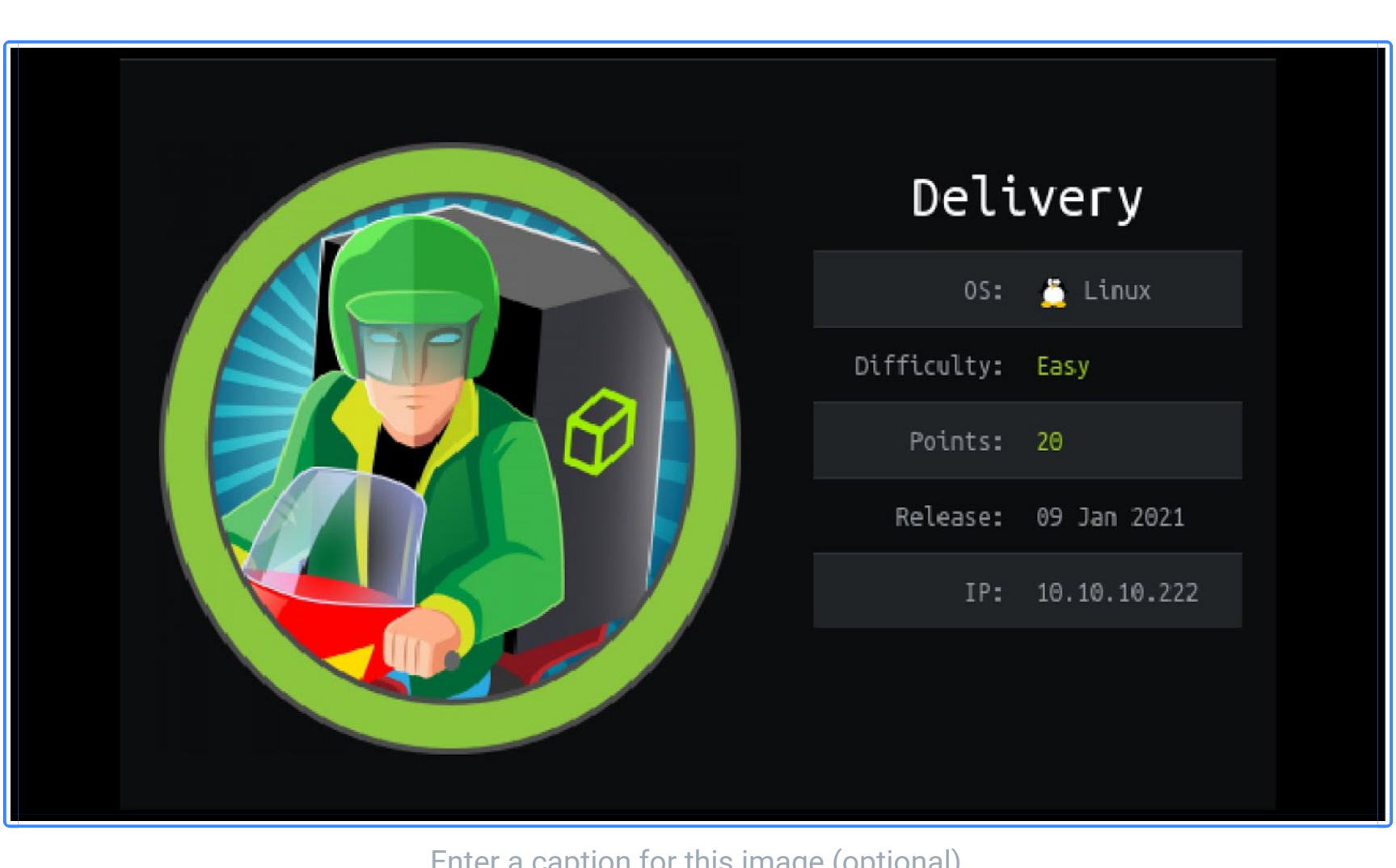
Delivery - 16th Jan 21

10.10.10.222



Enter a caption for this image (optional)

Scanning

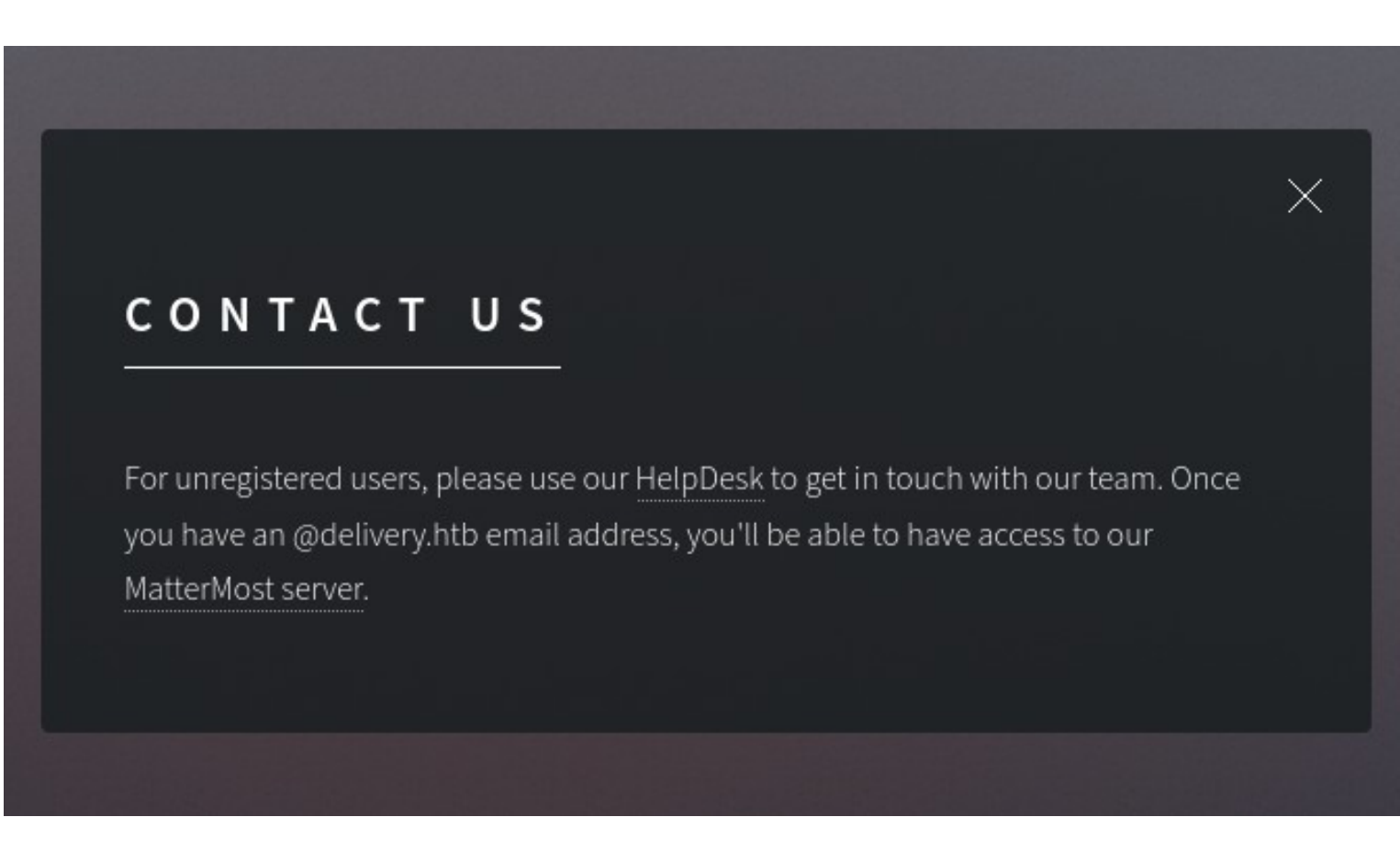
We can run `masscan_to_nmap.py` , a tool I made that you can find on my Github. It runs a Masscan, identifies open ports, and then takes those open ports over to Nmap, and scans for versions and default scripts against those ports.

```
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 9c:40:fa:85:9b:01:ac:ac:0e:bc:0c:19:51:8a:ee:27 (RSA)
|   256 5a:0c:c0:3b:9b:76:55:2e:6e:c4:f4:b9:5d:76:17:09 (ECDSA)
|_  256 b7:9d:f7:48:9d:a2:f2:76:30:fd:42:d3:35:3a:80:8c (ED25519)
80/tcp   open  http    nginx 1.14.2
|_http-server-header: nginx/1.14.2
|_http-title: Welcome
8065/tcp open  unknown
| fingerprint-strings:
|   GenericLines, Help, RTSPRequest, SSLSessionReq, TerminalServerCookie:
|     HTTP/1.1 400 Bad Request
|     Content-Type: text/plain; charset=utf-8
|     Connection: close
|     Request
|   GetRequest:
|     HTTP/1.0 200 OK
|     Accept-Ranges: bytes
|     Cache-Control: no-cache, max-age=31556926, public
|     Content-Length: 3108
|     Content-Security-Policy: frame-ancestors 'self'; script-src 'self' cdn.rudder
|     Content-Type: text/html; charset=utf-8
|     Last-Modified: Sun, 10 Jan 2021 13:57:22 GMT
|     X-Frame-Options: SAMEORIGIN
|     X-Request-Id: a5usmiiy7tr8ufs8sngqgpohfr
|     X-Version-Id: 5.30.0.5.30.1.57fb31b889bf81d99d8af8176d4bbaaa.false
|     Date: Mon, 11 Jan 2021 09:21:11 GMT
```

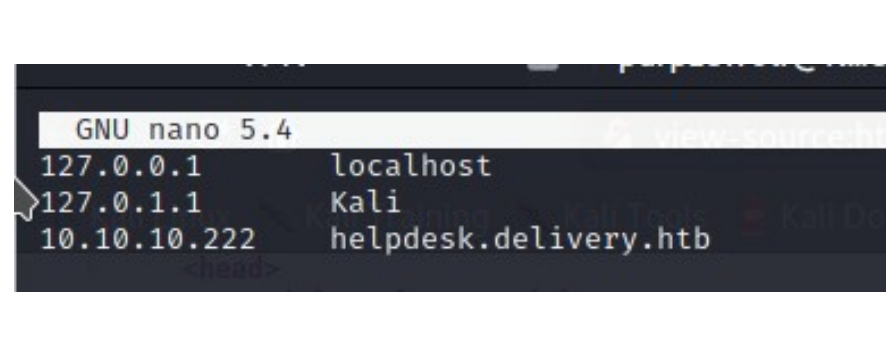
Enumeration

Port 80

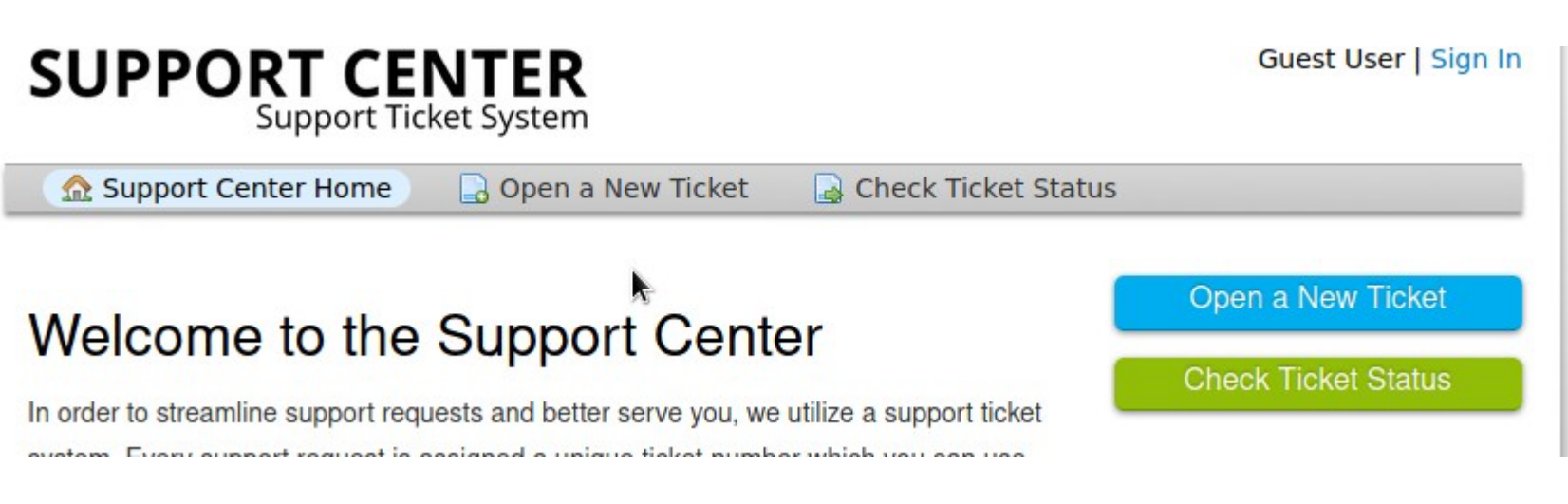
If we go to the website on port 80 and take a look, we can see that a **helpdesk subdirectory** is referenced.



To traverse to this, we need to add it to our hosts file: `sudo nano /etc/hosts.` . You may as well add `delivery.htb` here whilst you're at it.

