

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 oneliners.

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 (<a href="https://example.com/other-name="https://example.com/ot

- 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 (<u>36210e0d-5b19-485d-a087-c096088885f0</u>)
- CrackMapExec PowerShell Obfuscation (6f8b3439-a203-45dc-a88b-abf57ea15ccf)
- CrackMapExec Command Execution (058f4380-962d-40a5afce-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 (<u>obfuscated IEX</u> 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 <a href="Invoke-Obfuscation">Invoke-Obfuscation</a> framework, you can pick up some of the tasks in that table and develop Sigma rules for them. You will need to use <a href="regular expression value modifier">regular expression value modifier</a>, provided by Sigma converter (sigmac).

Here is an example of <u>Sigma rule</u> that utilizes a regular expression value modifier ( | re ):

```
title: Invoke-Obfuscation obfuscated IEX invocation 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 <u>Invoke-Obfuscation</u> framework. The description of the approach is following.

# Original code (before obfuscation)

```
# command example
Invoke-Expression (New-Object Net.WebClient).Downloader
# 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
```

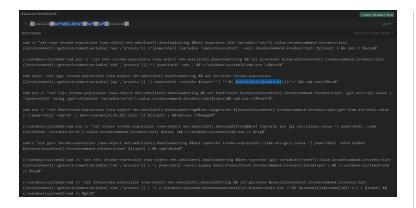
These regexes provide detection of the <u>IEX invocation obfuscation</u> function. This function is included into almost every encoding method so they can maintain zero dependencies and work on their own. That's why you'll see similar obfuscation results in different tasks, but it shouldn't distract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

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

- 1. Copy all obfuscated commands examples into <u>Sublime</u> or other text editor of your choice
- 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 <u>Sigma rule</u> <u>created by Daniel Bohannon</u> 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  $\sin (s*d{1,3}\s*,){2}$  covers only one example (5th):



- 4.5. Regex  $\mbox{\mbox{\mbox{$\ast$}}\name} \mbox{\mbox{\mbox{doesn't cover any examples.}}$
- 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 particular 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 <a href="regex101">regex101</a> 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\_obfu scated\_iex\_commandline.yml
- <u>rules/windows/powershell/powershell\_invoke\_obfuscation\_obfuscated\_iex.yml</u>
- 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

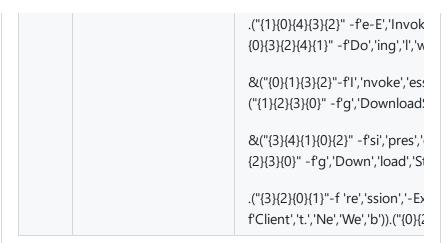
#### **TOKEN OBFUSCATION**

#### 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	
		TOKEN\COMMAND\1 IN`V`o`Ke-eXp`ResSIOn (Ne`\
		IN`V`OKE-exPRE`Ss`i`oN (n`e\
		IN'VOke-expr'eSS'ioN (NE'w-
	TOKEN\COMMAND\1	TOKEN\ARGUMENT\2 Invoke-Expression (New-Obje
1	TOKEN\ARGUMENT\2	Invoke-Expression (New-Obj
	TOKEN\MEMBER\2	Invoke-Expression (New-Obj
		TOKEN\MEMBER\2 Invoke-Expression (New-Obje
		Invoke-Expression (New-Obj
		Invoke-Expression (New-Obj
		&('In'+'voke-Expressi'+'o'+'n
2	TOKEN\COMMAND\2	.('Inv'+'oke-Ex'+'pr'+'ess'+'io
2		.('Invok'+'e-'+'Ex'+'pressio'+'
		&('Invok'+'e-'+'Expr'+'ession
		&("{3}{4}{2}{1}{0}{5}"-f'o','essi'
3	TOKEN\COMMAND\3	.("{0}{3}{2}{1}{4}" -f'l','-Ex','oke
