Capítulo 12
Seguridad

1. La seguridad: ¿para quién? ¿Por qué?

La aparición de las redes locales y de Internet ha cambiado muchas cosas en la manera de proteger su PC. Ya no basta con encadenar el disco duro a la mesa y cerrar la puerta del despacho por la noche para que no le roben o pirateen sus datos. Ahora, proteger su puesto de trabajo se ha convertido en algo esencial para no tener que remediar intrusiones o malas intenciones.

Pero entonces ¿contra quién prevenir? Pues contra todo lo que se mueve… y también los que no se mueven. En efecto, ya sean programas malintencionados, usuarios con malas intenciones, o hasta usuarios novatos, todos están considerados una amenaza. Por este motivo debe asegurar su sistema estableciendo reglas de seguridad, aplicándolas y asegurándose de que los demás hacen lo mismo.

2. Los riesgos vinculados al scripting

Adivinará rápidamente que lo que hace la fuerza del scripting se convierte en su debilidad. La facilidad con la que puede hacer todo, ya sea haciendo clic en un script o ejecutándolo desde la ventana de comandos, puede meterle en problemas si no presta atención.

Imagine un script de inicio de sesión que en la apertura de la sesión ¡la bloquea enseguida! Claro, es divertido entre amigos, pero en una empresa, dudamos que esto sea bueno. Todavía peor un script de una persona malintencionada o realmente novata en PowerShell (en este caso, le aconsejamos comprarle un ejemplar de este libro…) puede perfectamente bloquear cuentas de usuarios en Active Directory, formatear el disco, reiniciar el sistema en un bucle infinito… Lo ha entendido, un script puede hacer de todo. Aunque a día de hoy se notifica al usuario con alertas para prevenirle de la ejecución de un script, estas no son capaces de determinar si un script es nocivo para el correcto funcionamiento del sistema.

Los riesgos vinculados al scripting se resumen a una historia de compromisos: o bien impide toda ejecución de scripts, es decir asumir el riesgo de «amargarle la vida» teniendo que hacer y rehacer tareas básicas y muy ingratas, o bien elige abrir su sistema a PowerShell, teniendo cuidado de tomar las precauciones que se imponen.

Pero no se deje desanimar ya que aunque la ejecución de scripts le expone a ciertos problemas de seguridad, PowerShell está dotado de ciertos conceptos que hacen de él uno de los lenguajes de script más seguros. No hay que olvidar que en caso de problemas de seguridad, es la imagen de Microsoft en su conjunto la que sufre…

3. Optimizar la seguridad de PowerShell

3.1 La seguridad de PowerShell por defecto

Lo ha entendido, la seguridad es una área muy importante, sobre todo en el dominio del scripting. Por este motivo los creadores de PowerShell han incluido dos reglas de seguridad por defecto.

Los archivos ps1 asociados al bloc de notas

La extensión «**`.ps1`**» de los scripts PowerShell está por defecto asociada al bloc de notas (o Notepad). Este procedimiento permite evitar ejecutar automáticamente scripts potencialmente peligrosos por una mala manipulación. El bloc de notas es, realmente, un editor un poco clásico, pero tiene la doble ventaja de ser inofensivo y de no bloquear la ejecución de un script cuando este está abierto con el editor. Observará sin embargo que la edición (clic con el botón derecho + **Modificar**) de archivos **`.ps1`** está asociada al editor ISE.

Observación

Este tipo de seguridad no existía con los scripts VBS cuya apertura estaba directamente asociada a Windows Script Host.

Una directiva de ejecución restringida

La segunda barrera de seguridad es la aplicación de la directiva de ejecución **Restricted** por defecto para los puestos clientes y **RemoteSigned** por defecto para los servidores (desde Windows Server 2012 R2).

La directiva **Restricted** es la más restrictiva. Es decir que bloquea sistemáticamente la ejecución de todos los scripts. Solo se ejecutan los comandos tecleados en la shell. Para remediar a esta situación, PowerShell requiere que el usuario cambie el modo de ejecución con el comando **Set-ExecutionPolicy <modo de ejecución>**. Pero para ello debe ser administrador del equipo.

Observación

Igual comprende mejor por qué el uso de PowerShell en sus equipos no constituye un aumento del riesgo, en la medida en que se respetan ciertas reglas.

3.2 Las directivas de ejecución

PowerShell integra un concepto de seguridad que llamamos directivas de ejecución (execution policies) para que un script no autorizado no pueda ejecutarse en contra de la voluntad del usuario. Existen siete configuraciones posibles: **Restricted**, **RemoteSigned**, **AllSigned**, **UnRestricted**, **Bypass**, **Default** y **Undefined**. A cada una de ellas le corresponde un nivel de autorización de ejecución de determinados scripts. Podrá llegar a cambiar en función de la directiva que desee aplicar.

3.2.1 Las diferentes directivas de ejecución

Restricted: es la directiva más restrictiva y también la directiva por defecto en los puestos cliente (de Windows 7 a Windows 10). No permite la ejecución de scripts pero autoriza las instrucciones por la línea de comandos tecleadas en la consola (modo interactivo). Esta directiva puede considerarse como la más radical dado que protege frente a la ejecución de archivos **.ps1**.

Con esta directiva, al tratar de ejecutar un script, se muestra un mensaje de este tipo en la consola:

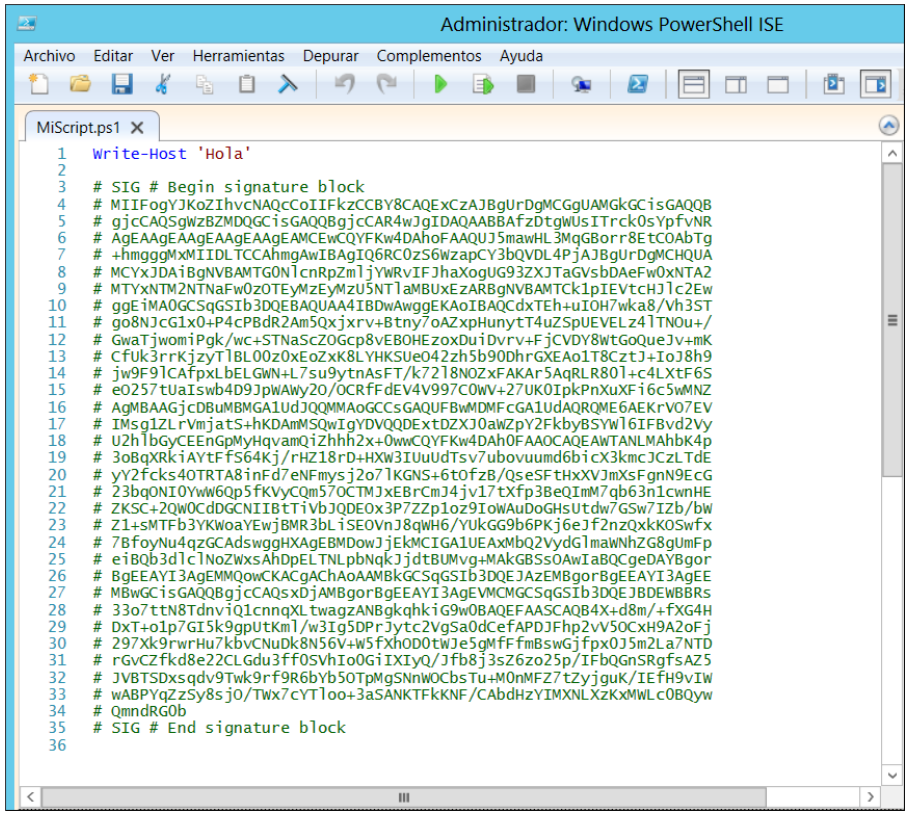
```
.\lanzadera.ps1 : File C:\temp\lanzadera.ps1 cannot be loaded because running
scripts is disabled on this system. For more information, see
about_Execution_Policies at https://go.microsoft.com/fwlink/?LinkID=135170.
At line:1 char:1 *
+ .\lanzadera.ps1
+ ~~~~~~~~~~~~
+ CategoryInfo            : SecurityError: (:) [], PSSecurityException
+ FullyQualifiedErrorId : UnauthorizedAccess
```

Si esta es la directiva definida por defecto en la instalación de PowerShell, tendrá que cambiarla para la ejecución de su primer script.

Observación

Para poder modificar la directiva de ejecución PowerShell, deberá ser administrador local de la máquina.

`AllSigned`: es la directiva más «segura» que permite la ejecución de scripts. Autoriza únicamente la ejecución de scripts firmados. Un script firmado es un script que contiene una firma digital como la que se muestra en la figura de abajo.



Ejemplo de script firmado

Con la directiva **`AllSigned`**, la ejecución de scripts firmados necesita que esté en posesión de los certificados correspondientes (consulte la sección Firma de scripts).

RemoteSigned: esta directiva es similar a **AllSigned** con la diferencia de que solo los scripts que tienen un origen distinto al local necesitan una firma. Por consiguiente, todos sus scripts creados localmente pueden ser ejecutados sin necesidad de firma.

A partir de Windows Server 2012 R2, PowerShell se ejecuta con esta directiva de ejecución por defecto, lo que no era el caso en las versiones anteriores de Windows Server.

Si intenta ejecutar un script descargado de Internet sin estar firmado, obtendrá el siguiente mensaje de error.