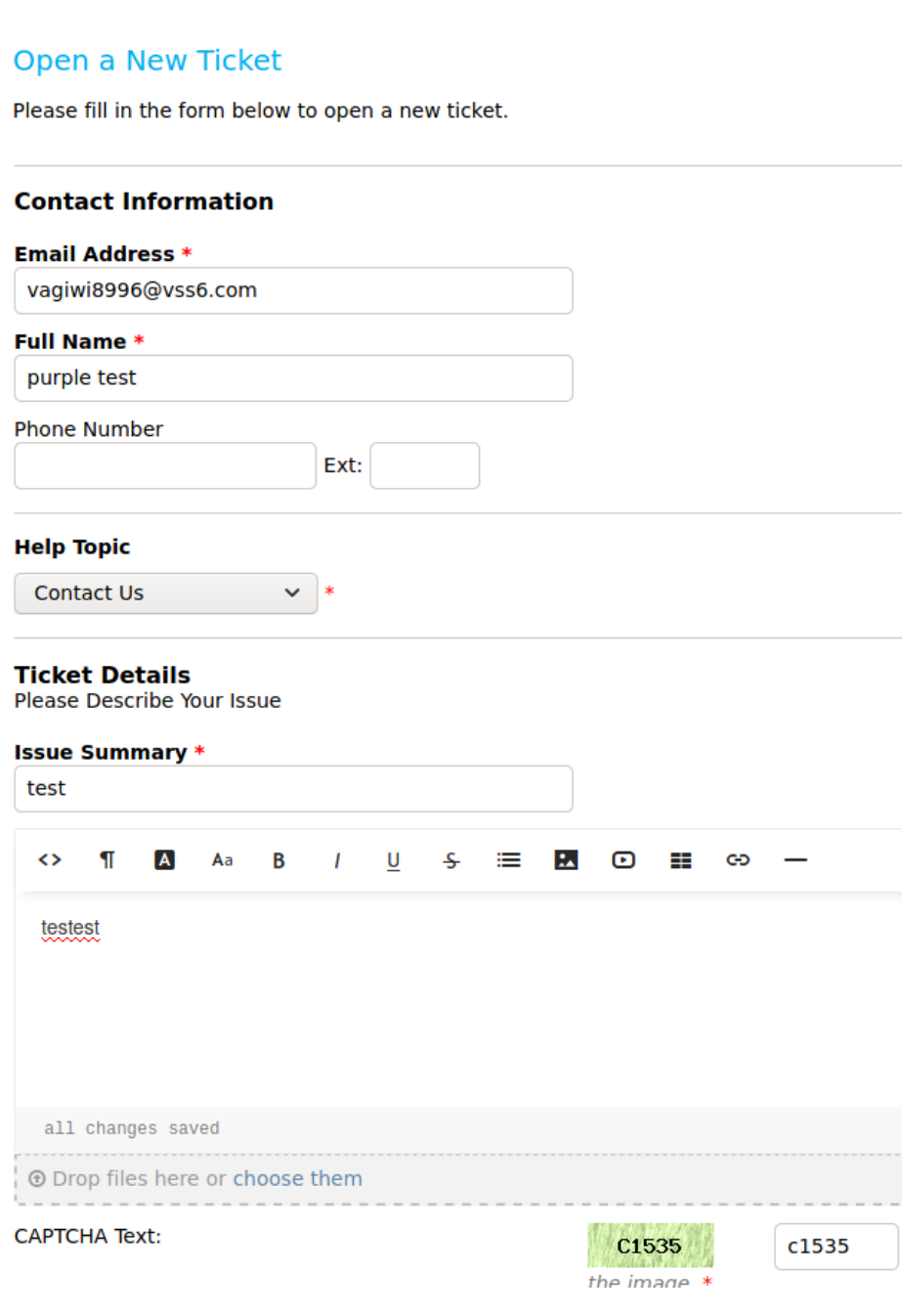


This helpdesk ticketing site has some functionality to set up accounts, raise tickets.

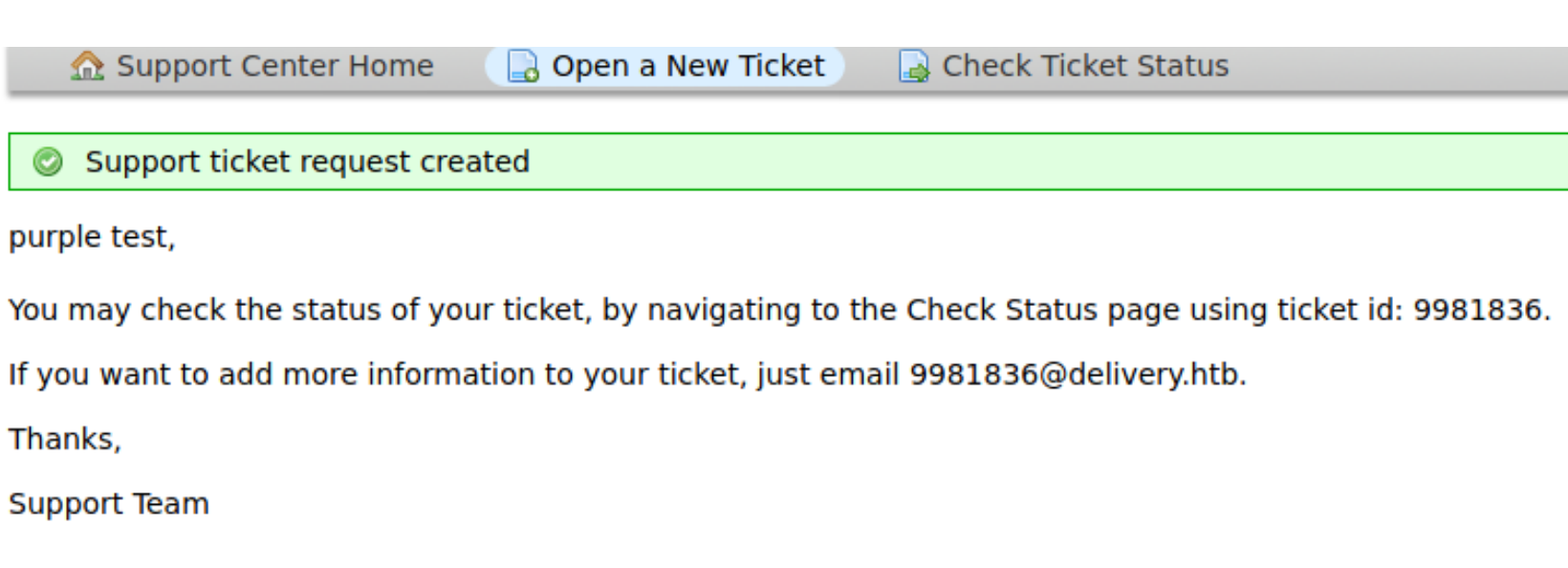


Make ticket

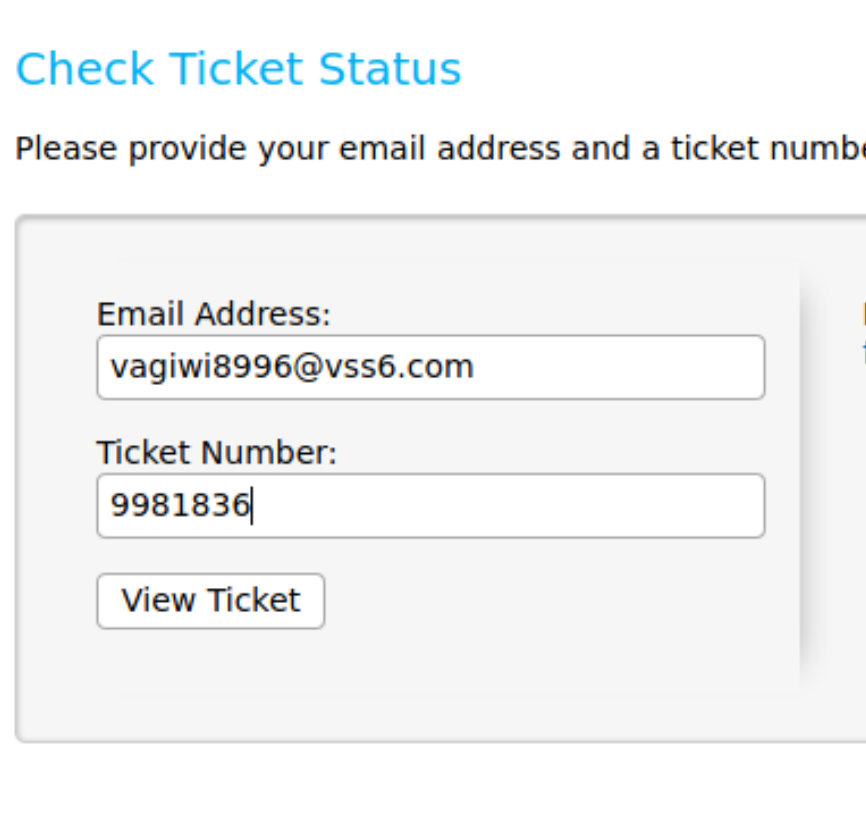
Open up a ticket and register some details