3	1011214 (001111111111111111111111111111111111	.("{2}{3}{0}{1}"-f'o','n','Invoke-
		&("{2}{3}{0}{4}{1}"-f 'e','Expres
4	TOKEN\ARGUMENT\3	TOKEN\ARGUMENT\3 Invoke-Expression (New-Obj
	TOKEN\MEMBER\3	Invoke-Expression (New-Obj
		Invoke-Expression (New-Obj
		TOKEN\MEMBER\3

		Invoke-Expression (New-Obj
		Invoke-Expression (New-Obj
		Invoke-Expression (New-Obj
		TOKEN\ARGUMENT\4 Invoke-Expression (New-Obje
		Invoke-Expression (New-Obj
F	TOKEN\ARGUMENT\4	Invoke-Expression (New-Obje
5	TOKEN\MEMBER\4	TOKEN\MEMBER\4 Invoke-Expression (New-Obj
		Invoke-Expression (New-Obj
		Invoke-Expression (New-Obj
		\${En`V:`p`ATh}
6	TOKEN\VARIABLE\1	\${e`Nv:pATh}
		\${ENv:`path}
		Set-ItEM VaRIABLe:Lcx ( [TyP
7	TOKEN\TYPE\1	sV ("5Y"+"X") ( [typE]('SCrIpT
1		SET F9cg ( [tYpE]('scr'+'l'+'PT
		SET-Variable ('V'+'IR') ([TyPE]
		Set-itEM vaRiAbLE:YsB ( [tYPe senv:path")
8	TOKEN\TYPE\2	set-ITEm ('VAri'+'aBL'+'E'+':Y ('VARI'+'aBL'+'e'+':y'+'7w8O
		SEt-ItEM ('vAriAb'+'l'+'e:p87: ('VaRiab'+'L'+'E:P87Z2')).vaLl
		\$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



## **STRING OBFUSCATION**

Task #	Option	Results	Comments
10	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	These options can Concatenate entire command    Reorder entire command after concatenating    Reverse entire command after concatenating

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

# **ENCODING OBFUSCATION**

Task #	Option	
11	ENCODING\1	Partialy covered by the same Sigma  IEx([StrING]::JOin(", ( '34@32@36:40l-32P44z52T48u32@44T55_56u44_49-32T44u49R49_54R44T52T49u44~52-116@123~32z40T91k105T110~116  "\$( SET-ItEM 'vARiABLE:oFs' '')"+[STrIngle ( '73%110q118q111<107x101K45!6'-101V0Ke-ExPResSion ( -jOiN((73, 110,1)))
12	ENCODING\2	Partialy covered by the same Sigma  -joln ( '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-vAriAl
		[STRINg]::JOIN(", ( '111V156~166~1!
		Partialy covered by the same Sigma
		iNvOKE-EXPReSsiON ( ( (1001001 , 1 [COnveRT]::toinT16(([sTriNG]\$_ ) ,2 ) )
14	ENCODING\4	lex ([stRlng]::jOIN( " , ((1001001 , 110 2 )-as [CHaR]) }) ))
		( ( 1001001 ,1101110,1110110, 110 ) JoiN "  INvOKE-eXpRessiON
		IEX( -jOIN ('1001001C1101110M111 SPIIT'x'-SPlit 'M' -spLIt'C'-SPLiT'!'-spli
		Partialy covered by the same Sigma
		([rUnTImE.InteropSErvICes.mARShAL] DYANwA3ADQAMwBiAGYANwA1AG )) ) ieX
		([RuntimE.intEropseRvICes.MArsHAl]::
15	ENCODING\5	xAGEAMgAwADMANwAwAGYAYwAC SeCuRESTriNG -K (4514)))))   INvOkE
		( [rUNTiMe.intEROpSErvIcEs.MaRshaL
		gBhADEAOAA4ADMAZgA3ADEANg/
		15,12,5,100,60,48,36,108,163,9,81,20
		Iex(([RUntime.INTerOPSeRVICEs.marSIAZgBmADEAYQBhADkAMABiADIAN
		Partialy covered by the same Sigma
16	ENCODING) C	[sTRIng]::Joln(", ('66z101J125!100J96
10	ENCODING\6	[sTrinG]::JoIn( '', ([Char[]]( 100 ,67 , 91
		[STriNg]::JOin(",('87G112V104l113A1
17	ENCODING\7	Example 1
		Example 2

		Example 3
		Example 4
		Example 1
10	18 ENCODING\8	Example 2
18		Example 3
		Example 4

#### **COMPRESS OBFUSCATION**

Back to the Contents

Task #	Option	
