





Product Solutions Resources Open Source Enterprise Pricing



Sign in

Sign up

danielbohannon / Invoke-Obfuscation

Public

 Notifications

 Fork 768

 Star 3.7k

<> Code

 Issues 11

 Pull requests 2

 Actions


 Projects

 Security


 Insights


Invoke-Obfuscation / Out-ObfuscatedStringCommand.ps1



 Daniel Bohannon

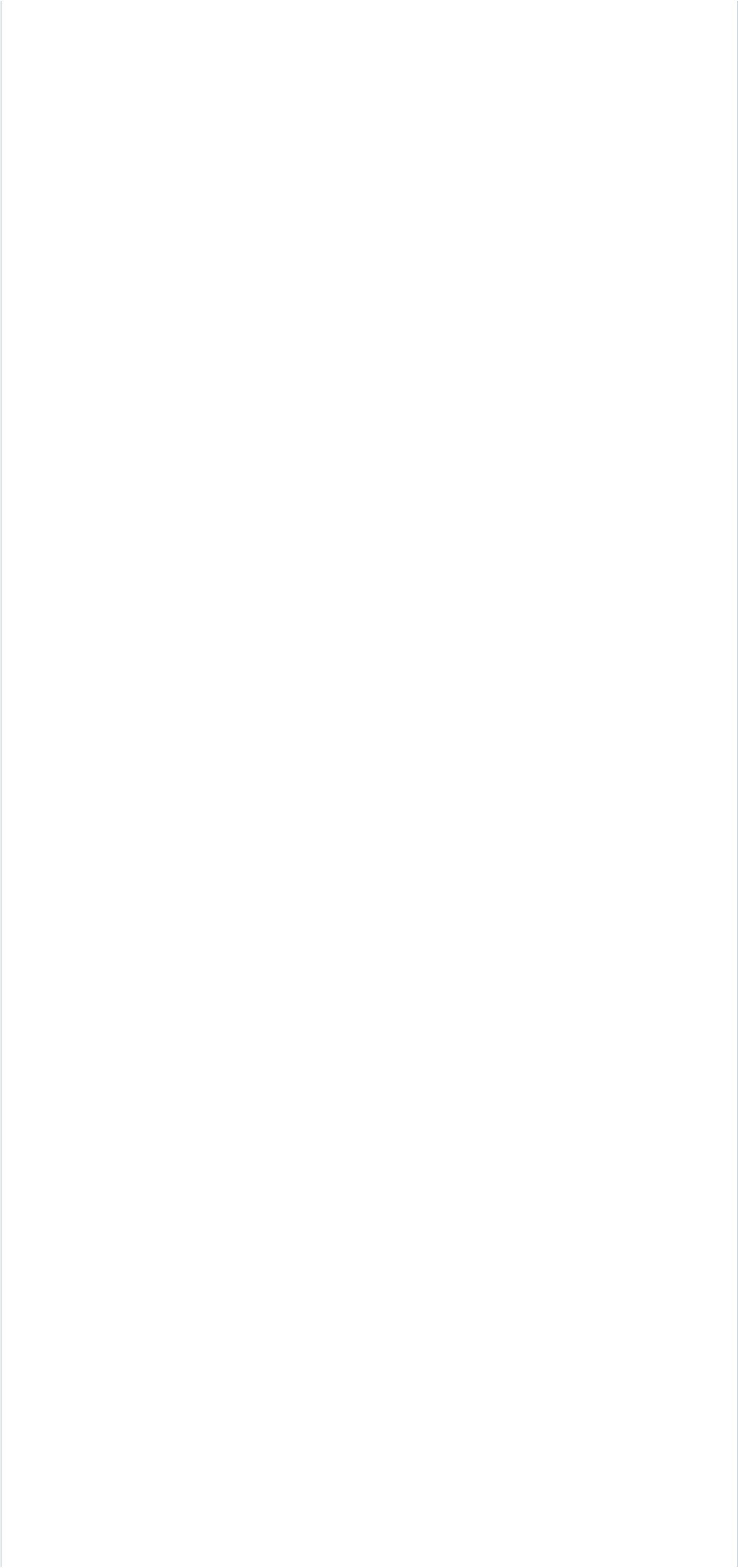
Removing \$env:Public invocation option for co...

