SCENARIO

The application contains a reflected cross site scripting vulnerability in the search box functionality because it backslashes any unexpected character including single quotes automatically in order to prevent user from breaking user out of the attribute. We will try to trigger an alert message by injecting a payload into the application.

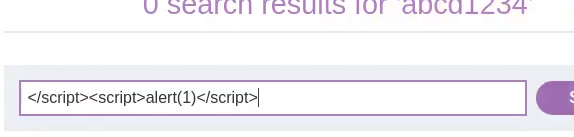
**PROCEDURE**

1. Go to the vulnerable application and search for anything containing a single quote and we see that it got back slashed.
2. In this scenario we will try to break out of the parent </script> tag in order to execute any malicious script.
3. At the end, we will create a payload depending on the conditions of the application so that we can generate an alert to exploit the vulnerability.

**PAYLOAD**

</script><script>alert("HACKED!")</script>

**PROOF OF CONCEPT**

****

**REMEDIATION**

1. **Proper Output Encoding:** Ensure that all user data is HTML-encoded before being returned and rendered by the browser. This will ensure that any characters that have special meaning in HTML (like <, >, and &) are properly encoded and not interpreted by the browser.
2. **Validate and Sanitize Input:** All user inputs should be validated against a well-defined pattern. Only accept data that matches the expected pattern. Any other data should be rejected. For instance, if a field is expected to contain only numbers, any input that contains non-numeric characters should be rejected.
3. **Avoid Reflecting User Input:** If it's not necessary, avoid echoing user data back to the page. If there's a need to reflect user data, ensure it is properly encoded and sanitized.
4. **Use Safe APIs and Libraries:** Utilize frameworks and libraries that automatically escape user input. For JavaScript context, instead of eval() or document.write(), use safer alternatives like textContent or setAttribute.