19	COMPRESS\1	Partialy covered by the same Sigma I function so they can maintain zero do (neW-obJECT sYSTEm.io.CompReSSiO [sysTEm.COnVErT]::frOMBase64strIn()), [sYSTEM.IO.compReSSiON.cOMPReS [ForEAcH{ \$reADToEND()})] IEx  lex( new-obJeCt sYStem.IO.CoMpreSs '88wry89O1XWtKChKLS7OzM9T0PBI [io.CoMpRessiON.COmpresSionmOD InvOKE-ExPresSiOn (nEW-ObjeCt SyS [CONvERT]::frOMBASe64stRING('88w [SYSteM.iO.CoMPRessioN.ComPression).reADtOend()  IEX (NEw-object SYSTEM.io.streamrea [coNvert]::FROmbase64stRiNg('88wry [System.lo.cOMpResSioN.comPresSionsystem.lo.coMpResSioN.comPresSionsystem.lo.coMpResSioN.comPresSionsystem.lo.compResSionsystem.lo.compRe

#### PS LAUNCHER OBFUSCATION

Task #	Option	
20	LAUNCHER\PS\*	LAUNCHER\PS\0 NO EXECUTION poWeRsHEII "Invoke-Expression (N POwErShell "Invoke-Expression (Ne
		LAUNCHER\PS\1 -NoExit  PowERsheLI -NOe "Invoke-Express  poWerSHEII -NOEXIT "Invoke-Expres  PoweRsheLI -NoexI "Invoke-Express  PowerSHEII -nOEX "Invoke-Express  LAUNCHER\PS\2 -NonInteractive  pOweRShELL -NONinte "Invoke-Express  powersheLL -noNiNtEraCTi "Invoke-Express  POWERSHELL -NONi "Invoke-Express
		POWERSHELI -NONiNteR "Invoke- LAUNCHER\PS\3 -NoLogo POWERSHEIL -Nol "Invoke-Express  POWERSHEIL -noloGo "Invoke-Exp PoWERSHEIL -NOLO "Invoke-Expre  LAUNCHER\PS\4 -NoProfile PoWerSHELL -NOp "Invoke-Expres  pOWERSHELI -NOpROFi "Invoke-E  pOWErsHEII -nOpROfile "Invoke-E  PowErsHELL -NopROFil "Invoke-Expres

LAUNCHER\PS\5 -Command POWERshElL -c "Invoke-Expression powerSHELL -CO "Invoke-Expressi PoWerShEll -cOMmAn "Invoke-Exp poWeRShEIL -COMmANd "Invoke-LAUNCHER\PS\6 -WindowStyle I POWershEll -wINdOWs HIDden "Ir pOWERsheLL -wIn hIdd "Invoke-E powersHELL -wINd 1 "Invoke-Expr poWerShelL -WinDoW 1 "Invoke-E POWERSHELI -wINDowsTYI 1 "Invo poWeRshell -WIndOWStyL hl "Invo POWERshElL -Wi HiDdEN "Invoke-LAUNCHER\PS\7 -ExecutionPolic pOwerShelL -EXEcUt BYPasS "Invo PoWeRsheLL -Ep bypasS "Invoke-E pOwersHELI -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	
		Options LAUNCHER\CMD\0 - obfuscation methods for PS ke only hunt for CMD indicators:
		cMD /c poWersHEII
		C:\wINDOWs\SYstEM32\CmD.E
		cMd.EXe /c PoweRSHell -nonin
21	LAUNCHER\CMD\*	C:\winDOWs\sYstEM32\cmD.eX
		CMd.exE/c powERsHeLL -nOPR
		cMD/c pOWersHeLI -c
		C:\WiNDoWS\SysTEM32\cMD /
		cmd /c poWERSHeLL -Ep bYPAS
		CMd.exE/CC:\wiNdows\SySwOv

## WMIC LAUNCHER OBFUSCATION

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" CAIL crea
wmlc.eXE "PRoCEss" "cALI" 'Cre

#### **RUNDLL LAUNCHER OBFUSCATION**

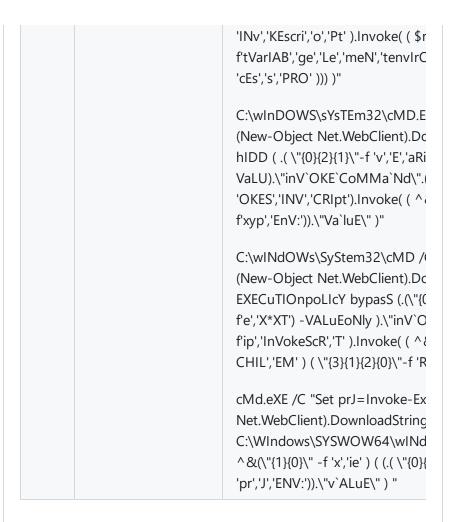
[Back to the Contents ] #1009 (comment))

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

## **VAR+ LAUNCHER OBFUSCATION**

|--|

# 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 SIDb=Invoke-I Net.WebClient).DownloadString f 'eT-vaR','G','iab','IE' ) (\"{0}{1}\" ) ( ( ^&(\"{0}{1}\" -f'g','CI' ) (\"{0] c:\wiNdOWS\sYSteM32\CMD.e (New-Object Net.WebClient).Dc sEt-Item (\"Var\" + \"IAblE:v\" + f'ROnM','E','ENvi','nt')); \${exEcuTIONCoNtEXT}.\"InVo`ki GCi ( \"VAR\" + \"iABIE:v\" +\"yı '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).Dc SET-iteM ( 'VAR' + 'i'+ 'A' + 'blE 'iRoN','mENT','e','nv') ); \${exECUtIONCOnTEXT}.\"IN`VC GEt-VAriAble ('a' + 'o610') -val f'e','gETenvIR','NtvaRla','BL','ON {1}\" -f'pRoC','esS') )) )" C:\WIndoWs\systeM32\cMD /c Object Net.WebClient).Downloa $f(m)^{5} = [TYPE](\"{1}{2}{0})$ \${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



#### STDIN+ LAUNCHER OBFUSCATION

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]\$VERBosEPrEfErENcl

