**SURE TRUST ORGANIZATION**

**Domain :** Cyber Security

**Trainer :** Mr Himaneesh sir

**By :**

L . Bhavani Shankar

Cross-site scripting (XSS) is a type of vulnerability that allows an attacker to inject malicious script into web pages that other users view. XSS dates back to the early days of the World Wide Web and has evolved over time as web technologies and security practices evolve.

**Here we briefly explain its history and evolution.**

A web developer began incorporating user input into her web pages without proper validation, causing the first security vulnerability.

**Late 1990s -** JavaScript was introduced, allowing developers to create more interactive and dynamic Web pages. Developers often failed to validate and sanitize user input, giving attackers an opportunity to inject malicious script.

**Early 2000s -** Discovery of XSS: Security researchers and attackers began to exploit the lack of input validation in web applications. The term "cross-site scripting" was coined to describe this type of vulnerability.

**2005 - AJAX and XSS:** The advent of AJAX (Asynchronous JavaScript and XML) has enabled more dynamic and responsive Web applications. XSS attacks are designed to take advantage of the asynchronous nature of AJAX, making them difficult to detect and prevent.

**2008 -** Improved browser security: Browser vendors such as Mozilla, Google, and Microsoft began implementing security mechanisms to prevent XSS attacks.The Content Security Policy (CSP) header was introduced to prevent XSS by defining allowed content sources.

**2010s - DOM-based XSS:** Attackers began exploiting vulnerabilities in the Document Object Model (DOM), resulting in DOM-based XSS attacks. DOM-based XSS occurs when client-side script manipulates the DOM based on user input.

**Modern - Persistent and Non-persistent XSS:** malicious input is only temporarily present in the response).

Client-side frameworks: The rise of client-side frameworks such as React, Angular, and Vue has introduced new challenges and attack vectors. Developers need to be aware of how these frameworks handle data to prevent XSS vulnerabilities.

**Security Practices:** The importance of secure coding practices, input validation, output encoding, and appropriate security headers (such as content security policies) is widely recognized to reduce XSS risks.

The evolution of XSS reflects the ongoing cat-and-mouse game between attackers and security experts. As security measures improve, attackers find new ways to exploit vulnerabilities, requiring continued efforts to improve web security.

It's important for developers to stay up to date with the latest security practices, and it's important for organizations to implement robust security measures to protect against XSS and other web-based attacks.

Learning cross-site scripting (XSS) in the IT field offers several opportunities and benefits.

**Increasing security awareness:** Understanding XSS helps IT professionals become more security aware. This allows you to identify potential vulnerabilities in your web applications and take proactive steps to protect them.

**Web Development Skills:** For a developer, it is important for him to learn XSS in order to write secure code. This will help you understand the importance of input validation, output encoding, and other security best practices when building web applications.

**Penetration Testing and Ethical Hacking:** Knowledge of XSS is valuable for professionals involved in penetration testing and ethical hacking. This allows you to identify and exploit vulnerabilities in web applications, helping improve your company's security posture.

**Security Analyst Role:** Security Analysts and Cybersecurity Professionals utilize their knowledge of XSS to analyze and mitigate security threats. This relates to roles such as vulnerability assessment, incident response, and security monitoring.

**Incident Response and Forensics:** When a security incident occurs, knowledge of XSS is beneficial to incident responders and digital forensics professionals. Investigate and analyze web-based attacks to understand their scope and impact.

**Security Consulting:** Professionals in security consulting or advisory roles can leverage their XSS knowledge to help clients identify and remediate vulnerabilities in their web applications.

**Compliance and Risk Management**: Understanding XSS is important for compliance and risk management professionals. This allows you to assess and mitigate risks related to web application security and ensure compliance with industry standards and regulations.

