CMPE 279 Assignment 4
Abhijeet Padwal 015219958

Perform a stored XSS attack against DVWA. The payload is your choice.

1. Describe the attack you used. How did it work?
   Ans:
   - In the XSS attack, I inspected the source code, there was not much cleaning done on the user inputs (just trimming and removing backslashes).
   - After that the database insert operation is performed.
   - And then those fields are displayed back on the browser.
   - The first field has an input size limit of 10 but the second field has much bigger input size.
   - We can inject payload into second field like: <script>alert(document.cookie)</script>
   - The html snippet gets stored in the database and when you switch browser window and come back to it or after page refresh the alert pops.

2. Does your attack work in "Medium" security level?
   Ans:
   - The same malicious code does not work at medium security level.
   - After inspecting the source code, we can see that in addition to trimming whitespaces, more input sanity is performed for the message field.
   - Like escaping double quotation marks and stripping the html tags (lowercase script).
   - But we can see that the name input is not sanitized.
   - So we can attack the name tag.
   - Name input tag has a max size of 10, we can just modify the page source code by inspecting the name input tag.
   - Again the lowercase script tag are also filtered so we can try using uppercase script tags.
   - We can inject payload in first field like: <SCRIPT>alert(document.cookie)</SCRIPT>

3. 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?
   Ans:
   - After inspecting the code in high security level we can see that in addition to trimming whitespaces, escaping the slashes and stripping tags some additional input sanity is performed for the name tag.
   - The name input sanity checks like completely replacing the script tags and escaping the quotation marks prevent the previous attack from successfully executing.
   - With the regular expression matching the script tags are filtered from user input.
   - For the name input, if we use anything else than script tag, it will work.
   - If we use an image html tag with invalid source, and define the onerror attribute for image tag, we can execute the javascript code on users browser window.
   - I used following injection script: