Invoke-Obfuscation #1009

Closed zinint opened this issue on Sep 14, 2020 · 25 comments

zinint commented on Sep 14, 2020 • edited Contributor

Summary

- Tool: Invoke-Obfuscation — PowerShell command and script obfuscation framework
- Author: Daniel Bohannon, @danielhbohannon
- Type: Offensive tool, threat simulation
- Materials:
  - The Invoke-Obfuscation Usage Guide :: Part 1;
  - The Invoke-Obfuscation Usage Guide :: Part 2;
  - Invoke-Obfuscation: PowerShell obFUsk8tion Techniques & How To (Try To) D""e `Tec` T 'Th'+'em'

Problem

Sigma rules heavily rely on process execution (with command-line) events (Windows Event Log Security Event ID 4688 and Sysmon Event ID 1).
Many of them provide detection of malicious PowerShell one-liners.
At the same time, the presence of Sigma rules for Powershell Obfuscation Indicators detection is quite limited.

There are a five Sigma rules for PowerShell obfuscation detection, developed by Thomas Patzke (**@thomaspatzke**), Florian Roth (**@Neo23x0**), Sami Ruohonen (**@samsson**) and Harish Segar (**@HarishHary**):

- Suspicious XOR Encoded PowerShell Command Line (812837bb-b17f-45e9-8bd0-0ec35d2e3bd6)
- Suspicious XOR Encoded PowerShell Command Line (bb780e0c-16cf-4383-8383-1e5471db6cf9)
- Suspicious PowerShell Parameter Substring (36210e0d-5b19-485d-a087-c096088885f0)
- CrackMapExec PowerShell Obfuscation (6f8b3439-a203-45dc-a88b-abf57ea15ccf)
- CrackMapExec Command Execution (058f4380-962d-40a5-afce-50207d36d7e2)

At the same time, there and only three Sigma rules (developed by Daniel Bohannon, @danielhbohannon) that are focusing on detection of one of the obfuscation functions (obfuscated IEX invocation) provided by Invoke-Obfuscation framework.

There are at least 30 more obfuscation methods that Invoke-Obfuscation framework provides.
We would like to collaborate on Sigma rules development in this area.

Solution

We developed a table with pre-generated PowerShell commands, obfuscated by the Invoke-Obfuscation framework, you can pick up some of the tasks in that table and develop Sigma rules for them. You will need to use regular expression value modifier, provided by Sigma converter (sigmac).

Here is an example of Sigma rule that utilizes a regular expression value modifier (`|re`):

```
title: Invoke-Obfuscation obfuscated IEX invocatic
id: 4bf943c6-5146-4273-98dd-e958fd1e3abf
description: "Detects all variations of obfuscated power
status: experimental
author: Daniel Bohannon (@Mandiant/@FireEye), oscd.commu
date: 2019/11/08
tags:
```

```
    - attack.defense_evasion
    - attack.t1027
logsource:
    product: windows
    service: process_creation
detection:
    selection:
        - CommandLine|re: '\$PSHome\[\s*\d{1,3}\s*\]\s*\
        - CommandLine|re: '\$ShellId\[\s*\d{1,3}\s*\]\s*
        - CommandLine|re: '\$env:Public\[\s*\d{1,3}\s*\]
        - CommandLine|re: '\$env:ComSpec\[(\s*\d{1,3}\s*
        - CommandLine|re: '\*mdr\*\W\s*\)\.Name'
        - CommandLine|re: '\$VerbosePreference\.ToString
        - CommandLine|re: '\String\]\s*\$VerbosePreferen
    condition: selection
falsepositives:
    - Unknown
level: high
```

The approach

We developed a table with pre-generated PowerShell commands, obfuscated by the Invoke-Obfuscation framework. The description of the approach is following.

Original code (before obfuscation)

```
# command example
Invoke-Expression (New-Object Net.WebClient).DownloadStr
# variable example
$env:path
# type token example
[Scriptblock]::Create("Write-Host $env:path")
```

The main goal is to detect the obfuscation method itself, not a specific command

Some of the obfuscation methods are already covered by Sigma rules, developed by the Invoke-Obfuscation author. He used the following regexes in the rules:

```
\$PSHome\[\s*\d{1,3}\s*\]\s*\+\s*\$PSHome\[
\$ShellId\[\s*\d{1,3}\s*\]\s*\+\s*\$ShellId\[
\$env:Public\[\s*\d{1,3}\s*\]\s*\+\s*\$env:Public\[
\$env:ComSpec\[(\s*\d{1,3}\s*,){2}
```

```
\*mdr\*\W\s*\)\.Name
\$VerbosePreference\.ToString\(
\String\]\s*\$VerbosePreference
```

These regexes provide detection of the IEX invocation obfuscation function. This function is included into almost every encoding method so they can maintain zero dependencies and work on their own. That's why you'll see similar obfuscation results in different tasks, but it shouldn't distract you from the main goal.

Let's walk through the task 28 to get more details on the regex development approach:

1. Copy all obfuscated commands examples into Sublime or other text editor of your choice

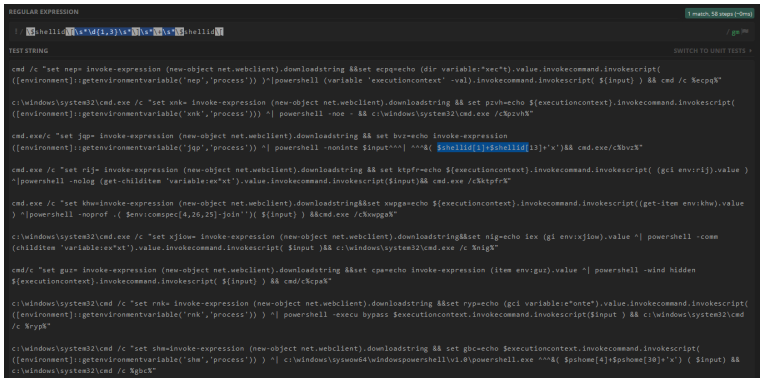
2. Select all examples and lowercase them. In Sublime you can do it by pressing `Ctrl+k, Ctrl+l` (Windows) / `CMD+k, CMD+l` (Mac)

3. Paste the lowecased examples to the regex editor of your choice

4. Start to apply lowercased regexes from existing Sigma rule created by Daniel Bohannon one by one:

   4.1. Regex `\$pshome\[\s*\d{1,3}\s*\]\s*\+\s*\$pshome\[` covers only one example (9th):

   4.2. Regex `\$shellid\[\s*\d{1,3}\s*\]\s*\+\s*\$shellid\[` covers only one example (3rd):



4.3. Regex `\$env:public\[\s*\d{1,3}\s*\]\s*\+\s*\$env:public\[` doesn't cover any examples.

4.4. Regex `\$env:comspec\[(\s*\d{1,3}\s*,){2}` covers only one example (5th):

4.5. Regex `\*mdr\*\w\s*\)\.name` doesn't cover any examples.

4.6. Regex `\$verbosepreference\.tostring\(` doesn't cover any examples.

4.7. Regex `\string\]\s*\$verbosepreference` doesn't cover any examples.

5. Start to develop your own regex that will cover all of the obfuscation examples of this particuar obfuscation method, e.g.:

   5.1. Regex `.*cmd.*\/c.*\^\|.*powershell.*&&.*cmd.*\/c` covers all examples:

This is our main goal - detect the obfuscation method looking for similar patterns in all of it obfuscation examples.