c:\WiNDOws\sysTEm32\cmd. Net.WebClient).DownloadStri \${EXEcUtIONCONTeXT}.INvO CMd.eXe /c "eCHO/Invoke-Ex Net.WebClient).DownloadStri \${EXecUTiONCOnTEXT}.iNVO C:\wiNDoWS\sYSTEm32\cMd Net.WebClient).DownloadStri c:\wInDows\SYsteM32\CMd.I Net.WebClient).DownloadStri iTeM 'VariABLE:eX\*Xt').ValuE.I c:\wiNDoWS\SySTem32\cmd Net.WebClient).DownloadStri \$SHEILID[1]+\$ShELIId[13]+'x cMD /C "ECHO\Invoke-Expre Net.WebClient).DownloadStri C:\wiNdOwS\SYswow64\WInd 'variabLE:EXECuTiONcontext') )"

#### CLIP+ LAUNCHER OBFUSCATION

Task #	Option	
26	LAUNCHER\CLIP+\*	Options LAUNCHER\CLIP+\0 launcher just apply different F 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 /(
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.CIIF
'xT','TE'),'GeT').\"I`Nvo`Ke\"())
\"{1}{0}\" -f 'ear','CI').\"IN`Voke\

 $\label{eq:cmd/c} Cmd/c"\ echo/Invoke-Expressic Net.WebClient).DownloadString $$\{1\}_{2}_{0}''-f'pe','Ad',(''\{1\}_{0}''-f'\{4\}''-f'ows','y','.F',(''\{0\}_{1}_{2}''-),'S'); ([SySTEM.wiNDows.FoRnf'T','TTeX'),'gE').\"invO`Ke\"()) $$\{0\}''-f'KE-','o'),(\"\{2\}_{1}_{0}''-f'p[System.Windows.Forms.Clipbc),'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'); ^& (([Stl + 'x'-JoiN'') (([sySTem.WInDO'ftTe','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.E>
{1}{0}{2}\"-f'p',(\"{1}{0}\"-f'Ty','/

{2}{0}\" -f'nC','Pr','esentatio' ) ) ; \${eXeCUtIONConteXT}.\"InvOk [WiNdoWs.ClIPBoARd]::( \"{0}{1 [Windows.Clipboard]::( \"{1}{0}\

c:\wInDOws\SYStEm32\cmD.E>
Object Net.WebClient).Downloon
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

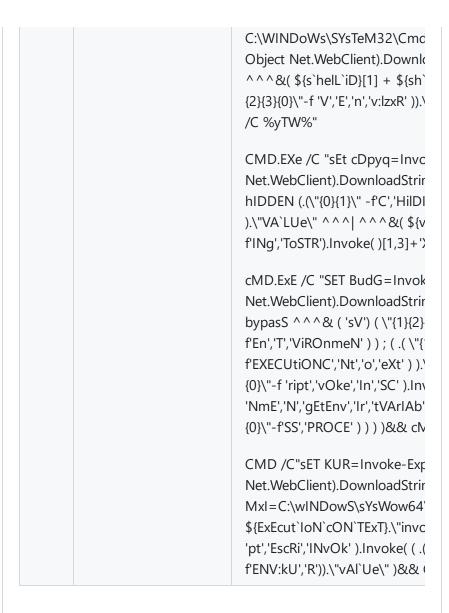
cmD.exE /c " ECHo Invoke-Expi Net.WebClient).DownloadString exEcUTioNPoL BypAss ^&( \"{1} ) -Assem ( \"{0}{2}{1}{3}\" -f 'Sy! (\"{1}{0}\"-f 'rms','Fo' ) ) ; (^& ( ','G'),( \"{1}{0}\"-f'rla','va')) ( \"{1] )).\"v`AlUE\".\"In`VO`k`ecOMm/ [systeM.WiNdoWS.FormS.cliPb f'XT','ttE'),'GE' ).\"i`NvOke\"( ) ) ; \"{0}{1}\"-f'Cle','ar' ).\"I`N`VOKe

CMd.eXE /C "ECho/Invoke-Exp Net.WebClient).DownloadString C:\wINdowS\SYSwOW64\window-StA \${Nu`II} = [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

|--|

27	LAUNCHER\VAR++\*	Options LAUNCHER\VAR++
	E/OTVETTER(V/IICT /	just apply different PS keys t so in this task we should only
		C:\wINDOwS\SYStEM32\CmE Object Net.WebClient).Downk {1}{0}\"-f'ex','I')((.(\"{1}{0}\"-f'E','nv',':jXgL')).\"v`AluE\")&&
		c:\WiNDOWS\sYSTEm32\Cm[ (New-Object Net.WebClient).[ noeX ^^&(\"{2}{0}{1}\"-f'-It ) ([TYpE]( \"{2}{3}{0}{1}\"-f'e','N [sTrIng]\${VE`Rbo`SepReFER`Er 'RIAbLe:z8j' + 'u2' +'I')).vALL 'IRo','Nm','GETE','ABIE','I','nv',' {0}\"-f'cEss','P','RO'))))&& c:\\
		cMD /c "SeT xClr=Invoke-Exp Net.WebClient).DownloadStrir \${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).Downk (\"{0}{1}{2}{3}\"-f 'g','Et','-VA','F f'EXECUTiOnCOnT','t','eX' )).\"' {0}\" -f'rlpt','keS','invO','c' ).Inv {1}\"-f 'eNV:G','jQ' ) ).\"VAI`UE' %qBZO%"
		C:\WIndOwS\sYStem32\Cmd. Object Net.WebClient).Downk NOPROFiL Set-iTEM VArIAbLe 'eNVi','Nt','ronme' ) ); ( .( \"{2}{ 'VaRIa','X*xT','ble',':E') ).\"V`ALL f't','RIp','c','invoKes' ).Invoke( ( f'g','et','E','roN','iabLe','NVI','M {0}{1}\"-f'pRo','cEss') ) ) )&& Colored



#### STDIN++ LAUNCHER OBFUSCATION

Task #	Option	
28	LAUNCHER\STDIN++\*	Options LAUNCHER\STDIN launcher just apply differer LAUNCHER\PS\* (task 10), STDIN++ indicators:
		cmD /c "SEt nEp= Invoke-E: Net.WebClient).DownloadSt vaRIAblE:*XeC*T).valuE.iNvC ([eNViROnMenT]::geTenvIR0

)^|PowersHEIL (VArIABle 'e> VAL).InVokeCoMmand.InvC

C:\wiNdOWs\SystEm32\cM (New-Object Net.WebClient \${EXECutIoNcOnTExT}.inVol ([eNvirOnMEnT]::GETenVIrC poweRSHelL -NoE - && C:\

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

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

CmD.EXE /C "SeT khW=Involution Net.WebClient).DownloadSt \${EXECuTlonCOntext}.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.iNva 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\cN Object Net.WebClient).Dow vaRIABIE: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

Task #	Option	
29	LAUNCHER\CLIP++\*	Options LAUNCHER\CLIP++ same way as LAUNCHER\PS'  C:\WINdoWS\sySteM32\CMc Net.WebClient).DownloadStri f'dd-',(\"{0}{1}\" -f 'T','ype'),'A \"{2}{1}{0}\" -f 'rms','Fo','s.'),'i', [sYSteM.wiNDoWS.forMs.CIIF [System.Windows.Forms.Clipk  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 / cllp&&C:\wiNdowS\syStEm32\cmd / cllp&\text{1}\tex

C:\WINDowS\sYsTEM32\CmE 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.Clipk

C:\WINdOws\sYsTeM32\Cmc |CLIp&&C:\WINdOws\sYsTeN Assem ( \"{1}{3}{0}{4}{2}\" -f'e f'rlab','L'),'va','e' ) ( \"{1}{0}{4}{: ).\"va`IUe\".\"invok`E`cOmM`/4 {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`(

cmd /C" eChO\Invoke-Expres -ST -WINdOwStY HiddeN \${L f'd','Loa' ),'l',( \"{0}{1}\"-f 'N','aı 'ws.','Forms','y','st','Windo','S', ).\"inVO`kE\"( )) ^^^|^^^& ( ).\"In`V`OKe\"( )[1,3]+'x'-JOIn ).\"iN`VOke\"( )"

c:\WINdoWS\SYsteM32\cmd. |C:\wInDows\sYSTEM32\Cllp. ST  $^^&(^"{0}{2}{1}\"-f(^"{0}'re','nCo','entatio')); ([WINd(^^^|.(([sTRING]${ve`RBosE}{1}\"-f't','Text'),'e','S').\"In`VO$ 

CMd/C " ecHo Invoke-Expres C:\wiNdows\system32\Cllp.E: -Sta . (\"{1}{0}{2}\" -f 'T',( \"{0} 'tem','s.F','.','Window' ),'Sys','o [wiNDOWs.fOrmS.cllPbOARd [Windows.Forms.Clipboard]::(

# **RUNDLL++ LAUNCHER OBFUSCATION**

Task #	Option	
30	LAUNCHER\RUNDLL++\*	Options LAUNCHER\RUI launcher just apply diffe (task 10), so in this task v
		c:\WiNdOws\sySTeM32\c Object Net.WebClient).Dc ShellExec_RunDLL "pOWE ^ .('{1}{0}'-f'ex','i')"
		C:\wIndows\sysTEM32\cl 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 ).VAIUe::( '{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'ahl','EN','V:')).'ValUE' ^|. cmd /C "seT LFM=Invoke Net.WebClient).Download SHELL32.DLL ShellExec\_R  $"$PGRV4H = [TyPe]( '{3}{2})$ \${exeCUTIoNcONText}.'IN ).Invoke( ( ( gi variAbLE:po 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 'ItEN )).'VAlUE'.'InVokeCommal ('{3}{0}{2}{1}'-f 't-','m','CH CMD.ExE /C "SeT vPu=In Net.WebClient).Download SHELL32.DLL,ShellExec\_R "C:\WinDOWs\SYSwOw6 "( .( '{1}{0}' -f'Ci','g' ) ( '{0}{ \${eNV:cOMSPeC}[4,26,25

#### MSHTA++ LAUNCHER OBFUSCATION

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\c\ VBScript:CReATEOBjeCt("V {0}{1}' -f 'P','t','Okescrl','iNv C:\WindOws\SySTeM32\cr Net.WebClient).DownloadS NoLoG (.('{1}{0}' -f 'ITem',' (WInDow.Close)" cMD/C "sET Nkl=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' )).'VAIUe' ^| c:\WIndOws\SYStem32\CN VBScriPt:CREatEObJECT("V {1}' -f'vOkEScRi','Pt','in' ).In cMd /C "sET yAt=Invoke-E VBSCRiPT:CrEaTeOBjECT("\ ( .('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





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.