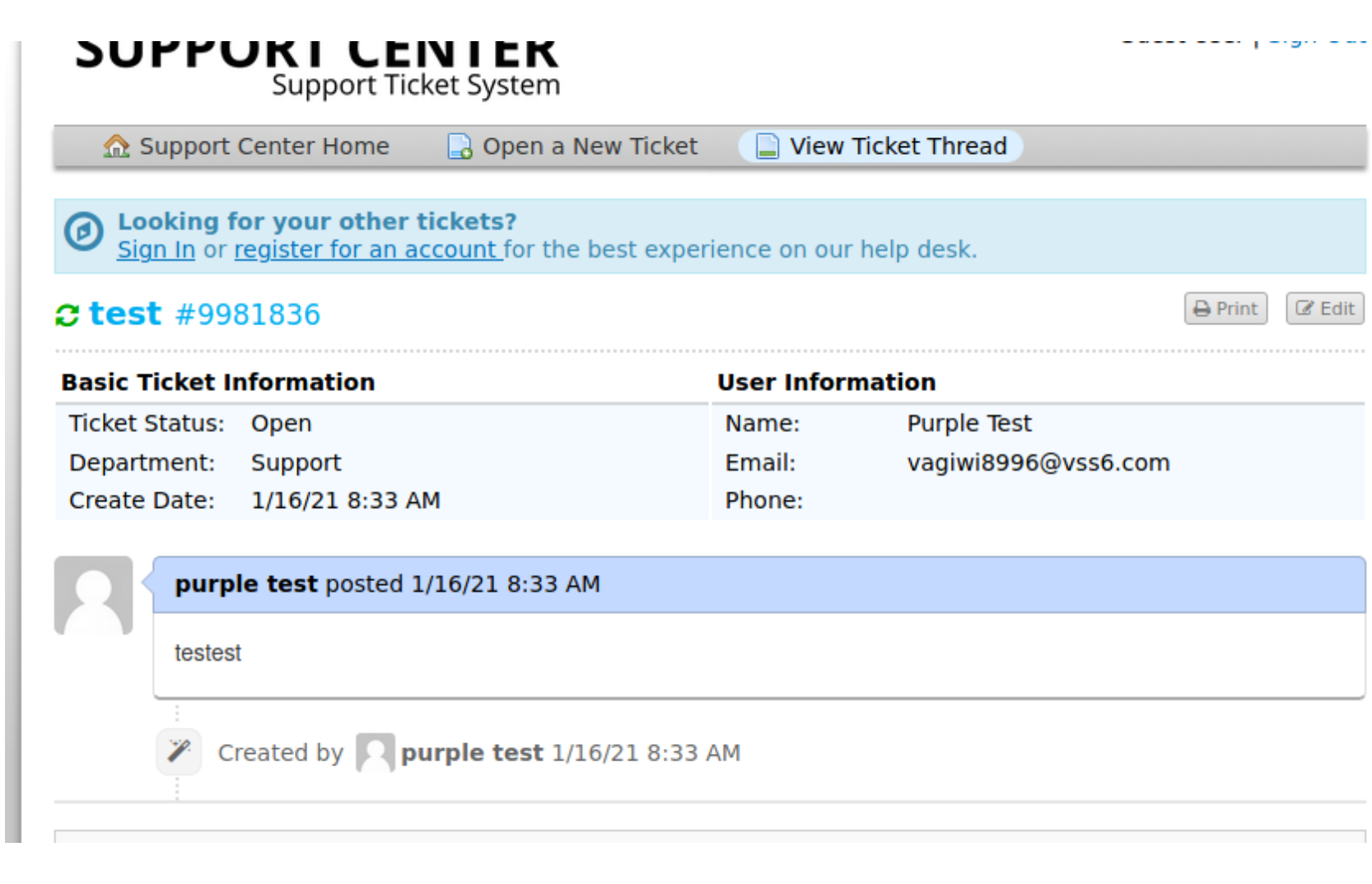


Once we click create, it gives us a ticket id and an email address:



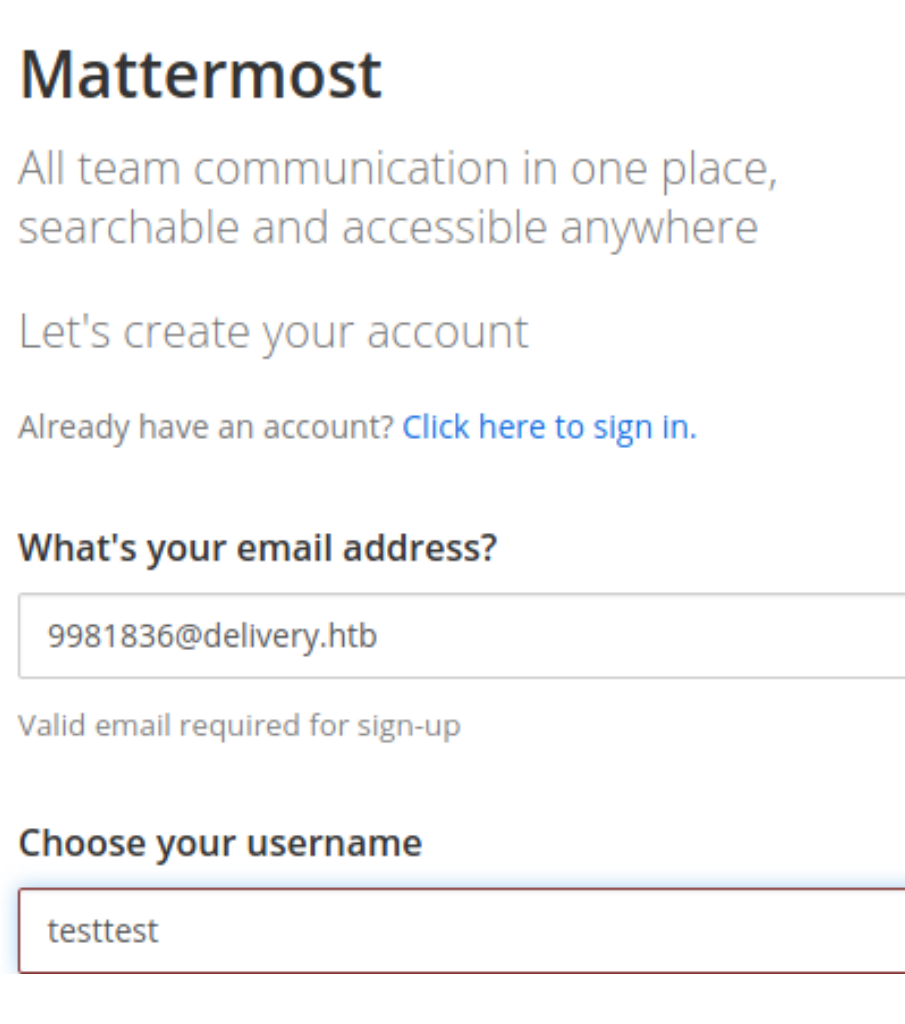
Check the ticket's status by putting the email address you gave and the ticket number.



