SCENARIO

The application contains a reflected cross site scripting vulnerability in the home page as it reflects the user supplied input in canonical tags and escape angle brackets when we press a combination of keys together. We will try to trigger an alert message by injecting a payload into the application.

**PROCEDURE**

1. Go to the vulnerable application and inspect the source of the HTML page and we can find the href tag enclosed with canonical relation.
2. Canonical tag tells the browser that if you find a lot of copies of the required page then this is the original page.
3. In our scenario we can see that our page’s URL is enclosed with canonical tag so we will break out of it.
4. Then we will inject the payload which will force the browser to trigger an alert whenever a particular combination of keys is pressed.

**PAYLOAD**

/?'accesskey='x'onclick='alert(“HACKED!”)

**REMEDIATION**

1. **Output Encoding:** Always encode user input before reflecting it back on the webpage. In this context, especially ensure that single quotes, double quotes, and other special characters are properly HTML-encoded to prevent breaking out of HTML attributes.
2. **Whitelist Input Validation:** Instead of blocking or filtering out known bad inputs, use a whitelist approach. Clearly define what constitutes valid input, and reject any input that doesn't meet this criteria. For instance, if a URL parameter should only accept alphanumeric characters, block or sanitize any input containing special characters.
3. **Avoid Reflecting User Input:** If possible, avoid reflecting user input directly back to the webpage. If user input must be reflected, ensure it is both validated and sanitized.
4. **Use Safe APIs:** Make use of safe frameworks and libraries that automatically handle user input in a secure manner. For instance, instead of using traditional string concatenation, utilize functions that safely encode strings for insertion into HTML.