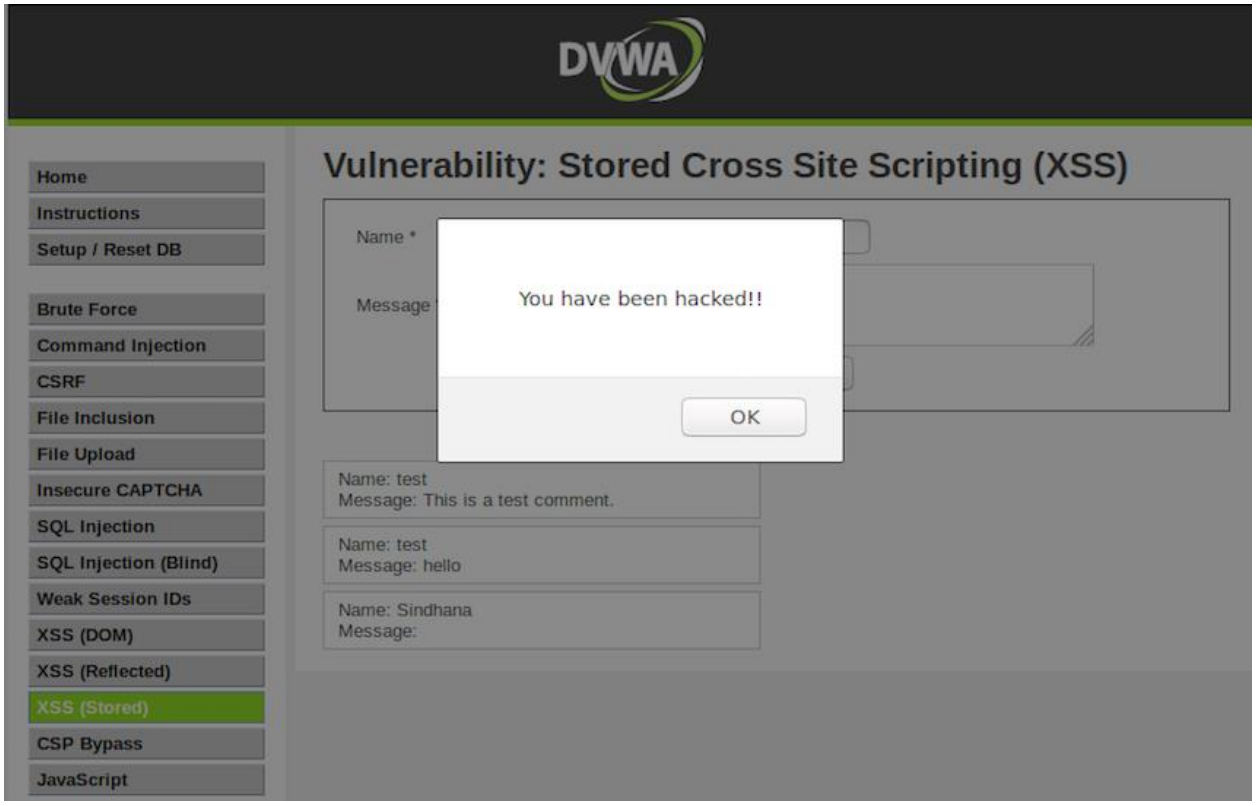
CMPE 279 – Assignment 4
DVWA – Damn Vulnerable Web App

