Secretos de PowerShell Remoting

Secrets of PowerShell Remoting (Spanish)

The DevOps Collective, Inc.

Este libro está a la venta en
http://leanpub.com/secrets-of-powershell-remoting-spanish

Esta versión se publicó en 2018-10-28

Este es un libro de Leanpub. Leanpub anima a los autores y publicadoras con el proceso de publicación. Lean Publishing es el acto de publicar un libro en progreso usando herramientas sencillas y muchas iteraciones para obtener feedback del lector hasta conseguir tener el libro adecuado.

También por The DevOps Collective, Inc.

Creating HTML Reports in Windows PowerShell

A Unix Person's Guide to PowerShell

The Big Book of PowerShell Error Handling

DevOps: The Ops Perspective

Ditch Excel: Making Historical and Trend Reports in PowerShell

Secrets of PowerShell Remoting

The Big Book of PowerShell Gotchas

The Monad Manifesto, Annotated

Why PowerShell?

Windows PowerShell Networking Guide

The PowerShell + DevOps Global Summit Manual for Summiteers

Why PowerShell? (Spanish)

DevOps: The Ops Perspective (Spanish)

The Monad Manifesto: Annotated (Spanish)

Creating HTML Reports in PowerShell (Spanish)

The Big Book of PowerShell Gotchas (Spanish)

The Big Book of PowerShell Error Handling (Spanish)

DevOps: WTF?

PowerShell.org: History of a Community

Índice general

Secretos de PowerShell Remoting **1**

Fundamentos de Remoting **3**
¿Qué es Remoting? 4
Examinando la arquitectura de Remoting 4
Habilitando Remoting 7
Entorno de pruebas 9
Primeros pasos con Remoting 10
Tareas "core" de Remoting 13
Remoting devuelve datos deserializados 18
Enter-PSSession vs. Invoke-Command 19

Acceso a equipos remotos **21**
Configuración de un HTTPS Listener 23
Autenticación de certificados 44
Modificación de la lista TrustedHosts 53
Conexión a través de dominios 57
Administradores de otros dominios 60
El segundo salto 61

Trabajar con Endpoints (también conocido como Configuraciones de Sesión) **67**
Conexión a un punto final diferente 67
Creación de un punto de extremo personalizado 69
Precauciones de seguridad con puntos finales personalizados 78

Diagnóstico y solución de problemas **81**
Ejemplos de diagnósticos 81
Metodología Estándar de Solución de Problemas 104
Resumen 106

Gestión de sesiones **107**
Sesiones Ad-Hoc vs. Persistentes 107
Desconexión y Reconexión de Sesiones 107
Opciones de Sesión 110

PowerShell, Remoting y la Seguridad **113**
Ni PowerShell ni Remoting son una "puerta trasera" para el Malware 113
Remoting no transmite ni almacena credenciales 115
Remoting utiliza el cifrado 116
Remoting es transparente para la seguridad 116
Remoting es una sobrecarga menor 117
Remoting utiliza autenticación mutua 117
Resumen 118

Configuración de Remoting mediante GPO **119**
Advertencias de GPO 119
Permitir la configuración automática de los escuchas (Listeners) de WinRM 120
Configuración del servicio WinRM para que se inicie automáticamente 121
Creación de una excepción de Firewall de Windows . . . 123
¡Darle una oportunidad! 125
Lo que no se puede hacer con una GPO 127

Secretos de PowerShell Remoting

Autor principal: Don Jones
Autor colaborador: Dr. Tobias Weltner
Contribuciones de: Dave Wyatt y Aleksandar Nikolik

Introducido en Windows PowerShell 2.0, Remoting es una de las tecnologías más útiles e importantes de PowerShell. Permite ejecutar casi cualquier comando que existe en un equipo remoto, abriendo un universo de posibilidades para la administración en masa y de forma remota. Remoting subyace otras tecnologías, incluyendo Workflow, Desired State Configuration, ciertos tipos de jobs en background y mucho más. Esta guía no pretende ser un documento completo de referencia, aunque sí busca proporcionar una buena introducción. En su lugar, esta guía está diseñada para documentar algunos pequeños detalles de configuración que no parecen estar documentados en otras partes.

¿Ha sido útil este libro? El (los) autor (es) le pide (n) que haga una donación deducible de impuestos (en los EE.UU., consulte sus

leyes si vive en otro lugar) de cualquier cantidad a The DevOps Collective[1] para apoyar su trabajo.

** Revise las actualizaciones! ** Nuestros ebooks se actualizan a menudo con contenido nuevo y corregido. Los hacemos disponibles de tres maneras:

- Nuestra rama principal GitHub organization[2], con un repositorio para cada libro. Visite https://github.com/devops-collective-inc/
- Nuestra GitBook page[3], donde puede navegar por los libros en línea, o descargarlos en formato PDF, EPUB o MOBI. Utilizando el lector en línea, puede saltar a capítulos específicos. Visite https://www.gitbook.com/@devopscollective
- En LeanPub[4], donde se pueden descargar como PDF, EPUB, o MOBI (login requerido), y "comprar" los libros haciendo una donación a DevOps. También puede elegir recibir notificaciones de actualizaciones. Visite https://leanpub.com/u/devopscollective

GitBook y LeanPub generan la salida del formato PDF ligeramente diferente, por lo que puede elegir el que prefiera. LeanPub también le puede notificar cada vez que liberamos alguna actualización. Nuestro repositorio de GitHub es el principal; los repositorios en otros sitios suelen ser sólo espejos utilizados para el proceso de publicación. GitBook normalmente contendrá nuestra última versión, incluyendo algunos bits no terminados; LeanPub siempre contiene la más reciente "publicación liberada" de cualquier libro.

[1] https://devopscollective.org/donate
[2] https://github.com/devops-collective-inc
[3] https://www.gitbook.com/@devopscollective
[4] https://leanpub.com/u/devopscollective

Fundamentos de Remoting

Windows PowerShell 2.0 introdujo una potente tecnología, Remoting, refinada y ampliada en PowerShell 3.0. Basada principalmente en protocolos y técnicas estandarizadas, el sistema de Remoting es posiblemente uno de los aspectos más importantes de PowerShell: los futuros productos de Microsoft se basarán en él casi en su totalidad para las comunicaciones administrativas a través de una red.

Desafortunadamente, Remoting es también un sistema complejo de componentes, y mientras que Microsoft ha intentado proporcionar la dirección sólida para usarla en una variedad de escenarios, muchos administradores todavía luchan con esta. Este "mini e-book" está diseñado para ayudarle a entender mejor lo que es el Remoting, cómo funciona y, lo que es más importante, cómo usarlo en una variedad de situaciones diferentes.

Nota Tenga en cuenta que esta guía no pretende reemplazar la gran variedad de libros existentes que cubren los fundamentos de Remoting, como el propio *Learn Windows PowerShell in a Month of Lunches* (http://morelunches.com) de Don Jones o *PowerShell in Depth*. En su lugar, esta guía complementa a aquellas que proporcionan instrucciones paso a paso para muchos de los escenarios "alrededor" de un sistema de comunicación remota, e intenta explicar algunos de los comportamientos y requerimientos de los sistemas remotos más inusuales.

¿Qué es Remoting?

En esencia, el acceso remoto le permite acceder a máquinas remotas a través de una red y recuperar datos o ejecutar código en una o varias computadoras remotas. Esto no es una idea nueva. Ya en el pasado una serie de diferentes tecnologías remotas han intentado lo mismo. Algunos Cmdlets de PowerShell han proporcionado tradicionalmente capacidades propias de acceso remoto limitadas, mientras que la mayoría de los Cmdlets no admiten la conexión remota por su propia cuenta.

Con PowerShell Remoting se encuentra finalmente un entorno genérico que permite la ejecución remota para, literalmente, cualquier comando que se puede ejecutar en una instanica de PowerShell de forma local. Por lo que en lugar de agregar capacidades de acceso remoto a cada Cmdlet y/o aplicación, simplemente se deja a PowerShell transferir la ejecución de su código al equipo de destino y a continuación, enviar los resultados de vuelta.

A lo largo de este libro nos centraremos en el control remoto de PowerShell pero no cubriremos las funciones remotas privadas no estándar incorporadas en algunos Cmdlets seleccionados .

Examinando la arquitectura de Remoting

Como se muestra en la figura 1.1, la arquitectura remota genérica de PowerShell se compone de numerosos componentes y elementos diferentes e interrelacionados.

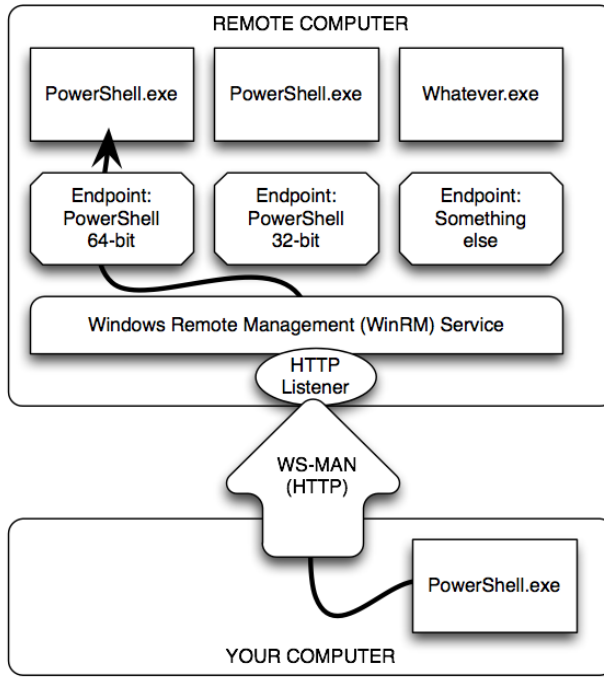

image003.png

Figura 1.1: Los elementos y componentes de PowerShell Remoting

Aquí está la lista completa:

- En la parte inferior de la figura está su computadora, o más correctamente su cliente. Es donde usted se sienta físicamente, y donde iniciará la mayor parte de sus actividades de control remoto.
- Su computadora se comunicará a través de WS-MAN, o del protocolo de servicios web para la administración. Este es un protocolo basado en http(s) que puede encapsular una variedad de tipos de comunicación. Hemos ilustrado el uso de

http, que es la configuración predeterminada, pero también podría ser fácilmente https
- En el equipo remoto, en la terminología adecuada, el servidor (que no hace referencia al sistema operativo), ejecuta el servicio de administración remota de Windows (WinRM). Este servicio está configurado para tener uno o más oyentes. Cada oyente espera el tráfico entrante de WS-MAN en un puerto específico, cada uno ligado a un protocolo específico (http o https), y en direcciones IP específicas (o todas las direcciones locales)
- Cuando un oyente recibe tráfico, el servicio WinRM busca el EndPoint a donde se debe enviar el tráfico. Para nuestro propósito, un EndPoint usualmente estará asociado con una instancia de Windows PowerShell. En términos de PowerShell, un EndPoint también se denomina una configuración de sesión. Esto se debe a que además de lanzar PowerShell, se pueden cargar secuencias de comandos y módulos, agregar restricciones sobre lo que puede hacer el usuario conectado y aplicar configuraciones adicionales de sesión específicas que no se mencionan aquí.

Nota Aunque mostramos **powershell.exe** en nuestro diagrama, eso solo para propósitos de ilustración. **Powershell.exe** es la aplicación de consola de PowerShell, y no tendría sentido tener esta ejecución como un proceso de fondo en un equipo remoto. El proceso real se denomina **wsmprovhost.exe**, que aloja PowerShell en segundo plano para conexiones remotas.

Como se puede ver, un único equipo remoto puede tener fácilmente decenas o incluso cientos de EndPoints, cada uno con una configuración diferente. PowerShell 3.0 configura tres EndPoints por defecto: uno para PowerShell de 32 bits (en sistemas de 64 bits), un EndPoint de PowerShell por defecto (que es de 64 bits en sistemas x64) y otro para PowerShell Workflow. Comenzando con Windows Server 2008 R2, hay un cuarto EndPoint predeterminado para las tareas de Server Manager Workflow.

Habilitando Remoting

La mayoría de las versiones cliente de Windows, iniciando con Windows Vista, no habilitan las conexiones remotas entrantes de forma predeterminada, aunque las versiones de servidor más recientes de Windows vienen con Remoting habilitado. El primer paso con Remoting suele ser habilitarlo en los equipos en que se desean recibir conexiones entrantes. Hay tres maneras de habilitar Remoting. La tabla 1.1 compara lo que se puede lograr con cada una de ellas.

Tabla 1.1 comparando las maneras de habilitar Remoting

	Enable-PSRemoting	**Política de grupo**	**Manualmente paso a paso**
Establecer WinRM para auto-iniciar e iniciar el servicio	Si	Si	Si - utilice **Set-Service** y **Start-Service**.
Configurar el detector de HTTP	Si	Puede configurar el registro automático de Listeners, sin crear Listeners personali-zados	Si - Utilice la utilidad de línea de comandos WSMAN y la unidad **WSMAN:** de PowerShell

	Enable-PSRemoting	**Política de grupo**	**Manualmente paso a paso**
Configurar el detector de HTTPS	No	No	Si - Utilice la utilidad de línea de comandos WSMAN y la unidad **WSMAN:** de PowerShell
Configurar EndPoints / Configurar sesiones	Si	No	Si - utilice el Cmdlet PSSession-Configuration
Configurar la excepción de Firewall de Windows	Si*	Si*	Si* - Utilice Cmdlets del Firewall o la GUI del Firewall de Windows

Nota Tenga en cuenta que las versiones existentes de cliente de Windows, como Windows Vista, no permiten excepciones de Firewall en una red identificada como "pública". Las redes deben ser "casa" o "trabajo/dominio" para permitir excepciones. En PowerShell 3.0, se puede ejecutar **Enable-PSRemoting** con el modificador **-SkipNetworkProfileCheck** para evitar este problema..

Estaremos habilitando la administración remota en nuestro entorno de prueba ejecutando **Enable-PSRemoting**. Es rápido, fácil e incluye todo lo necesario. También vera una gran cantidad de tareas manuales a realizar en las siguientes secciones.

Entorno de pruebas

Usaremos un entorno de pruebas consistente en las siguientes secciones. Consiste en seis máquinas virtuales en *cloudshare.com* configuradas como se muestra en la figura 1.2.

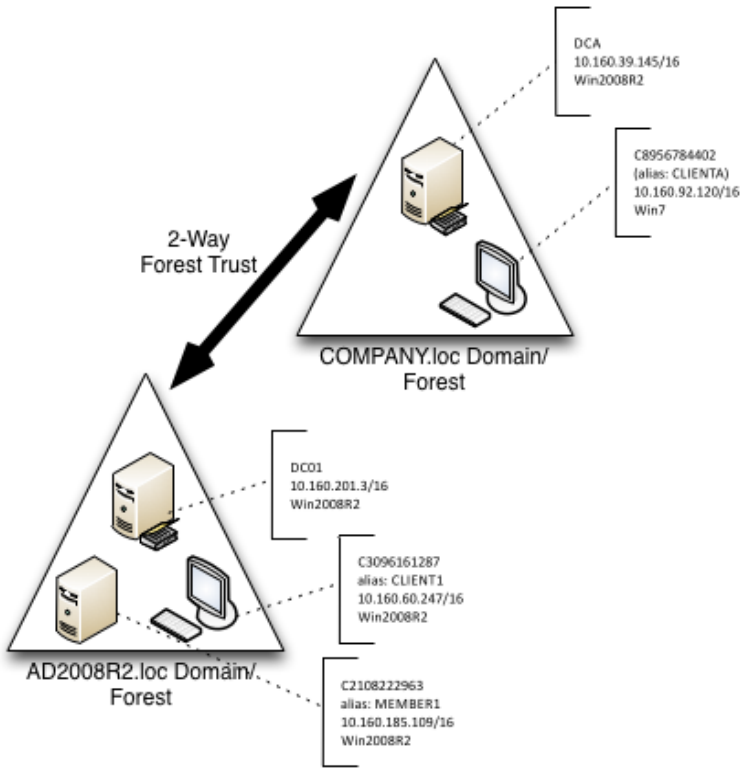

image004.png

Figura 1.2: configuración del entorno de pruebas

Algunas notas importantes:

- .NET Framework v4 y PowerShell 3.0 están instalados en todos los equipos. La mayor parte de lo que cubriremos

también se aplica a PowerShell 2.0.

- Como se muestra, la mayoría de las computadoras tienen un nombre de computadora numérico (c2108222963, y así sucesivamente); El controlador de dominio para cada dominio (que también es un servidor DNS) tiene registros CNAME con nombres más fáciles de recordar.
- Cada controlador de dominio tiene un reenviador condicional configurado para el otro dominio, de modo que las máquinas de cualquiera de los dominios puedan resolver nombres de equipos en el otro dominio.
- Realizamos todas las tareas como miembro del grupo de administradores del dominio, a menos que se indique lo contrario.
- Creamos un sexto servidor completamente independiente que no está en ningún dominio. Esto será útil para cubrir algunas de las situaciones que no son de dominio con las que puede encontrarse en un sistema de comunicación remota.

Tenga cuidado al abrir PowerShell en un equipo que tenga habilitado el control de cuenta de usuario (UAC), asegúrese de hacer clic con el botón derecho en el icono de PowerShell y seleccione "Ejecutar como administrador". Si la barra de título de la ventana PowerShell resultante no comienza con la palabra Administrador: entonces no tiene privilegios administrativos. Puede comprobar los permisos de forma programática con esto *(whoami /all | select-string S-1-16-12288) -ne $null* en una consola de PowerShell. En un Shell con permisos de administrador se devuelve **True**, de lo contrario será **False**.

Primeros pasos con Remoting

Comenzamos ejecutando Enable-PSRemoting en las seis computadoras. Debemos asegurarnos que el comando finaliza sin errores.

Cualquier error en este punto es una señal para se detenga y resuelva el error antes de intentar continuar. La figura 1.3 muestra la salida esperada.

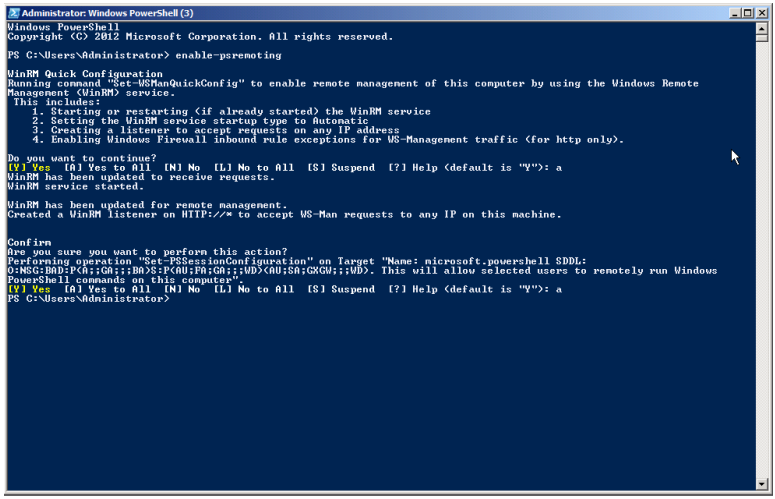

image005.png

Figura 1.3: salida esperada de Enable-PSRemoting

Nota: Observara un uso desmedido de capturas de pantalla a lo largo de esta guía. Me permiten asegurar que no cometo errores ortográficos o errores del tipo copiar/pegar. Verá exactamente lo que escribimos y los resultados de su ejecución.

Ejecutar Get-PSSessionConfiguration debe revelar los tres o cuatro EndPoints creados por Enable-PSRemoting. La figura 1.4 muestra la salida esperada en uno de los servidores.

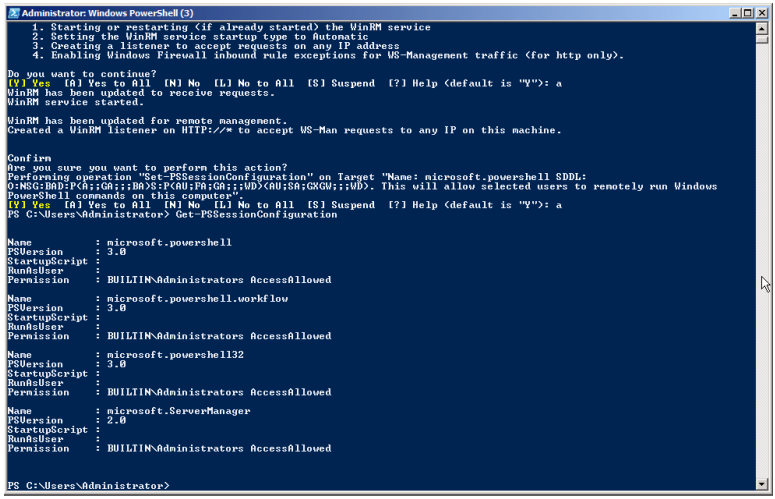

image006.png

Figura 1.4: Salida esperada de Get-PSSessionConfiguration

Nota: la figura 1.4 ilustra que se puede esperar que diferentes EndPoints se configuren en diferentes máquinas. Este ejemplo fue con un servidor Windows 2008 R2 equipo, que tiene menos EndPoints que una máquina con Windows 2012.

Vale la pena tomar un momento para comprobar rápidamente la configuración de Remoting. Para los equipos que forman parte del mismo dominio, al iniciar sesión como administrador de dominio de ese dominio, el sistema de comunicación remota debería "funcionar". Compruébelo rápidamente conectarse de una computadora a otra usando Enter-PSSession.

Nota: en otros entornos, es posible que una cuenta de administrador de dominio no sea la única que pueda usar Remoting. Si en su hogar o entorno de trabajo se tienen cuentas adicionales en el grupo de administradores locales como estándar en su dominio, también podrá utilizar esas cuentas para realizar llamadas remotas.

La figura 1.5 muestra la salida esperada, en la que también ejecutamos un comando dir rápido y luego salimos de la sesión remota.

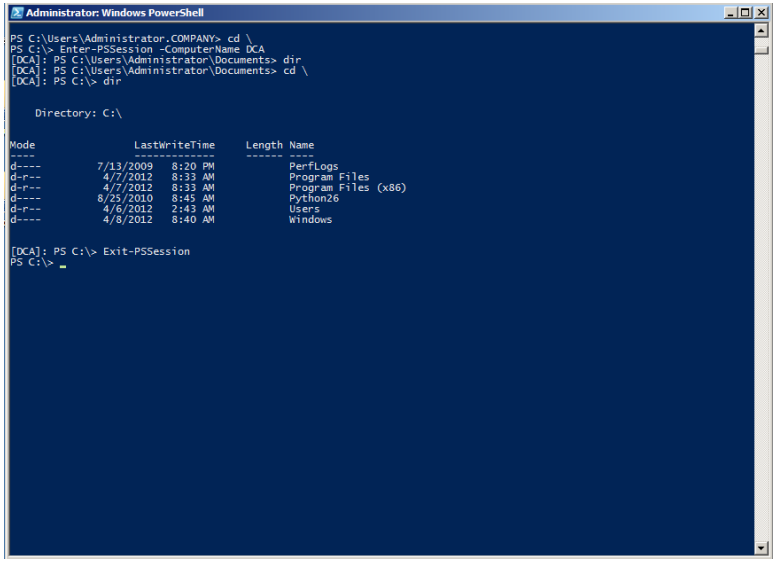

image007.png

Figura 1.5: comprobación de la conectividad remota desde el cliente al controlador de dominio DCA.

Precaución: si está configurando su propio entorno de pruebas, no continúe hasta que haya confirmado la conectividad de los sistemas remotos entre dos equipos del mismo dominio. Por ahora no necesitamos comprobar otros escenarios.

Tareas “core” de Remoting

PowerShell proporciona dos escenarios principales para el uso de Remoting. El primero, Remoting 1-a-1, es similar en su naturaleza al shell SSH disponible en sistemas UNIX y Linux. Con él, obtendrá acceso a una línea de comandos en un único equipo remoto. El segundo, Remoting 1-a-muchos, permite enviar un comando (o una lista de comandos) en paralelo, a un conjunto de equipos

remotos. Hay otro par de técnicas secundarias útiles que veremos más adelante.

Remoting 1-a-1

El comando Enter-PSSession se conecta a un equipo remoto y permite el acceso a una línea de comandos en ese equipo. Puede ejecutar cualquier comando en dicho equipo, siempre y cuando tenga permiso para realizar esa tarea. Tenga en cuenta que no está creando un inicio de sesión interactivo. La conexión se auditará como un inicio de sesión de red, al igual que si se estuviera conectando al recurso compartido administrativo C$ de la computadora. PowerShell no cargará ni procesará las secuencias de comandos del perfil de usuario en el equipo remoto. Cualquier script que elija ejecutar (y esto incluye la importación de módulos de script) sólo funcionará si la política de ejecución de la máquina remota lo permite.

```
1 Enter-PSSession -computerName DC01
```

Nota: Mientras esté conectado a una máquina remota a través de Enter-PSSession, el "prompt" de la línea de comandos cambia y muestra el nombre del sistema remoto al que está conectado entre corchetes. Si ha personalizado su "prompt", todas esas personalizaciones se perderán porque el "prompt" se creará en el sistema remoto y se transferirá de regreso a usted. Todas las entradas de teclado se envían a la máquina remota, y todos los resultados son devueltos a usted. Es importante tener en cuenta esto porque no puede utilizar Enter-PSSession en un script. Si lo hace, el script seguiría ejecutándose en su máquina local, ya que no se ingresó ningún código (en el teclado) de forma interactiva.

Remoting 1-a-muchos

Con esta técnica, se especifican uno o más nombres de equipo y un comando (o una lista de comandos separados por punto y coma). PowerShell envía los comandos, a través de Remoting, a los equipos especificados. Esas computadoras ejecutan los comandos, serializan los resultados en XML y transmiten los resultados de vuelta. El equipo deserializa el XML de nuevo en objetos y los coloca en la canalización (pipeline) de la sesión de PowerShell. Esto se logra a través del Cmdlet Invoke-Command.

```
1  Invoke-Command -computername DC01,CLIENT1 -scriptBlock { \
2  Get-Service }
```

Si tiene una secuencia de comandos para ejecutar, puede hacer que Invoke-Command la lea, transmita el contenido a los equipos remotos y haga que se ejecuten dichos comandos.

```
1  Invoke-Command -computername DC01,CLIENT1 -filePath c:\Sc\
2  ripts\Task.ps1
```

Tenga en cuenta que Invoke-Command, de forma predeterminada, se comunicará con hasta 32 equipos a la vez. Si especifica más, los equipos extra se pondrán en cola e Invoke-Command comenzará a procesarlos al terminar los primeros 32. El parámetro -ThrottleLimit puede aumentar este límite. El único costo es para su computadora, ya que debe tener recursos suficientes para mantener una sesión única de PowerShell para cada equipo al que esté contactando simultáneamente. Si espera recibir grandes cantidades de datos de los equipos remotos, el ancho de banda de red disponible puede ser otro factor limitante.

Sesiones

Cuando ejecuta Enter-PSSession o Invoke-Command y utiliza el parámetro -ComputerName, Remoting crea una conexión (o sesión), hace lo que se le pidió y luego cierra la conexión (en el caso de una sesión interactiva creada con Enter-PSSession, PowerShell sabe que ha terminado cuando ejecuta Exit-PSSession). Hay algunas sobrecargas involucradas en esa configuración y arranque, por lo que PowerShell también ofrece la opción de crear una conexión persistente, llamada PSSession. Se ejecuta New-PSSession para crear una sesión nueva y persistente. Entonces, en lugar de usar -ComputerName con Enter-PSSession o Invoke-Command, utilice su parámetro -Session y pase un objeto PSSession existente y abierto. Esto permite a los comandos volver a utilizar la conexión persistente que se había creado anteriormente.

Cuando utiliza el parámetro -ComputerName y trabaja con sesiones "ad hoc", cada vez que envía un comando a una máquina remota, hay un retraso significativo causado por la sobrecarga que se tarda en crear una nueva sesión. Como cada llamada a Enter-PSSession o Invoke-Command configura una nueva sesión, tampoco se puede conservar el estado. En el ejemplo siguiente, la variable $test se pierde en la segunda llamada:

```
PS> Invoke-Command -computername CLIENT1 -scriptBlock { $\
test = 1 }
PS> Invoke-Command -computername CLIENT1 -scriptBlock { $\
test }
PS>
```

Cuando se utilizan sesiones persistentes, por otro lado, las re-conexiones son mucho más rápidas, y puesto que se están manteniendo y reutilizando las sesiones, se conservará el estado. Así que aquí, en la segunda llamada a Invoke-Command todavía podrá acceder a la variable $test que se configuró en la primera llamada.

```
PS> $Session = New-PSSession -ComputerName CLIENT1
PS> Invoke-Command -Session $Session -scriptBlock { $test\
 = 1 }
PS> Invoke-Command -Session $Session -scriptBlock { $test\
 }
1
PS> Remove-PSSession -Session $Session
```

Existen otros comandos para verificar el estado de la sesión y recuperar sesiones (Get-PSSession), cerrarlos (Remove-PSSession), desconectar y volver a conectarlos (Disconnect-PSSession y Reconnect-PSSession, agregados en PowerShell v3), etc. En PowerShell v3, también puede pasar una sesión abierta a Get-Module e Import-Module, lo que le permite ver los módulos listados en una computadora remota (a través de la PSSession abierta) o importar un módulo desde una computadora remota a su computadora. Revise la ayuda de esos comandos para obtener más información.

