



Homeland
Security

NCCIC

National Cybersecurity and
Communications Integration Center

WMI for Detection and Response

August 2016



Homeland
Security

SUMMARY

This document provides an introduction to and guidance on methods available for mitigating the adversarial use of Windows Management Instrumentation (WMI).

CONTENTS

SUMMARY	iii
ACRONYMS	vii
1. INTRODUCTION	1
1.1 High-Level WMI Architecture.....	1
1.2 Remote WMI.....	2
2. Querying WMI	2
3. Attacker WMI Detection	3
3.1 Existing Detection Utilities	3
3.1.1 Sysinternals Autoruns	3
3.1.2 Kansa.....	4
4. Defensive WMI Eventing.....	5
4.1 The Two Classes of WMI Events	5
4.2 Three Important Parts of a WMI Event.....	5
4.3 Event Filters	6
4.4 Event Consumers	6
4.5 Filter to Consumer Binding.....	7
5. Event Types.....	8
5.1 Intrinsic Events	8
5.2 Extrinsic Events	9
6. Using WMI CLI commands for Detection and Removal of malicious wmi.....	10
6.1 Manual Detection: WMI Command Line Tool.....	10
6.2 Manual Removal: WMI Command Line Tool.....	10
6.3 WMI Intrusion Detection Using PowerShell Code Ex	10
6.4 Event Logs	11
7. Additional WMI Mitigation	11
7.1 WMI Backup & Restore	11
7.2 WMI Access Control	12
Appendix A.....	14

FIGURES

Figure 1. WMI Architecture Diagram.....	1
Figure 2. Sysinternals Autoruns screenshot with startup items.	4
Figure 3. Event Filters shown with wmicmgmt.msc.	6
Figure 4. Event Consumer Listing.	7

Figure 5. Showing how Filters and Consumers interact with one another by using a FilterToConsumerBinding to tie them together.....	8
Figure 6. How to start wmicmgmt tool.....	12
Figure 7. wmicmgmt.msc main menu	12
Figure 8. wmicmgmt backup/restore window.	12
Figure 9. Accessing Root directory to apply access control (security) settings.....	13
Figure 10. Screenshot of Security for Root.....	13
Figure A-1. After selecting namespace, class, and method script will be generated for user which can be further modified. Users then need only supply desired value(s) for script if required.....	14
Figure A-2. Powershell prompt and table of popular WMI cmdlets.....	15
Figure A-3. Administrators and malicious attackers can use the Wmic.exe terminal to execute WMI methods.	16
Figure A-4. Wbemtest.exe dashboard once connected.	17
Figure A-5. Contents of root\CIMV2 directory and WMI Explorer generated WQL language to select all contents from Win32_Processor object.....	17
Figure A-6. Exploring down 5 levels into the root\CIMV2 directory and displaying the contents of selected object	18

ACRONYMS

CLI	command line interface
IDS	intrusion detection system
VM	virtual machine
WMI	Windows Management Instrumentation
WQL	WMI Query Language

WMI for Detection and Response

1. INTRODUCTION

Windows Management Instrumentation (WMI) is composed of a powerful set of tools used to manage Windows systems both locally and remotely. While it has been well known and used heavily by system administrators since its inception, WMI has been gaining popularity amongst attackers for its ability to perform system reconnaissance, antivirus and virtual machine (VM) detection, code execution, lateral movement, persistence, and data theft.

This whitepaper will present an introduction to WMI, actual and proof-of-concept attacks using WMI, how WMI can be used as a rudimentary intrusion detection system (IDS), how to defend against adversarial use of WMI, and present how to perform forensics on the WMI repository file format.

1.1 High-Level WMI Architecture

WMI represents most data related to operating system information and actions in the form of objects. An object is a member of a class, a class is a member of a namespace, and all namespaces derive from the “Root” namespace. This paper will later show examples of how to list, find, and use namespaces, classes, and objects through multiple tools and methods such as PowerShell, WQL (WMI Query Language), WMI Code Creator and others. See Figure 1.

WMI classes can be found on the MSDN site at:

[https://msdn.microsoft.com/en-us/library/aa394554\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/aa394554(v=vs.85).aspx)

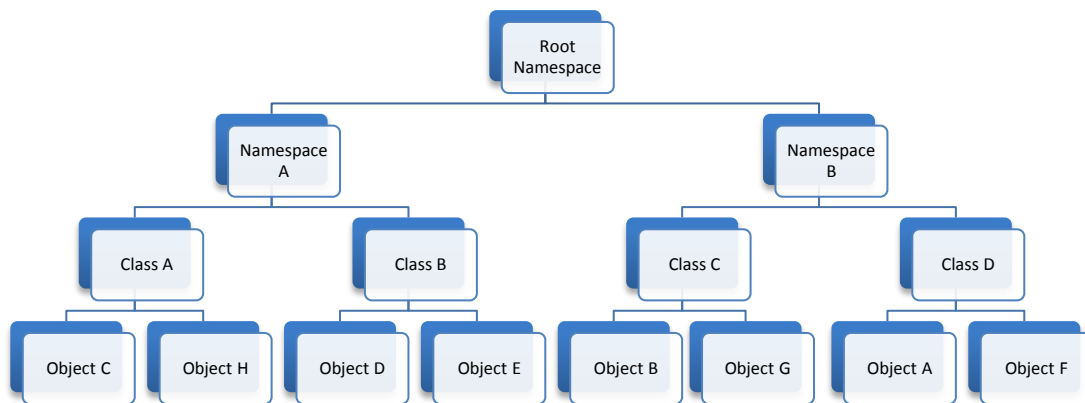


Figure 1: WMI Architecture Diagram.