 0e31fab · 7 years ago

 History

```
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 funct
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-ObfuscatedStrin
31     Optional Dependencies: None
32
33     .DESCRIPTION
34
35     Out-ObfuscatedStringCommand orchestrates the application of all string-based obfuscatio
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 n
52     The available ObfuscationLevel/function mappings are:
53     1 --> Out-StringDelimitedAndConcatenated
54     2 --> Out-StringDelimitedConcatenatedAndReordered
55     3 --> Out-StringReversed
56
57     EXAMPLE E
```

```
57 .EXAMPLE
58
59 C:\PS> Out-ObfuscatedStringCommand {Write-Host 'Hello World!' -ForegroundColor Green; W
60
61 IEX (((Write-H'+ost x'+lcHello'+ Wor'+ld!xlc -F'+oregroundC'+o'+lor Gre'+en'+
62
63 C:\PS> Out-ObfuscatedStringCommand {Write-Host 'Hello World!' -ForegroundColor Green; W
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
66
67 C:\PS> Out-ObfuscatedStringCommand {Write-Host 'Hello World!' -ForegroundColor Green; W
68
69 $I4 ="noisserpxE-ekovnI|)93]rahC[]gnirtS[, '1Yp'(ecalpeR.)'ne'+erG roloCd'+nuo'+rgero
70
71 .NOTES
72
73 Out-ObfuscatedStringCommand orchestrates the application of all string-based obfuscatio
74 This is a personal project developed by Daniel Bohannon while an employee at MANDIANT,
75
```








```
754         # Encapsulate in necessary IEX/Invoke-Expression(s).
755         $JoinOption = Out-EncapsulatedInvokeExpression $JoinOption
756
757         $ScriptString = $ScriptString + $JoinOption
758     }
759     2 {
760         # 2) $StringVar = [Char[]]$String; [Array]::Reverse($StringVar); $StringVar
761
762         # Replace placeholder with appropriate value for this Switch statement.
763         $RandomVarSet = $RandomVarSet.Replace($RandomVarValPlaceholder, ("[$CharStr[
764
765         # Set $ScriptStringReversed as environment variable $Random.
766         $ScriptString = $RandomVarSet + ' '*(Get-Random -Input @(0,1)) + ';' + ' '*
767         $ScriptString = $ScriptString + ' '*(Get-Random -Input @(0,1)) + "[$ArraySt
768
769         # Build out random syntax depending on whether -Join is prepended or -Join
770         # Now also includes [String]::Join .Net syntax and [String] syntax after mo
771         $JoinOptions = @()
772         $JoinOptions += "-$JoinStr" + ' '*(Get-Random -Input @(0,1)) + $RandomVarGe
773         $JoinOptions += $RandomVarGet + ' '*(Get-Random -Input @(0,1)) + "-$JoinStr
774         $JoinOptions += "[$StringStr]::$JoinStr" + '(' + ' '*(Get-Random -Input @(0
775         $JoinOptions += "'" + ' '*(Get-Random -Input @(0,1)) + '$(' + ' '*(Get-Rand
776         $JoinOption = (Get-Random -Input $JoinOptions)
777
778         # Encapsulate in necessary IEX/Invoke-Expression(s).
779         $JoinOption = Out-EncapsulatedInvokeExpression $JoinOption
780
781         $ScriptString = $ScriptString + $JoinOption
782     }
783     3 {
784         # 3) -Join[Regex]::Matches($String, '.', 'RightToLeft')
785
786         # Randomly choose to use 'RightToLeft' or concatenated version of this stri
787         If(Get-Random -Input (0..1))
788         {
789             $RightToLeft = Out-ConcatenatedString $RightToLeftStr ""
790         }
791         Else
792         {
793             $RightToLeft = ""$RightToLeftStr""
794         }
795
796         # Build out random syntax depending on whether -Join is prepended or -Join
797         # Now also includes [String]::Join .Net syntax and [String] syntax after mo
798         $JoinOptions = @()
799         $JoinOptions += ' '*(Get-Random -Input @(0,1)) + '(' + ' '*(Get-Random -Inp
800         $JoinOptions += ' '*(Get-Random -Input @(0,1)) + '(' + ' '*(Get-Random -Inp
801         $JoinOptions += ' '*(Get-Random -Input @(0,1)) + "[$StringStr]::$JoinStr("
802         $JoinOptions += "'" + ' '*(Get-Random -Input @(0,1)) + '$(' + ' '*(Get-Rand
803         $ScriptString = (Get-Random -Input $JoinOptions)
```