A little tip for the regex development

You can copy all pre-generated obfuscated powershell one-liners from a particular task (that are generated by a specific obfuscation

method) and paste them to regex101 web-app for regular expression development. It will simplify the process a lot, and help you to find patterns to detect. (you can save your progress there and even apply a dark theme (:).

One obfuscation method = 3 Sigma rules

Each Sigma rule for a specific PowerShell obfuscation method should be developed for `process_creation` log category, **service creation** events (windows system eid 7045, windows sysmon eid 6, windows security eid 4697) and `powershell` log source. You can follow the approach used for obfuscated IEX invocation rules — there are 3 rules that rely on the same set of regular expressions:

- rules/windows/process_creation/win_invoke_obfuscation_obfuscated_iex_commandline.yml
- rules/windows/powershell/powershell_invoke_obfuscation_obfuscated_iex.yml
- rules/windows/builtin/win_invoke_obfuscation_obfuscated_iex_services.yml

Case Sensitivity

We consider that we're able to apply all regexes as not case sensitive or that all events are lowercased in a log pipeline before indexing in SIEM/LM system.

Tasks

If you would like to assign yourself to some of the Tasks listed below, you should comment on the Issue with a specific Task you are going to solve. This way, the other participants will see that you will work on a particular task so they will do something else and not intersect with you.

SINGLE OBFUSCATION

- TOKEN OBFUSCATION
- STRING OBFUSCATION
- ENCODING OBFUSCATION
- COMPRESS OBFUSCATION

- PS LAUNCHER OBFUSCATION
- CMD LAUNCHER OBFUSCATION
- WMIC LAUNCHER OBFUSCATION
- RUNDLL LAUNCHER OBFUSCATION
- VAR+ LAUNCHER OBFUSCATION
- STDIN+ LAUNCHER OBFUSCATION
- CLIP+ LAUNCHER OBFUSCATION
- VAR++ LAUNCHER OBFUSCATION
- STDIN++ LAUNCHER OBFUSCATION
- CLIP++ LAUNCHER OBFUSCATION
- RUNDLL++ LAUNCHER OBFUSCATION
- MSHTA++ LAUNCHER OBFUSCATION

TOKEN OBFUSCATION

Back to the Contents 📄

TOKEN\STRING\1&2 skipped, because there are not any String tokens to obfuscate, but they do Concatenate and Reoder just like TOKEN\ARGUMENT\3&4 (Tasks #4&5)

Task #	Option	

1	TOKEN\COMMAND\1 TOKEN\ARGUMENT\2 TOKEN\MEMBER\2	**TOKEN\COMMAND\1** IN`V`o`Ke-eXp`ResSIOn (Ne`V IN`V`OKE-exPRE`Ss`i`oN (n`e\ IN`VOke-expr`eSS`ioN (NE`w- **TOKEN\ARGUMENT\2** Invoke-Expression (New-Obj Invoke-Expression (New-Obj Invoke-Expression (New-Obj **TOKEN\MEMBER\2** Invoke-Expression (New-Obj Invoke-Expression (New-Obj Invoke-Expression (New-Obj
2	TOKEN\COMMAND\2	&('In'+'voke-Expressi'+'o'+'n .('Inv'+'oke-Ex'+'pr'+'ess'+'io .('Invok'+'e-'+'Ex'+'pressio'+' &('Invok'+'e-'+'Expr'+'ession
3	TOKEN\COMMAND\3	&("{3}{4}{2}{1}{0}{5}"-f'o','essi' .("{0}{3}{2}{1}{4}" -f'I','-Ex','oke .("{2}{3}{0}{1}"-f'o','n','Invoke- &("{2}{3}{0}{4}{1}"-f 'e','Expres
4	TOKEN\ARGUMENT\3 TOKEN\MEMBER\3	**TOKEN\ARGUMENT\3** Invoke-Expression (New-Obj Invoke-Expression (New-Obj Invoke-Expression (New-Obj **TOKEN\MEMBER\3** Invoke-Expression (New-Obj Invoke-Expression (New-Obj

		Invoke-Expression (New-Obj
5	TOKEN\ARGUMENT\4 TOKEN\MEMBER\4	**TOKEN\ARGUMENT\4** Invoke-Expression (New-Obj Invoke-Expression (New-Obj Invoke-Expression (New-Obj **TOKEN\MEMBER\4** Invoke-Expression (New-Obj Invoke-Expression (New-Obj Invoke-Expression (New-Obj
6	TOKEN\VARIABLE\1	${En`V:`p`ATh}  ${e`Nv:pATh} ${ENv:`path}
7	TOKEN\TYPE\1	Set-ItEM VaRIABLe:Lcx ([TyP sV ("5Y"+"X") ([typE]('SCrIpT SET F9cg ([tYpE]('scr'+'I'+'PT SET-Variable ('V'+'IR') ([TyPE]
8	TOKEN\TYPE\2	Set-itEM vaRiAbLE:YsB ([tYPe $env:path") set-ITEm ('VAri'+'aBL'+'E'+':Y ('VARI'+'aBL'+'e'+':y'+'7w8O SEt-ItEM ('vAriAb'+'l'+'e:p87 ('VaRiab'+'L'+'E:P87Z2')).vaL $094 = [tyPE]("{1}{0}{3}{2}"-F
9	TOKEN\ALL\1	.("{0}{3}{1}{2}{4}{5}" -f 'Inv','Ex ("{2}{0}{1}{3}" -f 'ownl','oad','D .("{1}{0}{4}{3}{2}" -f'e-E','Invok {0}{3}{2}{4}{1}" -f'Do','ing','I','w

		&("{0}{1}{3}{2}"-f'I','nvoke','es ("{1}{2}{3}{0}" -f'g','Download &("{3}{4}{1}{0}{2}" -f'si','pres',' {2}{3}{0}" -f'g','Down','load','St .("{3}{2}{0}{1}"-f 're','ssion','-Ex f'Client','t.','Ne','We','b')).("{0}{

STRING OBFUSCATION

Back to the Contents

Task #	Option	Results	Comments
10	STRING\1 STRING\2 STRING\3	**Covered by the Invoke-Obfuscation author himself, even for the method commented out in the code:** Rule # 1 Rule # 2 Rule # 3 **You'll encounter patterns from these rules further on, that's because the source code block is copy/pasted into almost every encoding function so they can maintain zero dependencies**	These options can Concatenate entire command || Reorder entire command after concatenating || Reverse entire command after concatenating

		and work on their own. Again, don't hesitate to check the work done and improve it, if you know how.	

