Advanced Scripting   
CIM Namespaces and Event Subscription

Last Updated: 6/18/2024 8:19 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. List all of the namespaces that you see under **root\StandardCimv2** : Click or tap here to enter text.

# Task 2 — Explore namespaces with **Get-CimClass** and **Get-CimInstance**

Continue using your administrator privileged PowerShell from the previous task.

## Steps

1. List all the classes in the root\Hardware namespace:   
   **Get-CimClass -Namespace 'root\Hardware'**
2. List all 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 CimClassMethods contained in that class (on one line, separated by commas): Click or tap here to enter text.
   2. List the CimClassProperties contained in that class (on one line, separated by commas): Click or tap here to enter text.
4. Get the CIM *instances* (objects) of the **Win32\_OperatingSystem** class:   
   **Get-CimInstance -ClassName Win32\_OperatingSystem**
   1. List the Properties contained in that CimInstance 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. (*Hint: the properties of the CimInstance should be the same as the CimClassProperties you listed in step 3.2.*)
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 — Look at an existing CIM event subscription

## Setup

For this and the next task, use your Windows VM in our class lab. Start by connecting 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. Let’s get *specific*: here’s how we will follow these steps to write a log whenever the License Manager service starts or stops:

1. We’ll query **\_\_InstanceModificationEvent** for a **Win32\_Service** named **LicenseManager**.
2. We’ll specify the name **'**LicenseManagerFilter**'** in our **\_\_EventFilter** CimInstance.
3. We’ll use consumer class **LogFileEventConsumer**. (Since this is an exercise and not a production configuration, we’ll just put a log file in **C:\Temp**. If this folder doesn’t exist on your VM, enter:   
    **mkdir C:\Temp**   
   to create it now.)
4. We’ll combine our event filter and consumer with a **\_\_FilterToConsumerBinding** CimInstance.

There are a lot of details to configure correctly in each of these steps. If we were doing this on a local workstation, it would be better to script our procedure’s steps in a .ps1 script file. But we’re doing this on a remote machine, so we’ll just do each step at the command line.

## Steps

For this task, let’s check for any event subscription related CIM instances that already exist.

1. Enter:  
   **Get-CimInstance -Namespace root\subscription -ClassName \_\_EventFilter**   
   You should see an **\_\_EventFilter** instance named *SCM Event Log Filter*. What is the value of its **Query** property? Click or tap here to enter text.   
   (SCM means *Service Control Manager*. This event filter is a default event logging subscription on every contemporary Microsoft Windows host. Take extra care that you don’t delete or remove this!)
2. Enter (all on one line):  
   **Get-CimInstance -Namespace root\subscription -ClassName NTEventLogEventConsumer**   
   You should see an **NTEventLogEventConsumer** instance named *SCM Event Log Consumer*.
   1. What is the value of its SourceName property? Click or tap here to enter text.
   2. In the next task, we’ll create a new **LogFileEventConsumer** instance, so let’s check for those. Enter (all on one line):  
      **Get-CimInstance -Namespace root\subscription -ClassName LogFileEventConsumer**   
      (You probably won’t get any output, but if you do, take extra care that you don’t delete or remove those pre-existing consumer instances!)
3. Enter (all on one line):   
   **Get-CimInstance -Namespace root\subscription -ClassName \_\_FilterToConsumerBinding**   
   You should see a **\_\_FilterToConsumerBinding** instance.
   1. What’s the name embedded in its **Consumer** property? Click or tap here to enter text.   
      (*Hint: It should match the consumer you found in step 2.*)
   2. What’s the name embedded in its **Filter** property? Click or tap here to enter text.   
      (*Hint: It should match the filter you found in step 1.*)
   3. Again, take extra care that you don’t delete or remove this binding instance!

# Task 4 —Set Up a new CIM event subscription

## Steps

1. Create a query string using *WMI Query Language* (WQL) syntax. (*It’s similar to SQL.*)
   1. Enter (all on one line):   
      **$query = "select \* from \_\_InstanceModificationEvent within 30 where TargetInstance ISA 'Win32\_Service' AND TargetInstance.name='LicenseManager'"**   
        
      This query inspects *Instance Modification Events* every thirty seconds, checking to see whether it finds one for a *Win32\_Service* instance named *LicenseManager*.
2. Create an **\_\_EventFilter** CimInstance.
   1. Reminder: notice the pre-existing **\_\_EventFilter** instances you found in the previous task.
   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";**   
      **}**   
         
      *Alternatively, if you prefer, you could build this hash table one property at a time:*   
        
      **$EventFilterProperties = @{}**   
      **$EventFilterProperties.Name = 'LicenseManagerFilter'**   
      **$EventFilterProperties.Query = $query**   
      **$EventFilterProperties.QueryLanguage = "WQL"**   
      **$EventFilterProperties.EventNamespace = "\root\cimv2"**
   3. Double-check your hash table, making sure all of property names and values are correct:   
        
      **$EventFilterProperties | Format-List**
   4. To create the event filter CimInstance, enter (all on one line):   
        
      **$FilterCimInstance = New-CimInstance -ClassName \_\_EventFilter -Namespace "root\subscription" -Property $EventFilterProperties**
   5. Review the **\_\_EventFilter** instances:   
      **Get-CimInstance -Namespace root\subscription -ClassName \_\_EventFilter**   
      Compare to step 1 of the previous task. 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. (*Hint: if your answer is* **SCM Event Log Filter** *then you did it wrong. That’s one of the previously existing filters, not your new filter! Try again.*)
3. Create a **LogFileEventConsumer** CimInstance.
   1. Create a summary of the consumer’s properties as a hash table object:   
         
      **$ConsumerProperties = @{**   
       **Name = 'LicenseManagerConsumer';**   
       **Text = "License manager state changed on $Env:COMPUTERNAME";**   
       **Filename = "C:\Temp\LicenseManagerEvent.log";**   
       **MaximumFileSize = [uint64](512 \* 1kb);**   
      **}**   
        
      Alternatively, you could build this hash table one property at a time:  
        
       **$ConsumerProperties = @{}**   
       **$ConsumerProperties.Name = "LicenseManagerConsumer"**   
       **$ConsumerProperties.Text = "License Manager's state changed"**   
       **$ConsumerProperties.Text += " on $Env:COMPUTERNAME"**   
       **$ConsumerProperties.Filename = "C:\Temp\LicenseManagerEvent.log"**   
       **$ConsumerProperties.MaximumFileSize = [uint64](512 \* 1kb)**
   2. Double-check your hash table, making sure all of property names and values are correct:   
        
      **$ConsumerProperties | FL**
   3. To create the event consumer CimInstance, enter (all on one line):   
        
      **$ConsumerCimInstance = New-CimInstance -ClassName LogFileEventConsumer -Namespace 'ROOT\subscription' -Property $ConsumerProperties**
   4. Review the LogFileEventConsumer instances:   
        
      **Get-CimInstance -Namespace root\subscription -ClassName LogFileEventConsumer**   
      Compare to the previous task’s step 2.2. 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. Create a summary of the binder’s properties as a hash table object:   
        
      **$BinderProperties = @{**   
       **Filter = [Ref]$FilterCimInstance;**   
       **Consumer = [Ref]$ConsumerCimInstance;**   
      **}**
   2. To create the binder CimInstance, enter (all on one line):   
        
      **$BinderCimInstance = New-CimInstance -ClassName \_\_FilterToConsumerBinding -Namespace "root/subscription" -Property $BinderProperties**
   3. Review the **\_\_FilterToConsumerBinding** instances:   
        
      **Get-CimInstance -Namespace root\subscription -ClassName \_\_FilterToConsumerBinding**   
        
      Compare to the previous task’s step 3. Verify that you can find your new binding in your 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. Look for the log in the \temp folder:   
      **Get-ChildItem C:\Temp**   
      (It might take a few seconds for your new CIM event subscription to create the log file.)
   4. Examine the log:  
      **Get-Content C:\Temp\LicenseManagerEvent.log**   
      Your output: Click or tap here to enter text.
   5. Query the Microsoft-Windows-WMI-Activity/Operational event log:   
        
      **Get-WinEvent Microsoft-Windows-WMI-Activity/Operational | ? { $\_.Id   
      -eq 5861 } | FL | more**   
        
      Find the event that was produced when you created your subscription. What’s the value of its **TimeCreated** property? Click or tap here to enter text.

# Task 5 (optional) — Remove your CIM event subscription

For this task, continue to 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.

## Steps

1. If you haven’t yet closed the remote PowerShell session you used in task 4, removals are easy:  
   **$BinderCimInstance | Remove-CimInstance**   
   **$ConsumerCimInstance | Remove-CimInstance**   
   **$FilterCimInstance | Remove-CimInstance**   
     
   If these work successfully, you’re done; skip the following steps 2-4! Otherwise, you’ll need to find the CimInstance objects that you created.
2. To find the binder you created, enter (all on one line):   
      
   **Get-CimInstance -Namespace 'root\subscription' -ClassName \_\_FilterToConsumerBinding -Filter "Filter = ""\_\_EventFilter.name='LicenseManagerFilter'"""**   
   1. When you’re sure you’ve found the right binder, pipe it into the **Remove-CimInstance** cmdlet:   
        
      **Get-CimInstance -Namespace 'root\subscription' -ClassName \_\_FilterToConsumerBinding -Filter "Filter = ""\_\_EventFilter.name='LicenseManagerFilter'""" | Remove-CimInstance**
3. To find and remove the consumer you created:   
      
   **Get-CimInstance -Namespace "root/subscription" -ClassName LogFileEventConsumer -Filter "name='LicenseManagerConsumer'"  
    | Remove-CimInstance**
4. To find and remove the filter you created:   
   **Get-CimInstance -Namespace 'ROOT/subscription' -ClassName \_\_EventFilter  
    -Filter "name='LicenseManagerFilter'" | Remove-CimInstance**

# Deliverable

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