Hello,

Here are the steps I followed to make my xss attack website, unfortunately, I didn’t use htmlpasta because I didn’t know if it had a php support so I implemented the xss attack on a local host with xampp.

First, here are the prerequisites to launch this website:

->Apache,

->Php

I made a very simplistic form so that i could submit the xss attacks to a php file which justs echoes out the form’s input; here is the source code for xss-attack.php

***<?php***

***$h=$\_GET["xss"]; // get the input***

***if($h){***

***echo $\_GET["xss"]; //echo it out***

***}***

***?>***

***<!DOCTYPE html>***

***<html>***

***<head>***

***<meta charset='utf-8'>***

***<meta http-equiv='X-UA-Compatible' content='IE=edge'>***

***<title>Hack me</title>***

***<meta name='viewport' content='width=device-width, initial-scale=1'>***

***<link rel='stylesheet' type='text/css' media='screen' href='main.css'>***

***</head>***

***<body>***

***<h1>XSS ME!</h1>***

***<form action="action.php" method="get">***

***<ul class="form-style-1">***

***<li><input type="text" name="xss" class="field-divided" placeholder="Hack" /></li>***

***<li>***

***<input type="submit" value="Submit" />***

***</li>***

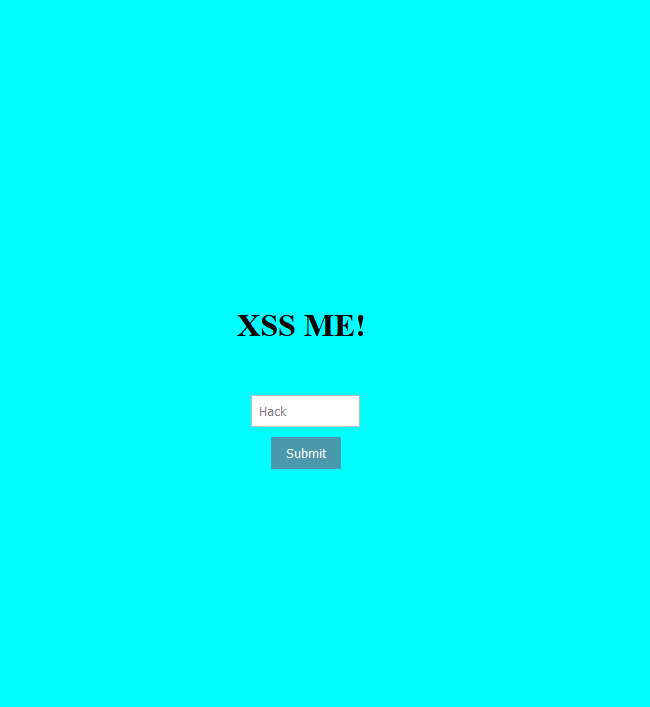
***</ul>***

***</form>***

***</body>***

***</html>***

**HERE IS THE SCREENSHOT:**



The php code I used is pretty simple because it just stupidly echoes out everything we give as input.