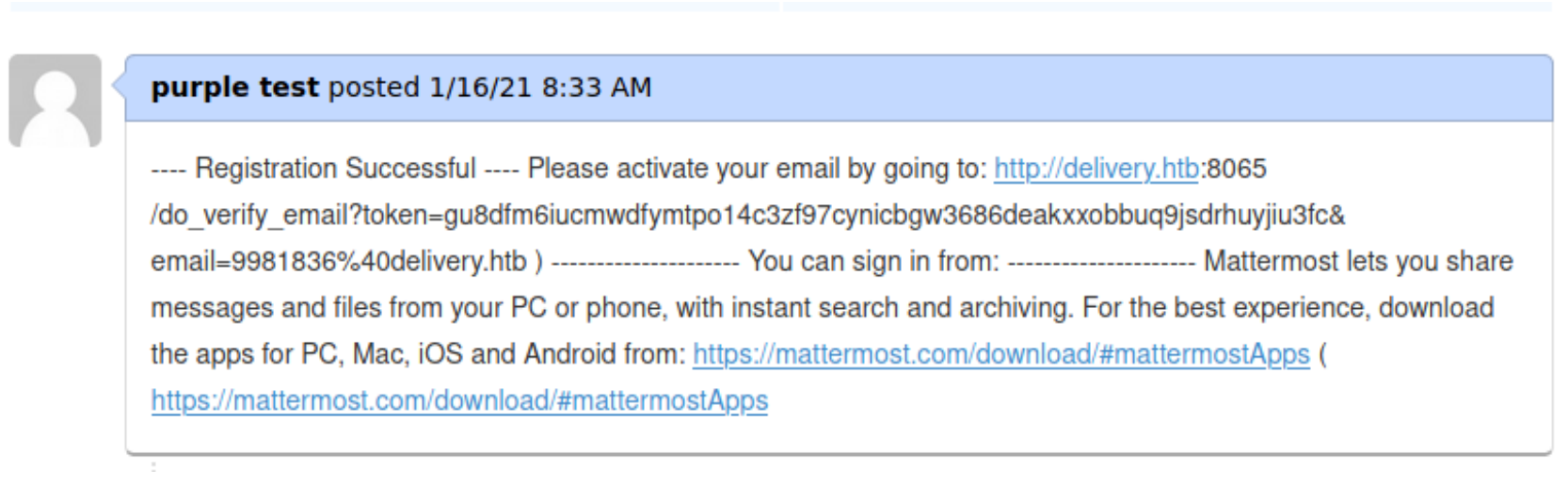


Port 8065

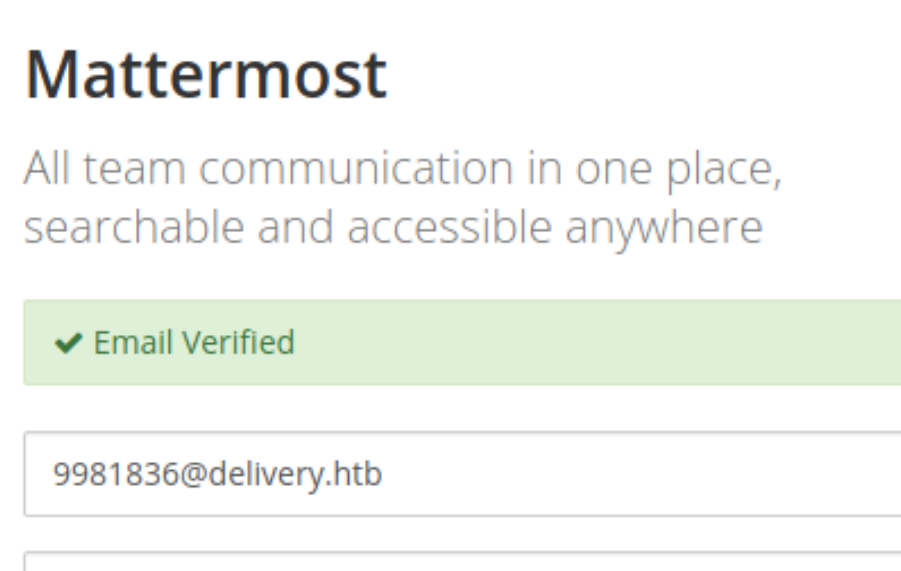
Now make your way to port **8065**. Create an account via the ticket number email address you were given, in this case: **9981836@delivery.htb**



Once you've made an account, go back to your open ticket on the **helpdesk** site. Click the **ticket thread**. It will give you a registration confimration from the port 8065 site we just visited:



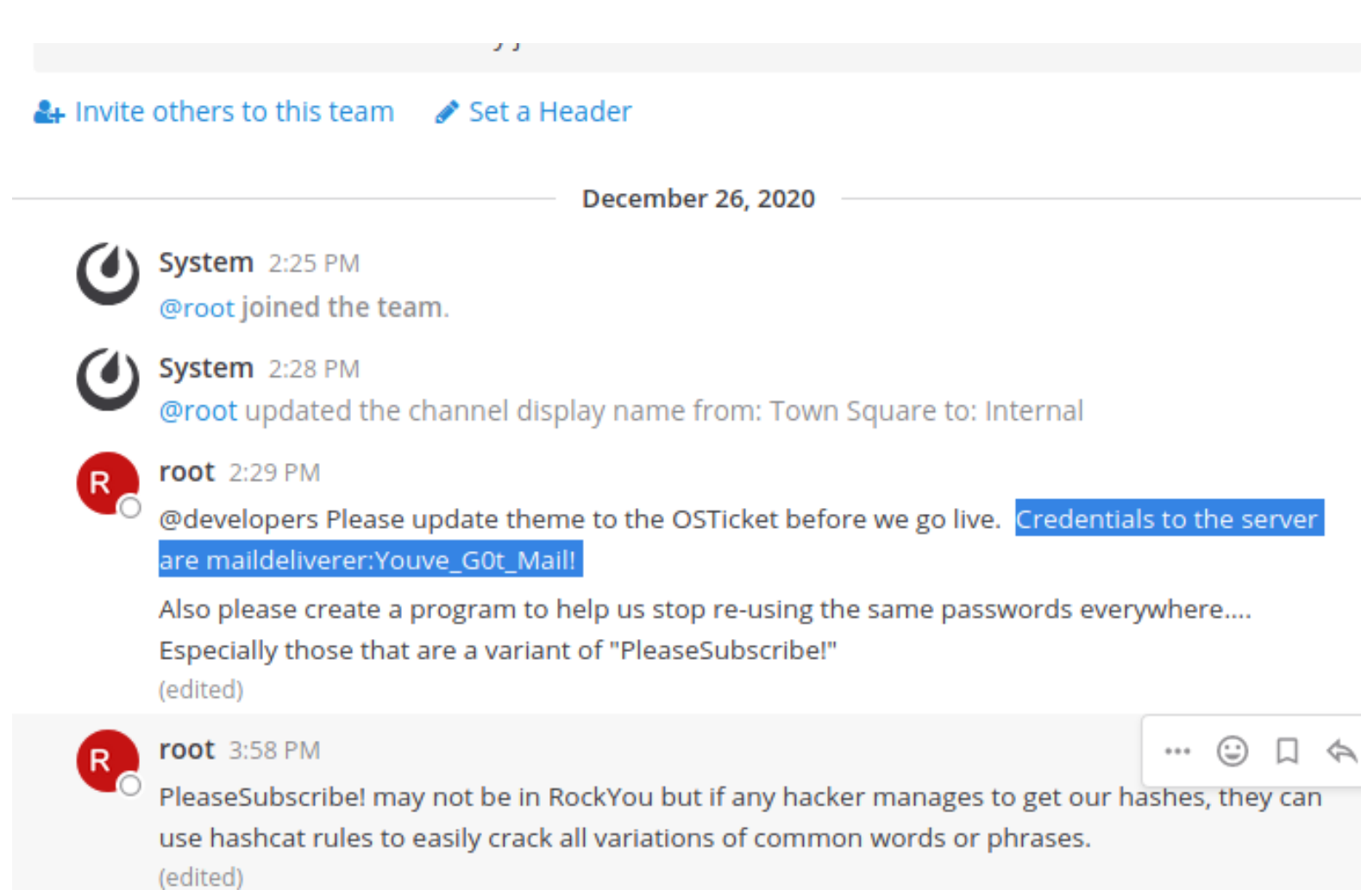
Copy the link into your URL, and hit enter and wait fifteen seconds or so until you get the confirmation.



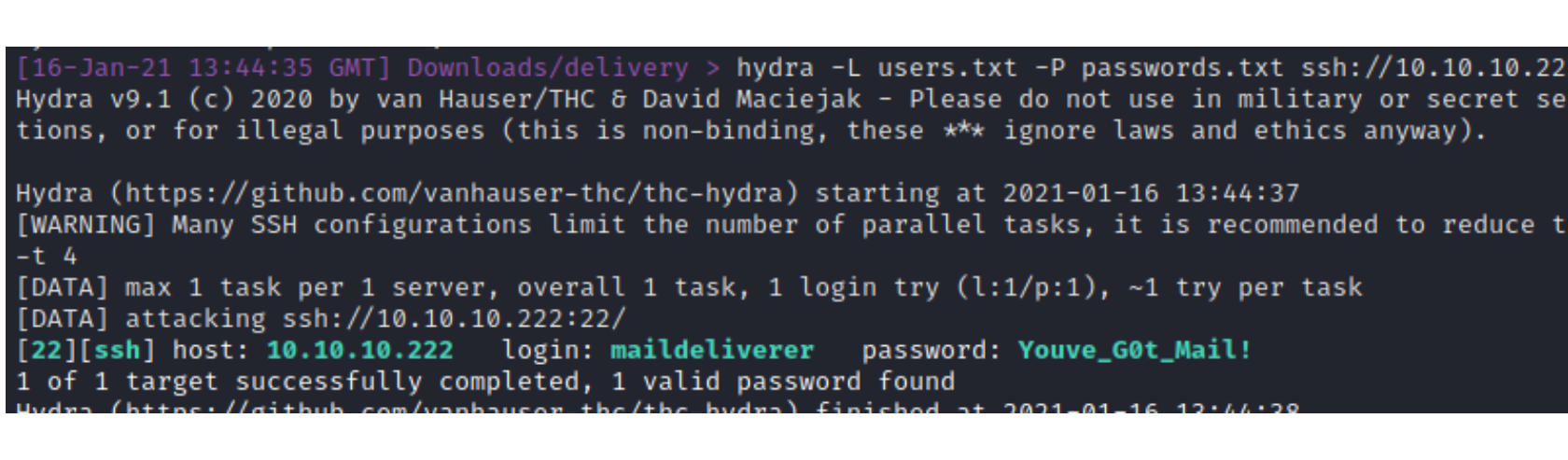
Sign in, and let's start looking around.

Mattermost

Once we enumerate through this authenticated web page, we quickly find some credentials pasted into the chat.



We can test the validity of these credentials through `hydra` ....and it confirms that we can login.



So let's **ssh** in the box:

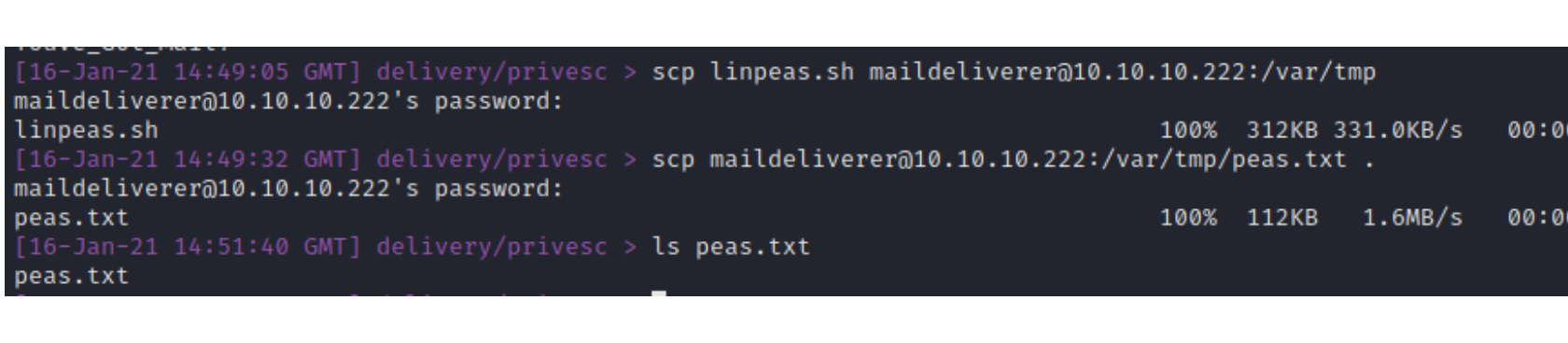
Maildeliverer Shell

We can retrieve the **user.txt** flag, and then focus on enumerating the box to **escalate** our **privileges**.

