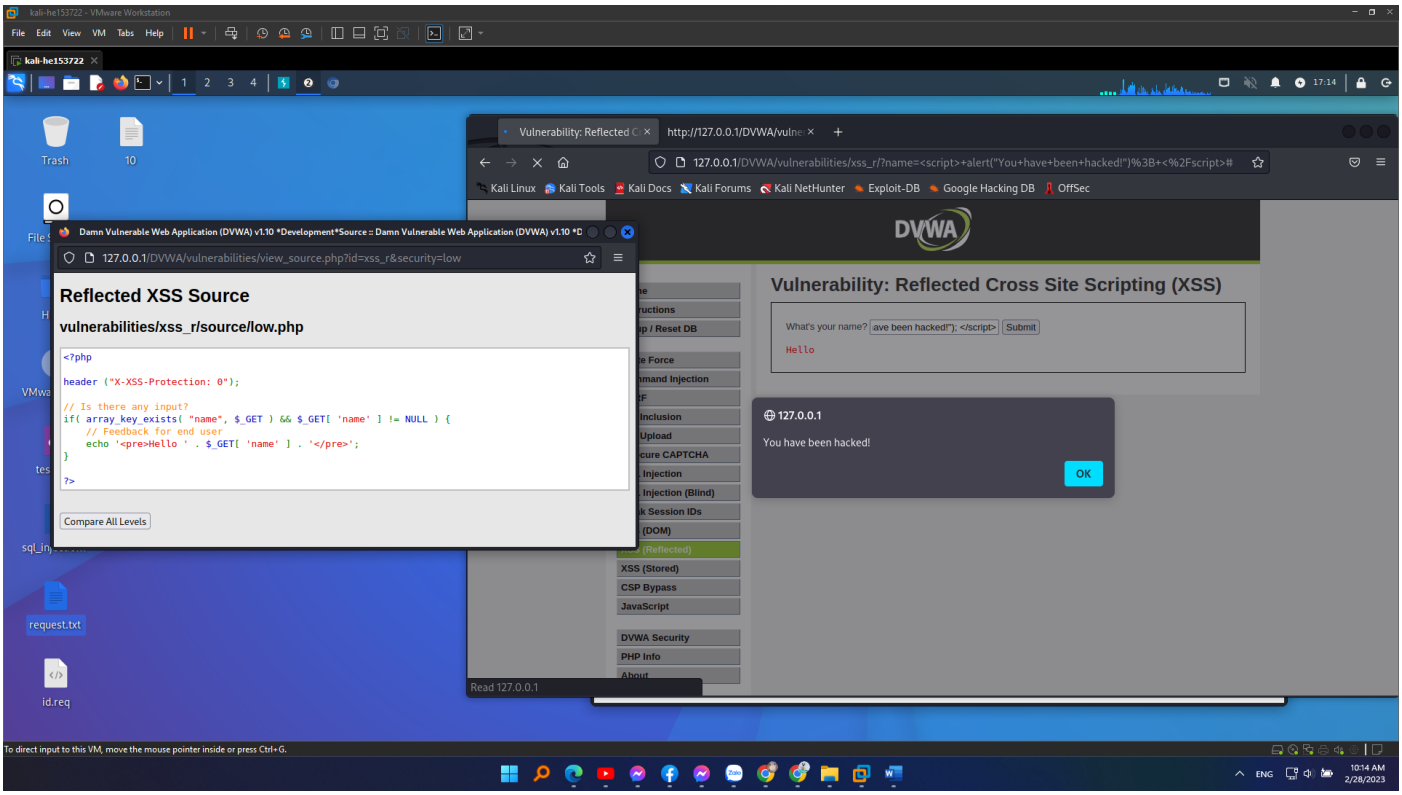
Lab 14: Reflected XSS

Reflected XSS (Cross-site scripting) is a type of security vulnerability that occurs when a web application does not properly sanitize or validate user input before displaying it back to the user in the form of a response. In a Reflected XSS attack, an attacker sends a specially crafted link or script to the victim, which when clicked, sends the malicious script to the vulnerable web application. The web application then reflects the script back to the victim's browser, executing it in the context of the victim's session.

LOW

Since the input didn't pass any check I could run whatever I wanted and I ran a script that would execute an alert box saying "You have been hacked!"

