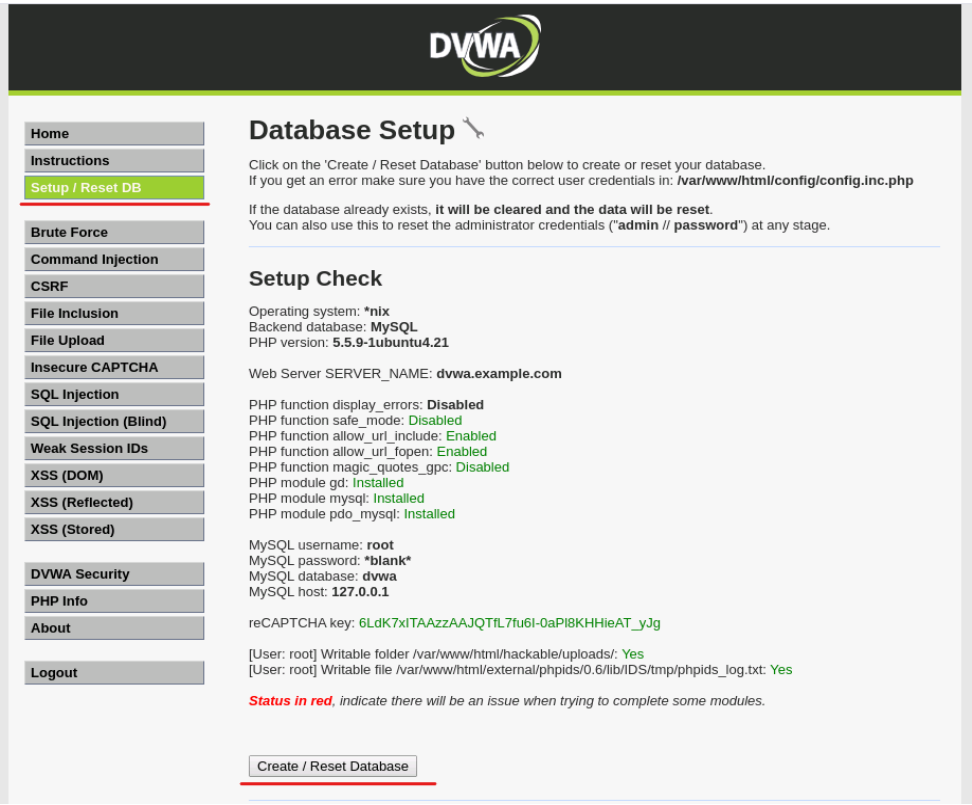
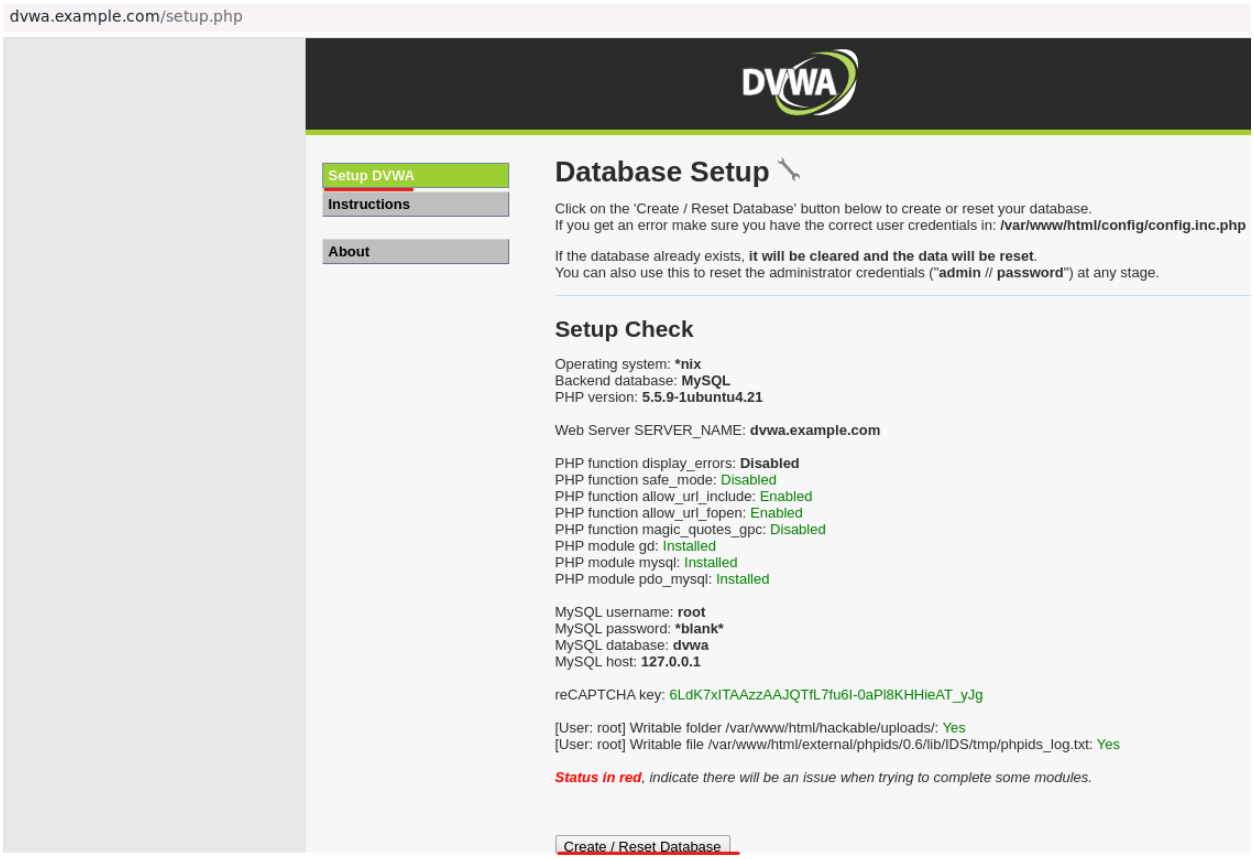
