QUESTIONS

1. I would do something similar to what is already done. The registered name would be set as a key variable and duplicates would be checked through this.

2. One step further, I would implement a hash system to hash the password and check for duplicate password usage on the same localhost system.

3. The comment was answered, screenshots are at the bottom of the report.

4. This script steals the PHP session cookie data and places it in a file

5. The previous comment was displayed, however the new comment with <script> could not be rendered. Screenshot is at the bottom of the report.

    a. Cookie Retrieval Explanation:
        i. PHPSESSID, This is the session ID of the current PHP webpage.
        ii. Etnewman=Nickname, This is the key field for the Nickname.
        iii. Time: March 22 2021, 01:54:52 PM UTC, This is the timestamp of the cookie created.
        iv. Hashed Password: , This is the hashed password data of the user.

6. This script adds an alert that pops up as a message box that states, “The fun begins now”. It also steals cookies and logs them exactly the same as question 4.

7. Yes, a pop-up message occurs. Screenshot is at the bottom of the report.

8. No, the comment is not rendered and displayed. Screenshot is at the bottom of the report.

9. It can execute any action within the <script> parameters; Yes, it can execute Javascript code; No, I do not think users of the web portal will like it.

10. Yes, it limits what can written within <script> and restricts more complex attacks.

11. No, my comment does not render as the PHP code I have inserted does not allow for special characters. I included a JS alert stating that the user cannot use special characters.

12. Post fix, I would allow anonymous comments as they cannot attack my page now.

13. No, because this method is not secure. Passwords should always be stored securely using hashes.

14. In this particular example, I used a simple way to protect my comment box from an XSS attack, by not allowing special characters inside of it. This method is known as 'Escaping', and by escaping the user input, we filter the data our webpage can receive. We can use this method to escape any and all HTML, URL, and Javascript entities.

    There are many alternative ways to protect a webpage from an XSS attack, however. Another way is validating input. This is the process of ensuring an application si rendering the correct data and preventing malicious data from doing harm to the site, database, and users. This is common in SQL injection; however, it can also be a method for XSS attacks.

    Another way is by Sanitization. By sanitizing user input, we significantly reduce the risk involved in operating a website. Sanitization is especially helpful on sites that allow HTML markup to ensure data received can do no harm to users as well as your database by scrubbing the data clean of potentially harmful markup, changing unacceptable user input into an acceptable format.

    In conclusion, to prevent XSS attacks we will usually have to use all three of these methods at the same time to prevent any serious damage from occurring. These methods work well when utilized together.

15. Yes, there is a DBMS in use. It is mySQL, but I do not know the name of the database nor what version it is. I believe it to be posted in the config.php, however I don't have access to it.

16. XSS.PHP = This is the file that handles the website for posting comments. It stores cookies here for login/logout purposes. It contains a DBMS for storing old comments.

    REGISTER.PHP = This file handles the registering of a user. It stores it in a DBMS for login/logout purposes as well as cookie session values. It can figure out if a user is unique or not.

    COOKIERETRIEVAL.PHP = This is a simple HTML file that display the cookies that were stolen using cookieStealer.php

    LOGIN.PHP = Handles the login of a user. It checks if a user is already logged in, and if not, logs them in. It also handles user input and checks whether a user exists or not. This also handles the login page itself as a HTML file.

    LOGOUT.PHP = Handles the logout of a user. It deletes the stored cookies of a user and destroys the session. It then redirects to the login page.

LITERATURE REVIEW

A) CVE Reports:

1) CVE-2021-26722

I) LinkedIn Oncall through 1.4.0 allows reflected XSS via /query because of mishandling of the “No results found for” message in the search bar.

II) Date: 02/05/2021

III) This attack was done on LinkedIn’s Oncall website. “By clicking the search bar, a search will be done to the search API endpoint. Because nothing can be found a No results found for "<img src="x" onerror="alert(document.domain)">" message will be shown. Because this message includes the search query and lacks the proper HTML output encoding, the query is interpreted as HTML/JS and an alert containing the document.domain is shown.” This could let the attacker fully compromise a user, from stealing user credentials to viewing and modifying all of the information that user is able to view or modify.