Nota: Una vez que utilice New-PSSession y cree sus propias sesiones persistentes, es su responsabilidad "hacer el trabajo" y luego cerrar la sesión cuando haya terminado. Hasta que lo haga, las sesiones persistentes permanecen activas, consumen recursos y pueden impedir que otros se conecten. De forma predeterminada, sólo se permiten 10 conexiones simultáneas a una máquina remota. Si mantiene demasiadas sesiones activas, se encontrará fácilmente al borde de los límites de recursos. Esta línea muestra lo que sucede si intenta configurar demasiadas sesiones simultáneas:

```
PS> 1..10 | Foreach-Object { New-PSSession -ComputerName \
CLIENT1 }
```

Remoting devuelve datos deserializados

Los resultados que recibe de una computadora remota se han serializado en XML y luego se han deserializado en su computadora. En esencia, los objetos colocados en el pipeline de su shell son instantáneas estáticas y separadas de lo que estaba en el equipo remoto en el momento en que se completó el comando. Estos objetos deserializados carecen de los métodos de los objetos originales, y en cambio solo ofrecen propiedades estáticas.

Si necesita acceder a métodos o cambiar propiedades, o en otras palabras, si debe trabajar con los objetos en vivo, asegúrese de hacerlo en el lado remoto, antes de que los objetos se serialicen y regresen al llamador. Este ejemplo utiliza métodos de objeto en el lado remoto para determinar los propietarios de un proceso que funciona bien:

```
PS> Invoke-Command -ComputerName CLIENT1 -scriptBlock { G\
et-WmiObject -Class Win32_Process | Select-Object Name, {\
 $_.GetOwner().User } }
```

Una vez que los resultados vuelven a usted, ya no puede invocar métodos de objetos porque ahora trabaja con objetos "rehidratados", diferentes a los "objetos vivos" por lo que ahora ya no pueden acceder a sus métodos:

```
PS> Invoke-Command -ComputerName CLIENT1 -scriptBlock { G\
et-WmiObject -Class Win32_Process } | Select-Object Name,\
 { $_.GetOwner().User }
```

Serializar y deserializar es relativamente costoso. Puede optimizar la velocidad y los recursos asegurándose de que su código remoto emita sólo los datos que realmente necesita. Por ejemplo, puede

utilizar Select-Object y seleccionar cuidadosamente las propiedades que desea volver en lugar de serializar y deserializar todo.

Enter-PSSession vs. Invoke-Command

Muchos de los recién llegados pueden confundirse un poco acerca de la comunicación remota, en parte debido a cómo PowerShell ejecuta los scripts. Tenga en cuenta lo siguiente y asuma que SERVER2 contiene una secuencia de comandos denominada C:RemoteTest.ps1:

```
1 Enter-PSSession -ComputerName SERVER2
2 C:\RemoteTest.ps1
```

Si se sentara y escribiera estos comandos de forma interactiva en la ventana de comandos de su equipo cliente, funcionaría (suponiendo que se configurara el sistema, tuvieran permisos y todo eso). Sin embargo, si los pegó en un script y ejecutó ese script, no funcionaría. El script intentaría ejecutar C:RemoteTest.ps1 *en su equipo local.*

El resultado práctico de esto es que Enter-PSSession *está realmente destinado a un uso interactivo por parte de un ser humano, y no a un uso por lotes de un script. Si desea enviar un comando a una computadora remota, desde dentro de una secuencia de comandos, Invoke-Command es la forma correcta de hacerlo.* Puede configurar una sesión de antemano (útil si va a enviar más de un comando) o puede utilizar un nombre de equipo si sólo desea enviar un solo comando. Por ejemplo:

```
$session = New-PSSession -ComputerName SERVER2
Invoke-Command -session $session -ScriptBlock { C:\Remote\
Test.ps1 }
```

Obviamente, tendrá que tener un poco de precaución. Si esas eran las dos únicas líneas en el script, entonces cuando el script termine de ejecutarse, $session dejaría de existir. Eso podría desconectar (en cierto sentido) de la sesión que se ejecuta en SERVER2. Los resultados dependen mucho de lo que hace y de cómo lo hace. En este ejemplo, todo estaría bien, porque Invoke-Command "mantendría" el script local en ejecución hasta que el script remoto terminara y devolviera su salida (si la hubiera).

Acceso a equipos remotos

Principalmente existen dos escenarios al acceder una computadora remota. La diferencia entre estos escenarios radica especialmente en la respuesta a una pregunta: ¿Puede WinRM identificar y autenticar la máquina remota?

Obviamente, la máquina remota necesita saber quién es usted, porque estará ejecutando comandos en su nombre. Pero usted necesita saber quién es, también. Esta autenticación mutua, es un paso de seguridad importante. Significa que cuando escribe SERVER2, se está conectando realmente con el SERVER2 real, y no con alguna máquina haciéndose pasar por SERVER2. Mucha gente ha publicado artículos de blog sobre cómo deshabilitar las verificaciones de autenticación. Hacerlo, hace que Remoting " funcione" y se deshaga de los molestos mensajes de error, pero abre brechas de seguridad y hace posible que alguien "secuestre" o "falsifique" su conexión y potencialmente capture información confidencial como sus credenciales .

Precaución: Tenga en cuenta que Remoting implica delegar una credencial en el equipo remoto. Usted está haciendo algo más que simplemente enviar un nombre de usuario y una contraseña (que en realidad no ocurre todo el tiempo). Está dando a la máquina remota la capacidad de ejecutar las tareas como si estuviera allí ejecutándolas usted mismo. Un impostor podría hacer mucho daño con ese poder. Es por eso que Remoting se enfoca en la autenticación mutua, para que los impostores no tengan esa oportunidad.

En los escenarios de Remoting más sencillos, usted se conecta a una máquina que está en el mismo dominio de AD utilizando su nombre de equipo real, tal como está registrado en AD. AD maneja

la autenticación mutua y todo funciona de maravilla. Pero las cosas se pueden poner un poco más difíciles en otros escenarios:

- Conectar a una máquina en otro dominio
- Conectar a una máquina que no está en un dominio en absoluto
- Conectarse a través de un alias de DNS, o a través de una dirección IP, en lugar de a través del nombre de equipo real de la máquina como está registrado con AD

En estos casos, AD no puede hacer la autenticación mutua, por lo que tendrá que hacerlo usted mismo. En este punto tiene dos opciones:

- Configurar la máquina remota para aceptar conexiones HTTPS (en lugar de HTTP) y equiparla con un certificado SSL. El Certificado SSL debe ser emitido por una Autoridad de Certificación (CA) en la que confíe la máquina. Esto permite que el certificado SSL proporcione la autenticación mutua que WinRM usara luego.
- O, agregar el nombre de la máquina remota (lo que esté especificando, ya sea un nombre de equipo real, una dirección IP o un alias CNAME) a la lista de WinRM TrustedHosts de su equipo local. Tenga en cuenta que esto básicamente inhabilita la autenticación mutua, ya que permite a WinRM conectarse con ese identificador (nombre, dirección IP o lo que sea) sin utilizar la autenticación mutua. Esto abre la posibilidad para que una máquina pretenda ser la que usted desea, así que es mejor que tenga la debida precaución.

En ambos casos, también debe especificar un parámetro -Credential en el comando Remoting, aunque sólo esté especificando la misma credencial que está utilizando para ejecutar PowerShell. Cubriremos ambos casos en las siguientes dos secciones.

Nota: A lo largo de esta guía, usaremos "Comando Remoting" para referirnos genéricamente a cualquier comando que implique la creación de una conexión Remoting. Estos incluyen (pero no se limitan a) New-PSSession, Enter-PSSession, Invoke-Command, y así sucesivamente.

Configuración de un HTTPS Listener

Esta es una de las cosas más complejas que puede hacer con Remoting, e implicará ejecutar una gran cantidad de utilitarios externos. Lo siento - es sólo que así se hace- En este momento no parece haber una manera fácil de hacer esto totalmente desde PowerShell, o al menos no la hemos encontrado. Algunas cosas, podrían hacerse a través de PowerShell, pero como resulta más fácil hacerlo de otra forma, así lo he hecho.

El primer paso es identificar el nombre del host que la gente utilizará para acceder a su servidor. Esto es muy, muy importante, y no es necesariamente lo mismo que el nombre de equipo real del servidor. Por ejemplo, la gente que accede a "www.ad2008r2.loc" podría estar golpeando un servidor llamado "DC01", pero el certificado SSL que creará debe ser emitido para el nombre de host "www.ad2008r2.loc" porque eso es lo que la gente estará escribiendo Por lo tanto, el nombre del certificado debe coincidir con el nombre que la gente va a escribir para llegar a la máquina - incluso si es diferente de su verdadero nombre de equipo. ¿Lo tiene?

Nota: Nota: Parte de la configuración de un listener de HTTPS es obtener un certificado SSL. Utilizaré una Autoridad de Certificación (CA) pública llamada DigiCert.com. También puede usar una PKI interna, si su organización tiene una. No recomiendo usar MakeCert.exe, ya que los equipos que intentan conectarse no pueden confiar implícitamente en dicho certificado. Me doy cuenta de que cada blog en el universo le dice que use MakeCert.exe para crear un certificado auto-firmado local. Sí, es fácil, pero está mal. Usarlo

requiere que apague la mayor parte de la seguridad de WinRM, así que ¿por qué molestarse con SSL si planea apagar la mayoría de sus características de seguridad?

También necesita asegurarse de conocer el nombre completo usado para conectar con una computadora. Si la gente tiene que escribir "dc01.ad2008r2.loc", entonces eso es lo que debe aparecer en el certificado. Si simplemente necesita digitar "dca", y saber que un DNS puede resolver eso a una dirección IP, entonces "dca" es lo que debe llevar el certificado. Estamos creando un certificado que solo dice "dca" y debemos asegurarnos que nuestros equipos puedan resolver eso a una dirección IP.

Creación de una solicitud de certificado

A diferencia de IIS, PowerShell no ofrece una forma amigable y gráfica de crear una Solicitud de Certificado (de hecho no ofrece ninguna). Entonces, vaya a http://DigiCert.com/util[5] y descargue su versión gratuita del "Utilitario para certificados". La Figura 2.1 muestra el utilitario. Tenga en cuenta el mensaje de advertencia.

[5] http://DigiCert.com/util

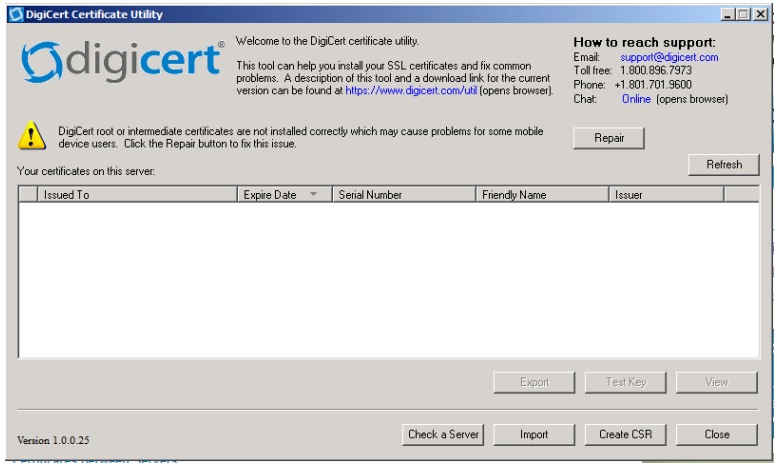

image008.png

Figura 2.1: Ejecutando DigiCertUtil.exe

Sólo tiene que preocuparse por la advertencia si planea adquirir su certificado de la CA de DigiCert. Haga clic en el botón Repair para instalar los certificados intermedios en su computadora, permitiendo que su certificado sea confiable y se pueda utilizar. La figura 2.2 muestra el resultado de hacerlo. Una vez más, si planea llevar la Solicitud de Certificado (CSR) eventual a una CA diferente, no se preocupe por el botón Repair o por el mensaje de advertencia

Nota También puede abrir una consola MMC en blanco y agregar el complemento “

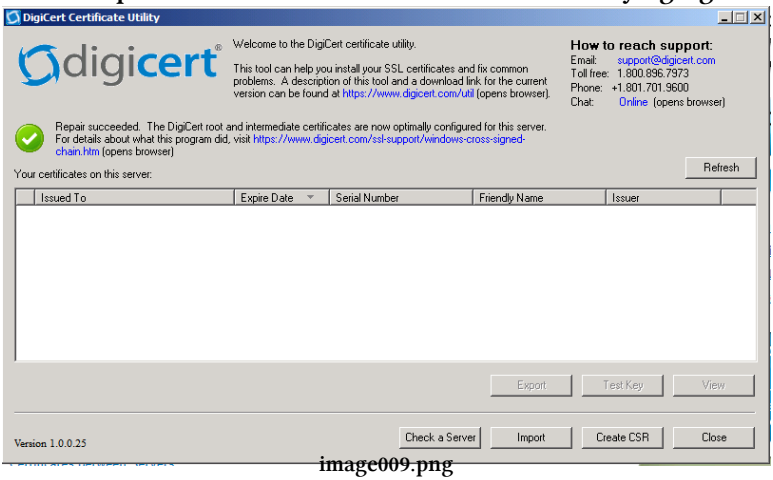

image009.png

Figura 2.2: Después de agregar los certificados intermedios de DigiCert

Haga clic en “ Create CSR”. Como se muestra en la figura 2.3, complete la información sobre su organización. Esto tiene que ser exacto: El “nombre común” es exactamente lo que la gente escribirá para acceder al equipo en el que se instalará este certificado SSL. Podría ser simplemente “dca”, en nuestro caso, o “dc01.ad20082.loc” si se necesita un nombre completo, y así sucesivamente. El nombre de su empresa también debe ser preciso: la mayoría de las CA verificarán esta información.

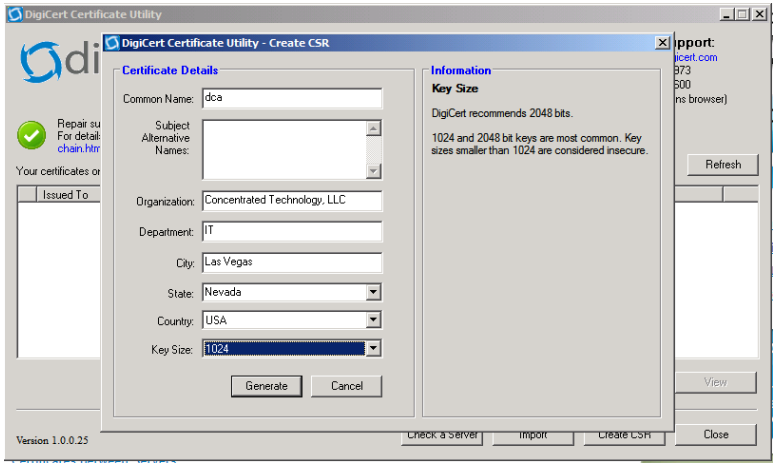

image010.png

Figura 2.3: Diligenciar el CSR

Por lo general, se guarda la CSR en un archivo de texto, como se muestra en la figura 2.4. También puede copiarlo en el Portapapeles. Cuando vaya a su CA, asegúrese de que está solicitando un certificado SSL (“Servidor Web”, en algunos casos). Un certificado de correo electrónico u otro tipo no funcionará.

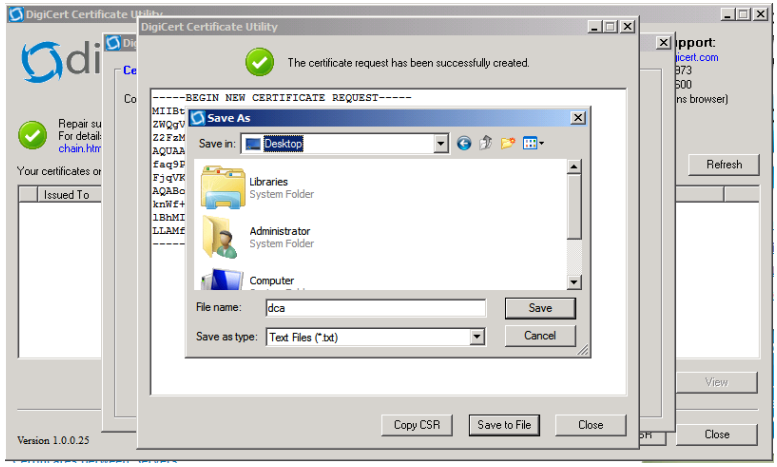

image011.png

Figura 2.4: Guardar el CSR en un archivo de texto

A continuación, lleve esa CSR a su CA y solicite su certificado. Verá algo como la figura 2.5 si está utilizando DigiCert. Obviamente será diferente con otra CA, con una PKI interna. Tenga en cuenta que con la mayoría de las CA comerciales tendrá que seleccionar el tipo de servidor Web que está utilizando (IIS o el que corresponda).

Nota: El uso del utilitario MakeCert.exe en el SDK de Windows generará un certificado local en el que solo su máquina confiará. Esto no es útil. Mucha gente le dirá que haga esto en varias publicaciones o blogs, porque es rápido y fácil. También le dirán que deshabilite algunas comprobaciones de seguridad para que el certificado inherentemente inútil funcione. Es una pérdida de tiempo. Usted estará utilizando cifrado, pero no tendrá la seguridad de que la máquina remota es a la que tenía la intención de conectarse. Si alguien está secuestrando su información, ¿a quién le importa si se cifró antes de enviarla a ellos?

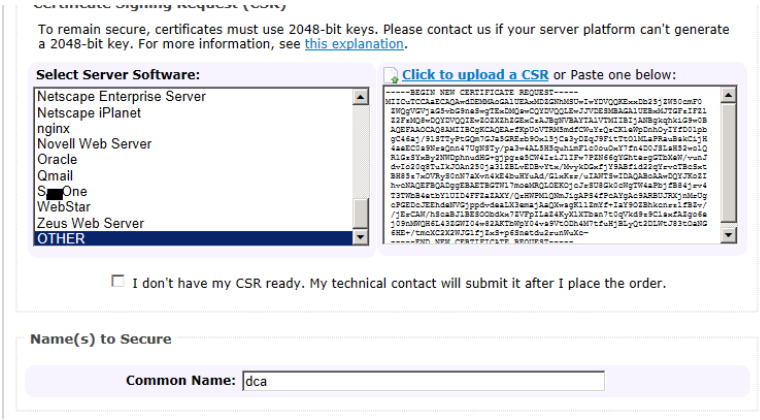

image012.png

Figura 2.5: Carga del CSR en una CA

Precaución: Observe el mensaje de advertencia en la figura 2.5. Mi CSR necesita ser generado con una clave de 2048 bits. La utilidad de DigiCert ofrece eso o 1024 bits. Muchas CA tendrán un requisito de bit-alto. Asegúrese de que su RSE cumple con lo que necesita. Observe también que se trata de un certificado de servidor Web lo que estamos solicitando. Como escribimos anteriormente, es el único tipo de certificado que funcionará.

Eventualmente, la CA emitirá su certificado. La Figura 2.6 muestra el sitio a dónde fuimos para descargarlo. Elegimos descargar todos los certificados. Queríamos asegurarnos de tener una copia del certificado raíz de la CA, en caso de que necesitáramos configurar otra máquina para confiar en esa raíz.

Sugerencia: El truco con los certificados digitales es que la máquina que los utiliza y las máquinas a las que se presentarán, deben confiar en la entidad emisora de certificados que emitió el mismo. Es por eso que descarga el certificado raíz de la CA, para que pueda instalarlo en las máquinas que necesitan confiar en dicha CA. En un entorno grande, esto se puede hacer a través de una directiva de grupo, si se quisiera.

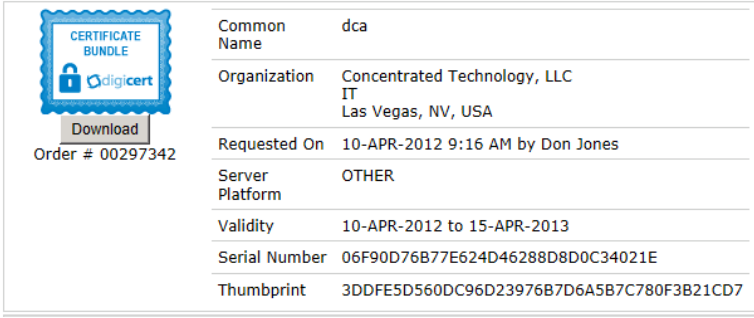

image013.png

Figura 2.6: Descarga del certificado emitido

Asegúrese de hacer una copia de seguridad de los archivos de certificados. Aunque la mayoría de las CA las publicarían de nuevo de ser necesario, es mucho más fácil tener una copia de seguridad.

Instalación del certificado

No intente hacer doble clic en el archivo de certificado para instalarlo. Si lo hace, lo instalará en el almacén de certificados de su cuenta de usuario. Lo necesita en el almacén de certificados de su computadora. Para instalar el certificado, abra una nueva consola de administración de Microsoft (mmc.exe), seleccione Agregar o quitar complementos y agregue el complemento Certificados, como se muestra en la figura 2.7.

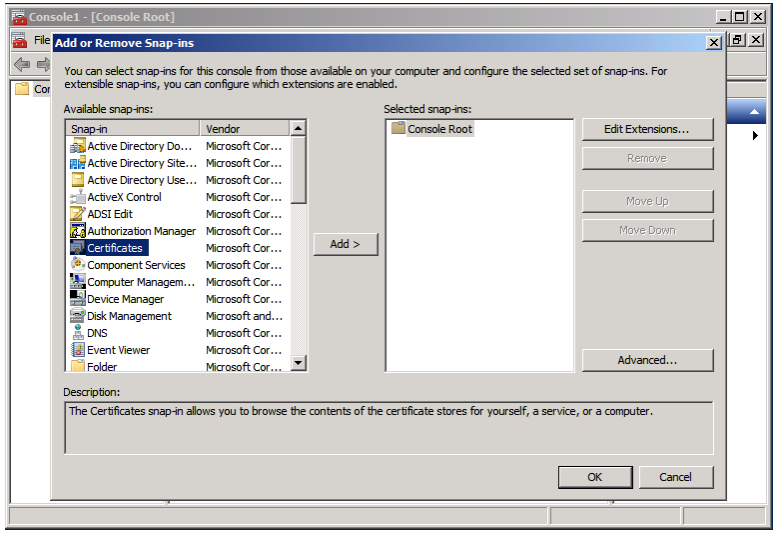

image014.png

Figura 2.7: Agregar el complemento Certificados a la MMC

Como se muestra en la figura 2.8, establezca el complemento en la cuenta de equipo.

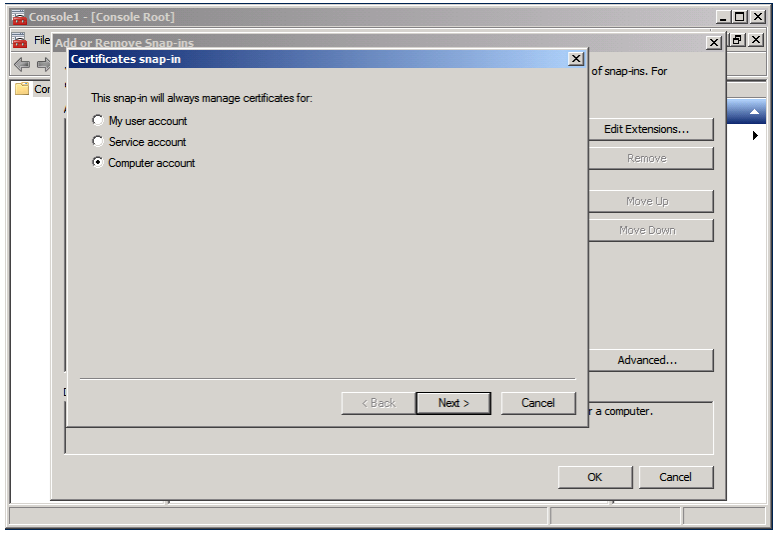

image015.png

Figura 2.8: Establecer el complemento Certificados en la cuenta de equipo

A continuación, como se muestra en la figura 2.9, establezca el equipo local. Por supuesto, si está instalando un certificado en una computadora remota, establezca esa computadora en su lugar. Esta es una buena forma de instalar un certificado en un ambiente Windows sin GUI como en un Server Core, por ejemplo.

Nota: Quisiéramos poder mostrarle una forma de hacer todo esto desde PowerShell. No pudimos encontrar una que no implicara un montón de pasos más además de complejos. Dado que esto no es algo que tendrá que hacer a menudo o automatizarlo, la GUI es más fácil y debería ser suficiente.

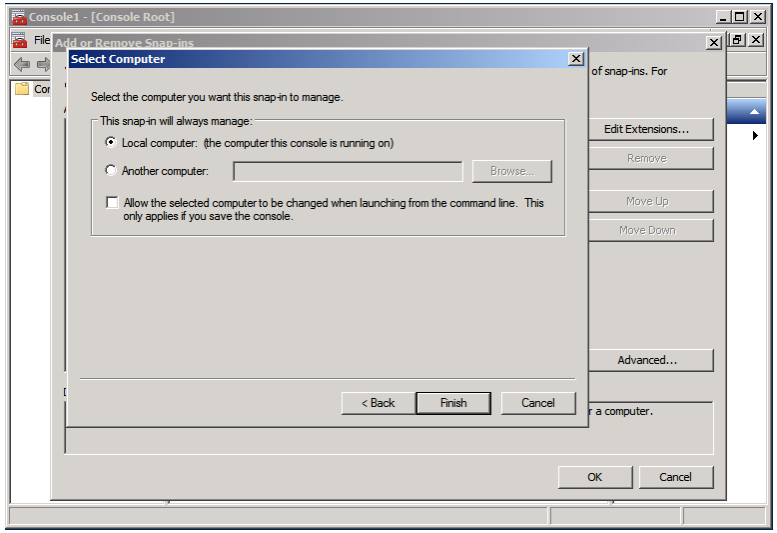

image016.png

Figura 2.9: Establecer el complemento Certificados en el equipo local

Con el complemento cargado, como se muestra en la figura 2.10, haga clic con el botón derecho en el almacén “Personal” y seleccione “Import”.

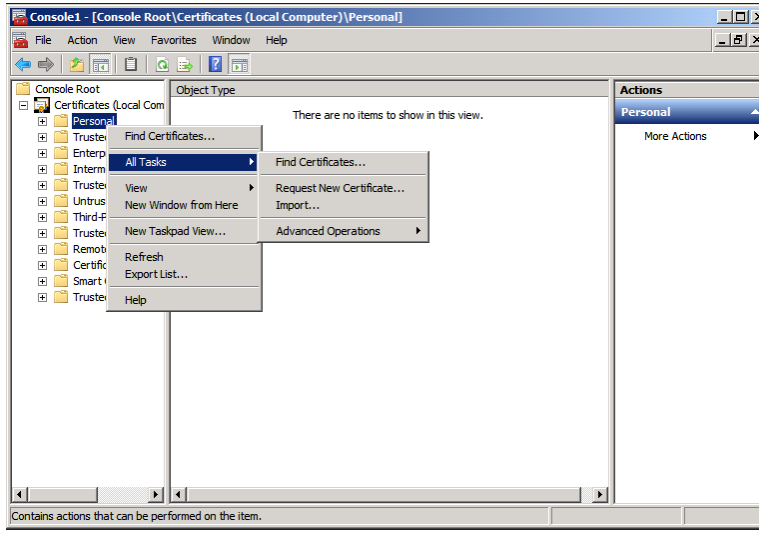

image017.png

Figura 2.10: Inicio del proceso de importación en el almacén Personal

Como se muestra en la figura 2.11, vaya al archivo de certificado que descargó de su CA. A continuación, haga clic en Next.

Precaución: Si ha descargado varios certificados, tal vez los certificados raíz de la CA junto con el certificado, asegúrese de importar el certificado SSL que se le entregó. Si hay alguna confusión, PARE. Vuelva a su CA y descargue sólo su certificado, para que sepa cuál importar. No experimente, necesita realizar bien esto a la primera vez.

