Invoke-Obfuscation / Out-ObfuscatedStringCommand.ps1

904 lines (704 loc) · 97.9 KB

Code | Blame | Raw

```
1   #   This file is part of Invoke-Obfuscation.
2   #
3   #   Copyright 2017 Daniel Bohannon <@danielhbohannon>
4   #         while at Mandiant <http://www.mandiant.com>
5   #
6   #   Licensed under the Apache License, Version 2.0 (the "License");
7   #   you may not use this file except in compliance with the License.
8   #   You may obtain a copy of the License at
9   #
10  #       http://www.apache.org/licenses/LICENSE-2.0
11  #
12  #   Unless required by applicable law or agreed to in writing, software
13  #   distributed under the License is distributed on an "AS IS" BASIS,
14  #   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15  #   See the License for the specific language governing permissions and
16  #   limitations under the License.
17
18
19
20  Function Out-ObfuscatedStringCommand
21  {
22  <#
23  .SYNOPSIS
24
25  Master function that orchestrates the application of all string-based obfuscation functions to prov
```

```
26
27      Invoke-Obfuscation Function: Out-ObfuscatedStringCommand
28      Author: Daniel Bohannon (@danielhbohannon)
29      License: Apache License, Version 2.0
30      Required Dependencies: Out-EncapsulatedInvokeExpression (located in Out-ObfuscatedStringCommand.ps1
31      Optional Dependencies: None
32
33      .DESCRIPTION
34
35      Out-ObfuscatedStringCommand orchestrates the application of all string-based obfuscation functions
36      The available ObfuscationLevel/function mappings are:
37      1 --> Out-StringDelimitedAndConcatenated
38      2 --> Out-StringDelimitedConcatenatedAndReordered
39      3 --> Out-StringReversed
40
41      .PARAMETER ScriptBlock
42
43      Specifies a scriptblock containing your payload.
44
45      .PARAMETER Path
46
47      Specifies the path to your payload.
48
49      .PARAMETER ObfuscationLevel
50
51      (Optional) Specifies the obfuscation level for the given input PowerShell payload. If not defined t
52      The available ObfuscationLevel/function mappings are:
53      1 --> Out-StringDelimitedAndConcatenated
54      2 --> Out-StringDelimitedConcatenatedAndReordered
55      3 --> Out-StringReversed
56
57      .EXAMPLE
58
59      C:\PS> Out-ObfuscatedStringCommand {Write-Host 'Hello World!' -ForegroundColor Green; Write-Host 'C
60
61      IEX ((('Write-H'+'ost x'+'lcHello'+' Wor'+'ld!xlc -F'+'oregroundC'+'o'+'lor Gre'+'en'+'; Write-Host
62
63      C:\PS> Out-ObfuscatedStringCommand {Write-Host 'Hello World!' -ForegroundColor Green; Write-Host 'C
64
65      IEX( (("{17}{1}{6}{19}{14}{3}{5}{13}{16}{11}{20}{15}{10}{12}{2}{4}{8}{18}{7}{9}{0}" -f ' Green','-H
66
67      C:\PS> Out-ObfuscatedStringCommand {Write-Host 'Hello World!' -ForegroundColor Green; Write-Host 'C
68
69      $I4 ="noisserpxE-ekovnI|)93]rahC[]gnirtS[,'1Yp'(ecalpeR.)'ne'+'erG roloCd'+'nuo'+'rgero'+'F- 1'+'Y'
70
71      .NOTES
```

```
72
73      Out-ObfuscatedStringCommand orchestrates the application of all string-based obfuscation functions
74      This is a personal project developed by Daniel Bohannon while an employee at MANDIANT, A FireEye Co
75
76      .LINK
77
78      http://www.danielbohannon.com
79      #>
80
81          [CmdletBinding( DefaultParameterSetName = 'FilePath')] Param (
82              [Parameter(Position = 0, ValueFromPipeline = $True, ParameterSetName = 'ScriptBlock')]
83              [ValidateNotNullOrEmpty()]
84              [ScriptBlock]
85              $ScriptBlock,
86
87              [Parameter(Position = 0, ParameterSetName = 'FilePath')]
88              [ValidateNotNullOrEmpty()]
89              [String]
90              $Path,
91
92              [ValidateSet('1', '2', '3')]
93              [Parameter(Position = 1)]
94              [ValidateNotNullOrEmpty()]
95              [Int]
96              $ObfuscationLevel = (Get-Random -Input @(1..3)) # Default to random obfuscation level if $O
97          )
98
99          # Either convert ScriptBlock to a String or convert script at $Path to a String.
100         If($PSBoundParameters['Path'])
101         {
102             Get-ChildItem $Path -ErrorAction Stop | Out-Null
103             $ScriptString = [IO.File]::ReadAllText((Resolve-Path $Path))
104         }
105         Else
106         {
107             $ScriptString = [String]$ScriptBlock
108         }
109
110         # Set valid obfuscation levels for current token type.