IV) References: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-26722, https://github.com/linkedin/oncall/issues/341

2) CVE-2021-1351

I) A vulnerability in the web-based interface of Cisco Webex Meetings could allow an unauthenticated, remote attacker to conduct a cross-site scripting (XSS) attack against a user of the web-based interface of the affected service. The vulnerability is due to insufficient validation of user-supplied input by the web-based interface of the affected service. An attacker could exploit this vulnerability by persuading a user of the interface to click a maliciously crafted link. A successful exploit could allow the attacker to execute arbitrary script code in the context of the affected interface or access sensitive, browser-based information.

II) Date: 11/13/2020

III) This vulnerability only affects Cisco Webex Meetings. Due to the size of these Webex meetings in today’s environment, this vulnerability had the potential to cause

a massive amount of damage if not caught by Nguyen Anh Tien of Sun Asterisk Cyber Security Research Team.

IV) References: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-1351, https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-webex-xss-Lz6HbGCt

B) A Study on XSS Attacks: Intelligent Detection Methods

1) Citation: V S Stency and N Mohanasundaram 2021 J. Phys.: Conf. Ser. 1767 012047

2) This article depicts the image of Cross-site scripting and how it is traditionally done. In it, Stency and Mohanasundaram go in detail the various different detection methods and how powerful they are in their own ways. They cite many other researchers and their work done towards detecting XSS attacks.

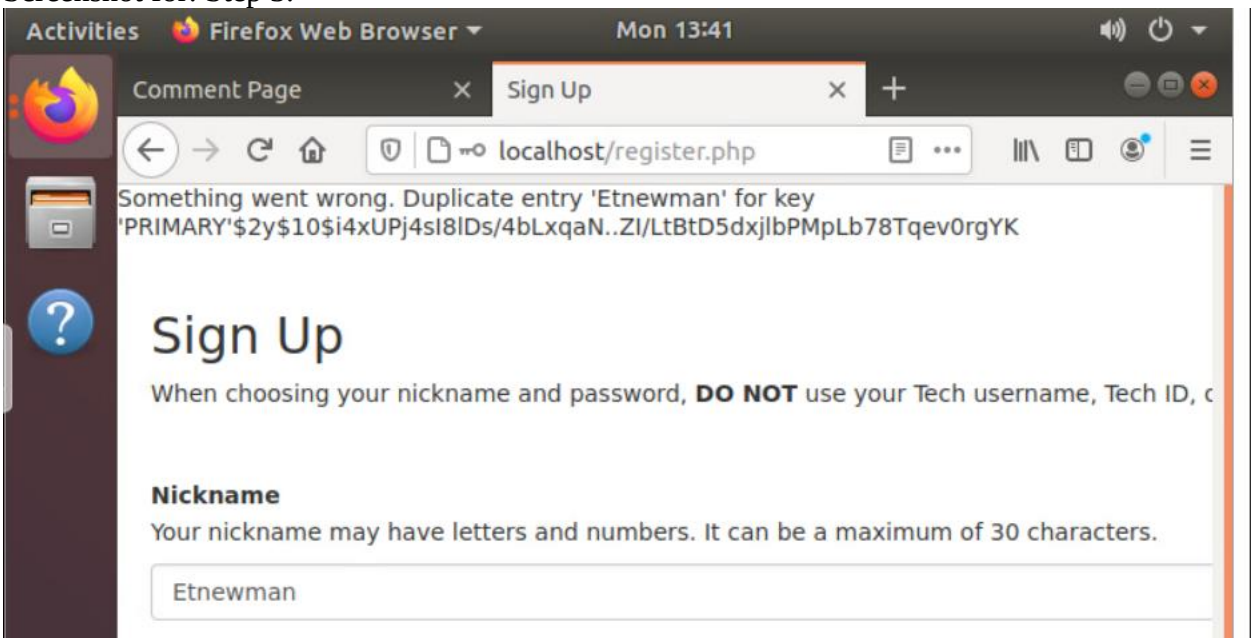
   After the introduction, it talks about XSS attack basics. It goes into detail the different types of attacks such as persistent, non-persistent and DOM-based. It talks about the attacks in terms of precision, recall and accuracy as a performance evaluation.

   The next section talks about XSS-Smart way of detection. They use deep learning techniques called DeepXSS, in this method the XSS has been tested and trained with some features. Afterwards, they compare many different methods to each and evaluate their performance metrics and give a result.

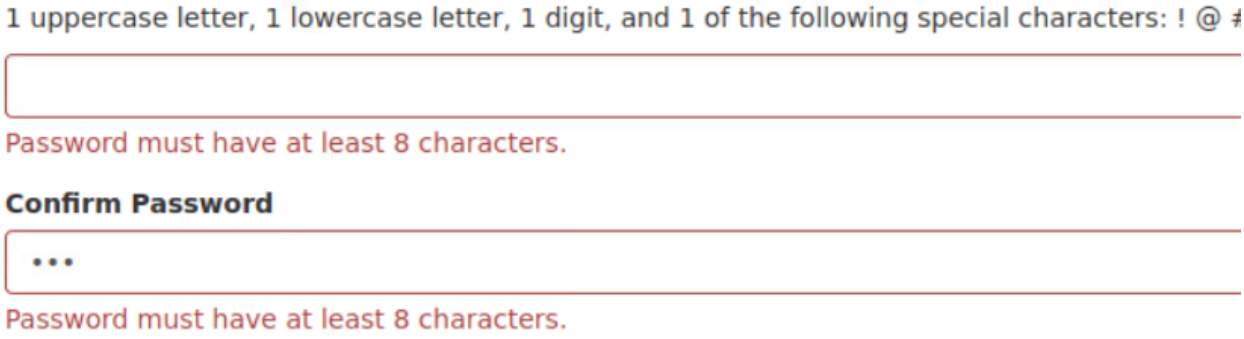
   The next section discusses the other various methods of XSS detection. These are not performed using DeepXSS, however they are evaluated on the same set of rules as the previous. It ends with a discussion concluding that DeepXSS are the superior methods of XSS detection.

SCREENSHOTS

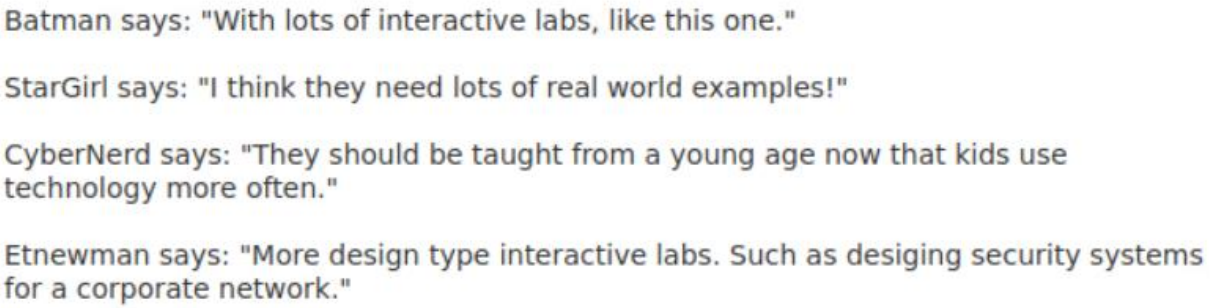
1. Screenshot for: Step 3.



2. Screenshot for: Question 1.



3. Screenshot for: Question 3.



4. Screenshot for: Question 5.

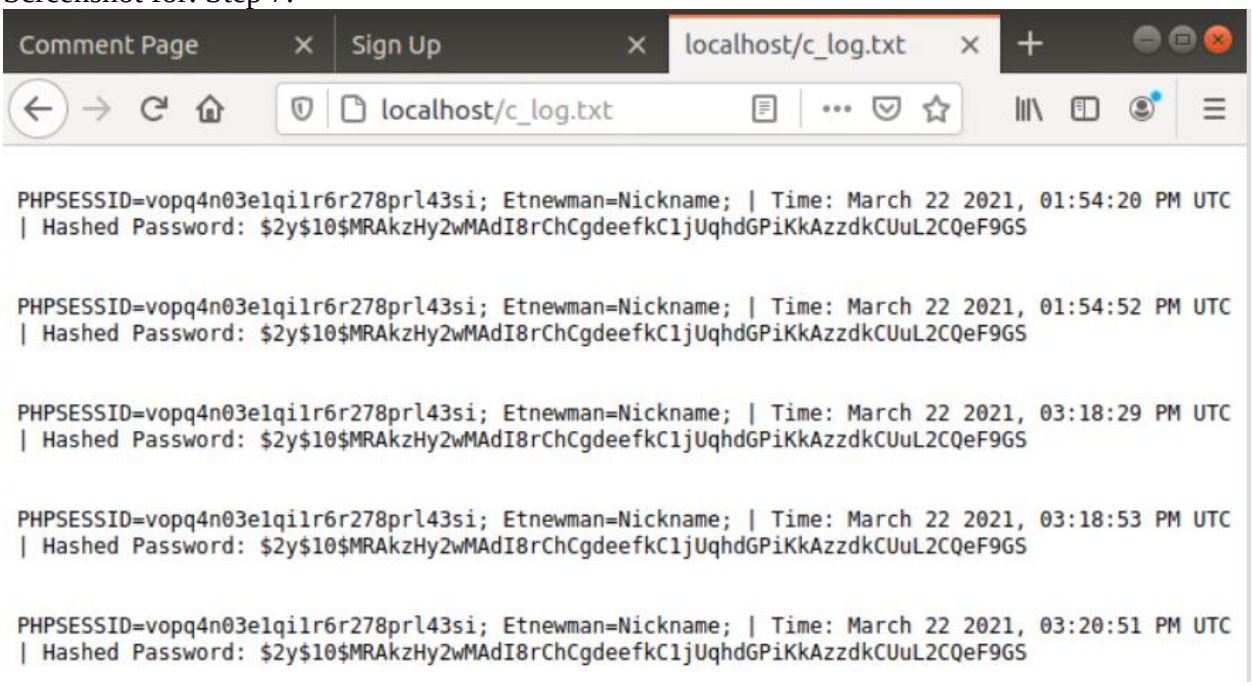
Etnewman says: "More design type interactive labs. Such as desiging security systems for a corporate network."

Etnewman's comment could not be processed and displayed.

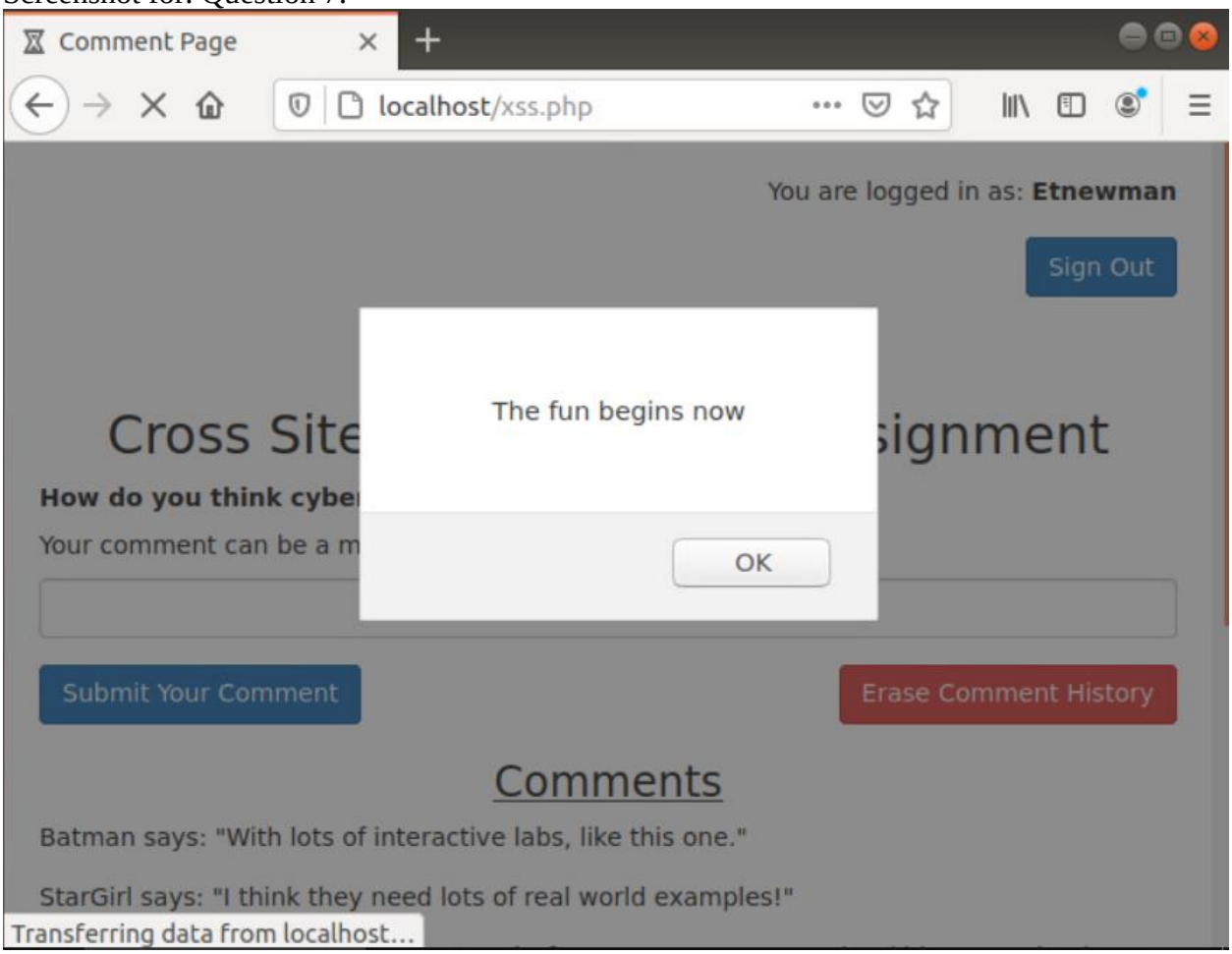
Etnewman's comment could not be processed and displayed.

Etnewman's comment could not be processed and displayed.

5. Screenshot for: Step 7.



6. Screenshot for: Question 7.



7. Screenshot for: Question 8.

Batman says: "With lots of interactive labs, like this one."

StarGirl says: "I think they need lots of real world examples!"

CyberNerd says: "They should be taught from a young age now that kids use technology more often."

Etnewman says: "More design type interactive labs. Such as desiging security systems for a corporate network."

Etnewman's comment could not be processed and displayed.

Etnewman's comment could not be processed and displayed.

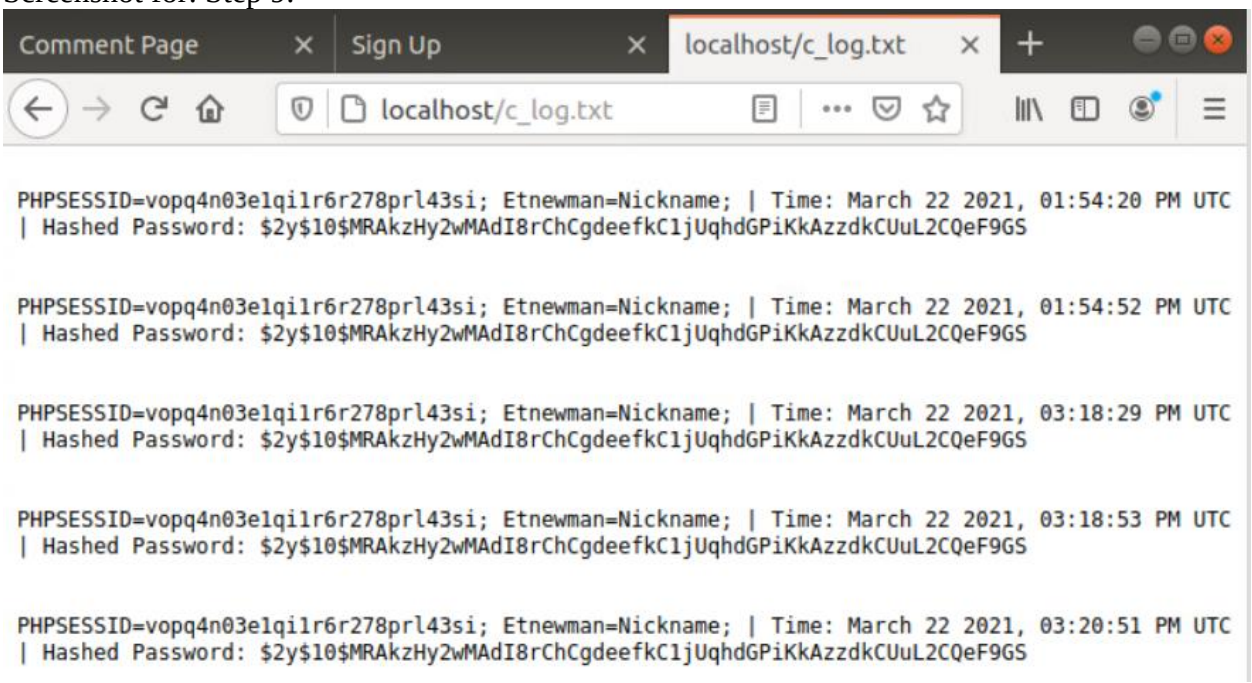
Etnewman's comment could not be processed and displayed.