1.2 Remote WMI

The real threat and power of WMI is realized when used over the network.

Currently, two protocols exist that enable remote object queries, event registration, WMI class method execution, and class creation:

- DCOM TCP Port 135
- WinRM TCP Ports 5985 (HTTP) and 5986 (HTTPS).

These protocols are viewed as advantageous to an attacker because most organizations and vendors generally don't inspect the content of this traffic for signs of malicious activity.

All that an attacker needs to leverage remote WMI is valid, privileged user credentials. In the case of the Linux `wmis-pth` utility, all that is needed is the hash of the victim user.

2. QUERYING WMI

WMI provides a straightforward syntax for querying WMI object instances, classes, and namespaces—Windows Query Language (WQL). From a defensive perspective, it is vital to understand and be able to adequately use queries. Queries are used regularly for malicious purposes and should be used for defensive purposes. These queries can do everything from attacker reconnaissance to intrusion detection. The three categories of WQL queries are as follows:

1. Instance queries

- a. Are used to query WMI class instances.
- b. Primarily are used by attackers for conducting reconnaissance and gathering information about a targeted system.

Format:

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

Example:

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

2. Event queries

- a. Are used as a WMI event registration mechanism, e.g., WMI object creation, deletion, or modification.

- b. Will be one area of focus later in the paper when discussing WMI defense and mitigation.

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'`

3. Meta queries

- a. Are used to query WMI class schemas.

Format:

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

The following query lists all WMI classes that start with the string “Win32.”

Example:

- `SELECT * FROM Meta_Class WHERE __Class LIKE "Win32%"`

NOTE: *When performing any WMI query, the default namespace ROOT\CIMV2 is implied unless explicitly provided.*

3. ATTACKER WMI DETECTION

3.1 Existing Detection Utilities

In addition to using WMI events to alert users to possible attacks, detection utilities are also available.

3.1.1 Sysinternals Autoruns

Autoruns is a free utility that unveils every startup item on a Windows-based PC. All images are stored in the startup folders, the Registry, and other areas.

Autoruns shows the name and location of each image. For files, it displays the directory path; for Registry entries, it provides the exact key. Autoruns also supplies the name of the publisher and a brief description based on the item’s version data. Double-clicking on an entry guides the user to its directory or Registry key; right-clicking opens a popup menu with more options,

including a Properties command that displays the standard File Properties window with full version information.

Users can check on the digital signature of an entry through the Verify command, which queries web sites with certificate revocation lists (CRLs) to determine if an image is digitally signed and whether the signature is valid.

Another option to “Hide Signed Microsoft Entries” excludes entries already signed by Microsoft, allowing the user to focus on third-party images.

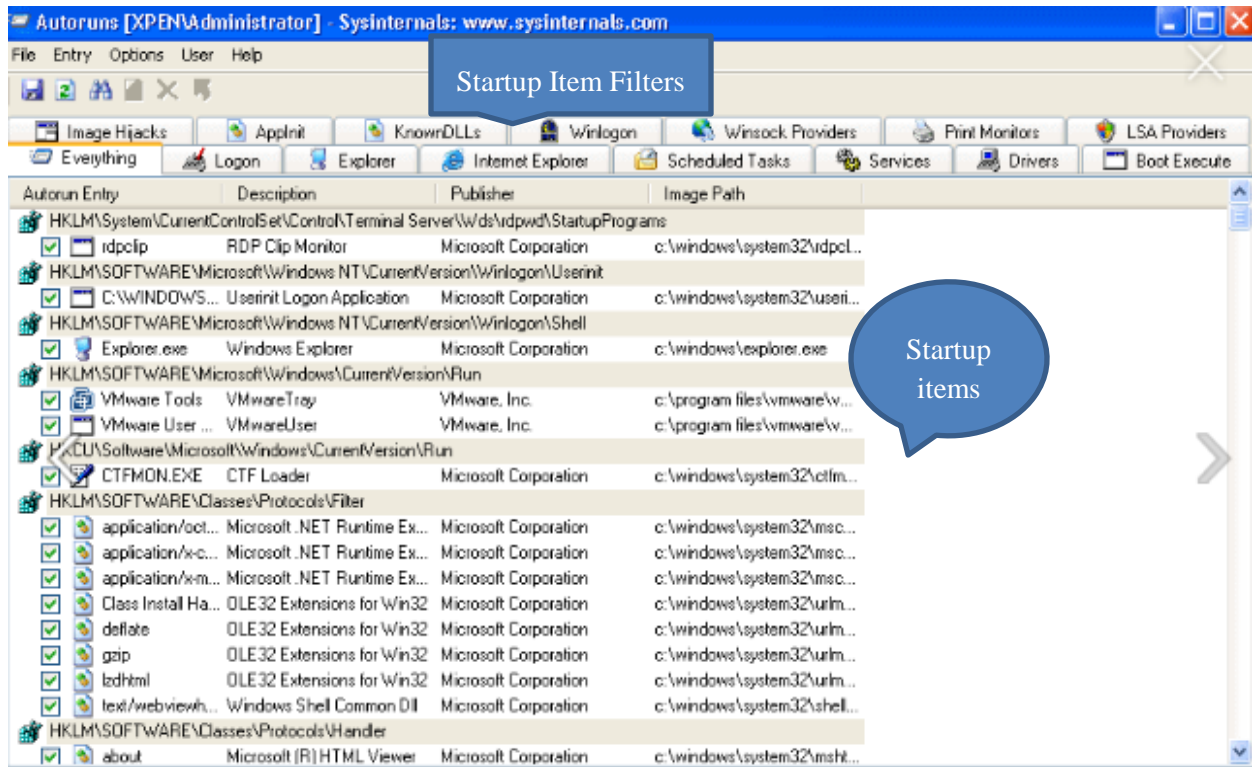


Figure 2: Sysinternals Autoruns screenshot with startup items.