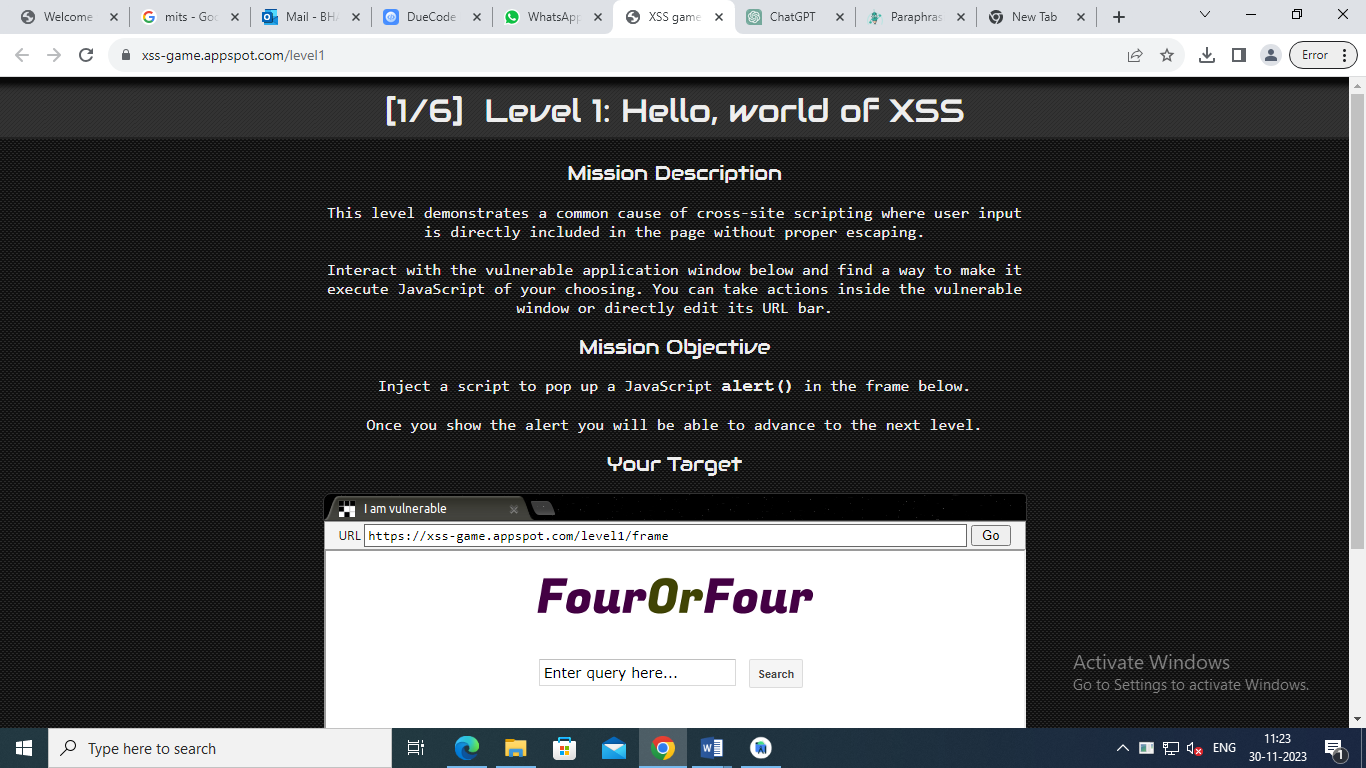
**Bug Bounty Program:** Participation in a bug bounty program provides individuals with the opportunity to apply their XSS knowledge ethically. By discovering and responsibly disclosing XSS vulnerabilities, you can earn rewards and help improve web security.

This allows professionals to adapt to new attack vectors, tools, and defenses. It's important to note that learning XSS is valuable, but it needs to be part of a broader understanding of web application security.

So to learn this we can start it by playing it as a game,for beginners to learn from scratch to know how it works and how the things are running in the background . To identify the vulnerabilities in a web site or an application you need to have some understanding on some web programming languages like HTML, JAVA SCRIPT.

The first level is very simple if you had the general idea about XSS.