**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 (:



edited -



NikitaStormwind commented on Oct 8, 2020 •

Contributor

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

30 #1094 #1097 #1108

31 #1098 #1099 #1109



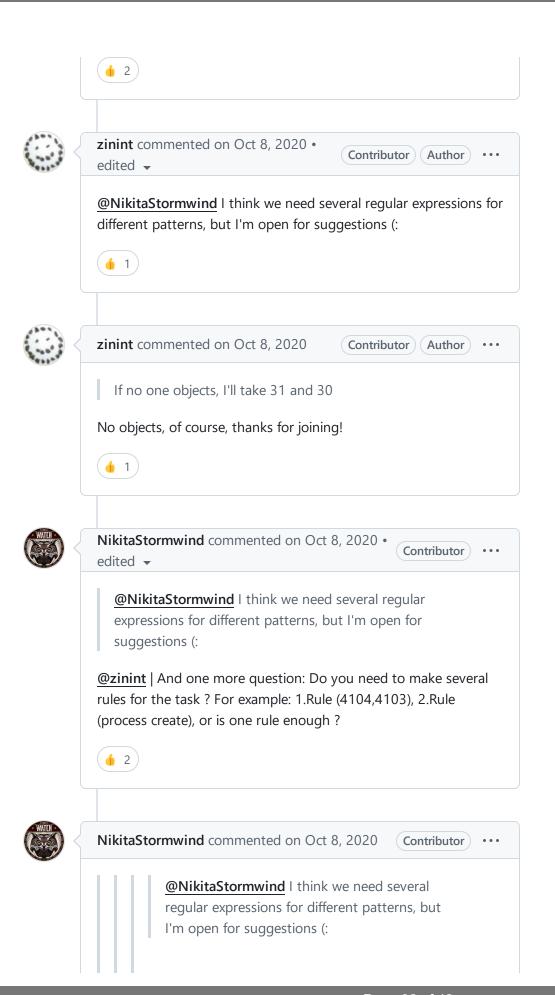


NikitaStormwind commented on Oct 8, 2020 (Contributor)

@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\_obfuscat ed\_iex\_commandline.yml

rules/windows/powershell/powershell\_invoke\_obfuscation\_obfusca

rules/windows/builtin/win\_invoke\_obfuscation\_obfuscated\_iex\_serv ices.yml



@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





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

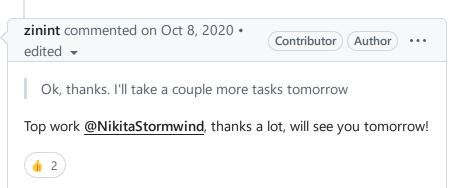
Each Sigma rule for a specific PowerShell obfuscation method should be developed for process\_creation log category, service creation events (windows system eid 7045, windows sysmon eid 6, windows security eid 4697) and powershell log source. You can follow the approach used for obfuscated IEX invocation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

rules/windows/process\_creation/win\_invoke\_obfuscation\_obfu scated\_iex\_commandline.yml

- rules/windows/powershell/powershell invoke obfuscation obf uscated\_iex.yml
- rules/windows/builtin/win\_invoke\_obfuscation\_obfuscated\_iex \_services.yml









