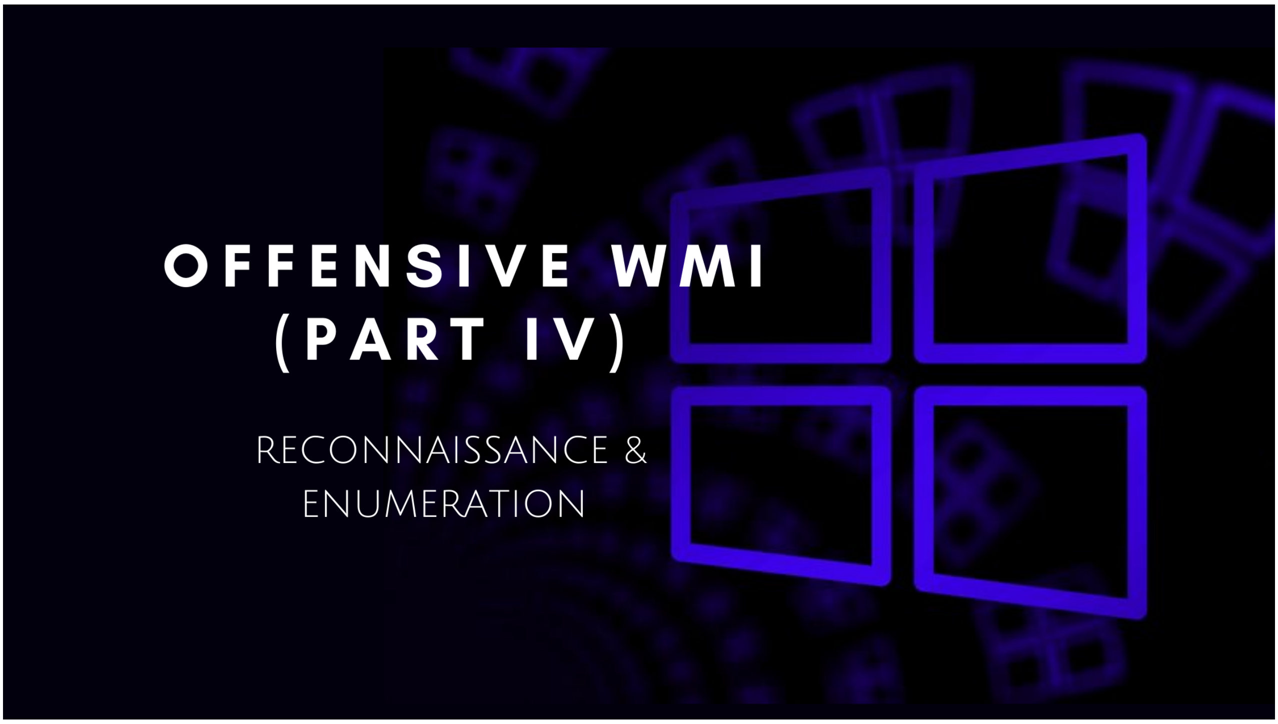
Offensive WMI - Reconnaissance & Enumeration (Part 4)

0xinfection.github.io/posts/wmi-recon-enum

October 2, 2021

2021-10-02



This is the fourth part of the “Offensive WMI” series which will focus a bit more on information gathering and enumeration. WMI provides a plethora of classes from which we can enumerate a lot of stuff. So let’s dive in without wasting any more time.

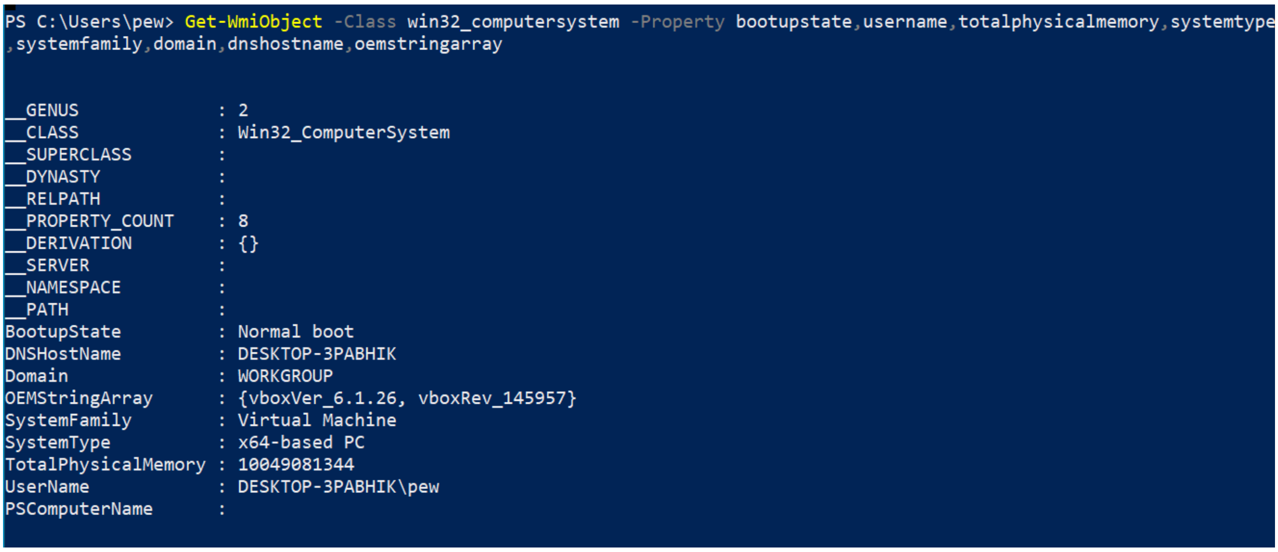
Gathering basic information

In our previous blogs, we have already seen a lot of classes that provide us with valuable information about a system, e.g. `StdRegProv` for the registry, `Win32_Process` for processes running on the system, `Win32_Bios` for BIOS information etc. Let us try exploring a bit more.

Host/OS info

Getting to know the host/OS is a very basic step when it comes to reconnaissance. WMI has two classes, namely `Win32_OperatingSystem` and `Win32_ComputerSystem` that provides us with the relevant information. For our example, we’ll be filtering out junk to print only the necessary information needed.

```
Get-WmiObject -Class win32_computersystem -Property
bootupstate,username,totalphysicalmemory,systemtype,systemfamily,domain,dnshostnam
e,oemstringarray
```

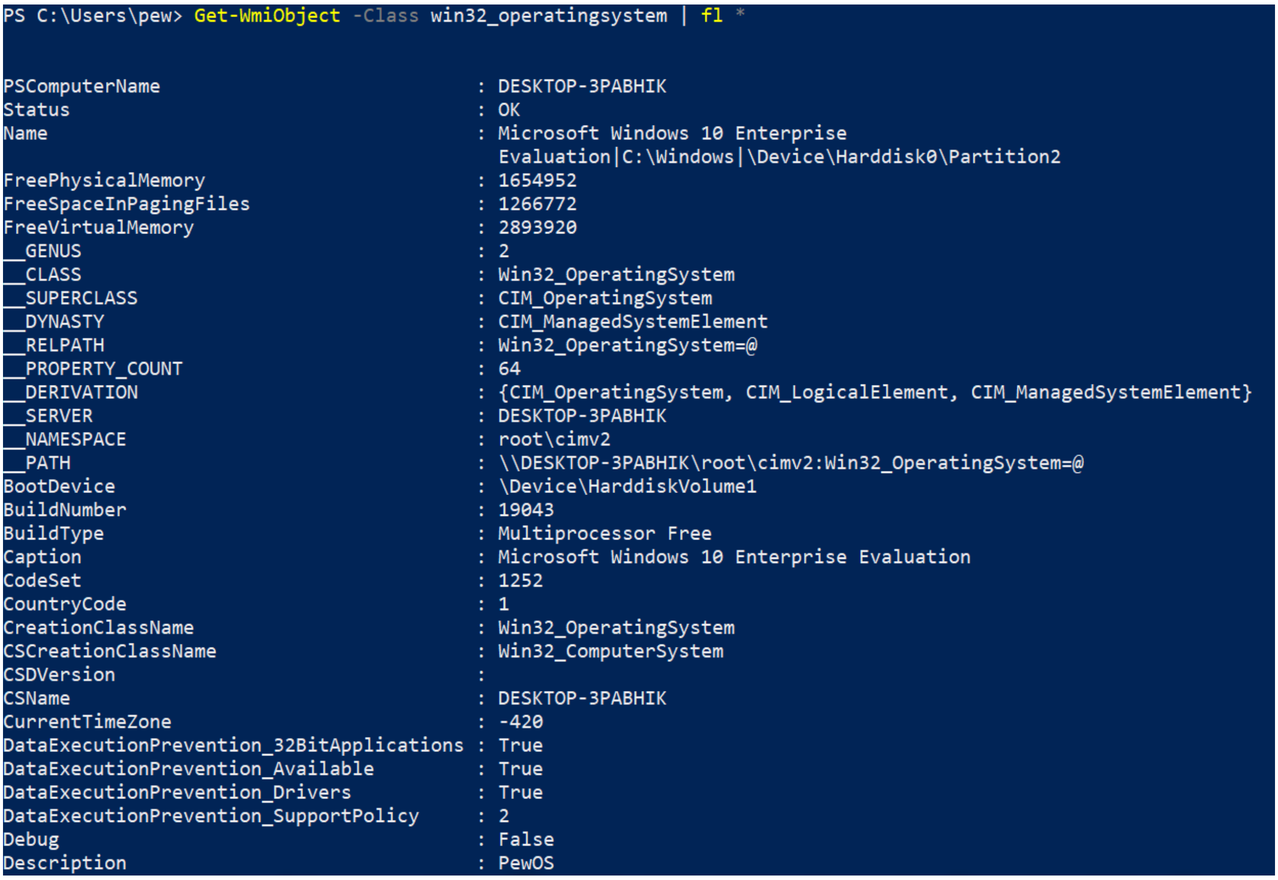


```
PS C:\Users\pew> Get-WmiObject -Class win32_computersystem -Property bootupstate,username,totalphysicalmemory,systemtype
,systemfamily,domain,dnshostname,oemstringarray


__GENUS             : 2
__CLASS             : Win32_ComputerSystem
__SUPERCLASS        :
__DYNASTY           :
__RELPATH           :
__PROPERTY_COUNT    : 8
__DERIVATION        : {}
__SERVER            :
__NAMESPACE         :
__PATH              :
BootupState         : Normal boot
DNSHostName         : DESKTOP-3PABHIK
Domain              : WORKGROUP
OEMStringArray      : {vboxVer_6.1.26, vboxRev_145957}
SystemFamily        : Virtual Machine
SystemType          : x64-based PC
TotalPhysicalMemory : 10049081344
UserName            : DESKTOP-3PABHIK\pew
PSComputerName      :
```

So most of the information that we have now helps us in one major thing – figuring out whether we are in an emulated environment. The bootup state for our current run indicates that the system wasn't booted in fail-safe mode. We can also see that our current user is `pew` and the box is not a part of any AD domain. We also get the processor architecture and the RAM available for us to use. This is useful for VM detection, for example – if the number of logical processors is less than 4 and the RAM available is below 2 Gigs, then the probability of the box being a VM is high. Of course, the same data is given away by the `SystemFamily` and the `OEMStringArray` properties, but in controlled environments, there might be other indicators as well.