ENCODING OBFUSCATION

Back to the Contents

Task #	Option	
11	ENCODING\1	**Partialy covered by the same Sigma** IEx([StrING]::JOin('', ('34@32@36:40 32P44z52T48u32@44T55_56u44_49 32T44u49R49_54R44T52T49u44~52 116@123~32z40T91k105T110~116 "$(SET-ItEM 'vARiABLE:oFs' '')"+[STrI ('73%110q118q111<107x101K45!6 inVoKe-ExPResSion (-jOiN((73 , 110,1
12	ENCODING\2	**Partialy covered by the same Sigma** -joIn ('49_6e-76_6fP6b_65{2d!45_78 ('49}6eU76w6f:6b:65U2dV45w78V7 IEX([StRIng]::jOin('' ,('49>6ex76~6f>6 "$(sEt-ITeM 'VarIABle:ofs' '') " +[STrin
13	ENCODING\3	**Partialy covered by the same Sigma** IEX (-jOIn ('111x156P166<157C153 [STRinG]::JOiN('',((111,156 ,166 , 157 INvOkE-EXpReSsION (" $(sET-vAriA

		[STRINg]::JOIN('', ('111V156~166~1!
14	ENCODING\4	**Partialy covered by the same Sigma** iNvOKE-EXPReSsiON (((1001001 , 1 [COnveRT]::toinT16(([sTriNG]$_) ,2)) Iex ([stRIng]::jOIN('' , ((1001001 , 11C 2)-as [CHaR]) }))) ((1001001 ,1101110,1110110, 110 JoiN ''| INvOKE-eXpRessiON IEX(-jOIN ('1001001C1101110M111 SPlIT'x'-SPlit 'M' -spLIt'C'-SPLiT'!'-spll
15	ENCODING\5	**Partialy covered by the same Sigma** ([rUnTImE.InteropSErvICes.mARShAL] DYANwA3ADQAMwBiAGYANwA1AG)))|ieX ([RuntimE.intEropseRvICes.MArsHAl]:: xAGEAMgAwADMANwAwAGYAYwA0 SeCuRESTriNG -K (45..14))))) | INvOkE ([rUNTiMe.intEROpSErvIcEs.MaRshaL gBhADEAOAA4ADMAZgA3ADEANgA 15,12,5,100,60,48,36,108,163,9,81,2 Iex(([RUntime.INTerOPSeRVICEs.marS IAZgBmADEAYQBhADkAMABiADIAM
16	ENCODING\6	**Partialy covered by the same Sigma** [sTRIng]::JoIn('', ('66z101J125!100J96 [sTrinG]::JoIn('', ([Char[]](100 ,67 , 91 [STriNg]::JOin('',('87G112V104I113A1
17	ENCODING\7	Example 1 Example 2 Example 3 Example 4

18	ENCODING\8	Example 1 Example 2 Example 3 Example 4

COMPRESS OBFUSCATION

Back to the Contents

Task #	Option	
19	COMPRESS\1	**Partialy covered by the same Sigma r** **function so they can maintain zero d** (neW-obJECT sYSTEm.io.CompReSSiO [sysTEm.COnVErT]::frOMBase64strInC) ,[sYsTEM.IO.compReSSiON.cOMPReS |ForEAcH{ $_.reADToEND()})| IEx Iex(new-oBJeCt sYStem.IO.CoMprESs '88wry89O1XWtKChKLS7OzM9T0PBI [io.CoMpREssiON.COmpresSionmODe InvOKE-ExPresSiOn (nEW-ObjeCt SyS [CONvERT]::frOMBASe64stRING('88w [SYSteM.iO.CoMPREssIoN.ComPressiC).rEADtOend() IEX (NEw-oBjEcT SYsTEM.io.streamrEa [coNvert]::FROmbase64sTRiNg('88wry [SystEm.Io.cOMpREsSiON.coMPReSSI

PS LAUNCHER OBFUSCATION

Back to the Contents

Task #	Option	

20	LAUNCHER\PS*	**LAUNCHER\PS\0 NO EXECUTION** poWeRsHEll "Invoke-Expression (N POwErShell "Invoke-Expression (Ne ---------- **LAUNCHER\PS\1 -NoExit** PowERsheLl -NOe "Invoke-Express poWerSHEll -NOEXIT "Invoke-Expr PoweRsheLl -Noexl "Invoke-Expres PowerSHEll -nOEX "Invoke-Express ---------- **LAUNCHER\PS\2 -NonInteractive** pOweRShELL -NONinte "Invoke-Ex powersheLL -noNiNtEraCTi "Invoke POwErSheLL -nONi "Invoke-Expres POWeRSHeLl -NONiNteR "Invoke- ---------- **LAUNCHER\PS\3 -NoLogo** POWeRShelL -Nol "Invoke-Express POWeRsHElL -noloGo "Invoke-Exp PoWeRSheLl -NOLO "Invoke-Expre ---------- **LAUNCHER\PS\4 -NoProfile** PoWerSHeLL -NOp "Invoke-Expres pOWeRSHeLl -NOpROFi "Invoke-E pOWErsHEll -nOpROfILE "Invoke-E PowErsHELL -NopROFil "Invoke-Ex ---------- **LAUNCHER\PS\5 -Command** POWERshElL -c "Invoke-Expression

		powerSHELL -CO "Invoke-Expressi PoWerShEll -cOMmAn "Invoke-Exp poWeRShElL -COMmANd "Invoke- ---------- **LAUNCHER\PS\6 -WindowStyle H** POWershEll -wINdOWs HIDden "In pOWERsheLL -wIn hIdd "Invoke-E powersHELL -wINd 1 "Invoke-Expr poWerShelL -WinDoW 1 "Invoke-E POwERsHELl -wINDowsTYl 1 "Invo poWeRshell -WIndOWStyL hI "Invo POwERshElL -Wi HiDdEN "Invoke- ---------- **LAUNCHER\PS\7 -ExecutionPolic** pOwerShelL -EXEcUt BYPasS "Invo PoWeRsheLL -Ep bypasS "Invoke-E pOwersHELl -EXec byPaSs "Invoke PoWeRshell -eXecUtIO ByPaSs "Inv poWErsHeLL -eX ByPass "Invoke-E ---------- **LAUNCHER\PS\8 -Wow64 (to pat** C:\WInDows\sySwoW64\wINDowS c:\WindoWs\SYsWOw64\WiNDOW c:\WINDOws\SYSwOw64\Windows

CMD LAUNCHER OBFUSCATION

Back to the Contents

Task #	Option	
21	LAUNCHER\CMD*	**Options LAUNCHER\CMD\0 - obfuscation methods for PS ke only hunt for CMD indicators:** cMD /c poWersHEll C:\wINDOWs\SYstEM32\CmD.E cMd.EXe /c PoweRSHell -nonin C:\winDOWs\sYstEM32\cmD.eX CMd.exE/c powERsHeLL -nOPR cMD/c pOWersHeLl -c C:\WiNDoWS\SysTEM32\cMD / cmd /c poWERSHeLL -Ep bYPAS CMd.exE/CC:\wiNdows\SySwOw

WMIC LAUNCHER OBFUSCATION

Back to the Contents 📄

Task #	Option	
22	LAUNCHER\WMIC*	**Options LAUNCHER\WMIC\0 obfuscation methods for PS k only hunt for WMIC indicator** WMIC "ProcESs" CaLL CREATE wMIC.exE 'PRoceSS' 'caLL' crEa c:\wINdoWS\sYstEM32\wbem\ wmic 'pRoCEss' "caLL" cReaTE ' WMIC PrOCESS "caLL" 'cReAte C:\windoWS\sysTEm32\wbem\

		c:\wINdOWS\systEm32\WbEM wMic.Exe "PrOCESS" CAlL crea wmIc.eXE "PRoCEss" "cALl" 'Cre

