Challenge: Operation Blackout 2025: Phantom Check

Platform: HackTheBox

Category: Endpoint Forensics

Difficulty: Medium

Tools Used: EvtxECmd, Timeline Explorer

Summary: This challenge involved investigating PowerShell logs to uncover virtualisation detection techniques. I personally didn't enjoy this Sherlock that much, as it only required you to look at a couple of logs.

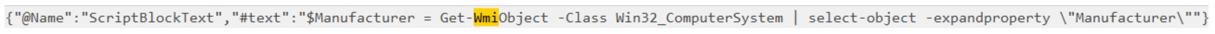
Scenario: Talion suspects that the threat actor carried out anti-virtualization checks to avoid detection in sandboxed environments. Your task is to analyze the event logs and identify the specific techniques used for virtualization detection. Byte Doctor requires evidence of the registry checks or processes the attacker executed to perform these checks.

Which WMI class did the attacker use to retrieve model and manufacturer information for virtualization detection?

Before we dive into this question, we first need to parse the event logs. I am going to use a tool created by Eric Zimmerman called EvtxECmd, although feel free to use Event Log Explorer, Event Viewer, etc:

- `.\EvtxECmd.exe -d . --csv . --csvf out.csv`
    - Where -d . specifies to recursively parse the event logs in the current directory.
    - --csv . specifies to output a csv file, and
    - --csvf specifies the output filename

I am then going to use Timeline Explorer, another Eric Zimmerman tool, to view the CSV file. If you search for the keyword WMI and focus on Event ID 4104 (logs the contents of executed PowerShell scripts), we can find something interesting:

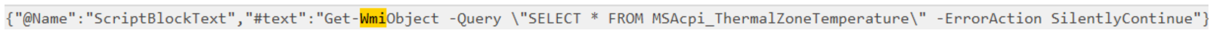


```
{"@Name":"ScriptBlockText","#text":"$Manufacturer = Get-WmiObject -Class Win32_ComputerSystem | select-object -expandproperty \"Manufacturer\""}
```

Answer: Win32_ComputerSystem

Which WMI query did the attacker execute to retrieve the current temperature value of the machine?

If you continue exploring Event ID 4104, you will come across a script that used the MSAcpi_ThermalZoneTemperature class:



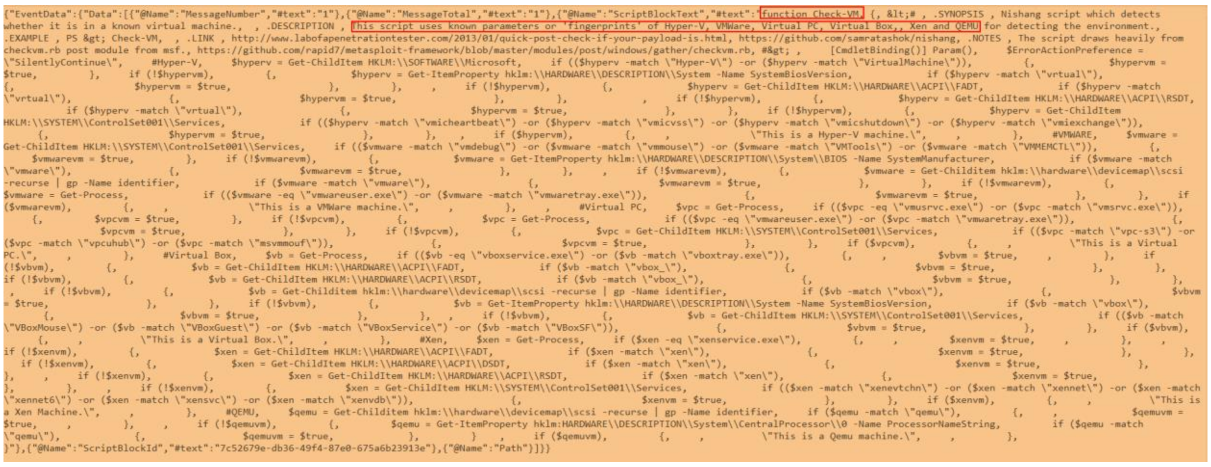
```
{"@Name":"ScriptBlockText","#text":"Get-WmiObject -Query \"SELECT * FROM MSAcpi_ThermalZoneTemperature\" -ErrorAction SilentlyContinue"}
```

After Googling this class, and using some common sense, I determined that this is used to retrieve the current temperature of components like the CPU.

Answer: SELECT * FROM MSAcpi_ThermalZoneTemperature

The attacker loaded a PowerShell script to detect virtualization. What is the function name of the script?

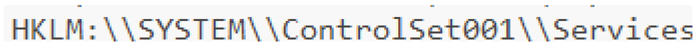
After exploring Event ID 4104 further, I came across a script that contained a function called Check-VM:



```
{"EventData":{"Data":[{"@Name":"MessageNumber","#text":"1"},{"@Name":"MessageTotal","#text":"1"},{"@Name":"ScriptBlockText","#text":"function Check-VM, {, &lt;# , .SYNOPSIS , Nishang script which detects
whether it is in a known virtual machine.,  , .DESCRIPTION , This script uses known parameters or 'fingerprints' of Hyper-V, VMWare, Virtual PC, Virtual Box,, Xen and QEMU for detecting the environment.,
.EXAMPLE , PS &gt; Check-VM,  , .LINK , http://www.labofapenetrationtester.com/2013/01/quick-post-check-if-your-payload-is.html, https://github.com/samratashok/nishang, .NOTES , The script draws heavily from
checkvm.rb post module from msf., https://github.com/rapid7/metasploit-framework/blob/master/modules/post/windows/gather/checkvm.rb, #&gt; ,    [CmdletBinding()] Param(),    $ErrorActionPreference =
\"SilentlyContinue\",    #Hyper-V,    $hyperv = Get-ChildItem HKLM:\\SOFTWARE\\Microsoft,    if (($hyperv -match \"Hyper-V\") -or ($hyperv -match \"VirtualMachine\")),    {,    $hypervm =
$true,    },    if (!$hypervm),    {,    $hyperv = Get-ItemProperty hklm:\\HARDWARE\\DESCRIPTION\\System -Name SystemBiosVersion,    if ($hyperv -match \"vrtual\"),
{,    $hypervm = $true,    },    },  ,    if (!$hypervm),    {,    $hyperv = Get-ChildItem HKLM:\\HARDWARE\\ACPI\\FADT,    if ($hyperv -match
\"vrtual\"),    {,    $hypervm = $true,    },    },  ,    if (!$hypervm),    {,    $hyperv = Get-ChildItem HKLM:\\HARDWARE\\ACPI\\RSDT,
    if ($hyperv -match \"vrtual\"),    {,    $hypervm = $true,    },    },    if (!$hypervm),    {,    $hyperv = Get-ChildItem
HKLM:\\SYSTEM\\ControlSet001\\Services,    if (($hyperv -match \"vmicheartbeat\") -or ($hyperv -match \"vmicvss\") -or ($hyperv -match \"vmicshutdown\") -or ($hyperv -match \"vmiexchange\")),
  {,    $hypervm = $true,    },    },  ,    if ($hypervm),    {,  ,    \"This is a Hyper-V machine.\",  ,    },    #VMWARE,    $vmware =
Get-ChildItem HKLM:\\SYSTEM\\ControlSet001\\Services,    if (($vmware -match \"vmdebug\") -or ($vmware -match \"vmmouse\") -or ($vmware -match \"VMTools\") -or ($vmware -match \"VMMEMCTL\")),    {,
  $vmwarevm = $true,    },    if (!$vmwarevm),    {,    $vmware = Get-ItemProperty hklm:\\HARDWARE\\DESCRIPTION\\System\\BIOS -Name SystemManufacturer,    if ($vmware -match
\"vmware\"),    {,    $vmwarevm = $true,    },    },  ,    if (!$vmwarevm),    {,    $vmware = Get-Childitem hklm:\\hardware\\devicemap\\scsi
-recurse | gp -Name identifier,    if ($vmware -match \"vmware\"),    {,    $vmwarevm = $true,    },    },    if (!$vmwarevm),    {,
$vmware = Get-Process,    if (($vmware -eq \"vmwareuser.exe\") -or ($vmware -match \"vmwaretray.exe\")),    {,    $vmwarevm = $true,    },    },    if
