Ejecutando PowerShell desde C#: análisis y explicación de un ejemplo práctico

Una de las ventajas de .NET es la posibilidad de lanzar procesos externos desde nuestras aplicaciones. Esto permite ejecutar comandos de PowerShell, scripts administrativos o herramientas del sistema directamente desde código C#.

Veamos un ejemplo corregido y funcional basado en el código original.

Código corregido

```
using System;
using System.Diagnostics;

namespace Hola
{
    class Program
    {
        static void Main()
        {
            ProcessStartInfo psi = new ProcessStartInfo();

            psi.FileName = "powershell.exe";

            psi.Arguments =
                "-NoProfile -Command \"1..10 | ForEach-Object
{ rundll32 zipfldr.dll,RouteTheCall notepad }\"";

            psi.UseShellExecute = false;
            psi.CreateNoWindow = true;

            Process.Start(psi);

            Console.WriteLine("Comando ejecutado.");
```

```
        }
    }
}
```

¿Qué hace este programa?

El código inicia un proceso de PowerShell desde una aplicación C#.

La secuencia es la siguiente:

1. Se crea un objeto ProcessStartInfo.
2. Se especifica que el ejecutable será powershell.exe.
3. Se envían argumentos para ejecutar un comando de PowerShell.
4. Se inicia el proceso mediante Process.Start().

Analizando cada línea

Crear la configuración del proceso

```
ProcessStartInfo psi = new ProcessStartInfo();
```

Este objeto almacena toda la información necesaria para lanzar un proceso externo.

Especificar el ejecutable

```
psi.FileName = "powershell.exe";
```

Indica que se ejecutará PowerShell.

Pasar argumentos

```
psi.Arguments =
     "-NoProfile -Command \"1..10 | ForEach-Object { rundll32 zipfldr.dll,RouteTheCall notepad }\"";
```

Aquí encontramos varios elementos interesantes.

-NoProfile

Evita cargar el perfil del usuario.

Esto hace que la ejecución sea más rápida y predecible.

-Command

Permite ejecutar un comando directamente sin necesidad de crear un archivo .ps1.

1..10

Genera una secuencia de números del 1 al 10.

```
1
2
3
4
5
6
7
8
9
10
```

ForEach-Object

Ejecuta una acción para cada elemento de la secuencia.

```
1..10 | ForEach-Object {
    Write-Host $_
}
```

rundll32

Es una utilidad clásica de Windows que permite ejecutar funciones exportadas por archivos DLL.

```
rundll32.exe archivo.dll,Funcion
```

¿Qué hace realmente este comando?

rundll32 zipfldr.dll,RouteTheCall notepad

Utiliza una función interna de la DLL asociada a las carpetas comprimidas de Windows (zipfldr.dll).

Durante años esta técnica ha aparecido en investigaciones de seguridad, pruebas de laboratorio y demostraciones de ejecución indirecta de procesos.

Al ejecutarse dentro del bucle:

```
1..10 | ForEach-Object {
    rundll32 zipfldr.dll,RouteTheCall notepad
}
```

la llamada se realiza diez veces consecutivas.

Ocultando la ventana

psi.CreateNoWindow = true;

Impide que aparezca una consola visible durante la ejecución.

Deshabilitando el shell

psi.UseShellExecute = false;

Hace que el proceso sea creado directamente por la aplicación en lugar de delegarlo al shell de Windows.

Esta configuración es habitual cuando queremos automatizar tareas.

Capturando la salida de PowerShell

Una mejora interesante consiste en leer la salida generada por el comando.

```
using System;
using System.Diagnostics;

class Program
{
    static void Main()
    {
        ProcessStartInfo psi = new ProcessStartInfo
        {
            FileName = "powershell.exe",
            Arguments = "-NoProfile -Command \"Get-Date\"",
            UseShellExecute = false,
            RedirectStandardOutput = true,
            CreateNoWindow = true
        };

        Process process = Process.Start(psi);

        string salida = process.StandardOutput.ReadToEnd();

        process.WaitForExit();

        Console.WriteLine(salida);
    }
}
```

En este caso el programa ejecuta PowerShell y muestra la fecha actual devuelta por el cmdlet Get-Date.