ASSIGNMENT 4

Team Members**: Sherin Thlakulathil Elias (014635504)**
Anagha Sethuraman (015220517)

Q1. Describe the attack you used. How did it work?

A stored XSS attack was performed for this assignment. "Stored attacks" occur when the injected script is left on the target servers for good. When the victim accesses the saved data from the server, the malicious scripts are downloaded.
This attack was carried out by entering the crypto in the name field and the <script>alert("Your system has been hacked")</script> in the message field, which brought up an alert window with the message "Your system has been hacked". Despite the fact that the URL of this page contains nothing malicious, we always receive this warning since it has been preserved in some form of permanent storage. Cookies belonging to the victim may be retrieved with this attack by using document.cookie.

Q2. Does your attack work at a "Medium" security level?

The script tag with alert did not function when the security levels were adjusted to "Medium," even though case sensitivity was checked. The message text box, however, was subject to strict enforcements that would check for any html special characters, as we discovered when we examined the source code. Comparing the Name input field to the message input text, we saw that there were less checks. The maximum length of the name field was set to 10. We were able to retrieve the cookie by using <script>alert(document.cookie)</script> and set the text box element's maximum length to 100 by inspecting the text box element.

Q3. Set the security mode to "Low" and examine the code that is vulnerable, and then set the security mode to "High" and reexamine the same code.
What changed? How do the changes prevent the attack from succeeding?

By lowering the security levels, we can observe that only the message input has been sanitized using stripslashes(), and the name input field has not been subjected to any sanity tests. The main purpose of stripslashes() is to eliminate any backslashes that may be present in the input string as well as the quotations. Since this approach does not stop the insertion of javascript commands, the security system is weak, so we were able to display an alert box when we tested the message with a <script> tag.

The message tag is sanitized using strip_tags(addslashes($message)) and htmlspecialchars() when the security settings are set to high. Here, addslashes() adds any backslashes necessary to escape the character sequence, and strip_tags() is used to remove all null characters from the output. After the input string has been tested for this, the resulting string is sent to the htmlspecialchars() function, which looks for special characters such as, >, <, etc. By using the regex ('/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', '') to look for characters that match <script>, the Name input is additionally sanitized. Due to this, the name field cannot accept any special characters, script tags, etc. as valid inputs.