RUNDLL LAUNCHER OBFUSCATION

[Back to the Contents]#1009 (comment))

Task #	Option	
23	LAUNCHER\RUNDLL*	**Options LAUNCHER\RUND obfuscation methods for PS only hunt for RUNDLL indic** C:\wINDoWs\systEm32\RUn c:\WindowS\sysTEm32\RunD C:\windOwS\sySTEm32\rUN RunDLL32 SHELL32.DLL She c:\wIndoWs\SystEM32\Rund c:\WINdOwS\SySTem32\runl C:\wIndOWS\SySteM32\ruN rUNDLL32 SHELL32.DLL, ,Sh RUndLL32 SHELL32.DLL She

VAR+ LAUNCHER OBFUSCATION

Back to the Contents

Task #	Option	
24	LAUNCHER\VAR+*	**Options LAUNCHER\VAR+\0 - just apply different PS keys the 10), so in this task we should c**

```
cMD.exe /C "seT SlDb=Invoke-I
Net.WebClient).DownloadString
f 'eT-vaR','G','iab','lE' ) (\"{0}{1}\"
) ( ( ^&(\"{0}{1}\" -f'g','CI' ) (\"{0

c:\wiNdOWS\sYSteM32\CMD.e
(New-Object Net.WebClient).Do
sEt-Item (\"Var\" + \"IAblE:v\" +
f'ROnM','E','ENvi','nt' ) ) ;
${exEcuTIONCoNtEXT}.\"InVo`k
GCi ( \"VAR\" + \"iABlE:v\" +\"y
'lE','Ria','EnviROnMeN','GET','b','
{1}{2}{0}\" -f 's','Pr','Oces') )) )"

CMD.ExE/C"sEt iXH=Invoke-Exp
Net.WebClient).DownloadString
[TyPE]( \"{1}{0}{2}\"-F 'oN','enviR
{1}\" -f'aB','e','i','GETEN','viRon','l
f 'P','S','ROCES' )) )^| . ( \"{1}{0}\'

C:\winDoWs\SySTeM32\cmd.Ex
(New-Object Net.WebClient).Do
SET-iteM ( 'VAR' + 'i'+ 'A' + 'blE
'iRoN','mENT','e','nv') ) ;
${exECUtIONCOnTEXT}.\"IN`VC
GEt-VAriAble ( 'a' + 'o6I0') -vaL
f'e','gETenvIR','NtvaRIa','BL','ON
{1}\" -f'pRoC','esS') )) )"

C:\WIndoWs\systeM32\cMD /c
Object Net.WebClient).Downloa
${m`FLj`92} = [TYPE](\"{1}{2}{0}\
${mF`LJ`92}::(\"{4}{2}{3}{0}{1}\" -
).Invoke( ( \"{0}{1}\" -f 'qTHS','A'
{0}{1}{2}\" -f'Ke-','eXP','rEsSiOn',

c:\wiNDOws\systeM32\CmD.ex
Object Net.WebClient).Downloa
$RiJGI = [TyPe]( \"{0}{2}{1}\" -f 'I
{ExeCutIONConTeXT}.\"iNVo`ke
'INv','KEscri','o','Pt' ).Invoke( ( $r
f'tVarIAB','ge','Le','meN','tenvIrC
'cEs','s','PRO' ))) )"
```

		C:\wInDOWS\sYsTEm32\cMD.E (New-Object Net.WebClient).Dc hIDD (.(\"{0}{2}{1}\"-f 'v','E','aRi VaLU).\"inV`OKE`CoMMa`Nd\". 'OKES','INV','CRIpt').Invoke( ( ^ f'xyp','EnV:')).\"Va`luE\")" C:\wINdOWs\SyStem32\cMD / (New-Object Net.WebClient).Dc EXECuTIOnpoLIcY bypasS (.(\"{ f'e','X*XT') -VALuEoNly).\"inV`O f'ip','InVokeScR','T' ).Invoke( ( ^& CHIL','EM' ) ( \"{3}{1}{2}{0}\"-f 'R  cMd.eXE /C "Set prJ=Invoke-Ex Net.WebClient).DownloadString C:\WIndows\SYSWOW64\wINd ^&(\"{1}{0}\" -f 'x','ie' ) ( (.( \"{0}{ 'pr','J','ENV:')).\"v`ALuE\") "

STDIN+ LAUNCHER OBFUSCATION

Back to the Contents

Task #	Option	
25	LAUNCHER\STDIN+*	**Options LAUNCHER\STDIN-** **just apply different PS keys t** **so in this task we should onl** cmd /C"echo\Invoke-Expressi Net.WebClient).DownloadStri $EXECUTionCOnteXT.iNVoKE c:\windows\sYstEm32\CmD.e Net.WebClient).DownloadStri c:\wInDOws\SYstem32\CMd Net.WebClient).DownloadStri ([sTRiNg]$VERBosEPrEfErENcE c:\WiNDOws\sysTEm32\cmd. Net.WebClient).DownloadStri ${EXEcUtIONCONTeXT}.INvO

		CMd.eXe /c "eCHO/Invoke-E Net.WebClient).DownloadStri ${EXecUTiONCOnTEXT}.iNVO C:\wiNDoWS\sYSTEm32\cMd Net.WebClient).DownloadStri c:\wInDows\SYsteM32\CMd.E Net.WebClient).DownloadStri iTeM 'VariABLE:eX*Xt').ValuE.I c:\wiNDoWS\SySTem32\cmd Net.WebClient).DownloadStri $SHElLID[1]+$ShELlId[13]+'x cMD /C "ECHO\Invoke-Expre Net.WebClient).DownloadStri C:\wiNdOwS\SYswow64\WIn 'variabLE:EXECuTiONcontext')"

