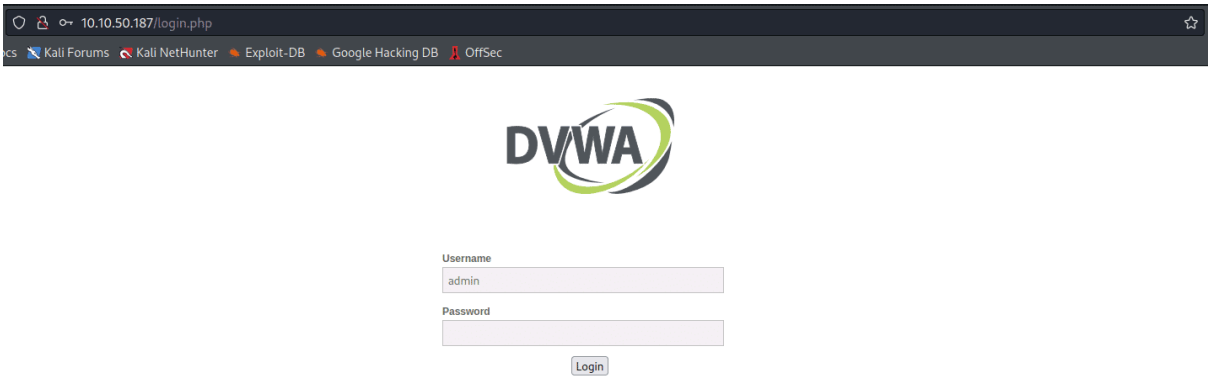
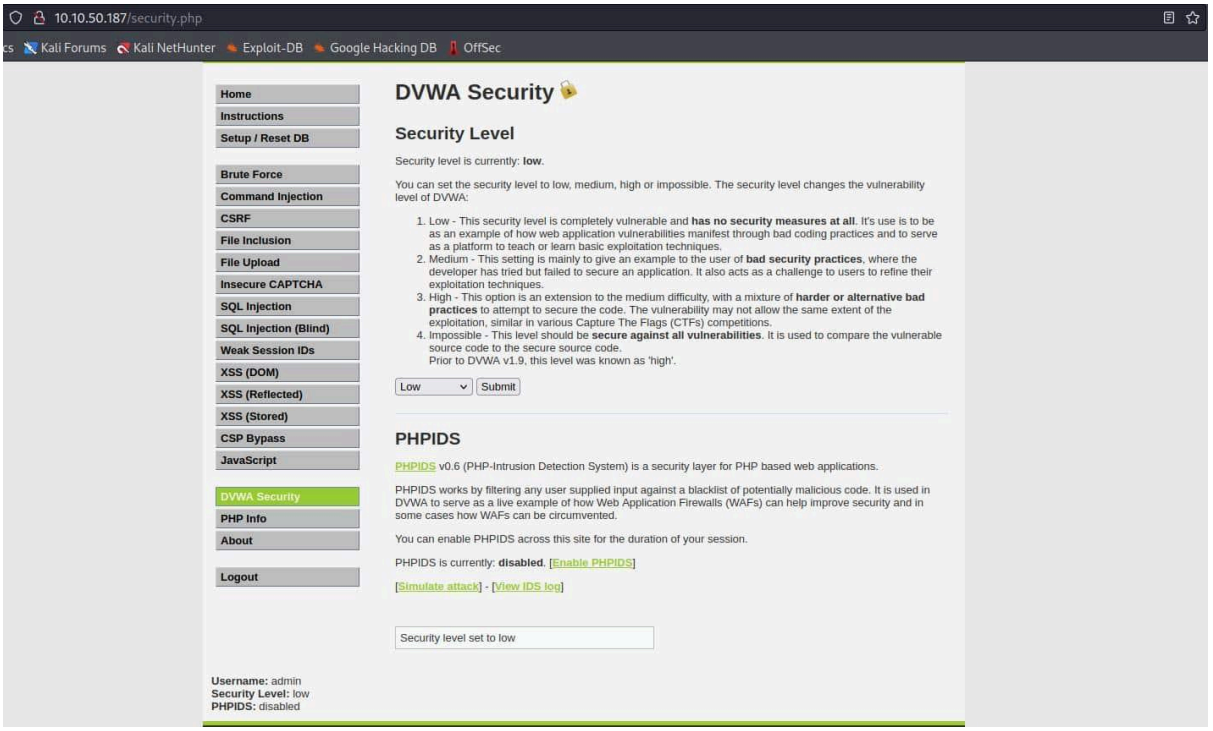
Week6: Perform various types of XS attacks on vulnerable application us

