

# CTF Report: Using BurpSuite to Bypass OTP and Extract the Flag

Ayoub Goubraim

September 8, 2025

## Abstract

This report documents the solution to the picoCTF challenge **IntroToBurp**. We configure a web proxy (BurpSuite), intercept and manipulate HTTP requests, and bypass an OTP (2FA) gate to access a protected page that reveals the flag. The objective is to illustrate a typical client-side validation weakness and a proper reporting structure for pentesting notes.

## 1 Challenge Overview

The challenge page indicates an on-demand instance with an OTP-protected flow. Our task is to analyze the requests and identify where server-side validation fails.

The screenshot shows the 'IntroToBurp' challenge page. At the top, the challenge name 'IntroToBurp' is displayed with a bookmark icon and a user profile icon. Below this, there are three tags: 'Easy' (green), 'Web Exploitation' (red), and 'picoCTF 2024' (blue). The page is divided into two main sections. The left section, titled 'Description', shows the author 'NANA AMA ATOMBO-SACEY & SABINE GISAGARA' and a message stating 'Additional details will be available after launching your challenge instance.' The right section shows the challenge status: 'This challenge launches an instance on demand. Its current status is: NOT\_RUNNING'. Below this is a 'Launch Instance' button. Further down, there is a 'Hints' section with a question mark icon and two hint buttons labeled '1' and '2'. At the bottom, a grey bar indicates '32,616 users solved' and '48% Liked'. Below this is a 'Submit Flag' button and a text input field containing 'picoCTF{FLAG}'.

IntroToBurp

Easy Web Exploitation picoCTF 2024

AUTHOR: NANA AMA ATOMBO-SACEY & SABINE GISAGARA

**Description**

Additional details will be available after launching your challenge instance.

This challenge launches an instance on demand.  
Its current status is: **NOT\_RUNNING**

**Launch Instance**

**Hints**

1 2

32,616 users solved

48% Liked

Submit Flag

picoCTF{FLAG}

Figure 1: picoCTF challenge page *IntroToBurp*.

## 2 Environment & Tools

- **OS:** Kali Linux.
- **Browser:** Firefox configured via **FoxyProxy** to forward traffic to BurpSuite (127.0.0.1:8080).
- **Proxy:** BurpSuite Community Edition, Proxy Intercept enabled.

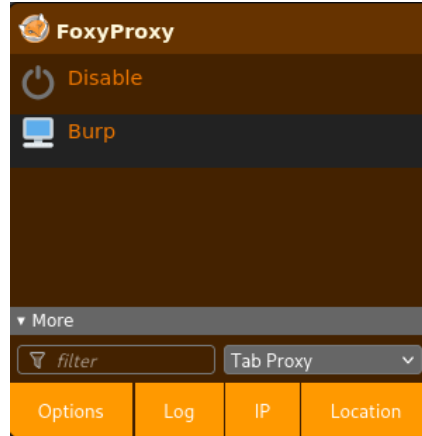


Figure 2: FoxyProxy profile switching to route the tab through Burp.

## 3 Methodology

We follow a concise, reproducible workflow:

- M1 Recon:** Browse flow and identify endpoints (registration, dashboard, OTP).
- M2 Intercept:** Capture HTTP requests/responses with Burp Proxy.
- M3 Manipulate:** Modify parameters related to OTP handling.
- M4 Validate:** Forward altered requests and confirm server behavior.
- M5 Document:** Capture evidence and extract flag.

## 4 Registration & OTP Flow

We begin by creating an account on the provided instance. After registration, accessing the dashboard triggers a 2FA page requesting an OTP value.

← → ↻ 🏠 titan.picocf.net:62042

🔊 OffSec 🐧 Kali Linux 📦 Kali Tools 📄 Kali Docs 🗨️ Kali Forums 🏹 Kali NetHunter 🔥 Exploit-DB 🔍 Google Hacking DB

### Registration

Full Name:

Username:

Phone Number:

City:

Password:

Figure 3: Registration form.

← → ↻ 🏠 titan.picocf.net:62042/dashboard

🔊 OffSec 🐧 Kali Linux 📦 Kali Tools 📄 Kali Docs 🗨️ Kali Forums 🏹 Kali NetHunter 🔥 Exploit-DB 🔍 Google Hacking DB

### 2fa authentication

Figure 4: OTP authentication page (2FA gate).

## 5 Intercepting Registration Data

During registration, BurpSuite captured the POST request that contained all the user-supplied data (`full_name`, `username`, `phone_number`, `city`, and `password`). This interception confirmed that sensitive information was transmitted in clear form, which emphasizes the importance of properly protecting user input.