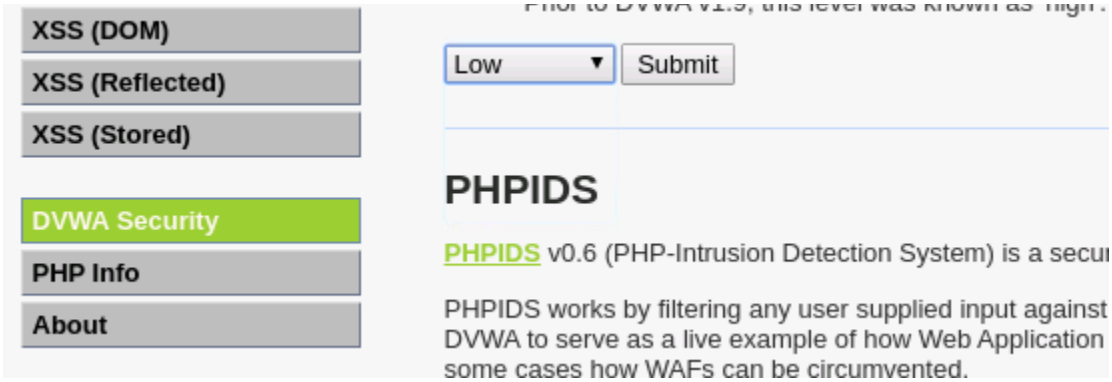
Stored Cross-Site Scripting