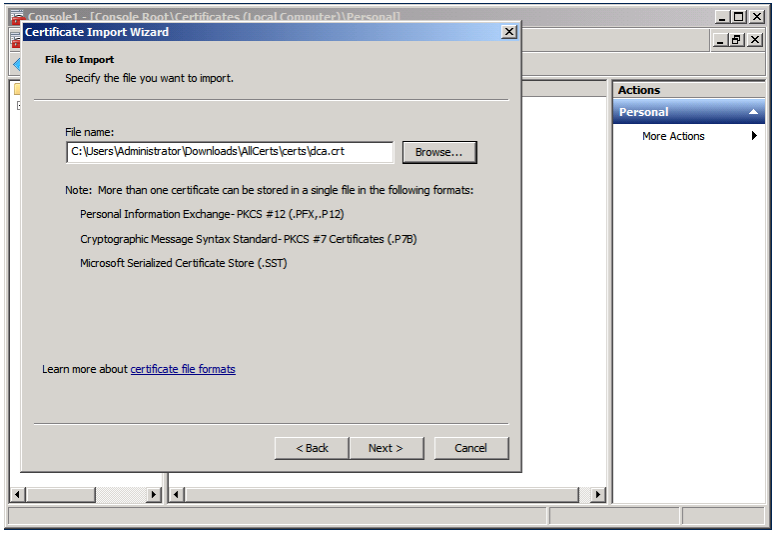

image018.png

Figura 2.11: Selección del archivo de certificado SSL recién publicado

Como se muestra en la figura 2.12, asegúrese de que el certificado se ubicará en el almacén Personal.

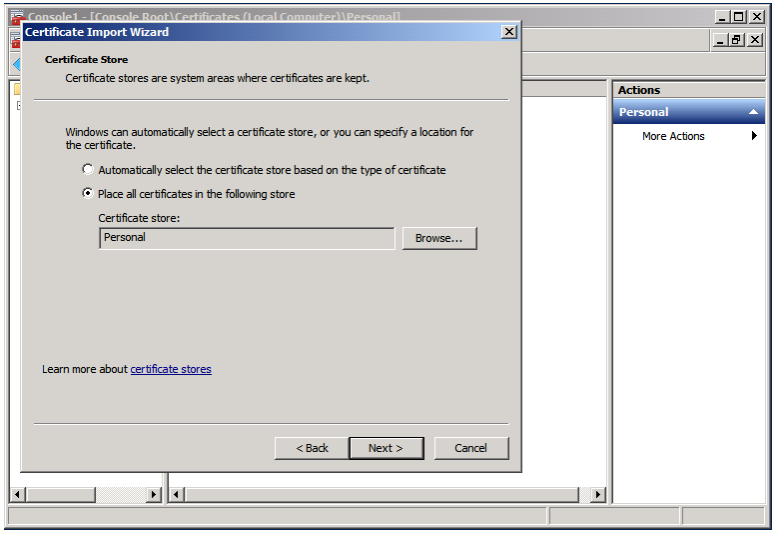

image019.png

Figura 2.12: Asegúrese de ubicar el certificado en el almacén Personal, que debe estar preseleccionado.

Como se muestra en la figura 2.13, haga doble clic en el certificado para abrirlo. O bien, haga clic con el botón derecho y seleccione Abrir. No seleccione Propiedades - no le proporcionará la información que necesita-.

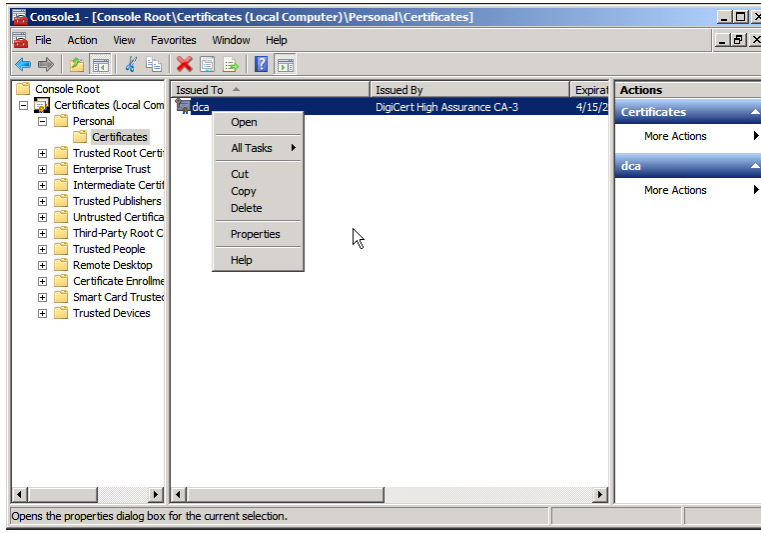

image020.png

Figura 2.13: Haga doble clic en el certificado o haga clic con el botón derecho del ratón y seleccione Open

Finalmente, como se muestra en la figura 2.14, seleccione la huella digital del certificado. Deberá anotar esto o copiarlo en el Portapapeles. Así WinRM identificará el certificado que desea utilizar.

Nota: Es posible listar su certificado en la unidad CERT: de PowerShell, lo que hará que la huella digital sea más fácil de copiar en el Portapapeles. En PowerShell, ejecute Dir CERT:LocalMachineMy. Asegúrese que selecciona el certificado correcto. Si no se muestra toda la huella digital, ejecute Dir CERT:LocalMachineMy | FL * en su lugar.

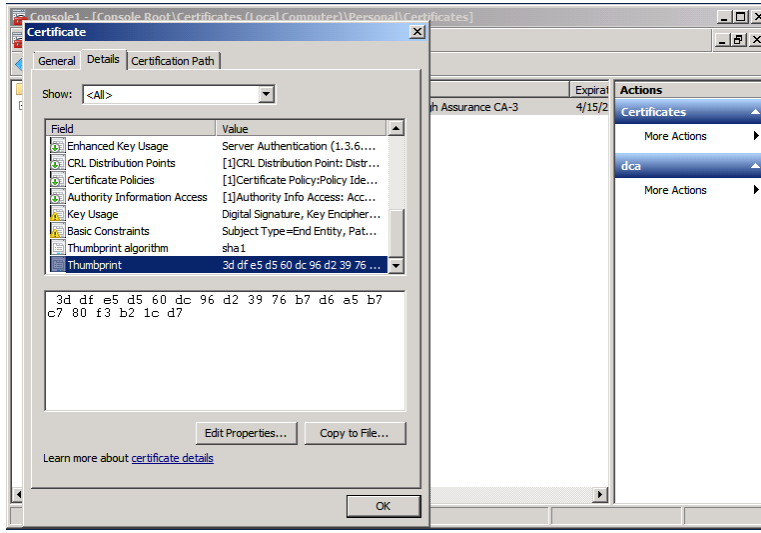

image021.png

Figura 2.14: Validación de la huella digital del certificado

Configuración del listener de HTTPS

Los siguientes pasos se llevarán a cabo en el shell Cmd.exe, no en PowerShell. La sintaxis de la utilidad de línea de comandos requiere un ajuste significativo y escapar algunas cosas en PowerShell, así que es mucho más sencillo de escribir y entender directamente en el shell de Cmd.exe (que es donde la utilidad tiene que ejecutarse de todos modos). Ejecutarlo en PowerShell sólo lanzaría Cmd. Exe tras bambalinas.

Como se muestra en la figura 2.15, ejecute el siguiente comando:

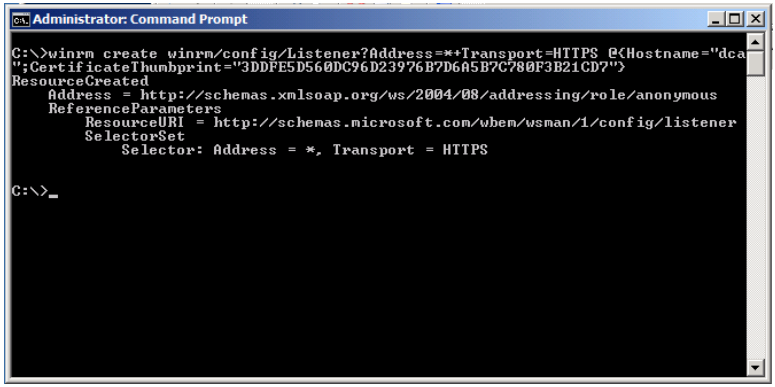

image022.png

Figura 2.15: Configuración del listener de HTTPS WinRM

```
Winrm create winrm/config/Listener?Address=\*+Transport=H\
TTPS @{Hostname="xxx";CertificateThumbprint="yyy"}
```

Hay dos o tres piezas de información que necesitará colocar en este comando:

- En lugar de *, puede poner una dirección IP individual. El uso de * hará que el oyente (listener) escuche todas las direcciones IP locales.
- En lugar de xxx, coloque el nombre de equipo exacto para el que se emitió el certificado. Si eso incluye un nombre de dominio (como dc01.ad2008r2.loc), póngalo. Lo que está en el certificado debe ir aquí, de lo contrario tendrá un error de coincidencia CN. Como nuestro certificado fue emitido a "dca", puse "dca"
- En lugar de yyy, coloque la huella digital de certificado exacta que copió anteriormente. Está bien si contiene espacios.

Eso es todo lo que debe hacer para que el oyente (listener) funcione.

Nota: Teníamos el Firewall de Windows deshabilitado en este servidor, por lo que no necesitamos crear una excepción. La excepción no se crea automáticamente; por lo tanto, si tiene un firewall habilitado en su computadora, deberá crear manualmente la excepción para el puerto 5986.

También puede ejecutar un comando equivalente de PowerShell para realizar esta tarea:

```
1 New-WSManInstance winrm/config/Listener -SelectorSet @{Ad\
2 dress='\*';
3 Transport='HTTPS'} -ValueSet @{HostName='xxx';Certificate\
4 Thumbprint='yyy'}
```

En ese ejemplo, "xxx" y "yyy" se reemplazan como lo hicieron en el ejemplo anterior.

Probando el HTTPS Listener

He probado esto desde el equipo C3925954503 independiente, tratando de llegar al controlador de dominio DCA en COMPANY.loc. He configurado C3925954503 en el archivo HOSTS, de modo que podría resolver el nombre de host DCA a la dirección IP correcta sin necesidad del DNS. Estaba seguro de que correría:

```
1 Ipconfig /flushdns
```

Esto aseguró que el archivo HOSTS se leyó en el caché de nombres DNS. Los resultados se muestran en la figura 2.16. Tenga en cuenta que no puedo acceder a DCA utilizando directamente su dirección IP, porque el certificado SSL no contiene una dirección IP. El certificado SSL se emitió a "dca", por lo que tenemos que ser capaces de acceder a la computadora escribiendo "dca" como el nombre de la computadora. El uso del archivo HOSTS permitirá que Windows resuelva eso a una dirección IP.

Nota: Recuerde, hay dos cosas que suceden aquí: Windows debe poder resolver el nombre a una dirección IP, que es lo que hace el archivo HOSTS, con el fin de hacer una conexión física. Pero WinRM necesita autenticación mutua, lo que significa que cualquier cosa que escribimos en el parámetro -ComputerName debe coincidir con lo que está en el certificado SSL. Es por eso que no podemos simplemente proporcionar una dirección IP al comando - habría funcionado para la conexión, pero no la autenticación.

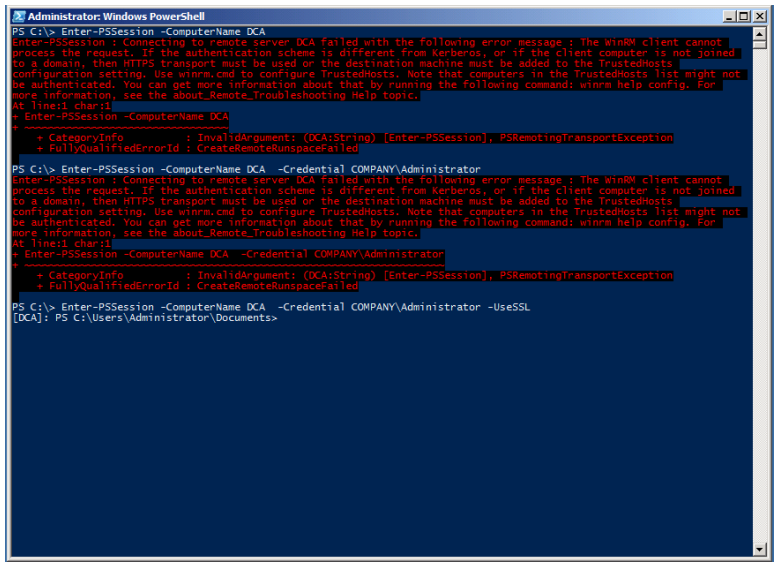

image023.png

Figura 2.16: Prueba del Listener de HTTPS

Comenzamos con esto:

```
1 Enter-PSSession -computerName DCA
```

No funcionó, como se esperaba. Entonces intentamos esto:

```
Enter-PSSession -computerName DCA -credential COMPANY\Adm\
inistrator
```

Proporcionamos una contraseña válida para la cuenta de administrador, pero como se esperaba, el comando no funcionó. Finalmente:

```
Enter-PSSession -computerName DCA -credential COMPANY\Adm\
inistrator -UseSSL
```

Nuevamente proporcionando una contraseña válida, se mostró el aviso remoto que esperábamos. ¡Funcionó! Esto cumple las dos condiciones que especificamos anteriormente: Estamos utilizando una conexión HTTPS y proporcionamos una credencial. Ambas condiciones son necesarias porque el equipo no está en mi dominio (ya que en este caso el equipo de origen no está ni siquiera en un dominio). Solo para recordar, la figura 2.17 muestra, en verde, la conexión que creamos y usamos.

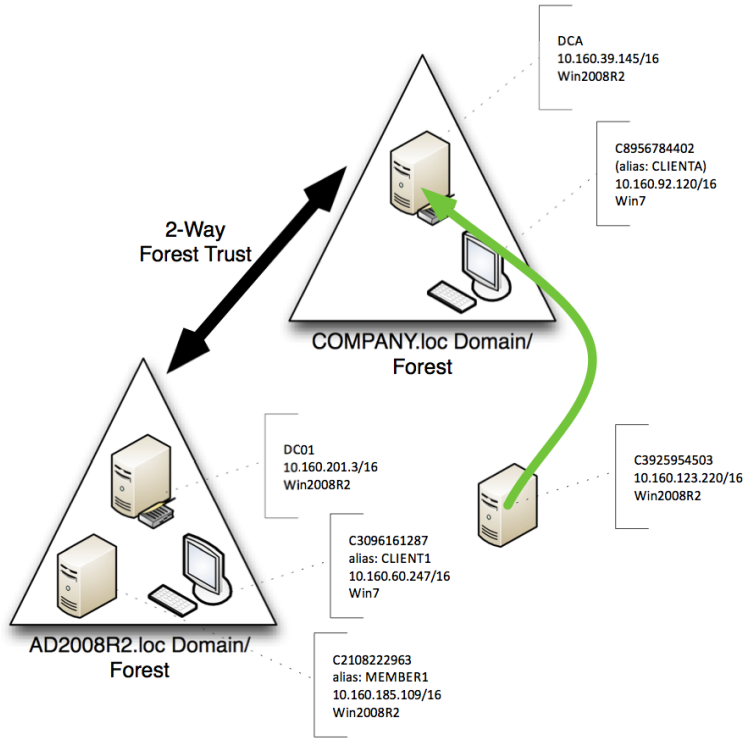

image024.png

Figura 2.17: La conexión utilizada para la prueba de escucha de HTTPS

Modificadores

Hay dos modificadores que puede utilizar en una conexión, ya sea con Invoke-Command, Enter-PSSession o algún otro comando Remoting, que se relacionan con los oyentes (listeners) HTTPS. Éstos se crean como parte de un objeto de opción de sesión.

- -SkipCACheck hace que WinRM no se preocupe si el certificado SSL fue emitido por una entidad de confianza o no. Sin embargo, utilizar CAs no confiables en realidad puede

ser poco fiable. Una CA "pobre" puede emitir un certificado para una computadora falsa, lo que le lleva a creer que se está conectando a la máquina correcta cuando de hecho se está conectando a una maquina impostora. Esto es riesgoso, así que úselo con precaución.

- -SkipCNCheck hace que WinRM no se preocupe si el certificado SSL en la máquina remota se emitió realmente para esa máquina o no. Una vez más, esta es una gran manera de encontrarse conectado a un impostor. La mitad del punto de SSL es la autenticación mutua, y este parámetro desactiva esa mitad.

El uso de una o ambas de estas opciones seguirán activando el cifrado SSL en la conexión, pero habrá aniquilado el otro propósito esencial de SSL, que es la autenticación mutua por medio de una autoridad intermedia de confianza.

Para crear y utilizar un objeto de sesión que incluye ambos parámetros:

```
$option = New-PSSessionOption -SkipCACheck -SkipCNCheck
Enter-PSSession -computerName DCA -sessionOption $option
        -credential COMPANY\Administrator -useSSL
```

Precaución: Sí, esta es una manera fácil de hacer que los mensajes de error molestos desaparezcan. Pero esos errores están intentando advertirle de un problema potencial y le defienden de riesgos potenciales de la seguridad que son muy reales, y que están muy en uso por los atacantes modernos.

Autenticación de certificados

Una vez que haya establecido un oyente (listener) de HTTPS, tendrá la opción de autenticarse con Certificados. Esto le permite

conectarse a equipos remotos, incluso aquellos en un dominio o grupo de trabajo no confiable, sin requerir el ingreso de usuario-clave, lo que puede ser útil cuando se programa una tarea para ejecutar un script de PowerShell, por ejemplo.

En la autenticación de certificados, el cliente tiene un certificado con una clave privada y el equipo remoto asigna la clave pública de ese certificado a una cuenta de Windows local. WinRM requiere un certificado que tenga "Client Authentication (1.3.6.1.5.5.7.3.2)" que aparece en el atributo Enhanced Key Usage, y que tiene un nombre principal de usuario listado en el atributo Subject Alternative Name. Si utiliza la Autoridad de Certificación de Microsoft Enterprise, la plantilla de certificado "Usuario" cumple estos requisitos.

Obtención de un certificado de autenticación de cliente

Estas instrucciones asumen que tiene una CA de Microsoft Enterprise. Si está utilizando un método diferente de inscripción de certificados, siga las instrucciones proporcionadas por su proveedor o el administrador de CA.

En el equipo cliente, realice los siguientes pasos:

- Ejecute certmgr.msc para abrir la consola " Certificates - Current User".
- Haga clic derecho en el nodo "Personal" y seleccione All Tasks -> Request New Certificate
- En el cuadro de diálogo Certificate Enrollment, haga clic en Next. Seleccione "Active Directory Enrollment Policy " y haga clic en Next de nuevo. Seleccione la plantilla User y haga clic en Enroll

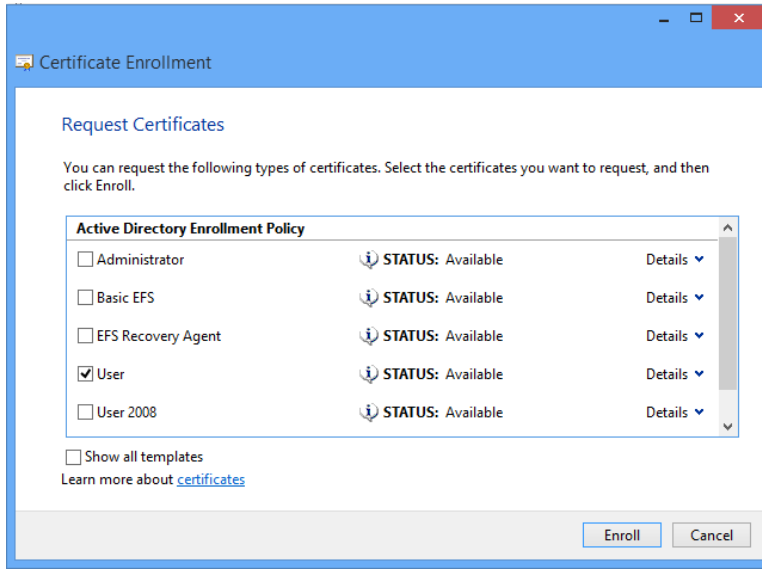

image025.png

Figura 2.18: Solicitud de un certificado de usuario.

Una vez finalizado el proceso de inscripción (enrolamiento) y regrese de nuevo a la consola de certificados, debería ver el nuevo certificado en la carpeta PersonalCertificates:

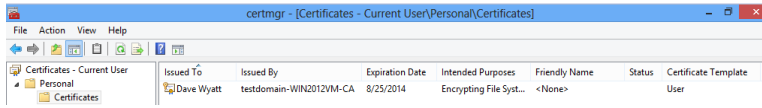

image026.png

Figura 2.19: El certificado de autenticación de cliente instalado del usuario.

Antes de cerrar la consola de Certificados, haga clic con el botón derecho en el nuevo certificado y seleccione All Tasks -> Export. En las pantallas siguientes, elija “ do not export the private key” y guarde el certificado en un archivo en disco. Copie el certificado exportado al equipo remoto, para utilizarlo en los pasos siguientes.

Configuración del equipo remoto para permitir la autenticación de certificados

En el equipo remoto, ejecute la consola de PowerShell como Administrador e introduzca el siguiente comando para habilitar la autenticación del certificado:

```
Set-Item -Path WSMan:\localhost\Service\Auth\Certificate \
-Value $true
```

Importar el certificado del cliente en el equipo remoto

El certificado del cliente debe agregarse al almacén de certificados de " Trusted People" del equipo. Para ello, realice los pasos siguientes para abrir la consola "Certificados (equipo local)":

- Ejecutar "mmc".
- En el menú Archivo, elija " Add/Remove Snap-in".
- Resalte " Certificates" y haga clic en el botón Add.
- Seleccione la opción " Computer Account" y haga clic en Next.
- Seleccione "Local Computer", haga clic en Finish y a continuación, haga clic en OK

Nota: Este es el mismo proceso que siguió en la sección "Instalación del certificado" en Configuración y escucha de HTTPS. Consulte las figuras 2.7, 2.8 y 2.9 si es necesario.

En la consola de Certificados (Local Computer), haga clic con el botón secundario "Trusted People" y seleccione All Tasks -> Import.

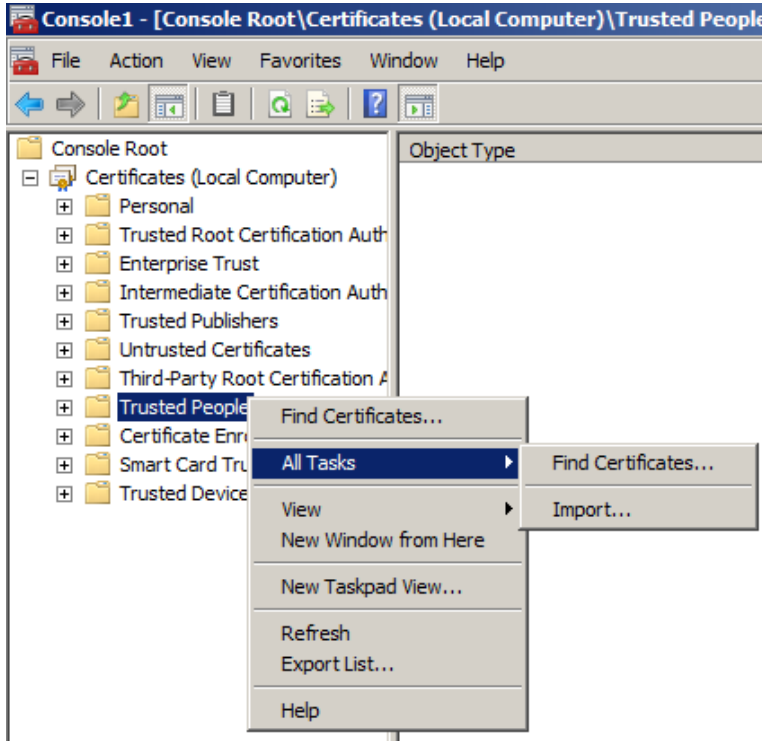

image027.png

Figura 2.20: Inicio del proceso de importación de certificados.

Haga clic en Next y busque la ubicación donde copió el archivo de certificado del usuario.

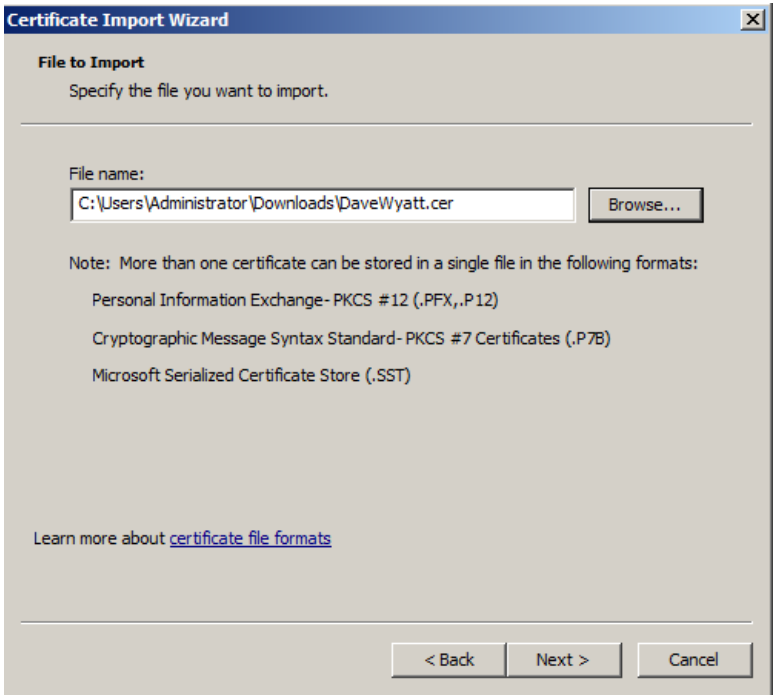

image028.png

Figura 2.21: Selección del certificado del usuario.

Asegúrese de que el certificado se coloca en el almacén de “Trusted People”:

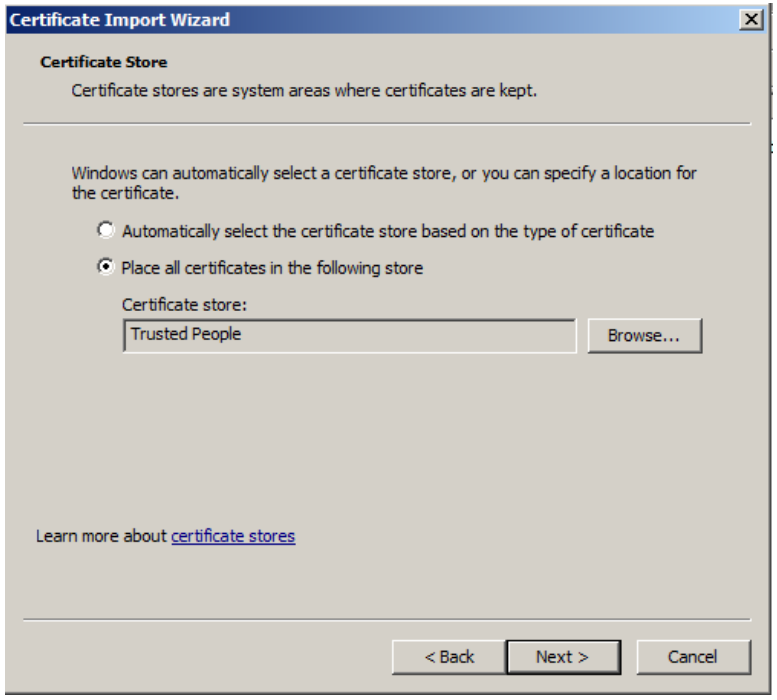

image029.png

Figura 2.22: Colocación del certificado en el almacén de “Trusted People”.

Creación de una asignación de certificados de cliente en el equipo remoto

Abra una consola de PowerShell como Administrador en el equipo remoto. Para el paso siguiente, necesitará la huella digital de certificado de la CA que emitió el certificado del cliente. Debería poder encontrarlo utilizando uno de estos comandos (dependiendo de si el certificado de la entidad emisora de certificados se encuentra en las “ Trusted Root Certification Authorities “ o en la “Intermediate Certification Authorities”):

```
Get-ChildItem -Path cert:\LocalMachine\Root
Get-ChildItem -Path cert:\LocalMachine\CA
```

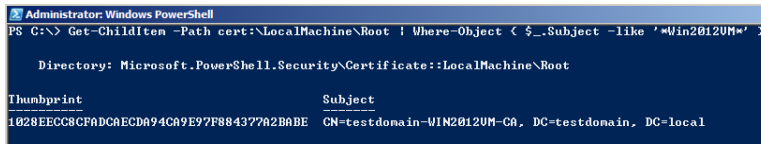

image030.png

Figura 2.23: Obtención de la huella digital del certificado de la CA.

Una vez que tenga la huella digital, emita el siguiente comando para crear la asignación de certificados:

```
New-Item -Path WSMan:\localhost\ClientCertificate -Creden\
tial (Get-Credential) -Subject <userPrincipalName> -URI \\
* -Issuer <CA Thumbprint> -Force
```

Cuando se le pidan credenciales, ingrese el nombre de usuario y la contraseña de una cuenta local con derechos de administrador.

Nota: No es posible especificar las credenciales de una cuenta de dominio para la asignación de certificados, incluso si el equipo remoto es un miembro de un dominio. Debe utilizar una cuenta local y la cuenta debe ser miembro del grupo Administradores.

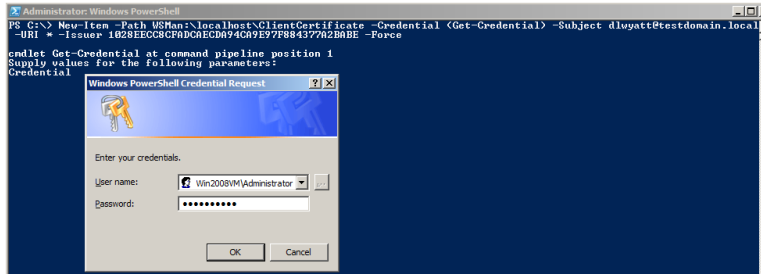

image031.png

Figura 2.24: Configuración de la asignación de certificados de cliente.

Conexión al equipo remoto mediante la autenticación de certificados

Ahora, debe estar listo para autenticarse en el equipo remoto utilizando su certificado. En este paso, necesitará la huella digital del certificado de autenticación de cliente. Para obtenerlo, puede ejecutar el siguiente comando en el equipo cliente:

```
1  Get-ChildItem -Path Cert:\CurrentUser\My
```

Una vez que tenga esta huella digital, puede autenticarse en el equipo remoto mediante los Cmdlets Invoke-Command o New-PSSession con el parámetro -CertificateThumbprint, como se muestra en la figura 2.25.

Nota: El Cmdlet Enter-PSSession no parece funcionar con el parámetro -CertificateThumbprint. Si desea introducir una sesión de acceso remoto interactiva con autenticación de certificado, utilice primero New-PSSession y, a continuación, Enter-PSSession.

Nota: El modificador -UseSSL está implícito cuando se utiliza -CertificateThumbprint en cualquiera de estos comandos. Incluso si no escribe -UseSSL, seguirá conectándose al equipo remoto a través de HTTPS (puerto 5986, de forma predeterminada, en Windows 7/2008 R2 o posterior). La Figura 2.26 muestra esto.

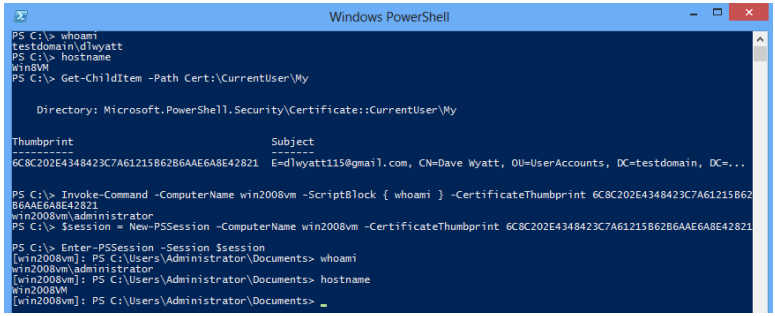

image032.png

Figura 2.25: Uso de un certificado para autenticarse con PowerShell Remoting.

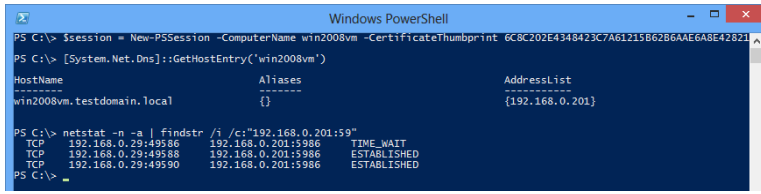

image033.png

Figura 2.26: Demostración de que la conexión está sobre el puerto SSL 5986, incluso sin el modificador -UseSSL.

Modificación de la lista TrustedHosts

Como mencioné anteriormente, el uso de SSL es sólo una opción para conectarse a un equipo para el que no es posible la autenticación mutua. La otra opción es desactivar selectivamente la necesidad de autenticación mutua proporcionando a su equipo una lista de "hosts de confianza". En otras palabras, le está diciendo a su computadora, "Si intento acceder a SERVER1 [por ejemplo], no se molesten con la autenticación mutua. Estoy seguro que SERVER1 no puede ser falsificado o suplantado, por lo que estoy tomando esa responsabilidad."

La figura 2.27 ilustra la conexión que vamos a intentar.

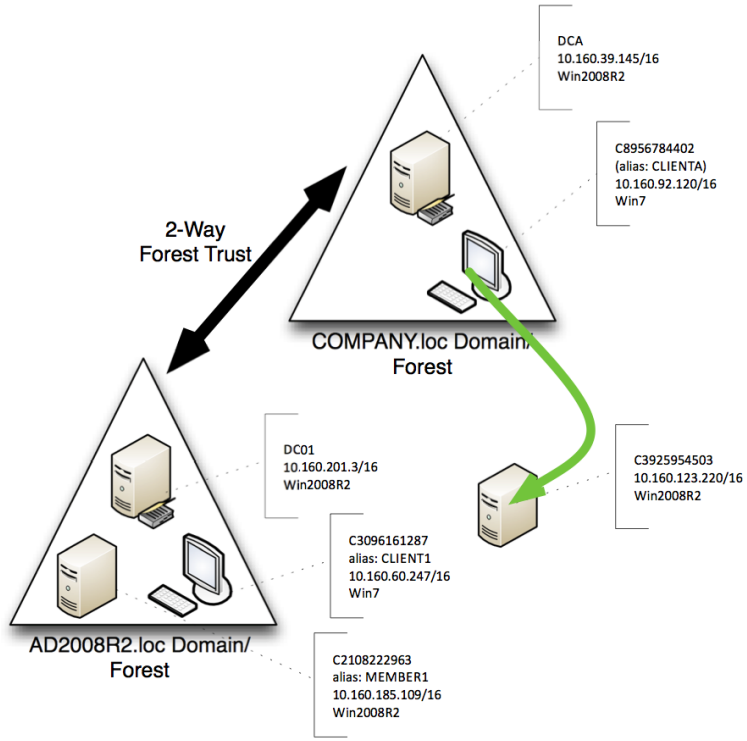

image034.png

Figura 2.27: Prueba de conexión a un TrustedHosts

A partir de un cliente, con una configuración Remoting completamente predeterminada, intentaremos conectarnos a C3925954503, que también tiene una configuración de Remoting por defecto. La figura 2.28 muestra el resultado. Tenga en cuenta que estoy conectando a través de la dirección IP, en lugar de al hostname. Nuestro cliente no tiene ninguna manera de resolver el nombre de la computadora a una dirección IP, y para esta prueba preferimos no modificar mi archivo local HOSTS.

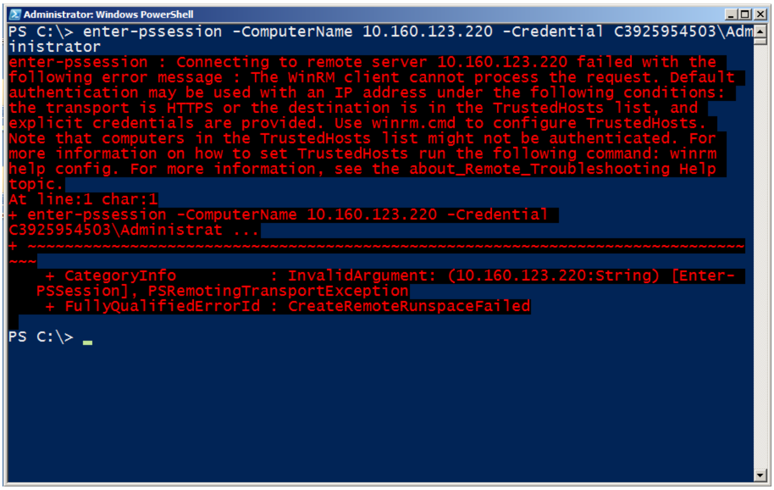

image035.png

Figura 2.28: Intentando conectarse al equipo remoto

Esto es lo que esperábamos: El mensaje de error es claro. No podemos usar una dirección IP (o un nombre de host para un equipo que no sea de dominio, aunque el error no lo diga) a menos que utilicemos HTTPS y una credencial o que agregue la computadora a mi lista de TrustedHosts y use una credencial. Elegiremos esta última opción. La figura 2.29 muestra el comando que debemos ejecutar. Si hubiéramos querido conectarnos a través del nombre de la computadora (C3925954503) en lugar de su dirección IP, habríamos añadido ese nombre de equipo a la lista de TrustedHosts (sería nuestra responsabilidad asegurar que mi computadora pudiera de alguna manera resolver ese nombre de equipo a una dirección IP para realizar la conexión física).

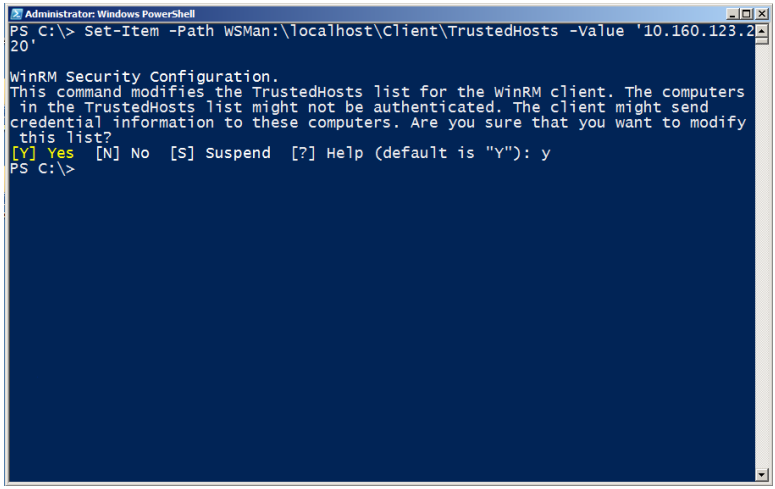

image036.png

Figura 2.29: Agregar la máquina remota a nuestra lista TrustedHosts

Este es otro caso en el que muchos blogs aconsejarán simplemente poner "*" en la lista de TrustedHosts. ¿De verdad? ¿No hay ninguna posibilidad de que cualquier computadora, nunca, en ningún lugar, pudiera ser suplantada o falsificada? Preferimos agregar un conjunto limitado y controlado de nombres de host o direcciones IP. Utilice una lista separada por comas. Está bien usar comodines junto con otros caracteres (como un nombre de dominio, como *.COMPANY.loc), para permitir un rango amplio pero no ilimitado de computadoras. La figura 2.30 muestra una conexión correcta.

Sugerencia: Utilice el parámetro -Concatenate de Set-Item para agregar su nuevo valor a los existentes, en lugar de sobrescribirlos.

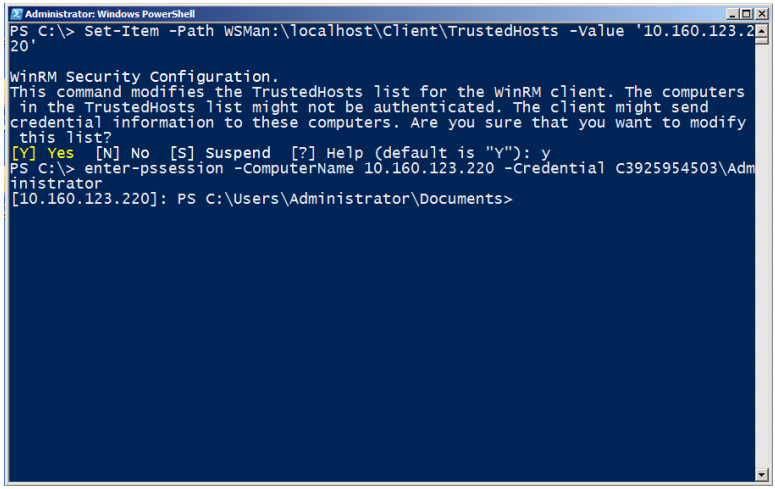

image037.png

Figura 2.30: Conexión a la computadora remota

Administrar la lista de TrustedHosts es probablemente la forma más fácil de conectarse a un equipo que no puede ofrecer autenticación mutua, siempre y cuando esté absolutamente seguro de que la suplantación no es una posibilidad. En una intranet, por ejemplo, donde ya tiene buenas prácticas de seguridad, la suplantación puede ser una posibilidad remota y puede agregar un rango de direcciones IP o un rango de nombres de host utilizando comodines.

Conexión a través de dominios

La Figura 2.31 ilustra la siguiente conexión que trataremos de hacer, que se encuentra entre dos equipos en dominios diferentes de confianza.

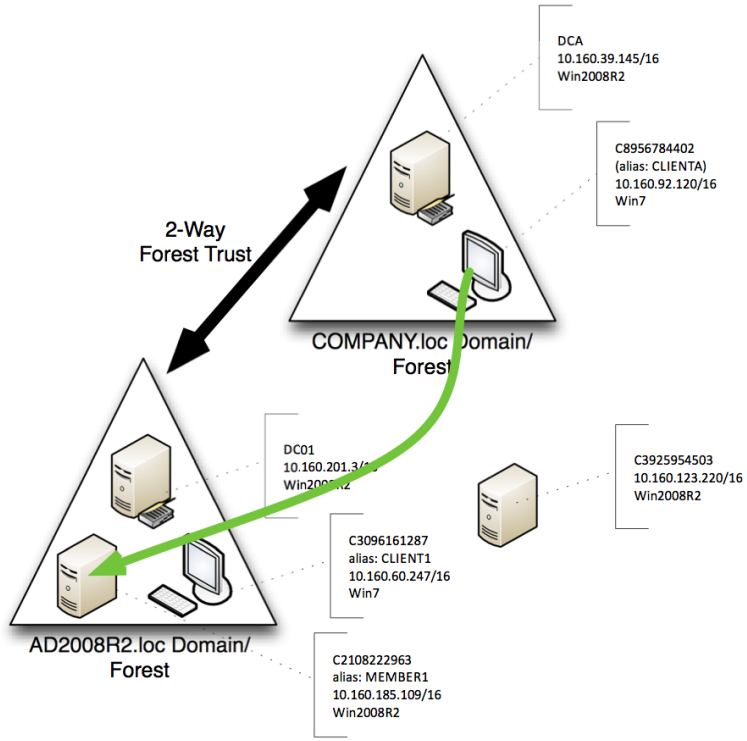

image038.png

Figura 2.31: Conexión de prueba entre dominios

Nuestra primera prueba está en la figura 2.32. Tenga en cuenta que estamos creando una credencial reutilizable en la variable $cred, para que no tengamos que volver a teclear la contraseña mientras lo intentamos. Sin embargo, los resultados de la prueba de Remoting todavía no tienen éxito.

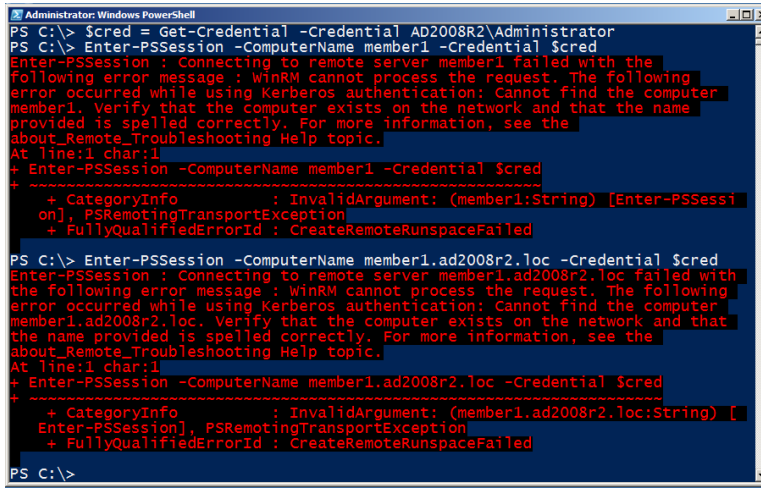

image039.png

Figura 2.32: Intentar conectarse al equipo remoto

¿El problema? Estamos usando un alias CNAME (MEMBER1), no el nombre de host real de la computadora (C2108222963). Aunque WinRM puede utilizar un CNAME para resolver un nombre a una dirección IP para la conexión física, no puede utilizar el alias CNAME para buscar el equipo en AD, ya que AD no utiliza el registro CNAME (incluso en una Zona AD-DNS integrada). Como se muestra en la figura 2.33, la solución es usar el nombre de host real de la computadora.

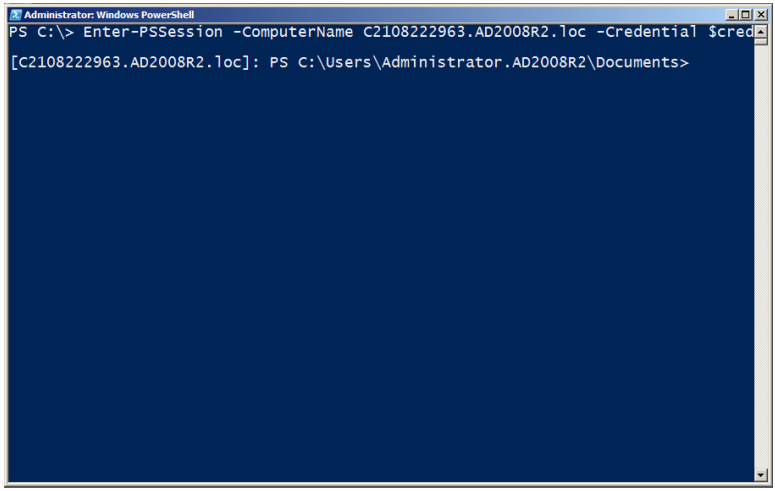

image040.png

Figura 2.33: Conectar correctamente a través de dominios

¿Qué pasa si *necesita* usar una dirección IP o alias CNAME para conectarse? Tendrá que volver a la lista de TrustedHosts o a un detector de HTTPS, exactamente como si se estuviera conectando a un equipo que no pertenece al dominio. Esencialmente, si no puede utilizar el nombre de host real de la computadora, tal como aparece en AD, entonces no puede confiar en el dominio para acelerar el proceso de autenticación.

Administradores de otros dominios

Hay una peculiaridad en Windows que tiende a obviar el token de la cuenta de administrador para las cuentas de administrador procedentes de otros dominios, lo que significa que terminan ejecutándose bajo privilegios de usuario estándar, lo que a menudo no es suficiente. En el dominio de destino, se puede cambiar ese comportamiento.

Para ello, ejecute esto en el equipo de destino (escriba todo esto en una línea y pulse Enter):

```
1 New-ItemProperty -Name LocalAccountTokenFilterPolicy
2 -Path HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\
3 Policies\System -PropertyType Dword -Value 1
```

Eso debería solucionar el problema. Tenga en cuenta que esto desactiva el Control de cuentas de usuario (UAC) en la máquina donde lo ejecutó, así que asegúrese de lo que está haciendo antes de hacerlo.

El segundo salto

Una limitación predeterminada con Remoting es a menudo referida como el segundo salto. La Figura 2.25 ilustra el problema básico: Puede realizar una conexión Remoting de un host a otro (la línea verde), pero pasar de ese segundo host a un tercero (la línea roja) simplemente se rechaza. Este "segundo salto" no funciona porque, de forma predeterminada, Remoting no puede delegar su credencial por segunda vez. Esto es incluso un problema si realiza el primer salto y posteriormente intenta acceder a cualquier recurso de red que requiera autenticación. Por ejemplo, si accede a otro equipo y, a continuación intenta tener acceso a algún archivo compartido pero necesita autenticación, la operación falla.

La solución CredSSP

Los siguientes cambios de configuración son necesarios para habilitar el segundo salto:

Nota: Esto sólo funciona en Windows Vista, Windows Server 2008 y versiones posteriores de Windows. No funcionará en Windows XP o Windows Server 2003 o versiones anteriores.

- CredSSP debe estar habilitado en su computadora de origen y el servidor intermedio al que se conecta. En PowerShell, en el equipo de origen, ejecute:

```
Set-Item WSMAN:\localhost\client\auth\credssp -value $true
```

- En su (s) servidor (es) intermedio (s), realiza un cambio similar al anterior, pero en una sección diferente de la configuración:

```
Set-Item WSMAN:\localhost\service\auth\credssp -value $tr\
ue
```

- Su política de dominio debe permitir la delegación de nuevas credenciales. En un objeto de directiva de grupo (GPO), se encuentra en Configuración del equipo> Políticas> Plantillas administrativas> Sistema> Delegación de credenciales> Permitir delegación de nuevas credenciales. Debe proporcionar los nombres de las máquinas a las que se pueden delegar las credenciales o especificar un comodín como "*.ad2008r2.loc" para permitir un dominio completo. Asegúrese de dar tiempo para que el GPO actualizado se aplique o ejecute Gpupdate en el equipo de origen (o reinícielo).

Nota: Una vez más, el nombre que usted proporciona aquí es importante. Lo que realmente va a escribir para el parámetro -ComputerName es lo que debe aparecer aquí. Esto hace que sea realmente difícil delegar credenciales a, digamos, direcciones IP, sin agregar simplemente "*" como delegado permitido. La adición de "*", por supuesto, significa que puede delegar en CUALQUIER computadora, lo que es potencialmente peligroso, ya que facilitaría a un atacante suplantar una máquina y apoderarse de su cuenta super-privilegiada de administrador de dominio!

- Al ejecutar un comando Remoting, debe especificar el parámetro "-Authentication CredSSP". También debe utilizar el parámetro -Credential y proporcionar un valor DOMINIO\Usuario (se le pedirá la contraseña) - incluso si es el mismo nombre de usuario que utilizó para abrir PowerShell al inicio.

Después de configurar lo anterior, pudimos utilizar Enter-PSSession para pasar de nuestro controlador de dominio a mi servidor miembro y, a continuación, utilizar Invoke-Command para ejecutar un comando en un equipo cliente: la conexión ilustrada en la figura 2.34.

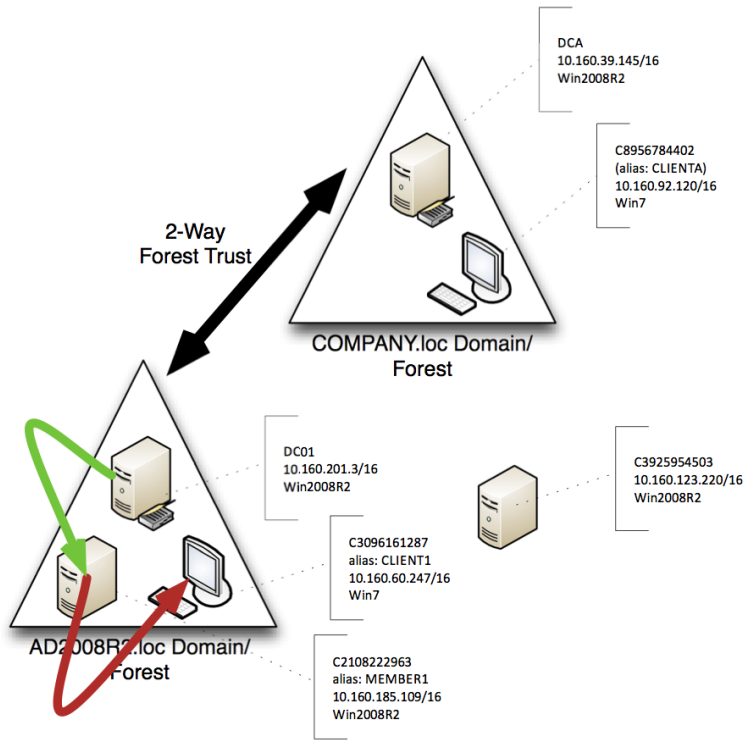

image041.png

Figura 2.34: Las conexiones para la prueba del segundo salto

¿Le parece tedioso y tedioso hacer todos esos cambios? Hay un camino más rápido. En el equipo de origen, ejecute esto:

```
1 Enable-WSManCredSSP -Role Client -Delegate name
```

Donde "nombre" es el nombre de los equipos que planea remitir al siguiente. Esto puede ser un comodín, como *, o un comodín parcial, como *.AD2008R2.loc. A continuación, en el equipo intermedio (aquél al que delegará sus credenciales), ejecute lo siguiente:

```
1 Enable-WSManCredSSP -Role Server
```

Entre ellos, estos dos comandos logran casi todos los puntos de configuración que enumeramos anteriormente. La única excepción es que modificarán su política local para permitir una nueva delegación de credenciales, en lugar de modificar la directiva de dominio a través de un GPO. Puede optar por modificar la directiva de dominio usted mismo, utilizando la GPMC, para que esa configuración particular sea más universal.

La solución Kerberos

CredSSP no se considera el protocolo más seguro del mundo (vea https://msdn.microsoft.com/en-us/library/cc226796.aspx). Las credenciales se transmiten en texto claro, lo cual es un problema. Afortunadamente, dentro de un dominio, hay otra forma de habilitar el multi-salto Remoting, utilizando el protocolo nativo Kerberos, que no transmite credenciales. Específicamente, se llama delegación de restricciones Kerberos basada en recursos, Ashley McGlone (@goateePFE) escribió sobre ello[6].

[6] https://blogs.technet.microsoft.com/ashleymcglone/2016/08/30/powershell-remoting-kerberos-double-hop-solved-securely/

Esta técnica básica funciona desde Windows Server 2003, por lo que debería cubrir cualquier situación que necesite. La idea aquí es que se puede permitir a una máquina delegar credenciales específicas para servicios en otra máquina. Windows Server 2012 simplificó el diseño de esta técnica, anteriormente indocumentada y compleja, por lo que nos centraremos en eso. Por lo tanto, cada máquina involucrada necesita tener Windows Server 2012 o posterior, incluyendo al menos un controlador de dominio Win2012 en el dominio. También necesitará un equipo Windows de última generación con el RSAT instalado (he usado Windows 10). Sabrá que tiene la versión de ejecución si puede ejecutar esto:

```
Import-Module ActiveDirectory
Get-Command -ParameterName PrincipalsAllowedToDelegateToA\
ccount
```

Y obtener algunos resultados de vuelta. Si no obtiene nada, tienes una versión anterior del RSAT - necesitara una más nueva, lo que probablemente requerirá una versión más reciente de Windows en su cliente. Por lo tanto, supongamos que estamos en ClientA, queremos conectarnos a ServerB y que delegue una credencial a través de un segundo salto a ServerC.

```
$ClientA = $env:COMPUTERNAME
$ServerB = Get-ADComputer -Identity ServerB
$ServerC = Get-ADComputer -Identity ServerC

Set-ADComputer -Identity $ServerC -PrincipalsAllowedToDel\
egateToAccount $ServerB
```

Esto permite que ServerC acepte una credencial delegada de ServerB. Si está prestando atención, esto significa que *el equipo al final del segundo salto* es lo que se necesita modificar, para que pueda recibir una credencial del intermediario. Además, si ya ha intentado un segundo salto antes de configurar esto, tendrá

que esperar alrededor de 15 minutos para que la "memoria caché incorrecta" de Active Directory expire y permita que todo funcione correctamente. También podría reiniciar ServerB, si está en un laboratorio o algo así.

El -PrincipalsAllowedToDelegateToAccount también puede ser una matriz, como en @($ServerB, $ServerZ, $ ServerX), etc., permitiendo que varios orígenes deleguen una credencial en la cuenta de equipo que está actualizando. Puede hacer este trabajo a través de límites de confianza, también - vea el artículo original de Ashley para aplicar esta técnica.

Trabajar con Endpoints (también conocido como Configuraciones de Sesión)

Como aprendió al principio de esta guía, Remoting está diseñado para trabajar con múltiples puntos finales distintos en un equipo. En la terminología de PowerShell, cada punto final es una configuración de sesión o simplemente una configuración. Cada uno puede ser configurado para ofrecer servicios y capacidades específicos, así como tener restricciones y limitaciones específicas.

Conexión a un punto final diferente

Cuando utiliza un comando como Invoke-Command o Enter-PSSession, normalmente se conecta al punto final predeterminado de un equipo remoto. Eso es lo que hemos hecho hasta ahora. Pero puede ver los otros puntos finales habilitados ejecutando Get-PSSessionConfiguration, como se muestra en la figura 3.1.

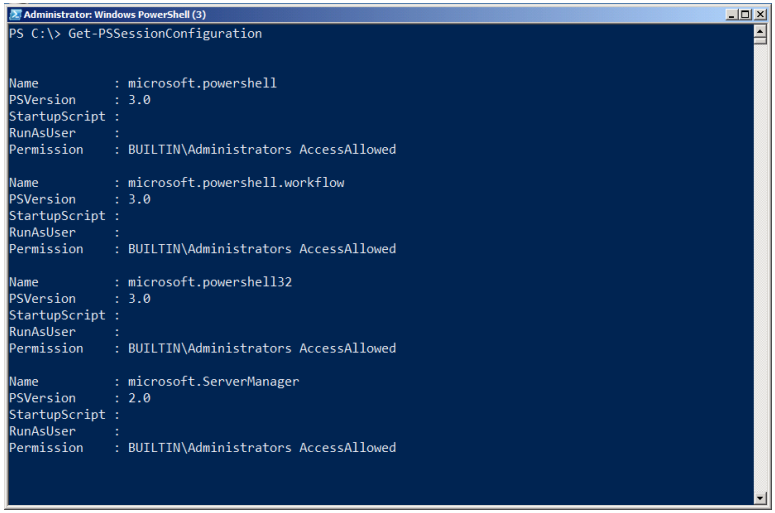

image042.png

Figura 3.1: Listando los puntos finales instalados

Nota: Como señalamos en un capítulo anterior, cada computadora mostrará puntos finales diferentes por defecto. Nuestra salida era de un equipo con Windows Server 2008 R2, que tiene menos puntos finales predeterminados que, por ejemplo, un equipo con Windows 2012.

Cada punto final tiene un nombre, como "Microsoft.PowerShell" o "Microsoft.PowerShell32". Para conectarse a un punto final específico, agregue el parámetro -ConfigurationName al comando Remoting, como se muestra en la Figura 3.2.

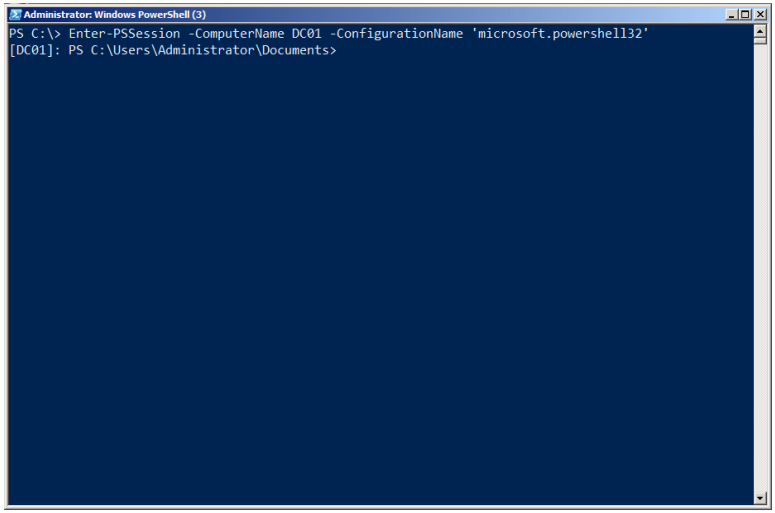

image043.png

Figura 3.2: Conexión a una configuración específica (punto final) por nombre

Creación de un punto de extremo personalizado

Existen varias razones para crear un punto final personalizado (o una configuración):

- Puede tener scripts y módulos de carga automática cada vez que alguien se conecta.
- Puede especificar un descriptor de seguridad (SDDL) que determina quién tiene permiso para conectarse.
- Puede especificar una cuenta alternativa que se utilizará para ejecutar todos los comandos dentro del punto final, en lugar de utilizar las credenciales de los usuarios conectados.

- Puede limitar los comandos que están disponibles para los usuarios conectados, restringiendo así sus capacidades.

Hay dos pasos para configurar un punto final: Crear un archivo de configuración de sesión que definirá las capacidades de los puntos finales y luego registrar ese archivo, que habilita el punto final y define sus configuraciones. La Figura 3.3 muestra la ayuda para el comando New-PSSessionConfigurationFile, que realiza el primero de estos dos pasos.

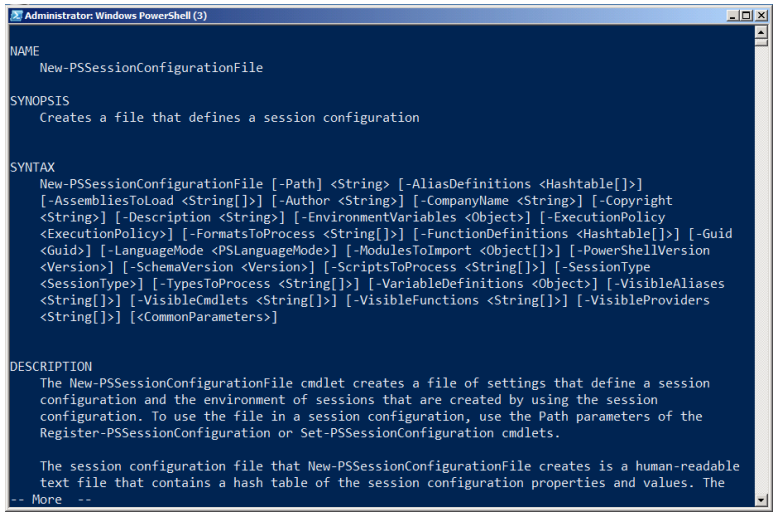

image044.png

Figura 3.3: El comando New-PSSessionConfigurationFile

Aquí algunos de los parámetros que le permiten especificar (revise el archivo de ayuda por los otros parámetros):

- -Path: El único parámetro obligatorio, es la ruta y el nombre de archivo del archivo de configuración que creará. Ingrese un nombre y utilice una extensión .PSSC para el nombre de archivo.

- -AliasDefinitions: Esta es una tabla hash de alias y sus definiciones. Por ejemplo, @ {Name = 'd'; Definition = 'Get-ChildItem'; Options = 'ReadOnly'} definiría el alias d. Utilice una lista separada por comas de estas tablas hash para definir varios alias.
- -EnvironmentVariables: Una tabla hash única de variables de entorno para cargar en el punto final: @{'MyVar'='SERVERShare';'MyOtherVa
- -ExecutionPolicy: Por defecto es Restricted si no especifica otra cosa. Utilice Unrestricted, AllSigned o RemoteSigned. Establece la directiva de ejecución de secuencias de comandos para el punto final.
- -FormatsToProcess y -TypesToProcess: Cada una de estas es una lista separada por comas de la ruta de acceso y los nombres de los archivos a cargar. El primero especifica los archivos .format.ps1xml que contienen definiciones de vista, mientras que el segundo especifica un archivo .ps1xml para el ETS (Extensible Type System) de PowerShell.
- -FunctionDefinitions: Una lista separada por comas de tablas hash, cada una de las cuales define una función para aparecer dentro del punto final. Por ejemplo, @{Name='MoreDir';Options='ReadOnly';
  Dir | more }}
- -LanguageMode: El modo para el lenguaje de script de PowerShell. "FullLanguage" y "NoLanguage" son las opciones. Este último sólo permite ejecutar funciones y Cmdlets. También hay "RestrictedLanguage" que permite un subconjunto muy pequeño del lenguaje de scripting para trabajar - vea la ayuda para más detalles.
- -ModulesToImport: Una lista de nombres de módulos separados por comas para cargar en el punto final. También puede utilizar tablas hash para especificar versiones de módulo específicas. Lea la ayuda completa del comando para obtener más detalles.
- -PowerShellVersion: '2.0' o '3.0', especifica la versión de PowerShell que desea que el punto final utilice. 2.0 sólo se puede especificar si PowerShell v2 se instala independientemente en

el equipo que aloja el punto final (instalando v3 "en la parte superior de" v2 permite que v2 continúe existiendo).

- -ScriptsToProcess: Una lista separada por comas de nombres de rutas y archivos de secuencias de comandos que se ejecutan cuando un usuario se conecta al punto final. Puede usar esto para personalizar el espacio de ejecución del punto final, definir funciones, cargar módulos o hacer cualquier otra cosa que un script pueda hacer. Sin embargo, para ejecutar la directiva de ejecución de secuencias de comandos debe permitir la secuencia de comandos.
- -SessionType: " Empty " no carga nada por defecto, dejándolo a usted libre de cargar lo que quiera a través de scripts o los parámetros de este comando. "Default" carga las extensiones principales de PowerShell normales, además de cualquier otra cosa que haya especificado a través del parámetro. "RestrictedRemoteServer" agrega una lista fija de siete comandos, además de lo que haya especificado. Consulte la ayuda para obtener detalles sobre lo que se ha cargado

Precaución: Algunos comandos son importantes, como Exit-PSSession, que permite a alguien salir de forma limpia de una sesión de Remoting interactiva. RestrictedRemoteServer carga estos, pero Empty no.

-VisibleAliases, -VisibleCmdlets, -VisibleFunctions y -VisibleProviders: estas listas separadas por comas definen cuáles de los alias, cmdlets, funciones y PSProviders disponibles serán visibles para el usuario del punto final. Estos le permiten cargar un módulo completo, pero sólo exponen uno o dos comandos, si lo desea.

Nota: No puede utilizar un punto de terminación personalizado solo para controlar los parámetros a los que un usuario tendrá acceso. Si necesita ese nivel de control, una opción es sumergirse en la programación de .NET Framework, lo que le permite crear una configuración remota más fina. Eso está más allá del alcance de esta guía. También puede crear un punto de extremo personalizado que

sólo incluya funciones de proxy, otra forma de "integrar" comandos incorporados y agregar o eliminar parámetros, pero eso también está fuera del alcance de esta guía.

Una vez que haya creado el archivo de configuración, estará listo para registrarlo. Esto se hace con el comando Register-PSSessionConfiguration, como se muestra en la figura 3.4.

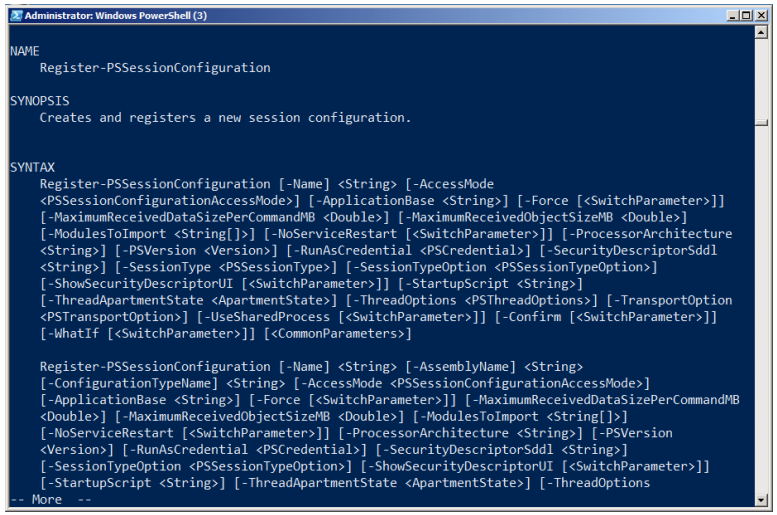

image045.png

Figura 3.4: El comando Register-PSSessionConfiguration

Como puede ver, hay mucho que hacer con este comando. Algunos de los parámetros más interesantes son:

- -RunAsCredential: Permite especificar una credencial que se utilizará para ejecutar todos los comandos dentro del punto final. Proporcionar esta credencial permite a los usuarios conectarse y ejecutar comandos que normalmente no tendrían permiso para ejecutarse. Limitando los comandos disponibles (a través del archivo de configuración de sesión), puede restringir lo que los usuarios pueden hacer con este privilegio elevado.

- -SecurityDescriptorSddl: Le permite especificar quién puede conectarse al punto final. El lenguaje de especificación es complejo. Considere el uso de -ShowSecurityDescriptorUI en su lugar, que muestra un cuadro de diálogo gráfico para establecer los permisos de punto final.
- -StartupScript: Especifica un script para ejecutarse cada vez que se inicia el punto final.

Puede explorar las otras opciones por su cuenta en el archivo de ayuda. Echemos un vistazo a la creación y el uso de uno de estos extremos personalizados. Como se muestra en la figura 3.5, hemos creado una nueva cuenta de usuario de AD para SallyS del departamento de ventas. Sally, por alguna razón, debe ser capaz de enumerar a los usuarios en nuestro dominio de AD - pero eso es todo lo que debe ser capaz de hacer. Su cuenta no tiene permiso para hacerlo.

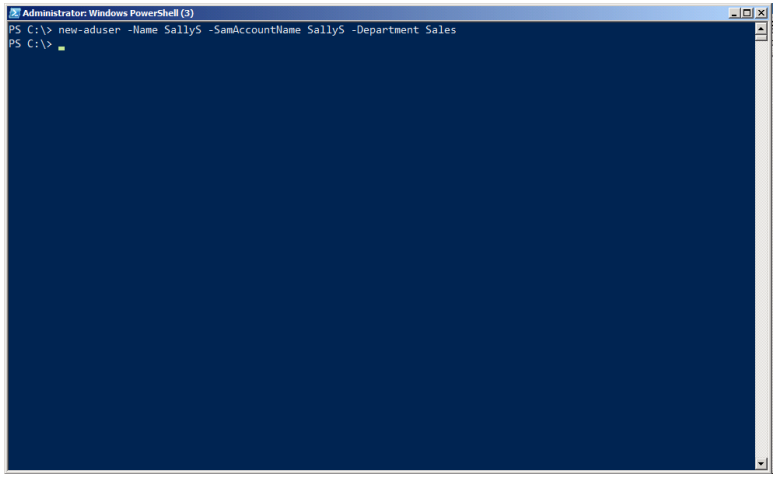

image046.png

Figura 3.5: Creación de una nueva cuenta de usuario de AD para la prueba

La Figura 3.6 muestra la creación del nuevo archivo de configura-

ción de la sesión y el registro de la sesión. Observe que la sesión importará automáticamente el módulo ActiveDirectory, pero sólo hará que el cmdlet Get-ADUser sea visible para Sally. Hemos especificado un tipo de sesión remota restringida, que proporcionará algunos otros comandos clave a Sally. También desactivamos el lenguaje de scripting de PowerShell. Al registrar la configuración, especificamos una credencial "Ejecutar como" (se nos pidió la contraseña), que es la cuenta bajo la qie que todos los comandos ejecutarán.

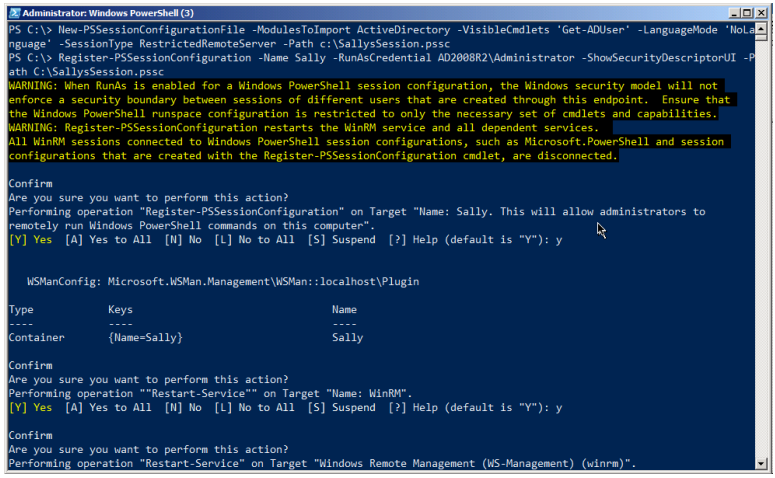

image047.png

Figura 3.6: Creación y registro del nuevo punto final

Debido a que usamos el parámetro "ShowSecurityDescriptorUI", tenemos un cuadro de diálogo como el que se muestra en la figura 3.7. Esta es una manera más fácil de establecer los permisos para quién puede usar este nuevo punto final. Tenga en cuenta que el punto final ejecutará los comandos bajo una cuenta de administrador de dominio, por lo que debemos tener mucho cuidado de a quien realmente dejamos ingresar. Sally necesita, como mínimo, permiso de ejecución y lectura, que ya se le ha dado.

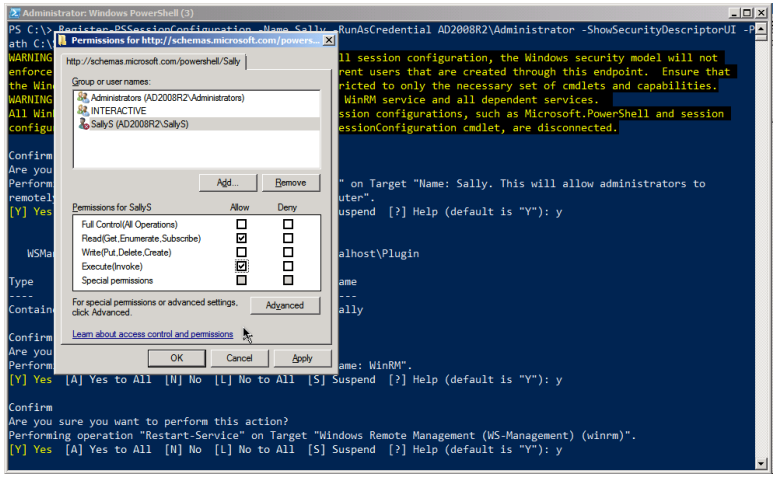

image048.png

Figura 3.7: Configuración de los permisos en el punto final

A continuación, establecer una contraseña para Sally y activar su cuenta de usuario. Todo hasta este punto se ha hecho en el ordenador DC01.AD2008R2.loc. La figura 3.8 se desplaza al equipo cliente de Windows 7 de ese dominio, donde iniciamos sesión con la cuenta de Sally. Como puede ver, no pudo ingresar a la sesión predeterminada en el controlador de dominio. Pero cuando intentó entrar en la nueva sesión especial que creamos sólo para ella, la operación tuvo éxito. También pudo ejecutar Get-ADUser.

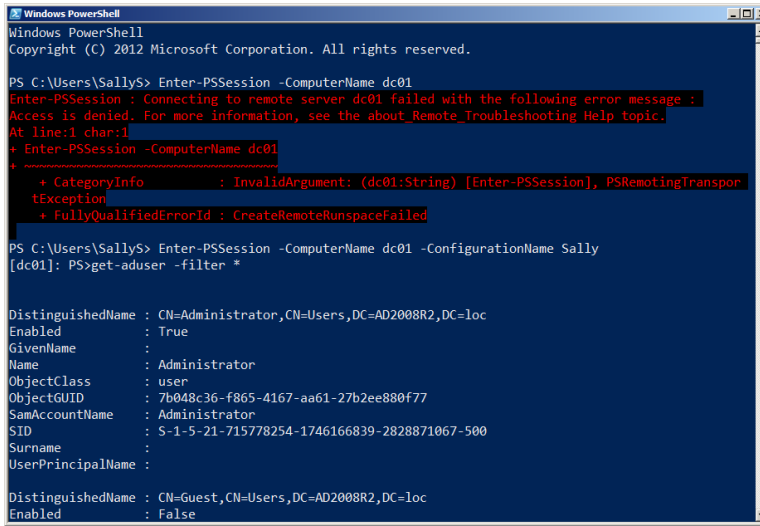

image049.png

Figura 3.8: Prueba del nuevo punto final iniciando sesión como Sally

La Figura 3.9 confirma que Sally tiene un número muy limitado de comandos para ejecutar. Algunos de estos comandos, como Get-Help y Exit-PSSession, son muy importantes para usar el punto final. Otros, como Select-Object, le dan a Sally una cantidad mínima de comodidad no destructiva para que su salida de comandos se vea como ella necesita. Esta lista de comandos (aparte de Get-ADUser) se establece automáticamente cuando se especifica el tipo de sesión "restricted remote" en el archivo de configuración de la sesión.

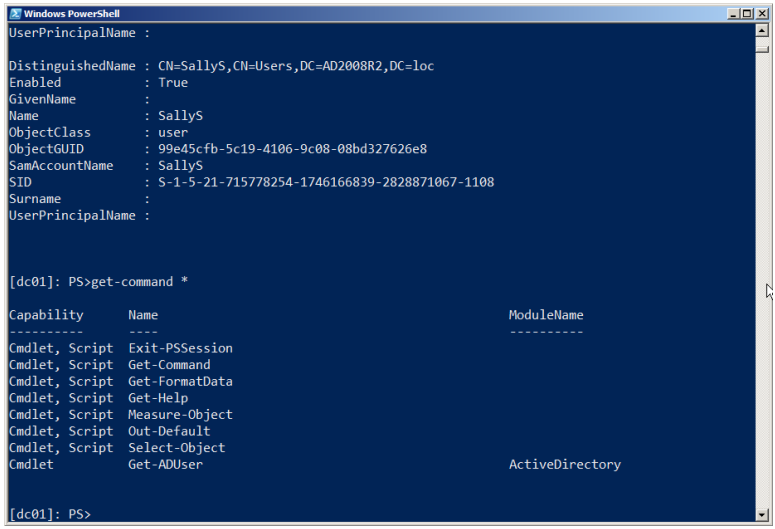

image050.png

Figura 3.9: Solamente ocho comandos, incluido el Get-ADUser que hemos agregado, están disponibles dentro del punto final.

En realidad, es poco probable que un usuario de ventas como Sally estuviera ejecutando comandos en la consola de PowerShell. Lo más probable es que utilizara alguna aplicación basada en GUI que ejecutara los comandos "detrás de escenas". De cualquier manera, nos hemos asegurado de que ella tiene exactamente la funcionalidad que necesita para hacer su trabajo, y nada más.

Precauciones de seguridad con puntos finales personalizados

Cuando crea un archivo de configuración de sesión personalizado, tal como lo ha visto, puede configurar su modo de idioma. El modo de idioma determina qué elementos del lenguaje de secuencias de comandos de PowerShell están disponibles en el punto final y

el modo de lenguaje puede ser como una laguna. Con el modo de lenguaje "Full", obtendrá todo el lenguaje de secuencias de comandos, incluidos los bloques de secuencia de comandos. Un bloque de secuencia de comandos es cualquier trozo ejecutable de código de PowerShell contenido dentro de {curly brackets}. Ellos son la escapatoria. Siempre que permita el uso de bloques de script, puede ejecutar cualquier comando legal, incluso si su punto final usó -VisibleCmdlets o -VisibleFunctions u otro parámetro para limitar los comandos en el punto final.

En otras palabras, si registra un punto final que utiliza -VisibleCmdlets para exponer sólo Get-ChildItem, pero crea el archivo de configuración de sesión del punto final para que tenga el modo de lenguaje Full, cualquier bloque de secuencia dentro del punto final puede utilizar cualquier comando. Alguien podría ejecutar:

```
PS C:\> & { Import-Module ActiveDirectory; Get-ADUser
-filter \* | Remove-ADObject }
```

¡Eek! Esto puede ser especialmente peligroso si ha configurado el punto final para utilizar una credencial de RunAs para ejecutar comandos bajo privilegios elevados. También es algo fácil dejar que esto suceda por error, ya que se establece el modo de idioma cuando se crea el nuevo archivo de configuración de sesión (New-PSSessionConfigurationFile), no cuando se registra la sesión (Register-PSSessionConfiguration). Por lo tanto, si está utilizando un archivo de configuración de sesión creado por otra persona, abra y confirme su modo de idioma antes de usarlo.

Puede evitar este problema estableciendo el modo de idioma en NoLanguage, que deshabilita los bloques de secuencia de comandos y el resto del lenguaje de secuencias de comandos. O bien, vaya a RestrictedLanguage, que bloquea bloques de secuencia de comandos al mismo tiempo que permite el uso de algunos operadores básicos si desea que los usuarios del punto final puedan hacer filtrado y comparaciones básicas.

Es importante distinguir que esto no es un error, pues el comporta-

miento que estamos describiendo aquí es por diseño. Puede ser un problema si no lo sabe o no entiende lo que está haciendo.

Nota: Muchas gracias al compañero MVP Aleksandar Nikolic por ayudarme a entender la lógica de esta laguna!

Diagnóstico y solución de problemas

La solución de problemas y diagnóstico de Remoting puede ser una de las tareas más difíciles que un administrador tenga que tratar. Cuando funciona Remoting, funciona. Cuando no lo hace, a menudo es difícil saber por qué. Afortunadamente, PowerShell v3 y su implementación adjunta de Remoting tienen mensajes de errores mucho más claros y descriptivos que las anteriores versiones. Sin embargo, incluso v2 incluye un módulo indocumentado y poco valorado llamado PSDiagnostics, que está diseñado específicamente para facilitar la solución de problemas Remoting. Esencialmente, el módulo le permite activar la información detallada del registro de seguimiento antes de intentar iniciar una conexión remota. A continuación, puede utilizar esa información de registro detallada para obtener una mejor idea de dónde está fallando Remoting.

Ejemplos de diagnósticos

Para los siguientes escenarios, comenzaremos por importar el módulo PSDiagnostics (tenga en cuenta que se implementa como un módulo de script y requiere una directiva de ejecución que le permita ejecutarse, como RemoteSigned o Unrestricted). La Figura 4.1 muestra la ejecución del comando Enable-PSWSManCombinedTrace, que inicia el registro de diagnóstico extendido.

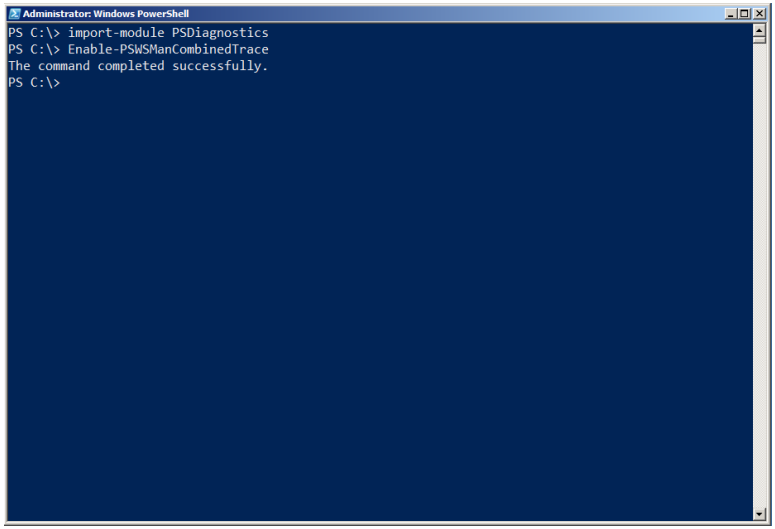

image051.png

Figura 4.1: Carga del módulo de diagnóstico e inicio de un rastreo

Para cada escenario, seguimos ejecutando uno o más comandos que utilizan Remoting, como se muestra en la figura 4.2. A continuación, desactivamos la traza ejecutando Disable-PSWSManCombinedTrace, de modo que el registro sólo contendrá los detalles de ese intento en particular (borramos el registro entre intentos, para que cada escenario proporcione un nuevo registro de diagnósticos).

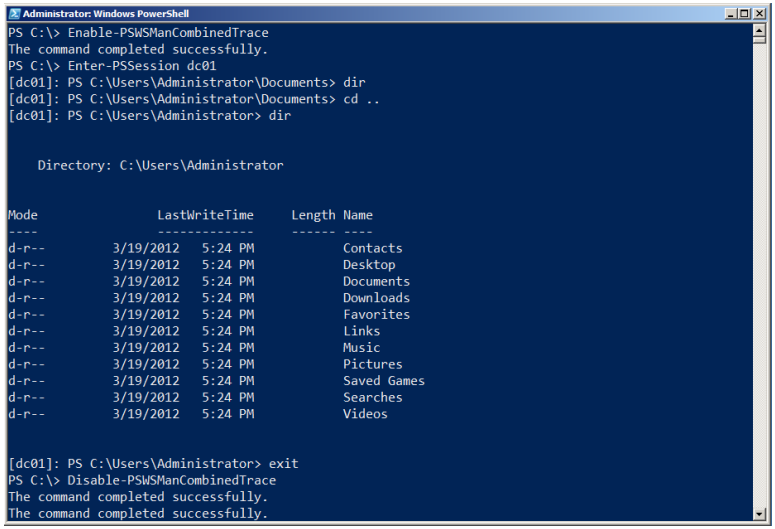

image052.png

Figura 4.2: Ingresando a una sesión y ejecutando un comando

Finalmente, como se muestra en la figura 4.3, recuperamos los mensajes del registro. En los escenarios que siguen, proporcionaremos una versión detallada de éstos. Tenga en cuenta que típicamente truncaremos gran parte de la salida para poder centrarnos en las partes más significativas. Observe también que hay algo de diferencia al leer la información de la arquitectura del registro de eventos, como lo hacemos en la figura 4.3, y leer el archivo de seguimiento .EVT directamente, como lo haremos en algunos de nuestros escenarios. Este último proporcionará información combinada de diferentes registros, lo que a veces puede ser más útil.