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

**№** Merged

(process\_creation) #1099

This was referenced on Oct 9, 2020

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

**№** Merged

#1108

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

**№** Merged



NikitaStormwind commented on Oct 9, 2020

#1109

Contributor

ibutor

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

<u>@zinint</u> | 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

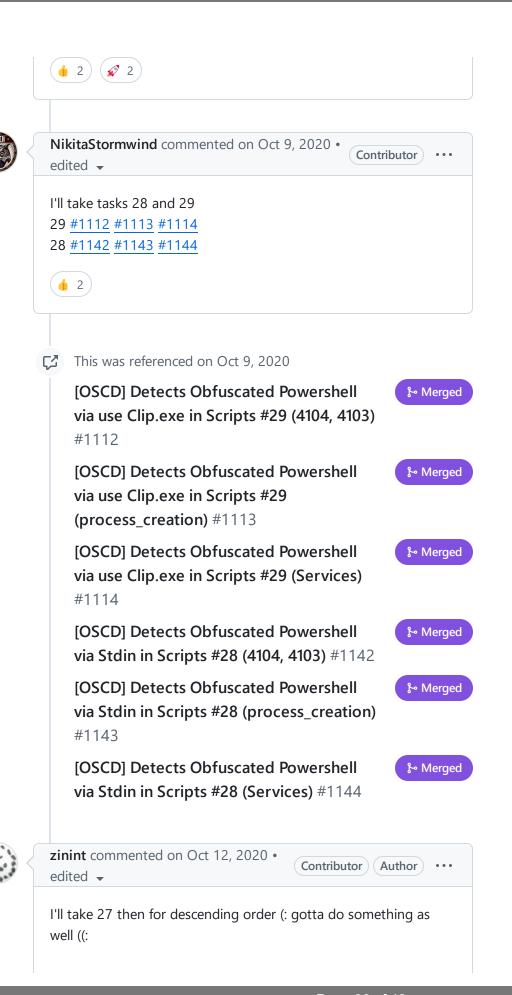
Each Sigma rule for a specific PowerShell obfuscation method should be developed for process\_creation log category, service creation events (windows system eid 7045, windows sysmon eid 6, windows security eid 4697) and powershell log source. You can follow the approach used for obfuscated IEX invocation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

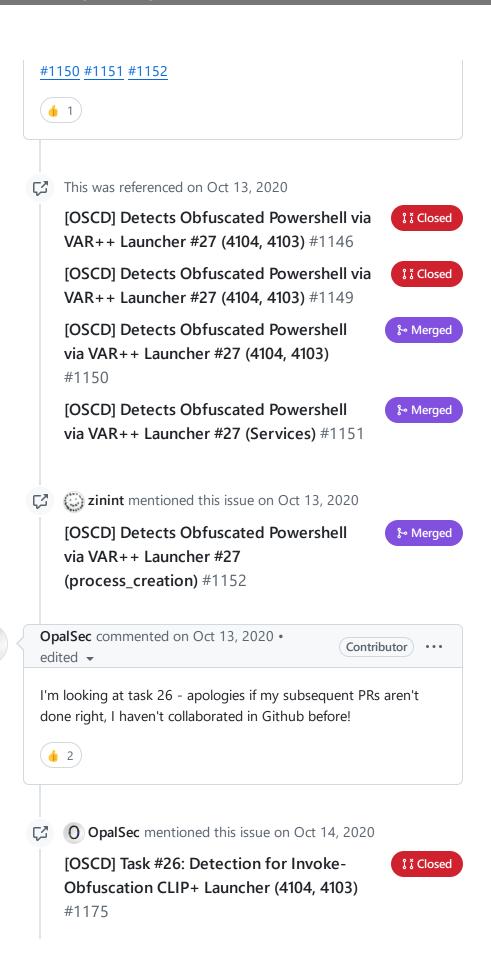
- rules/windows/process\_creation/win\_invoke\_obfuscation \_obfuscated\_iex\_commandline.yml
- rules/windows/powershell/powershell\_invoke\_obfuscatio
   n\_obfuscated\_iex.yml
- rules/windows/builtin/win\_invoke\_obfuscation\_obfuscate
   d\_iex\_services.yml

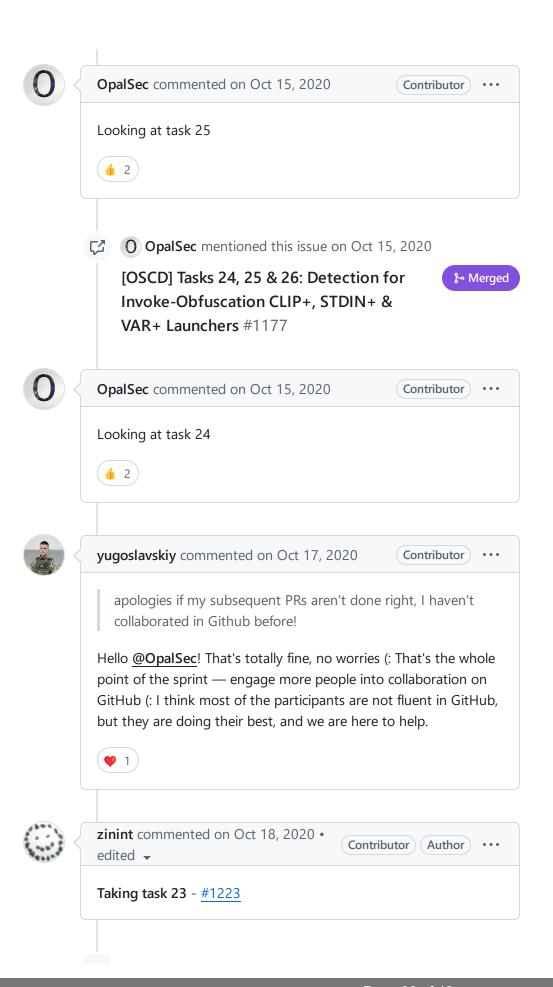
<u>@zinint</u> | I made 3 rules for one task. If the check is successful, I will continue to write other tasks using the same method.

