SCENARIO

The application contains a HTML Janitor library, which is vulnerable to DOM clobbering. We’ll try to exploit this vulnerability by forcing the user to inject a cookie that will cause XSS on the page and call the alert() function.

**PROCEDURE**

1. Access the application and go to any blog post.
2. Then add the Payload 1 into the comment section of that blog.
3. Then go to the exploit server and paste the Payload 2 into the body of the exploit.
4. The library uses the attributes property to filter HTML attributes. However, it is still possible to clobber the attributes property itself, causing the length to be undefined. This allows us to inject any attributes we want into the form element. In this case, we use the onfocus attribute to smuggle the print() function. When the iframe is loaded, after a 500ms delay, it adds the #x fragment to the end of the page URL. The delay is necessary to make sure that the comment containing the injection is loaded before the JavaScript is executed. This causes the browser to focus on the element with the ID "x", which is the form we created inside the comment. The onfocus event handler then calls the print() function.

**PAYLOAD**

1. <form id=x tabindex=0 onfocus=print()><input id=attributes>
2. <iframe src=https://YOUR-LAB-ID.web-security-academy.net/post?postId=3 onload="setTimeout(()=>this.src=this.src+'#x',500)">

**PROOF OF CONCEPT**

**REMEDIATION**

1. **Thoroughly Review and Update Libraries:** If you use third-party libraries like HTML Janitor or DOMPurify, ensure they're always updated to their latest versions. Developers of these libraries often release updates to address newly discovered vulnerabilities.
2. **Explicit DOM References:** Use explicit DOM references instead of implicit global ones. For instance, document.getElementById('someID') is preferred over window.someID.
3. **Disallow Potentially Harmful HTML Elements and Attributes:** Even if the janitor library allows them, disallow elements and attributes that can be used to build DOM clobbering payloads, especially forms and input fields.
4. **Use Non-Clobberable Methods:** To access DOM elements, use methods that cannot be easily clobbered. For instance, use document.querySelector() or document.querySelectorAll() instead of other global methods.