PCTF – SecureAuth™ Write-Up

Challenge Link: http://18.212.136.134:5200/
Category: Web
Difficulty: Beginner

Challenge Description

One of our interns just finished their Python coding bootcamp and made this fancy shmancy demo web page. They say it's really secure, but I don't know... The flag can be wrapped in FLAG{...} or pctf{...} — both are accepted.

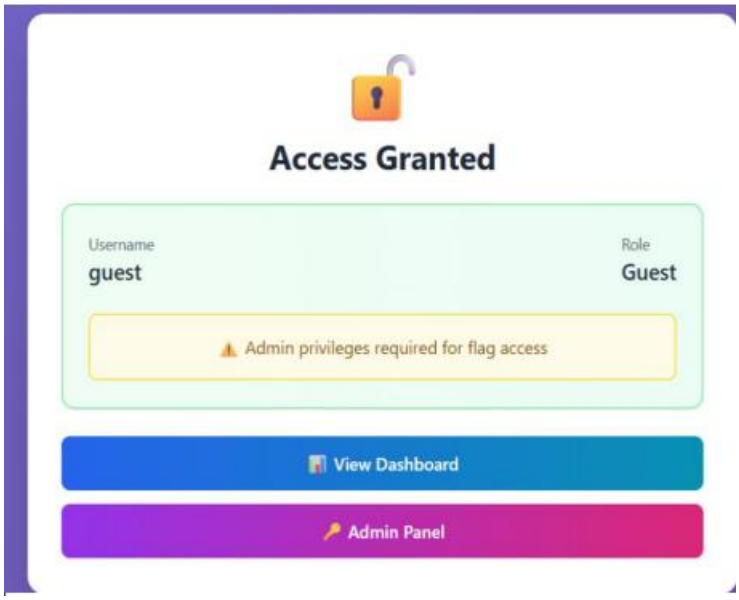
Step-by-Step Walkthrough

1. Visit the login page

You can log in using:

guest / guest123

This shows a normal user page, but not the flag. We need **admin**.



2. Look closely at the login page

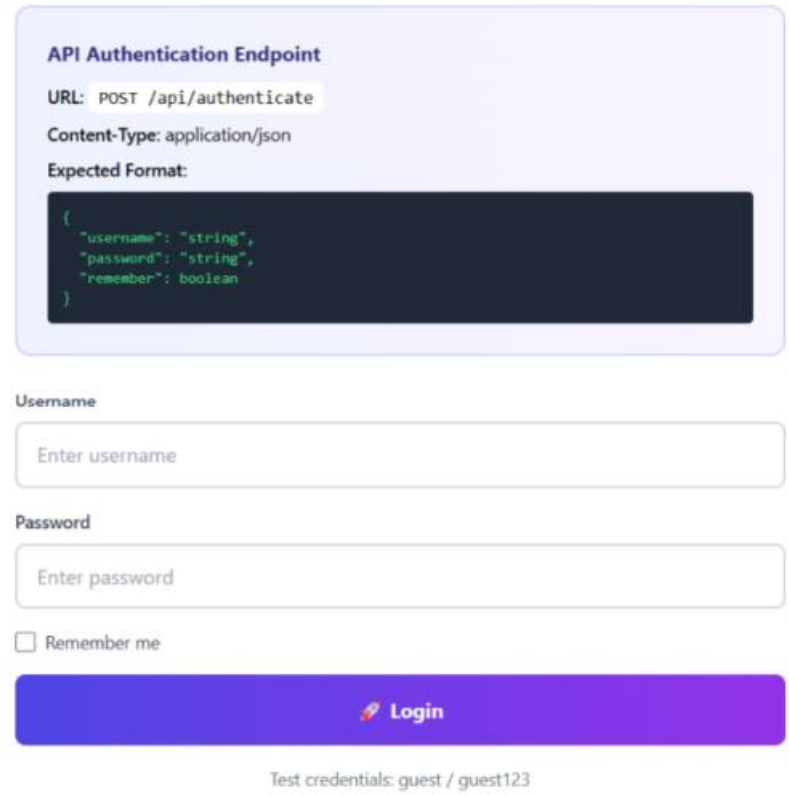
It mentions an API format:

