GUIA BURROS SCRIPT WINDOWS
Joel Sánchez Fernández

1. Abrir el puerto en Windows

Lo primero de todo, crearemos nuestro Script. En este caso voy a utilizar Visual Studio para configurar el script.

Le indicamos a nuestro Script el puerto y nombre que queremos, y luego usamos el comando para abrir y depurar el puerto.

```
$puerto = 8080
$nombreRegla = "Abrir puerto $puerto"

New-NetFirewallRule -DisplayName $nombreRegla -Direction Inbound

Write-Host "Se ha abierto el puerto &puerto"
```

Luego para ejecutar este script, usamos lo siguiente:

```
PS C:\WINDOWS\system32> Set-ExecutionPolicy Bypass -Scope Process -Force
>> abrir_puerto.ps1
>>


Name                          : {0c432a64-e2b0-4a59-86eb-5632e14baaa6}
DisplayName                   : Abrir puerto 8080
Description                   :
DisplayGroup                  :
Group                         :
Enabled                       : True
Profile                       : Any
Platform                      : {}
Direction                     : Inbound
Action                        : Allow
EdgeTraversalPolicy           : Block
LooseSourceMapping            : False
LocalOnlyMapping              : False
Owner                         :
PrimaryStatus                 : OK
Status                        : Se analizó la regla correctamente desde el almacén. (65536)
EnforcementStatus             : NotApplicable
PolicyStoreSource             : PersistentStore
PolicyStoreSourceType         : Local
RemoteDynamicKeywordAddresses :
PolicyAppId                   :

Se ha abierto el puerto 8080
```

2. Creamos un servidor de Escucha

Para hacer nuestro servidor de Escucha y podamos observar quien se va conectando al puerto asignado al servidor.

Para esto configuramos nuestro Script para que abra el servidor, en un servidor de escucha:

```powershell
$puerto = 8080
$nombreRegla = "Abrir puerto $puerto"

New-NetFirewallRule -DisplayName $nombreRegla -Direction Inbound -

Write-Host "Se ha abierto el puerto $puerto"

# Configura el puerto de escucha
$listener = [System.Net.Sockets.TcpListener]::new($puerto)

# Iniciar el servidor de escucha
$listener.Start()
Write-Host "Servidor escuchando en el puerto $puerto..."

while ($true) {
    $cliente = $listener.AcceptTcpClient()
    $stream = $cliente.GetStream()
    $reader = New-Object System.IO.StreamReader($stream)
    $data = $reader.ReadLine()

    Write-Host "Mensaje recibido: $data"

    # Responder al cliente
    $writer = New-Object System.IO.StreamWriter($stream)
    $writer.WriteLine("Mensaje recibido en el servidor")
    $writer.Flush()

    # Cerrar la conexión
    $cliente.Close()
}

# Detener el servidor (usar Ctrl + C para interrumpirlo)
$listener.Stop()
```

Cuando ejecutamos el script, el PowerShell es como que se queda esperando a que algo ocurra. Y efectivamente esta haciendo eso:

```
PS C:\WINDOWS\system32> Set-ExecutionPolicy Bypass -Scope Process -Force
>> abrir_puerto.ps1
>>


Name                          : {270889b5-024d-475a-943c-27744fed0545}
DisplayName                   : Abrir puerto 8080
Description                   :
DisplayGroup                  :
Group                         :
Enabled                       : True
Profile                       : Any
Platform                      : {}
Direction                     : Inbound
Action                        : Allow
EdgeTraversalPolicy           : Block
LooseSourceMapping            : False
LocalOnlyMapping              : False
Owner                         :
PrimaryStatus                 : OK
Status                        : Se analizó la regla correctamente desde el almacén. (65536)
EnforcementStatus             : NotApplicable
PolicyStoreSource             : PersistentStore
PolicyStoreSourceType         : Local
RemoteDynamicKeywordAddresses :
PolicyAppId                   :

Se ha abierto el puerto 8080
Servidor escuchando en el puerto 8080...
```

3. Archivo de logs

Para crear nuestro archivo de logs, necesitaremos una linea de codigo, para que nos la cree si no existe.