3.1.2 Kansa

Kansa is modular. It features a core script, dozens of collector modules and analysis scripts to help make sense of the data collected. *Kansa* takes advantage of Windows Remote Management and PowerShell remoting by using PowerShell’s default nondelegated Kerberos network logons, not *CredSSP* and, therefore, does not expose credentials to harvesting.

Kansa is a great tool with many uses but particularly useful are its *Get-Autoruncs*, *Get-WMIEvtConsumer*, *Get-WMIEvtFilter* and *Get-WMIFltConBind* PowerShell scripts.

1. Get-Autoruncs

- A great utility for gathering data from many known ASEP (Auto Start Extension Point) locations, including the path to the executable or script, command line arguments, and cryptographic hashes such as MD5.

2. Get-WMIEvt(Consumer/Filter)/Get-WMIFltConBind

- Collects data about WMI Event Consumers/Filters/Consumer – Filter Binding.
- Kansa provides modules that can query and return information that an admin would need to detect WMI persistence.
- A walk-through on setting up Kansa and an in depth explanation of its many utilities and capabilities can be found at:
<http://www.powershellmagazine.com/2014/07/18/kansa-a-powershell-based-incident-response-framework/>

The downside to these tools is that they only detect WMI persistence artifacts at a certain snapshot in time. This means that tools like Sys Internals Autoruns and Kansa won't detect persistence from clever attackers who clean up their code once they've performed their actions. *The solution to this problem is to use WMI eventing.*

4. DEFENSIVE WMI EVENTING

The eventing subsystem present in WMI could be thought of as the free host-based IDS from Microsoft.

Because almost all operating system actions fire a WMI event, such as Instance, Class, Namespace, and Registry Creation and Modification events, WMI is well positioned to catch and alert admins of attacker actions as they occur.

Administrators can choose how to receive alerts on events they have created. One popular method is to have a user send an email or an alert pop up to notify the admin when an event fires.

One of the most powerful features of WMI from an attacker's or defender's perspective is the WMI event, which can be used to respond to nearly any operating system event.

- For example, a WMI event may be used to trigger an event upon process creation. This could then be used as a means to perform command-line auditing on any Windows OS.

4.1 The Two Classes of WMI Events

1. Events that run locally in the context of a single process.
 - Local events last for the lifetime of the host process.
2. Permanent WMI event subscriptions.
 - Permanent WMI events are stored in the WMI repository, run as SYSTEM, and persist across reboots.

4.2 Three Important Parts of a WMI Event

In order to install a permanent WMI event subscription, three things are required:

1. An event filter—The event of interest,
2. An event consumer—An action to perform upon triggering an event,

3. A filter to consumer binding—The registration mechanism that binds a filter to a consumer.

4.3 Event Filters

Once a filter has been configured, it can be used to receive alerts when new events are created.

Event filters are stored in an instance of ROOT\Subscription: __EventFilter object

As an example, event filters might be used to describe some of the following events:

- Creation of a process with a certain name;
- Loading of a DLL into a process,
- Creation of an event log with a specific ID;
- Insertion of removable media;
- User logoff; and
- Creation, modification, or deletion of any file or directory.

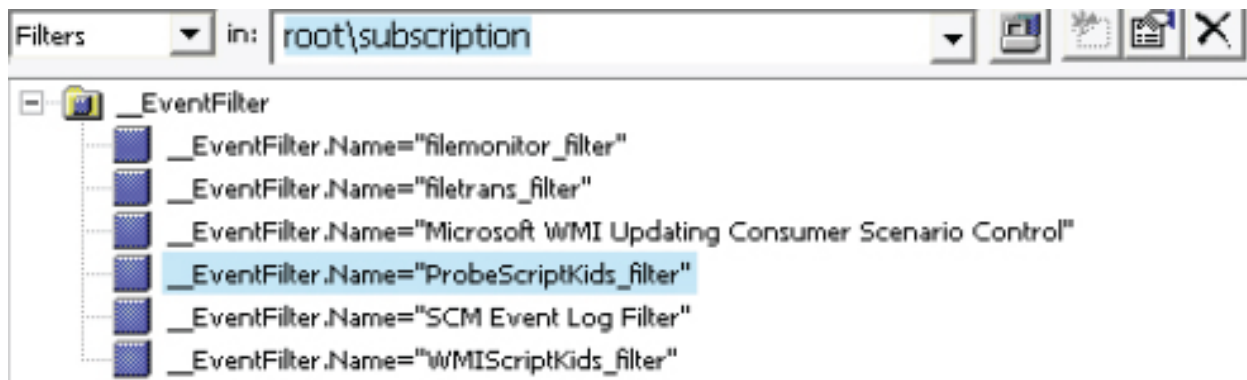


Figure 3: Event Filters shown with wmimgmt.msc.

4.4 Event Consumers

An event consumer is a class that is derived from the __EventConsumer system class that represents the action to take on firing an event.

The following useful standard event consumer classes:

- LogFileEventConsumer—Writes event data to a specified log file.
- ActiveScriptEventConsumer—Executes an embedded VBScript or JScript script payload.
- NTEventLogEventConsumer—Creates an event log entry containing the event data.
- SMTPEventConsumer—Sends an email containing the event data.
- CommandLineEventConsumer—Executes a command-line program.

`__EventConsumer` is an abstract base class used in registering a permanent event consumer

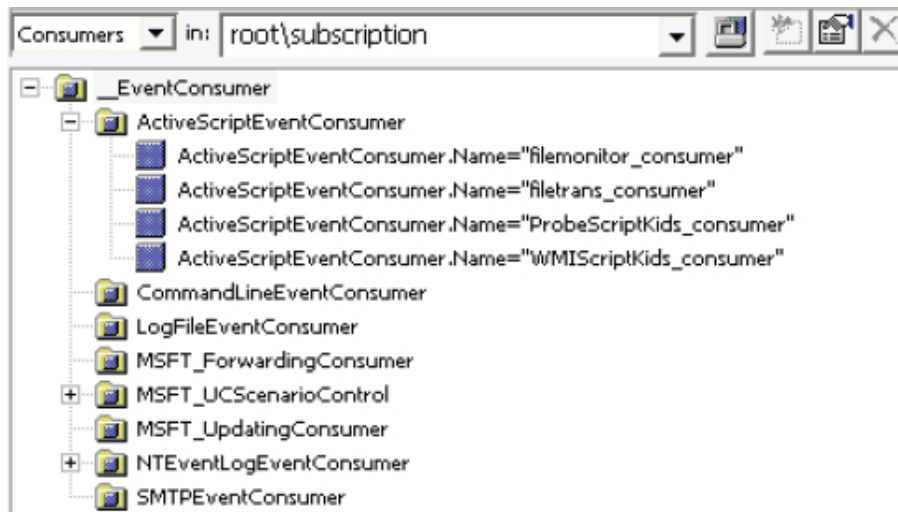


Figure 4: Event consumer Listing.

Attackers make heavy use of the `ActiveScriptEventConsumer` and `CommandLineEventConsumer` classes when responding to their events.

Both event consumers offer a tremendous amount of flexibility for attackers to execute any payload they want—all without needing to drop a single malicious executable or script to disk.

4.5 Filter to Consumer Binding

Once an event filter and an event consumer have been created, the only thing left to do is to bind them together so the consumer knows off of which filter to base itself.

This class instance associates an `__EventFilter` instance with an `__EventConsumer` instance. It completes the cycle by relating the class instances with each other. It answers the question, “With what Windows event (`__EventFilter`) will I execute my script program (`__EventConsumer`)?”

`__FilterToConsumerBinding` is used in registering permanent event consumers to relate an `__EventConsumer` instance to an `__EventFilter` instance.

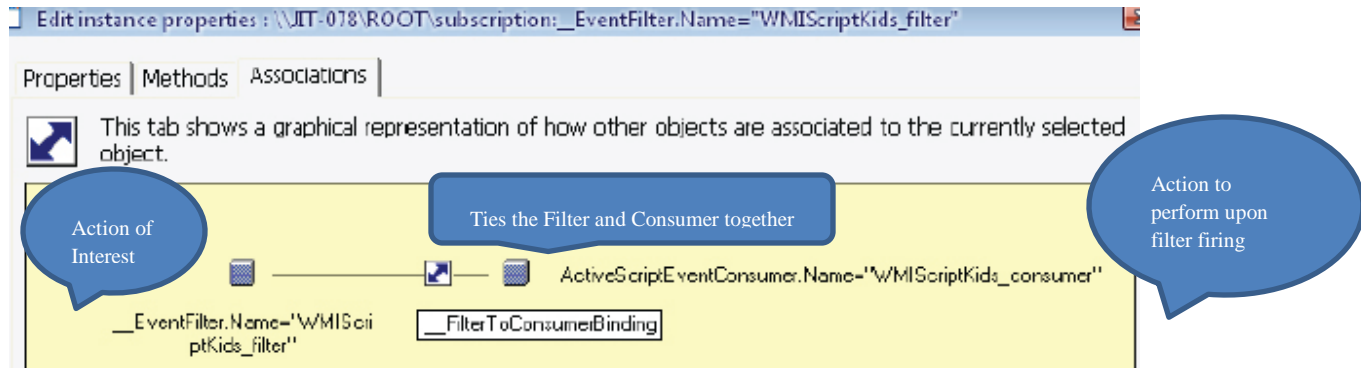


Figure 5: Showing how Filters and Consumers interact with one another by using a FilterToConsumerBinding to tie them together.

For more WMI Event examples, prewritten scripts or a deeper delve into WMI; MSDN and Microsoft TechNet are two great resources. Here are a few links with which to start:

- Enhanced WMI Monitoring Scripts—<https://technet.microsoft.com/en-us/library/ee156569.aspx>
- Monitoring Resources by Using WMI Event Notifications—<https://technet.microsoft.com/en-us/library/ee198937.aspx>
- How WMI Event Notification Works—<https://technet.microsoft.com/en-us/library/ee156572.aspx>
- WMI Script Repository—<https://gallery.technet.microsoft.com/scriptcenter/site/search?f%5B0%5D.Type=Tag&f%5B0%5D.Value=WMI>

5. EVENT TYPES

5.1 Intrinsic Events

Intrinsic events are events that use polling and fire upon the creation, modification, and deletion of any WMI class, object, or namespace. They can also be used to alert to the firing of timers or the execution of WMI methods. The following intrinsic events take the form of system classes (those that start with two underscores) and are present in every WMI namespace:

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

- __InstanceOperationEvent,
- __InstanceCreationEvent,
- __MethodInvocationEvent,
- __InstanceModificationEvent,
- __InstanceDeletionEvent, and
- __TimerEvent.

These events are extremely powerful, because they can be used as triggers for nearly any conceivable event in the operating system.

Because of the rate at which intrinsic events can fire, a polling interval must be specified in queries specified with the WQL WITHIN clause.

