Examining the source code reveals the way the database works: We are dealing with a **GraphQL DB**, to which we can make direct queries via the **/query** endpoint. The first part of the challenge is finding a way to making any request we want, which I solved with a simple python script:

```
import base64, json
import requests

query = """query_here"""
data = {
    'query': query
}
data = "query="+base64.b64encode(json.dumps(data).encode('utf-8')).decode('utf-
8')
url = 'http://34.159.151.77:31736/query'
headers = {
    'Content-Type': 'application/x-www-form-urlencoded',
}
response = requests.post(url, data=data, headers=headers)
print("Response: ", response.text)
```

Using this script we can make any request we want, so let's start with a general enumeration. The payloads are taken from https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/GraphQL%20Injection

{__schema{types{name}}}

Sending this query reveals the types of the table. What is interesting are the **UserObject** and **PostObject** types, so we need to look deeper. Next, I used the "**URL decoded query to dump the database schema**", which I will not paste here as it's very long. This query dumps the whole definitions of each type, so it will be used in understanding the types mentioned previously.

After saving the big response to a file and using a json formatter, we can find the definition of **UserObject**. Its fields are name, email, **randomStr1ngtoInduc3P4in**, posts. Hmmm...

{allUsers{edges{node{randomStr1ngtoInduc3P4in}}}}

This query results in a lot of fake flags telling us "Try harder!" and "Nope!" and other stuff. I simply copied the whole node and used Replace in a text file to remove all the useless text. Turns out, among all the junk was the flag.

Made with love by: AndreiCat