PowerShellScripts / proxyTunnel.ps1

Arno0x First commit 50b930d · 8 years ago History

Code Blame 147 lines (109 loc) · 4.98 KB Raw

```
<#
.SYNOPSIS

Creates a TCP Tunnel through the default system proxy. As such, it automatically handle
Author: Arno0x0x (https://twitter.com/Arno0x0x)
License: GPL3
Required Dependencies: None
Optional Dependencies: None

.DESCRIPTION

Creates a TCP Tunnel through the default system proxy. As such, it automatically handle

.PARAMETER bindIP

The local IP the tunnel will bind to. Defaults to 127.0.0.1 .

.PARAMETER bindPort

The local TCP port the tunnel will bind to. Defaults to 5555 .

.PARAMETER destHost

The destination host that should be reached through the tunnel.

.PARAMETER destPort

The destination port that should be reached through the tunnel.

.EXAMPLE

powershell .\proxyTunnel.ps1 -bindPort 4444 -destHost myserver.example.com -destPort 22

Then, an SSH connection to 127.0.0.1:4444 will be tunneled, through the corporate proxy
#>

Param (
        [String]$bindIP = "127.0.0.1",

        [Int]$bindPort = 5555,

        [String]$destHost = $( Read-Host "Enter tunnel destination IP or Hostname: " ),

        [Int]$destPort = $( Read-Host "Enter tunnel destination port: " )
    )

$clientBuffer = new-object System.Byte[] 1024
$request = [System.Net.HttpWebRequest]::Create("http://" + $destHost + ":" + $destPort
$request.Method = "CONNECT"

# Detect and set automatic proxy and network credentials
$proxy = [System.Net.WebRequest]::GetSystemWebProxy()
$proxy.Credentials = [System.Net.CredentialCache]::DefaultNetworkCredentials
$request.Proxy = $proxy
```

```
57      $listener = new-object System.Net.Sockets.TcpListener([System.Net.IPAddress]::Parse($bi
58
59
60      #---------------------------------------------------------------------------
61      # This script block is executed in a separate PowerShell object, as another
62      # thread. It reads data from the serverStream and writes it to the clientStream
63      # as long as there's data
64      $Script = {
65              param($state)
66              $serverBuffer = new-object System.Byte[] 1024
67
68              $count = 0
69              do {
70                      $count = $state.serverStream.Read($serverBuffer, 0 ,$serverBuffer.lengt
71                      $state.clientStream.Write($serverBuffer, 0 , $count)
72                      $state.clientStream.Flush()
73              } while ($count -gt 0)
74      }
75
76      #---------------------------------------------------------------------------
77      # Starting the TCP listener
78      $listener.start()
79
80      write-host "Waiting for a connection on port $bindPort..."
81      $client = $listener.AcceptTcpClient()
82      write-host "Connected from $($client.Client.RemoteEndPoint)"
83
84      #---------------------------------------------------------------------------------
85      # Get the client side stream object to read/write to
86      $clientStream = $client.GetStream() # This is a System.Net.Sockets.NetworkStream
87
88      #---------------------------------------------------------------------------------
89      # Get the server side response and corresponding stream object to read/write to
90      $serverResponse = $request.GetResponse()
91      $responseStream = $serverResponse.GetResponseStream()
92
93      #---------------------------------------------------------------------------------
94      # Reflection inspection to retrieve and reuse the underlying networkStream instance
95      $BindingFlags= [Reflection.BindingFlags] "NonPublic,Instance"
96      $rsType = $responseStream.GetType()
97      $connectionProperty = $rsType.GetProperty("Connection", $BindingFlags)
98      $connection = $connectionProperty.GetValue($responseStream, $null)
99      $connectionType = $connection.GetType()
100     $networkStreamProperty = $connectionType.GetProperty("NetworkStream", $BindingFlags)
101     $serverStream = $networkStreamProperty.GetValue($connection, $null)
102
103     # This state object is used to pass various object by reference to the child PowerShell
104     # that is created afterwards
105     $state = [PSCustomObject]@{"serverStream"=$serverStream;"clientStream"=$clientStream}
106
107     # Create a child PowerShell object to run the background Socket receive method.
108     $PS = [PowerShell]::Create()
109     $PS.AddScript($Script).AddArgument($state) | Out-Null
110     [System.IAsyncResult]$AsyncJobResult = $null
111
112     try
113     {
114             # The receive job is started asynchronously.
115             $AsyncJobResult = $PS.BeginInvoke()
116
117             do {
118                     $bytesReceived = $clientStream.Read($clientBuffer, 0, $clientBuffer.len
119                     $serverStream.Write($clientBuffer, 0 , $bytesReceived)
120                     #$text = [System.Text.Encoding]::ASCII.GetString($buffer, 0, $bytesRece
121                     #Write-Host $text
122
123             } while ($client.Connected -or $clientStream.DataAvailable)
124     }
125     catch {
126             $ErrorMessage = $_.Exception.Message
127             Write-Host $ErrorMessage
128     }
129     finally {
130             # Cleanup the client socket and child PowerShell process.
131         if ($client -ne $null) {
```

```
132             $client.Close()
133             $client.Dispose()
134             $client = $null
135         }
136 
137             if ($listener -ne $null) {
138                     $listener.Stop()
139             }
140 
141             write-host "Connection closed."
142 
143         if ($PS -ne $null -and $AsyncJobResult -ne $null) {
144             $PS.EndInvoke($AsyncJobResult)
145             $PS.Dispose()
146         }
147     }
```