Because of the polling interval, it is possible on occasion to miss events. For example, if an event query is formed targeting the creation of a WMI class instance and if that instance is created and destroyed within the polling interval, that event would be missed.

This query is translated to firing on the creation of an instance of a Win32_LogonSession class with a logon type of 2 (Interactive).

Example:

```

■ SELECT * FROM __InstanceCreationEvent WITHIN 15 WHERE TargetInstance
  ISA 'Win32_LogonSession' AND TargetInstance.LogonType = 2

```

5.2 Extrinsic Events

Extrinsic events solve the potential polling issues related to intrinsic events because they fire immediately on an event occurring.

- The downside is not many extrinsic events are present in WMI.
- The events that do exist are extremely powerful but the following extrinsic events may also be of value to an attacker or defender:
 - 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.

This query would capture all executable modules loaded into every process:

Example:

- `SELECT * FROM Win32_ModuleLoadTrace`

6. USING WMI CLI COMMANDS FOR DETECTION AND REMOVAL OF MALICIOUS WMI

6.1 Manual Detection: WMI Command Line Tool

To manually detect instances of the threat in a system, the following commands can be used with the Command line tool:

- `wmic/namespace:\\root\\subscription PATH __EventConsumer get/format:list`
- `wmic/namespace:\\root\\subscription PATH __EventFilter get/format:list`
- `wmic/namespace:\\root\\subscription PATH __FilterToConsumerBinding get/ format:list`
- `wmic/namespace:\\root\\subscription PATH __TimerInstruction get/format:list.`

6.2 Manual Removal: WMI Command Line Tool

To manually remove instances of the threat in a system, the following commands can be used with the Command line tool:

- `wmic/namespace:\\root\\subscription PATH__EventConsumer delete`
- `wmic/namespace:\\root\\subscription PATH__EventFilter delete`
- `wmic/namespace:\\root\\subscription PATH__FilterToConsumerBinding delete`
- `wmic/namespace:\\root\\subscription PATH__TimerInstruction delete`

6.3 WMI Intrusion Detection Using PowerShell Code Ex

The six examples given below show the PowerShell syntax as used for WMI detection.

Format:

- `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[]>]`

Below are three examples of how one could use PowerShell to alert on either EventConsumers, RegistryKey's or StartupCommands.

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'`
- `New-AlertTrigger -StartupCommand | New-AlertAction -Uri 'http://www.awesomeSIEM.com' | Register-Alert`

6.4 Event Logs

WMI, DCom and WinRM events to the following event logs:

- Microsoft-Windows-WinRM/Operational
 - Shows failed WinRM connection attempts including the originating IP address
- Microsoft-Windows-WMIActivity/Operational
 - Contains failed WMI queries and method invocations that may contain evidence of attacker activity
- Microsoft-Windows-DistributedCOM.
 - Shows failed DCOM connection attempts including the originating IP address.

Many Network level IDS and IPS's can incorporate logs from WMI events stored in the above locations and be configured to perform an action such as marking action for review on dashboard, emailing administrator, preventing an action from performing (ex: file being executed), and more depending on the system.

7. ADDITIONAL WMI MITIGATION

7.1 WMI Backup & Restore

Having a WMI baseline for the computer systems within an organization is a great first step in understanding and being able to identify malicious WMI activity and mitigate it.

Wmimgmt.msc as well as the tools listed previously in this paper can all be used to explore the namespaces and associated classes and objects with ease. However, with dozens of namespaces and thousands of classes, it may not be feasible to manually gain a close familiarity with WMI and a baseline thereof.

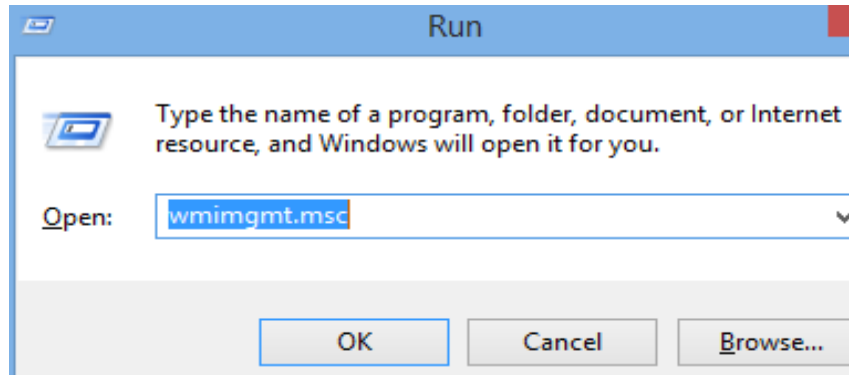


Figure 6: How to start wmicmgmt tool.

NOTE: Click more actions or right click on WMI Control (Local) -> actions -> properties to access the WMI Control (Local) Properties window



Figure 7: wmicmgmt.msc main menu

An alternative to familiarization and baselining WMI is to back up the WMI repository once an organization has configured the system. Having a backup of the WMI repository allows users the option to restore from the backup file when users uncover evidence of malicious WMI activity or suspect potential malicious WMI activity from attackers.

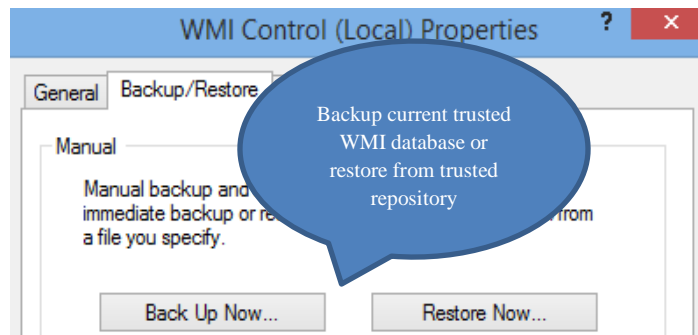


Figure 8: wmicmgmt backup/restore window.