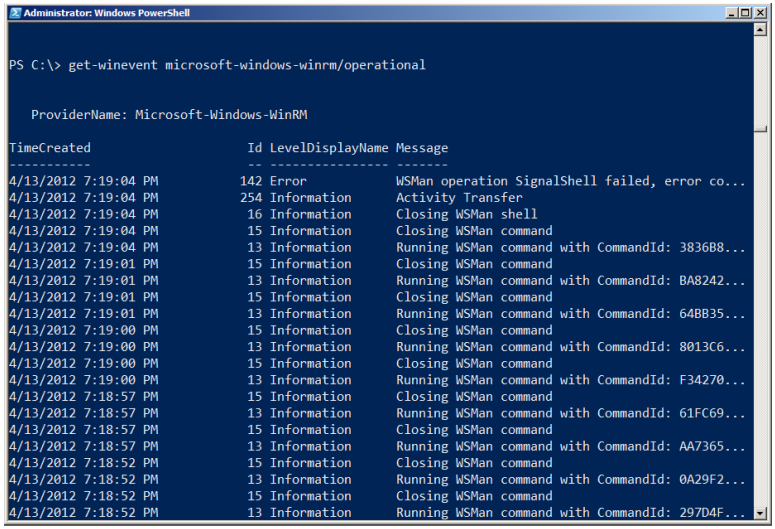

image053.png

Figura 4.3: Examinar la información de diagnóstico registrada

También vamos a hacer uso del Microsoft-Windows-WinRM/analytic log, que normalmente no contiene información fácilmente legible por humanos. Para utilizar el contenido del registro, utilizaremos un utilitario interno de Microsoft (que se nos ha dado permiso para distribuir y que encontraremos en la página de descargas en http://ConcentratedTech.com) para convertir el contenido del registro en algo que podemos leer.

La información de rastreo se almacena en la carpeta de instalación de PowerShell (ejecute cd $PSHome para llegar allí y luego cambie a la carpeta Traces). La extensión del nombre de archivo es .ETL y puede usar Get-WinEvent -Path Filename.etl para leer un archivo en particular. El comando Construct-PSRemoteDataObject, incluido en el archivo ZIP al que hacemos referencia, puede traducir partes de la propiedad Message del registro analítico en texto (Analytic log's) legible para humanos. Un script de demostración incluido en el archivo ZIP evidencia cómo utilizarlo. Como se muestra en la figura 4.4, hemos utilizado "dot-sourcing" con el

archivo Construct-PSRemoteDataObject.ps1 en nuestro shell para obtener acceso a los comandos que expone.

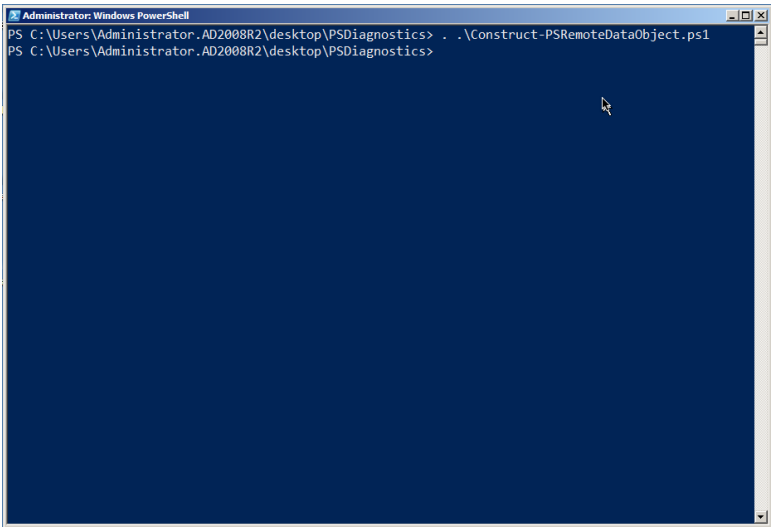

image054.png

Figura 4.4 Dot-sourcing del script Construct-PSRemoteDataObject.ps1

También eliminamos el contenido de C:WindowsSystem32WindowsPowerShell\v1.0 antes de iniciar cada uno de los ejemplos siguientes.

Una conexión remota perfecta

Para esta conexión, pasamos del equipo cliente de Windows 7 en el dominio AD2008R2 al controlador de dominio DC01. En la DC, cambiamos a la carpeta C:\, ejecutamos el comando dir y terminamos la sesión. La Figura 4.5 muestra todo el escenario.

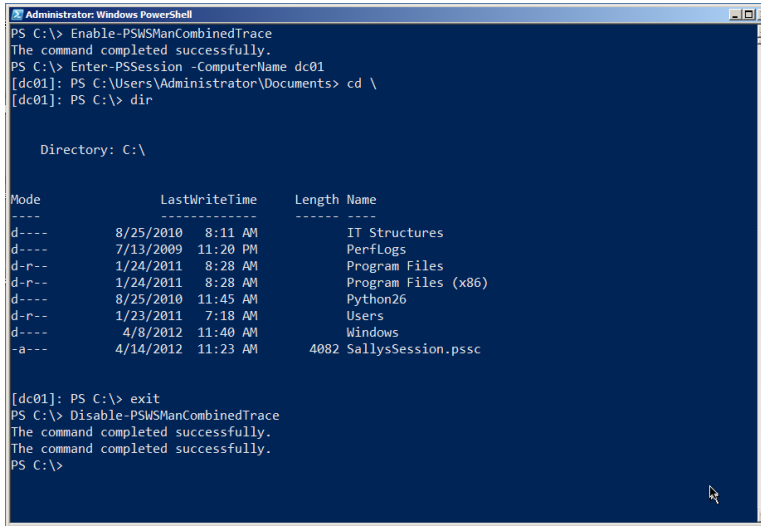

image055.png

Figura 4.5: Ejemplo completo del escenario

A continuación, leemos el registro en orden cronológico. Tiene que ser cuidadoso. Ejecutando Enable-PSWSManCombinedTrace y Disable-PSWSManCombined se crearan eventos de registro de ellos mismos. A menudo ejecutaremos el comando Enable, y luego esperaremos algunos minutos para hacer cualquier cosa con Remoting. De esa manera, podemos establecer por la marca de tiempo en el registro cuando comenzó el tráfico "real. Esperaremos unos minutos más antes de ejecutar el comando Disable, para que podamos saber fácilmente cuándo finalizó el tráfico de registro "real". También tenga en cuenta que vamos a obtener información de dos registros, WinRM y PowerShell, aunque leer el archivo .ETL con Get-WinEvent tomará todo en secuencia.

Nota: hemos experimentado problemas al utilizar Get-WinEvent en PowerShell v3 en máquinas "non-US English". Si tiene problemas, considere ejecutar el comando desde PowerShell v2 o utilice la aplicación GUI Event Viewer para ver el registro de eventos.

La conexión comienza con (en este ejemplo) Enter-PSSession y la resolución de nombres, como se muestra en la figura 4.6

```
4/14/2012 3:03:39 PM Command Enter-PSSession is Started.


                    Context:
                            Severity = Informational
                            Host Name = ConsoleHost
                            Host Version = 3.0
                            Host ID = 5daaddbe-8c9d-4ee4-ab44-fac774eedc6f
                            Engine Version = 3.0
                            Runspace ID = f47408cf-bd95-4ced-ace8-e799421d646b
                            Pipeline ID = 294
                            Command Name = Enter-PSSession
                            Command Type = Cmdlet
                            Script Name =
                            Command Path =
                            Sequence Number = 89
                            User = AD2008R2\Administrator
                            Shell ID = Microsoft.PowerShell


                    User Data:


4/14/2012 3:03:39 PM ComputerName resolved to localhost
4/14/2012 3:03:39 PM ComputerName resolved to dc01
4/14/2012 3:03:39 PM ComputerName resolved to dc01
4/14/2012 3:03:39 PM ComputerName resolved to dc01
```

image056.png

Figura 4.6: Inicio de la conexión remota

WinRM tiene que "iniciar" un espacio de ejecución (esencialmente, un proceso de PowerShell) en el equipo remoto. Esto incluye establecer varias opciones para la configuración regional, la temporización, etc, como se muestra en la figura 4.7.

```
4/14/2012 3:03:39 PM Creating Runspace object
                        Instance Id: cd32125b-0290-4887-89a9-910ca224b3f7
4/14/2012 3:03:39 PM Creating RunspacePool object
                        InstanceId 4358d585-0eab-47ef-a0e6-4b98e71f34ab
                        MinRunspaces 1
                        MaxRunspaces 1
4/14/2012 3:03:39 PM Creating WSMan Session. The connection string is: dc01/wsman?PSVersion=3.0
4/14/2012 3:03:39 PM WSMan Create Session operation completed successfuly
4/14/2012 3:03:39 PM Getting WSMan Session Option (29) - INVALID_SESSION_OPTION.
4/14/2012 3:03:39 PM Getting WSMan Session Option (11) - WSMAN_OPTION_MAX_RETRY_TIME.
4/14/2012 3:03:39 PM Setting WSMan Session Option (26) - WSMAN_OPTION_UI_LANGUAGE with value
                    (en-US) completed successfully.
4/14/2012 3:03:39 PM Setting WSMan Session Option (25) - WSMAN_OPTION_LOCALE with value (en-US)
                    completed successfully.
4/14/2012 3:03:39 PM Setting WSMan Session Option (1) - WSMAN_OPTION_DEFAULT_OPERATION_TIMEOUTMS
                    with value (180000) completed successfully.
4/14/2012 3:03:39 PM Setting WSMan Session Option (12) - WSMAN_OPTION_TIMEOUTMS_CREATE_SHELL with
                    value (180000) completed successfully.
4/14/2012 3:03:39 PM Setting WSMan Session Option (17) - WSMAN_OPTION_TIMEOUTMS_CLOSE_SHELL with
                    value (60000) completed successfully.
4/14/2012 3:03:39 PM Setting WSMan Session Option (16) - WSMAN_OPTION_TIMEOUTMS_SIGNAL_SHELL with
                    value (60000) completed successfully.
4/14/2012 3:03:39 PM Opening RunspacePool
4/14/2012 3:03:39 PM Runspace state changed to Opening
```

image057.png

Figura 4.7: Inicio del espacio de ejecución remota

Esto puede tomar un tiempo. Eventualmente, verá que WinRM comienza a enviar “trozos”, que son comunicaciones empaquetadas. Estos son enviados a través del Protocolo de Acceso a Objetos Simples, por lo que esperamos ver muchas referencias “SOAP” (WS-MAN es un servicio Web, recuerde, y SOAP es el lenguaje de comunicaciones de los servicios Web). La Figura 4.8 muestra un par de estos trozos de 1500 bytes. Tenga en cuenta que la carga real es más o menos ilegible.

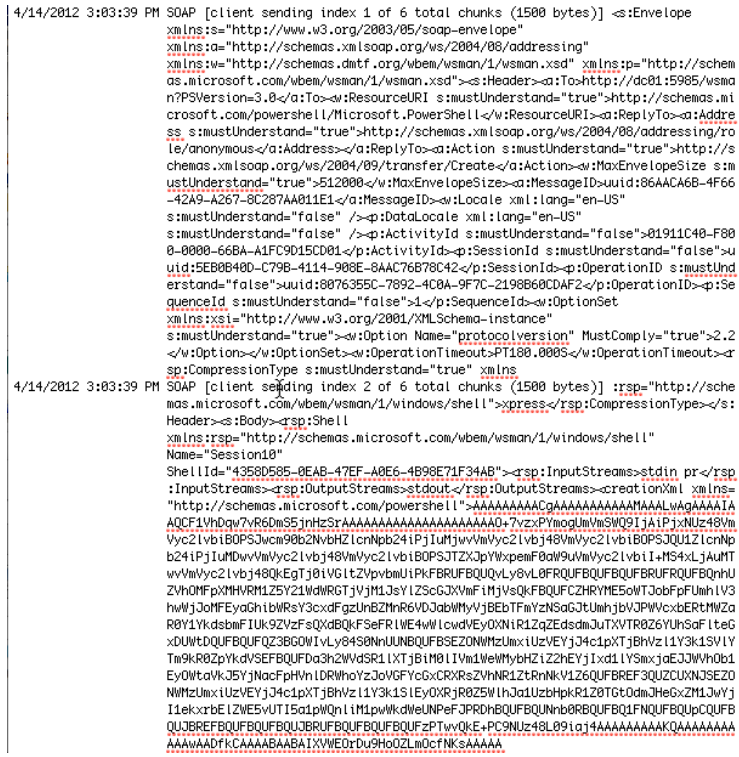

image058.png

Figura 4.8: Los datos comienzan a transferirse a través de la conexión

Este texto ilegible es lo que el comando Construct-PSRemoteDataObject puede traducir. Por ejemplo, los mensajes de "envío" tienen un ID de evento de 32868. Buscando sólo esos eventos podemos ver lo que se está enviando, como se muestra en la figura 4.9.

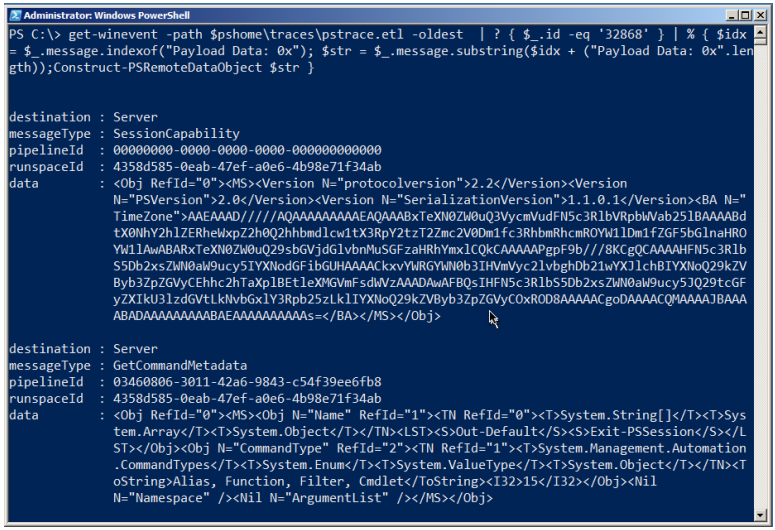

image059.png

Figura 4.9: Traducir los datos enviados

En este caso, el cliente estaba preguntando al servidor (que está listado como el destino) acerca de sus capacidades y algunos metadatos en el comando Exit-PSSession (que es el segundo mensaje). Así es como el cliente calcula con qué tipo de servidor está hablando, y otra información importante de manera preliminar. En este punto el cliente sabe qué versión del protocolo de serialización se utilizará para enviar datos de ida y vuelta, en qué zona horaria está el servidor y otros detalles.

Nota: Event ID 32868 es tráfico de cliente a servidor; ID 32867 representa tráfico de servidor a cliente. El uso de estos dos IDs junto con Construct-PSRemoteDataObject puede revelar la mayoría del trafico de sesión una vez que se establece la conexión.

Continuando. Como se muestra en la figura 4.10, verá una autenticación de ida y vuelta, durante la cual se pueden esperar algunos errores. El sistema acabará por superarlo y, como se muestra, comenzará a recibir trozos de datos del servidor.

```
4/14/2012 3:03:39 PM An error was encountered while processing an operation.
                     Error Code: 11001
4/14/2012 3:03:39 PM The chosen authentication mechanism is Kerberos
4/14/2012 3:03:39 PM Sending the request for operation CreateShell to destination machine and port
                     dc01:5985
4/14/2012 3:03:39 PM An error was encountered while processing an operation.
                     Error Code: 11001
4/14/2012 3:03:39 PM The chosen authentication mechanism is Kerberos
4/14/2012 3:03:39 PM Received the response from Network layer; status: 200 (HTTP_STATUS_OK)
4/14/2012 3:03:39 PM Received the response from Network layer; status: 200 (HTTP_STATUS_OK)
4/14/2012 3:03:39 PM Activity Transfer
4/14/2012 3:03:39 PM Activity Transfer
4/14/2012 3:03:39 PM SOAP [client receiving index 1 of 2 total chunks (3000 bytes)] <s:Envelope
                     xml:lang="en-US" xmlns:s="http://www.w3.org/2003/05/soap-envelope"
                     xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing"
                     xmlns:x="http://schemas.xmlsoap.org/ws/2004/09/transfer"
                     xmlns:w="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
                     xmlns:rsp="http://schemas.microsoft.com/wbem/wsman/1/windows/shell" xmlns:p="h
                     ttp://schemas.microsoft.com/wbem/wsman/1/wsman.xsd"><s:Header><a:Action>http:/
                     /schemas.xmlsoap.org/ws/2004/09/transfer/CreateResponse</a:Action><a:MessageID
                     >uuid:67E26C83-FCD7-41EA-9B26-636BBE961791</a:MessageID><p:OperationID s:mustU
                     nderstand="false">uuid:8076355C-7892-4C0A-9F7C-2198B60CDAF2</p:OperationID><p:
```

image060.png

Figura 4.10: Obtención de la autenticación

Una cantidad bastante sorprendente de datos de ida y vuelta puede ocurrir a medida que las dos computadoras intercambian y comparten información sobre el otro y cómo trabajan, y así sucesivamente. Vamos a cambiar nuestra salida del registro de eventos, ahora, para incluir números de ID de evento, porque pueden ser muy útiles al intentar obtener datos específicos. En este punto, el registro consistirá principalmente en el cliente que envía comandos y el servidor que envía los resultados. Esto es más legible cuando se utiliza Construct-PSRemoteDataObject, así que aquí están los datos "de aquí para allá": Primero aparece la declaración del cliente y de sus capacidades de sesión:

```
  destination : Server messageType : SessionCapability
pipelineId : 00000000-0000-0000-0000-000000000000 runspaceId
: 4358d585-0eab-47ef-a0e6-4b98e71f34ab data : <Obj RefId="0"><MS><Version
N="protocolversion">2.2</Version><Version N="PSVersion">2.0</Version><Vers
N="SerializationVersion">1.1.0.1</Version><BA N="TimeZon e">AAEAAAD/////AQ
3RlbVRpbWVab25lBAAAABdtX0NhY2hlZERheWxpZ2h0Q2hhbmdlcw1tX 3RpY2tzT2Zmc2V0Dm
wABARxTeXN0ZW0uQ29sbGVjdGlvbnMuSGFzaHRhYmxlCQkCAAAAAPgpF 9b///8KCgQCAAAAHF
AAACkxvYWRGYWN0b3IHVmVyc2lvbghDb21wYXJlchBIYXNoQ29kZVByb 3ZpZGVyCEhhc2hTaX
S5Db2xsZWN0aW9ucy5JQ29tcGFyZXIkU3lzdGVtLkNvbGxlY3Rpb25zL klIYXNoQ29kZVByb3
BADAAAAAAAAABAEAAAAAAAAAs=</BA></MS></Obj>
```

Entonces el servidor:

```
  destination  :  Client  messageType  :  SessionCapability
pipelineId : 00000000-0000-0000-0000-000000000000 runspaceId
: 00000000-0000-0000-0000-000000000000 data : <Obj RefId="0"><MS><Version
N="protocolversion">2.2</Version><Version N="PSVersion">2.0</Version><Vers
N="SerializationVersion">1.1.0.1</Version></MS></Obj>
```

A continuación se muestra el objeto $PSVersionTable del servidor, que lista varias informaciones de control de versiones:

```
 destination : Client messageType : ApplicationPrivateData
pipelineId : 00000000-0000-0000-0000-000000000000 runspaceId
: 4358d585-0eab-47ef-a0e6-4b98e71f34ab data : <Obj RefId="0"><MS><Obj
N="ApplicationPrivateData" RefId="1"><TN RefId="0"><T>System.Management.Au
PSPrimitiveDictionary</T><T>System.Collections.Hashtable </T><T>System.Obj
N="Key">PSVersionTable</S><Obj  N="Value"  RefId="2"><TNRef
RefId="0" /><DCT><En><S N="Key">PSVersion</S><Version N="Value">2.0</Versio
N="Key">PSCompatibleVersions</S><Obj N="Value" RefId="3"><TN
RefId="1"><T>System.Version[]</T><T>System .Array</T><T>System.Object</T><
rsion><Version>2.0</Version><Version>3.0</Version></LST> </Obj></En><En><S
N="Key">BuildVersion</S><Version N="Value">6.2.8314.0</Version></En><En><S
N="Key">PSRemotingProtocolVersion</S><Version N="Value">2.2</Version></En>
N="Key">WSManStackVersion</S><Version N="Value">3.0</Version></En><En><S
N="Key">CLRVersion</S><Version N="Value">4.0.30319.261</Version></En><En><
N="Key">SerializationVersion</S><Version N="Value">1.1.0 .1</Version></En>
>
```

A continuación, el servidor envía información sobre el espacio de ejecución que se utilizará:

```
 destination : Client messageType : RunspacePoolStateInfo
pipelineId : 00000000-0000-0000-0000-000000000000 runspaceId
: 4358d585-0eab-47ef-a0e6-4b98e71f34ab data : <Obj RefId="0"><MS><I32
N="RunspaceState">2</I32></MS></Obj>
```

El cliente envía información sobre su comando Exit-PSSession:

```
  destination  :  Server  messageType  :  GetCommandMetadata
```

```
pipelineId : 03460806-3011-42a6-9843-c54f39ee6fb8 runspaceId
: 4358d585-0eab-47ef-a0e6-4b98e71f34ab data : <Obj RefId="0"><MS><Obj
N="Name" RefId="1"><TN RefId="0" ><T>System.String[]</T><T>System.Array</T>
ct</T></TN><LST><S>Out-Default</S><S>Exit-PSSession</S>< /LST></Obj><Obj
N="CommandType" RefId="2"><TN RefId="1"> <T>System.Management.Automation.Co
m.Enum</T><T>System.ValueType</T><T>System.Object</T></T N><ToString>Alias
Function, Filter, Cmdlet</ToString><I32>15</I32></Obj><Nil
N="Namespace" /><Nil N="ArgumentList" /></MS></Obj>
```

Un poco más adelante veremos el resultado del comando CD C:\, que un nuevo mensaje de PowerShell que refleja la nueva ubicación de la carpeta:

```
 destination : Client messageType : PowerShellOutput pipelineId
: c913b8ae-2802-4454-9d9b-926ca6032018 runspaceId : 4358d585-0eab-47ef-a0e6
data : <S>PS C:\&gt; </S>
```

A continuación, veremos la salida del comando Dir. El primer bit define los encabezados de columna para Mode, LastWriteTime, Length, Name y así sucesivamente. Todo esto se envía a nuestro cliente - solo incluiremos las primeras líneas, cada una de las cuales aparece en su propio bloque:

```
 destination : Client messageType : RemoteHostCallUsingPowerShellHost
pipelineId : c259c891-516a-46a7-b287-27c96ff86d5b runspaceId
: 4358d585-0eab-47ef-a0e6-4b98e71f34ab data : <Obj RefId="0"><MS><I64
N="ci">-100</I64><Obj N="mi" RefId="1"><TN RefId="0"><T>System.Management.A
Remoting.RemoteHostMethodId</T><T>System.Enum</T><T>Syst em.ValueType</T><
Line2</ToString><I32>16</I32></Obj><Obj N="mp" RefId="2"><TN
RefId="1"><T>System.Collections.ArrayList< /T><T>System.Object</T></TN><LS
LastWriteTime Length Name </S></LST></Obj></MS></Obj> destination
: Client messageType : RemoteHostCallUsingPowerShellHost
pipelineId : c259c891-516a-46a7-b287-27c96ff86d5b runspaceId
: 4358d585-0eab-47ef-a0e6-4b98e71f34ab data : <Obj RefId="0"><MS><I64
N="ci">-100</I64><Obj N="mi" RefId="1"><TN RefId="0"><T>System.Management.A
Remoting.RemoteHostMethodId</T><T>System.Enum</T><T>Syst em.ValueType</T><
Line2</ToString><I32>16</I32></Obj><Obj N="mp" RefId="2"><TN
RefId="1"><T>System.Collections.ArrayList< /T><T>System.Object</T></TN><LS
------------- ------ ---- </S></LST></Obj></MS></Obj> destination
```

```
: Client messageType : RemoteHostCallUsingPowerShellHost
pipelineId : c259c891-516a-46a7-b287-27c96ff86d5b runspaceId
: 4358d585-0eab-47ef-a0e6-4b98e71f34ab data : <Obj RefId="0"><MS><I64
N="ci">-100</I64><Obj N="mi" RefId="1"><TN RefId="0"><T>System.Management.A
Remoting.RemoteHostMethodId</T><T>System.Enum</T><T>Syst em.ValueType</T><
Line2</ToString><I32>16</I32></Obj><Obj N="mp" RefId="2"><TN
RefId="1"><T>System.Collections.ArrayList< /T><T>System.Object</T></TN><LS
8/25/2010 8:11 AM IT Structures </S></LST></Obj></MS></Obj>
destination : Client messageType : RemoteHostCallUsingPowerShellHost
pipelineId : c259c891-516a-46a7-b287-27c96ff86d5b runspaceId
: 4358d585-0eab-47ef-a0e6-4b98e71f34ab data : <Obj RefId="0"><MS><I64
N="ci">-100</I64><Obj N="mi" RefId="1"><TN RefId="0"><T>System.Management.A
Remoting.RemoteHostMethodId</T><T>System.Enum</T><T>Syst em.ValueType</T><
Line2</ToString><I32>16</I32></Obj><Obj N="mp" RefId="2"><TN
RefId="1"><T>System.Collections.ArrayList< /T><T>System.Object</T></TN><LS
7/13/2009 11:20 PM PerfLogs </S></LST></Obj></MS></Obj>
```

Finalmente, el comando finaliza y recibimos el "prompt" de nuevo:

```
destination : Client messageType : PowerShellOutput pipelineId
: f5c8bc7a-ec54-4180-b2d4-86479f9ea4b9 runspaceId : 4358d585-0eab-47ef-a0e6
data : <S>PS C:\&gt; </S>
```

También verá intercambios periódicos sobre el estado de la tubería (pipeline) - esto indica que el comando ha finalizado

```
destination : Client messageType : PowerShellStateInfo
pipelineId : f5c8bc7a-ec54-4180-b2d4-86479f9ea4b9 runspaceId
: 4358d585-0eab-47ef-a0e6-4b98e71f34ab data : <Obj RefId="0"><MS><I32
N="PipelineState">4</I32></MS></Obj>
```

Definitivamente hay una gran cantidad de datos que pasan de un lado a otro, pero es posible entenderlo usando estas herramientas. Francamente, la mayoría de los problemas de Remoting se producen durante la fase de conexión, es decir, que una vez se haya conectado, es bastante probable que no tenga más problemas. Así que en los próximos escenarios, nos centraremos en errores de conexión específicos.

Nota: Para borrar el registro y prepararse para una nueva traza, usualmente eliminamos los archivos .ETL y entramos en el Visor de sucesos para borrar los registros de Applications and Services Logs > Microsoft > Windows > Windows Remote Management. Si está recibiendo errores al ejecutar Enable-PSWSManCombinedTrace, una de esas dos tareas probablemente no se ha completado.

Problema de conexión: puerto bloqueado

La Figura 4.11 muestra lo que sucede cuando intenta conectarse a una computadora y el puerto necesario (5985 por defecto) no está abierto. Veamos cómo esto aparece en el registro. Tenga en cuenta que estamos asumiendo que ya ha comprobado el nombre del equipo y se aseguró de que se resuelve a la dirección IP correcta. Lo que está viendo es definitivamente un puerto bloqueado (porque lo configuramos de esa manera) en este ejemplo.

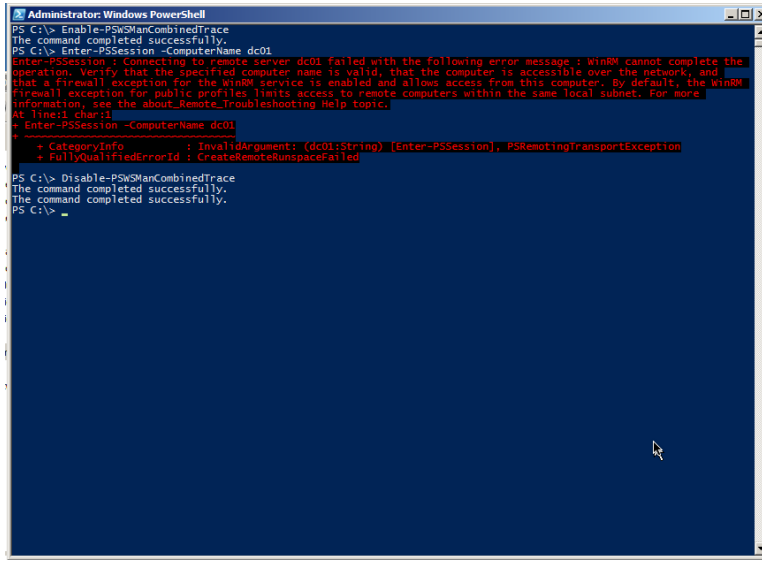

image061.png

Figura 4.11: Error de conexión debido a un cortafuegos u otro

problema de bloqueo de puertos.