The other class `Win32_OperatingSystem` too provides us with a lot of useful info:

```
Get-WmiObject -Class win32_operatingsystem | fl *
```

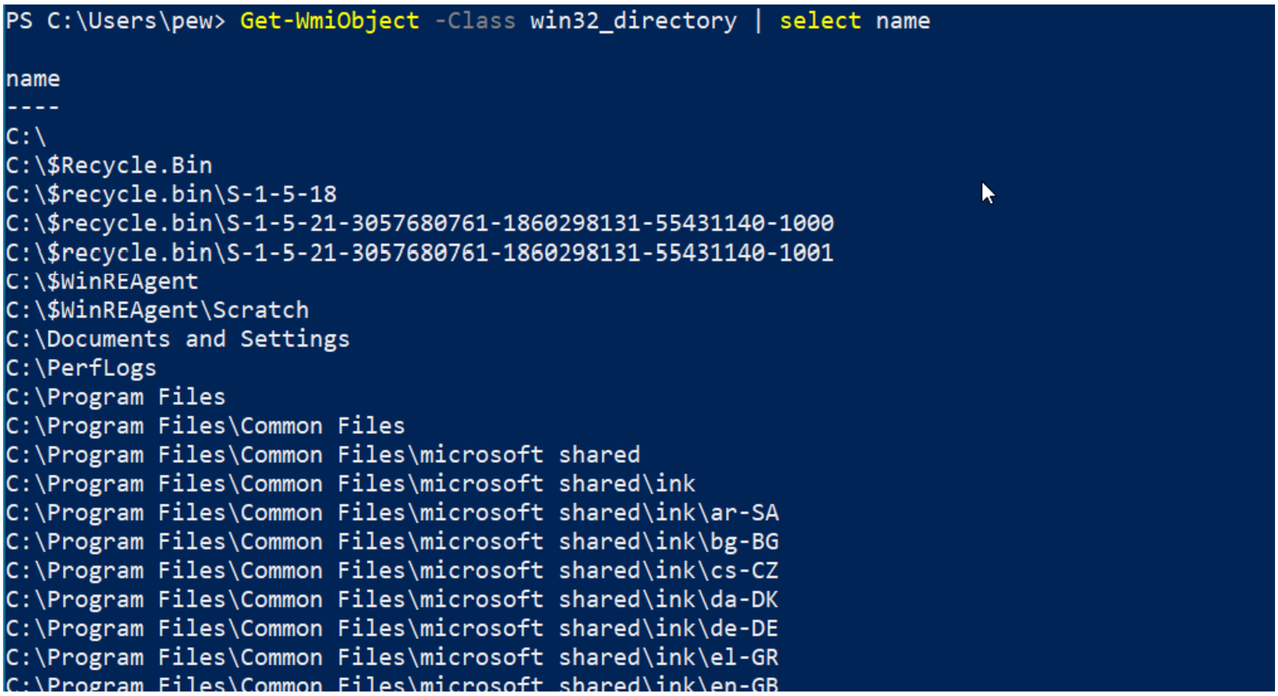


```
PS C:\Users\pew> Get-WmiObject -Class win32_operatingsystem | fl *


PSComputerName                            : DESKTOP-3PABHIK
Status                                    : OK
Name                                      : Microsoft Windows 10 Enterprise
                                            Evaluation|C:\Windows|\Device\Harddisk0\Partition2
FreePhysicalMemory                        : 1654952
FreeSpaceInPagingFiles                    : 1266772
FreeVirtualMemory                         : 2893920
__GENUS                                   : 2
__CLASS                                   : Win32_OperatingSystem
__SUPERCLASS                              : CIM_OperatingSystem
__DYNASTY                                 : CIM_ManagedSystemElement
__RELPATH                                 : Win32_OperatingSystem=@
__PROPERTY_COUNT                          : 64
__DERIVATION                              : {CIM_OperatingSystem, CIM_LogicalElement, CIM_ManagedSystemElement}
__SERVER                                  : DESKTOP-3PABHIK
__NAMESPACE                               : root\cimv2
__PATH                                    : \\DESKTOP-3PABHIK\root\cimv2:Win32_OperatingSystem=@
BootDevice                                : \Device\HarddiskVolume1
BuildNumber                               : 19043
BuildType                                 : Multiprocessor Free
Caption                                   : Microsoft Windows 10 Enterprise Evaluation
CodeSet                                   : 1252
CountryCode                               : 1
CreationClassName                         : Win32_OperatingSystem
CSCreationClassName                       : Win32_ComputerSystem
CSDVersion                                :
CSName                                    : DESKTOP-3PABHIK
CurrentTimeZone                           : -420
DataExecutionPrevention_32BitApplications : True
DataExecutionPrevention_Available         : True
DataExecutionPrevention_Drivers           : True
DataExecutionPrevention_SupportPolicy     : 2
Debug                                     : False
Description                               : PewOS
```

Directory listing

Listing files on a system is a very fundamental operation. WMI has a class called `Win32_Directory` that helps in listing the files. Alternatively, there is another class named `CIM_DataFile` that can also be utilized to achieve the same.

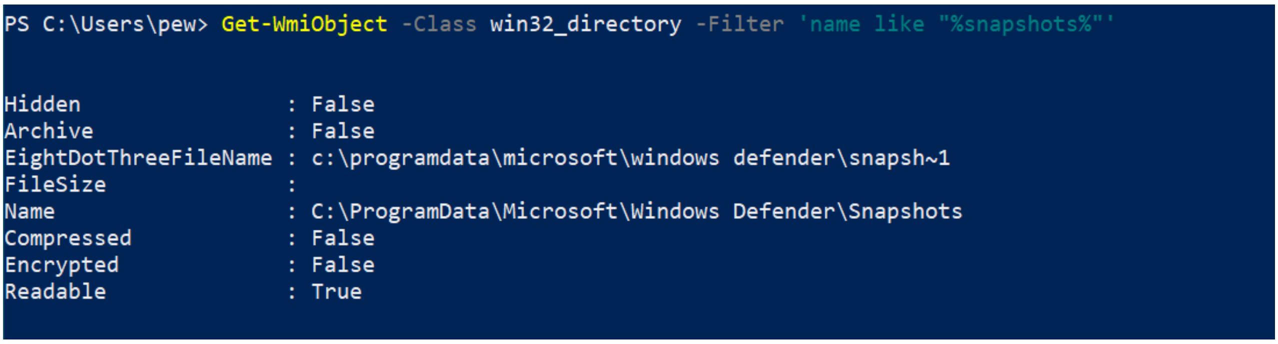
```
Get-WmiObject -Class win32_directory
```



```
PS C:\Users\pew> Get-WmiObject -Class win32_directory | select name

name
----
C:\
C:\$Recycle.Bin
C:\$recycle.bin\S-1-5-18
C:\$recycle.bin\S-1-5-21-3057680761-1860298131-55431140-1000
C:\$recycle.bin\S-1-5-21-3057680761-1860298131-55431140-1001
C:\$WinREAgent
C:\$WinREAgent\Scratch
C:\Documents and Settings
C:\PerfLogs
C:\Program Files
C:\Program Files\Common Files
C:\Program Files\Common Files\microsoft shared
C:\Program Files\Common Files\microsoft shared\ink
C:\Program Files\Common Files\microsoft shared\ink\ar-SA
C:\Program Files\Common Files\microsoft shared\ink\bg-BG
C:\Program Files\Common Files\microsoft shared\ink\cs-CZ
C:\Program Files\Common Files\microsoft shared\ink\da-DK
C:\Program Files\Common Files\microsoft shared\ink\de-DE
C:\Program Files\Common Files\microsoft shared\ink\el-GR
C:\Program Files\Common Files\microsoft shared\ink\en-GB
```

Often searching for file patterns using wildcards is helpful. We can make use of the `-Filter` argument of the cmdlet to achieve something similar. Let's say we're interested in directory paths that have a folder called `snapshots`. Querying it with WMI would look like this:

```
Get-WmiObject -Class win32_directory -Filter 'name LIKE "%snapshots%"'
```

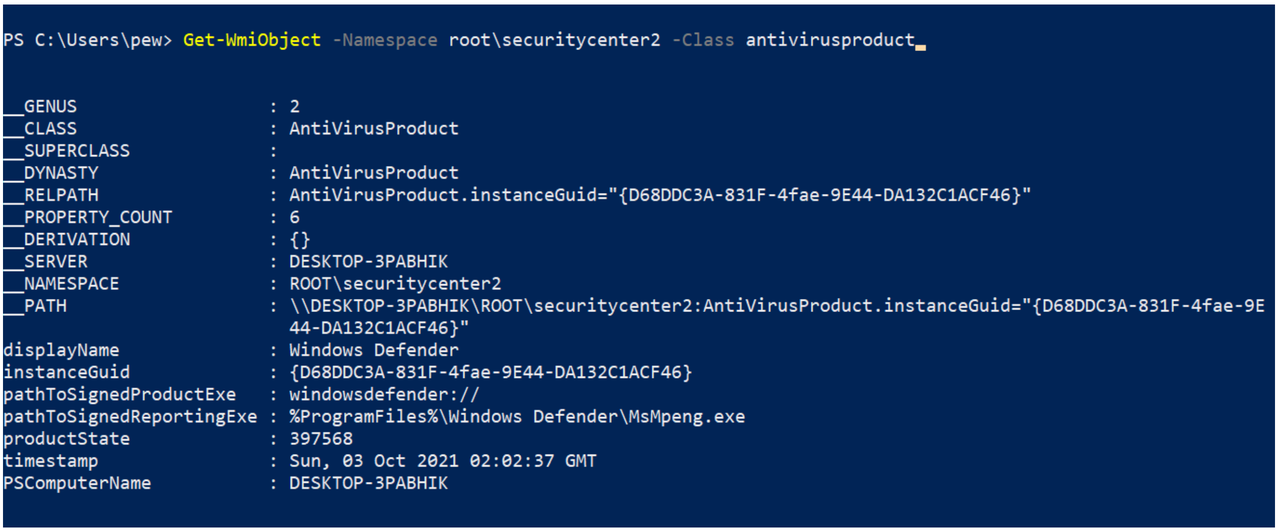


```
PS C:\Users\pew> Get-WmiObject -Class win32_directory -Filter 'name like "%snapshots%"'


Hidden                : False
Archive               : False
EightDotThreeFileName : c:\programdata\microsoft\windows defender\snapsh~1
FileSize              :
Name                  : C:\ProgramData\Microsoft\Windows Defender\Snapshots
Compressed            : False
Encrypted             : False
Readable              : True
```

AV product

One of the first steps when it comes to recon is to enumerate what kind of product is providing security to a system. WMI provides a class called `AntiVirusProduct` under the `root\SecurityCenter2` namespace that contains information about the AV installed on the system. In my case, it's the default Windows Defender.

```
Get-WmiObject -Namespace root\securitycenter2 -Class antivirusproduct
```



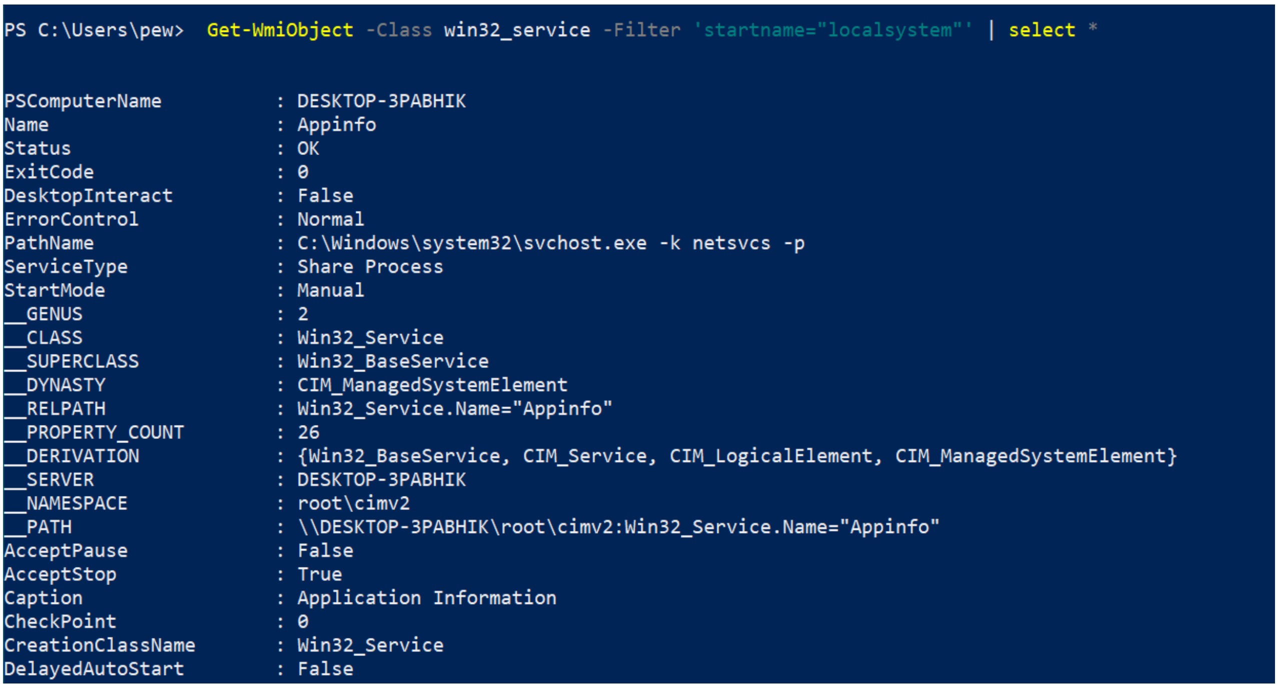
```
PS C:\Users\pew> Get-WmiObject -Namespace root\securitycenter2 -Class antivirusproduct_


__GENUS                  : 2
__CLASS                  : AntiVirusProduct
__SUPERCLASS             :
__DYNASTY                : AntiVirusProduct
__RELPATH                : AntiVirusProduct.instanceGuid="{D68DDC3A-831F-4fae-9E44-DA132C1ACF46}"
__PROPERTY_COUNT         : 6
__DERIVATION             : {}
__SERVER                 : DESKTOP-3PABHIK
__NAMESPACE              : ROOT\securitycenter2
__PATH                   : \\DESKTOP-3PABHIK\ROOT\securitycenter2:AntiVirusProduct.instanceGuid="{D68DDC3A-831F-4fae-9E
                           44-DA132C1ACF46}"
displayName              : Windows Defender
instanceGuid             : {D68DDC3A-831F-4fae-9E44-DA132C1ACF46}
pathToSignedProductExe   : windowsdefender://
pathToSignedReportingExe : %ProgramFiles%\Windows Defender\MsMpeng.exe
productState             : 397568
timestamp                : Sun, 03 Oct 2021 02:02:37 GMT
PSComputerName           : DESKTOP-3PABHIK
```

Services

Services on a Windows system are similar to Unix daemons, or simply non-UI processes running in the background. This is useful information when it comes to privilege escalation, especially, in cases where there is a service created by `SYSTEM` with weak file permissions.

To list the services, we need to make use of the `Win32_Service` class. For our example, we'll only print those services which are initiated by the `LocalSystem` (or the `NT Authority\System`). Note the usage of the `select` Powershell utility that expands the output significantly as compared to without it.

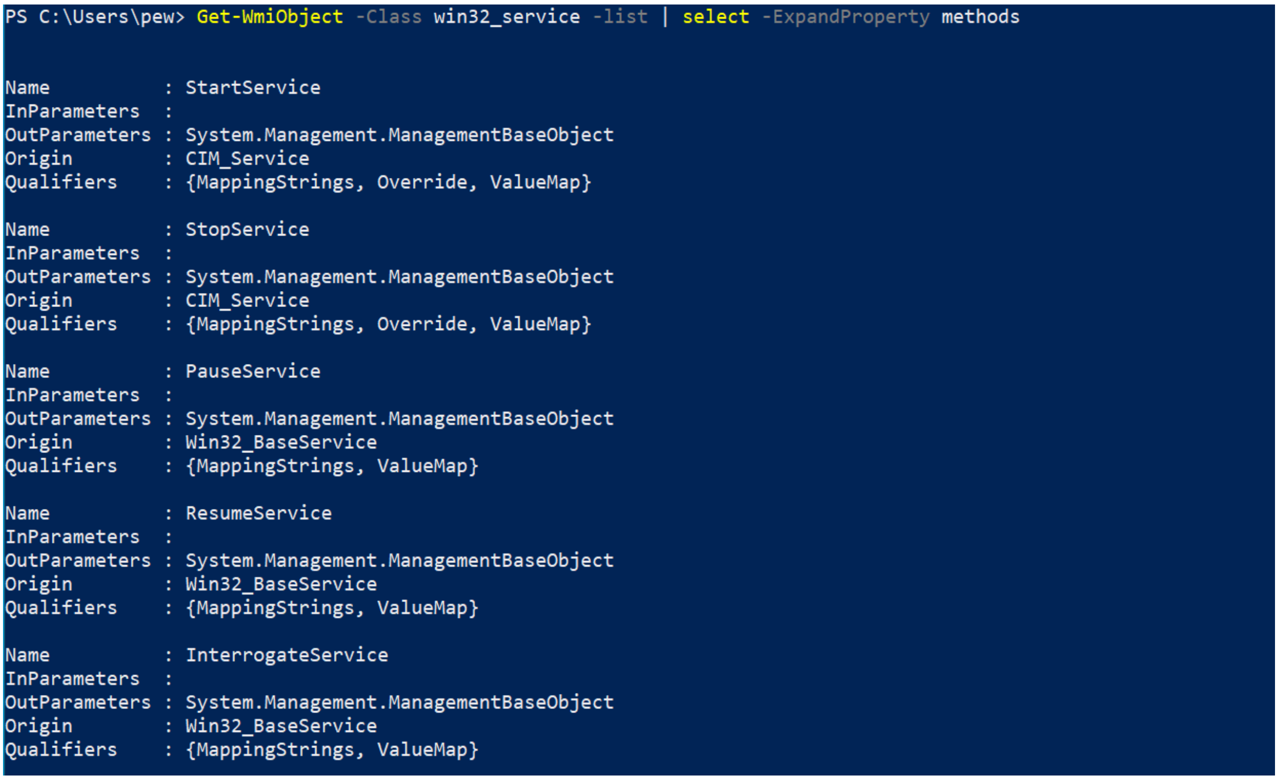
```
Get-WmiObject -Class win32_service -Filter 'startname="localsystem"' | select *
```



```
PS C:\Users\pew>  Get-WmiObject -Class win32_service -Filter 'startname="localsystem"' | select *


PSComputerName          : DESKTOP-3PABHIK
Name                    : Appinfo
Status                  : OK
ExitCode                : 0
DesktopInteract         : False
ErrorControl            : Normal
PathName                : C:\Windows\system32\svchost.exe -k netsvcs -p
ServiceType             : Share Process
StartMode               : Manual
__GENUS                 : 2
__CLASS                 : Win32_Service
__SUPERCLASS            : Win32_BaseService
__DYNASTY               : CIM_ManagedSystemElement
__RELPATH               : Win32_Service.Name="Appinfo"
__PROPERTY_COUNT        : 26
__DERIVATION            : {Win32_BaseService, CIM_Service, CIM_LogicalElement, CIM_ManagedSystemElement}
__SERVER                : DESKTOP-3PABHIK
__NAMESPACE             : root\cimv2
__PATH                  : \\DESKTOP-3PABHIK\root\cimv2:Win32_Service.Name="Appinfo"
AcceptPause             : False
AcceptStop              : True
Caption                 : Application Information
CheckPoint              : 0
CreationClassName       : Win32_Service
DelayedAutoStart        : False
```

WMI also provides several methods when it comes to interacting with services. They allow creation, deletion, starting, stopping, resuming, updating and a lot of other capabilities to manipulate the services. To list the methods available under the `Win32_Service` class, we can use the following command:

```
Get-WmiObject -Class win32_service -List | select -ExpandProperty methods
```

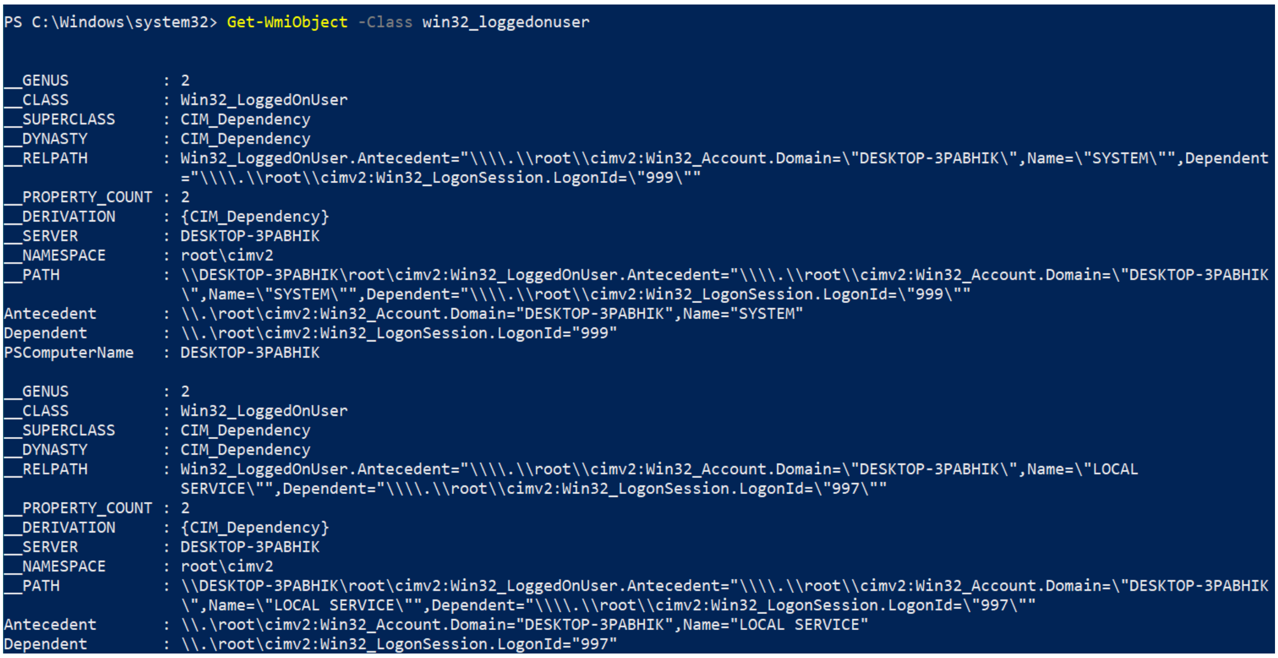


```
PS C:\Users\pew> Get-WmiObject -Class win32_service -list | select -ExpandProperty methods


Name          : StartService
InParameters  :
OutParameters : System.Management.ManagementBaseObject
Origin        : CIM_Service
Qualifiers    : {MappingStrings, Override, ValueMap}

Name          : StopService
InParameters  :
OutParameters : System.Management.ManagementBaseObject
Origin        : CIM_Service
Qualifiers    : {MappingStrings, Override, ValueMap}

Name          : PauseService
InParameters  :
OutParameters : System.Management.ManagementBaseObject
Origin        : Win32_BaseService
Qualifiers    : {MappingStrings, ValueMap}

Name          : ResumeService
InParameters  :
OutParameters : System.Management.ManagementBaseObject
Origin        : Win32_BaseService
Qualifiers    : {MappingStrings, ValueMap}

Name          : InterrogateService
InParameters  :
OutParameters : System.Management.ManagementBaseObject
Origin        : Win32_BaseService
Qualifiers    : {MappingStrings, ValueMap}
```

Logged-on Users

Getting the logged-on users on a system is pretty trivial. There are two classes – `Win32_LoggedOnUser` and `Win32_LogOnSession` that holds the particulars about the session and users logged onto the system. Querying the class from a privileged user gives us much more information about the logged in users:

```
Get-WmiObject -Class win32_loggedonuser
```

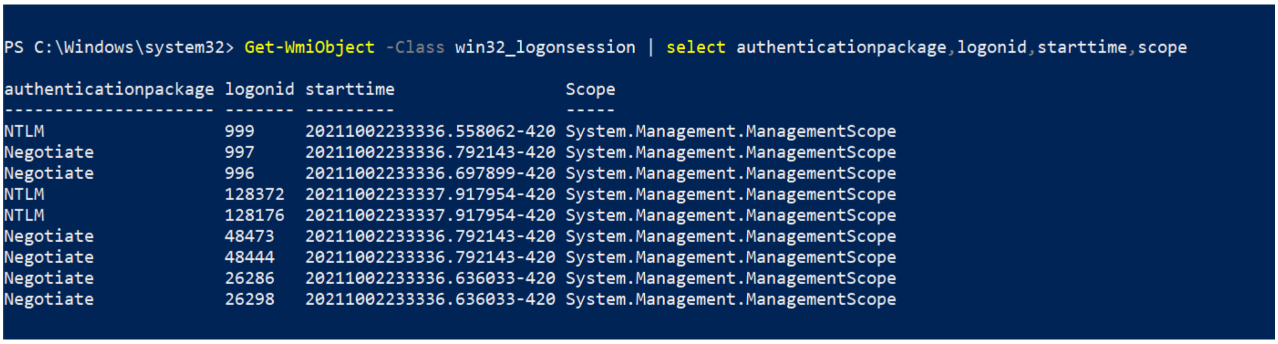


```
PS C:\Windows\system32> Get-WmiObject -Class win32_loggedonuser


__GENUS          : 2
__CLASS          : Win32_LoggedOnUser
__SUPERCLASS     : CIM_Dependency
__DYNASTY        : CIM_Dependency
__RELPATH        : Win32_LoggedOnUser.Antecedent="\\\\.\\root\\cimv2:Win32_Account.Domain=\"DESKTOP-3PABHIK\",Name=\"SYSTEM\"",Dependent
                   ="\\\\.\\root\\cimv2:Win32_LogonSession.LogonId=\"999\""
__PROPERTY_COUNT : 2
__DERIVATION     : {CIM_Dependency}
__SERVER         : DESKTOP-3PABHIK
__NAMESPACE      : root\cimv2
__PATH           : \\DESKTOP-3PABHIK\root\cimv2:Win32_LoggedOnUser.Antecedent="\\\\.\\root\\cimv2:Win32_Account.Domain=\"DESKTOP-3PABHIK
                   \",Name=\"SYSTEM\"",Dependent="\\\\.\\root\\cimv2:Win32_LogonSession.LogonId=\"999\""
Antecedent       : \\.\root\cimv2:Win32_Account.Domain="DESKTOP-3PABHIK",Name="SYSTEM"
Dependent        : \\.\root\cimv2:Win32_LogonSession.LogonId="999"
PSComputerName   : DESKTOP-3PABHIK

__GENUS          : 2
__CLASS          : Win32_LoggedOnUser
__SUPERCLASS     : CIM_Dependency
__DYNASTY        : CIM_Dependency
__RELPATH        : Win32_LoggedOnUser.Antecedent="\\\\.\\root\\cimv2:Win32_Account.Domain=\"DESKTOP-3PABHIK\",Name=\"LOCAL
                   SERVICE\"",Dependent="\\\\.\\root\\cimv2:Win32_LogonSession.LogonId=\"997\""
__PROPERTY_COUNT : 2
__DERIVATION     : {CIM_Dependency}
__SERVER         : DESKTOP-3PABHIK
__NAMESPACE      : root\cimv2
__PATH           : \\DESKTOP-3PABHIK\root\cimv2:Win32_LoggedOnUser.Antecedent="\\\\.\\root\\cimv2:Win32_Account.Domain=\"DESKTOP-3PABHIK
                   \",Name=\"LOCAL SERVICE\"",Dependent="\\\\.\\root\\cimv2:Win32_LogonSession.LogonId=\"997\""
Antecedent       : \\.\root\cimv2:Win32_Account.Domain="DESKTOP-3PABHIK",Name="LOCAL SERVICE"
Dependent        : \\.\root\cimv2:Win32_LogonSession.LogonId="997"
```

From the above, we can see that each logged-in user has an LUID (locally-unique identifier). Some LUIDs are predefined. For example, the LUID for the System account's logon session is always 0x3e7 (999 decimal), the LUID for Network Service's session is 0x3e4 (996), and Local Service's is 0x3e5 (997). Most other LUIDs are randomly generated.

Each logged-on user defines its dependents via the `Dependent` property. We can get a list of logon IDs, the authentication type, start time and scope of every session using the `Win32_LogOnSession` class:

```
Get-WmiObject -Class win32_logonsession | select
authenticationpackage,logonid,starttime,scope
```

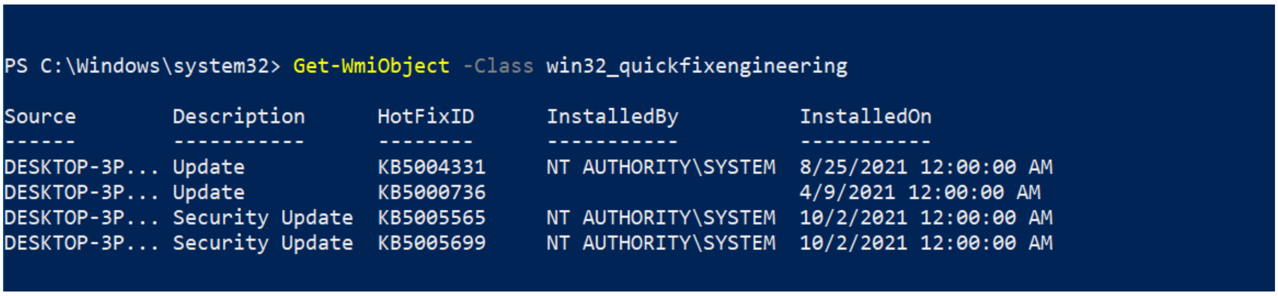


```
PS C:\Windows\system32> Get-WmiObject -Class win32_logonsession | select authenticationpackage,logonid,starttime,scope

authenticationpackage logonid starttime                 Scope
--------------------- ------- ---------                 -----
NTLM                  999     20211002233336.558062-420 System.Management.ManagementScope
Negotiate             997     20211002233336.792143-420 System.Management.ManagementScope
Negotiate             996     20211002233336.697899-420 System.Management.ManagementScope
NTLM                  128372  20211002233337.917954-420 System.Management.ManagementScope
NTLM                  128176  20211002233337.917954-420 System.Management.ManagementScope
Negotiate             48473   20211002233336.792143-420 System.Management.ManagementScope
Negotiate             48444   20211002233336.792143-420 System.Management.ManagementScope
Negotiate             26286   20211002233336.636033-420 System.Management.ManagementScope
Negotiate             26298   20211002233336.636033-420 System.Management.ManagementScope
```

Installed patches

It’s often useful to enumerate the updates/patches installed on a machine. If the system is missing important patches, that might open up an easy possibility to compromise the system in one quick shot. WMI has a class known as `Win32_QuickFixEngineering` which contains info about the installed updates and security patches. Querying the class is a piece of cake:

```
Get-WmiObject -Class win32_quickfixengineering
```

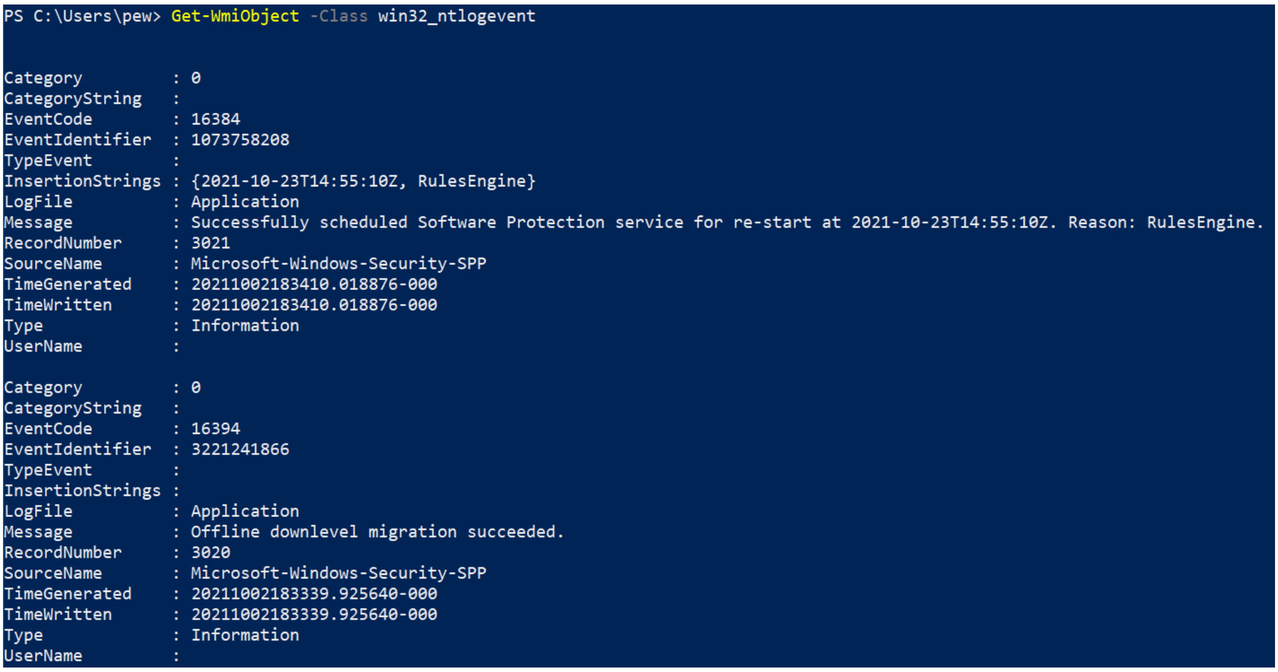


```
PS C:\Windows\system32> Get-WmiObject -Class win32_quickfixengineering

Source        Description      HotFixID      InstalledBy          InstalledOn
------        -----------      --------      -----------          -----------
DESKTOP-3P... Update           KB5004331     NT AUTHORITY\SYSTEM  8/25/2021 12:00:00 AM
DESKTOP-3P... Update           KB5000736                          4/9/2021 12:00:00 AM
DESKTOP-3P... Security Update  KB5005565     NT AUTHORITY\SYSTEM  10/2/2021 12:00:00 AM
DESKTOP-3P... Security Update  KB5005699     NT AUTHORITY\SYSTEM  10/2/2021 12:00:00 AM
```

Event logs

The class `Win32_NtLogEvent` gives us useful data about the events logs captured by the system. We can query it like the following:

```
Get-WmiObject -Class win32_ntlogevent
```



```
PS C:\Users\pew> Get-WmiObject -Class win32_ntlogevent


Category         : 0
CategoryString   :
EventCode        : 16384
EventIdentifier  : 1073758208
TypeEvent        :
InsertionStrings : {2021-10-23T14:55:10Z, RulesEngine}
LogFile          : Application
Message          : Successfully scheduled Software Protection service for re-start at 2021-10-23T14:55:10Z. Reason: RulesEngine.
RecordNumber     : 3021
SourceName       : Microsoft-Windows-Security-SPP
TimeGenerated    : 20211002183410.018876-000
TimeWritten      : 20211002183410.018876-000
Type             : Information
UserName         :

Category         : 0
CategoryString   :
EventCode        : 16394
EventIdentifier  : 3221241866
TypeEvent        :
InsertionStrings :
LogFile          : Application
Message          : Offline downlevel migration succeeded.
RecordNumber     : 3020
SourceName       : Microsoft-Windows-Security-SPP
TimeGenerated    : 20211002183339.925640-000
TimeWritten      : 20211002183339.925640-000
Type             : Information
UserName         :
```

Each log entry carries details like time, the source generating the event, severity and a message. The severity is indicated by the `Type` property in the output. Talking about event types, there are five different levels which are depicted in the table below:

Value	Meaning
1	Error
2	Warning

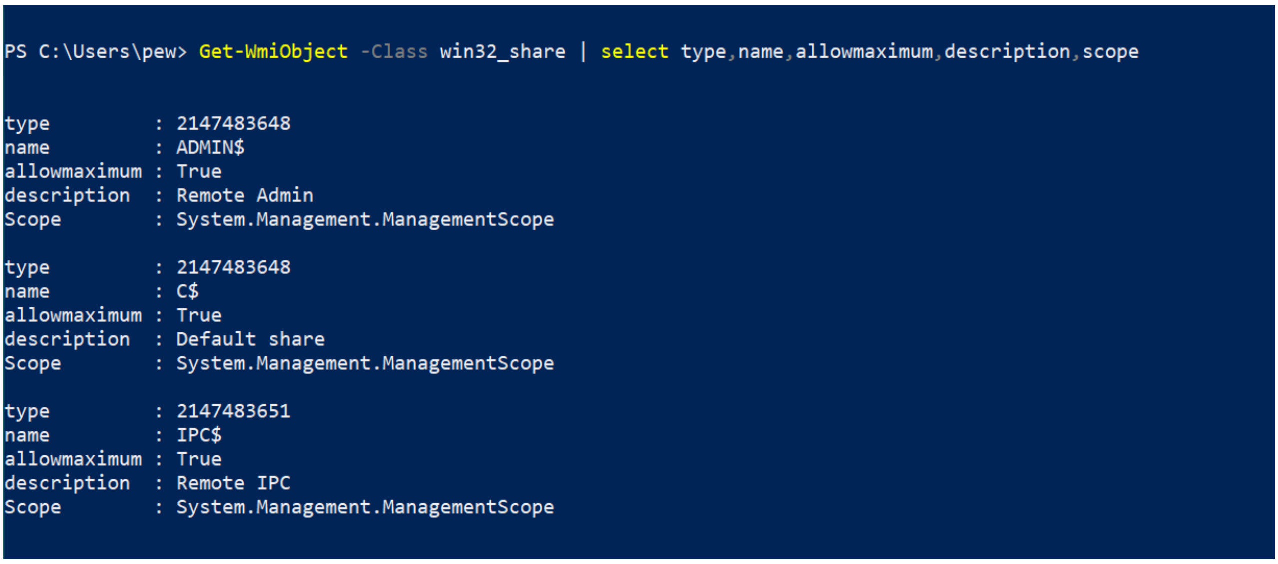
Value	Meaning
4	Information
8	Security Audit Success
16	Security Audit Failure

We can, of course, make use of the `-Filter` switch to search for specific event types.

Shares

The `Win32_Share` class represents a shared resource on a system. This may be a disk drive, printer, interprocess communication, or other sharable devices. In enterprise networks, there are usually a lot of shares that might come in handy during a penetration test. Let us see how we can enumerate the available shares:

```
Get-WmiObject -Class win32_share | select type,name,allowmaximum,description,scope
```



```
PS C:\Users\pew> Get-WmiObject -Class win32_share | select type,name,allowmaximum,description,scope


type         : 2147483648
name         : ADMIN$
allowmaximum : True
description  : Remote Admin
Scope        : System.Management.ManagementScope

type         : 2147483648
name         : C$
allowmaximum : True
description  : Default share
Scope        : System.Management.ManagementScope

type         : 2147483651
name         : IPC$
allowmaximum : True
description  : Remote IPC
Scope        : System.Management.ManagementScope
```

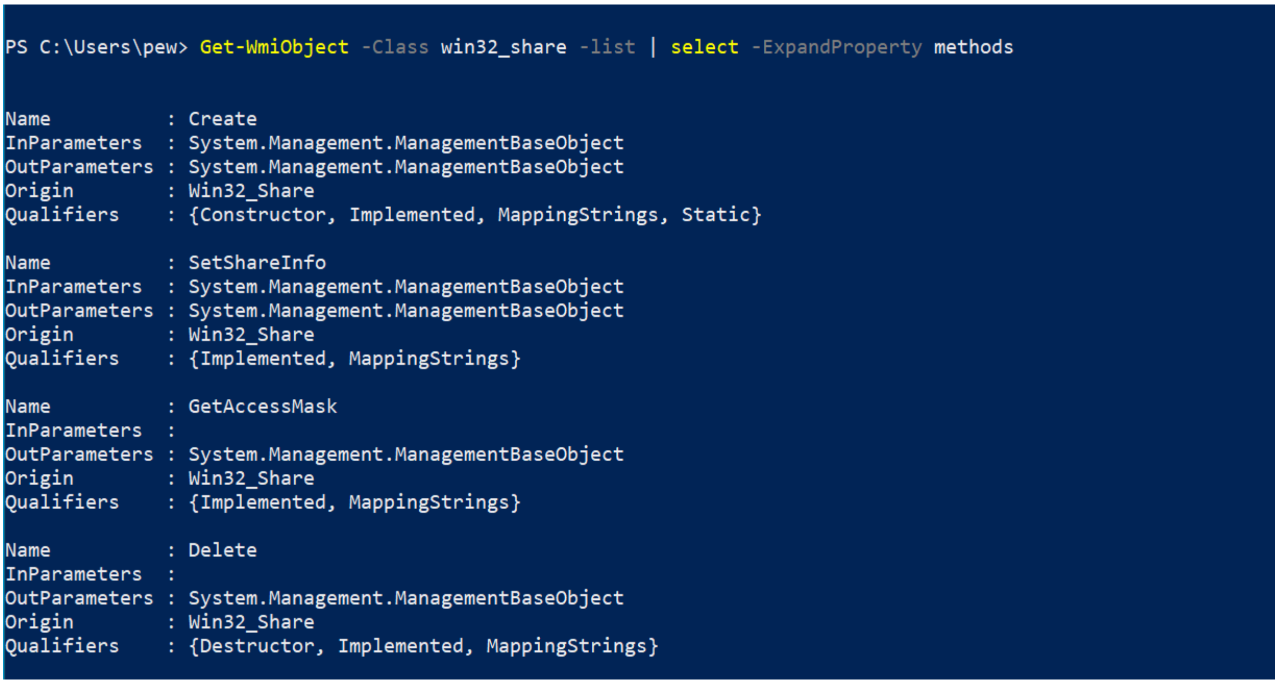
In the above example, we filtered only the required useful information using `select`. We have the share type, name, concurrent access permission, description and scope of every available share from the output of the command. Once again, types are constants that define the type of resources being shared:

Value	Meaning
0	Disk Drive
1	Print Queue
2	Device
3	IPC
2147483648	Disk Drive Admin

Value	Meaning
2147483649	Print Queue Admin
2147483650	Device Admin
2147483651	IPC Admin

The `AllowMaximum` is a boolean property indicating whether concurrent access to the resource has been restricted or not. If the value is set to `True`, then there is no restriction on the shared access, which otherwise might indicate that there is something sensitive in the resource, or better might have monitoring for clients accessing the share.

WMI also provides methods like `Create`, `SetShareInfo` and `Delete` for creating, updating and deleting shares.

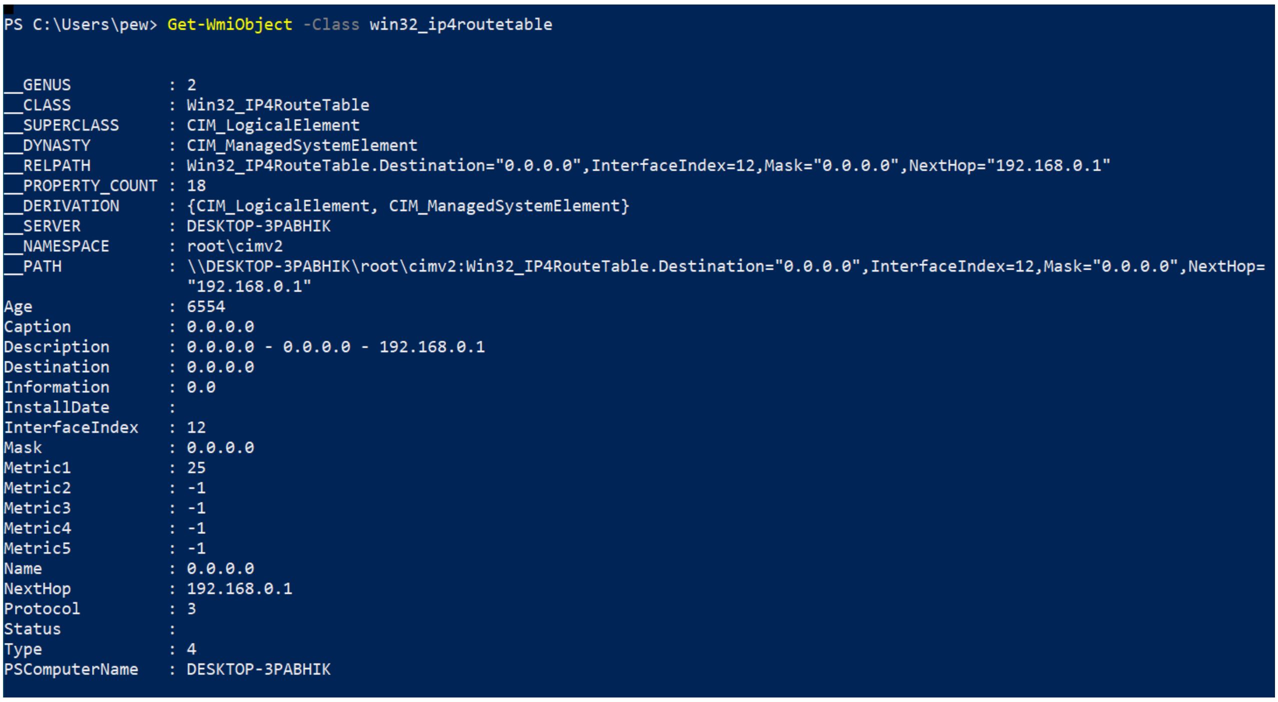


```
PS C:\Users\pew> Get-WmiObject -Class win32_share -list | select -ExpandProperty methods


Name          : Create
InParameters  : System.Management.ManagementBaseObject
OutParameters : System.Management.ManagementBaseObject
Origin        : Win32_Share
Qualifiers    : {Constructor, Implemented, MappingStrings, Static}

Name          : SetShareInfo
InParameters  : System.Management.ManagementBaseObject
OutParameters : System.Management.ManagementBaseObject
Origin        : Win32_Share
Qualifiers    : {Implemented, MappingStrings}

Name          : GetAccessMask
InParameters  :
OutParameters : System.Management.ManagementBaseObject
Origin        : Win32_Share
Qualifiers    : {Implemented, MappingStrings}

Name          : Delete
InParameters  :
OutParameters : System.Management.ManagementBaseObject
Origin        : Win32_Share
Qualifiers    : {Destructor, Implemented, MappingStrings}
```

Network info

Network information is provided by the `Win32_IP4RouteTable` class. This gives us details similar to the `ipconfig` command but in a much more detailed fashion.

```
Get-WmiObject -Class win32_ip4routetable
```



```
PS C:\Users\pew> Get-WmiObject -Class win32_ip4routetable


__GENUS          : 2
__CLASS          : Win32_IP4RouteTable
__SUPERCLASS     : CIM_LogicalElement
__DYNASTY        : CIM_ManagedSystemElement
__RELPATH        : Win32_IP4RouteTable.Destination="0.0.0.0",InterfaceIndex=12,Mask="0.0.0.0",NextHop="192.168.0.1"
__PROPERTY_COUNT : 18
__DERIVATION     : {CIM_LogicalElement, CIM_ManagedSystemElement}
__SERVER         : DESKTOP-3PABHIK
__NAMESPACE      : root\cimv2
__PATH           : \\DESKTOP-3PABHIK\root\cimv2:Win32_IP4RouteTable.Destination="0.0.0.0",InterfaceIndex=12,Mask="0.0.0.0",NextHop=
                   "192.168.0.1"
Age              : 6554
Caption          : 0.0.0.0
Description      : 0.0.0.0 - 0.0.0.0 - 192.168.0.1
Destination      : 0.0.0.0
Information      : 0.0
InstallDate      :
InterfaceIndex   : 12
Mask             : 0.0.0.0
Metric1          : 25
Metric2          : -1
Metric3          : -1
Metric4          : -1
Metric5          : -1
Name             : 0.0.0.0
NextHop          : 192.168.0.1
Protocol         : 3
Status           :
Type             : 4
PSComputerName   : DESKTOP-3PABHIK
```

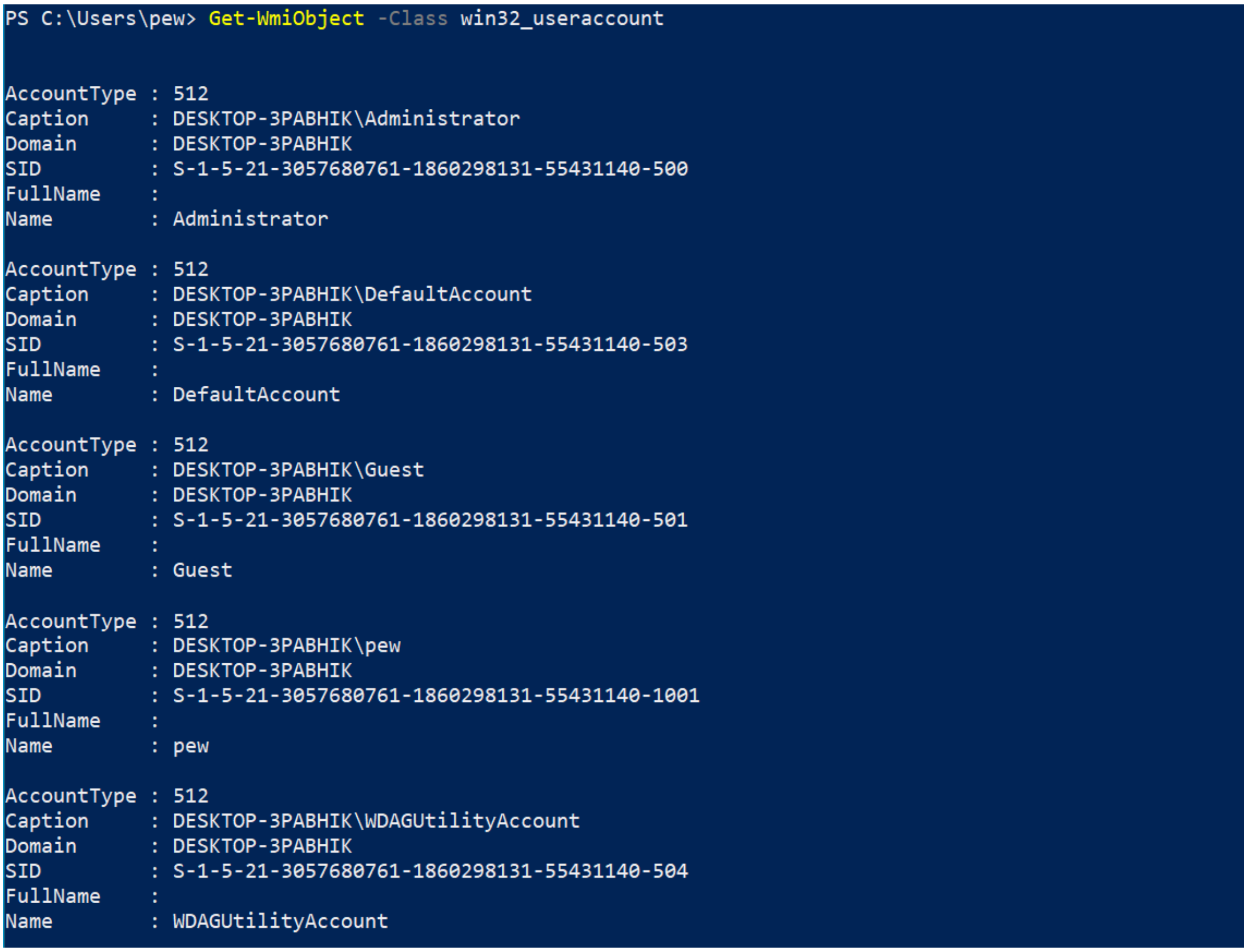
I would like to mention another useful class called `Win32_NetworkAdapter` while talking about network stuff. Querying it can give us a useful indication about the network hardware that the system has. This in-turn is useful for VM detection, for example, we can run the following queries to identify whether the system is virtualized by VMWare:

```
Get-WmiObject -Class Win32_NetworkAdapter -Filter 'name like "%vmware%"'
Get-WmiObject -Class Win32_NetworkAdapter -Filter 'manufacturer like "%vmware%"'
```

User accounts

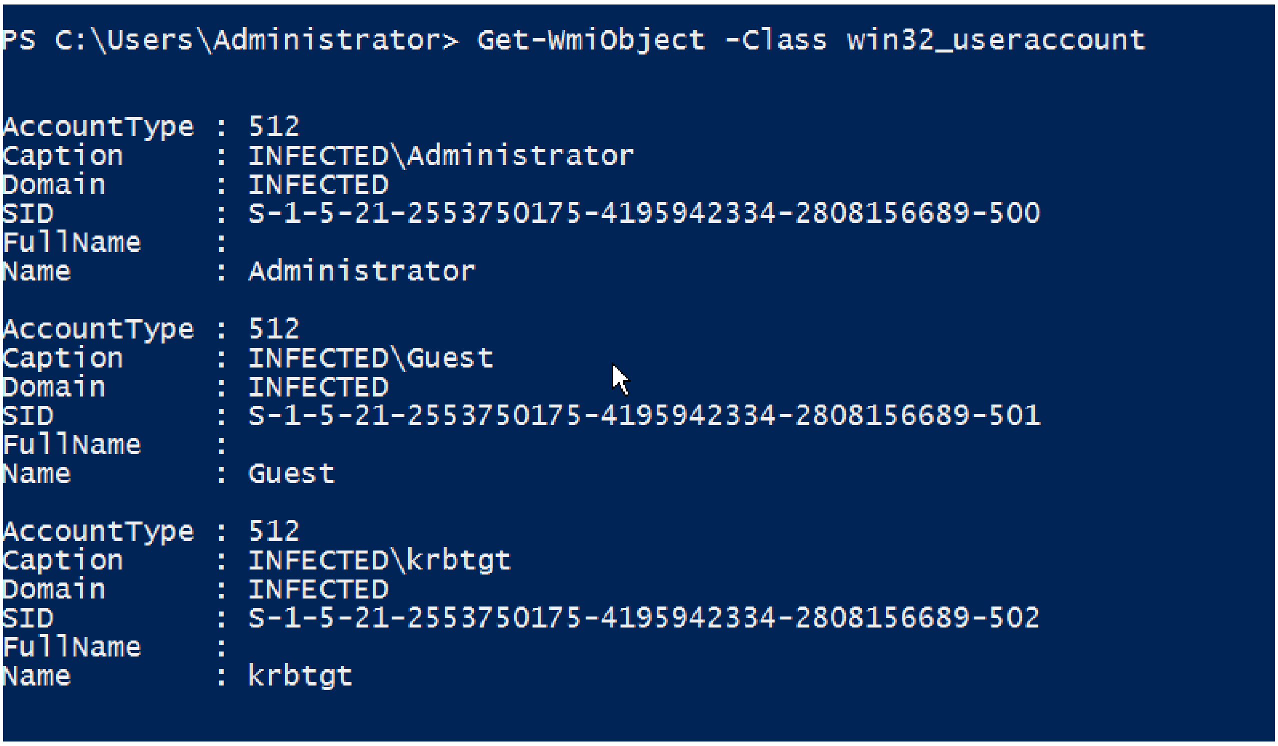
User account information is provided by the `Win32_UserAccount` class. For a default local system, there are only a few accounts, the most common ones being the administrator, guest, local users and the windows defender (`WDAGUtilityAccount`). We can get a list of users quickly via:

```
Get-WmiObject -Class win32_useraccount
```



```
PS C:\Users\pew> Get-WmiObject -Class win32_useraccount


AccountType : 512
Caption     : DESKTOP-3PABHIK\Administrator
Domain      : DESKTOP-3PABHIK
SID         : S-1-5-21-3057680761-1860298131-55431140-500
FullName    :
Name        : Administrator

AccountType : 512
Caption     : DESKTOP-3PABHIK\DefaultAccount
Domain      : DESKTOP-3PABHIK
SID         : S-1-5-21-3057680761-1860298131-55431140-503
FullName    :
Name        : DefaultAccount

AccountType : 512
Caption     : DESKTOP-3PABHIK\Guest
Domain      : DESKTOP-3PABHIK
SID         : S-1-5-21-3057680761-1860298131-55431140-501
FullName    :
Name        : Guest

AccountType : 512
Caption     : DESKTOP-3PABHIK\pew
Domain      : DESKTOP-3PABHIK
SID         : S-1-5-21-3057680761-1860298131-55431140-1001
FullName    :
Name        : pew

AccountType : 512
Caption     : DESKTOP-3PABHIK\WDAGUtilityAccount
Domain      : DESKTOP-3PABHIK
SID         : S-1-5-21-3057680761-1860298131-55431140-504
FullName    :
Name        : WDAGUtilityAccount
```

However, for a domain-joined box or domain controller, there will be several others including `krbtgt`, `sqladmin`, `webadmin`, etc. For a default Windows Server 2012 setup, there are just 3 accounts as displayed below.

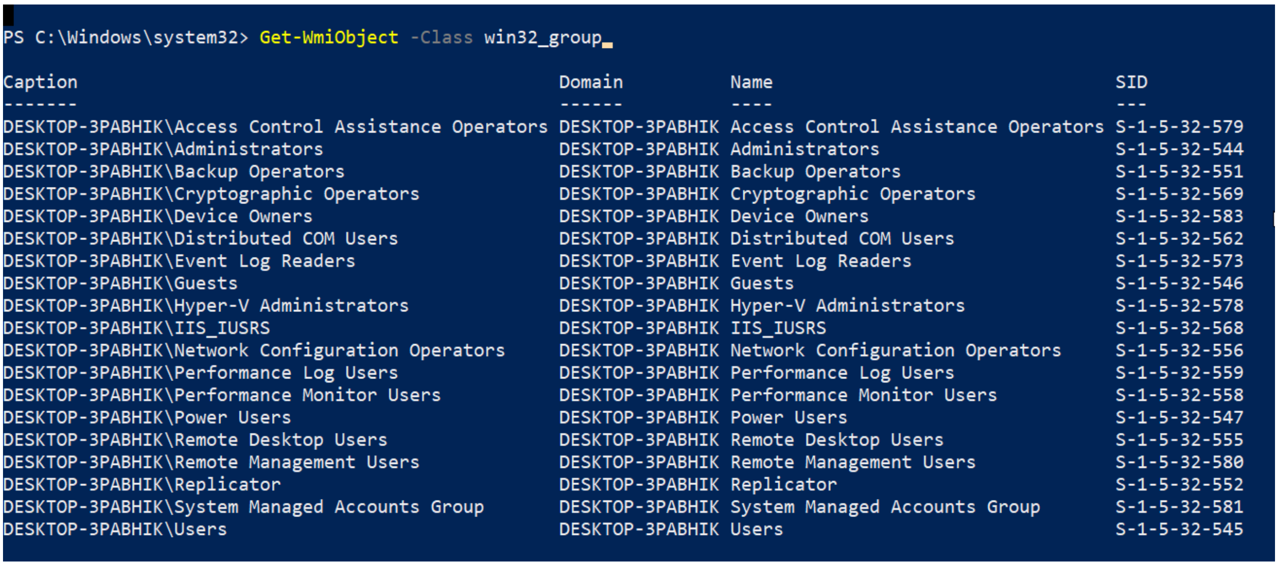


```
PS C:\Users\Administrator> Get-WmiObject -Class win32_useraccount


AccountType : 512
Caption     : INFECTED\Administrator
Domain      : INFECTED
SID         : S-1-5-21-2553750175-4195942334-2808156689-500
FullName    :
Name        : Administrator

AccountType : 512
Caption     : INFECTED\Guest
Domain      : INFECTED
SID         : S-1-5-21-2553750175-4195942334-2808156689-501
FullName    :
Name        : Guest

AccountType : 512
Caption     : INFECTED\krbtgt
Domain      : INFECTED
SID         : S-1-5-21-2553750175-4195942334-2808156689-502
FullName    :
Name        : krbtgt
```

