Lab 10: Cross-Site Scripting (XSS)

Thangamuthu Balaji

I. Introduction

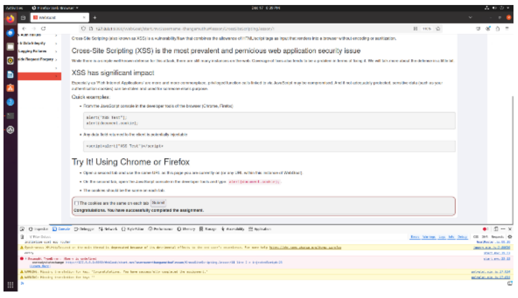
THIS is the lab report for Lab 10: Cross-Site Scripting (XSS) of the course, Cyber Defense Competitions. This lab was designed to teach students about different kinds of XSS attacks and how weaknesses in HTML can be taken advantage of through these attacks. The activities were carried out using WebGoat v2023.8.

II. WebGoat v2023.8 A3 Injection: Cross-Site Scripting

Cross-Site Scripting (XSS) is a kind of attack where harmful scripts are inserted into trusted websites, taking advantage of weaknesses in how web applications process user input. There are three main types of XSS attacks: Reflected XSS, which involves malicious input being included in immediate responses without being saved; Stored XSS, where the harmful script is kept on the server or in the user's storage and runs later; and DOM-Based XSS, which changes the browser’s Document Object Model (DOM) to run scripts without changing the server's response. In 2012, these types were further divided into Server XSS, linked to flaws in server-side code, and Client XSS, which comes from unsafe JavaScript calls that modify the DOM . To prevent cross-site scripting (XSS), it's important to use a mix of strategies that fit the specific needs of the web application. Some key steps include steering clear of HTML in user inputs by opting for alternatives like markdown or WYSIWYG editors. It's also crucial to validate inputs to ensure they match expected formats and to filter out any potentially harmful characters. After data is submitted to the server, sanitizing it is essential to eliminate any dangerous code. Improving cookie security can be done by linking cookies to certain IP addresses or restricting JavaScript access to prevent cookie theft. Furthermore, implementing Web Application Firewall (WAF) rules can help block suspicious requests, including those aimed at XSS, providing a wider range of protection against different threats.

Step 2

1) To complete this step, I needed to see if the cookies in the new tab matched those in the original one. I opened a new tab with the same URL and typed alert(document.cookie); into the console. When I did that, an alert popped up showing the cookie value, and it turned out to be the same as in the other tab.
2)
   Step 7:
   To complete this step, we needed to determine if a field was at risk for an XSS attack. According to what I learned from WebGoat, fields that include user input in their responses are usually susceptible to XSS attacks. In this situation, I clicked the purchase button and noticed the



Cross-Site Scripting (XSS) is the most prevalent and pernicious web application security issue

XSS has significant impact

Quick examples:

```
alert("XSS Test");
alert(document.cookie);
```

```
<script>alert("XSS Test")</script>
```

Try It! Using Chrome or Firefox

The cookies are the same on each tab

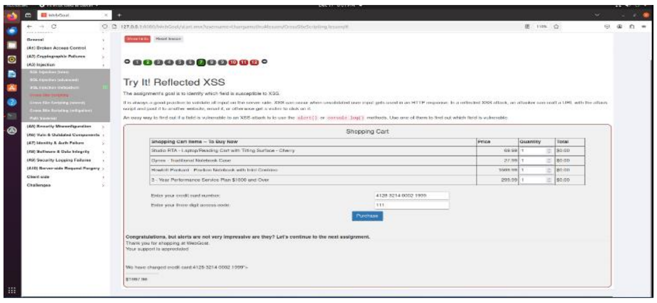
Congratulations. You have successfully completed the assignment.

Fig. 1.

The response had the credit card number in it, which is what I needed. I put my script into the input text field - ¡script¿alert(”info visible”)¡/script¿. This worked perfectly to get the job done.

3) ***Step 10:***
4) In this stage, we focused on executing a DOM-based XSS attack, aiming to locate the route for the test code in the production environment. I used the Developer tools to access the Debugger and examine the JavaScript code. While going through GoatApp.js, I discovered several test routes that I decided to try out. In the end, the route start.mvc#test was successful.

5) ***Step 11:***
6) In this task I had to use the base route that I found in the previous exercise. In the url I had to inject the script to call the phone number function. After this, I found the random number in the console and I passed The link I
accessed was http://localhost:8080/WebGoat/start.mvc#
test/<script>webgoat.customjs.phoneHome()</%2Fscript>



Try It! Reflected XSS

The assignment's goal is to identify which field is susceptible to XSS.

An easy way to find out if a field is vulnerable to an XSS attack is to use the alert() or console.log() methods. Use one of them to find out which field is vulnerable.

Shopping Cart

Shopping Cart Items -- To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry	69.99	1	$0.00
Dynex - Traditional Notebook Case	27.99	1	$0.00
Hewlett Packard - Pavilion Notebook with Intel Centrino	1599.99	1	$0.00
3 - Year Performance Service Plan $1000 and Over	299.99	1	$0.00

Enter your credit card number: 4128 3214 0002 1999

Enter your three digit access code: 111

Congratulations, but alerts are not very impressive are they? Let's continue to the next assignment.
Thank you for shopping at WebGoat.
Your support is appreciated

We have charged credit card:4128 3214 0002 1999">

$1997.96

Fig. 5. Step 7 task completed

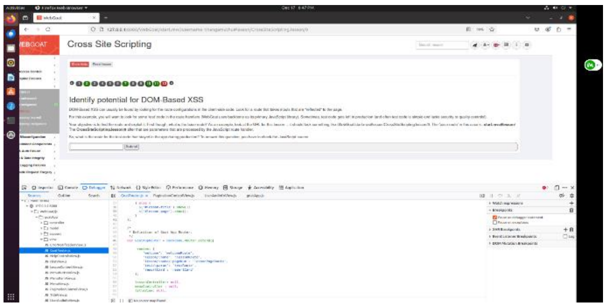


Fig. 6. Using the debugger to check JS files

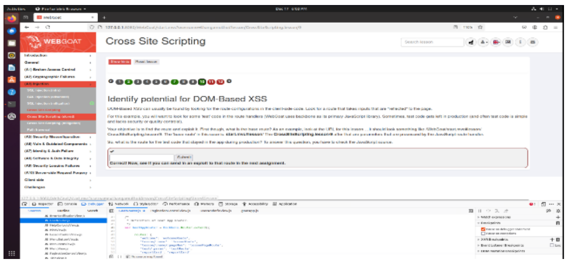


Fig. 7. Step 10 task completed

7) ***Step 12:***
The Quiz was not available at section 12 its some kind of bug .

III. Conclusion

I can conclude that I have successfully completed all lab tasks.

Fig. 8. Using the url to call the phone number function

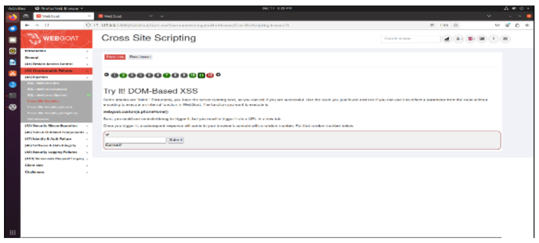
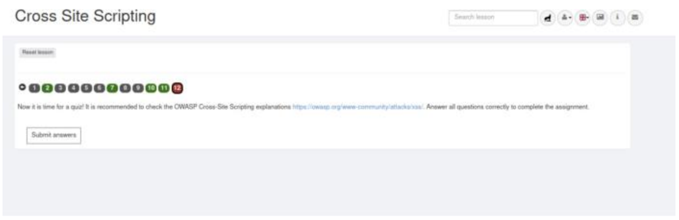


Fig. 9. Found the number using the next tab



Cross Site Scripting

Now it is time for a quiz! It is recommended to check the OWASP Cross-Site Scripting explanations https://owasp.org/www-community/attacks/xss/. Answer all questions correctly to complete the assignment.

Fig. 10. Step 12 task not visible

I gained knowledge about cross-site scripting (XSS) attacks, the different kinds that exist, and how to effectively prevent them. I discovered how harmful scripts can take advantage of weaknesses in web applications, putting user information and system security at risk. By looking at real-life examples, I learned about methods such as input validation, data cleaning, and utilizing secure JavaScript APIs to reduce these risks.

References

[1] ”Cross Site Scripting (XSS)” owasp.org https://owasp.org/www-community/attacks/xss/