La Figura 4.12 muestra que hemos resuelto satisfactoriamente el nombre del equipo. Encontramos que las pruebas con Enter-PSSession son las más fáciles, porque es muy sencillo detectar ese comando en el registro y ver cuándo empiezan los datos de registro "reales".

```
7937 4/14/2012 4:01:11 PM Command Enter-PSSession is Started.


                          Context:
                                  Severity = Informational
                                  Host Name = ConsoleHost
                                  Host Version = 3.0
                                  Host ID =
                          8fd53e17-bd16-456a-8c6e-174acb89ce8c
                                  Engine Version = 3.0
                                  Runspace ID =
                          f42d6bb0-43d6-4f7d-81e9-376113a63428
                                  Pipeline ID = 66
                                  Command Name = Enter-PSSession
                                  Command Type = Cmdlet
                                  Script Name =
                                  Command Path =
                                  Sequence Number = 255
                                  User = AD2008R2\Administrator
                                  Shell ID = Microsoft.PowerShell


                          User Data:

12035 4/14/2012 4:01:11 PM ComputerName resolved to localhost
12035 4/14/2012 4:01:11 PM ComputerName resolved to dc01
12035 4/14/2012 4:01:11 PM ComputerName resolved to dc01
12035 4/14/2012 4:01:11 PM ComputerName resolved to dc01
```

image062.png

Figura 4.12: Inicio del intento de conexión

Tenga en cuenta que gran parte del tráfico de registro inicial sigue siendo WinRM hablando por sí mismo mientras se alista para el intento de conexión real. Simplemente continúe desplazándose hasta que comience a ver las indicaciones de problemas. La Figura 4.13 muestra un tiempo de espera - nunca una buena señal - y el mensaje de error generado por WinRM. Como puede ver, esto es exactamente lo que tenemos en la pantalla, por lo que PowerShell no está ocultándonos nada.

```
 138 4/14/2012 4:01:34 PM The client got a timeout from the network layer
                          (ERROR_WINHTTP_TIMEOUT)
1840 4/14/2012 4:01:34 PM An error was encountered while processing an
                          operation.
                          Error Code: 2150859046
                          Error String:<f:WSManFault xmlns:f="http://schemas.mi
                          crosoft.com/wbem/wsman/1/wsmanfault"
                          Code="2150859046"
                          Machine="C3096161287.AD2008R2.loc"><f:Message>WinRM
                          cannot complete the operation. Verify that the
                          specified computer name is valid, that the computer
                          is accessible over the network, and that a firewall
                          exception for the WinRM service is enabled and
                          allows access from this computer. By default, the
                          WinRM firewall exception for public profiles limits
                          access to remote computers within the same local
                          subnet. </f:Message></f:WSManFault>
1840 4/14/2012 4:01:34 PM An error was encountered while processing an
                          operation.
                          Error Code: 2150859046
                          Error String:<f:WSManFault xmlns:f="http://schemas.mi
                          crosoft.com/wbem/wsman/1/wsmanfault"
                          Code="2150859046"
                          Machine="C3096161287.AD2008R2.loc"><f:Message>WinRM
                          cannot complete the operation. Verify that the
                          specified computer name is valid, that the computer
                          is accessible over the network, and that a firewall
                          exception for the WinRM service is enabled and
                          allows access from this computer. By default, the
                          WinRM firewall exception for public profiles limits
                          access to remote computers within the same local
                          subnet. </f:Message></f:WSManFault>
```

image063.png

Figura 4.13: El error de tiempo de espera en el registro de diagnósticos

Éste es realmente una de las cosas más difíciles de Remoting. No le puede decir porqué el servidor no respondió. No se da cuenta que el puerto no está abierto. Podríamos haber especificado un nombre de computadora que no existe. Todo lo que sabe WinRM es que se envió un mensaje a la red y nadie respondió. Al final, casi todos los posibles problemas de "nivel bajo" – son una dirección IP errada, un nombre incorrecto de la computadora, un puerto bloqueado, etc. Son iguales desde el punto de vista de WinRM. Usted tendrá que solucionar estos problemas.

Hemos encontrado que una técnica útil puede ser usar el antiguo cliente de Telnet de línea de comandos. Tenga en cuenta que WS-MAN es sólo HTTP, y HTTP, como muchos protocolos de Internet, simplemente envía texto de un lado a otro, más o menos lo mismo que hace Telnet. HTTP tiene un texto específico para enviar y

recibir, pero la transmisión real es Telnet de la vieja escuela. Así que vamos a ejecutar algo como telnet dc01 5985 sólo para ver si podemos conectar. Una pantalla en blanco es normal: pulsa Ctrl + C para salir, y verá un error HTTP "Solicitud incorrecta". Eso está bien. Al menos confirma que el nombre del equipo, la dirección IP, el puerto y todo lo demás "de bajo nivel" está bien.

Problema de conexión: Sin Permisos

Esto puede ser un problema complicado, ya que necesita ser un administrador para habilitar una traza de diagnóstico. Por otra parte, WinRM suele ser bastante claro cuando no se puede conectar porque su cuenta no tiene permiso para el punto final: "Acceso denegado" es el mensaje de error, y eso es bastante sencillo.

Pero también puede iniciar sesión como administrador (o abrir un shell bajo Credenciales de administrador), habilitar una traza y, a continuación, hacer que el otro usuario (o la otra cuenta de usuario) lo intente. Volver atrás como administrador y deshabilitar la traza y a continuación examinar el registro. La Figura 4.14 muestra lo que está buscando.

```
 1840 4/14/2012 4:18:53 PM An error was encountered while processing an
                           operation.
                           Error Code: 5
                           Error String:<f:WSManFault xmlns:f="http://schemas.mi
                           crosoft.com/wbem/wsman/1/wsmanfault" Code="5"
                           Machine="dc01"><f:Message>Access is denied.
                           </f:Message></f:WSManFault>
  254 4/14/2012 4:18:53 PM Activity Transfer
  142 4/14/2012 4:18:53 PM WSMan operation CreateShell failed, error code 5
32786 4/14/2012 4:18:53 PM Runspace Id 0d91c610-3c82-4b15-8858-76d833a013a3.
                           Callback received for WSMan Create Shell
 1840 4/14/2012 4:18:53 PM An error was encountered while processing an
                           operation.
                           Error Code: 122
                           Error String:<f:WSManFault xmlns:f="http://schemas.mi
                           crosoft.com/wbem/wsman/1/wsmanfault" Code="122"
                           Machine="C3096161287.AD2008R2.loc"><f:Message>The
                           data area passed to a system call is too small.
                           </f:Message></f:WSManFault>
  319 4/14/2012 4:18:53 PM Getting message for error code 5 completed
                           successfully. The languageCode parameter was: en-US
 8196 4/14/2012 4:18:53 PM Modifying activity Id and correlating
12039 4/14/2012 4:18:53 PM Modifying activity Id and correlating
32784 4/14/2012 4:18:53 PM Runspace Id: 0d91c610-3c82-4b15-8858-76d833a013a3
                           Pipeline Id: 00000000-0000-0000-0000-000000000000.
                           WSMan reported an error with error code: 5.
                            Error message: Connecting to remote server dc01
                           failed with the following error message : Access is
                           denied. For more information, see the
                           about_Remote_Troubleshooting Help topic.
                            StackTrace:
32776 4/14/2012 4:18:53 PM Runspace Id: 0d91c610-3c82-4b15-8858-76d833a013a3
                           Pipeline Id: 00000000-0000-0000-0000-000000000000.
                           WSMan reported an error with error code: 5.
                            Error message: Connecting to remote server dc01
                           failed with the following error message : Access is
                           denied. For more information, see the
                           about_Remote_Troubleshooting Help topic.
```

image064.png

Figura 4.14: “Acceso denegado” en el registro de diagnósticos

Los datos de registro le mostrarán la cuenta de usuario que se utilizó para intentar crear la conexión (AD2008R2SallyS, en nuestro ejemplo, por lo que el comando falló - ella no es un administrador). Una comprobación rápida con Get-PSSessionConfiguration en el equipo remoto confirmará los permisos en cualquier punto final de Remoting al que intente conectarse. Además, como se muestra en la figura 4.15, hemos descubierto que ejecutar Set-PSSessionConfiguration puede ser útil. Proporcione el -nombre del punto final que está comprobando y agregue -ShowSecurityDescriptorUI. Eso le permitirá confirmar los permisos del punto final en un formulario GUI más amigable, y puede modificarlo allí mismo si

es necesario.

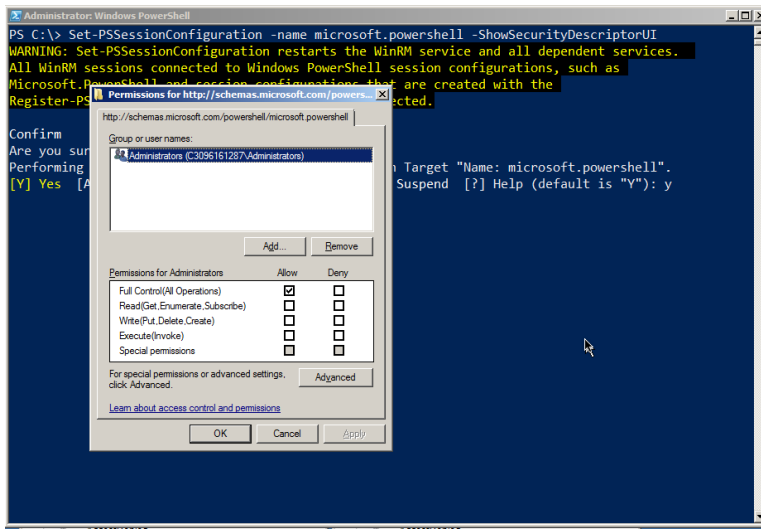

image065.png

Figura 4.15: Comprobación de los permisos de un punto final mediante Set-PSSessionConfiguration

Problema de conexión: Host no confiable

La Figura 4-16 muestra la conexión que estamos intentando realizar: Desde el cliente en el dominio AD2008R2 a un equipo independiente que no forma parte de un dominio.

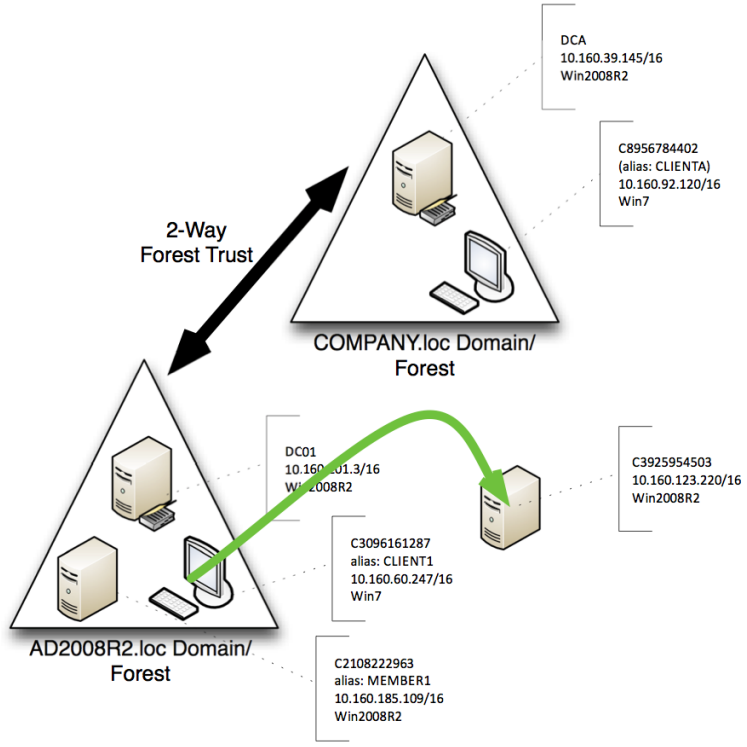

image066.png

Figura 4.16: Tentativa de conexión para este escenario

Como se muestra en la figura 4.17, el error se produce rápidamente, aunque hemos proporcionado una credencial válida. El problema es que estamos en una situación en la que WinRM no puede obtener la autenticación mutua que requiere. La parte 2 de esta guía cubre soluciones para este problema. Pero, ¿cómo se ve el problema en el registro de diagnósticos?

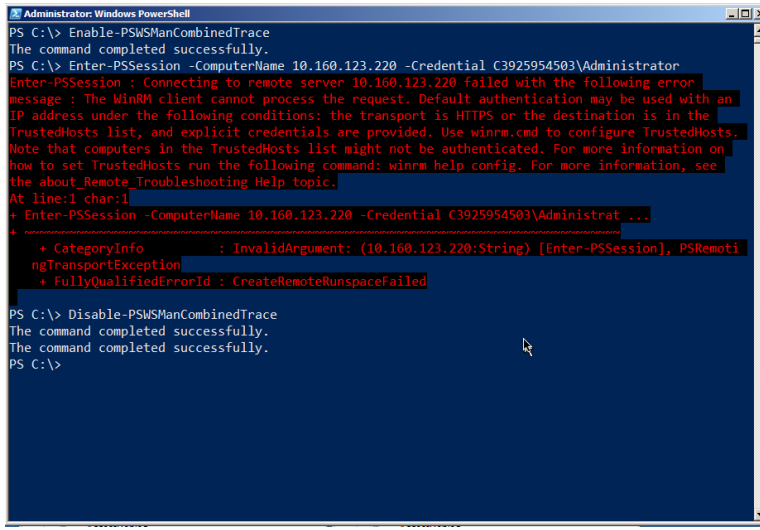

image067.png

Figura 4.17: El mensaje de error da buenas pistas sobre cómo resolver este problema

La Figura 4.18 muestra que WinRM todavía envía su salva inicial de tráfico al servidor. Es cuando la respuesta vuelve que el cliente se da cuenta que no puede autenticar este servidor, y se genera el error. Lo que ve en el registro es más o menos lo que aparece en el shell, literalmente.

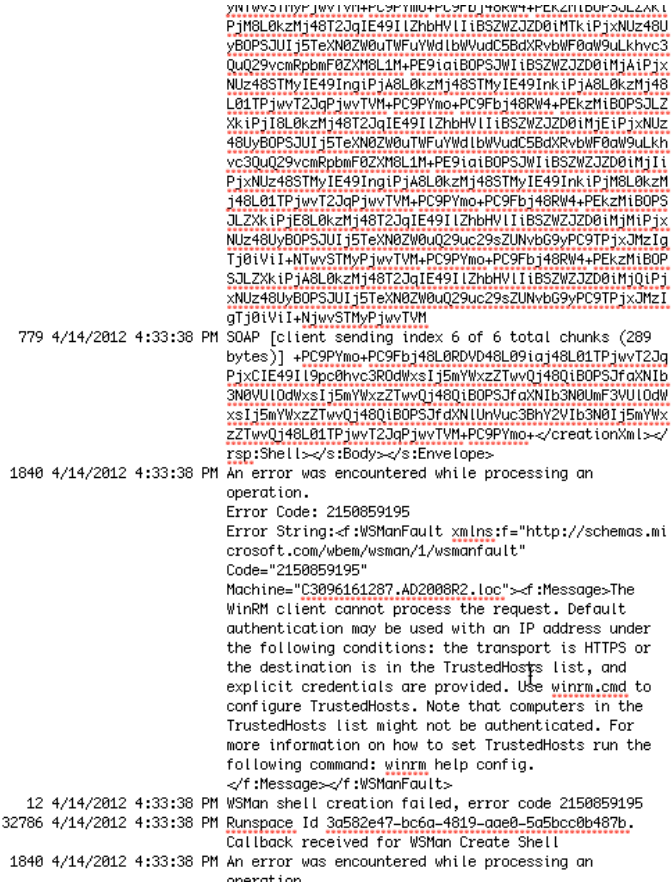

image068.png

Figura 4.18: El contenido del registro de diagnóstico al intentar conectarse a un host no confiable

La Figura 4.19 muestra un buen segundo paso: Ejecutar Test-WSMan. Proporcione el mismo nombre de equipo o dirección IP, pero deje fuera el parámetro -Credential. El Cmdlet puede al menos indicarle que WS-MAN y WinRM están funcionando en el equipo remoto y la versión que están ejecutando. Eso, por lo menos, reduce el problema a uno de autenticación: o bien sus permisos (que

habrían resultado en un "Acceso denegado") o el componente de autenticación mutua de Remoting.

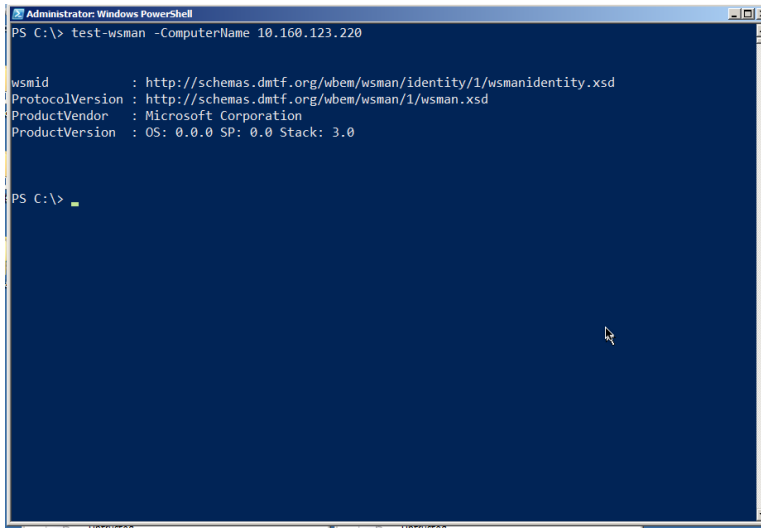

image069.png

Figura 4.19: Test-WSMan es como un "ping" para Remoting

Nota: Verá prácticamente el mismo comportamiento cuando intenta conectarse mediante HTTPS (el conmutador -UseSSL en los distintos comandos de Remoting) y el nombre del certificado SSL de la máquina remota no coincide con el nombre que utilizó en su comando. El mensaje de error es inequívoco tanto en pantalla como en el registro, y discutiremos las soluciones en la parte 2 de la guía.

Metodología Estándar de Solución de Problemas

Solucionar problemas puede ser difícil, especialmente con Remoting ya que hay muchas capas en las que algo puede salir mal. Seguir

un enfoque sencillo y estandarizado puede ayudarle a identificar problemas.

1. Probar Remoting con su configuración predeterminada. Si ha cambiado algo, deshacer los cambios y empezar de cero. Comience por intentar conectarse desde la máquina iniciadora a la máquina de destino utilizando algo distinto de Remoting, pero que siga siendo sensible a la seguridad. Por ejemplo, utilice el Explorador de Windows para abrir la carpeta compartida C$ de la máquina remota.
2. Si eso no funciona, tiene problemas de seguridad más generales. Anote si necesita o no proporcionar credenciales alternativas - si lo hace, Remoting las necesitará también.
3. Instalar un cliente Telnet en la máquina iniciadora (un simple cliente de línea de comandos, como el que viene con Windows). Intente conectarse al oyente HTTP WinRM ejecutando telnet nombre_máquina: 5985. Debería obtener una pantalla en blanco y Ctrl + C finalizará la sesión. Si esto no funciona, hay un problema básico de conectividad (como un puerto bloqueado) que necesita resolver.
4. Utilice Test-WSMan como se describió anteriormente, utilizando una credencial alternativa si es necesario. Asegúrese de que utiliza el nombre real de la máquina tal como aparece en Active Directory o que ha tomado uno de los otros enfoques (TrustedHosts más una credencial o SSL más una credencial) que describimos en la Sección 2 de esta guía. Si eso no funciona, tiene un problema en la configuración de WS-MAN

Simplemente avanzar por estos cuatro pasos, en este orden, puede ayudarle a identificar al menos la causa general de la mayoría de los problemas.

Resumen

Entonces, ¿por qué nos molestábamos en pasar por los registros cuando, en la mayoría de nuestros ejemplos, los registros simplemente hacían eco de lo que estaba en la pantalla? Simple: A medida que PowerShell se inserta en más y más aplicaciones GUI, es posible que no siempre tenga una consola, con sus mensajes de errores agradables, en la que confiar. Lo que puede hacer, sin embargo, es usar la consola para iniciar un seguimiento, ejecutar cualquier aplicación GUI que este fallando y luego buscar en el registro para ver si encuentra algunos de los signos que le hemos mostrado aquí.

Gestión de sesiones

Cuando crea una conexión Remoting entre dos máquinas, está creando una sesión en la terminología de PowerShell. Hay un número increíble de opciones que se pueden aplicar a estas sesiones, y en esta parte de la guía los guiaremos a través de ellas.

Sesiones Ad-Hoc vs. Persistentes

Cuando utiliza un comando Remoting, principalmente Invoke-Command o Enter-PSSession, y especifica un nombre de equipo utilizando su parámetro -ComputerName, está creando una sesión Ad Hoc. Básicamente, PowerShell crea una sesión, la utiliza, y luego ejecuta sus comandos, todo de forma automática.

De manera alternativa, puede utilizar New-PSSession para crear explícitamente una nueva sesión, que luego puede utilizarse pasándola como el parámetro -Session de Invoke-Command, Enter-PSSession y muchos otros comandos compatibles con Remoting. Cuando crea manualmente una sesión, es su responsabilidad deshacerse de ella cuando haya terminado de utilizarla. Sin embargo, si tiene una sesión abierta y cierra su instancia de PowerShell, esa sesión se eliminará automáticamente por usted, por lo que no estaría dejando nada pendiente que necesita ser limpiado.

Desconexión y Reconexión de Sesiones

En PowerShell v3, puede desconectar y volver a conectar sesiones utilizando Disconnect-PSSession y Connect-PSSession. Estos co-

mandos aceptan cada uno un objeto de sesión, que normalmente crearía con New-PSSession.

Una sesión desconectada deja una copia de PowerShell en funcionamiento en el equipo remoto. Esta es una buena manera de ejecutar una tarea de larga duración, desconectarse y luego volver a conectarse más tarde para comprobar su estado. Incluso puede desconectar una sesión en una computadora, moverse a otra computadora y volver a conectarse a esa sesión (aunque no puede conectarse a la sesión desconectada de otro usuario porque está limitado a volver a conectarse a la suya).

Por ejemplo, la figura 5.1 muestra una sesión que se está creando desde un cliente a un servidor. A la sesión se le asigna una tarea para realizarla como un trabajo de fondo y, a continuación, se desconecta la sesión. Es importante tener en cuenta que el comando y el trabajo de fondo están en el servidor (DC01), no en el cliente.

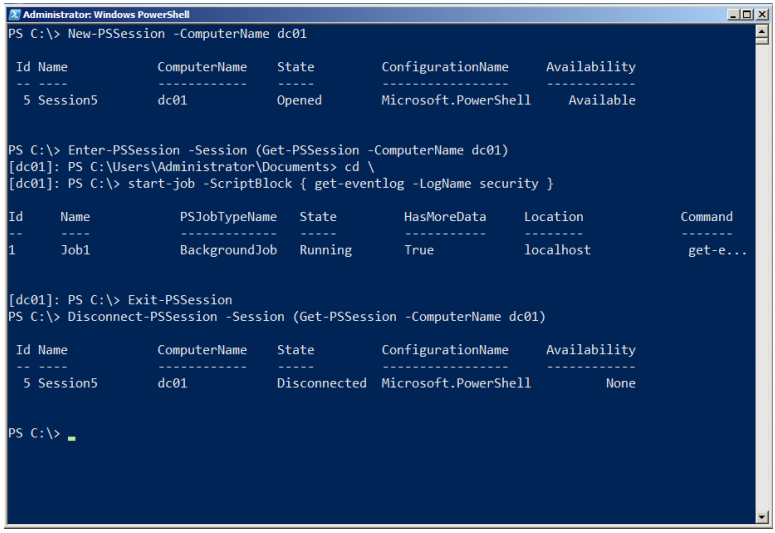

image070.png

Figura 5.1: Creación, uso y desconexión de una sesión

En la figura 5.2, nos hemos trasladado a una máquina diferente.

Hemos iniciado sesión y ejecutado PowerShell, como el mismo usuario que estábamos en el equipo cliente anterior. Recuperamos la sesión desde el equipo remoto y la reconectamos. Luego entramos en la sesión conectada nuevamente, mostramos ese trabajo en segundo plano y recibimos algunos resultados del mismo. Finalmente, salimos de la sesión remota y "apagamos la sesión" mediante Remove-PSSession.

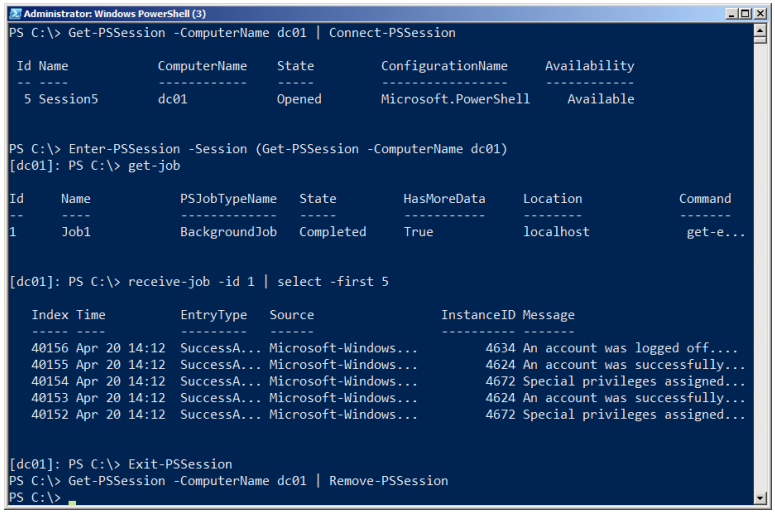

image071.png

Figura 5.2: Reconectar, utilizar y eliminar una sesión

Obviamente, las sesiones desconectadas pueden ser un reto para los procesos de administración, porque está dejando una copia de PowerShell en funcionamiento en una máquina remota y lo está haciendo de una manera que se hace difícil para alguien "verle". Ahí es donde entran en juego las opciones de sesión.

Opciones de Sesión

Cada vez que ejecuta un comando Remoting que crea una sesión, ya sea persistente o Ad Hoc, tiene la opción de especificar un parámetro -SessionOption que acepte un objeto PSSessionOption. El objeto de opción predeterminado se utiliza si no especifica uno, y ese objeto se puede encontrar en la variable global $PSSessionOption. Se muestra en la figura 5.3.

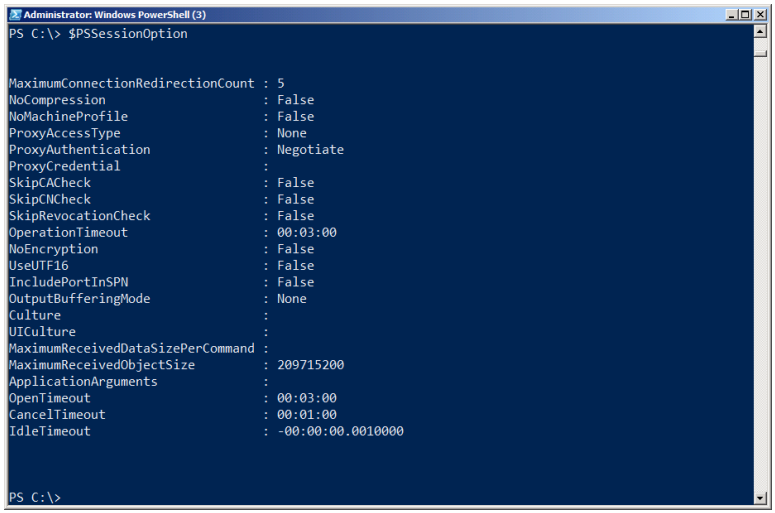

image072.png

Figura 5.3: El objeto PSSessionOption predeterminado almacenado en $PSSessionOption

Como se puede ver, especifica un número de valores predeterminados, incluyendo el tiempo de espera de la operación, el tiempo de espera inactivo y otras opciones. Puede cambiar estos valores simplemente creando un nuevo objeto de opción de sesión y asignándolo a $PSSessionOption. Tenga en cuenta que debe realizar esto en una secuencia de comandos de perfil si desea que los cambios se conviertan en el nuevo valor predeterminado cada vez

que abra una nueva copia de PowerShell. La figura 5.4 muestra un ejemplo..