POST /api/authenticate

Content-Type: application/json

So the site has two login systems:

- The normal HTML form: /login
- The JSON API: /api/authenticate (this is the vulnerable one)



3. Intercept the request in Burp Suite

- The normal form sends:
- POST /login
- Content-Type: application/x-www-form-urlencoded
- But the vulnerable endpoint is **/api/authenticate**, not /login.

4. Modify the request

Change:

- **Post:** /api/authenticate
- **Content-Type:** application/json
- **Body:** a crafted JSON object

Send this:

{

"username": "admin",

"password": null,

"remember": true

}

Why this works

Beginners often write insecure logic like:

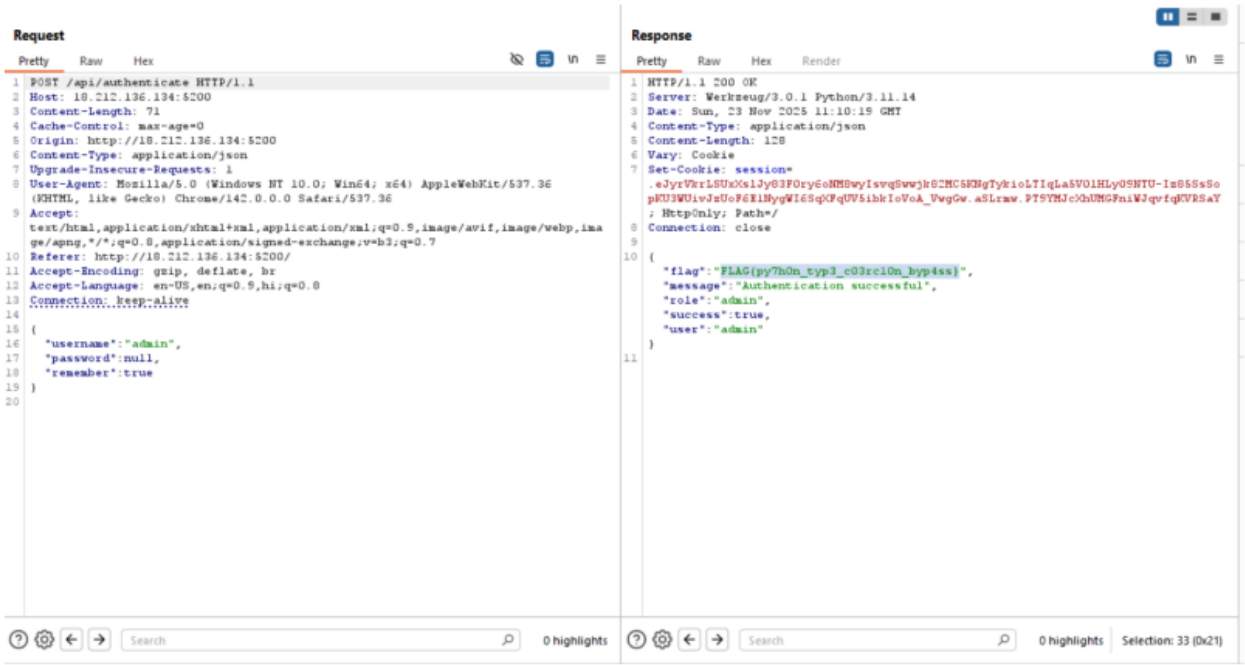
if data["remember"] == True:

allow_login()

This means setting "remember": true causes the server to skip checking the password completely.

Forward the Request

You get admin access — and the flag.



Flag

FLAG{py7h0n_typ3_c03rc10n_byp4ss}

Summary

The JSON login API has a logic bug.
If you send "remember": true, the backend skips password verification.
This lets us log in as admin without knowing the password.