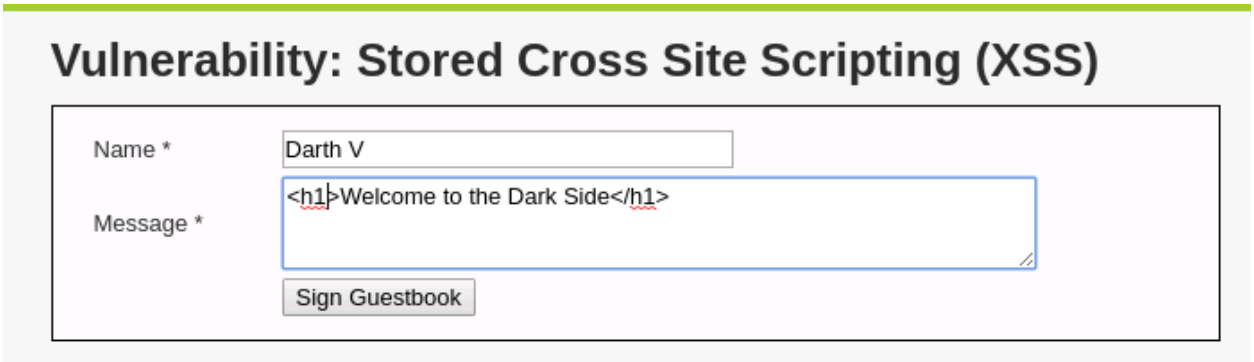


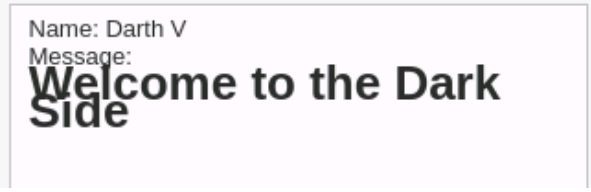


For this task, I need to log in to the DVWA and set the security level to low. Then I clicked the XSS (Stored) button on the left column.

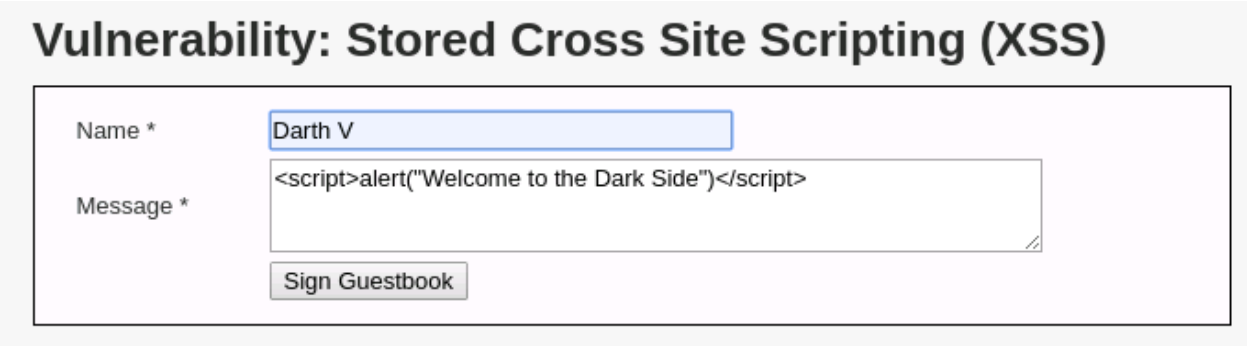


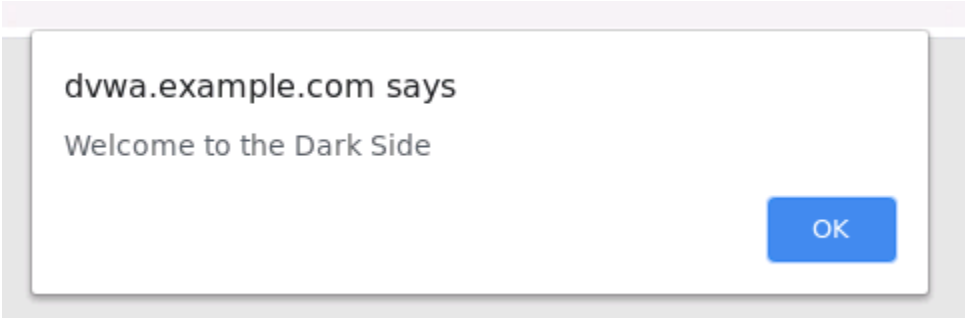
- In the form, I typed a name in the name submission box. Then I typed a message wrapped in a html h1 tag: **`<h1>message here</h1>`** and then clicked on the **Sign Guestbook** button.