XSS-Game | level 1



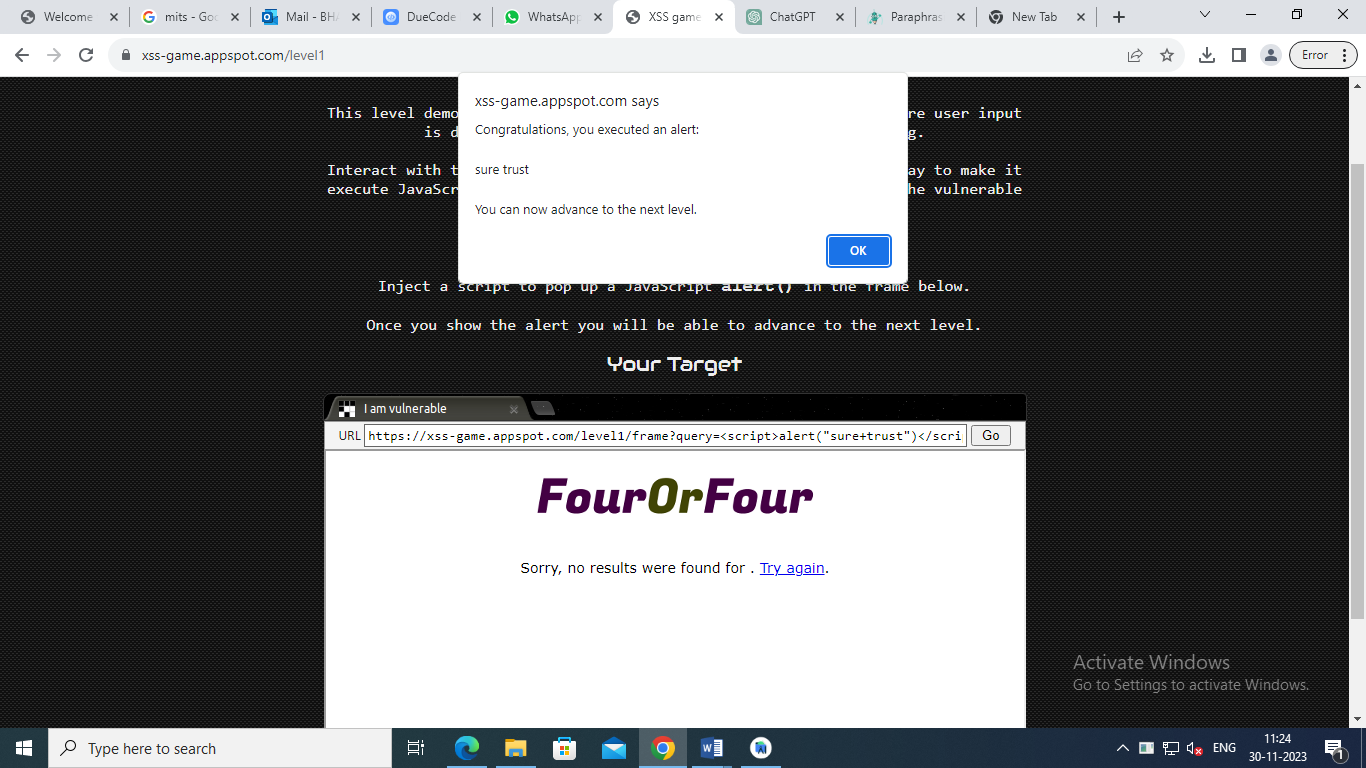
We have a website with input box, where we put text and it will search for it, so let’s try to search for something.



As you can see, the website returned the string that we have searched for, so let’s check if the website does any validation for the input.

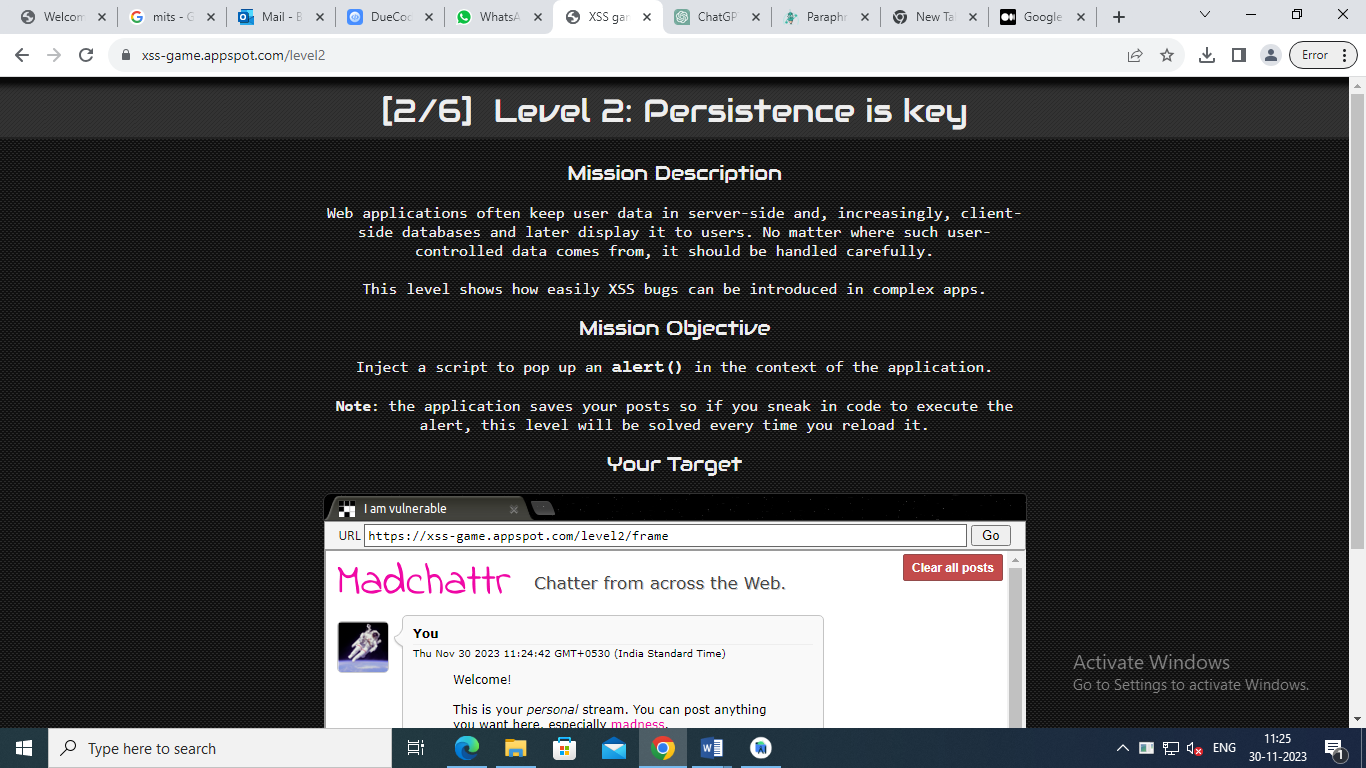
All we have to do is to add a simple script that shows a pop up and this is more than enough to prove that XSS vulnerability is in the site. To do that we can simply append our first search with the following code that allows the browser to display a popup with the message “sure trust”.

**<script> alert(“sure trust ”)</script>**



Let’s go to the next level, it’s a char forum where you can post a message, so let’s try the same previous code first

XSS-Game | level 2



but nothing shows except the text

A screenshot of a computer

Description automatically generated

let’s toggle the code and see which code is being executed from the moment we submit the text, if you use inspect element tools, you can see that when you submit the form, the following code will be executed

function() {  
var message = document.getElementById('post-content').value;  
DB.save(message, function() {  
displayPosts()  
});  
document.getElementById('post-content').value = "";  
return false;  
}

so the code will get the text we have inserted, it will send it to backend to save the data, then it will execute function “displayPosts()”, let’s check that function code.( you can get the code by clicking on “Toggle code and check index.html code)

function displayPosts() {  
var containerEl = document.getElementById("post-container");  
containerEl.innerHTML = "";  
  
var posts = DB.getPosts();  
for (var i=0; i<posts.length; i++) {  
var html = '<table class="message"> <tr> <td valign=top> '  
+ '<img src="/static/level2\_icon.png"> </td> <td valign=top '  
+ ' class="message-container"> <div class="shim"></div>';  
  
html += '<b>You</b>';  
html += '<span class="date">' + new Date(posts[i].date) + '</span>';  
html += "<blockquote>" + posts[i].message + "</blockquote";  
html += "</td></tr></table>"  
containerEl.innerHTML += html;  
}  
}

First, we will get the HTML element that has ID of “post-container”  
  
var containerEl = document.getElementById("post-container");