7.2 WMI Access Control

The security tab is located in the same properties window as the Backup/Restore tab. The security tab allows the user to configure user permissions for WMI Repository interaction.

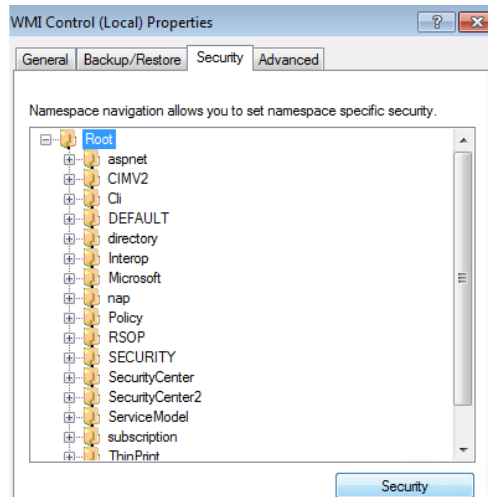


Figure 9: Accessing Root directory to apply access control (security) settings.

Organizations can set the controls to match its preferences. For example, permissions can be set for the entire root directory or for specific namespaces like CIMV2.

Another option is to have a separate account specifically for WMI management with a unique set of credentials. While some would argue that simple techniques, such as hash dumping, render this defensive method irrelevant from a defense-in-depth perspective, granular access control is applicable to WMI management.

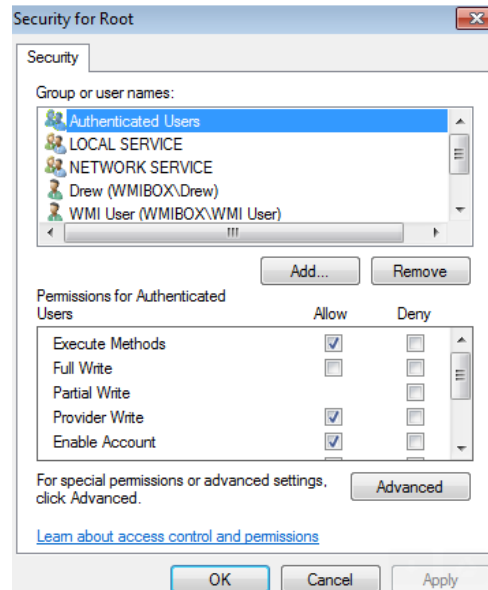


Figure 10: Screenshot of Security for Root.

Appendix A

WMI Interaction and Tools

A1. WMI CODE CREATOR

WMI Code Creator is a popular free tool made available by Microsoft.

The WMI Script interface allows for easy browsing of namespaces and classes that are all listed in the drop down menus. In addition, Code Creator will generate code based on what options have been selected. It lets the user choose which language to write the code: VB, VB Script, or C#.

WMI Code Creator will generate code for WMI queries, method execution, and setting up WMI events, which as discussed later in this paper is the most proactive defense available for mitigating WMI attacks.

