stores the last visited product page in a cookie named `lastViewedProduct`. This value is later read and inserted into the DOM on the homepage without any validation or encoding. By navigating the victim to a malicious product URL containing a script tag, an attacker can poison this cookie and trigger code execution the next time the homepage is loaded.

**Proof of Concept (PoC)**

html

CopyEdit

```
<iframe src="https://website.ID.net/product?productId=1&'><script>print()</script>" onload="if(!window.x)this.src='https://website.ID.net';window.x=1;"> </iframe>
```

**How it works**

- The `iframe` first loads a malicious product URL that embeds a script in the URL itself.
- This sets the `lastViewedProduct` cookie with the JavaScript payload.
- The iframe's `onload` handler immediately redirects to the homepage.
- When the homepage loads, the browser executes the script via the poisoned cookie.

**Impact Assessment**

-Full DOM-based XSS without user interaction.

- Attacker can execute arbitrary JavaScript in the victim's browser.
- Enables session hijacking, credential theft, or malware injection.
- No server-side validation or sanitization of cookie content.
- Very difficult for users to detect the manipulation.

**Recommendations**

1. Never trust data from browser cookies on the client side without proper validation or encoding.

2. Use `document.createElement()` and `textContent` instead of `innerHTML` when inserting dynamic data.

3. Apply context-aware output encoding for all client-side dynamic content.

4. Implement a strict Content Security Policy (CSP) to reduce the impact of XSS attacks.

5. Regularly audit and test client-side JavaScript for DOM-based vulnerabilities.

## Conclusion

This assessment confirmed the presence of a serious DOM XSS vulnerability via client-side cookie manipulation. The exploit demonstrates how client-side trust in unvalidated cookie data can be abused to achieve full script execution. Remediation should focus on robust validation, safe DOM API usage, and minimizing reliance on client-side state for critical operations.