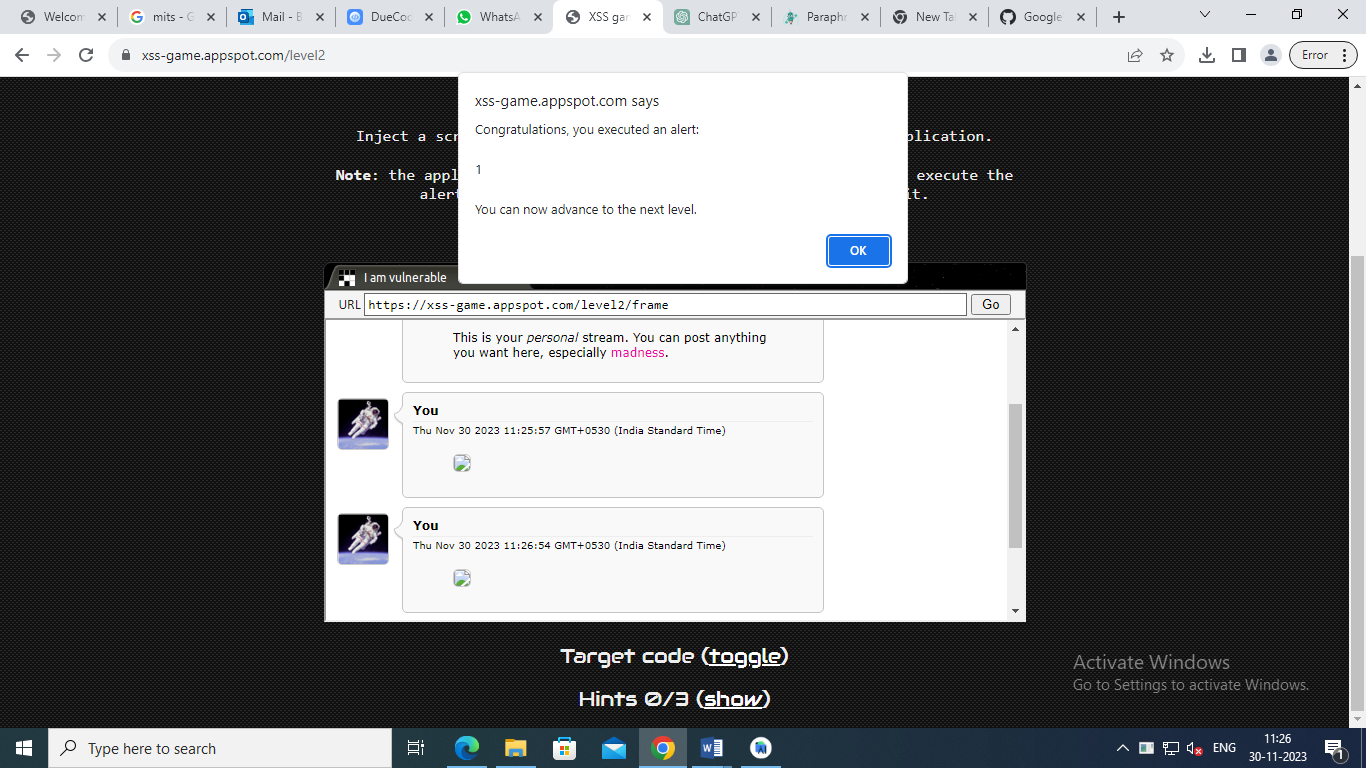
Then we will reset the inner HTML which means removing everything inside that element.  
  
containerEl.innerHTML = "";

After that, we will get all the posts saved in the Database in the backend  
  
var posts = DB.getPosts();