Stored XSS on DVWA with low security

Login into DVWA, Username: admin, Password: password



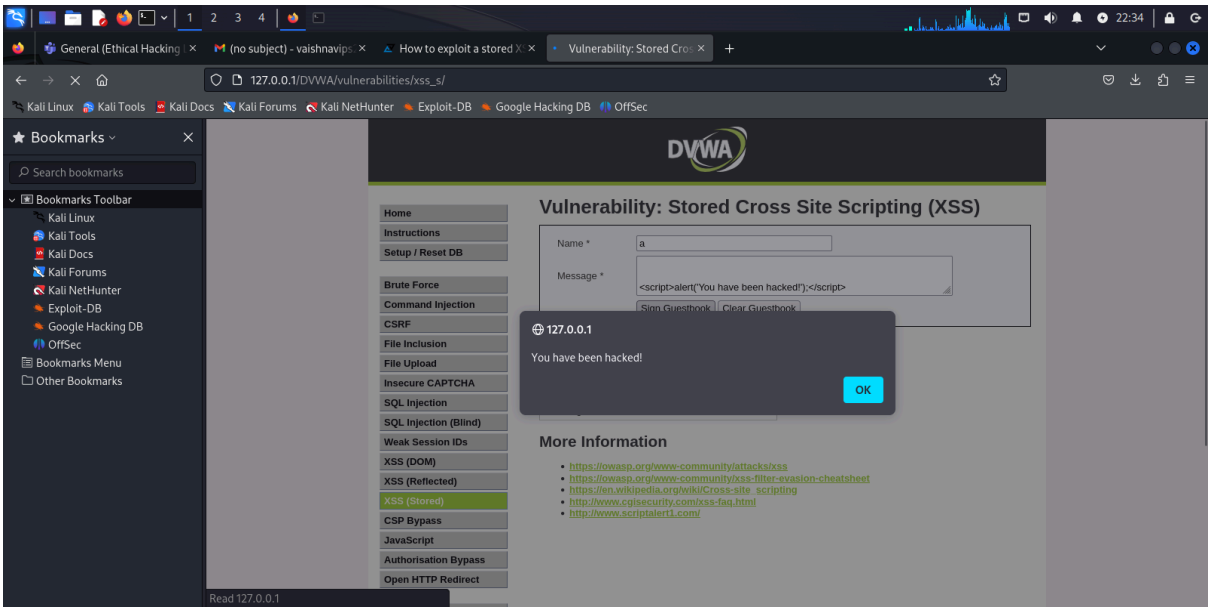
The security level is set by default as impossible, so change it to low from the settings on the left sidebar:



Click on XSS(Stored) on the left sidebar and run the following snippet.

```
<script>alert('You have been hacked!');</script>
```

We see below popup



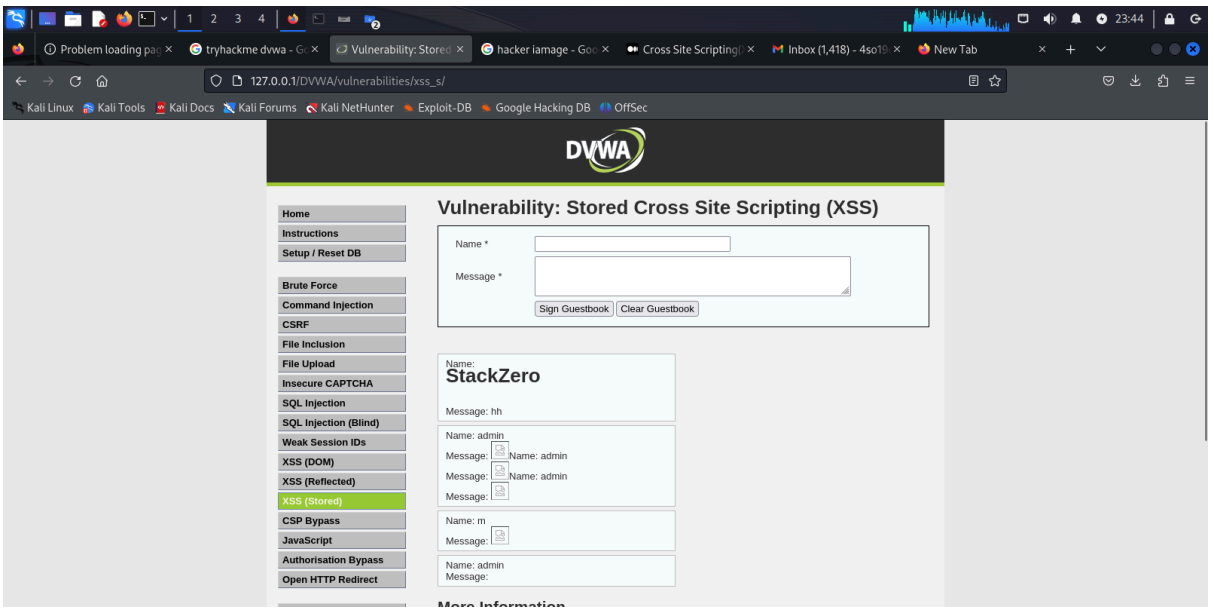
When we click on the button, a pop up displaying the message “You have been hacked” appears on the screen.

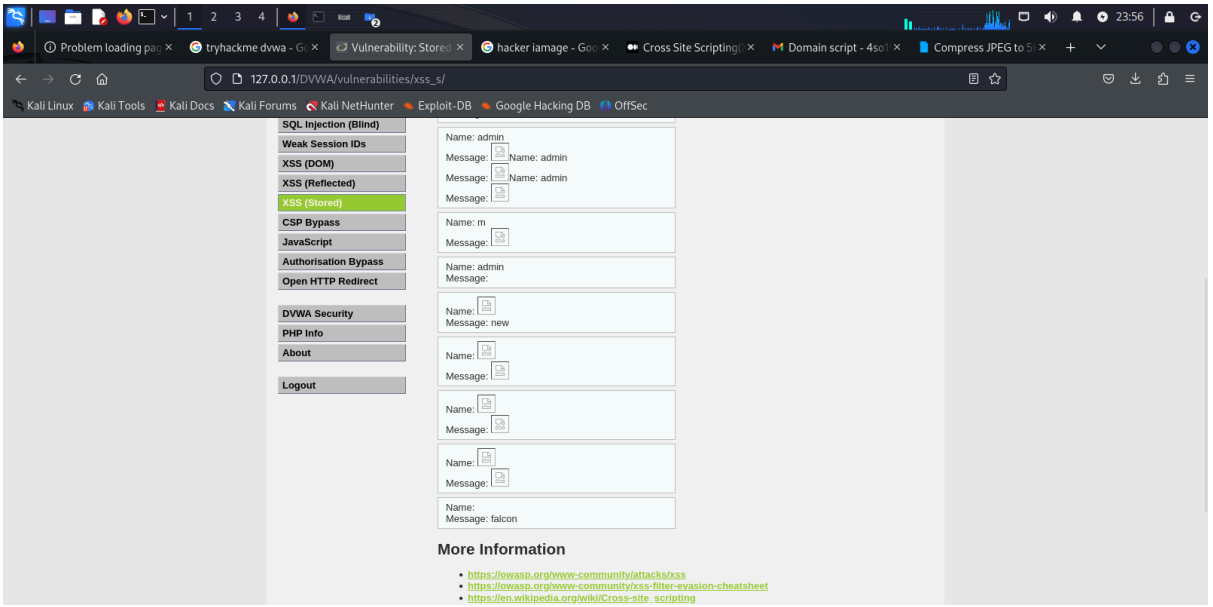
Exploiting the XSS Vulnerability: When the form is submitted, the web application processes the input provided by the user without proper validation or sanitization. As a result, the malicious JavaScript code injected into the "Message" field is executed within the context of the web page.