```
.\Get-Script.ps1: File C:\Temp\Script.ps1 cannot be loaded.
The file C:\Temp\Script.ps1 is not digitally signed.
```

Observación

Seguramente se pregunte ¿cómo hace PowerShell para saber que nuestro script tiene su origen en Internet? Respuesta: Gracias a los «Alternate Data Streams» que se implementan con la forma de flujos cachés desde las aplicaciones de comunicación tales como Microsoft Outlook, Internet Explorer, Outlook Express y Windows Messenger (consulte la sección dedicada a los Alternate Data Streams). En resumen, cuando descarga un script desde un cliente Microsoft, se le adjunta su origen.

Unrestricted: con esta directiva todos los scripts, sea cual sea su origen, se ejecutan sin solicitar una firma.

Esta directiva muestra aun así un aviso cuando intenta ejecutar algún script descargado de Internet.

```
PS > .\script.ps1

Security warning

Run only scripts that you trust. While scripts from the internet can
be useful, this script can potentially harm your computer. If you trust
this script, use the Unblock-File cmdlet to allow the script to run
without this warning message. Do you want to run C:\Temp\script.ps1?
[D] Do not run  [R] Run once  [S] Suspend  [?] Help (default is "D"):
```

Bypass: es la directiva menos restrictiva, y por lo tanto la menos segura. No se bloquea nada y no se muestra ningún mensaje de aviso. Es por lo tanto la directiva con el riesgo más elevado de ejecutar scripts malintencionados.

Undefined: no se ha definido ninguna directiva para el ámbito actual. Si no se definen directivas de ejecución en ningún ámbito entonces la directiva efectiva será la directiva **Restricted**.

Default: carga la directiva por defecto, a saber **Restricted**.

Observación

Microsoft ha puesto en marcha mecanismos con el fin de intentar limitar los riesgos vinculados a la ejecución de scripts que provengan del exterior de la empresa y por lo tanto potencialmente peligrosos. La configuración por defecto permite alcanzar este objetivo pero no garantiza en ningún caso una seguridad perfecta.

3.2.2 Los ámbitos de las directivas de ejecución

PowerShell permite gestionar el ámbito de las directivas. El orden de aplicación es el siguiente:

– **Ámbito `Process`**: la directiva de ejecución solo afecta a la sesión actual (proceso Windows PowerShell). El valor asignado al ámbito **`Process`** se almacena únicamente en memoria; por lo tanto no se conserva al cerrar la sesión de PowerShell.

– **Ámbito `CurrentUser`**: la directiva de ejecución aplicada al ámbito **`CurrentUser`** solo afecta al usuario actual. Se almacena el tipo de directiva de manera permanente dentro de la clave de registro **`HKEY_CURRENT_USER`**.

– **Ámbito `LocalMachine`**: la directiva de ejecución aplicada al ámbito **`LocalMachine`** afecta a todos los usuarios del equipo. Se almacena el tipo de directiva de manera permanente dentro de la clave de registro **`HKEY_LOCAL_MACHINE`**.

La directiva con prioridad 1 tiene preferencia sobre la que tenga prioridad 3. Por lo tanto si el ámbito **`LocalMachine`** es más restrictivo que el ámbito **`Process`**, la directiva aplicada será aun así la del ámbito **`Undefined`** en cuyo caso PowerShell aplicará la directiva del ámbito **`CurrentUser`** y después intentará aplicar la directiva **`LocalMachine`**.

Recuerde que el ámbito **`LocalMachine`** es el definido por defecto cuando aplicamos una directiva de ejecución sin precisar un ámbito particular.

3.2.3 Identificar la directiva de ejecución actual

La directiva de ejecución actual se obtiene con el comando **`Get-ExecutionPolicy`**.

Ejemplo

```
PS > Get-ExecutionPolicy
Restricted
```

Con este comando, tenemos la opción **`-List`**. Gracias a ella, sabremos qué directivas se aplican en cada ámbito.