Etnewman's comment could not be processed and displayed.

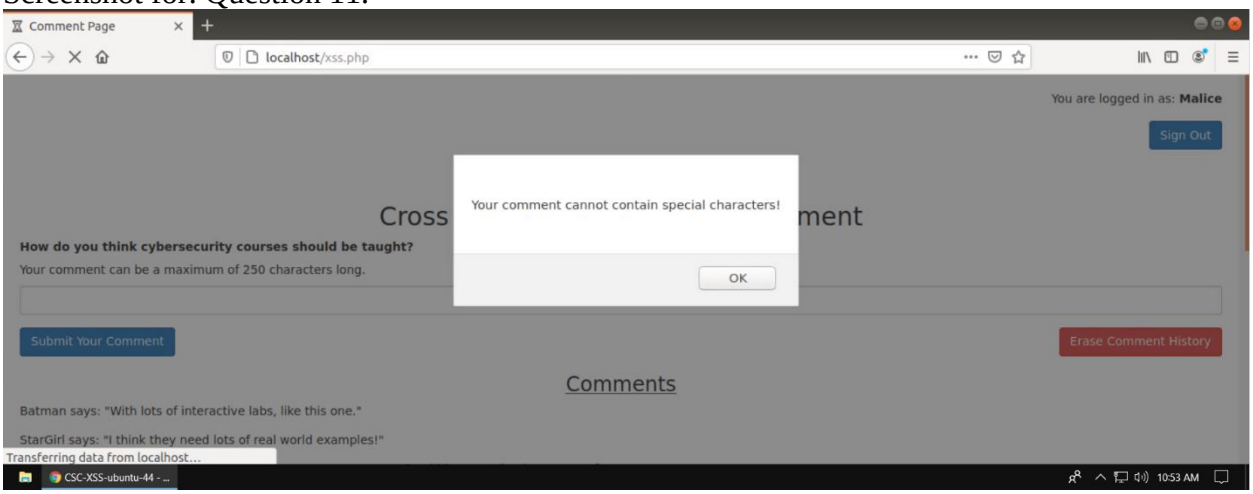
Etnewman's comment could not be processed and displayed.

Etnewman's comment could not be processed and displayed.

8. Screenshot for: Step 9.



9. Screenshot for: Question 11.



10. Screenshot for: Step 12.
    NOTE: For testing purposes, there was a time where I was testing, and the cookies were stolen under username ‘Malice’. I tested this three times but only the first time where the code was wrong is shown. The other two times cookies were NOT stolen, as shown.

```
PHPSESSID=vopq4n03elqi1r6r278prl43si; Etnewman=Nickname; | Time: March 22 2021, 03:20:51 PM UTC | Hashed Password: $2y$10$MRAkzHy2wMAdI8rChCgdeefkC1jUqhdGPiKkAzzdkCUuL2CQeF9GS

PHPSESSID=k9ndumoetjc8del0phfh2o8irg; Malice=Nickname; | Time: March 30 2021, 03:45:03 PM UTC | Hashed Password: $2y$10$nPBpYZcuxZBnz9olfpa02uA.teGn7d.VAK7UTi7aazNcbZ2nTOMWe
```