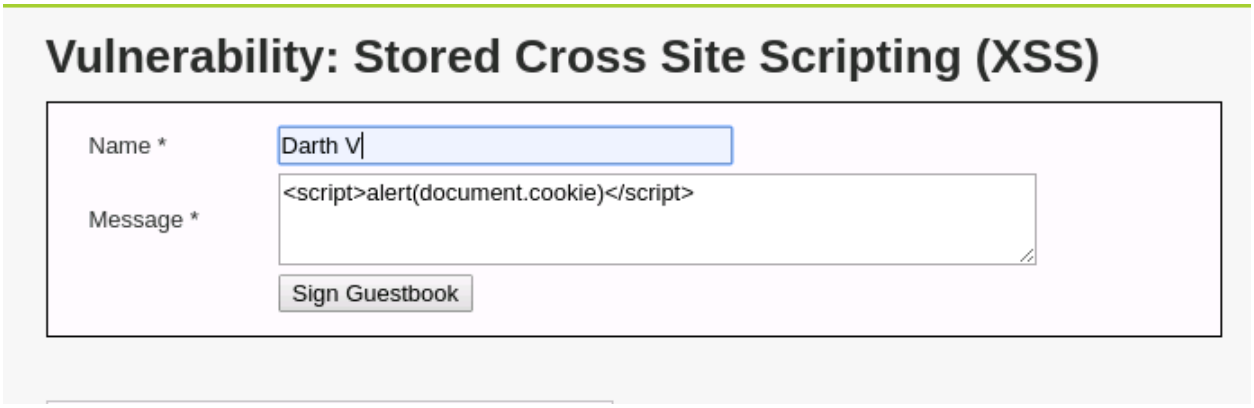


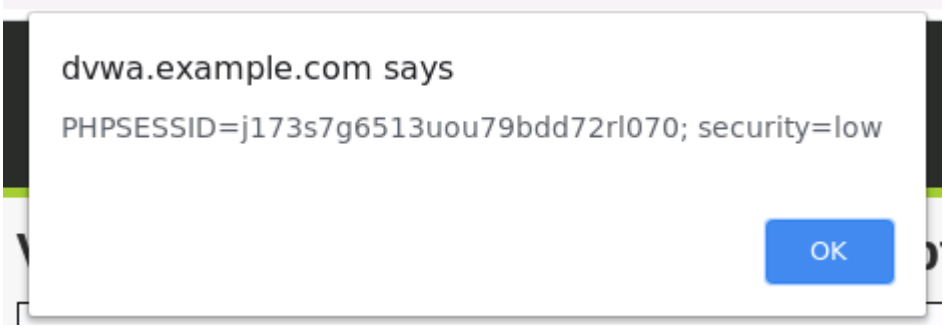
Notice that the <h1> tag has changed the message to be a heading. This means we can inject HTML. Let's try a script tag next. Type the following: **`<script>alert("Welcome to the Dark Side")</script>`** and click on the **Sign Guestbook** button.





I then clicked on the **OK** button. From here, it seems logical to grab the session cookie. I typed **`<script>alert(document. cookie)</script>`** and clicked on the **Sign Guestbook** button. Received both prompts as they are stored on the page.

