

# Cybersecurity Project Report

---

## Understanding Web Fuzzing, HTTP Requests, and Burp Suite

**Name:** Abhay Raj

**College:** LBS College of Engineering, Kasaragod

**Submission Date:** 20 June 2025

### 1. Introduction

This project explores the use of Burp Suite and web fuzzing to understand how data flows between a browser and server in web applications. By analyzing GET and POST requests and performing input fuzzing, I simulated the basic approach of ethical hackers in testing web applications.

### 2. Website Tested

- DVWA (Damn Vulnerable Web Application)
- Hosted locally using XAMPP
- URL: `http://localhost/dvwa`
- Purpose: To safely practice and understand web application vulnerabilities

### 3. GET and POST Request Analysis

- **POST Request (Captured via Burp Suite):**

`POST /dvwa/login.php HTTP/1.1`

`Host: localhost`

`Content-Type: application/x-www-form-urlencoded`

`username=test&password=123&Login=Login&user_token=...`

- **GET Request (Example from Navigation):**

`GET /dvwa/vulnerabilities/fi/?page=include.php HTTP/1.1`

`Host: localhost`

## 4. Tools Used

Tool	Purpose
1. Burp Suite	Intercept and modify HTTP requests
2. DVWA	Vulnerable web app for testing
3. Firefox / Burp Browser	Sending traffic through Burp Proxy
4. XAMPP	Local server for hosting DVWA

## 5. Fuzzing with Burp Suite Intruder

I performed fuzzing by sending multiple inputs to the username field of the login request using Burp Suite Intruder.

- Steps Taken:

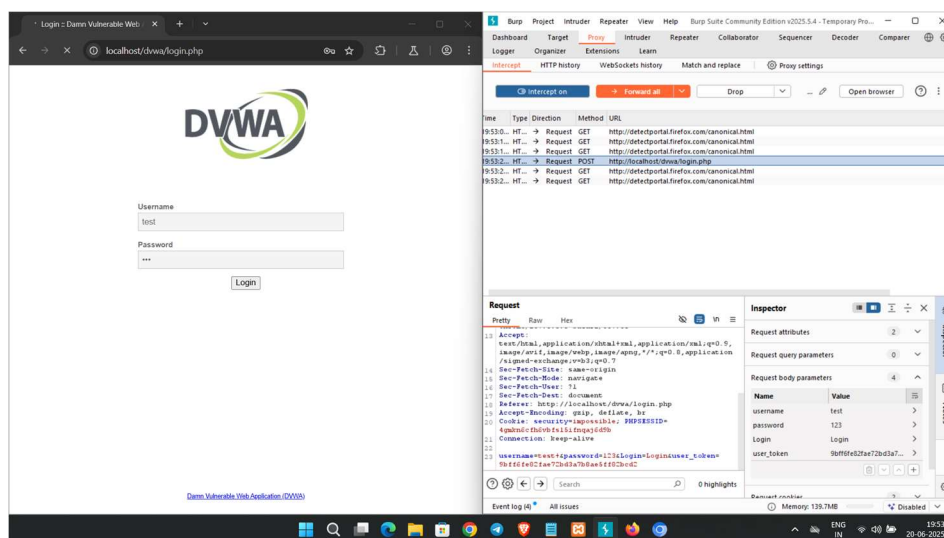
- Captured the POST request using Burp
- Highlighted the username field value (test) and marked it as a payload position
- Entered the following payloads in Burp's Simple List:

```
admin
root
' OR 1=1 --
<script>alert(1)</script>
test
```

## 6. Screenshots

### Screenshot 1: DVWA Login Page with Burp Intercepting POST Request

Shows the DVWA login form with Burp Suite capturing the login request to /login.php.



## Screenshot 2: Burp Suite – Intercepted POST Request to Login

Displays the raw intercepted request with username=test and request body parameters.

The screenshot shows the Burp Suite interface with the 'Intercept' tab selected. A list of intercepted requests is displayed, with the following request highlighted:

Time	Type	Direction	Method	URL	Status code	Length
19:53:09 20 Jul	HTTP	→ Request	GET	http://detectportal.firefox.com/canonical.html		
19:53:14 20 Jul	HTTP	→ Request	GET	http://detectportal.firefox.com/canonical.html		
19:53:19 20 Jul	HTTP	→ Request	GET	http://detectportal.firefox.com/canonical.html		
19:53:23 20 Jul	HTTP	→ Request	POST	http://localhost/dvwa/login.php		
19:53:24 20 Jul	HTTP	→ Request	GET	http://detectportal.firefox.com/canonical.html		
19:53:29 20 Jul	HTTP	→ Request	GET	http://detectportal.firefox.com/canonical.html		

The 'Request' tab is selected, showing the raw intercepted request. The request body parameters are visible in the 'Inspector' tab:

Name	Value
username	test
password	123
login	Login
user_token	9bffd62fae72bd3a7...

