Security Assessment Report

Vulnerability Type: DOM-Based Cross-Site Scripting (XSS)
Target: PortSwigger Academy Lab – *DOM XSS in document.write sink using source location.search*
Assessor: Abdelalim Saada
Tools Used: Browser Inspector, DevTools, PortSwigger Academy

1. Vulnerability Description

DOM-Based XSS occurs when the client-side JavaScript takes untrusted data from a user-controllable source (like the URL) and passes it into a sink (e.g., `document.write`, `innerHTML`, etc.) without sanitization. This allows the attacker to inject and execute malicious scripts **entirely on the client side**.

2. Steps to Reproduce

1. **Enter the Lab**:
   - PortSwigger Lab: *DOM XSS in `document.write` using `location.search`*
2. **Test the Search Function**:
   - Typed:
   - `test`
   - Observed `test` reflected **in the HTML source**, used in `<img src=...>`
3. **Review Source Code (in DevTools)**:
   - Found the vulnerable JS code:
   - `function trackSearch(query) {`
   - `    document.write('<img src="/resources/images/tracker.gif?searchTerms='+query+'">');`
   - `}`
   - `var query = (new URLSearchParams(window.location.search)).get('search');`
   - `if(query) {`
   - `    trackSearch(query);`
   - `}`
   - Vulnerable sink: `document.write()`
   - Vulnerable source: `location.search` (via `URLSearchParams`)
4. **Inject Payload**:
   - In the search bar, entered:
   - `test"onload="alert(123)`
   - The payload was reflected in the `<img src=...>`:
   - `<img src="/resources/images/tracker.gif?searchTerms=test"onload="alert(123)">`
   - Alert was triggered, lab marked **solved**
5. **Captured Screenshots**:
   - Input with payload
   - Source code with vulnerable JavaScript
   - Alert popup

3. Root Cause

- **Client-side JavaScript** directly inserted URL parameter into the DOM via `document.write()` **without sanitization or encoding**
- The `searchTerms` parameter became part of an `img` tag, which allowed triggering the `onload` event

4. Risk Assessment

Category	Details
Impact	High – JavaScript code execution on victim’s browser
Likelihood	High – Payload accepted via URL, no validation
OWASP	A03:2021 – Injection

5. Mitigation Recommendations

1. **Avoid `document.write()`**
   - Use `textContent` or DOM manipulation APIs like `createElement` instead
2. **Sanitize User Input**
   - Validate and encode inputs before inserting into the DOM
3. **Context-Aware Escaping**
   - Escape data depending on the context (e.g., HTML attribute, script, etc.)
4. **Content Security Policy (CSP)**
   - Enforce a restrictive CSP to block inline scripts

6. OWASP Mapping

OWASP Top 10	Vulnerability Type	Found
A03:2021	Injection (DOM-Based XSS)	☑ Yes