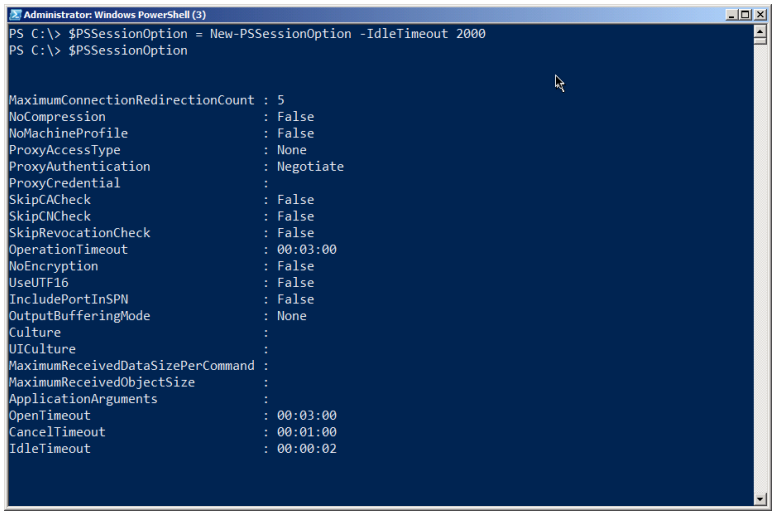

image073.png

Figura 5.4: Creación de un nuevo objeto PSSessionOption predeterminado

Por supuesto, un tiempo de inactividad de 2 segundos probablemente no es muy práctico (y de hecho no funcionará) por lo que debería especificar al menos un tiempo de espera de 60 segundos para lograr utilizar el objeto de sesión). Sin embargo, notará que sólo es necesario especificar los parámetros de opción que desea cambiar. Todo lo demás se establecerá a sus valores predeterminados. También puede especificar una opción de sesión única para cada sesión que cree. La figura 5.5 muestra una forma de hacerlo.

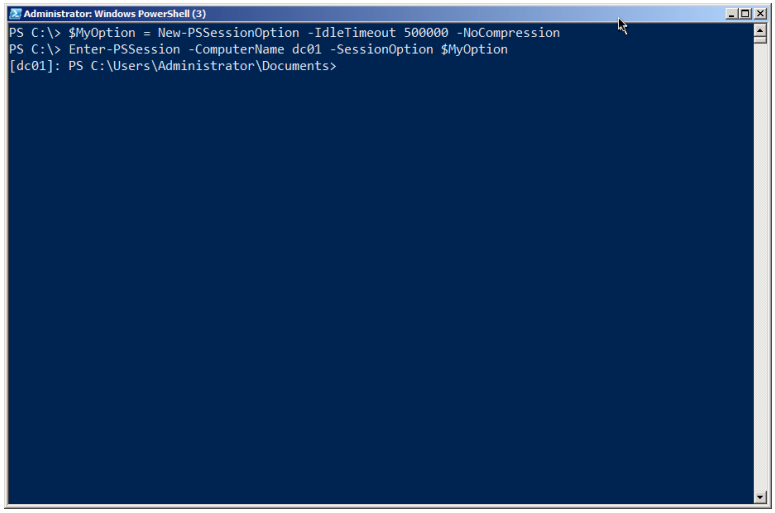

image074.png

Figura 5.5: Creación de un nuevo objeto PSSessionOption para usar con una conexión 1-a-1

Mediante la especificación de valores convenientes para estas opciones, puede ayudar a garantizar que las sesiones desconectadas no se cierren y funcionen de manera adecuada. Un tiempo de espera de inactividad razonable, por ejemplo, asegura que la sesión acabará cerrándose, incluso si un administrador se desconecta y posteriormente se olvida de ella. Tenga en cuenta que cuando se cierra una sesión, se perderán todos los datos de esa sesión, incluidos los resultados de los trabajos en segundo plano. Probablemente sea una buena idea adoptar alguna practica para guardar datos en un archivo (por ejemplo, utilizando Export-CliXML, por ejemplo), para que una sesión inactiva no se cierre y se pierda todo su trabajo.

PowerShell, Remoting y la Seguridad

Aunque PowerShell Remoting ha existido desde aproximadamente 2010, muchos administradores y organizaciones no pueden aprovecharse de ello, debido en gran parte a las políticas anticuadas o desinformadas de seguridad y prevención de riesgos. Este capítulo está diseñado para ayudar a abordar algunos de ellos, al proporcionar detalles técnicos honestos sobre cómo funcionan estas tecnologías. De hecho, presentan un riesgo significativamente menor que muchos de los protocolos de gestión y comunicaciones que ya están en uso generalizado; los protocolos más antiguos se benefician principalmente de estar "anclados" en políticas, pero nunca examinados de cerca.

Ni PowerShell ni Remoting son una "puerta trasera" para el Malware

Este es un gran error. Tenga en cuenta que de forma predeterminada, PowerShell no ejecuta secuencias de comandos. Cuando lo hace, sólo puede ejecutar comandos que el usuario ejecutor tiene permiso para ejecutar - no ejecuta nada bajo una cuenta super-privilegiada, y tampoco omite ni los permisos existentes ni la seguridad. De hecho, como PowerShell está basado en .NET, es improbable que algún autor de malware se moleste en utilizar PowerShell. Tal atacante podría simplemente llamar a la funcionalidad de .NET Framework directamente mucho más fácilmente.

De forma predeterminada, PowerShell Remoting sólo permite que los administradores se conecten y, una vez conectados, sólo pueden

ejecutar comandos con permisos para ejecutarlos, sin posibilidad de omitir permisos o seguridad subyacente. A diferencia de las herramientas anteriores que funcionaban bajo una cuenta altamente privilegiada (como LocalSystem), PowerShell Remoting ejecuta los comandos impersonando al usuario que envió los comandos.

Conclusión: Debido a la forma en que funciona, PowerShell Remoting no permite que ningún usuario, autorizado o no, haga algo que no pueda hacer a través de una docena de otros medios, incluido el inicio de sesión en la consola. Cualquier protección que usted tenga en su lugar para prevenir ese tipo de ataques (como mecanismos apropiados de autorización y autenticación) también protegerá a PowerShell y a Remoting. Si permite a los administradores iniciar sesión en las consolas de servidor, ya sea físicamente o mediante el Escritorio remoto, tiene una exposición de seguridad mucho mayor que la que realiza a través de PowerShell Remoting.

Además, PowerShell ofrece una mejor oportunidad para limitar incluso a los administradores. Un EndPoint Remoting (o la configuración de la sesión) se puede modificar para permitir que sólo los usuarios especificados se conecten a él. Una vez conectado, el EndPoint puede restringir más los comandos que esos usuarios pueden ejecutar. Esto proporciona una oportunidad mucho mejor para la administración delegada. En lugar de hacer que los administradores inicien sesión en las consolas y hagan lo que les plazca, puede hacer que se conecten a EndPoints restringidos y seguros y que sólo completen las tareas específicas que el EndPoint permite

PowerShell Remoting no es opcional

A partir de Windows Server 2012, PowerShell Remoting está habilitado de forma predeterminada y es obligatorio para la administración del servidor. Incluso cuando se ejecuta una consola de administración gráfica localmente en un servidor, la consola todavía "envía" y "responde" a través de Remoting para realizar sus tareas. Sin Remoting, la administración del servidor es imposible. Por lo tanto, las organizaciones están bien informadas para comenzar

inmediatamente a encontrar una forma de incluir Remoting en sus protocolos permitidos. De lo contrario, los servicios críticos no podrán ser administrados, ni siquiera a través de Escritorio remoto o directamente en la consola del servidor.

Este enfoque realmente ayuda a proteger mejor los centros de datos. Debido a que la administración local es exactamente la misma que la administración remota (a través de Remoting), ya no hay ninguna razón para acceder físicamente o de forma remota a las consolas de servidor. Las consolas pueden así permanecer más bloqueadas y protegidas, y los administradores pueden permanecer fuera del centro de datos por completo.

Remoting no transmite ni almacena credenciales

De forma predeterminada, Remoting utiliza Kerberos, un protocolo de autenticación que no transmite contraseñas a través de la red. En su lugar, Kerberos se basa en contraseñas con una clave de cifrado, asegurando que las contraseñas permanezcan seguras. Remoting puede configurarse para usar protocolos de autenticación menos seguros (como Basic), pero también puede configurarse para requerir el cifrado basado en certificados para la conexión.

Además, Remoting nunca almacena credenciales en ningún almacenamiento persistente de forma predeterminada. Una máquina remota nunca tiene acceso a las credenciales de un usuario. Sólo tiene acceso a un token de seguridad delegado (un "ticket" de Kerberos), que se almacena en la memoria volátil que no puede, por diseño del Sistema Operativo, ser escrito en el disco - incluso en el archivo de página (page file) del Sistema Operativo. El servidor presenta ese token al Sistema Operativo al ejecutar comandos, haciendo que el comando sea ejecutado con la autoridad del usuario original que invoca- y nada más

Remoting utiliza el cifrado

La mayoría de las aplicaciones habilitadas para Remoting aplican su propia encriptación a su tráfico a nivel de aplicación enviado a través de Remoting. Sin embargo, Remoting también puede configurarse para utilizar HTTPS (conexiones con cifrado de certificado) y puede configurarse para que HTTPS sea obligatorio. Esto cifra todo el canal utilizando cifrado de alto nivel, al tiempo que garantiza la autenticación mutua tanto del cliente como del servidor.

Remoting es transparente para la seguridad

Como se ha indicado, Remoting ni añade nada ni quita nada a su configuración de seguridad existente. Los comandos remotos se ejecutan utilizando las credenciales delegadas de cualquier usuario que invoque los comandos, lo que significa que sólo pueden hacer lo que tienen permiso para hacer, y lo que podrían presumiblemente hacer con media docena de otras herramientas de todos modos. Cualquiera que sea la auditoría que tenga en su entorno no puede ser ignorada por Remoting. A diferencia de muchas soluciones anteriores de "ejecución remota", Remoting no funciona bajo una cuenta "super-privilegiada" a menos que la configure de esa manera (lo que requiere varios pasos y no puede lograrse accidentalmente, ya que requiere la creación de EndPoints personalizados).

Recuerde: cualquier cosa que alguien puede hacer a través de Remoting, ya la puede hacer a través de media docena de formas diferentes. Remoting simplemente proporciona como un medio más consistente, controlable y escalable de hacerlo

Remoting es una sobrecarga menor

A diferencia de Remote Desktop Connection (RDC, que muchos Administradores utilizan actualmente para administrar servidores remotos), Remoting es una carga menor. No requiere que el servidor genere un entorno operativo gráfico entero, lo que afecta el rendimiento del servidor y la gestión de la memoria. El control remoto también es más escalable, permitiendo a los usuarios autorizados (principalmente administradores en la mayoría de los casos) ejecutar comandos contra varios servidores a la vez, lo que mejora la coherencia y reduce el error, al tiempo que acelera los tiempos de respuesta y reduce los gastos administrativos.

Remoting es el camino a seguir de Microsoft. No utilizar Remoting es intentar usar deliberadamente Windows de una manera para la que no que fue diseñado. Reducirá, no mejorará su seguridad, al mismo tiempo que aumentará la sobrecarga operacional, lo que permitirá mayores errores humanos y reducirá el rendimiento del servidor. Los Administradores de Microsoft han trabajado durante décadas bajo un paradigma operacional que estaba mal dirigido y era miope. Remoting está finalmente entregando a Windows el modelo administrativo que todos los sistemas operativos de red han utilizado durante años, si no décadas.

Remoting utiliza autenticación mutua

A diferencia de casi todas las demás técnicas de gestión remota - incluyendo herramientas como PSExec e incluso, en algunas circunstancias, Remote Desktop, PowerShell Remoting por defecto requiere autenticación mutua. El usuario que intenta conectarse a un servidor es autenticado y conocido. El sistema también asegura que el servidor conectado sea el servidor deseado y no un impostor.

Esto proporciona una seguridad mucho mejor que las técnicas anteriores, al mismo tiempo que ayuda a reducir errores ya que no se puede "iniciar sesión accidentalmente en la consola incorrecta", como podría hacerlo si tuviera que ingresar en el centro de datos.

Resumen

En este punto, negar PowerShell Remoting es como negar Ethernet. Es ridículo pensar que operará exitosamente su entorno sin él. Por primera vez, Microsoft ha proporcionado una tecnología oficial, soportada, para la administración de servidores remotos que no utiliza credenciales elevadas, no almacena credenciales de ninguna manera, que admite autenticación mutua y que es transparente en cuanto a seguridad. Esta es la tecnología de administración que deberíamos haber tenido todo el tiempo; Moviéndose a Remoting solamente hará su ambiente más manejable y más seguro, no menos.

Configuración de Remoting mediante GPO

La documentación de *About_remote_troubleshooting* de PowerShell proporciona un conjunto de pasos para configurar la funcionalidad de Remoting básica a través de objetos de directiva de grupo (GPO). Ejecutando Enable-PSRemoting también revela algunos detalles útiles, como las cuatro principales configuraciones necesarias. En esta sección, cubriremos estos pasos de configuración principales.

Nota: Nada de esto es necesario en Windows Server 2012 y versiones posteriores del sistema operativo de servidor. Remoting está habilitado de forma predeterminada y no debería encontrar problemas, ya que muchas de las herramientas de administración nativas (incluidas las consolas GUI, como el Administrador de servidores) dependen de Remoting.

Advertencias de GPO

Una cosa a tener en cuenta es que a través de una GPO sólo se pueden establecer cambios de configuración. No se puede cambiar el estado del ordenador. En otras palabras, mientras una GPO puede configurar el modo de inicio de un servicio como "Automático", no puede iniciar el servicio. Por lo tanto, en muchos casos, los cambios que realice a través de GPO (con respecto a Remoting) no surtirán efecto hasta la próxima vez que se reinicien los equipos afectados, ya que en la mayoría de los casos la computadora sólo mira la configuración durante el arranque. No pierda esto de vista.

Además, todo en esta sección supone que PowerShell ya está

instalado en los equipos de destino, algo que también se puede lograr con una GPO u otro mecanismo de implementación de software, por lo tanto no vamos a cubrir eso aquí. Tenga en cuenta que la mayor parte de esta sección debería aplicarse a PowerShell v2 o v3. En esto ejemplos, vamos a utilizar v2 en un equipo cliente con Windows 7 perteneciente a un dominio de Windows Server 2008 R2.

Nota: Algunas de las configuraciones de GPO que estaremos revisando estarán disponibles en Windows 2008 y Windows 2008 R2, por lo que debería ser capaz de instalar las plantillas administrativas necesarias en cualquier controlador de dominio. El Kit de herramientas de administración remota (RSAT) de Windows 7 (y versiones posteriores) contiene las plantillas necesarias.

No sabemos con certeza si los pasos de configuración de GPO deben realizarse en el orden en que los presentamos. En la mayoría de los casos, esperamos que los haga todos a la vez en un solo GPO, por lo que no debería importar. Lo llevaremos paso a paso en este orden para que podamos comprobar los resultados individuales a lo largo del camino.

Permitir la configuración automática de los escuchas (Listeners) de WinRM

Como se explicó anteriormente en esta guía, el servicio WinRM configura uno o más oyentes (listeners) para aceptar el tráfico entrante. Ejecutar Enable-PSRemoting, por ejemplo, configura un detector de HTTP y ya hemos cubierto cómo configurar un detector de HTTPS además de, o en lugar de, uno predeterminado.

Encontrará esta configuración en: Computer Configuration\Administrative Templates\Windows Components\Windows Remote Management

(WinRM)\WinRM Service. Habilite la directiva y especifique los filtros IPv4 e IPv6, que determinan en qué rangos de direcciones IP se configurará. Puede utilizar el comodín * para designar todas las direcciones IP, que es lo que hemos hecho en la Figura 7.1.

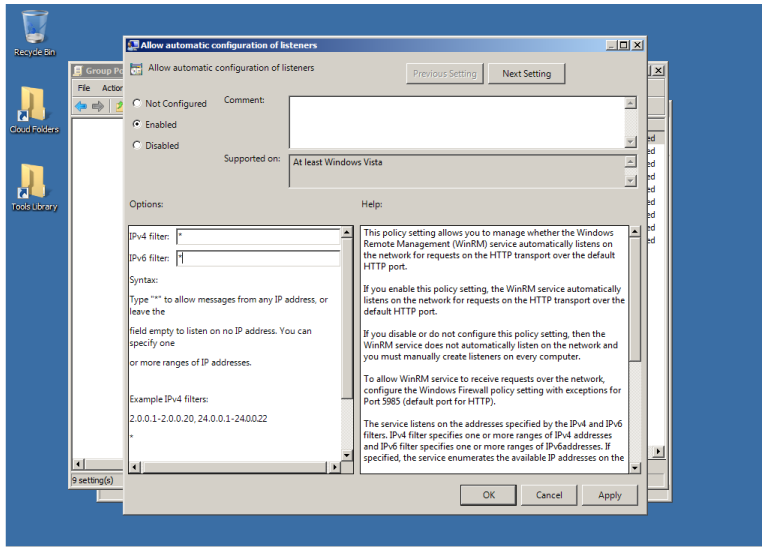

image075.png

Figura 7.1: Habilitación de la configuración automática de los oyentes de WinRM

Configuración del servicio WinRM para que se inicie automáticamente

Este servicio está configurado para iniciarse automáticamente en los sistemas operativos de servidor más recientes (Windows Server 2003 y posteriores), pero no en los clientes, así que este paso sólo será necesario para los equipos cliente. Una vez más, esto no iniciará el servicio, pero la próxima vez que se reinicie el equipo, el servicio

se iniciará automáticamente.

Microsoft sugiere realizar esta tarea ejecutando un comando de PowerShell, que no requiere que se habilite Remoting para funcionar:

```
Set-Service WinRM -computername $servers -startup Automatic
```

Puede llenar $servers de la forma que desee, siempre que contenga cadenas que sean nombres de equipos y siempre y cuando tenga credenciales de administrador en esos equipos. Por ejemplo, para capturar cada equipo de su dominio, ejecutaría lo siguiente (esto supone PowerShell v2 o v3, en un equipo con Windows 7 con el RSAT instalado):

```
 Import-Module ActiveDirectory $servers = Get-ADComputer
-filter \* | Select -expand name
```

Es probable que desee limitar el número de ordenadores especificando un - Filter distinto de "*" o especificando -SearchBase y limitando la búsqueda a una UO específica. Lea la ayuda de Get-ADComputer para obtener más información sobre esos parámetros.

Tenga en cuenta que Set-Service devolverá un error si no puede conectarse a una computadora o aquellas para las que el cambio no se pudo establecer y luego continuara con la siguiente computadora en la lista.

También puede configurar esto con una GPO. En Computer ConfigurationWindows SettingsSecurity SettingsSystem Services, busque "Windows Remote Management ". Haga clic con el botón derecho y establezca un modo de inicio automático. Eso es lo que hicimos en la figura 7.2.

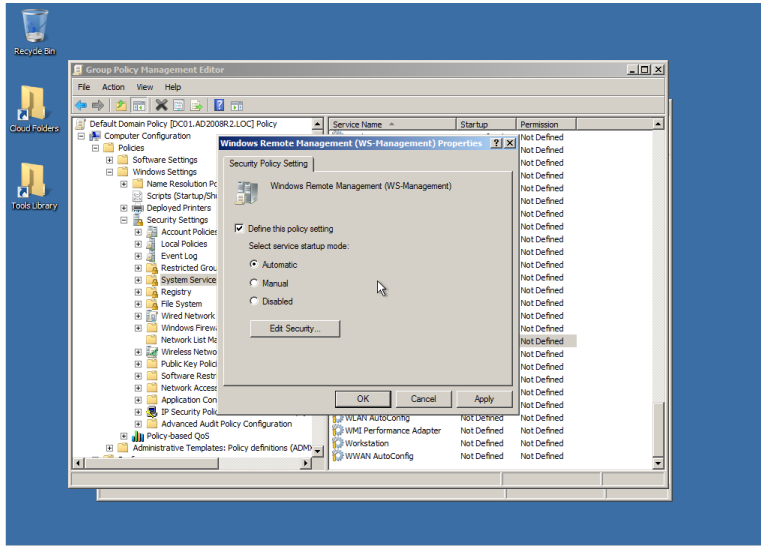

image076.png

Figura 7.2: Configuración del modo de inicio del servicio WinRM

Creación de una excepción de Firewall de Windows

Este paso será necesario en todos los equipos en los que esté habilitado el Firewall de Windows. Estamos asumiendo que sólo desea que Remoting esté habilitado en su perfil de firewall de dominio, de modo que eso es todo lo que haremos en nuestro ejemplo. Por supuesto, usted puede gestionar cualquier otra excepción que desee en los perfiles que sean apropiados para su entorno.

Encontrará una configuración Computer ConfigurationAdministrative TemplatesNetworkNetwork ConnectionsWindows Firewall-Domain Profile. Tenga en cuenta que la directiva “ Windows Firewall: Allow Local Port Exceptions “ simplemente permite a

los administradores locales configurar las excepciones de Firewall mediante el Panel de control. En realidad no crea excepciones.

Entonces, ubicamos la política “Define inbound port exceptions “ y lo habilitamos, como se muestra en la figura 7.3

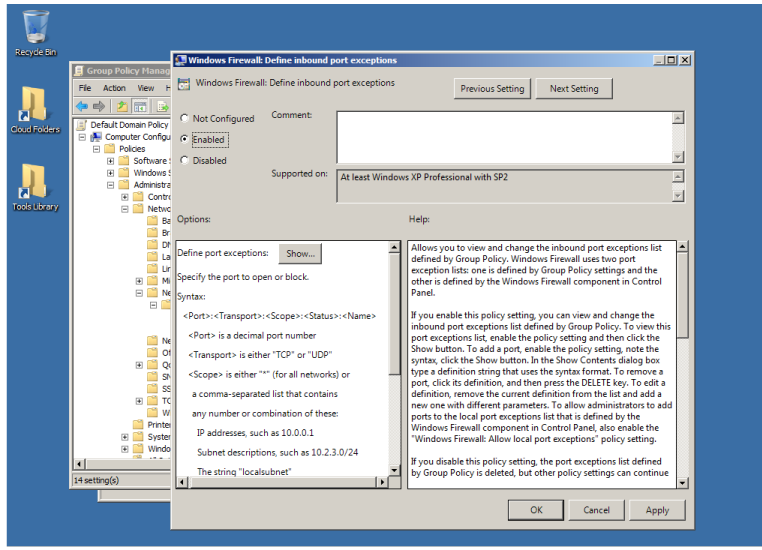

image077.png

Figura 7.3: Habilitación de excepciones de Firewall

A continuación, hicimos clic en “ Show “, y agregamos “ 5985:TCP:*:enabled:WinRM “ como una nueva excepción, como se muestra en la figura 7.4.

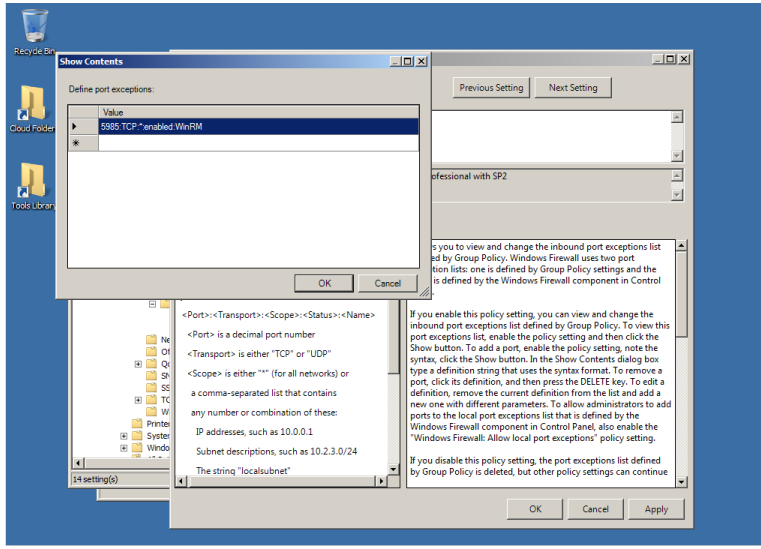

image078.png

Figura 7.4: Creación de la excepción de Firewall

¡Darle una oportunidad!

Después de aplicar los cambios de GPO anteriores, reiniciamos nuestro equipo cliente. Cuando se inicia el servicio WinRM, este comprueba si tiene oyentes configurados. Cuando descubra que no lo hace, debería intentar configurar automáticamente uno, lo que ahora le hemos permitido hacer mediante GPO. La excepción Firewall debe permitir que el tráfico entrante llegue al oyente.

Como se muestra en la figura 7.5, parece que funciona. ¡Hemos encontrado al oyente recién creado!

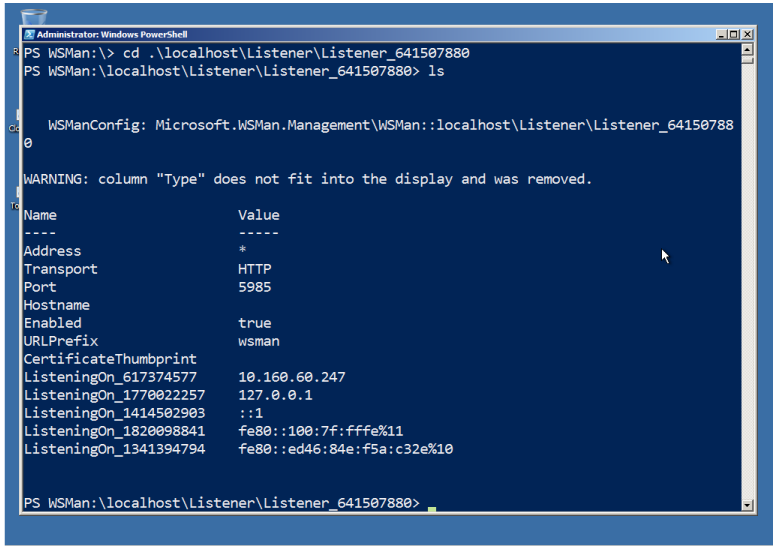

image079.png

Figura 7.5: Comprobación del escuchador WinRM recién creado

Por supuesto, no se puede saber si algo funciona, hasta que no se pone a prueba. Así que intentamos conectar desde otra computadora y, como se muestra en la figura 7.6, pudimos iniciar una sesión de Remoting interactiva en nuestro equipo cliente original. No hemos configurado nada excepto a través de GPO, y todo está funcionando.

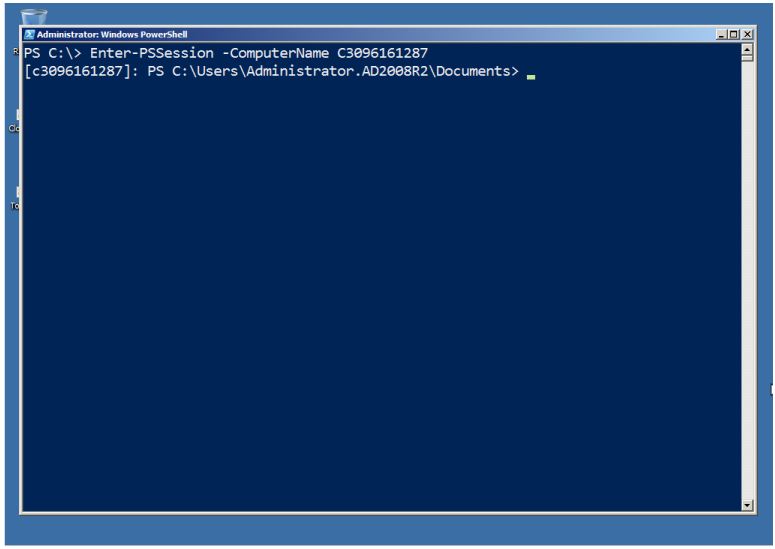

image080.png

Figura 7-6: Iniciando una sesión de Remoting 1-a-1 con el equipo cliente configurado mediante GPO

Lo que no se puede hacer con una GPO

No puede utilizar una GPO para iniciar el servicio WinRM, como ya lo hemos indicado. Tampoco se pueden crear escuchas (listeners) personalizadas a través de GPO, ni puede crear puntos finales de PowerShell personalizados (configuraciones de sesión). Sin embargo, una vez que se habilita el Remoting básico mediante GPO, puede utilizar el Cmdlet Invoke-Command de PowerShell para realizar de forma remota esas tareas. Incluso puede usar Invoke-Command para deshabilitar remotamente el oyente HTTP predeterminado, si así lo desea.

Además, tenga en cuenta que el WSMAN PSProvider de PowerShell

puede asignar la configuración WinRM de los equipos remotos a la unidad WSMAN local. Es por eso que, por defecto, la "carpeta" de nivel superior en esa unidad es "localhost"; De modo que hay un lugar para agregar otros ordenadores, si lo desea. Eso ofrece otra forma de configurar oyentes (listeners) y otros ajustes relacionados con Remoting.

La verdadera clave es utilizar GPO para habilitar Remoting en su forma básica. A partir de ahí, puede utilizar Remoting en sí para modificar, reconfigurar y/o establecer la configuración