111         $ValidObfuscationLevels = @(0,1,2,3)
112
113         # If invalid obfuscation level is passed to this function then default to highest obfuscation l
114         If($ValidObfuscationLevels -NotContains $ObfuscationLevel) {$ObfuscationLevel = $ValidObfuscati
115
116         Switch($ObfuscationLevel)
117         {
```

```
117             {
118                 0 {Continue}
```

```
831     HELPER FUNCTION :: Generates random syntax for invoking input PowerShell command.
832
833     Invoke-Obfuscation Function: Out-EncapsulatedInvokeExpression
834     Author: Daniel Bohannon (@danielhbohannon)
835     License: Apache License, Version 2.0
836     Required Dependencies: None
837     Optional Dependencies: None
838
839     .DESCRIPTION
840
841     Out-EncapsulatedInvokeExpression generates random syntax for invoking input PowerShell command. It
842
843     .PARAMETER ScriptString
844
845     Specifies the string containing your payload.
846
847     .EXAMPLE
848
849     C:\PS> Out-EncapsulatedInvokeExpression {Write-Host 'Hello World!' -ForegroundColor Green; Write-Ho
```

```
850
851     Write-Host 'Hello World!' -ForegroundColor Green; Write-Host 'Obfuscation Rocks!' -ForegroundColor
852
853     .NOTES
854
855     This cmdlet is most easily used by passing a script block or file path to a PowerShell script into
856     C:\PS> Out-ObfuscatedStringCommand {Write-Host 'Hello World!' -ForegroundColor Green; Write-Host 'C
857     C:\PS> Out-ObfuscatedStringCommand {Write-Host 'Hello World!' -ForegroundColor Green; Write-Host 'C
858     C:\PS> Out-ObfuscatedStringCommand {Write-Host 'Hello World!' -ForegroundColor Green; Write-Host 'C
859     This is a personal project developed by Daniel Bohannon while an employee at MANDIANT, A FireEye Co
860
861     .LINK
862
863     http://www.danielbohannon.com
864     #>
865
866         [CmdletBinding()] Param (
867             [Parameter(Position = 0)]
868             [ValidateNotNullOrEmpty()]
869             [String]
870             $ScriptString
871         )
872
873         # The below code block is copy/pasted into almost every encoding function so they can maintain
874         # Changes to below InvokeExpressionSyntax block should also be copied to those functions.
875         # Generate random invoke operation syntax.
876         $InvokeExpressionSyntax  = @()
877         $InvokeExpressionSyntax += (Get-Random -Input @('IEX','Invoke-Expression'))
878         # Added below slightly-randomized obfuscated ways to form the string 'iex' and then invoke it w
879         # Though far from fully built out, these are included to highlight how IEX/Invoke-Expression is
880         # These methods draw on common environment variable values and PowerShell Automatic Variable va
881         $InvocationOperator = (Get-Random -Input @('.','&')) + ' '*(Get-Random -Input @(0,1))
882         $InvokeExpressionSyntax += $InvocationOperator + "( `$ShellId[1]+`$ShellId[13]+'x')"
883         $InvokeExpressionSyntax += $InvocationOperator + "( `$PSHome[" + (Get-Random -Input @(4,21)) +
884         $InvokeExpressionSyntax += $InvocationOperator + "( `$env:ComSpec[4," + (Get-Random -Input @(15
885         $InvokeExpressionSyntax += $InvocationOperator + "((" + (Get-Random -Input @('Get-Variable','GV
886         $InvokeExpressionSyntax += $InvocationOperator + "( " + (Get-Random -Input @('$VerbosePreferenc
887         # Commenting below option since $env:Public differs in string value for non-English operating s
888         #$InvokeExpressionSyntax += $InvocationOperator + "( `$env:Public[13]+`$env:Public[5]+'x')"
889
890         # Randomly choose from above invoke operation syntaxes.
891         $InvokeExpression = (Get-Random -Input $InvokeExpressionSyntax)
892
893         # Randomize the case of selected invoke operation.
894         $InvokeExpression = Out-RandomCase $InvokeExpression
895
```

```
896         # Choose random Invoke-Expression/IEX syntax and ordering: IEX ($ScriptString) or ($ScriptStrir
897         $InvokeOptions  = @()
898         $InvokeOptions += ' '*(Get-Random -Input @(0,1)) + $InvokeExpression + ' '*(Get-Random -Input @
899         $InvokeOptions += ' '*(Get-Random -Input @(0,1)) + $ScriptString + ' '*(Get-Random -Input @(0,1
900 
901         $ScriptString = (Get-Random -Input $InvokeOptions)
902 
903         Return $ScriptString
904     }
```