($vmwarevm),    {,  ,    \"This is a VMWare machine.\",  ,    },  ,    #Virtual PC,    $vpc = Get-Process,    if (($vpc -eq \"vmusrvc.exe\") -or ($vpc -match \"vmsrvc.exe\")),
  {,    $vpcvm = $true,    },    if (!$vpcvm),    {,    $vpc = Get-Process,    if (($vpc -eq \"vmwareuser.exe\") -or ($vpc -match \"vmwaretray.exe\")),    {,
    $vpcvm = $true,    },    },    if (!$vpcvm),    {,    $vpc = Get-ChildItem HKLM:\\SYSTEM\\ControlSet001\\Services,    if (($vpc -match \"vpc-s3\") -or
($vpc -match \"vpcuhub\") -or ($vpc -match \"msvmmouf\")),    {,    $vpcvm = $true,    },    },    if ($vpcvm),    {,  ,    \"This is a Virtual
PC.\",  ,    },    #Virtual Box,    $vb = Get-Process,    if (($vb -eq \"vboxservice.exe\") -or ($vb -match \"vboxtray.exe\")),    {,  ,    $vbvm = $true,  ,    },    if
(!$vbvm),    {,    $vb = Get-ChildItem HKLM:\\HARDWARE\\ACPI\\FADT,    if ($vb -match \"vbox_\"),    {,    $vbvm = $true,    },    },
if (!$vbvm),    {,    $vb = Get-ChildItem HKLM:\\HARDWARE\\ACPI\\RSDT,    if ($vb -match \"vbox_\"),    {,    $vbvm = $true,    },    },
 ,    if (!$vbvm),    {,    $vb = Get-Childitem hklm:\\hardware\\devicemap\\scsi -recurse | gp -Name identifier,    if ($vb -match \"vbox\"),    {,    $vbvm
= $true,    },    },    if (!$vbvm),    {,    $vb = Get-ItemProperty hklm:\\HARDWARE\\DESCRIPTION\\System -Name SystemBiosVersion,    if ($vb -match \"vbox\"),
  {,    $vbvm = $true,    },    },  ,    if (!$vbvm),    {,    $vb = Get-ChildItem HKLM:\\SYSTEM\\ControlSet001\\Services,    if (($vb -match
\"VBoxMouse\") -or ($vb -match \"VBoxGuest\") -or ($vb -match \"VBoxService\") -or ($vb -match \"VBoxSF\")),    {,    $vbvm = $true,    },    },    if ($vbvm),
  {,  ,    \"This is a Virtual Box.\",  ,    },    #Xen,    $xen = Get-Process,    if ($xen -eq \"xenservice.exe\"),    {,  ,    $xenvm = $true,  ,    },  ,
if (!$xenvm),    {,    $xen = Get-ChildItem HKLM:\\HARDWARE\\ACPI\\FADT,    if ($xen -match \"xen\"),    {,    $xenvm = $true,    },    },
  if (!$xenvm),    {,    $xen = Get-ChildItem HKLM:\\HARDWARE\\ACPI\\DSDT,    if ($xen -match \"xen\"),    {,    $xenvm = $true,    },
},  ,    if (!$xenvm),    {,    $xen = Get-ChildItem HKLM:\\HARDWARE\\ACPI\\RSDT,    if ($xen -match \"xen\"),    {,    $xenvm = $true,
},    },  ,    if (!$xenvm),    {,    $xen = Get-ChildItem HKLM:\\SYSTEM\\ControlSet001\\Services,    if (($xen -match \"xenevtchn\") -or ($xen -match \"xennet\") -or ($xen -match
\"xennet6\") -or ($xen -match \"xensvc\") -or ($xen -match \"xenvdb\")),    {,    $xenvm = $true,    },    },    if ($xenvm),    {,  ,    \"This is
a Xen Machine.\",  ,    },    #QEMU,    $qemu = Get-Childitem hklm:\\hardware\\devicemap\\scsi -recurse | gp -Name identifier,    if ($qemu -match \"qemu\"),    {,  ,    $qemuvm =
$true,  ,    },  ,    if (!$qemuvm),    {,    $qemu = Get-ItemProperty hklm:HARDWARE\\DESCRIPTION\\System\\CentralProcessor\\0 -Name ProcessorNameString,    if ($qemu -match
\"qemu\"),    {,    $qemuvm = $true,    },    }  ,    if ($qemuvm),    {,  ,    \"This is a Qemu machine.\",  ,    },
}"},{"@Name":"ScriptBlockId","#text":"7c52679e-db36-49f4-87e0-675a6b23913e"},{"@Name":"Path"}]}}
```

