

# WAPH-jaindy Web Application Programming and Hacking

**Instructor:** Dr. Phu Phung

## Student

**Name:** Divyani Jain

**Email:** jaindy@mail.uc.edu

**Short-bio:** I have a total of 6 years of work experience in IT as a software developer.



Figure 1: Divyani Headshot!

## Repository Information

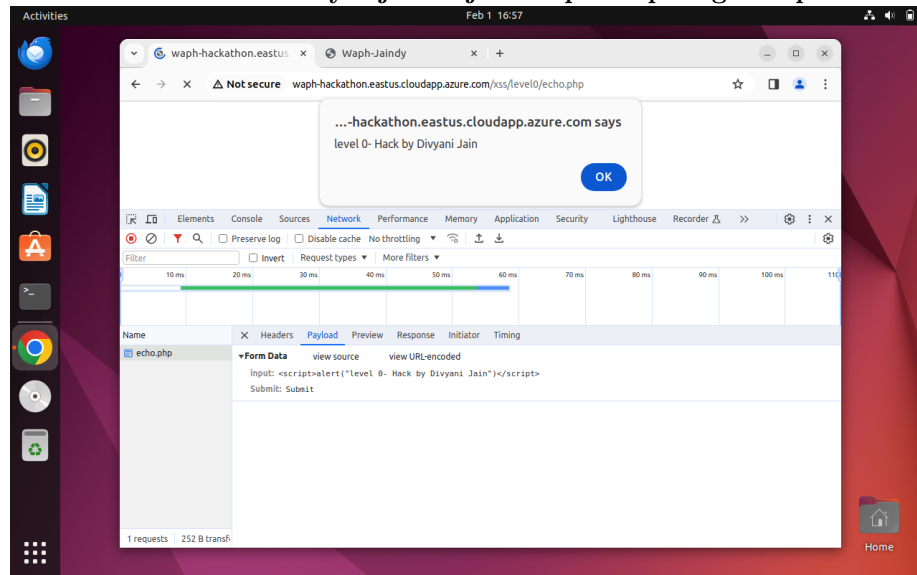
Repository's URL: <https://github.com/jaindy/waph-jaindy.git>

## Lab Overview Hackathon 1 - Cross-site Scripting Attacks and Defenses

In hackathon 1, I learned about cross site attacks and prevention technique. Learn about htmlentities and strip\_tag used to prevent hacking. Implemented the input validation and encoding input data in server side.

Task 1. Perform cross site scripting attacks on below Url and display my name in alert. [http://waph-hackathon.eastus.cloudapp.azure.com/xss/level\[0-6\]/echo.php](http://waph-hackathon.eastus.cloudapp.azure.com/xss/level[0-6]/echo.php)

## Level 0: Hacked level0 by injected javascript script tag in input field.

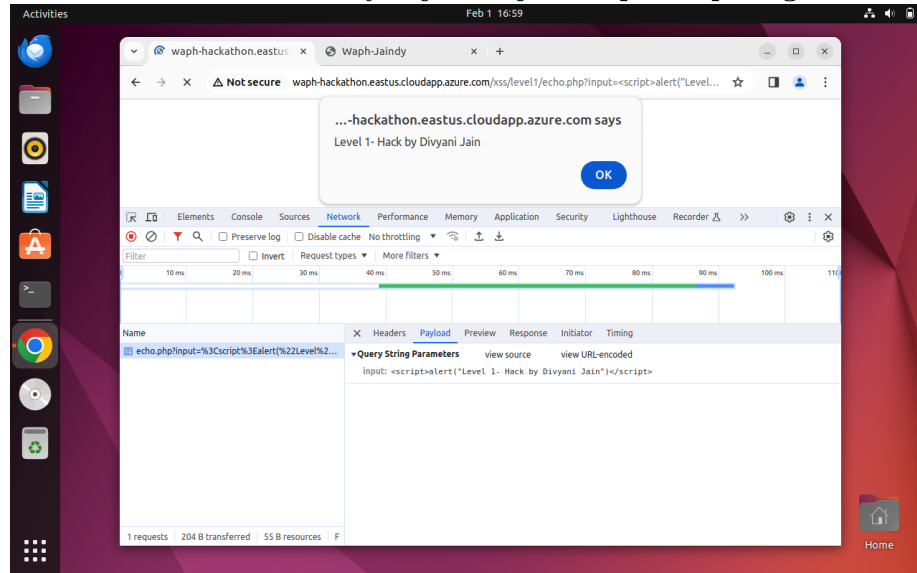


Source code:

```
<html>
<body>
<form action="/echo.php" method="POST">
  Input:<input type="text" name="input" >
  <input type="submit" name="Submit">
</form>
<?php
    echo $_REQUEST["input"];
?>
</body>
</html>
```