CLIP+ LAUNCHER OBFUSCATION

Back to the Contents

Task #	Option	
26	LAUNCHER\CLIP+*	**Options LAUNCHER\CLIP+\0 launcher just apply different P LAUNCHER\PS* (task 10), so CLIP+ indicators:** cmD /C "ECho\Invoke-Expressi Net.WebClient).DownloadString {1}{0}\"-f 'ype','-T','Add') -AN (f'C','ore'),'Pre',(\"{1}{0}\" -f 'n', [System.WIndOwS.CLiPBOARd).\"I`NvOKE\"( ) ) ^| ^& ( ( [StRI +'x'-JOIN'') ; [System.Windows f'Cl','ear').\"i`Nv`OkE\"( )" C:\WIndows\SystEm32\CMd /C Object Net.WebClient).Downloa

```
-st . ( \"{1}{0}{2}\"-f( \"{0}{1}\" -f
{3}\" -f 'tio','nCo',(\"{0}{1}\"-f 'Pr
${Sh`eL`lid}[13] + 'x' )( ([wiNDO
{1}\" -f 'get','tE'),'x','t').\"invO`Ke
{1}\"-f ( \"{1}{0}\" -f'e','etT'),'xt','

CmD /c " eCHO/Invoke-Expres
Net.WebClient).DownloadString
STa ${d`SCTG} = [Reflection.Ass
f'adWithP','a' ),( \"{1}{0}\" -f 'tia'
)).\"iNVo`ke\"( ( \"{5}{1}{2}{3}{4}{
) ; ${EXEcUtIONcontext}.\"i`N`V
( [sYSteM.winDoWs.FOrmS.ClIP
'xT','TE'),'GeT' ).\"I`Nvo`Ke\"( ) ) )
\"{1}{0}\" -f 'ear','Cl' ).\"IN`Voke\"

Cmd /c" echo/Invoke-Expressio
Net.WebClient).DownloadString
{1}{2}{0}\"-f'pe','Ad',(\"{1}{0}\" -f
{4}\" -f'ows','y','.F',(\"{0}{1}{2}\" -
),'S' ) ; ([SySTEM.wiNDows.FoRm
f'T','TTeX' ),'gE' ).\"invO`Ke\"( ) )
{0}\"-f'KE-','o' ),(\"{2}{1}{0}\"-f 'p
[System.Windows.Forms.Clipbo
),'xt').\"InV`oKe\"( ' ')"

CMD/c " ECho Invoke-Expressi
Net.WebClient).DownloadString
powershElL -noPRO -sTa ^& (\
),'A' ) -AssemblyN (\"{0}{3}{2}{1
f'e','ntatio'),'es','re' ) ; ^& ( ( [St
+ 'x'-JoiN'') ( ( [sySTem.WInDO
f'tTe','xt' ),'ge' ).\"IN`Vo`Ke\"( ) )
{1}{0}\" -f't',( \"{0}{1}\" -f 'tT','ex'

C:\WiNDOWS\SYSTem32\cMd
Object Net.WebClient).Downloa
C:\WINDOwS\System32\clIP.Ex
{1}{0}{2}\"-f 'p',(\"{1}{0}\" -f'Ty','A
{2}{0}\" -f'nC','Pr','esentatio' ) ) )
${eXeCUtIONConteXT}.\"InvOk
[WiNdoWs.ClIPBoARd]::( \"{0}{1
[Windows.Clipboard]::( \"{1}{0}\
```

		c:\wInDOws\SYStEm32\cmD.Ex Object Net.WebClient).Downlo WINDO Hid . (\"{2}{0}{1}\"-f (\ {1}{3}{0}\"-f'rms','.F','ows','o',(\" ${EXEcuTioncONtEXt}.\"iNvoKE [wIndOwS.ForMs.CLiPBOard]::( ).\"iNV`OkE\"( ) ) ) ; [Windows.F {0}{1}\"-f 'Se','tT' ),'xt' ).\"InVO`k cmD.exE /c " ECHo Invoke-Expr Net.WebClient).DownloadString exEcUTioNPoL BypAss ^&(\"{1) -Assem (\"{0}{2}{1}{3}\" -f 'Sys (\"{1}{0}\"-f 'rms','Fo')) ; (^& (','G'),(\"{1}{0}\"-f'rIa','va')) (\"{1)).\"v`AlUE\".\"In`VO`k`ecOMm/ [systeM.WiNdoWS.FormS.cliPb f'XT','ttE'),'GE').\"i`NvOke\"( ) ) ) \"{0}{1}\"-f'Cle','ar' ).\"I`N`VOKe  CMd.eXE /C "ECho/Invoke-Exp Net.WebClient).DownloadString C:\wINdowS\SYSwOW64\windo -StA ${Nu`ll} = [Reflection.Asse {1}\"-f 'Load','W'),'a','e','ith',(\"{ f'Part','i')).\"I`Nvo`ke\"((\"{2}{0} 'tem.Window','s','Sys','s','.Form' {0}{2}\" -f'x',(\"{0}{1}\" -f'GETt',' ${eNV`:c`o`MSPEc}[4,24,25]-Joi {0}{1}\"-f 'etT','ext','S' ).\"INVo`k

VAR++ LAUNCHER OBFUSCATION

Back to the Contents

Task #	Option	
27	LAUNCHER\VAR++*	**Options LAUNCHER\VAR++** **just apply different PS keys t** **so in this task we should only** C:\wINDOwS\SYStEM32\CmD Object Net.WebClient).Downlo

```
{1}{0}\"-f'ex','I' ) ( ( .(\"{1}{0}\" -
f'E','nv',':jXgL')).\"v`AluE\" ) &&

c:\WiNDOWS\sYSTEm32\CmD
(New-Object Net.WebClient).D
noeX ^ ^ ^&(\"{2}{0}{1}\"-f '-It
) ([TYpE]( \"{2}{3}{0}{1}\"-f 'e','N
[sTrIng]${VE`Rbo`SepReFER`Er
'RIAbLe:z8j' + 'u2' +'l' ) ).vALU
'IRo','Nm','GETE','ABlE','I','nv','e
{0}\"-f'cEss','P','RO') )) )&& c:\\

cMD /c "SeT xClr=Invoke-Exp
Net.WebClient).DownloadStrin
${L3`V`BF6} = [TypE]( \"{0}{2}{
${ExEcUtionCoNteXt}.\"i`NvOk
{1}{0}\" -f 'itEM','-ChIld','GeT' )
'V','GEtEn','riA','BLE','IronMen
f'eSs','PROc') )) )&& cMD /c %

C:\WINdOws\sYStEM32\cMD
Object Net.WebClient).Downlo
(\"{0}{1}{2}{3}\"-f 'g','Et','-VA','F
f'EXECUTiOnCOnT','t','eX' )).\"v
{0}\" -f'rIpt','keS','invO','c' ).Inv
{1}\"-f 'eNV:G','jQ' ) ).\"VAl`UE
%qBZO%"

C:\WIndOwS\sYStem32\Cmd.
Object Net.WebClient).Downlo
NOPROFiL Set-iTEM VArIAbLe
'eNVi','Nt','ronme' ) ); ( .( \"{2}{
'VaRIa','X*xT','ble',':E') ).\"V`ALu
f't','RIp','c','invoKes' ).Invoke( (
f'g','et','E','roN','iabLe','NVI','M
{0}{1}\"-f'pRo','cEss') ) ) )&& C:
/C%QexIO%"

C:\WINDoWs\SYsTeM32\Cmd
Object Net.WebClient).Downlo
^ ^ ^&( ${s`helL`iD}[1] + ${sh`
{2}{3}{0}\"-f 'V','E','n','v:lzxR' )).\
/C %yTW%"
```

		CMD.EXe /C "sEt cDpyq=Invc Net.WebClient).DownloadStrir hIDDEN (.(\"{0}{1}\" -f'C','HilDI).\"VA`LUe\" ^^^| ^^^&(${v f'INg','ToSTR').Invoke( )[1,3]+' cMD.ExE /C "SET BudG=Invok Net.WebClient).DownloadStrir bypasS ^^^& ('sV') (\"{1}{2} f'En','T','ViROnmeN')) ; (.(\"{ f'EXECUtiONC','Nt','o','eXt')). {0}\"-f 'ript','vOke','In','SC').Inv 'NmE','N','gEtEnv','Ir','tVArIAb' {0}\"-f'SS','PROCE'))))&& cM CMD /C"sET KUR=Invoke-Exp Net.WebClient).DownloadStrir MxI=C:\wINDowS\sYsWow64 ${ExEcut`IoN`cON`TExT}.\"invc 'pt','EscRi','INvOk' ).Invoke( ( .( f'ENV:kU','R')).\"vAl`Ue\")&&

STDIN++ LAUNCHER OBFUSCATION

Back to the Contents

Task #	Option	
28	LAUNCHER\STDIN++*	**Options LAUNCHER\STDIN launcher just apply differe LAUNCHER\PS* (task 10), STDIN++ indicators:** cmD /c "SEt nEp= Invoke-E Net.WebClient).DownloadSt vaRIAblE:*XeC*T).valuE.iNvC ([eNViROnMenT]::geTenvIR( )^|PowersHElL (VArIABle 'eX VAL).InVokeCoMmand.InvC C:\wiNdOWs\SystEm32\cM (New-Object Net.WebClient ${EXECutIoNcOnTExT}.inVok

```
([eNvirOnMEnT]::GETenVIrC
poweRSHelL -NoE - && C:\

CmD.ExE/c "SEt jqP= Invoke
Net.WebClient).DownloadSt
eXPreSsioN
([enviRONMent]::GEteNVIrC
POWerSHELl -NoNinTE $IN
$sheLlid[1]+$ShELlid[13]+'

cMd.EXE /C "SET RiJ= Invok
Net.WebClient).DownloadSt
${eXEcuTIONcOnTEXT}.iNVC
eNV:rIj).vaLUe ) ^|PoWeRsh
'VArIaBlE:ex*XT').vAlue.Invol
cMd.EXE /C%ktpfR%"

CmD.EXE /C "SeT khW=Inv
Net.WebClient).DownloadSt
${EXECuTIonCOntext}.inVO
EnV:khW).vaLuE ) ^|PoWER
$Env:cOmSPec[4,26,25]-jOi

c:\wiNDOwS\syStem32\CM
(New-Object Net.WebClient
ENv:XjIOW).valUE ^| power
'vARIaBle:eX*XT').vAlUE.iNv
c:\wiNDOwS\syStem32\CM

CMd/C "sEt Guz= Invoke-E
Net.WebClient).DownloadSt
exprESSiOn (iteM env:gUZ).
${ExecutioncOntexT}.invokE
CMd/C%Cpa%"

C:\wInDOWS\sYsTEM32\cM
Object Net.WebClient).Dow
vaRIABlE:E*oNTe*).VaLUe.iN
([eNVirONmENT]::GEtENVir
PowershelL -EXecu byPAsS
$eXecutiOnCONTeXT.invok
C:\wInDOWS\sYsTEM32\cM

C:\winDowS\SysteM32\Cm
Object Net.WebClient).Dow
```

		$eXECutionconTeXt.inVoKE(([ENVirOnment]::geTenVIrO C:\WiNDoWS\SYSwoW64\V ^^^&($PShOME[4]+$psH C:\winDowS\SysteM32\Cm

CLIP++ LAUNCHER OBFUSCATION

Back to the Contents

Task #	Option	
29	LAUNCHER\CLIP++*	**Options LAUNCHER\CLIP++** **same way as LAUNCHER\PS** C:\WINdoWS\sySteM32\CMd Net.WebClient).DownloadStri f'dd-',(\"{0}{1}\" -f 'T','ype'),'A \"{2}{1}{0}\" -f 'rms','Fo','s.'),'i', [sYSteM.wiNDoWS.forMs.CllP [System.Windows.Forms.Clipb C:\WInDows\System32\cMd |C:\wiNDOwS\SyStEm32\cLiP {2}\"-f 'Ad','d-T','ype') -A (\"{) ; ${EXEcUtIONcONtEXT}.\"IN {1}\"-f'GE',(\"{0}{1}\"-f 'TT','EXt f'le','ar')).\"iN`V`oKe\"( )" C:\wiNdowS\sySteM32\cmd / cIIp&&C:\wiNdowS\sySteM32 [System.Reflection.Assembly]: 'hPart','ia')).\"i`NvOke\"((\"{3}{ ${eX`Ec`UT`ioN`coNteXt}.\"I`N {0}\"-f'EXt',(\"{1}{0}\" -f 'T','gET 'tTe','Se'),'t').\"i`NvoKe\"(' ' )"  C:\WINDowS\sYsTEM32\CmD C:\WIndOWs\SYSteM32\CLip [System.Reflection.Assembly]: 'ial','N','ame'),'it','h' ).\"in`VO`K [wIndows.fOrms.cLIPBOArD]:

		{2}{1}{0}\"-f 'e',(\"{2}{1}{0}\"-f' jOin'') ; [Windows.Forms.Clipb C:\WINdOws\sYsTeM32\Cmc |CLIp&&C:\WINdOws\sYsTeM Assem (\"{1}{3}{0}{4}{2}\" -f'e f'rIab','L'),'va','e') (\"{1}{0}{4}{).\"va`lUe\".\"invok`E`cOmM`A {1}\"-f 'gEt','Te')).\"i`NVO`ke\" f'Se','tTex')).\"INvo`KE\"(' ')"  CmD/C "Echo/Invoke-Express &&CmD/C poweRshell -ST -c AssemblyNam ( \"{0}{3}{1}{2}\ ${exECUtioncONText}.\"iNVO \"{0}{1}\" -f'Ette','Xt' )).\"iN`V`C  cmd /C" eChO\Invoke-Expres -ST -WINdOwStY HiddeN ${U f'd','Loa' ),'l',( \"{0}{1}\"-f 'N','a 'ws.','Forms','y','st','Windo','S', ).\"inVO`kE\"( )) ^^^\|^^^& ( ).\"In`V`OKe\"( )[1,3]+'x'-JOIn).\"iN`VOke\"( )"  c:\WINdoWS\SYsteM32\cmd. \|C:\wInDows\sYSTEM32\CIIp. ST ^^^&(\"{0}{2}{1}\"-f ( \"{0} 're','nCo','entatio' ) ) ; ([WiNdC ^^^\| . ( ( [sTRING]${ve`RBosE {1}\"-f't','Text'),'e','S').\"In`VO CMd/C " ecHo Invoke-Expres C:\wiNdows\system32\CIIp.E -Sta . (\"{1}{0}{2}\" -f 'T',(\"{0} 'tem','s.F','.','Window'),'Sys','o [wiNDOWs.fOrmS.clIPbOARd [Windows.Forms.Clipboard]::(

RUNDLL++ LAUNCHER OBFUSCATION

Back to the Contents 📄

Task #	Option	
30	LAUNCHER\RUNDLL++*	**Options LAUNCHER\RU** **launcher just apply diffe** **(task 10), so in this task** c:\WiNdOws\sySTeM32\c Object Net.WebClient).Dc ShellExec_RunDLL "pOWE ^| . ('{1}{0}'-f'ex','i')" C:\wIndows\sysTEM32\cM Object Net.WebClient).Dc ,ShellExec_RunDLL "POW {3}'-F 'O','NVir','E','NmeN 'v','LE',':EXECu','IoNcOnTe {1}{3}'-f'I','KE','Nvo','sCRIp 'NvIrO','VA','getE','nMEnt f's','Proce','s'))))" c:\wInDOWS\SySTeM32\ Object Net.WebClient).Downloac SHELL32.DLL ShellExec_R [TypE]('{2}{0}{1}' -F'NMen f'pR','EsSio','n','ex','iNVokE).VAlUe::('{3}{5}{0}{4}{1}{6).Invoke('gSj',('{1}{0}{2}' - C:\winDoWS\sYStem32\C Net.WebClient).Downloac SHELL32.DLL,ShellExec_R [strinG]${VERBoSEPReFEF 'iTe','m','chILD') ('{1}{0}' - CmD.EXE /c "SEt igfM=In Net.WebClient).Downloac ShellExec_RunDLL "PoWE 'eM','GE','t-child','IT') ('{0 'x','ie')" C:\wINdoWs\sYsTEm32\C Object Net.WebClient).Dc