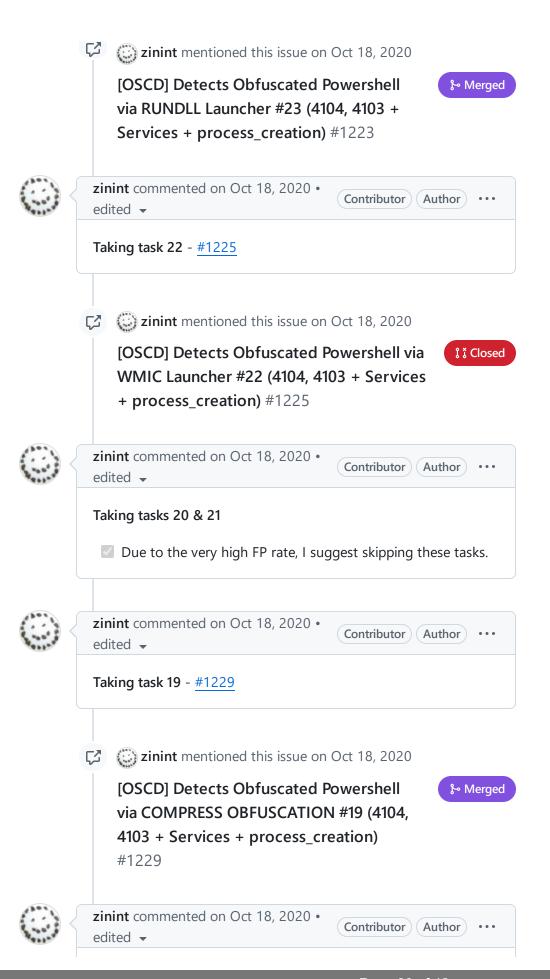
30 #1094 #1097 #1108

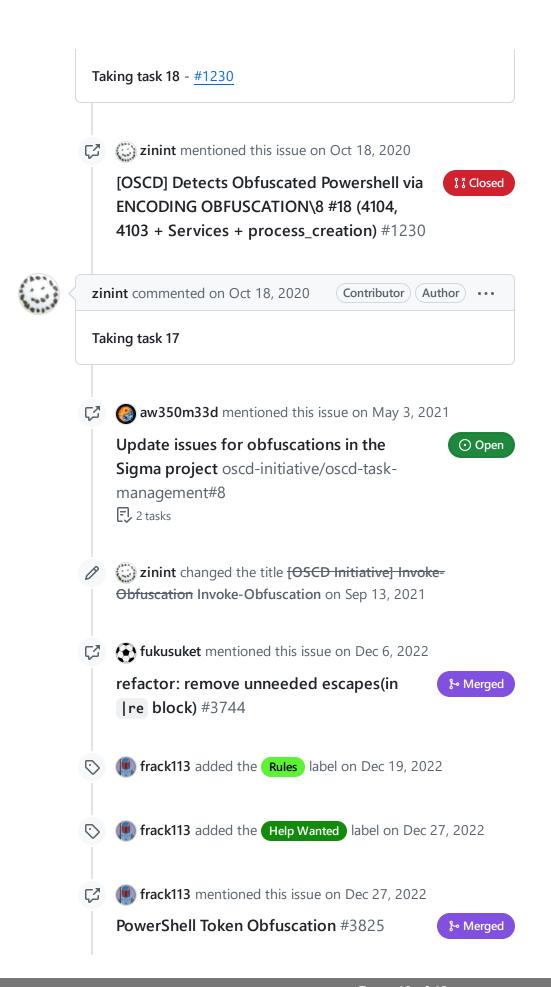
31 #1098 #1099 #1109













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	
1 7	
20	



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





ntering frack113 closed this as completed on Dec 28, 2022

Sign up for free

to join this conversation on GitHub. Already have an

account? Sign in to comment

Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information



© 2024 GitHub, Inc.