---

## Level 1: Hacked level1 by injected javascript script tag in url.



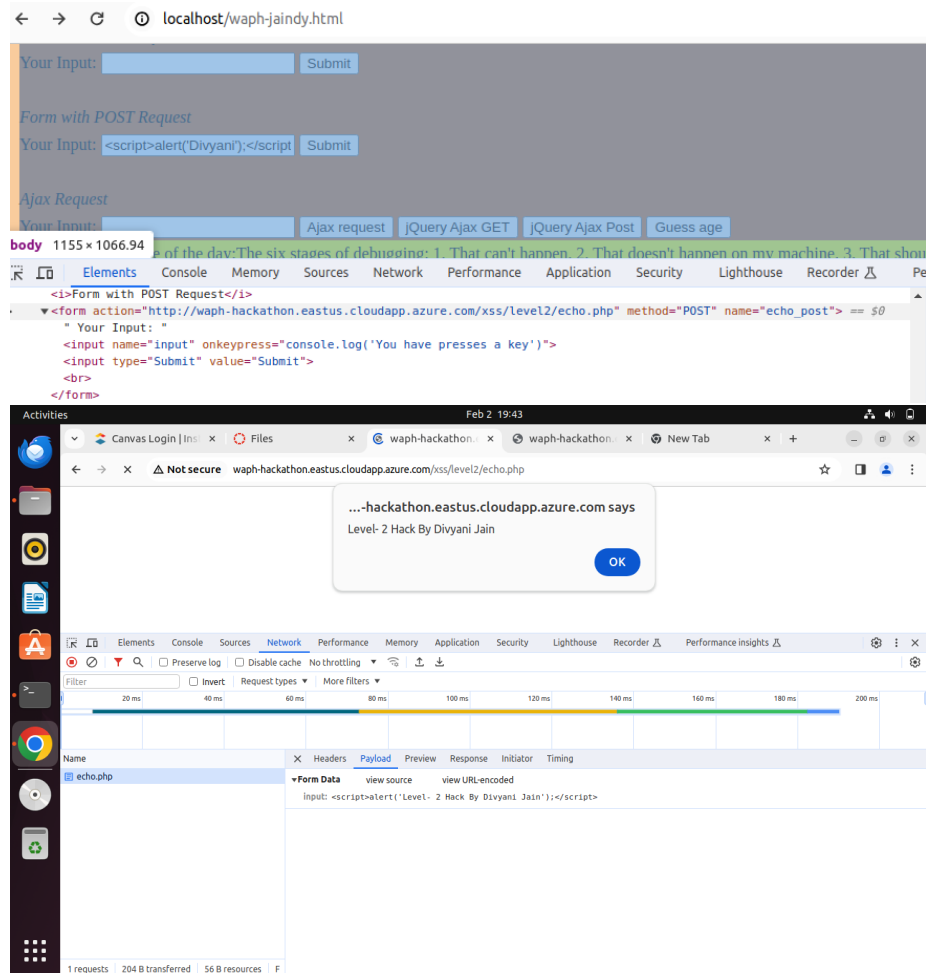
Source code:

```
<?php
```

```
if(!isset($_REQUEST["input"])) { <br />
    die("{\"error\":\"Please provide 'input' field in an HTTP GET Request\"}"); <br />
} else { <br />
    echo strip_tags($_REQUEST["input"],"<body>"); <br />
}
?>
```

---

Level 2: Hacked the level2 by changing the URL path in action attribute of the Form tag in HTML and send data in POST request.



Source code:

