Verificar la versión de software de una aplicación en remoto

Problemática

Una mañana, su superior viene a verle y le pide hacer un inventario de las diferentes versiones de una aplicación desplegada en un importante parque de equipos. Es en este momento cuando se da cuenta de que no tiene ni un instrumento de informes de aplicativos, ni un inventario del despliegue software actualizado.

Solución

Para hacer frente a esta problemática podemos pensar en la creación de un script que, para cada cuenta de equipo, va a verificar en el registro de Windows la versión de software del reproductor (Windows Media Player por ejemplo). Para este estudio de caso, vamos a reutilizar una parte de los scripts anteriores para obtener la lista de los puestos cliente registrados en el Active Directory.

Para que el acceso al registro de Windows en remoto funcione en equipos Windows 7, verifique que el servicio «Registro remoto (*RemoteRegistry*)» esté iniciado correctamente.

La solución:

```
# Get-MPVersion.ps1
# --- Declaración de las funciones ---
# --- Función para saber si un equipo está en línea ---
function Ping-Host
{
  param ($hostName)
  $tmp = ping.exe $hostName -n 1
  if ($LASTEXITCODE -ne 0)
  {
      $false
  }
  else
  {
      $true
  }
}

# --- Función para conocer la versión de la aplicación ---
function Get-Key
{
  param ([String]$hostName)
  #Acceso al registro de Windows
  $Clave = "SOFTWARE\Microsoft\MediaPlayer\PlayerUpgrade"
  $Tipo = [Microsoft.Win32.RegistryHive]::LocalMachine
  $Clave_Reg = [Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey($tipo,$hostName)
  $Clave_Reg = $Cla_Reg.OpenSubKey($Clave)
  if ($Clave_Reg -eq $null)
  {
      $Version="No Instalada"
  }
  else
  {
     $Lista = $Clave_Reg.GetValueNames()
     foreach($Nombre_Valor in $Lista)
     {
```

```
      if($Nombre_Valor -eq 'PlayerVersion')
     {
      $Version = $Clave_Reg.GetValue('PlayerVersion')
     }
     }
  }
  $Version
}


# --- Inicio del script ---

# - Petición Active Directory -
# Creación de un objeto objDominio que se conecta a un controlador de
dominio
# El hecho de no especificar nada entre las cuotas indica que el script va a
# consultar al primer controlador de dominio que encuentre.
$objDomaine = [ADSI]''

#Creación de un objeto de tipo DirectorySearcher para buscar
#en el servicio de directorio
$objBusqueda = New-Object System.DirectoryServices.DirectorySearcher
($objDominio)

#Creación de la consulta y aplicación de la misma
#Filtra en los equipos Windows.
$consulta = '(&(objectCategory=computer)(name=*)(operatingSystem=Windows*))'
$objBusqueda.Filter = $consulta

#Método de busqueda de cuentas en AD
$listaEquipos = $objBusqueda.Findall()

[PSObject[]]$tabla = $null
# - Bucle sobre toda la lista de servidores devueltos -
foreach ($equipo in $listaEquipos)
{
   $equipo = $equipo.Properties["name"]
   if (ping-host $equipo) #verificación del ping
   {
      $version = Get-Key $equipo
      if($version -ne $null)
      {
         $resultadoTemp = New-Object PSObject
         $resultadoTemp |
            Add-Member -memberType NoteProperty -name Machine -value $equipo
         $resultadoTemp |
            Add-Member -memberType NoteProperty -name Version -value $version
         [PSObject[]]$tabla += [PSObject]$resultadoTemp
      }
   }
   else
   {
      $resultadoTemp = New-Object PSObject
      $resultadoTemp |
```

```
         Add-Member -memberType NoteProperty -name Machine -value $equipo
      $resultadoTemp |
         Add-Member -memberType NoteProperty -name Version -value 'Desconectada'
      [PSObject[]]$tabla += [PSObject]$resultadoTemp
   }
}
$tabla
```

Utilización:

```
./Get-MPVersion
```

Resultado:

```
Machine                                              Version
------                                               --------
{W2K8R2VM}                                           No Instalada
{WIN7_US_X64}                                        12,0,7600,16415
{WINXP}                                              Desconectado
{WIN2K8X64}                                          No Instalada
{W2K3R2SRV}                                          10,0,0,3997
{WINXPPSV2}                                          9,0,0,4503
{EXCH2K10SRV}                                        No Instalada
{WINXPPSV1}                                          9,0,0,4503
```

Algunas aclaraciones:

En primer lugar, nuestro script incorpora una función capaz de devolver el valor de la clave PlayerVersion del registro de Windows de un equipo remoto. Para ello utiliza el método OpenRemoteBaseKey del framework (el acceso a un registro Windows remoto no se efectúa mediante el provider local de PowerShell).

```
function Get-Key
{
   param ([String]$hostName)
   #Acceso al registro de Windows
   $Clave = "SOFTWARE\Microsoft\MediaPlayer\PlayerUpgrade"
   $Tipo = [Microsoft.Win32.RegistryHive]::LocalMachine
   $Clave_Reg = [Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey($tipo,
$hostName)
   $Clave_Reg = $Cla_Reg.OpenSubKey($Clave)
   if ($Clave_Reg -eq $null)
   {
      $Version="No Instalada"
   }
   else
   {
     $Lista = $Clave_Reg.GetValueNames()
     foreach($Nombre_Valor in $Lista)
     {
       if($Nombre_Valor -eq 'PlayerVersion')
     {
       $Version = $Clave_Reg.GetValue('PlayerVersion')
     }
      }
   }
```

```
   $Version
}
```

Con PowerShell, la utilización del método OpenRemoteBaseKey en un equipo remoto se somete a a la condición siguiente: pertenecer a un dominio Windows. En efecto, si esta condición no se cumple, entonces el segundo argumento pasado a este método no deberá ser el nombre de la máquina sino su dirección IP.

A continuación bastará con recorrer el servicio de directorio para aplicar la función Get-Key sobre el conjunto de puestos cliente.