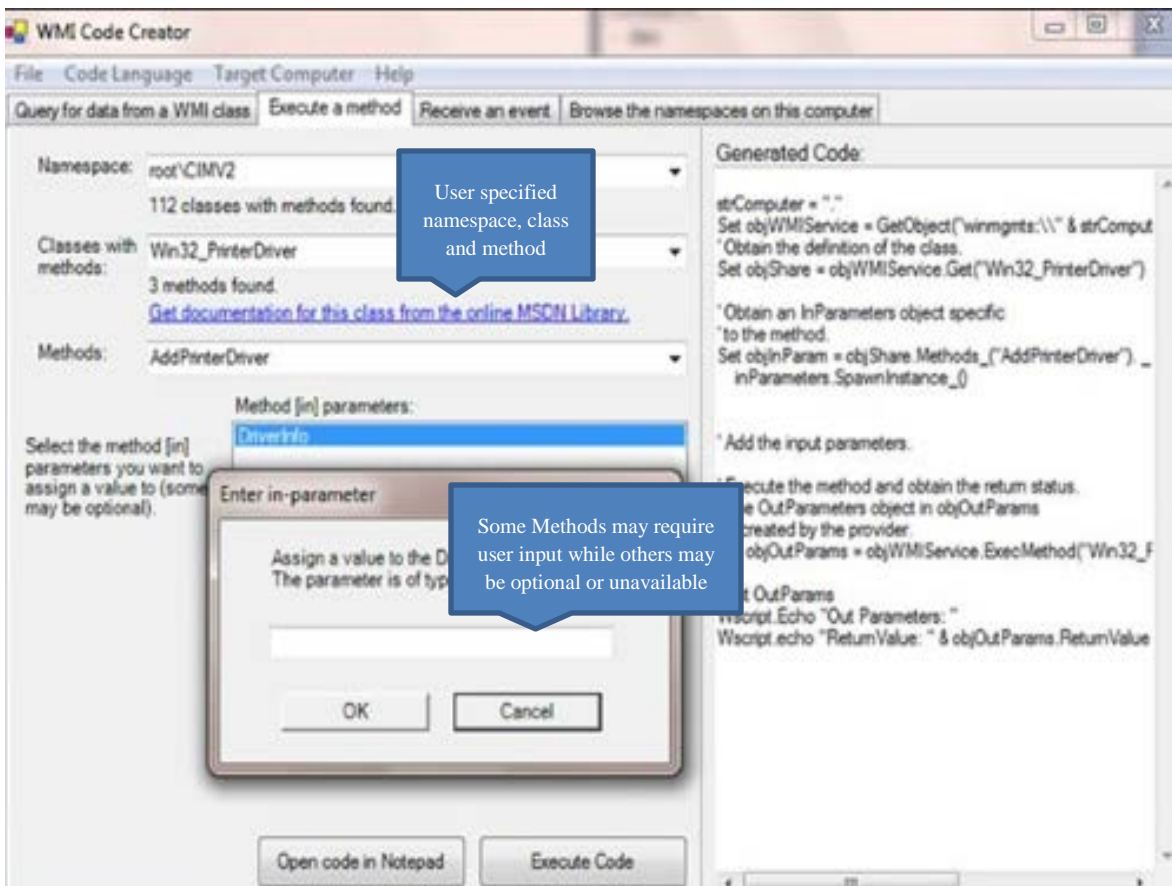


Figure A-1: After selecting namespace, class, and method script will be generated for user which can be further modified. Users then need only supply desired value(s) for script if required.

Here are a few links with a great introduction to Code Creator and a walkthrough for performing queries, executing methods, and returning events with WMI Code Creator:

- <http://hintdesk.com/introduction-to-wmi-code-creator/>
- <https://blogs.technet.microsoft.com/askperf/2010/02/01/two-minute-drill-wmi-code-creator/>

A1.1 PowerShell

PowerShell is becoming more popular and more widely used for a variety of reasons. One reason in particular PowerShell has become a popular method of interacting with WMI is the existence of WMI cmdlets such as:

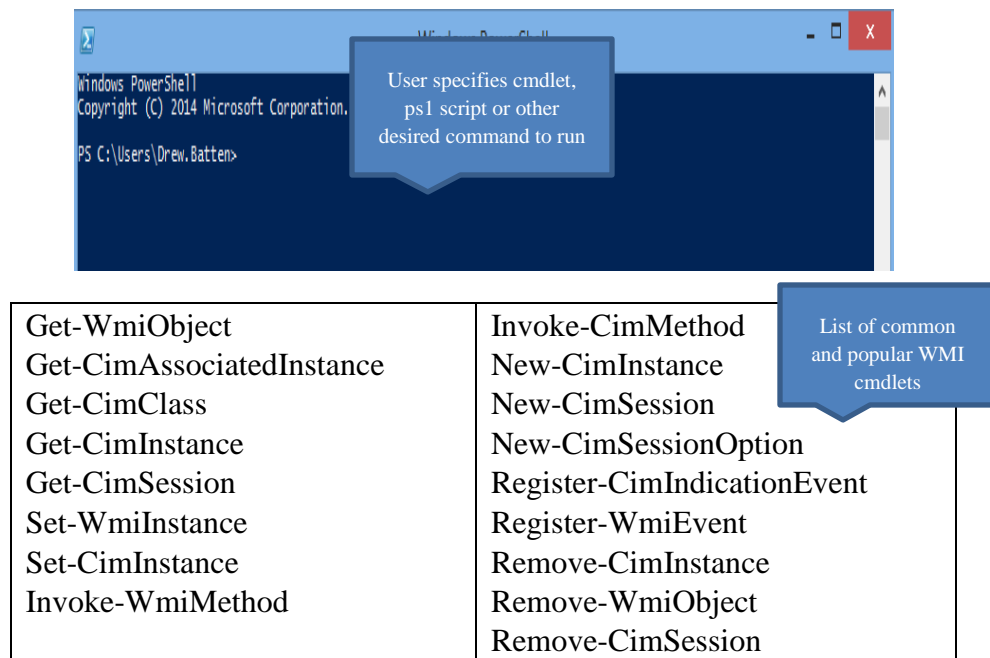


Figure A-2: Powershell prompt and table of popular WMI cmdlets.

This is PowerShell code that detects WMI persistence on the specified remote system. It shows just how easy it is to use PowerShell to interact with WMI.

Example:

- ```
$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
```

A plethora of prewritten WMI PowerShell scripts and walkthroughs is available on the web. An introduction to using PowerShell for event subscriptions can be found at these links:

- <https://learn-powershell.net/2013/08/14/powershell-and-events-permanent-wmi-event-subscriptions/>
- <https://learn-powershell.net/2013/08/02/powershell-and-events-wmi-temporary-event-subscriptions/>

A1.2 Wmic.exe

Wmic.exe is a powerful command line utility for interacting with WMI. It has a large amount of default aliases for WMI objects, and users can perform more complicated queries.

Wmic.exe can execute WMI methods, and attackers use it to perform lateral movement by using the Win32_ProcessCreate method.



Figure A-3: Administrators and malicious attackers can use the Wmic.exe terminal to execute WMI methods.

In circumstances where PowerShell is not available, Wmic.exe is a sufficient alternative for performing reconnaissance and basic method invocation.

A1.3 Wbemtest.exe

Wbemtest.exe is a powerful GUI WMI diagnostic tool. It isn't pretty, but it sure is useful. It allows you to explore deep into the WMI repository to discover what an administrator might be able to harness in PowerShell scripts. It is able to enumerate object instances, class names, get properties and methods, get property datatypes, perform queries, register events, modify WMI objects and classes, and invoke methods both locally and remotely.