Lo primero que haremos, sera asignar una ruta para nuestro archivo:

```powershell
# Configura el puerto de escucha y la ruta de 'logs'
$puerto = 8080
$nombreRegla = "Abrir puerto $puerto"
$logFile = "C:\Logs\server_log_8080.txt"

New-NetFirewallRule -DisplayName $nombreRegla -Direction Inbound -LocalPort $puerto -Protocol TCP -Action Allow

Write-Host "Se ha abierto el puerto $puerto"

# Crear directorio si no existe
if (!(Test-Path "C:\Logs")) {
    New-Item -ItemType Directory -Path "C:\Logs" | Out-Null
}

# Iniciar el servidor de escucha
$listener = [System.Net.Sockets.TcpListener]::new($puerto)
$listener.Start()
Write-Host "Servidor escuchando en el puerto $puerto..."

while ($true) {
    $cliente = $listener.AcceptTcpClient()
    $ipCliente = $cliente.Client.RemoteEndPoint.Address.IPAddressToString

    # Guardar en log
    $timestamp = Get-Date -Format "yyyy-MM-dd HH:mm:ss"
    "$timestamp - Conexión desde: $ipCliente" | Out-File -Append -FilePath $logFile
```

Como podemos ver, se ha generado nuestra carpeta ‘Logs’ pero al no tener actividad, pues no nos genera el archivo de momento...

4. Cerrar servidor en 5 minutos.

Lo primero, tendrémos que asignar un tiempo en segundos, en este caso 300 segundos son 5 minutos. Lo añadimos al scripts con el siguiente bloque de comandos:

```powershell
    # Verificar si ha pasado el tiempo de inactividad
    if ((Get-Date) - $ultimaConexion -gt (New-TimeSpan -Seconds $timeout)) {
        Write-Host "No se detectó actividad en los últimos $timeout segundos. Cerrando el servidor."
        break
    }

    Start-Sleep -Seconds 1  # Pequeña espera para reducir uso de CPU
```

Una vez pase ese tiempo, nos saldrá el siguiente mensaje en la consola de PowerShell.

```
Se ha abierto el puerto 8080
Servidor escuchando en el puerto 8080...
No se detectÃ³ actividad en los Ãºltimos 300 segundos. Cerrando el servidor.
Servidor detenido.
```

5. Alerta de la RAM

La alerta de RAM se tiene que poner indicando su limite y si la esta sobrepasando con dos variables. El bloque de comandos es:

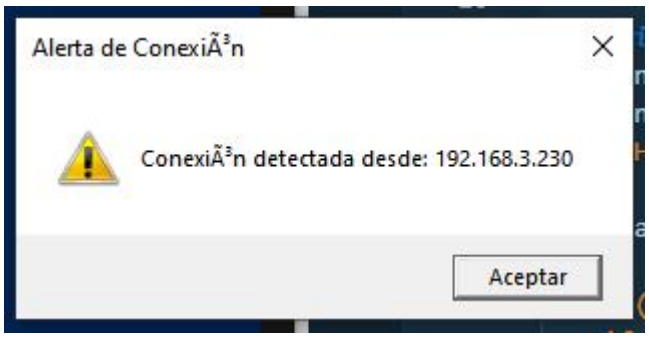
```
# Verificar el uso de RAM
$memoriaLibre = (Get-CimInstance Win32_OperatingSystem).FreePhysicalMemory / 1MB
$memoriaTotal = (Get-CimInstance Win32_OperatingSystem).TotalVisibleMemorySize / 1MB
$ramDisponible = ($memoriaLibre / $memoriaTotal) * 100

if ($ramDisponible -lt $ramThreshold -and -not $alertaMostrada) {
    [System.Windows.Forms.MessageBox]::Show("Advertencia: RAM por debajo del $ramThreshold%
    $alertaMostrada = $true  # Evita que la alerta se repita constantemente
}
elseif ($ramDisponible -ge $ramThreshold) {
    $alertaMostrada = $false  # Resetea la alerta cuando la RAM vuelve a un nivel seguro
}
```

```
$ramThreshold = 25
$alertaMostrada = $false  # Corregido el nombre de la variable
```

El ‘ramThreshold’ sera el límite de nuestra RAM para que nos salga el aviso.

6. Evidencia de que funciona ->

Aqui muestro una conexión en mi servidor:



Y aqui mi archivo de logs: