CMPE – 279
ASSIGNMENT – 4
DVWA – Damn Vulnerable Web App

GitHub Link: https://github.com/madhuridv/CMPE-279

Completed By :

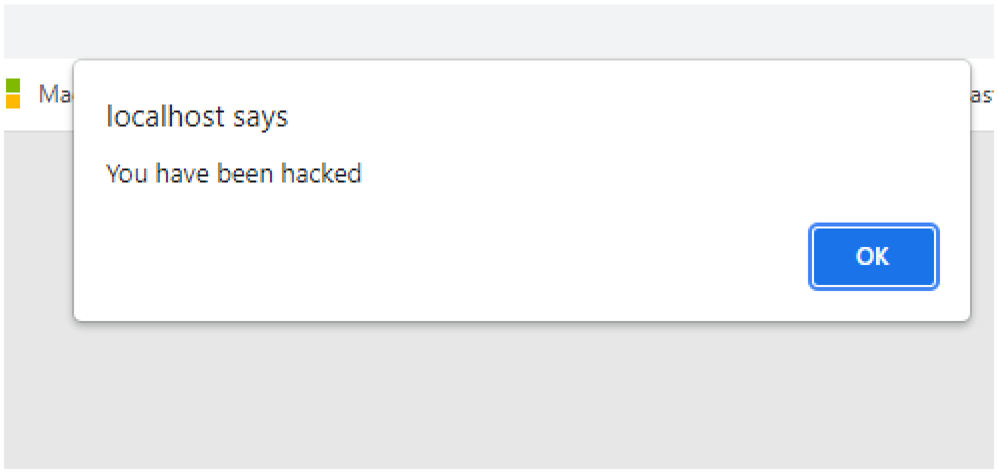
Madhuri Dhani Venu

Payal Ghule

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

Ans: For this assignment we used a Stored XSS attack. Stored attacks are those where the injected script is stored permanently on the target servers. The victim retrieves the malicious scripts from the server when it requests the stored information.

In order to perform this attack, we entered crypto in the name field and **<script>alert("You have been hacked")</script>** as a message which resulted in the below alert window:



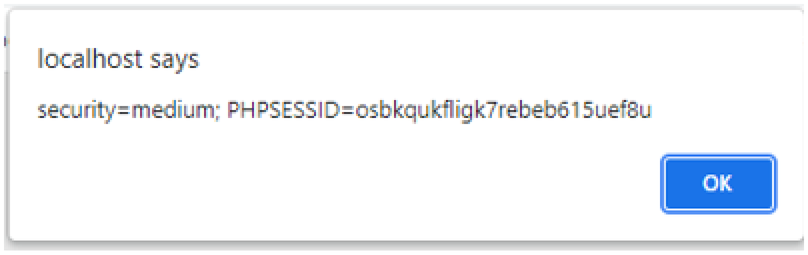
localhost says

You have been hacked

Every time we access this page even though there is nothing malicious in the URL, we see this alert as it has been saved in some permanent storage. With this attack , victim's cookies can be obtained by using **document.cookie.**

2) Does your attack work in "Medium" security level?

Ans: When we changed the security levels to "Medium" the script tag with alert did not work even if it was checked for case sensitivity. However, when we inspected the source code, we realized that there were strict enforcements on the message text box which would check for all the html special characters. We noticed that there were less checks on the Name input field when compared to the message input text. The name field was set to a max length of 10. By inspecting the text box element, we were able to change the max length to 100 and able to retrieve the cookie using **<script>alert(document.cookie)</script>**


localhost says

security=medium; PHPSESSID=osbkqukfligk7rebeb615uef8u

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: By setting the security levels to low, we can see that the only message input has been sanitized by using the stripslashes() and there are no sanity checks on the name input field. stripslashes() is mainly used to remove any back slashes if present and also the quotes in the input string. When we tested the message with a <script> tag , we were able to display an alert box as this mechanism does not prevent the insertion of javascript commands and hence the security mechanism is weak.

By setting the security levels to high, the message tag is sanitized using
strip_tags(addslashes($message)) and htmlspecialchars().

Here the addslashes() add any back slashes to escape the character sequence and the strip_tags() are applied to the output which strips all the null characters. Once this is checked on the input string, the resulting string is passed to the htmlspecialchars() which detects special characters like < , > etc.

The Name input is also sanitized by checking the regex ('/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', '') to search for characters that match <script> .

This prevents any special characters, script tags etc. to be processed as valid inputs to the name field.