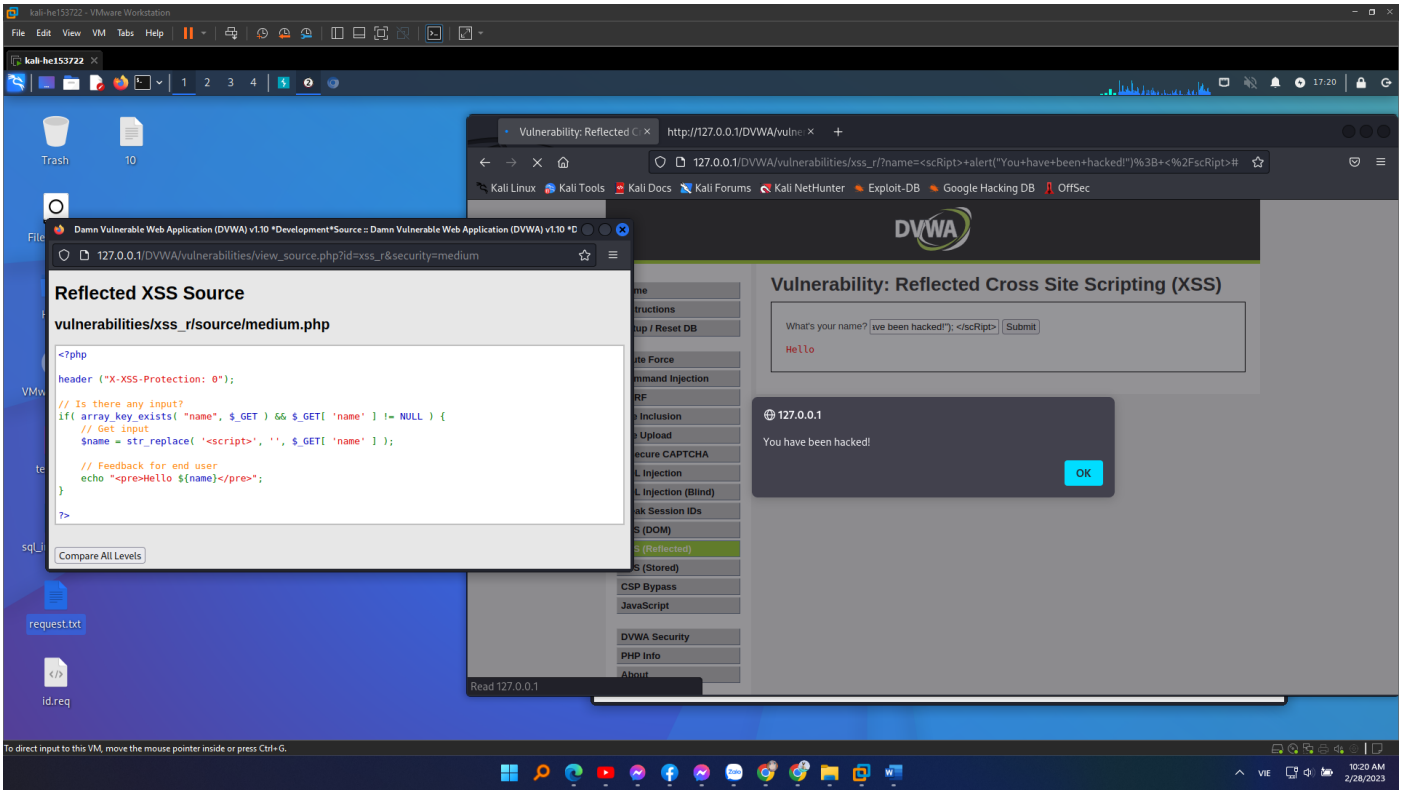
<script> alert("You have been hacked!"); </script>



MEDIUM

The code checks if the "name" parameter exists in the $_GET superglobal array and if it is not empty. If the parameter is present and not empty, the script removes the <script> tag from the input using the str_replace function, and then displays the sanitized input back to the user in the form of a preformatted text using the echo statement. Because browser doesn't care if a letter is uppercase or lowercase. On the contrary, PHP functions are mostly case sensitive, so a nice turnaround can be to alternate randomly upper and lower case characters

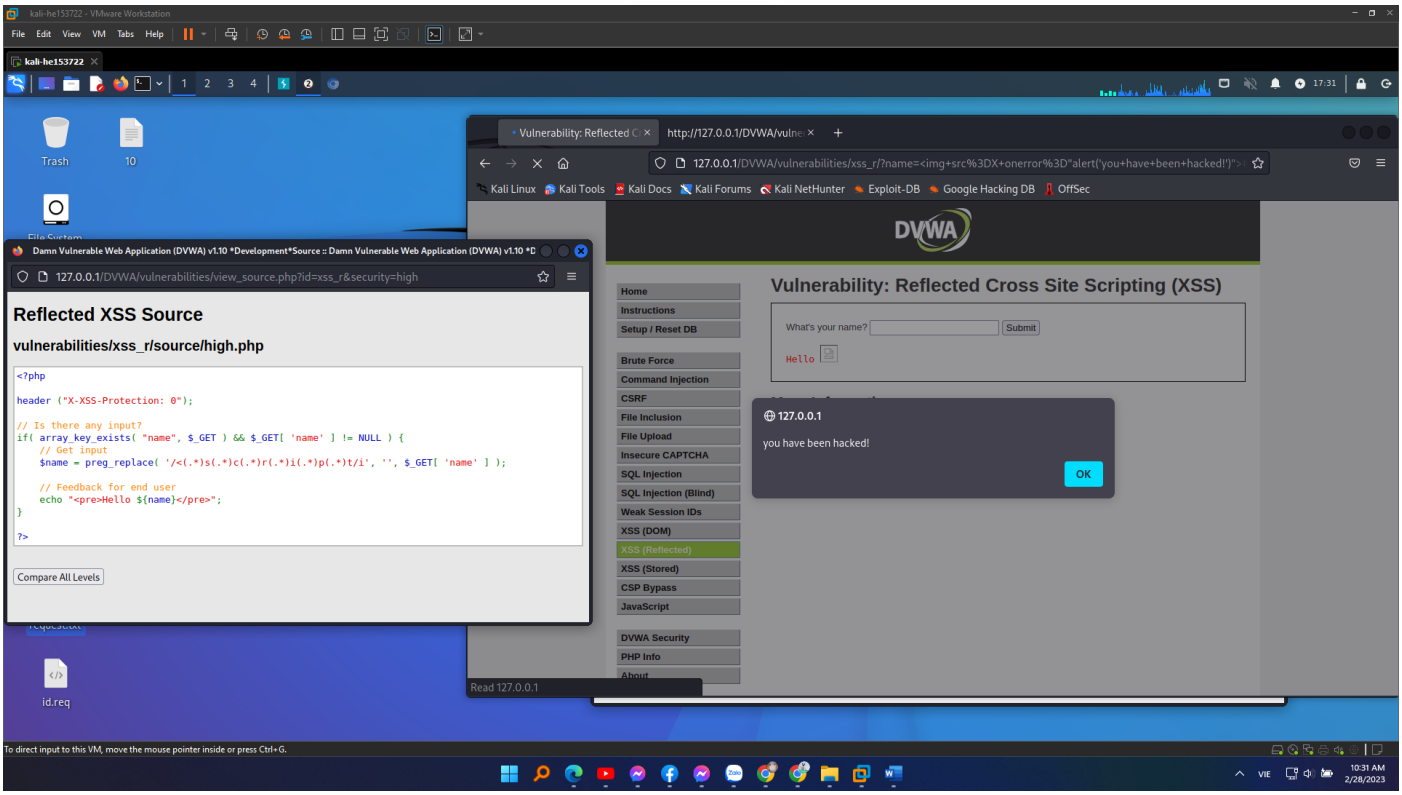
<scRipt> alert("You have been hacked!"); </scRipt>



HIGH

The code checks if the "name" parameter exists in the $_GET superglobal array and if it is not empty. If the parameter is present and not empty, the script removes the <script> tag from the input using a regular expression pattern that matches any variation of the word "script" with characters inserted in between each letter. The sanitized input is then displayed back to the user in the form of a preformatted text using the echo statement. Since the filter only filters <script> tags I used another HTML tag like <img>

<img src=X onerror="alert('you have been hacked!')">



Explain why we shouldn’t use “alert(1)” when checking for XSS in web applications?

"alert(1)" is a very basic example of an XSS payload and does not accurately represent the types of attacks that real-world attackers might use. Real-world attackers can use much more complex payloads that can bypass input filters and other security measures.

Limited scope: Injecting "alert(1)" only tests if the input is vulnerable to a certain type of XSS attack, specifically, the "stored XSS" attack where the malicious script is stored on the server and then executed on every page load. Other types of XSS attacks, such as "reflected XSS" or "DOM-based XSS," may not be detected by simply injecting "alert(1)".

Limited effectiveness: Modern browsers have built-in protection mechanisms that can block simple scripts like "alert(1)" from executing, which can lead to false negatives. As a result, even if the web application is vulnerable to XSS, "alert(1)" may not work as expected.

Limited value: Injecting "alert(1)" only identifies whether the web application is vulnerable to XSS, but it does not provide information on how to fix the vulnerability. To properly fix XSS vulnerabilities, it is important to understand the underlying cause of the vulnerability and identify the appropriate mitigation technique.