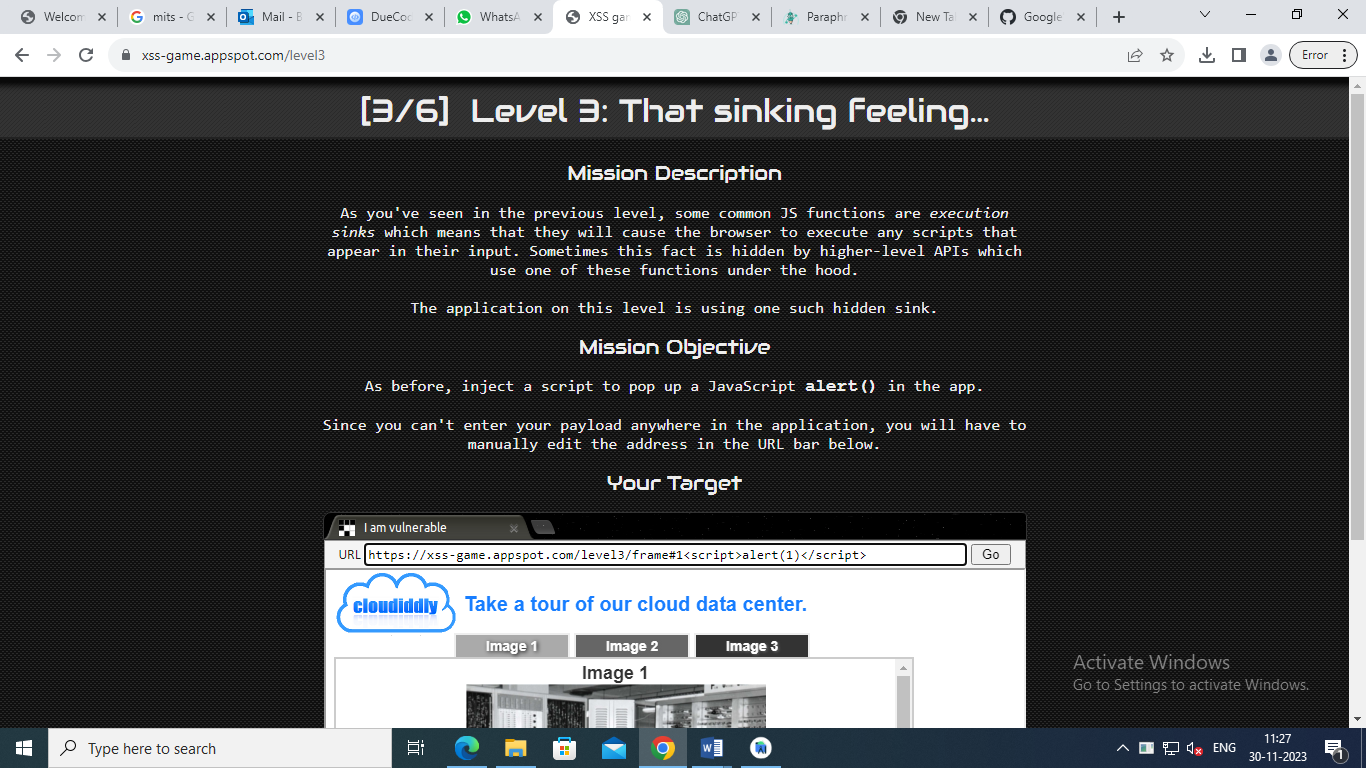
and now, we will go into for loop which allows us to traverse all the posts the script received from the backend.  
  
for (var i=0; i<posts.length; i++)

then the script will structure the html to be displayed, this is where it starts displaying each and single card and then it append it to the post before.  
  
But there’s one thing that is important, it uses the following command to append the html code.  
  
containerEl.innerHTML += html;

why is it important you might ask, well it’s simple because innerHTML doesn’t allow <script> tags to run when it’s added to document. you can check for more information from here and here.  
  
So what other way to execute a javascript, especially an alert() function. A simple but effective method is using the “onxxx” functions such as onload, onclick, onsubmit etc.  
  
But, you need an element that support the above methods, also you can find more information from here.  
  
so let’s try to do that, create html tag with onload event that will execute the alert() function to display a message.  
  
But, which elements supports onload event, a simple google search will give you many results, we only need one, some of the tags are:  
  
<body>, <frame>, <frameset>, <iframe>, <img>, <link>, <script>  
so let’s use the img tag  
  
