Advanced Scripting   
CIM Namespaces and Event Subscription

Last Updated: 2/20/2024 12:31 AM Version 1  
Document Prepared for: CYBER360 Student

# Name Click here to enter name ID Click here to enter id

# Instructions

Answer all questions directly in this document. You will save and upload this completed document as your homework submission.

# Overview

CIM isn’t just a framework of classes and objects; it’s an infrastructure for enabling system and security automation.

# Requirements

Microsoft Windows.

# Setup

Launch Windows PowerShell (either Core or Desktop).

# Task 1 — Download and use a script to enumerate CIM namespaces

The cmdlet **Get-CimInstance -ClassName '\_\_Namespace'** can only query one namespace at a time. To overcome this limitation, the textbook author wrote a function that can recursively enumerate all of the CIM namespaces.

## Steps

1. Launch a new PowerShell with elevated administrator privilege.
2. At your administrator PowerShell CLI prompt, change your working directory to **C:\Temp** (or to some other temporary folder of your choice):   
   **Set-Location C:\Temp**
3. Download the author’s script to the temporary directory. Enter (all on one line):   
   **Invoke-WebRequest https://raw.githubusercontent.com/PacktPublishing/PowerShell-Automation-and-Scripting-for-Cybersecurity/master/Chapter05/Get-CimNamespace.ps1   
   -OutFile C:\Temp\Get-CimNamespace.ps1**
4. *Dot-source* the downloaded script to install her Get-CimNamespace function:   
   **. .\Get-CimNamespace.ps1**
5. Read the help documentation she provided with her function:   
   **get-help get-cimnamespace**
6. Execute the function with its **-Recurse** switch parameter to enumerate all of the CIM namespaces:   
   **Get-CimNamespace -Namespace 'root' -Recurse**
   1. What are the two namespaces under **root\cimv2\Security** ? Click or tap here to enter text.
   2. What are the four namespaces under **root\Microsoft\PolicyPlatform** ? Click or tap here to enter text.

# Task 2 — Explore namespaces with Get-CimClass

Continue using your administrator privileged PowerShell from the previous task.

## Steps

1. List all of the classes in the root\Hardware namespace:   
   **Get-CimClass -Namespace 'root\Hardware'**
2. List all of the classes in the root\Microsoft\Windows\Storage namespace:   
   **Get-CimClass -Namespace 'root\Microsoft\Windows\Storage'**
   1. Let’s drill down into the **MSFT\_StorageSetting** class in that namespace. Enter (all on one line):   
      **Get-CimClass -Namespace 'root\Microsoft\Windows\Storage' -ClassName MSFT\_StorageSetting**
   2. Drilling down further, list the methods contained in that class:   
      **(Get-CimClass -Namespace 'root\Microsoft\Windows\Storage' -ClassName MSFT\_StorageSetting).CimClassMethods**
      1. Write a command to list just the method names on one line separated by commas. Your command: Click or tap here to enter text.
      2. Output of that command: Click or tap here to enter text.
   3. List the properties contained in that class:   
      **(Get-CimClass -Namespace 'root\Microsoft\Windows\Storage' -ClassName MSFT\_StorageSetting).CimClassProperties**
      1. What are the property names (on one line, separated by commas)? Click or tap here to enter text.
3. Explore the **Win32\_OperatingSystem** class from the default namespace:   
   **Get-CimClass -ClassName Win32\_OperatingSystem**
   1. List the Methods contained in that class (on one line, separated by commas):
   2. List the Properties contained in that class (on one line, separated by commas):
4. Get the CIM *instances* (objects) of the **Win32\_OperatingSystem** class:   
   **Get-CimInstance -ClassName Win32\_OperatingSystem**
   1. List the Properties contained in that object:   
      **(Get-CimInstance -ClassName Win32\_OperatingSystem).CimInstanceProperties**
   2. What are the property names (on one line, separated by commas)? Click or tap here to enter text.
5. List the CIM *event* classes:   
   **Get-CimClass -ClassName "\*Event"**
   1. Drill down to see the CimSystemProperties of the event classes:   
      (**Get-CimClass -ClassName "\*Event").CimSystemProperties**
   2. Filter the list to just the *intrinsic* event classes, which are those whose CIM class names start with two underscore “**\_**” characters:  
      (**Get-CimClass -ClassName "\*Event").CimSystemProperties | ? { $\_.ClassName -like "\_\_\*" }**
   3. What are the intrinsic event class names (on one line, separated by commas)? Click or tap here to enter text.
   4. Now filter the list from step 5.1 to just the *extrinsic* event classes:   
      **(Get-CimClass).CimSystemProperties | ? { $c=$\_.ClassName; ($c -notlike "\_\_\*") -and (($c -like "\*Event") -or ($c -like "\*Trace"))}**
   5. What are the extrinsic event class names (on one line, separated by commas)? Click or tap here to enter text.