Answer: Check-VM

Which registry key did the above script query to retrieve service details for virtualization detection?



```
HKLM:\\SYSTEM\\ControlSet001\\Services
```

Answer: HKLM:\SYSTEM\ControlSet001\Services

The VM detection script can also identify VirtualBox. Which processes is it comparing to determine if the system is running VirtualBox?

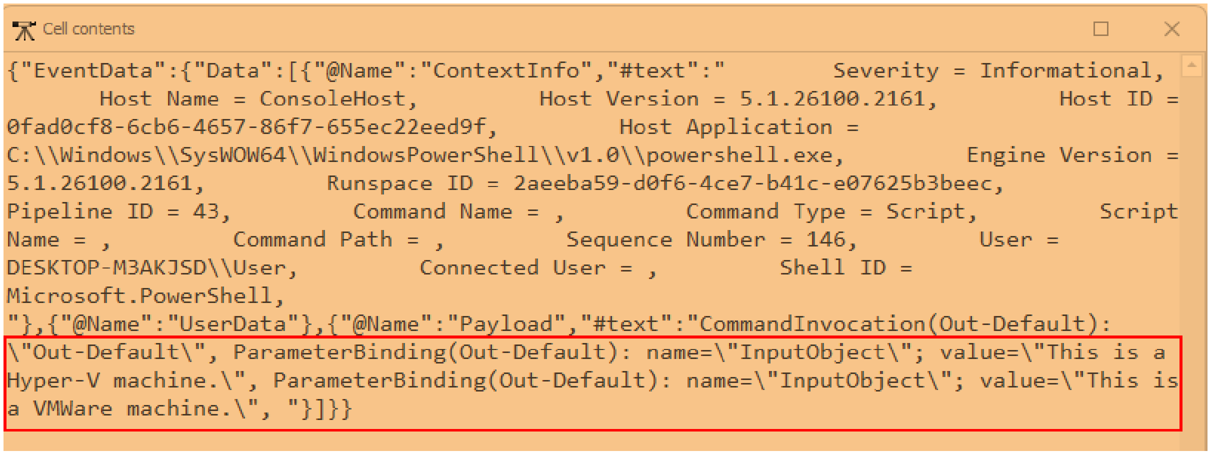
Within the script we can find the following:

- $vb = Get-Process
- if (($vb -eq "vboxservice.exe") -or ($vb -match "vboxtray.exe"))

Answer: vboxservice.exe, vboxtray.exe

The VM detection script prints any detection with the prefix 'This is a'. Which two virtualization platforms did the script detect?

If you filter for Event ID 4103, you can find what two virtualisation platforms were detected by the script:



```
{"EventData":{"Data":[{"@Name":"ContextInfo","#text":"        Severity = Informational,
       Host Name = ConsoleHost,         Host Version = 5.1.26100.2161,         Host ID =
0fad0cf8-6cb6-4657-86f7-655ec22eed9f,         Host Application =
C:\\Windows\\SysWOW64\\WindowsPowerShell\\v1.0\\powershell.exe,         Engine Version =
5.1.26100.2161,         Runspace ID = 2aeeba59-d0f6-4ce7-b41c-e07625b3beec,
Pipeline ID = 43,         Command Name = ,         Command Type = Script,         Script
Name = ,         Command Path = ,         Sequence Number = 146,         User =
DESKTOP-M3AKJSD\\User,         Connected User = ,         Shell ID =
Microsoft.PowerShell,
"},{"@Name":"UserData"},{"@Name":"Payload","#text":"CommandInvocation(Out-Default):
\"Out-Default\", ParameterBinding(Out-Default): name=\"InputObject\"; value=\"This is a
Hyper-V machine.\", ParameterBinding(Out-Default): name=\"InputObject\"; value=\"This is
a VMWare machine.\", "}]}}
```

Answer: Hyper-V, Vmware