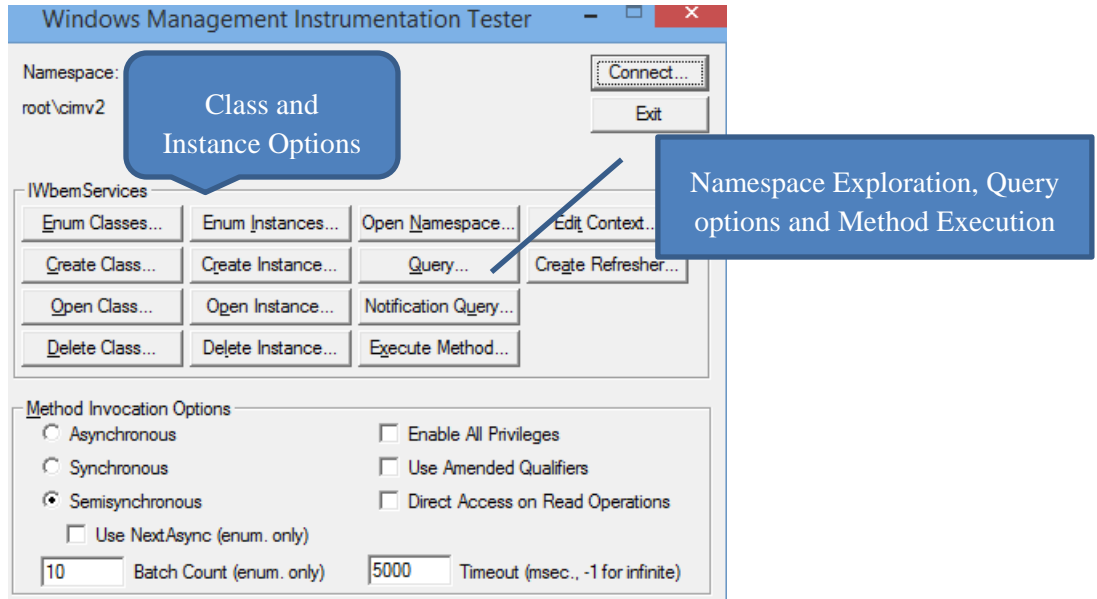


Figure A-4: Wbemtest.exe dashboard once connected.

A1.4 WMI Explorer

WMI Explorer is a great WMI class discovery tool. It has a polished GUI that allows the user to explore the WMI repository in a hierarchical fashion. It is also able to connect to remote WMI repositories and perform queries.

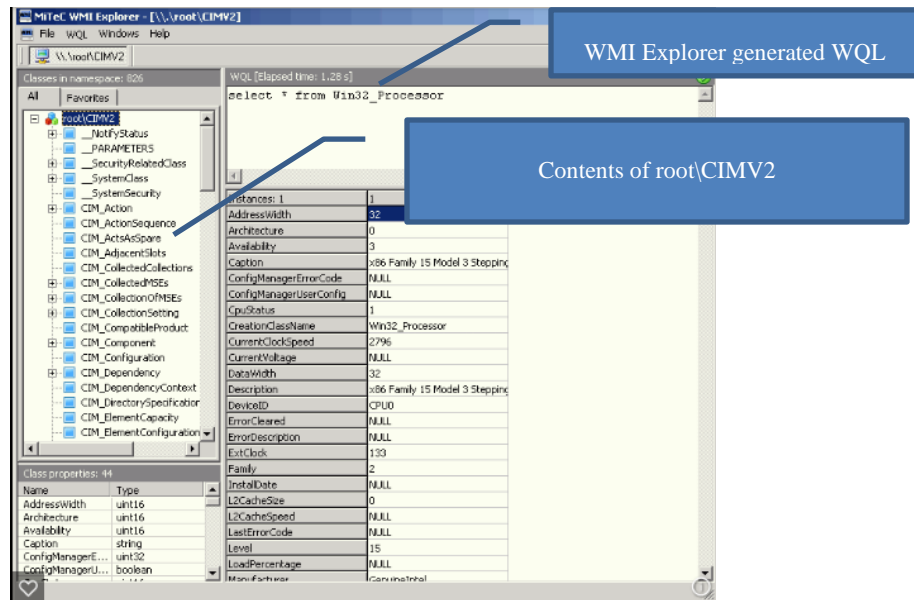


Figure A-5: Contents of root\CIMV2 directory and WMI Explorer generated WQL language to select all contents from Win32_Processor object

A1.5 CIM Studio

CIM Studio is a free, legacy tool from Microsoft that allows the user to easily browse the WMI repository. Like WMI Explorer, this tool is good for WMI class discovery.

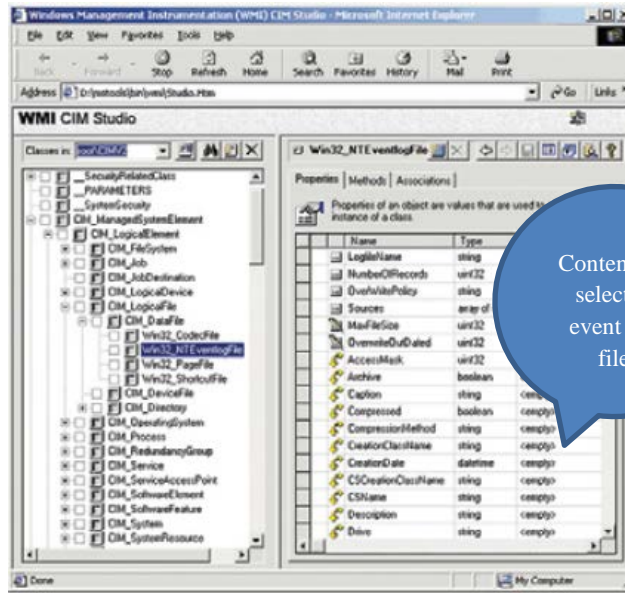


Figure A-6: Exploring down 5 levels into the root\CIMV2 directory and displaying the contents of selected object.