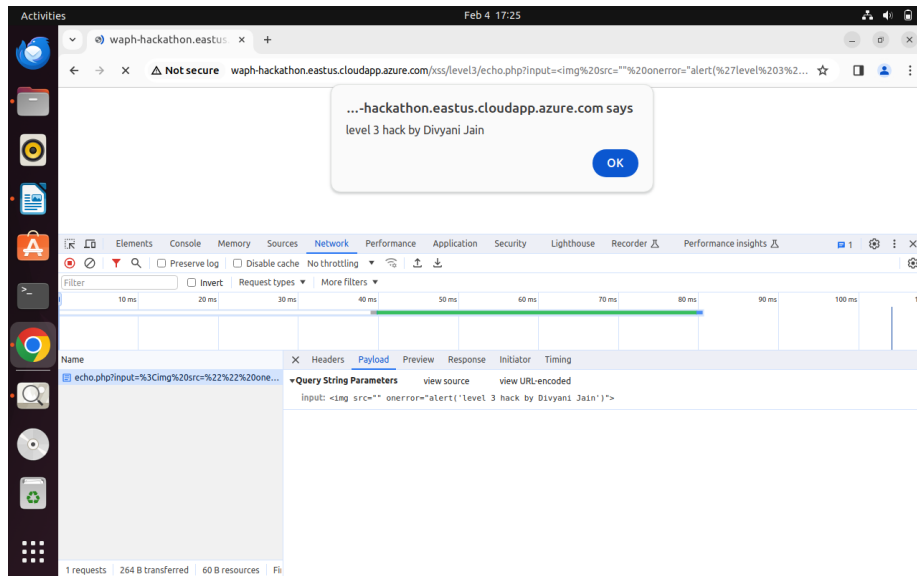
```
<?php
```

```
if(!isset($_REQUEST["input"])) { <br />
    die("{\"error\":\"Please provide 'input' field in an HTTP POST Request\"}"); <br />
} else { <br />
    echo strip_tags($_REQUEST["input"],"<body>"); <br />
}
```

```
?>
```

**Level 3: Hacked the level3 by providing alert in onerror event in image tag.** Source code of the echo.php web application: using strip\_tag to

stripped html and php tags.



Source code:

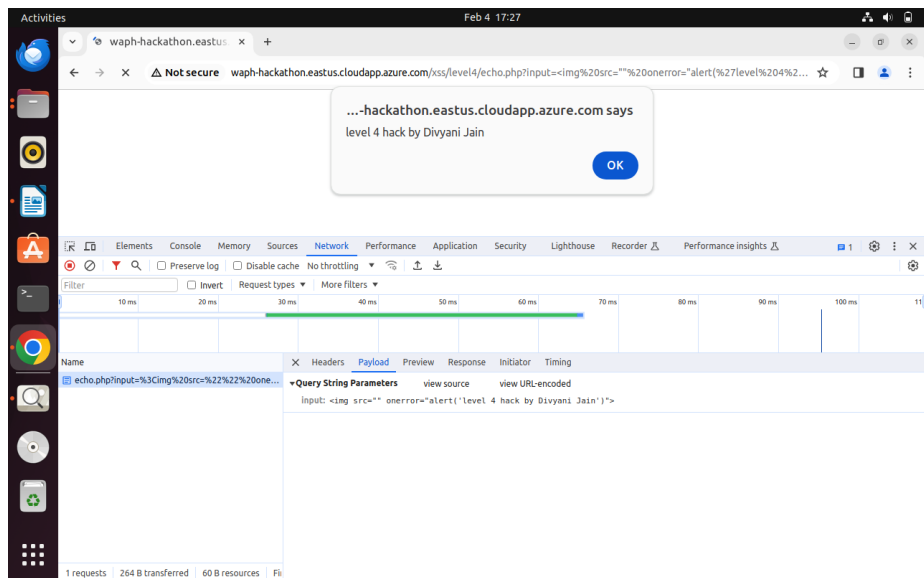
```
<?php
```

```
if(!isset($_REQUEST["input"])) { <br />
    die("{\"error\":\"Please provide 'input' field\"}"); <br />
} else { <br />
    echo strip_tags($_REQUEST["input"],"<body>"); <br />
}
```

```
?>
```

---

**Level 4: Hacked the level4 by providing alert in onerror event in image tag.** Source code of the echo.php web application: using strip\_tag to stripped html and php tags and specify img table in second parameter.



Source code:

```
<?php
if(!isset($_REQUEST["input"])) { <br />
    die("{\"error\":\"No 'script' is allowed!\"}"); <br />
} else { <br />
    echo strip_tags($_REQUEST["input"], "<body>"); <br />
}
?>
```

---

**Level 5: Hacked the level5 by providing window alert in onload event in body tag.** Source code of the echo.php web application: using strip\_tag to stripped html and php tags.