<img src="anyimagelink" onload="alert('hi')"/>



XSS-Game | level 3

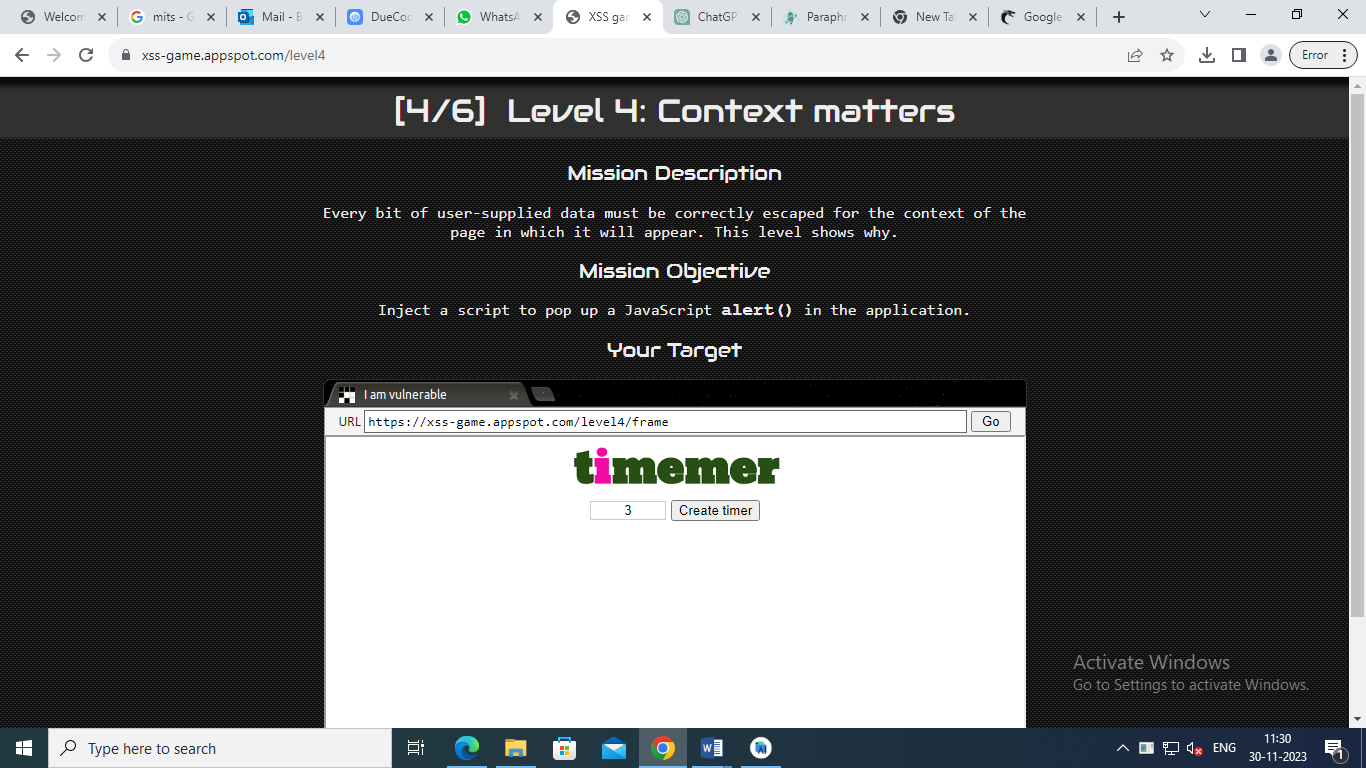


let’s open the javascript and analyze it, mainly we will focus on this part only  
  
function chooseTab(num) {  
// Dynamically load the appropriate image.  
var html = "Image " + parseInt(num) + "<br>";  
html += "<img src='/static/level3/cloud" + num + ".jpg' />";  
$('#tabContent').html(html);  
  
window.location.hash = num;  
  
also , you can see that when the tab is clicked will trigger the above  
  
<div class="tab active" id="tab1" onclick="chooseTab('1')">Image 1</div>  
and then the image will be displayed as the following  
  
<img src="/static/level3/cloud1.jpg">  
at the same time, you can see that the javascript is setting the url to the number was clicked  
  
https://xss-game.appspot.com/level3/frame#1  
so let’s try first to change the value after the # for example, let’s try the following:  
  
https://xss-game.appspot.com/level3/frame#2  
we can see the image is changing to the one in the second tab, without the need to click on the tab.  
  
So, what we’re looking for is to escape the <img> tag and execute on load event to success in our xss.  
  
since onload event requires the <img> tag to have a proper and valid source, we will use the image “1.jpg”  
  
so, what we want to do is to specify the image, close the src=”” and then execute the onload, so we should do something like.  
  
https://xss-game.appspot.com/level3/frame#1.jpg' onload="alert('hi')"  
and voila, it worked perfectly.

