**CVE Submission Request**

**Author: Nick Brinkley**

**Software**: CAE Learning Space Enterprise with Intuity License

**Image Version**: 267r

**Patch Level**: 639

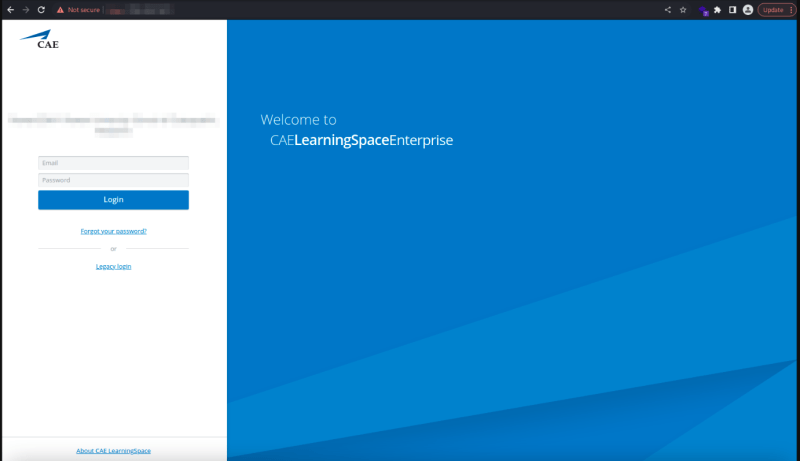
**Vulnerability**: DOM Based XSS

**Description**: The CAE Learning Space Enterprise management software is vulnerable to a DOM Based XSS attack. A DOM Based XSS attack is when a user sends an executed payload to the host and as a result, it modifies the DOM environment in the victim's browser from the original client-side script. This attack was found through manually searching burp requests, running tested parameters through a XSS finder tool called Dalfox (<https://github.com/hahwul/dalfox>), then verifying the DOM of the affected host to view the behavior of the application.

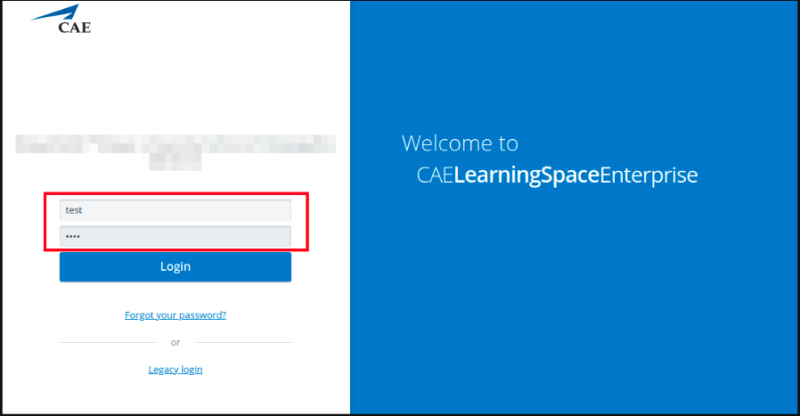
**Impact**: An attacker can insert malicious code within the application DOM. This code can be executed causing the application to run client code unexpectedly. For testing purposes, the impact showed a payload consisting of a basic alert being called within the DOM inspection from the “ontouchmove” and “onpointerup” JavaScript functions. Then after sending the payload, the tester could see within the DOM itself the break of the “value” html value and then the JavaScript function itself.

**Recommendation**: To prevent DOM XSS, you must sanitize all untrusted data, even if it is only used in client-side scripts. If you must use user input on your page, always use it in the text context, never as HTML tags or any other potential code. If you can, entirely avoid using user input, especially if it affects DOM elements.