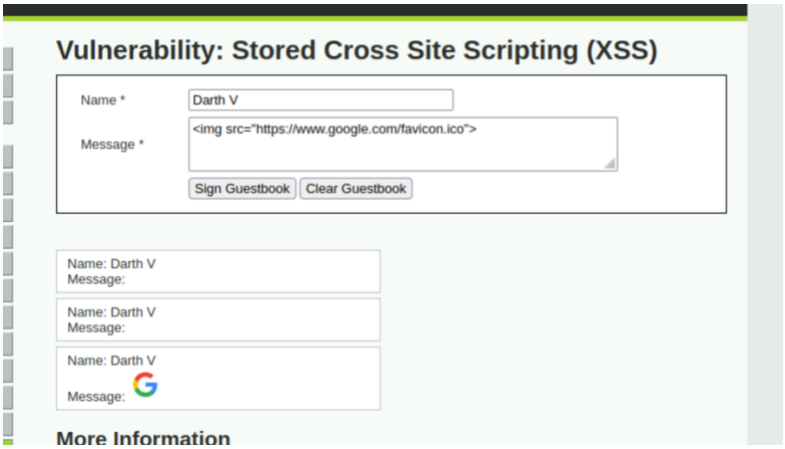




To be sure the XSS is stored on the page, I clicked on any other DWVA page and then clicked on the XSS (Stored) page again. Both of the XSS prompts appeared one at a time after clicking OK each time I saw a pop-up. If an attacker can get Stored XSS on a site, they can potentially gather all session cookies and even credentials by setting up a JavaScript keylogger.

Question 1: do some research online regarding basic JavaScript syntax. And then submit another message to display the image (https://www.google.com/favicon.ico) on the Stored XSS webpage. Explain what you have done to make it happen, and provide a screenshot of the webpage showing you have successfully inserted the image into the page.

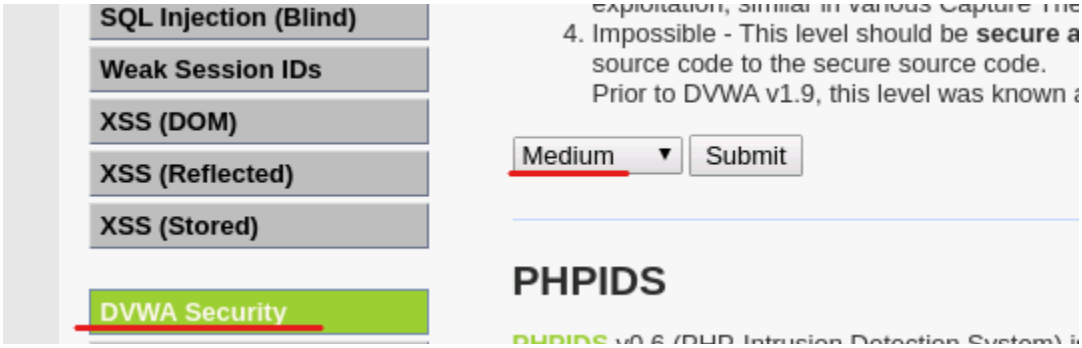
To display an image on a Stored XSS vulnerability on the DVWA webpage, I created an <img> element with the src attribute set to the URL of the desired image (https://www.google.com/favicon.ico). This crafted payload was then injected into a vulnerable input field on the webpage, exploiting the XSS vulnerability. When the page is loaded, the browser interprets the injected HTML code and renders the image specified in the payload.

