SCENARIO

The application contains a stored DOM based cross site scripting vulnerability in the blog comment functionality. The comment on the page then processes the comment data in an unsafe way, ultimately triggering malicious events. We will try to trigger an alert message by injecting a payload.

**PROCEDURE**

1. Go to the vulnerable application and navigate to any blogs comment section and post a comment.
2. Use BurpSuite’s Target page’s Site Map tab and Proxy Tab to study the request.
3. We can see that there is a page named**.js** which contains all the JavaScript logic code and there’s no code in the page’s source.
4. Study the post comment requests and we can see that the application is using the replace function to encode <> into strings but it only encodes the first occurences.
5. Now we will try to break out of that <> so, we will inject a specially defined payload in order to bypass that encoding and trigger an alert.
6. In the end, enter the payload into the comment box in order to trigger an alert and whenever someone will go to that blog, it will trigger that alert.

**PAYOAD**

\"-alert(1)}//

**REMEDIATION**

1. **Encode User Input:** Ensure that any user input, such as blog comments, is consistently and fully encoded when rendered back to any users. All special characters (like <, >, ', ", &, etc.) should be translated into their HTML-encoded equivalents (&lt;, &gt;, &apos;, &quot;, &amp;, etc.).
2. **Avoid Direct DOM Manipulation:** Directly manipulating the DOM based on user inputs without validation or encoding can lead to vulnerabilities. Always use safe methods or frameworks that automatically handle the encoding.
3. **Whitelist Input:** Implement an input whitelist. Allow only known safe values. If special characters are not necessary for comments, they should be outright denied or fully sanitized.