Source code:

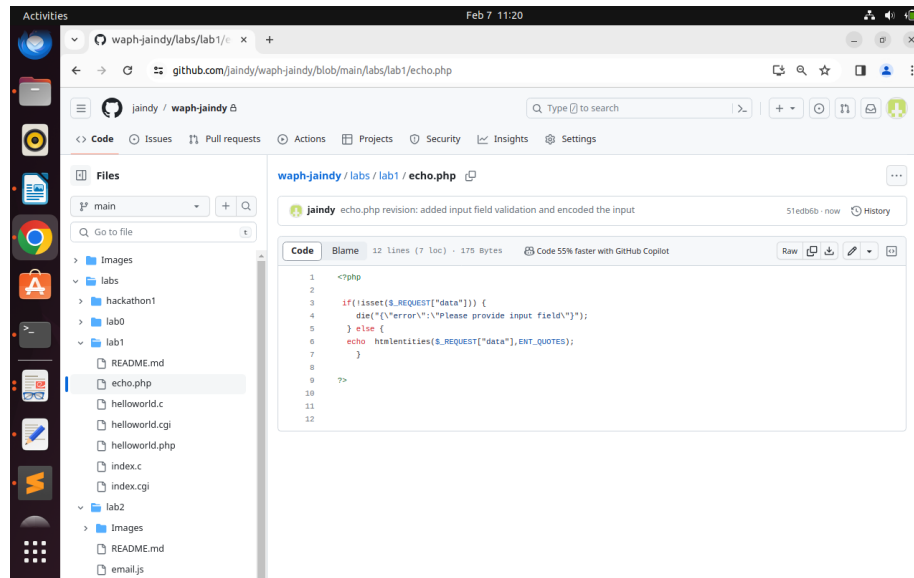
```
<?php

if(!isset($_REQUEST["input"])) { <br />
    die("{\"error\":\"Please provide 'input' field\"}"); <br />
} else { <br />
    echo strip_tags($_REQUEST["input"], "<body>"); <br />
}
?>
```

---



tag in input field, URL or via Post request. Also, added input validation for Joke API.



The screenshot shows a web browser displaying the GitHub repository for 'waph-jaindy/labs/lab1'. The file 'echo.php' is selected, showing its code. The code implements server-side validation for an input field. It checks if the 'data' parameter is set in the request. If not, it displays an error message. If it is set, it echoes the input after escaping HTML entities.

```
1 <?php
2
3 if(!isset($_REQUEST["data"])) {
4     die("\nerror!\nPlease provide input field\n");
5 } else {
6     echo htmlentities($_REQUEST["data"], ENT_QUOTES);
7 }
8
9 ?>
10
11
12
```

Figure 3: Server side validation!



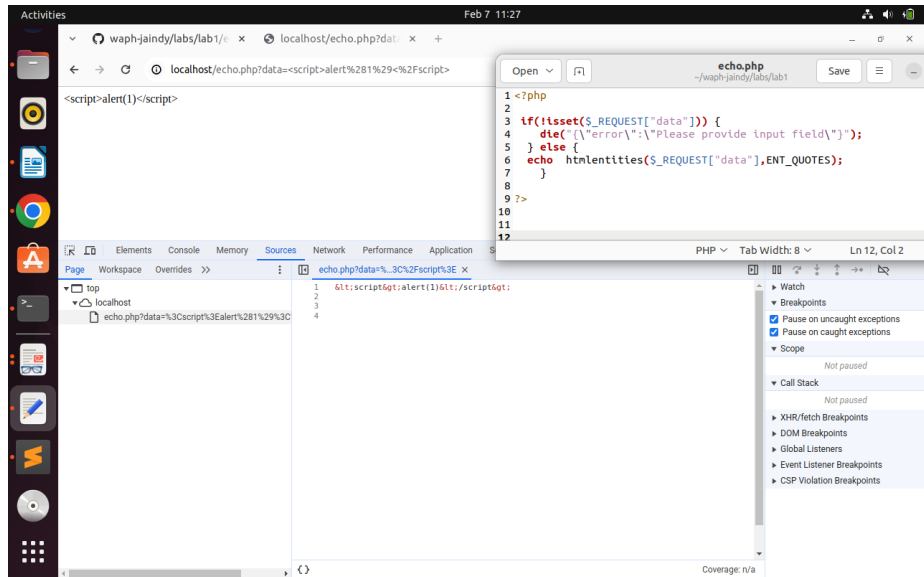


Figure 4: Input alert validation!