Team Members:
Sindhana Krishnamurthy
Nikitha Kallepalli

- **Describe the attack you used. How did it work?**

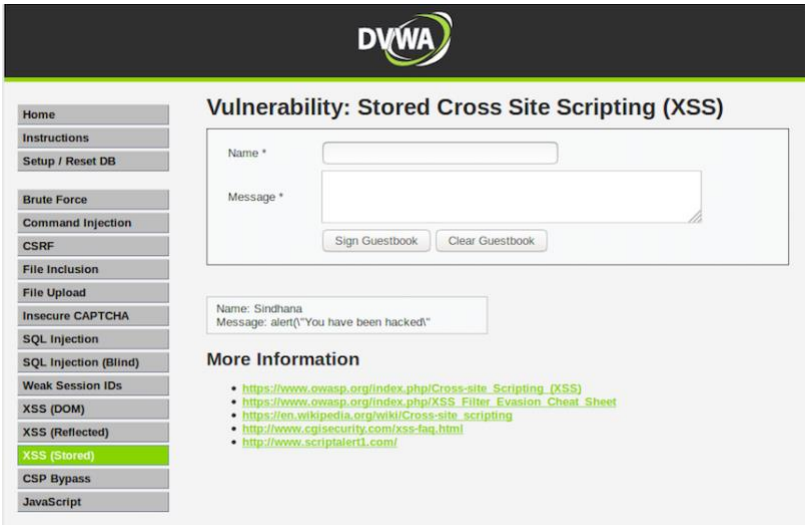
  We used the **Stored XSS attack** for this assignment. Stored XSS attacks is a form of attack when the injected script is permanently stored on the target servers. Some examples are database servers, log files, message forum, comment field etc. The victim receives the malicious script stored in the server when it requests for the stored information.

  In this case, the DVWA has two fields: Name and Message. The source code has no protection and when we enter an input like <script> alert (You have been hacked!!) </script> in the "Message" field we get a pop-up alert box as shown below:

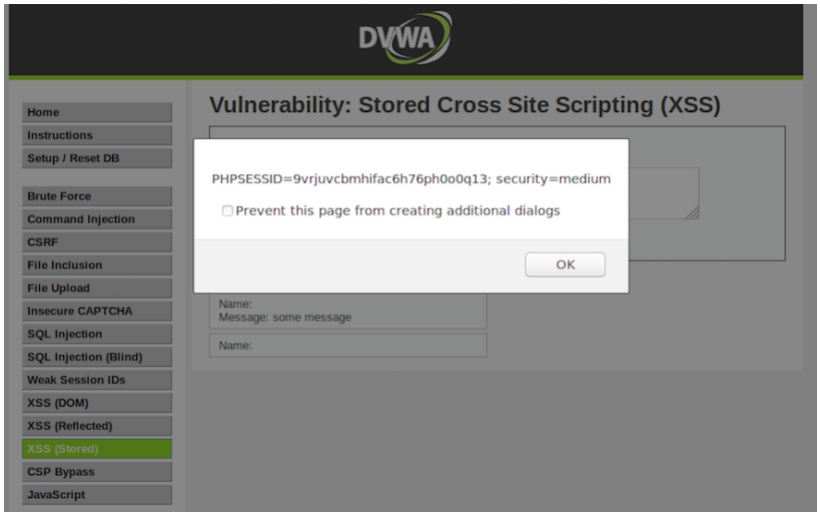


- **Does your attack work in “Medium” security level?**

  No, the attack using the same input in the Message field does not work when we set the security level to “Medium” as the developer added some additional protection such as strip_tags() along with addslashes() function to the message field along with htmlspecialchars() function. We get the following output when we use the same attack.



  But the name tag didn’t have the same level of protection. So, we used “Inspect Element” to change the length of the “Name” <input> tag to 100 and gave an input like <scr<script>ipt> alert(document.cookie)</script> and we are able to successfully hack the application which is shown in the output below.



- **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?**

  When we set the DVWA security to “**Low**” we noticed that the only protection incorporated is the stripslashes() function for the “Message” input field. Also, the “Name” input field has no protective measures added to it. Therefore, when we gave an input message with the <script> tag and alert() function, we were able to inject the JavaScript command and display an alert message saying “You have been hacked!!” to the user.

  When the DVWA security is set to “**High**”, we noticed that the “Message” input has been sanitized using the strip_tags(addslashes($message)) function along with the htmlspecialchars() function. The addslashes() function adds backslashes to escape character sequence whose output is then passed to the strip_tags() function which strips all null characters. The resulting string then goes through the htmlspecialchars() function that detects html special characters. We also noticed that the “Name” input field is also protected by checking for the pattern (‘/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/I’,”) to search for malicious input using the <script> tag. These security measures check for any malicious code containing special characters, <script> tags etc. to be injected thereby preventing our attack from succeeding.