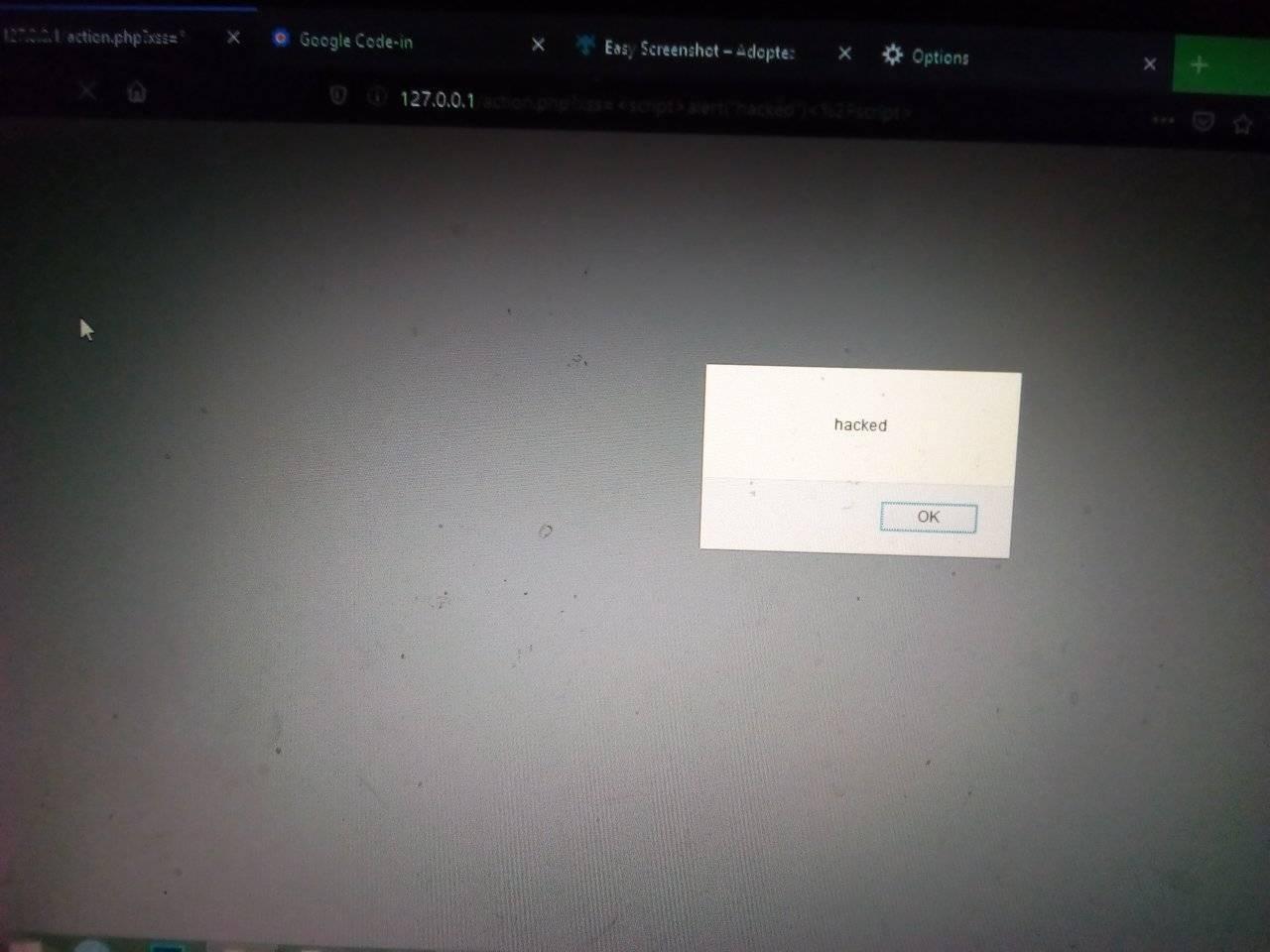
The first attack was to make the website display an alert box saying “HACKED”, I used the following JavaScript code:

***<script>***

***alert("hacked");***

***</script>***

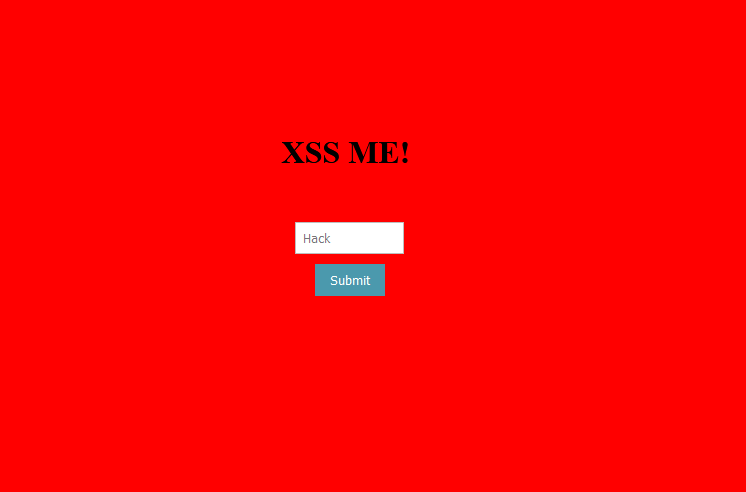
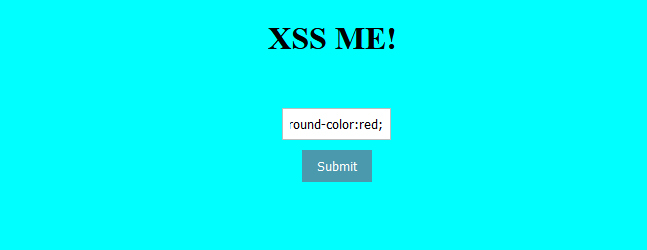
Here are the screenshots (sorry for the bad quality, Firefox doesn’t permit to take screenshots when there are alert boxes, I just used a phone):



The second attack was to change the website’s background color to red, I used this simple css code:

***<body style="background-color: red;">***

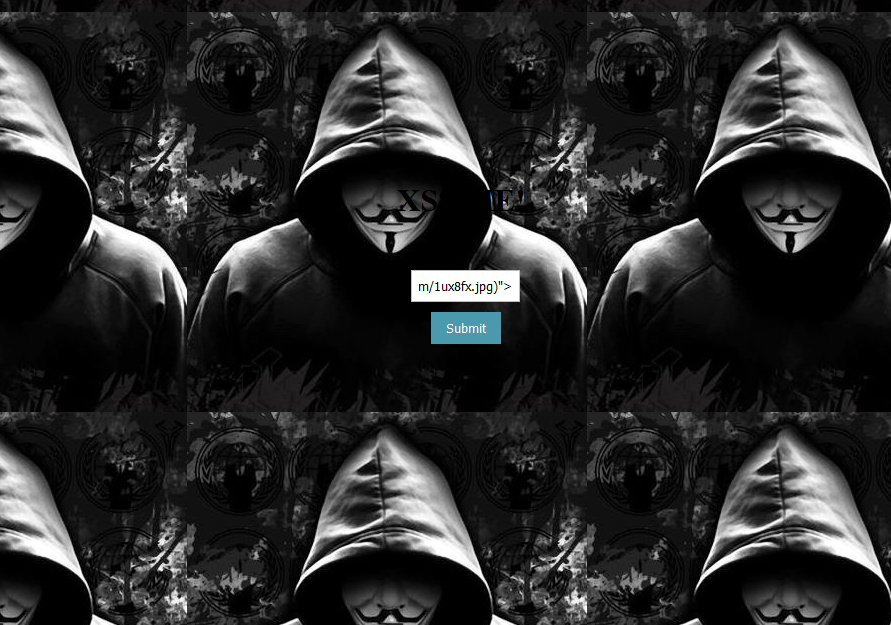
***HERE ARE THE SCREENSHOTS:***



The third attack was to change the background to an image of my choice, here is the code I used:

<body style="background-image:url(<https://i.imgflip.com/1ux8fx.jpg)>">

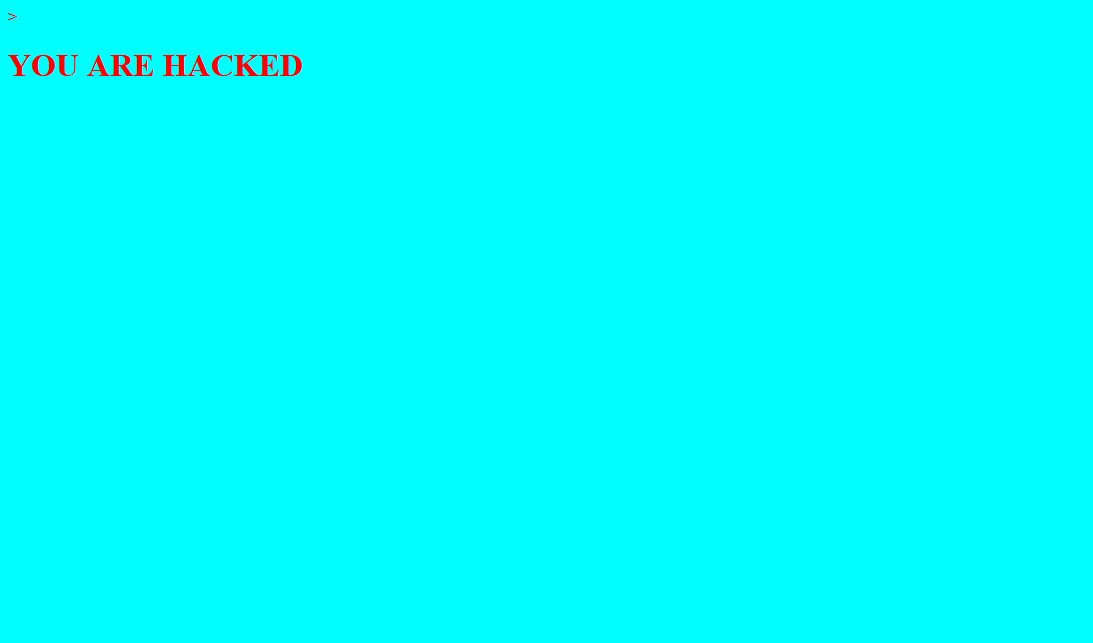
Here are the screenshots:



The fourth attack was to redirect the page to a website displaying “YOU ARE HACKED”; for this one, I first created the website with htmlpasta then I used this simple html as input to redirect the page:

<meta http-equiv="Refresh" content="0; url=https://clangorousdebater.htmlpasta.com/" />

I linke dit to the htmlpasta website which displays the said message.

Here is the sceenshot of the page(“my html knowledge ain’t that great”):

HOW CAN WE PREVENT THESE KIND OF ATTACKS:

1. The first method you can and should use to prevent XSS vulnerabilities from appearing in your applications is by ***escaping user input***. Escaping data means taking the data an application has received and ensuring it’s secure before rendering it for the end user. By escaping user input, key characters in the data received by a web page will be prevented from being interpreted in any malicious way. In essence, you’re censoring the data your web page receives in a way that will disallow the characters – especially < and > characters – from being rendered, which otherwise could cause harm to the application and/or users.

2.**Validate The input**;

Validating input is the process of ensuring an application is rendering the correct data and preventing malicious data from doing harm to the site, database, and users. While whitelisting and input validation are more commonly associated with SQL injection, they can also be used as an additional method of prevention for XSS. Whereas blacklisting, or disallowing certain, predetermined characters in user input, disallows only known bad characters, whitelisting only allows known good characters and is a better method for preventing XSS attacks as well as others.

3. **Sanitizing;**

A third way to prevent cross-site scripting attacks is to sanitize user input. Sanitizing data is a strong defense, but should not be used alone to battle XSS attacks. It’s totally possible you’ll find the need to use all three methods of prevention in working towards a more secure application. Sanitizing user input 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 to an acceptable format.

While using layers of security, such as the methods above, is a great way to helm the majority of XSS attacks, it’s important to remember that while the above prevention methods will cover most XSS attack vectors, they won’t cover everything. In order to be truly vigilant against XSS and other common, debilitating vulnerabilities, it’s important to use a mix of code review, automated testing during development and dynamic testing once the application is live, in addition, of course, to using secure coding practices that will help prevent vulnerabilities like cross-site scripting in the first place.