**Reproduction steps:**

›

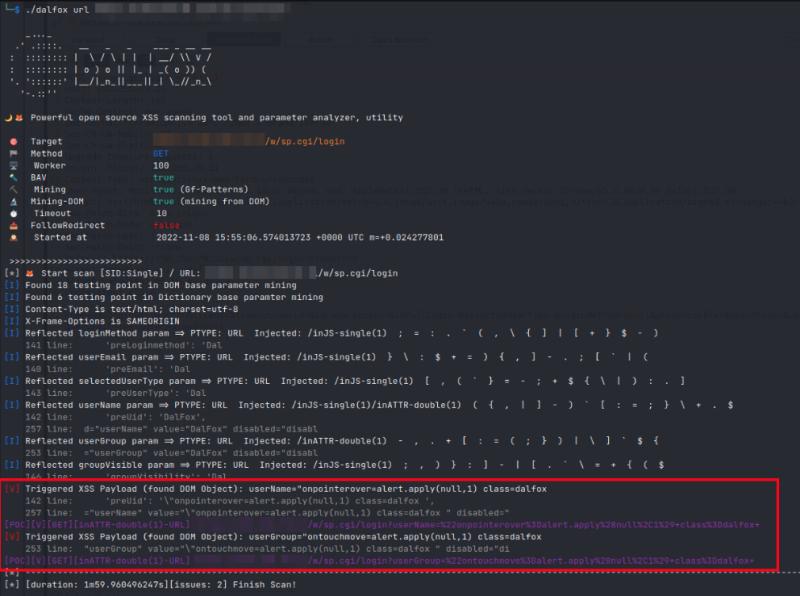
**Figure 1:** Hosted page of CAE Learning Space Enterprise



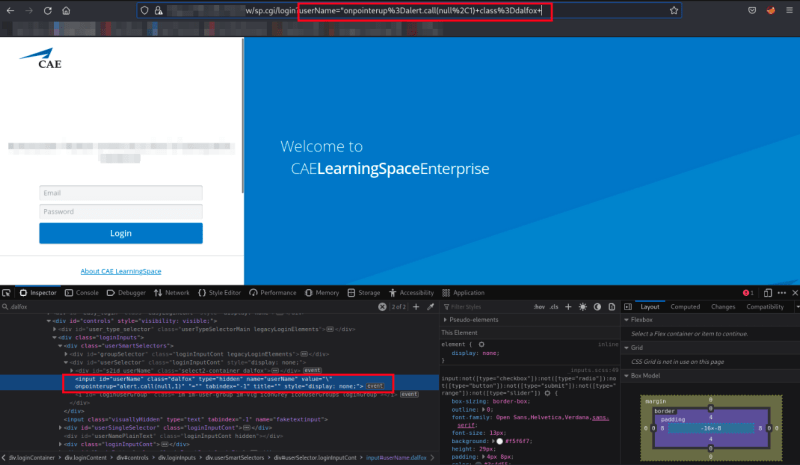
**Figure 2:** Dummy Text for Intercepted Parameter



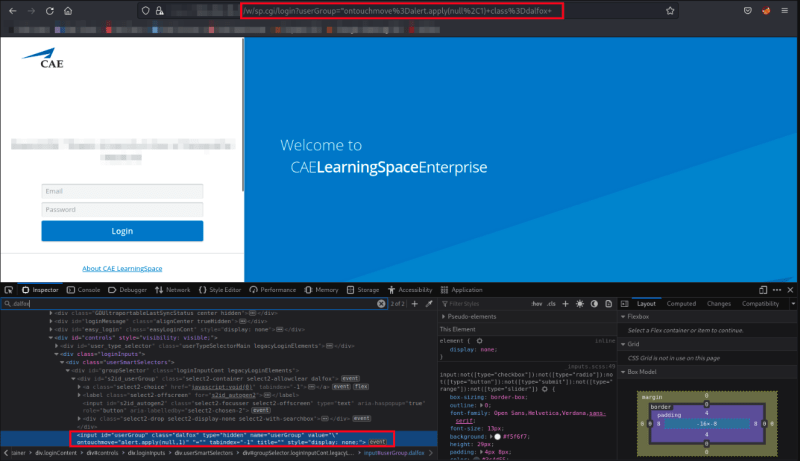
**Figure 3:** Intercepted Burp Request with Dummy test Parameters



**Figure 4:** Running the XSS finder tool called “Dalfox” which located two successfully launched DOM Based XSS payloads to the userName parameter and userGroup parameter



**Figure 5:** Proof of the manipulated DOM with the Malicious Payload Injected into the HTML – “onpointerup” Javascript Function



**Figure 6:** Proof of the manipulated DOM with the Malicious Payload Injected into the HTML – “ontouchmove” Javascript Function