Files

f20e7f8

Go to file

Invoke-Obfuscation.ps1

Invoke-Obfuscation.psd1

Invoke-Obfuscation.psm1

LICENSE

Out-CompressedCommand.ps1

Out-EncodedAsciiCommand.ps1

Out-EncodedBXORCommand.ps1

Out-EncodedBinaryCommand.ps1

Out-EncodedHexCommand.ps1

Out-EncodedOctalCommand.ps1

```
803         $ScriptString = (Get-Random -Input $JoinOptions)
804
805         # Encapsulate in necessary IEX/Invoke-Expression(s).
806         $ScriptString = Out-EncapsulatedInvokeExpression $ScriptString
807     }
808     default {Write-Error "An invalid value was passed to switch block."; Exit;}
809 }
810
811 # Perform final check to remove ticks if they now precede lowercase special charact
812 # E.g. "testin`G" in reverse would be "G`nitset" where `n would be interpreted as a
813 $SpecialCharacters = @('a','b','f','n','r','u','t','v','0')
814 ForEach($SpecialChar in $SpecialCharacters)
815 {
816     If($ScriptString.Contains("`"+$SpecialChar))
817     {
818         $ScriptString = $ScriptString.Replace("`"+$SpecialChar,$SpecialChar)
819     }
820 }
821
822 Return $ScriptString
823 }
824
825
826 Function Out-EncapsulatedInvokeExpression
827 {
828 <#
829 .SYNOPSIS
830
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
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 Gre
850
851 Write-Host 'Hello World!' -ForegroundColor Green; Write-Host 'Obfuscation Rocks!' -Fore
852
853 .NOTES
854
```

Invoke-Obfuscation / Out-ObfuscatedStringCommand.ps1 ↑ Top

Code

Blame

904 lines (704 loc) · 97.9 KB

Raw

```
859 This is a personal project developed by Daniel Bohannon while an employee at MANDIANI,
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 c
874 # Changes to below InvokeExpressionSyntax block should also be copied to those func
875 # Generate random invoke operation syntax.
876 $InvokeExpressionSyntax = @()
877 $InvokeExpressionSyntax += (Get-Random -Input @('IEX','Invoke-Expression'))
```

- Out-EncodedSpecialCharOnlyCo...
- Out-EncodedWhitespaceComma...
- Out-ObfuscatedAst.ps1
- Out-ObfuscatedStringCommand....
- Out-ObfuscatedTokenCommand....
- Out-PowerShellLauncher.ps1
- Out-SecureStringCommand.ps1
- README.md

```
878 # Added below slightly-randomized obfuscated ways to form the string 'iex' and then
879 # Though far from fully built out, these are included to highlight how IEX/Invoke-E
880 # These methods draw on common environment variable values and PowerShell Automatic
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
884 $InvokeExpressionSyntax += $InvocationOperator + "(`$env:ComSpec[4," + (Get-Random
885 $InvokeExpressionSyntax += $InvocationOperator + "(" + (Get-Random -Input @('Get-V
886 $InvokeExpressionSyntax += $InvocationOperator + "( " + (Get-Random -Input @('$Verb
887 # Commenting below option since $env:Public differs in string value for non-English
888 #$InvokeExpressionSyntax += $InvocationOperator + "(`$env:Public[13]+`$env:Public[
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 (
897 $InvokeOptions = @()
898 $InvokeOptions += ' '*(Get-Random -Input @(0,1)) + $InvokeExpression + ' '*(Get-Ran
899 $InvokeOptions += ' '*(Get-Random -Input @(0,1)) + $ScriptString + ' '*(Get-Random
900
901 $ScriptString = (Get-Random -Input $InvokeOptions)
902
903 Return $ScriptString
904 }
```