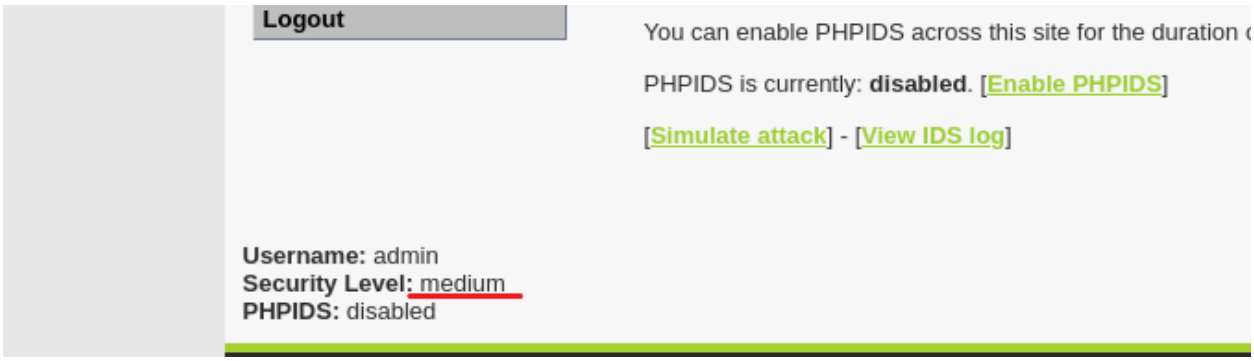


Task 2: DVWA Stored XSS on Medium Security

I reset the DB as shown on the first page. After resetting, I went to the XSS(stored) page and verified all previous messages I had posted had been removed.

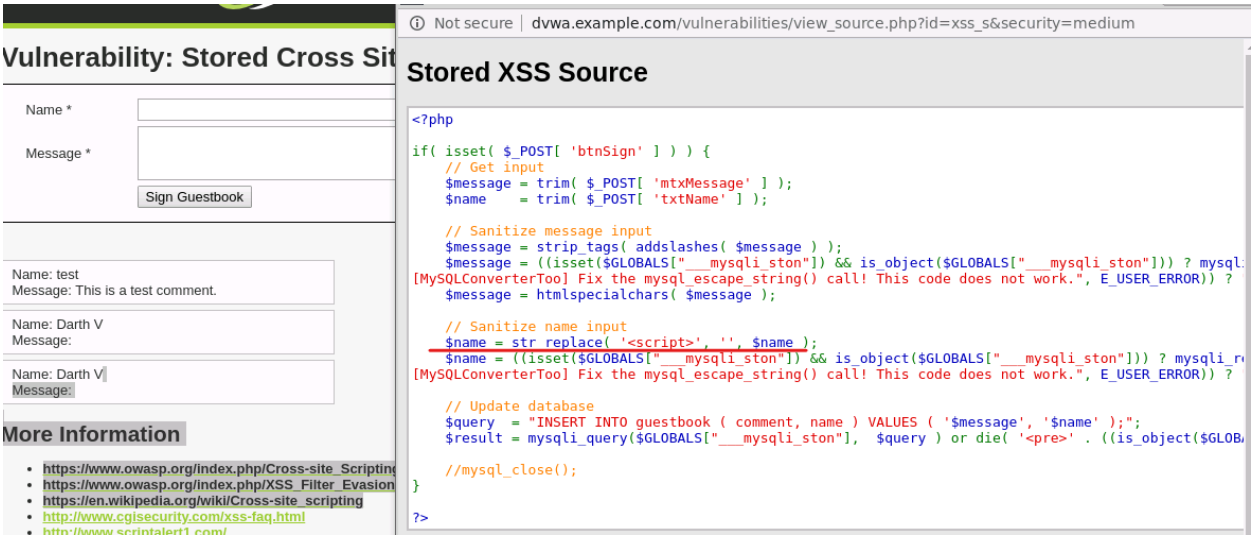
On the sidebar menu, I clicked on the DVWA Security button and used the dropdown menu to change the security settings to Medium. Sometimes I have to do this more than once for it to work. I then checked the bottom of the page to be sure the settings had changed to medium.





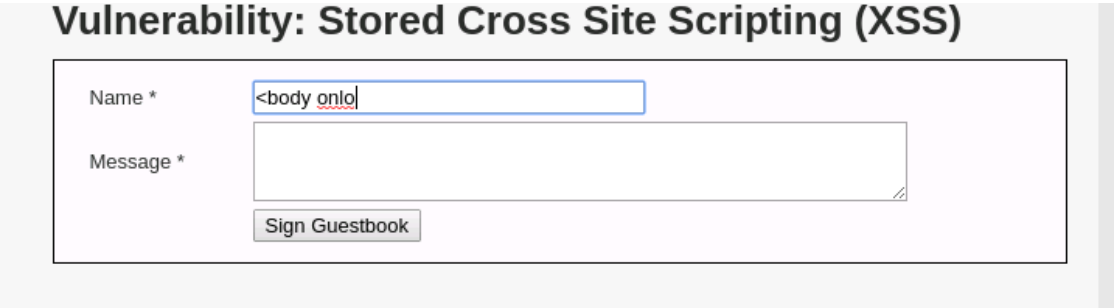
I tried to use the previous technique by directly inserting a script, i.e., putting `<script>alert("Welcome to the Dark Side")</script>` in the message box, but I found that the script will not be executed any longer.

I then navigated to the Stored XSS page and clicked OK on the alerts I created in the first task. If I look at the source by clicking on the **View Source** button located on the bottom right of the page, I can see there are a few changes to the security settings. The name submission box has been filtered by replacing the string **`<script>`** with double quotations.



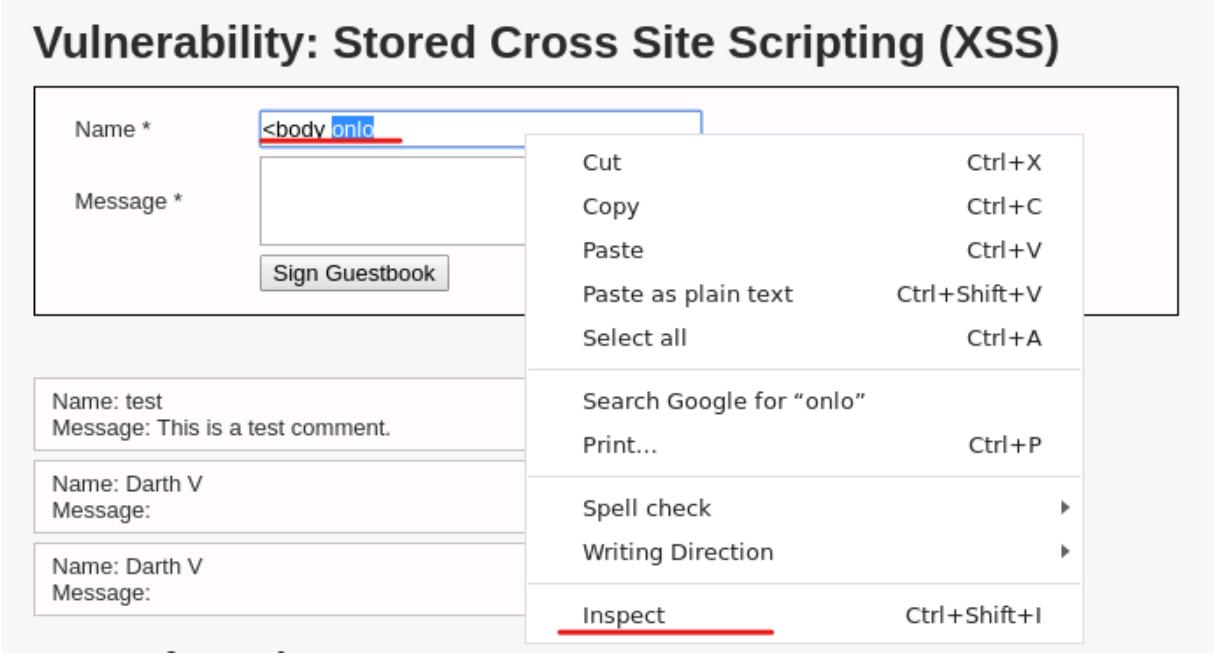
In order to still exploit this, I need to use an exploit that does not require **`<script>`**. I tried to place a script into the name submission box.

- I closed out of viewing the source.
- In the name submission box, I typed: **`<body onload=alert("medium")>`**



As you can see, the name submission box is only allowing 10 characters. I may be able to modify this using the developer tools.