## Screenshot 3: Burp Suite Intruder – Payload Position and Payload List

Shows the configured position on the username field and the list of fuzzing inputs.

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. The 'Payloads' tab is active, showing the configured position and list of fuzzing inputs.

**Target:** http://localhost

**Positions:** Add 5, Clear 5, Auto 5

**Payloads:**

- Payload position: All payload positions
- Payload type: Simple list
- Payload count: 2
- Request count: 2

**Payload configuration:**

This payload type lets you configure a simple list of strings that are used as payloads.

**Payload list:**

- admin root ' OR '1' = <script>alert(1)</script> test

**Payload processing:**

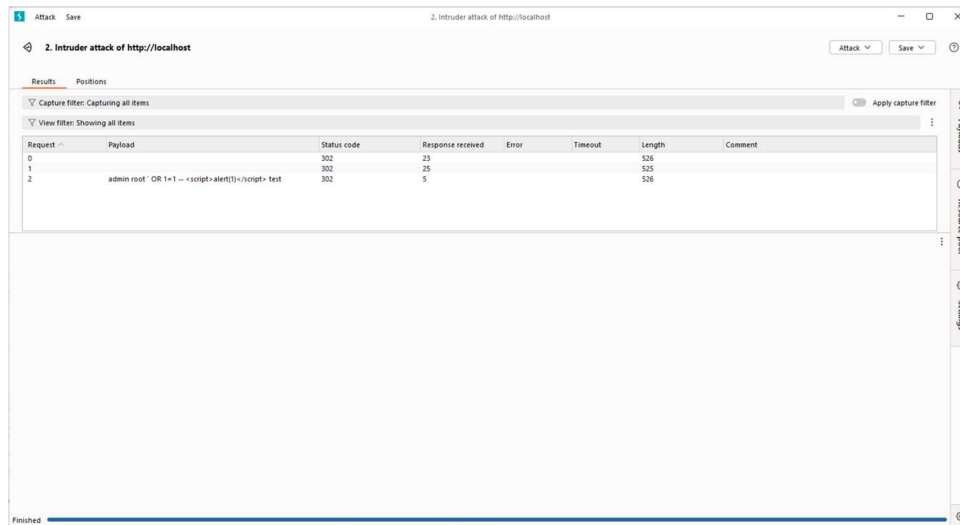
**Payload encoding:**

This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

☒ URL-encode these characters: ,/ = < > ? @ : ; ' " \ \\*

## Screenshot 4: **Intruder Attack Results**

Displays the results of each payload sent via Intruder, showing status and length differences.



The screenshot shows the 'Attack' window in Burp Suite, titled '2. Intruder attack of http://localhost'. The 'Results' tab is active, displaying a table of attack results. The table has columns for 'Request', 'Payload', 'Status code', 'Response received', 'Error', 'Timeout', 'Length', and 'Comment'. Three requests are listed, all with a status code of 302. The first two requests have a length of 526, while the third request, which includes a payload, has a length of 525. The 'Response received' column shows values of 23, 25, and 5 for the three requests respectively. The 'Error' column is empty for all three requests. The 'Timeout' column is empty for all three requests. The 'Comment' column is empty for all three requests. The 'Payload' column for the third request shows the payload: 'admin root' OR 1=1 -- <script>alert(1)</script> test. The 'Request' column for the third request shows the request: 'admin root' OR 1=1 -- <script>alert(1)</script> test. The 'Status code' column for the third request shows 302. The 'Response received' column for the third request shows 5. The 'Error' column for the third request is empty. The 'Timeout' column for the third request is empty. The 'Length' column for the third request shows 525. The 'Comment' column for the third request is empty. The 'Attack' button is disabled, and the 'Save' button is visible. The 'Finished' progress bar is at the bottom of the window.

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		302	23			526	
1		302	25			525	
2	admin root' OR 1=1 -- <script>alert(1)</script> test	302	5			525	

## 7. Observations

The server responded differently to each payload in terms of length and response time. Although no successful login or bypass occurred, the behavior confirmed that fuzzing can reveal how a server handles various input types. Some responses showed faster rejection, while others took longer to process — indicating backend handling differences.

## 8. Conclusion

This project gave me hands-on experience with HTTP request analysis and web fuzzing using Burp Suite. I learned how input fields can be tested for unexpected behavior using automated tools. The exercise improved my understanding of web security fundamentals and how attackers and testers approach application testing from a data flow perspective. Even though no major bug was exploited, the focus remained on ethical learning, structured testing, and effective tool usage.