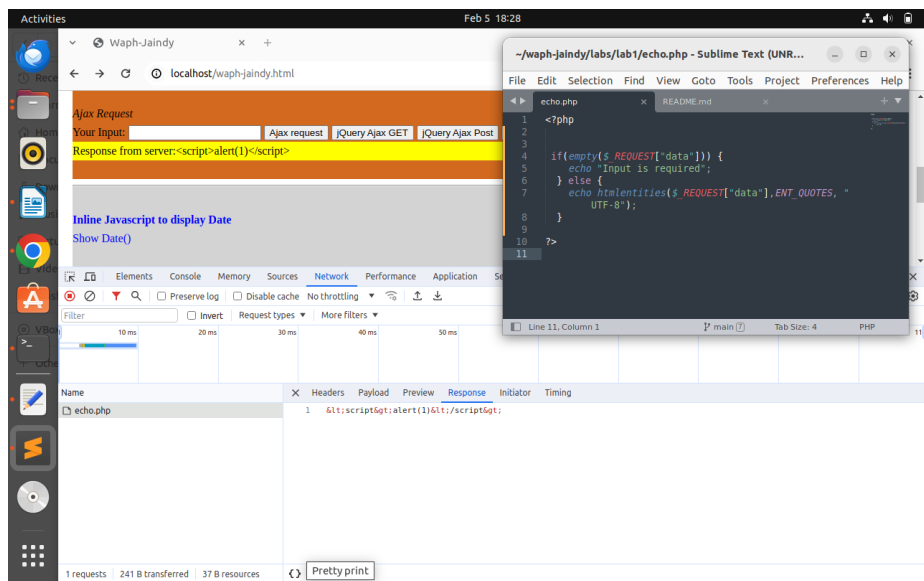


Figure 5: Prevent alert execution!

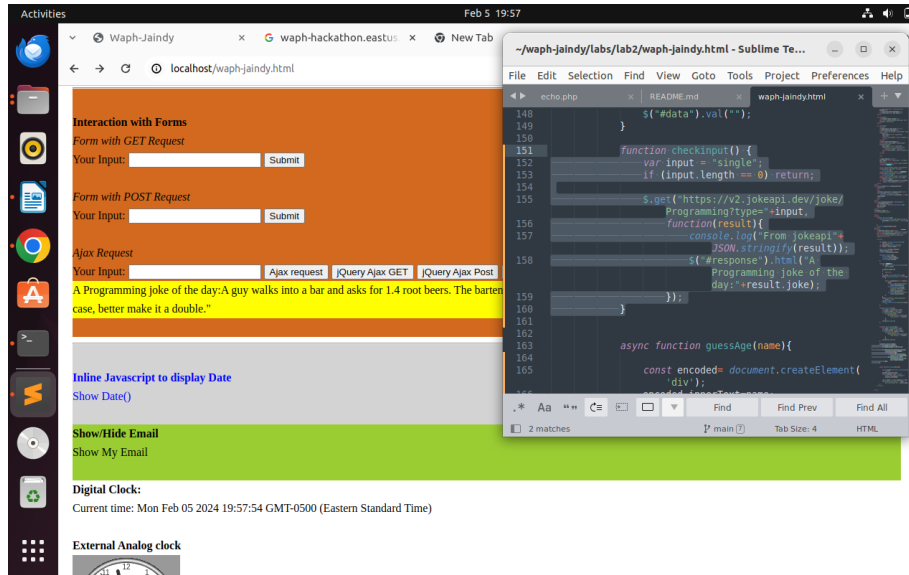


Figure 6: External API validation!

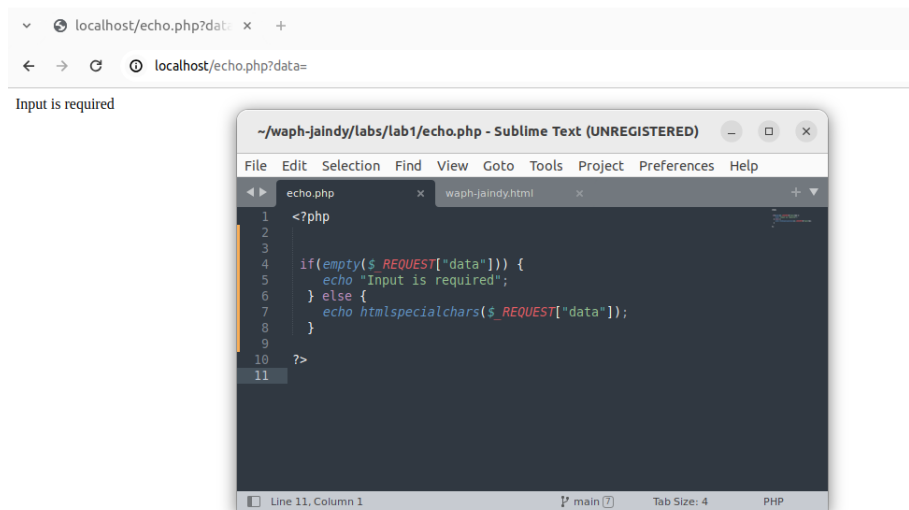


Figure 7: Input field validation!