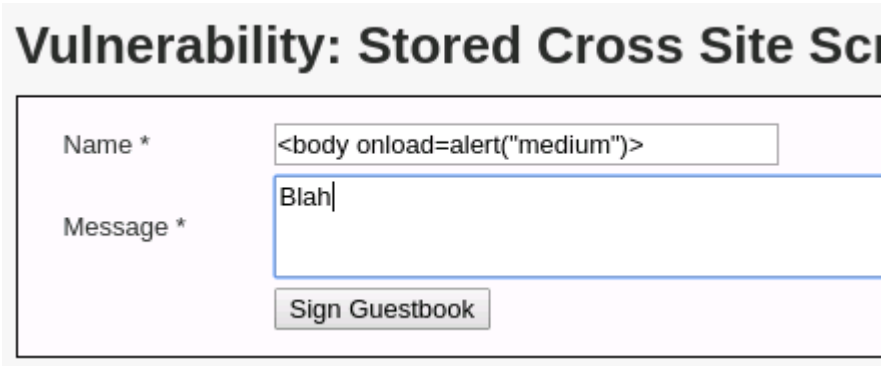
- I right-clicked on the Name submission box and clicked on Inspect.



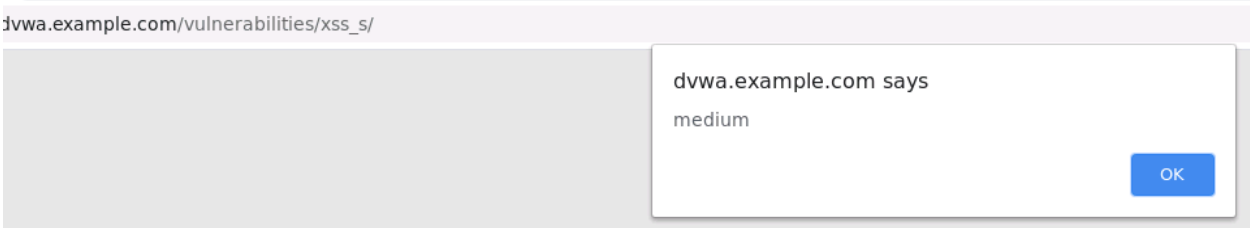

```
▼<tr>
    <td width="100">Name *</td>
  ▼<td>
      <input name="txtName" type="text" size="30" maxlength="10">
    </td>
  </tr>
```

I took note of the **maxlength="10"**. Double-clicked on the 10 and changed it to 100. Since the check for this is on the client side and not the server side I can bypass the limitation. I tried to exploit again.

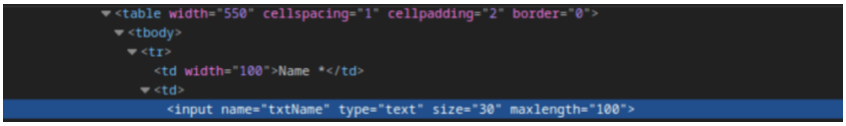
- In the name submission box, I typed **`<body onload=alert("medium")>`**, and in the message submission box, I type any message liked. Finally, I press the **Sign Guestbook** button.



Once again, we can test to see if the XSS is stored by navigating to any other page, then back to the XSS (Stored) page.



Question 2: Do a research online, find a second way, other than the <body> tag given in the example, that can also get the script executed. Explain what you have find, show that you have successfully override the length limit, and show that you have successfully have the script execute and the "medium" message popped out (you need to provide both verbal explanation and screenshots). Hint: you can get some inspiration from here https://owasp.org/www-community/xss-filter-evasion-cheatsheet



The <IMG SRC= onmouseover=" alert('medium')"> script is a concise yet potent payload for exploiting Stored Cross-Site Scripting (XSS) vulnerabilities. By injecting this script into a vulnerable input field on the DVWA Stored XSS page, attackers can store it on the server. When other users access the page, the script triggers a JavaScript alert() function with the message "medium" whenever their mouse pointer hovers over the invisible image.