# Task 3 — Set Up a CIM event subscription

For this task, use your Windows VM in our class lab. First, connect to the lab bastion host (HORACE):   
 **ssh *<your-username>*@cit361-lab.cit.byui.edu**   
Then from there, log into your independent Windows VM in the class lab network:   
 **ssh *<user-that-you-created>*@*<IP address of your Windows VM>***

The textbook outlines four steps for setting up an event subscription:

1. Create a query that identifies an event that we want to monitor.
2. Create a **CimInstance** of class **\_\_EventFilter** that contains the query.
3. Create a **CimInstance** of one of the consumer classes, to respond appropriately to the event.
4. Create a **CimInstance** of class **\_\_FilterToConsumerBinding** that brings it all together.

Those are general descriptions of the subscription process. Here are a few specific notes for how we will follow these steps to log a Windows Event when the License Manager service stops:

1. We’ll query **\_\_InstanceDeletionEvent**, looking for a **Win32\_Service** named **LicenseManager**.
2. We’ll specify the name **'**LicenseManagerFilter**'** in our **\_\_EventFIlter** CimInstance.
3. We’ll use **NTEventLogEventConsumer** for our consumer class.
4. We’ll combine our event filter and consumer with our **\_\_FilterToConsumerBinding** CimInstance.

## Steps

1. Create a query string using the *WMI Query Language* (WQL) syntax. (*It’s similar to SQL.*)
   1. Enter (all on one line):   
      **$query = "select \* from \_\_InstanceDeletionEvent within 15 where TargetInstance ISA 'Win32\_Service' AND TargetInstance.name='LicenseManager'"**   
      This query inspects *Instance Deletion Events* every fifteen seconds, checking to see whether it finds one for a *Win32\_Service* instance named *LicenseManager*.
2. Create an **\_\_EventFilter** CimInstance.
   1. Before we start, get a list of the current **\_\_EventFilter** instances:   
      **Get-CimInstance -Namespace root\subscription -ClassName \_\_EventFilter**
   2. Enter the following to create a summary of your new filter’s properties as a hash table object:   
      **$EventFilterProperties = @{**   
       **Name = 'LicenseManagerFilter';**   
       **Query = $query;**   
       **QueryLanguage = "WQL";**   
       **EventNamespace = "\root\cimv2";**   
      **}**
   3. Create the CimInstance:   
      **$LicenseManagerCimInstance = New-CimInstance -ClassName \_\_EventFilter -Namespace "root\subscription" -Property $EventFilterProperties**
   4. Get another list of the **\_\_EventFilter** instances:   
      **Get-CimInstance -Namespace root\subscription -ClassName \_\_EventFilter**   
      Compare to step 2.1. Verify that you can find your new filter in the output. Copy the value of its Name property here: Click or tap here to enter text.
3. Create an **NTEventLogEventConsumer** CimInstance.
   1. Before we start, get a list of the current **NTEventLogEventConsumer** instances:   
      **Get-CimInstance -Namespace root\subscription -ClassName NTEventLogEventConsumer**
   2. Make an array with a log message string that clearly communicates the event:   
      **$Message = @("%TargetInstance.Name% service stopped on $Env:COMPUTERNAME. Current status: %TargetInstance.Status%")**
   3. Create a summary of the consumer’s properties as a hash table object:   
      **$ConsumerProperties = @{**   
       **Name = 'LicenseManager service was terminated';**   
       **MachineName = $Env:COMPUTERNAME;**   
       **EventID = [uint32]7104867;**   
       **SourceName = 'Application';**   
       **NumberOfInsertionStrings = [uint32]$Message.Count;**   
       **InsertionStringTemplates = $Message;**   
       **Category = [uint16]108;**   
      **}**
   4. Create the CimInstance:   
      **$ConsumerCimInstance = New-CimInstance -ClassName NTEventLogEventConsumer -Namespace 'ROOT\subscription' -Property $ConsumerProperties**
   5. Get another list of the NTEventLogEventConsumer instances:   
      **Get-CimInstance -Namespace root\subscription -ClassName NTEventLogEventConsumer**   
      Compare to step 3.1. Verify that you can find your new consumer in the output. Copy the value of its Name property here: Click or tap here to enter text.
4. Create a **\_\_FilterToConsumerBinding** CimInstance.
   1. Before we start, get a list of the **\_\_FilterToConsumerBinding** instances:   
      **Get-CimInstance -Namespace root\subscription -ClassName \_\_FilterToConsumerBinding**
   2. Create a summary of the binder’s properties as a hash table object:   
      **$BinderProperties = @{**   
       **Filter = [Ref]$LicenseManagerCimInstance;**   
       **Consumer = [Ref]$ConsumerCimInstance;**   
      **}**
   3. Create the CimInstance:   
      **$BinderCimInstance = New-CimInstance -ClassName \_\_FilterToConsumerBinding -Namespace "root/subscription" -Property $BinderProperties**
   4. Get another list of the **\_\_FilterToConsumerBinding** instances:   
      **Get-CimInstance -Namespace root\subscription -ClassName \_\_FilterToConsumerBinding**   
      Compare to step 4.1. Verify that you can find your new binding in the output. Copy the value of its Filter property here: Click or tap here to enter text.
5. Now let’s try to trigger an event!
   1. Enter:   
      **Get-Service LicenseManager**
      1. If the License Manager is not currently running, start the service:   
         **Get-Service LicenseManager | Start-Service**   
         Then get the service again to verify that it’s running:   
         **Get-Service LicenseManager**
   2. Stop the service:   
      **Get-Service LicenseManager | Stop-Service**
   3. Query the Application event log. Try one or more the following:   
      **Get-WinEvent Application | ? { $\_.Id -eq 7104867 }**   
      **Get-WinEvent Application | ? { $\_.LevelDisplayName -eq 'Warning' }**   
      Do you see an event that was produced when you stopped the LicenseManager service? Click or tap here to enter text.
   4. Query the Microsoft-Windows-WMI-Activity/Operational event log:   
      **Get-WinEvent Microsoft-Windows-WMI-Activity/Operational | ? { $\_.Id   
      -eq 5861 } | fl**   
      Do you see an event produced when you created your subscription? Click or tap here to enter text.

# Task 4 (optional) — Remove your CIM event subscription

For this task, again use your independent Windows VM in the class lab network.

Use the Remove-CimInstance cmdlet to roll back the CimInstances you created in the previous task.

1. Enter:   
   **Get-CimInstance -Namespace 'root\subscription' -ClassName \_\_FilterToConsumerBinding -Filter "Filter = ""\_\_EventFilter.name='LicenseManagerFilter'""" | Remove-CimInstance**
2. Enter:   
   **Get-CimInstance -Namespace "root/subscription" -ClassName NTEventLogEventConsumer -Filter "name='LicenseManager service was terminated'" | Remove-CimInstance**
3. Enter:   
   **Get-CimInstance -Namespace 'ROOT/subscription' -ClassName \_\_EventFilter -Filter "name='LicenseManagerFilter'" | Remove-CimInstance**

# Deliverable

Upload this document with completed answers to I-Learn Canvas.