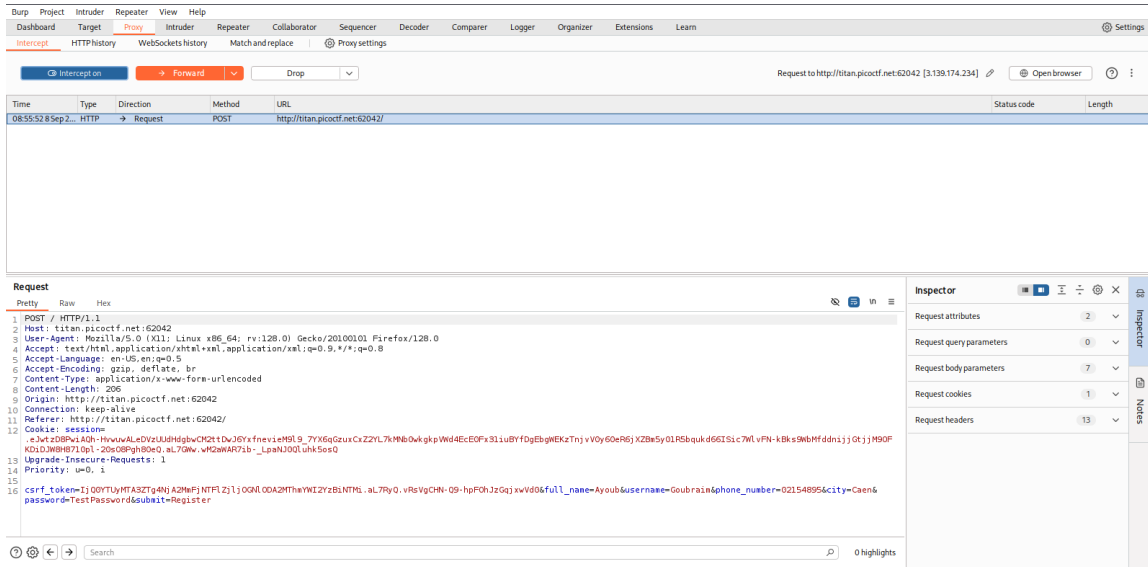


Figure 5: BurpSuite intercept showing submitted registration data.

## 6 Interception and Request Analysis

With Intercept *on*, we capture requests to the dashboard and OTP submission endpoints. Burp's Inspector shows cookies, headers, and body parameters.

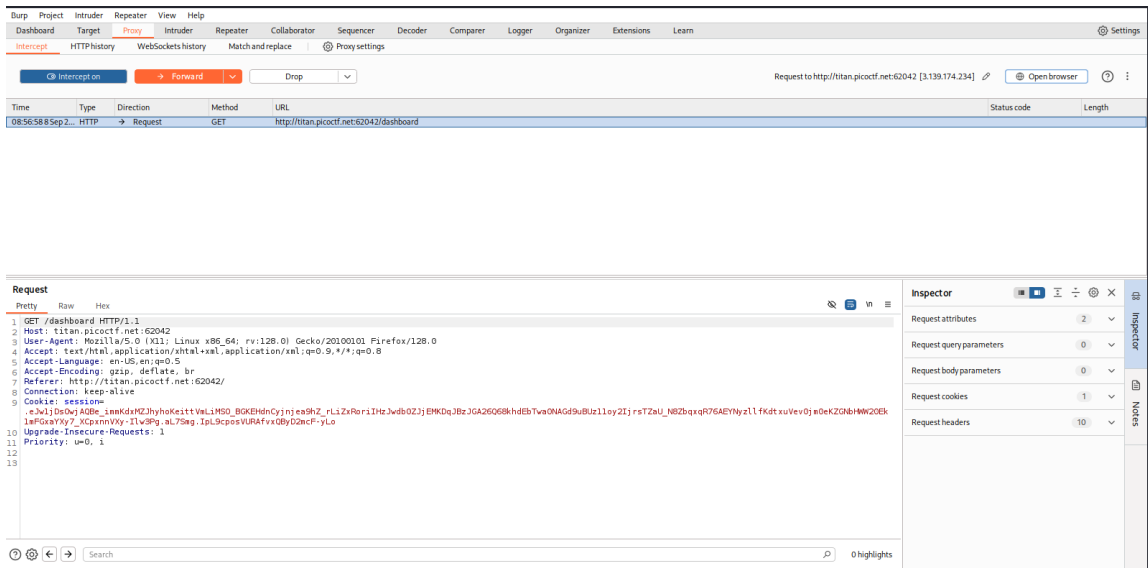


Figure 6: GET request to `/dashboard` captured in BurpSuite.