Displaying the Alert Box: As a consequence of the injected script executing, a pop-up alert box appears on the screen with the message "You have been hacked!", indicating successful exploitation of the XSS vulnerability.

Stored XSS on DVWA with high security.

- Increase the maxlength attribute of the name
- Type <h1>StackZero</h1>
- Press Sign Guestbook button





<u>Difference between reflected XSS and stored XSS:</u>

Reflected XSS: In reflected XSS, the malicious script is reflected back to the user immediately upon submission of a specially crafted input, typically through a URL parameter or form submission. The script is not stored on the server but is instead included in the response dynamically. When the user interacts with the compromised URL or form, the script executes within the context of their browser session. However, if the page is reloaded or the session ends, the malicious script is no longer active.

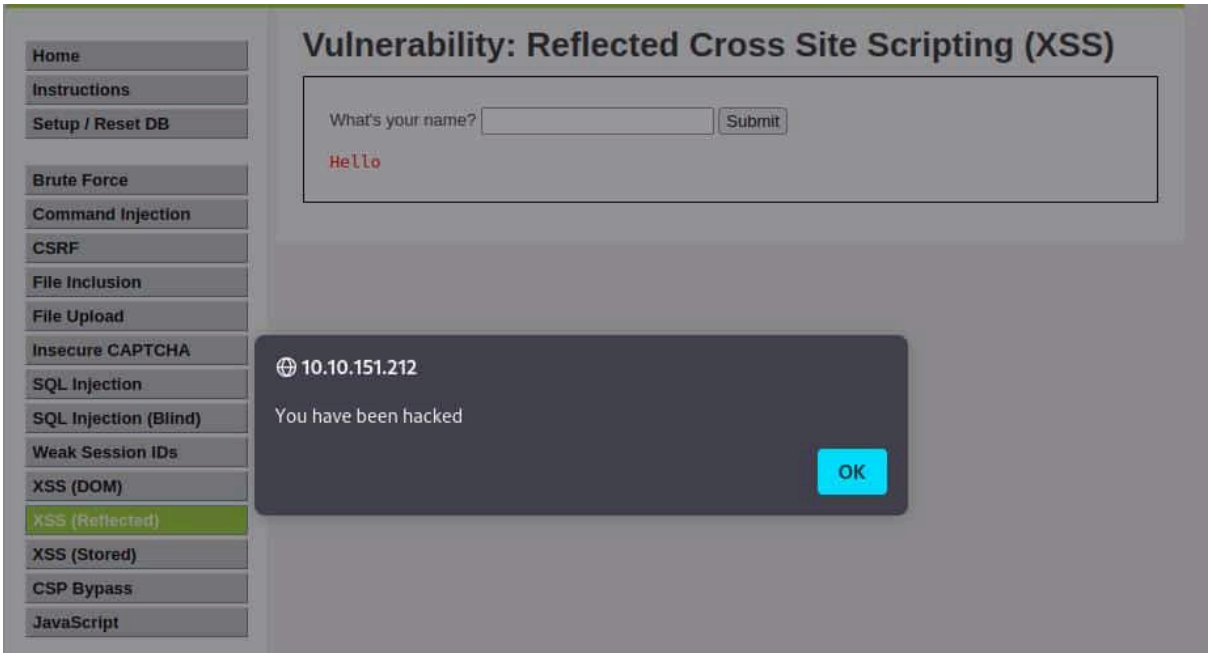
Stored XSS: In stored XSS, the malicious script is permanently stored on the server, often within a database or a file, as part of the application's content. This occurs when the application accepts user input and stores it without proper sanitization or validation. As a result, the malicious script becomes part of the application's data and can affect any user who accesses the compromised content. Even if the page is reloaded or the session ends, the stored script persists, and every time a user accesses the affected page, the script executes.