Linpeas (*fail*)

Let's run **linpeas:** https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/tree/master/linPEAS

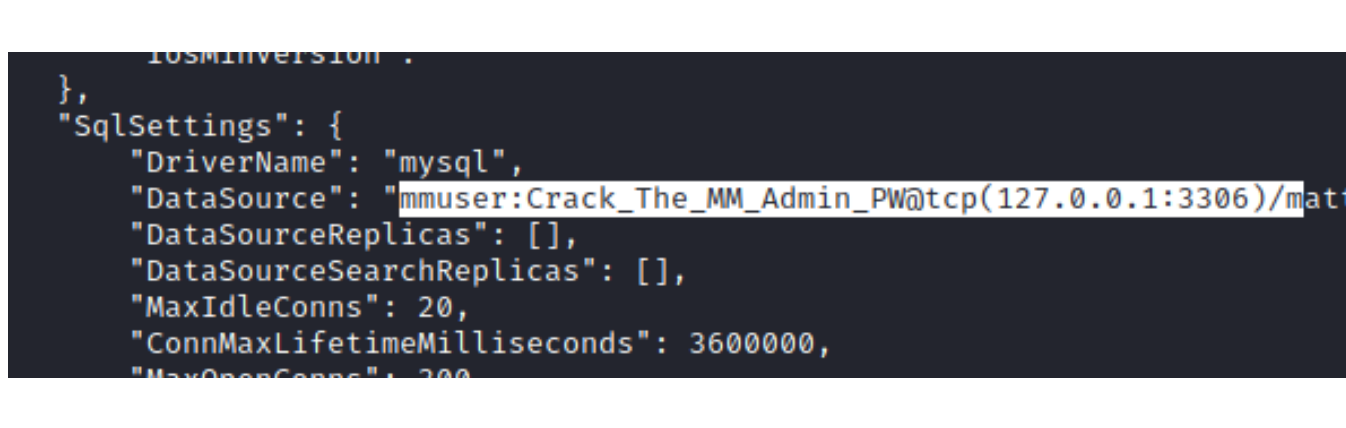
```
##transfer to victim box with password
scp linpeas.sh maildeliverer@10.10.10.222:/var/tmp

#run in victim shell
# -a offers a more thorough scan
bash linpeas.sh -a > peas.txt

##transfer from victim in Kali shell
scp maildeliverer@10.10.10.222:/var/tmp/peas.txt .
#the dot at the end puts peas.txt here in our kali directory
```



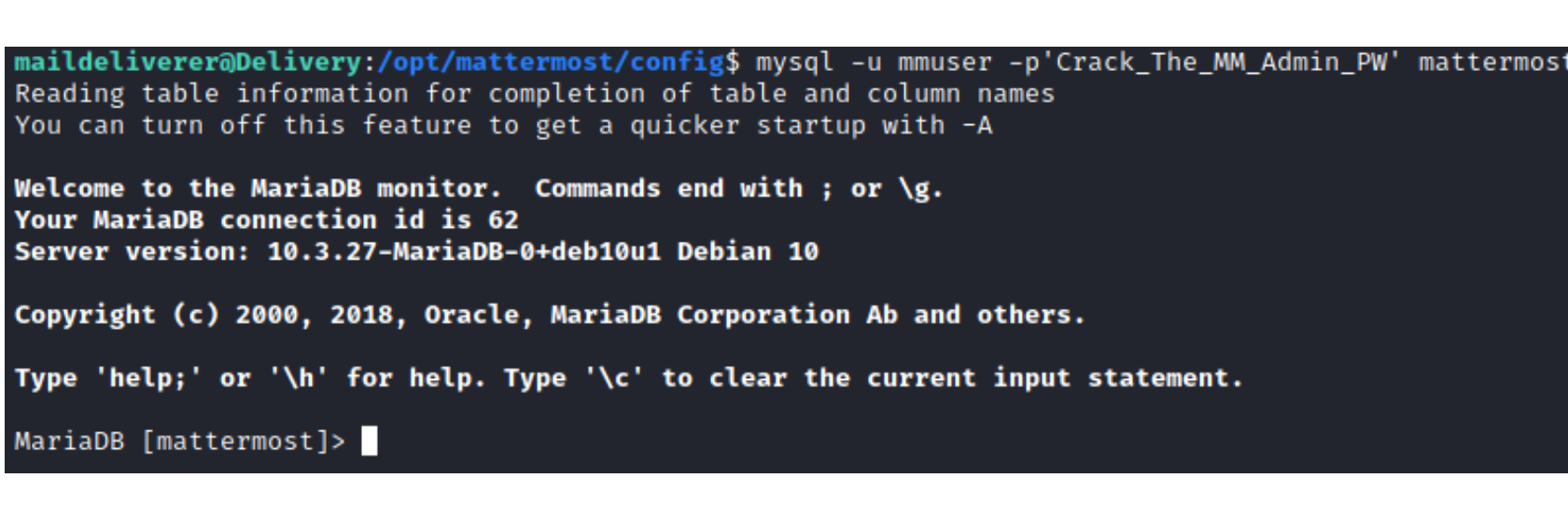
But I didn't find anything valuable from **linpeas**, so I manually enumerated the box.

Manual Hunt

If we take a look at **/opt/mattermost/config/config.json** we can find some **mysql creds**



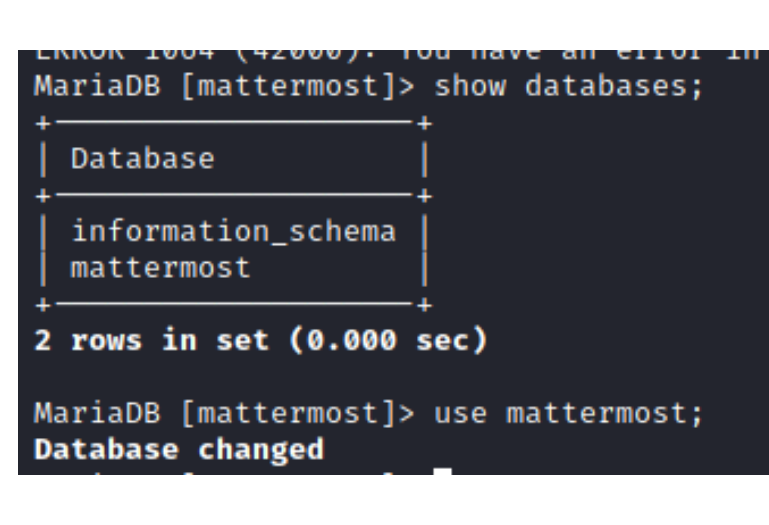
We can test these creds for **mysql:** `mysql -u mmuser -p'Crack_The_MM_Admin_PW' mattermost`

