SCENARIO

The application contains a reflected DOM based cross site scripting vulnerability. Reflected DOM vulnerabilities occur when the server-side application processes data from a request and echoes the data in the response. A script on the page then processes the reflected data in an unsafe way, ultimately writing it to a dangerous sink.

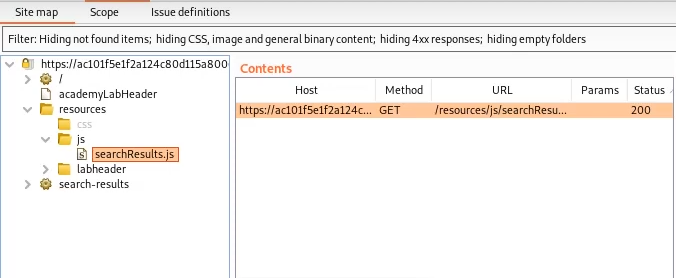
**PROCEDURE**

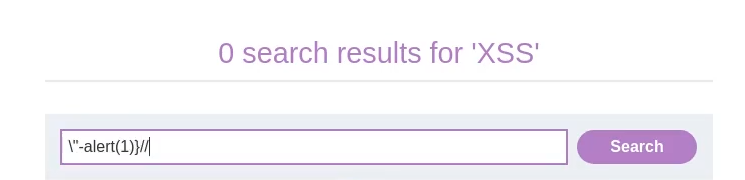
1. Go to the vulnerable application and search for some random string.
2. Use BurpSuite’s Target page’s Site Map tab.
3. We can see that there is a page named **searchresults.js** which contains all the JavaScript logic code and there’s no code in the page’s source.
4. Study the search requests and try sending different strings to see that we are getting the response as a JSON object.
5. Now we will try to break out of that JSON object by creating a payload based on the response we are getting.
6. In the end, enter the payload into the search box in order to trigger an alert.

**PAYLOAD**

\"-alert(1)}//

**PROOF OF CONCEPT**

****

****

**REMEDIATION**

1. **Validate Input on the Client Side:** All user inputs should be validated on the client-side before processing. Avoid using untrusted or unsanitized data in the DOM. Establish a whitelist of accepted values and deny everything else. Ensure that special characters, such as double quotes and slashes, are adequately sanitized or escaped to prevent breaking out of data structures.
2. **Safely Interact with the DOM:** Instead of using dangerous functions or methods that can write arbitrary content into the DOM, use safer alternatives. For example, instead of innerHTML, consider using textContent or innerText. When using frameworks or libraries, utilize their built-in mechanisms to update the DOM securely.
3. **Content Security Policy (CSP):** Implement a robust Content Security Policy (CSP) that blocks the execution of inline scripts. A strict CSP will deny the execution of inline scripts, reducing the risk associated with DOM-based XSS attacks.