Performing Reflected XSS with medium security:

Change to medium level security and select the XSS tab.

With medium security, the above statement does not work as it removes statements with <script> tag.

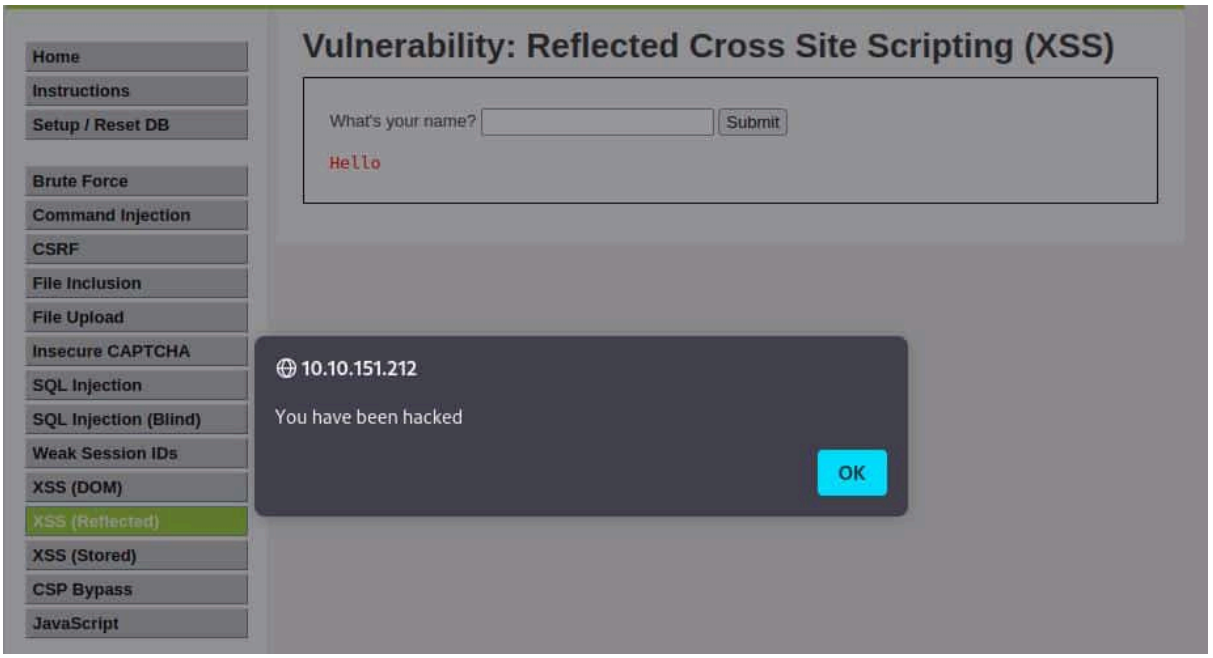
As case sensitivity can be taken an advantage of, the following with different types can be used to avoid pattern matching and elimination:
<sCRiPt> alert("You have been hacked!"); </script>



Performing Reflected XSS with high security:
Change to high level security and switch to XSS tab.

Here, an increased level of security is used so case sensitivity is included. So we cannot use statements such as the ones used in medium security:
<sCRiPt> alert("You have been hacked!"); </script>

The <script> tag is being ignored all together no matter in which pattern it is in. So, using different tags like <img> tag will be successful:
<img src=X onerror="alert('you have been hacked!')">



Stored XSS:

Cross-site scripting that is stored (XSS) is the web server and target website are both stored with malicious code by the hacker. Cross-site scripting, or XSS, occurs when an attacker inserts malicious JavaScript into a website and that script executes on the machines of other visitors.

With low security, in the name field, add a username and in the message field type:
<img src=x onerror=alert(document.domain)>

And specify the source of the image, such as the link to the image from some webpage, in the src component of the <img> tag.
The image will be displayed when the sign guestbook button is pressed.