SCENARIO

The application contains a simple DOM based cross site scripting vulnerability in the home page which uses jQuery's **$()** selector function to auto-scroll to a given post, whose title is passed via the **location.hash** property, so we will try to exploit it by delivering an exploit using our exploit server to force the application to show the print dialog box.

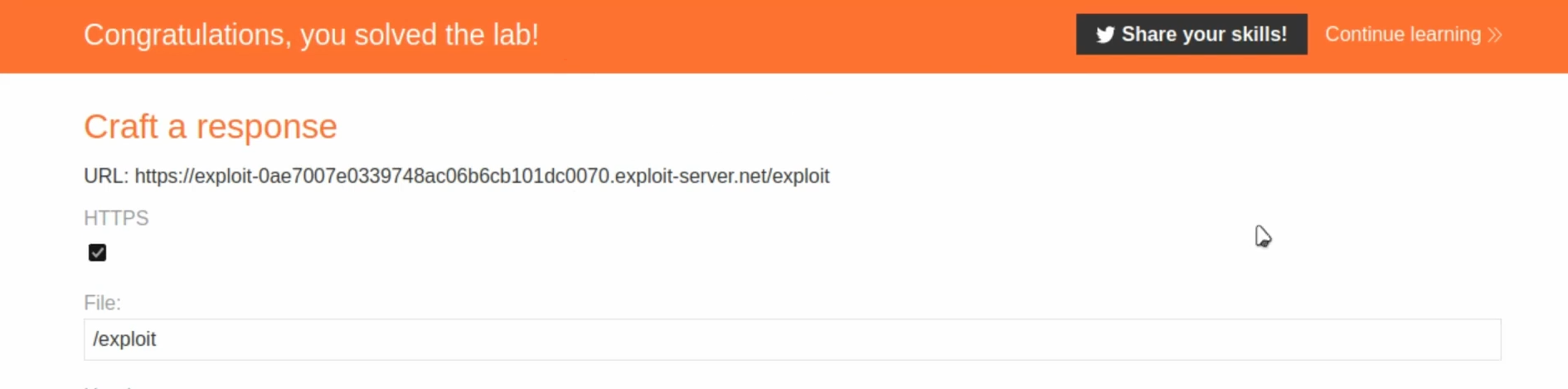
**PROCEDURE**

1. Go to the vulnerable web application’s home page.
2. Open the element inspector tab and got to debugger tab in that.
3. In there we can see that location.hash value is passed and is used to scroll to any given post.
4. Now go to the exploit server and paste the following payload into the body of your exploit, then store the exploit, test the exploit and in the end deliver the exploit to the victim to force the application into viewing a print page dialog box in the victim’s browser.

**PAYOAD**

<iframe src="https://YOUR-LAB-ID.web-security-academy.net/#" onload="this.src+='<img src=0 onerror=print()>'"></iframe>

**PROOF OF CONCEPT**

****

**REMEDIATION**

1. **Avoid Direct Input:** Do not allow direct input from user-controlled sources (like location.hash) into functions like jQuery's $() selector without proper sanitization. By doing this, you avoid the possibility of allowing malicious scripts to execute due to user input.
2. **Validate and Sanitize:** Always validate and sanitize user input, even if it comes from seemingly benign sources like URL fragments. For instance, in this case, ensure that the input for the auto-scroll function matches expected patterns (like post titles) and strip out any HTML or script content.
3. **Use Safe Functions:** When working with jQuery, be aware of functions that can execute arbitrary code. Instead of using $() selector with user-controlled data, employ safer methods that don't interpret the input as code or use direct DOM manipulation methods provided by the browser, which don't execute code.