A computer screen shot of a computer screen

Description automatically generated

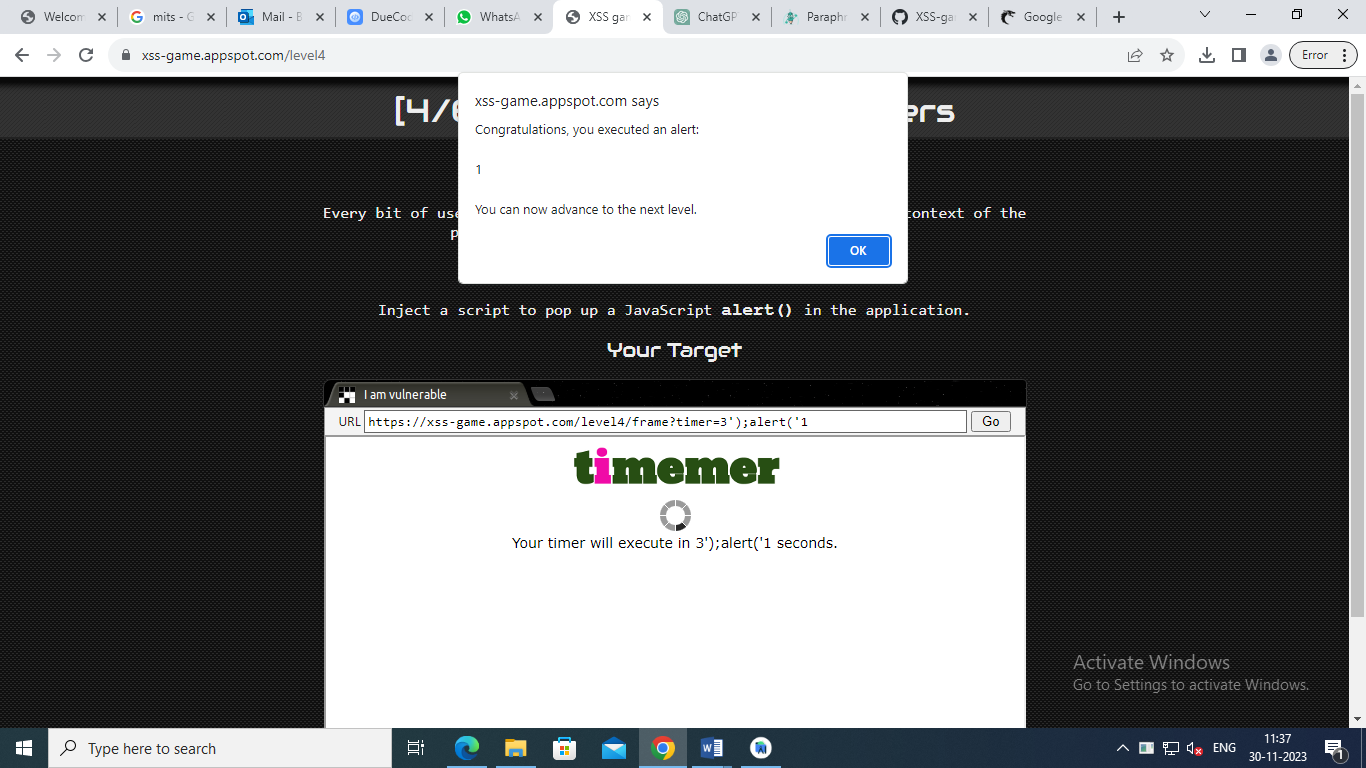
**XSS-Game | level 4**



where if you click on “create timer”, it will send it to new page and shows an alert after 3 seconds (the default)  
  
XSS-Game | level 4 Test  
also you can see the URL is structured like this  
  
https://xss-game.appspot.com/level4/frame?timer=3  
now let’s set the timer to large number so we can have enough time to see the code, if you use inspect element, you can see the following code:  
  
XSS-Game | level 4 Timer  
so the timer is inserted inside the onload in the html, this gives us huge advantage, cause we can close the “startTimer” function and write our own, for example if we use the following:  
  
https://xss-game.appspot.com/level4/frame?timer=')  
you can see the function is closed  
  
  
ok, so let’s try to end function

https://xss-game.appspot.com/level4/frame?timer=');alert('hi  
but nothing changed, you can see the input is filtered after “;”,

one way to escape that is by using URL encoding so if search online on what is the URL encoding of “;” you will get “%3B”, let’s try it  
  
and congrats, we got the right solution, now you might ask why we left the “alert(‘” as half open, well because we closed the “setTimer” function but the closing brackets are still in the HTML as you can see above, so if you closed it like this  
  
https://xss-game.appspot.com/level4/frame?timer=')%3Balert('hi')  
you will get the following in HTML which will break the code and won’t run  
  
<img src="/static/loading.gif" onload="startTimer('');alert('hi')');">



These are the four levels that are present in the game and different levels has different types of vulnerabilities has there exploits and we have seen how we can execute them to gain access to the site .So these are some basic vulnerabilities and there are more than we have seen here .

So this is the complete report on the XSS game containing all the levels .