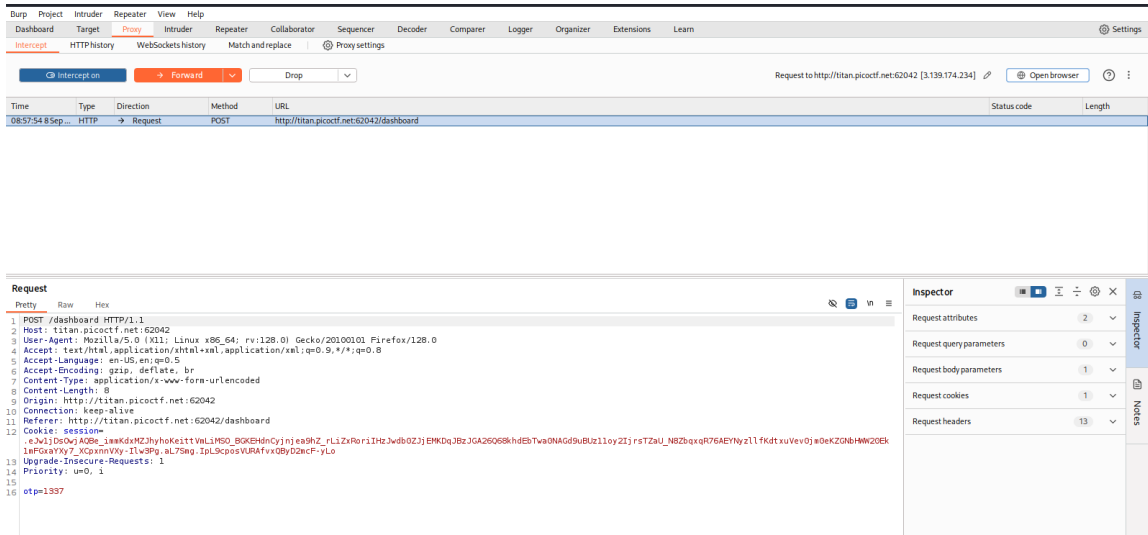


Figure 7: POST request with OTP parameter intercepted.

## 7 Manipulating the OTP Check

The OTP value is client-controlled. We test two manipulations:

1. Submit an arbitrary OTP (e.g., 1337).
2. Remove the `otp` parameter entirely before forwarding.

In this instance, removing the parameter bypasses validation and grants access.

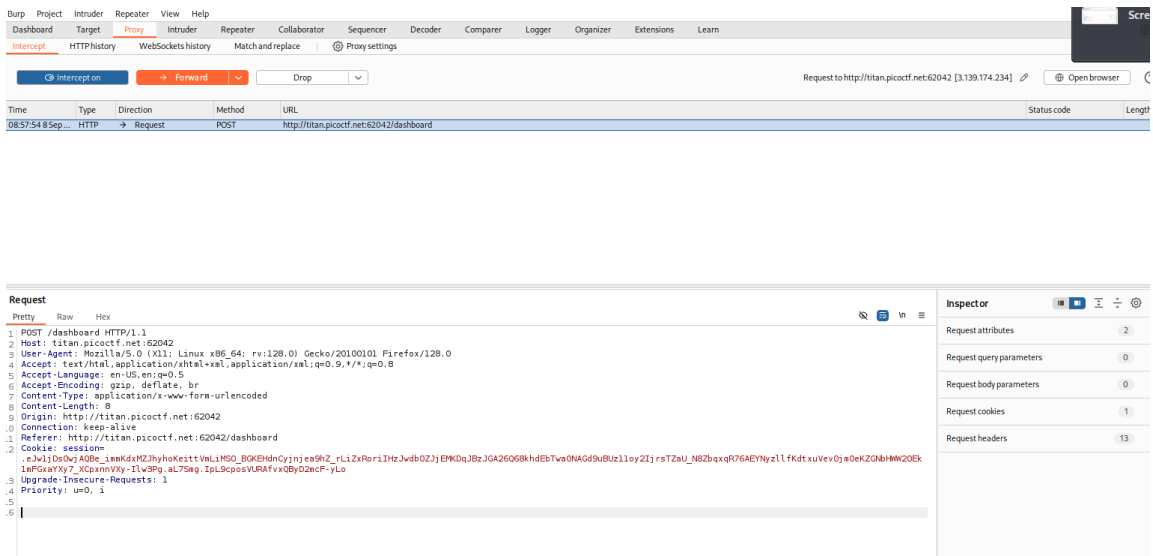


Figure 8: Deleting the `otp` parameter in the intercepted POST request.

## 8 Results: Flag Extraction

Upon forwarding the modified request, the server responds with a welcome message that includes the flag.

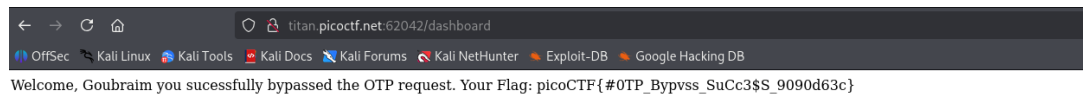


Figure 9: Successful bypass: the protected page reveals the flag.

## 9 Discussion & Mitigations

This behavior indicates insufficient server-side validation. Recommended fixes include:

- Enforce OTP verification strictly on the server, rejecting missing or invalid OTPs.
- Bind OTP to session/user and apply expiry, rate limiting, and replay protection.
- Use CSRF defenses and ensure consistent input validation/sanitization.
- Log and alert on anomalous OTP flows.

## 10 Conclusion

We demonstrated a straightforward OTP bypass via BurpSuite interception and parameter manipulation. The exercise underscores the necessity of robust server-side checks for authentication steps. Screenshots and steps herein provide a reproducible path to the result for audit and learning purposes.