		ShellExec_RunDLL "pOwe f'ahI','EN','V:')).'ValUE' ^| . cmd /C "seT LFM=Invoke Net.WebClient).Downloac SHELL32.DLL ShellExec_R "$PGRV4H = [TyPe]('{3}{2 ${exeCUTIoNcONText}.'IN).Invoke(((gi variAbLE:p f'M','GEtEn','vA','t','ViRoN f'PROc','E','SS'))))" c:\WINDOws\SysTEm32\((New-Object Net.WebClie SHELL32.DLL,ShellExec_R "(^& ('{2}{1}{3}{0}'-f 'ItE)).'VAlUE'.'InVokeCommal ('{3}{0}{2}{1}'-f 't-','m','CH CMD.ExE /C "SeT vPu=In Net.WebClient).Downloac SHELL32.DLL,ShellExec_R "C:\WinDOWs\SYSwOw6 "(.('{1}{0}' -f'Ci','g') ('{0}{ ${eNV:cOMSPeC}[4,26,25

MSHTA++ LAUNCHER OBFUSCATION

Back to the Contents

Task #	Option	
31	LAUNCHER\MSHTA++*	**Options LAUNCHER\MSH** **LAUNCHER\PS* (task 10)** c:\winDowS\syStEM32\Cm Net.WebClient).DownloadS '{1}{0}'-f'I','GC') ('{0}{2}{1}' - CMD.exE/C "SeT Qsk=Invo VBScRIpT:CREATeObjECt(" 'Sk','ENV:Q')).'vAlue'^|^&

		C:\WinDOwS\SystEm32\cM VBScript:CReATEOBjeCt("W {0}{1}' -f 'P','t','OkescrI','iNv C:\WindOws\SySTeM32\cr Net.WebClient).DownloadS NoLoG (.('{1}{0}' -f 'ITem',' (WInDow.Close)" cMD/C "sET NkI=Invoke-E VBSCRIPT:CreaTEObjeCT(" 'pT','nvoKEs','cRI','I').Invoke C:\WinDOWs\sySTEm32\C Net.WebClient).DownloadS -COMma (.('{1}{0}' -f 'i','G).'NamE'[3,11,2]-JoIN'')",(9 c:\wiNDoWs\sYStEm32\cm VBSCripT:CreaTEObjeCT("V f'E','Nv:spv','K')).'VAlUe' ^| c:\WIndOws\SYStem32\CM VBScriPt:CREatEObJECT("V {1}' -f'vOkEScRi','Pt','in').In cMd /C "sET yAt=Invoke-E VBSCRiPT:CrEaTeOBjECT("V (.('gV') ('{0}{1}'-f'eX','*xT' f'env','AT',':y')).'vAlUE')",(1

1

yugoslavskiy mentioned this issue on Sep 14, 2020

[Rules Development Backlog] Develop Sigma rules for Invoke-Obfuscation #578 Closed

Dmweiner commented on Oct 4, 2020

For the sprint I'm planning on starting with 20 and seeing how I can continue on from there with my mediocre regex skills.

👍 1

zinint commented on Oct 6, 2020 Contributor Author •••

> For the sprint I'm planning on starting with 20 and seeing how I can continue on from there with my mediocre regex skills.

Thanks, great! Wating for your PR, great chance to improve your regex skills BTW (: they are pretty handy (:

👍 1

NikitaStormwind commented on Oct 8, 2020 • edited Contributor •••

If no one objects, I'll take 31 and 30
30 #1094 #1097 #1108
31 #1098 #1099 #1109

👍 2

NikitaStormwind commented on Oct 8, 2020 Contributor

@zinint Do you want the rule to work on a single regular expression as specified in point 5 "Start to develop your own regex that will cover all of the obfuscation examples of this particuar obfuscation method, e.g" ? Or you need several regular expressions for different patterns as shown in the examples:
rules/windows/process_creation/win_invoke_obfuscation_obfuscated_iex_commandline.yml
rules/windows/powershell/powershell_invoke_obfuscation_obfuscated_iex.yml
rules/windows/builtin/win_invoke_obfuscation_obfuscated_iex_services.yml

👍 2

zinint commented on Oct 8, 2020 • edited Contributor Author

@NikitaStormwind I think we need several regular expressions for different patterns, but I'm open for suggestions (:

👍 1

zinint commented on Oct 8, 2020 Contributor Author

> If no one objects, I'll take 31 and 30

No objects, of course, thanks for joining!

👍 1

NikitaStormwind commented on Oct 8, 2020 • edited Contributor

> @NikitaStormwind I think we need several regular expressions for different patterns, but I'm open for suggestions (:

@zinint | And one more question: Do you need to make several rules for the task ? For example: 1.Rule (4104,4103), 2.Rule (process create), or is one rule enough ?

👍 2

NikitaStormwind commented on Oct 8, 2020 Contributor • • •

> > > > @NikitaStormwind I think we need several regular expressions for different patterns, but I'm open for suggestions (:
>
> > > @zinint | And one more question: Do you need to make several rules for the task ? For example: 1.Rule (4104,4103), 2.Rule (process create), or is one rule enough ?
>
> > It depends, but I think they should be a Rule Collection
>
> Saw you PRs, you went with 2 rules, I think that's fine, maybe later we will somehow rearrange that, but for now, that's a nice way, thanks a lot for your time and contribution. I'll get back to you in PRs after I review the rules.

Ok, thanks. I'll take a couple more tasks tomorrow

👍 2

zinint commented on Oct 8, 2020 • edited Contributor Author • • •

> @NikitaStormwind I think we need several regular expressions for different patterns, but I'm open for suggestions (:
>
> @zinint | And one more question: Do you need to make several rules for the task ? For example: 1.Rule (4104,4103), 2.Rule (process create), or is one rule enough ?

Forgive me (: but I forgot about one of the latest updates to the Issue before the sprint, it's in the end:

One obfuscation method = 3 Sigma rules

Each Sigma rule for a specific PowerShell obfuscation method should be developed for process_creation log category, service creation events (windows system eid 7045, windows sysmon eid 6, windows security eid 4697) and powershell log source. You can follow the approach used for obfuscated IEX invocation rules — there are 3 rules that rely on the same set of regular expressions:

- rules/windows/process_creation/win_invoke_obfuscation_obfuscated_iex_commandline.yml
- rules/windows/powershell/powershell_invoke_obfuscation_obfuscated_iex.yml
- rules/windows/builtin/win_invoke_obfuscation_obfuscated_iex_services.yml

❤️ 1

zinint commented on Oct 8, 2020 • edited Contributor Author

> Ok, thanks. I'll take a couple more tasks tomorrow

Top work **@NikitaStormwind**, thanks a lot, will see you tomorrow!

👍 2

This was referenced on Oct 8, 2020

[OSCD] Detects Obfuscated Powershell via use Rundll32 in Scripts #30 (4104, 4103) #1094 Merged

[OSCD] Detects Obfuscated Powershell via use Rundll32 in Scripts #30 (process_creation) #1097 Merged

[OSCD] Detects Obfuscated Powershell via use MSHTA in Scripts #31 (4104, 4103) #1098 Merged

[OSCD] Detects Obfuscated Powershell via use MSHTA in Scripts #31 (process_creation) #1099 Merged

This was referenced on Oct 9, 2020

[OSCD] Detects Obfuscated Powershell via use Rundll32 in Scripts #30 (Services) #1108 Merged

[OSCD] Detects Obfuscated Powershell via use MSHTA in Scripts #31 (Services) #1109 Merged

NikitaStormwind commented on Oct 9, 2020 Contributor

> > @NikitaStormwind I think we need several regular expressions for different patterns, but I'm open for suggestions (:
>
> @zinint | And one more question: Do you need to make several rules for the task ? For example: 1.Rule (4104,4103), 2.Rule (process create), or is one rule enough ?

Forgive me (: but I forgot about one of the latest updates to the Issue before the sprint, it's in the end:

One obfuscation method = 3 Sigma rules

Each Sigma rule for a specific PowerShell obfuscation method should be developed for process_creation log category, service creation events (windows system eid 7045, windows sysmon eid 6, windows security eid 4697) and powershell log source. You can follow the approach used for obfuscated IEX

> invocation rules — there are 3 rules that rely on the same set of regular expressions:
>
> - rules/windows/process_creation/win_invoke_obfuscation_obfuscated_iex_commandline.yml
> - rules/windows/powershell/powershell_invoke_obfuscation_obfuscated_iex.yml
> - rules/windows/builtin/win_invoke_obfuscation_obfuscated_iex_services.yml

@zinint | I made 3 rules for one task. If the check is successful, I will continue to write other tasks using the same method.
30 #1094 #1097 #1108
31 #1098 #1099 #1109

👍 2 🚀 2

NikitaStormwind commented on Oct 9, 2020 • edited Contributor

I'll take tasks 28 and 29
29 #1112 #1113 #1114
28 #1142 #1143 #1144

👍 2

This was referenced on Oct 9, 2020

[OSCD] Detects Obfuscated Powershell via use Clip.exe in Scripts #29 (4104, 4103) #1112 Merged

[OSCD] Detects Obfuscated Powershell via use Clip.exe in Scripts #29 (process_creation) #1113 Merged

[OSCD] Detects Obfuscated Powershell via use Clip.exe in Scripts #29 (Services) #1114 Merged

[OSCD] Detects Obfuscated Powershell via Stdin in Scripts #28 (4104, 4103) #1142 Merged

[OSCD] Detects Obfuscated Powershell via Stdin in Scripts #28 (process_creation) #1143 Merged

[OSCD] Detects Obfuscated Powershell via Stdin in Scripts #28 (Services) #1144 Merged

zinint commented on Oct 12, 2020 • edited Contributor Author

I'll take 27 then for descending order (: gotta do something as well ((:

#1150 #1151 #1152

1

This was referenced on Oct 13, 2020

[OSCD] Detects Obfuscated Powershell via VAR++ Launcher #27 (4104, 4103) #1146 Closed

[OSCD] Detects Obfuscated Powershell via VAR++ Launcher #27 (4104, 4103) #1149 Closed

[OSCD] Detects Obfuscated Powershell via VAR++ Launcher #27 (4104, 4103) #1150 Merged

[OSCD] Detects Obfuscated Powershell via VAR++ Launcher #27 (Services) #1151 Merged

zinint mentioned this issue on Oct 13, 2020

[OSCD] Detects Obfuscated Powershell via VAR++ Launcher #27 (process_creation) #1152 Merged

OpalSec commented on Oct 13, 2020 • edited Contributor

I'm looking at task 26 - apologies if my subsequent PRs aren't done right, I haven't collaborated in Github before!

2

OpalSec mentioned this issue on Oct 14, 2020

[OSCD] Task #26: Detection for Invoke-Obfuscation CLIP+ Launcher (4104, 4103) #1175 Closed

OpalSec commented on Oct 15, 2020 Contributor

Looking at task 25

2

OpalSec mentioned this issue on Oct 15, 2020

[OSCD] Tasks 24, 25 & 26: Detection for Invoke-Obfuscation CLIP+, STDIN+ & VAR+ Launchers #1177 Merged

OpalSec commented on Oct 15, 2020 Contributor

Looking at task 24

2

yugoslavskiy commented on Oct 17, 2020 Contributor

> apologies if my subsequent PRs aren't done right, I haven't collaborated in Github before!

Hello @OpalSec! That's totally fine, no worries (: That's the whole point of the sprint — engage more people into collaboration on GitHub (: I think most of the participants are not fluent in GitHub, but they are doing their best, and we are here to help.

❤️ 1

zinint commented on Oct 18, 2020 • edited Contributor Author

Taking task 23 - #1223

zinint mentioned this issue on Oct 18, 2020

[OSCD] Detects Obfuscated Powershell via RUNDLL Launcher #23 (4104, 4103 + Services + process_creation) #1223 Merged

zinint commented on Oct 18, 2020 • edited Contributor Author

Taking task 22 - #1225

zinint mentioned this issue on Oct 18, 2020

[OSCD] Detects Obfuscated Powershell via WMIC Launcher #22 (4104, 4103 + Services + process_creation) #1225 Closed

zinint commented on Oct 18, 2020 • edited Contributor Author

Taking tasks 20 & 21

- [x] Due to the very high FP rate, I suggest skipping these tasks.

zinint commented on Oct 18, 2020 • edited Contributor Author

Taking task 19 - #1229

zinint mentioned this issue on Oct 18, 2020

[OSCD] Detects Obfuscated Powershell via COMPRESS OBFUSCATION #19 (4104, 4103 + Services + process_creation) #1229 Merged

zinint commented on Oct 18, 2020 • edited Contributor Author

Taking task 18 - #1230

zinint mentioned this issue on Oct 18, 2020

[OSCD] Detects Obfuscated Powershell via ENCODING OBFUSCATION\8 #18 (4104, 4103 + Services + process_creation) #1230 Closed

zinint commented on Oct 18, 2020 Contributor Author

Taking task 17

aw350m33d mentioned this issue on May 3, 2021

Update issues for obfuscations in the Sigma project oscd-initiative/oscd-task-management#8 Open

2 tasks

zinint changed the title ~~[OSCD Initiative] Invoke-Obfuscation~~ **Invoke-Obfuscation** on Sep 13, 2021

fukusuket mentioned this issue on Dec 6, 2022

refactor: remove unneeded escapes(in `|re` block) #3744 Merged

frack113 added the Rules label on Dec 19, 2022

frack113 added the Help Wanted label on Dec 27, 2022

frack113 mentioned this issue on Dec 27, 2022

PowerShell Token Obfuscation #3825 Merged

frack113 commented on Dec 27, 2022 • edited Member

Summary rules to do

task	PR
1	X
2	X
3	X
4	X
5	X
6	X
7	X
8	X
9	X
10	dead link

11	
12	
13	
14	
15	
16	
17	
20	
21	

frack113 commented on Dec 28, 2022 Member

Most action are detected even if get no alert on the encoding.
Need to complex regex to catch then all

frack113 closed this as completed on Dec 28, 2022