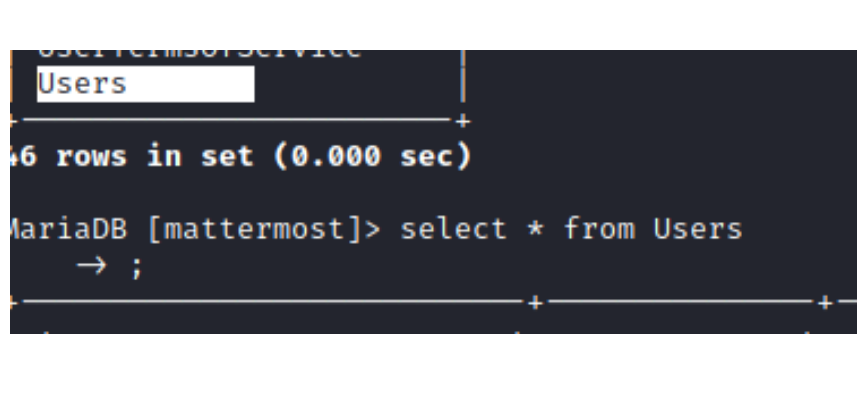


SQL db

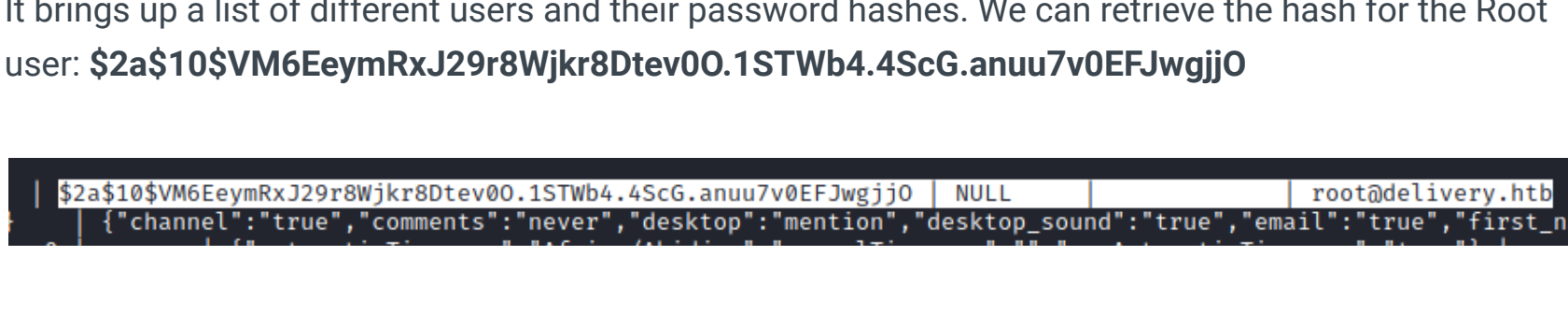
There are usually user's **hashed** passwords in an SQL databse. Let's hunt them down:

```
##show what databases are avaliable, and then choose mattermost
show databases;
use mattermost;

##then list tables, and select all from the Users table
show tables;
select * from Users
```



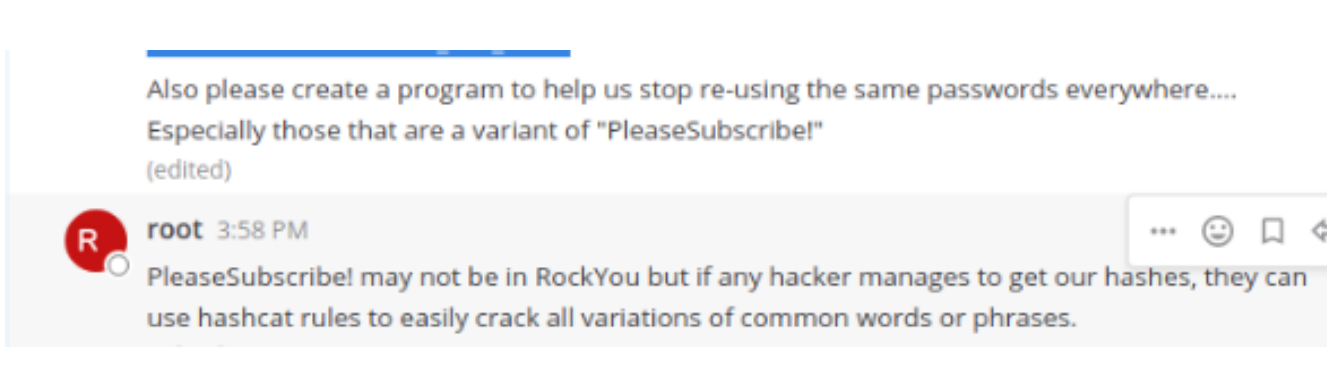


It brings up a list of different users and their password hashes. We can retrieve the hash for the Root user: **$2a$10$VM6EeymRxJ29r8Wjkr8Dtev0O.1STWb4.4ScG.anuu7v0EFJwgjjO**



Password Mutations

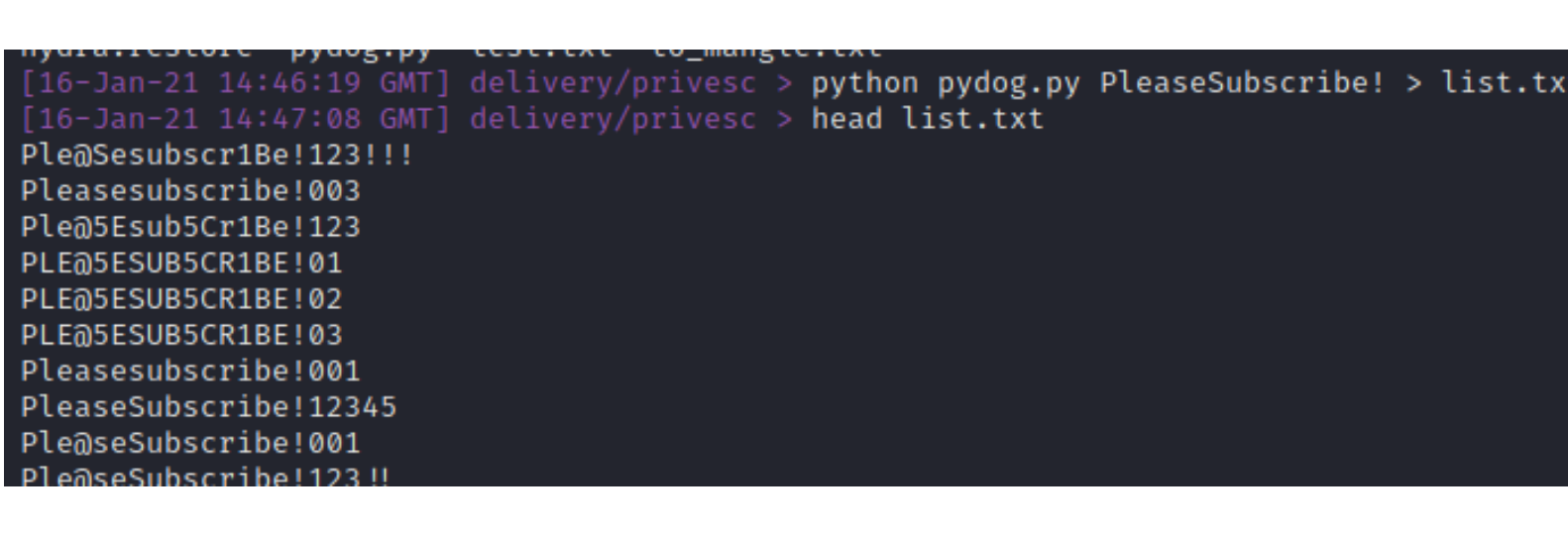
If we read the Chat on port 8065 again, what we can see is the **Root** user saying that the password `PleaseSubscribe!` can be **mutated / mangled** by an attacker to generate a list. This list can then be used to crack their hashed passwords, if we can find them....And of course, we just did find a **hash** buried deep in the **mysql** database



- By mangle or mutation, I mean that the original pass phrase is altered slightly iteratively until it has characters added or taken away
  - So for example, `password1` can be mutated to be `password!` or `p@assword1`
  - You can find more guidance on Hacktricks: https://book.hacktricks.xyz/brute-force

Mangling

I used this python tool to **mangle** my list: https://github.com/maudits/PyDog



As you can see, it creates all kinds of variations of the original phrase. We can see it into `mangled_list.txt` and also save the hash we retrieved into `hashes.txt`

Crack the Hash

What we now want to do is take this **mangled list**, and use it to **crack** the **hash** we just retrieved: `john --wordlist=mangled_list.txt hashes.txt`

And `john` is able to identify Root's password as: **PleaseSubscribe!21**

Root shell

We can't ssh into Root from our kali machine. But we can from the existing **maildeliverer** ssh shell we had