User groups

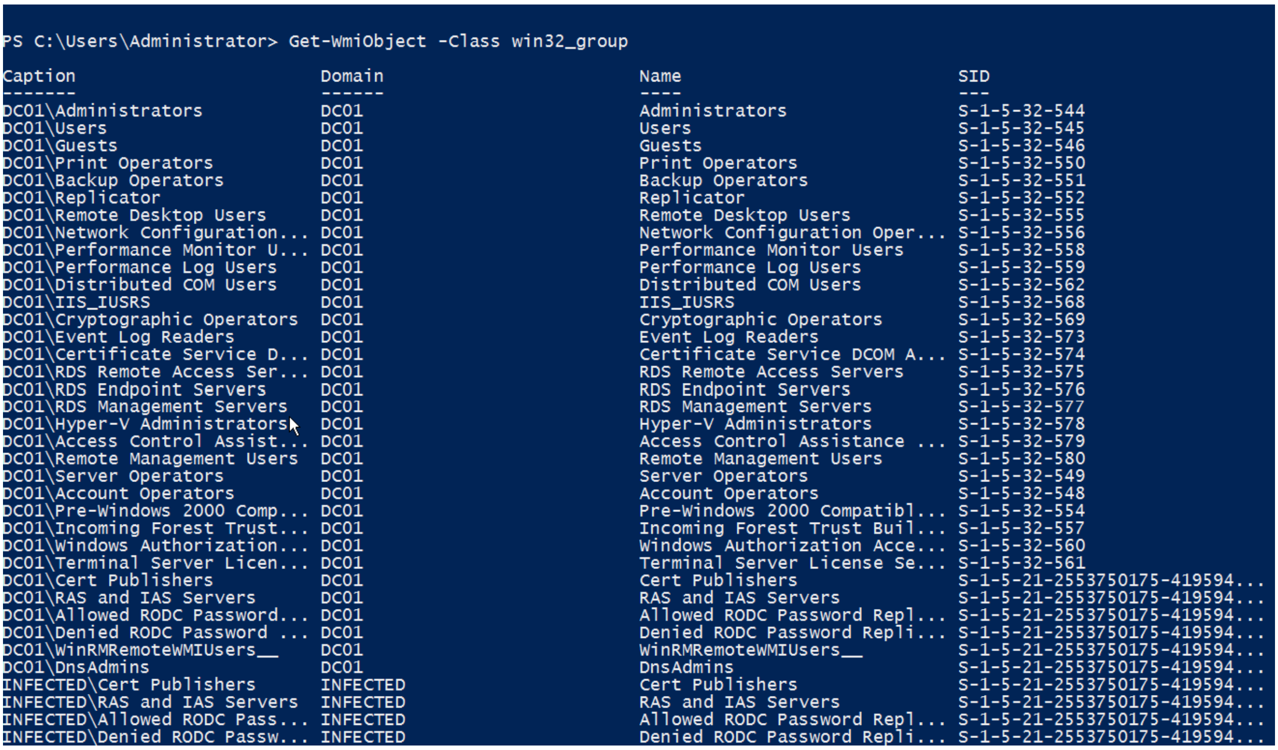
Similar to user accounts, user groups information is provided by the `Win32_Group` class. Querying the class on a local box is easy:

```
Get-WmiObject -Class win32_group
```



```
PS C:\Windows\system32> Get-WmiObject -Class win32_group_

Caption                                            Domain          Name                                 SID
-------                                            ------          ----                                 ---
DESKTOP-3PABHIK\Access Control Assistance Operators DESKTOP-3PABHIK Access Control Assistance Operators S-1-5-32-579
DESKTOP-3PABHIK\Administrators                     DESKTOP-3PABHIK Administrators                       S-1-5-32-544
DESKTOP-3PABHIK\Backup Operators                   DESKTOP-3PABHIK Backup Operators                     S-1-5-32-551
DESKTOP-3PABHIK\Cryptographic Operators            DESKTOP-3PABHIK Cryptographic Operators              S-1-5-32-569
DESKTOP-3PABHIK\Device Owners                      DESKTOP-3PABHIK Device Owners                        S-1-5-32-583
DESKTOP-3PABHIK\Distributed COM Users              DESKTOP-3PABHIK Distributed COM Users                S-1-5-32-562
DESKTOP-3PABHIK\Event Log Readers                  DESKTOP-3PABHIK Event Log Readers                    S-1-5-32-573
DESKTOP-3PABHIK\Guests                             DESKTOP-3PABHIK Guests                               S-1-5-32-546
DESKTOP-3PABHIK\Hyper-V Administrators             DESKTOP-3PABHIK Hyper-V Administrators               S-1-5-32-578
DESKTOP-3PABHIK\IIS_IUSRS                          DESKTOP-3PABHIK IIS_IUSRS                            S-1-5-32-568
DESKTOP-3PABHIK\Network Configuration Operators    DESKTOP-3PABHIK Network Configuration Operators      S-1-5-32-556
DESKTOP-3PABHIK\Performance Log Users              DESKTOP-3PABHIK Performance Log Users                S-1-5-32-559
DESKTOP-3PABHIK\Performance Monitor Users          DESKTOP-3PABHIK Performance Monitor Users            S-1-5-32-558
DESKTOP-3PABHIK\Power Users                        DESKTOP-3PABHIK Power Users                          S-1-5-32-547
DESKTOP-3PABHIK\Remote Desktop Users               DESKTOP-3PABHIK Remote Desktop Users                 S-1-5-32-555
DESKTOP-3PABHIK\Remote Management Users            DESKTOP-3PABHIK Remote Management Users              S-1-5-32-580
DESKTOP-3PABHIK\Replicator                         DESKTOP-3PABHIK Replicator                           S-1-5-32-552
DESKTOP-3PABHIK\System Managed Accounts Group      DESKTOP-3PABHIK System Managed Accounts Group        S-1-5-32-581
DESKTOP-3PABHIK\Users                              DESKTOP-3PABHIK Users                                S-1-5-32-545
```

If the same command is run in an enterprise environment, e.g. a domain-joined network, the number of groups would increase giving us a wider view of the user groups present on a network. This will include the local ones, the current domain, the trusted domain and the trusted forest as well:



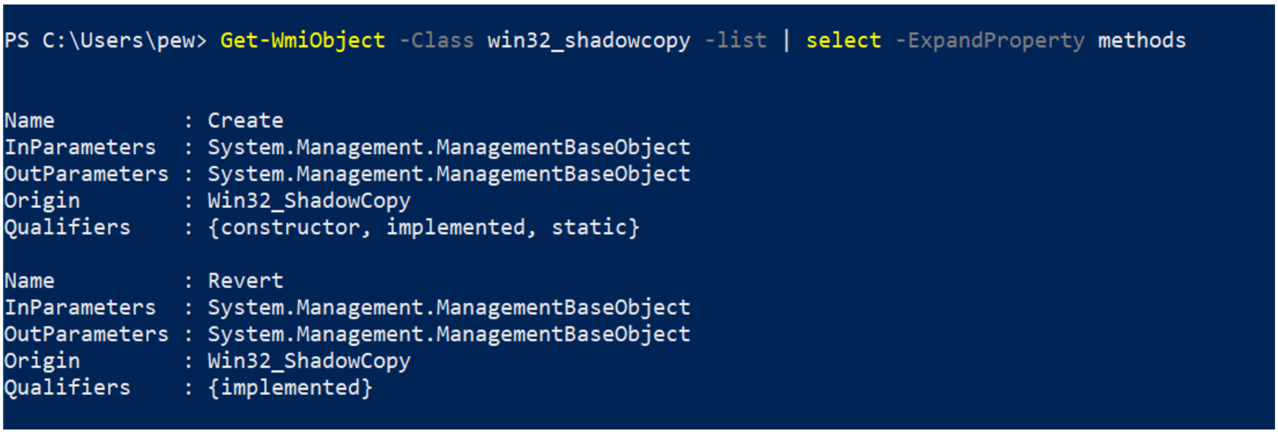
```
PS C:\Users\Administrator> Get-WmiObject -Class win32_group

Caption                      Domain                       Name                         SID
-------                      ------                       ----                         ---
DC01\Administrators          DC01                         Administrators               S-1-5-32-544
DC01\Users                   DC01                         Users                        S-1-5-32-545
DC01\Guests                  DC01                         Guests                       S-1-5-32-546
DC01\Print Operators         DC01                         Print Operators              S-1-5-32-550
DC01\Backup Operators        DC01                         Backup Operators             S-1-5-32-551
DC01\Replicator              DC01                         Replicator                   S-1-5-32-552
DC01\Remote Desktop Users    DC01                         Remote Desktop Users         S-1-5-32-555
DC01\Network Configuration... DC01                        Network Configuration Oper... S-1-5-32-556
DC01\Performance Monitor U... DC01                        Performance Monitor Users    S-1-5-32-558
DC01\Performance Log Users   DC01                         Performance Log Users        S-1-5-32-559
DC01\Distributed COM Users   DC01                         Distributed COM Users        S-1-5-32-562
DC01\IIS_IUSRS               DC01                         IIS_IUSRS                    S-1-5-32-568
DC01\Cryptographic Operators DC01                         Cryptographic Operators      S-1-5-32-569
DC01\Event Log Readers       DC01                         Event Log Readers            S-1-5-32-573
DC01\Certificate Service D... DC01                        Certificate Service DCOM A... S-1-5-32-574
DC01\RDS Remote Access Ser... DC01                        RDS Remote Access Servers    S-1-5-32-575
DC01\RDS Endpoint Servers    DC01                         RDS Endpoint Servers         S-1-5-32-576
DC01\RDS Management Servers  DC01                         RDS Management Servers       S-1-5-32-577
DC01\Hyper-V Administrators  DC01                         Hyper-V Administrators       S-1-5-32-578
DC01\Access Control Assist... DC01                        Access Control Assistance ... S-1-5-32-579
DC01\Remote Management Users DC01                         Remote Management Users      S-1-5-32-580
DC01\Server Operators        DC01                         Server Operators             S-1-5-32-549
DC01\Account Operators       DC01                         Account Operators            S-1-5-32-548
DC01\Pre-Windows 2000 Comp... DC01                        Pre-Windows 2000 Compatibl... S-1-5-32-554
DC01\Incoming Forest Trust... DC01                        Incoming Forest Trust Buil... S-1-5-32-557
DC01\Windows Authorization... DC01                        Windows Authorization Acce... S-1-5-32-560
DC01\Terminal Server Licen... DC01                        Terminal Server License Se... S-1-5-32-561
DC01\Cert Publishers         DC01                         Cert Publishers              S-1-5-21-2553750175-419594...
DC01\RAS and IAS Servers     DC01                         RAS and IAS Servers          S-1-5-21-2553750175-419594...
DC01\Allowed RODC Password... DC01                        Allowed RODC Password Repl... S-1-5-21-2553750175-419594...
DC01\Denied RODC Password ... DC01                        Denied RODC Password Repli... S-1-5-21-2553750175-419594...
DC01\WinRMRemoteWMIUsers__   DC01                         WinRMRemoteWMIUsers__        S-1-5-21-2553750175-419594...
DC01\DnsAdmins               DC01                         DnsAdmins                    S-1-5-21-2553750175-419594...
INFECTED\Cert Publishers     INFECTED                     Cert Publishers              S-1-5-21-2553750175-419594...
INFECTED\RAS and IAS Servers INFECTED                     RAS and IAS Servers          S-1-5-21-2553750175-419594...
INFECTED\Allowed RODC Pass... INFECTED                    Allowed RODC Password Repl... S-1-5-21-2553750175-419594...
INFECTED\Denied RODC Passw... INFECTED                    Denied RODC Password Repli... S-1-5-21-2553750175-419594...
```

System secrets

System secrets are once again useful info to enumerate when it comes to recon. If we have enough privileges on the system, we can create **shadow copies** of the disk and try to extract secrets from there. But before that for those of you not familiar with shadow copies:

> **Shadow Copy** is a technology included in Microsoft Windows that can create backup copies or snapshots of computer files or volumes, even when they are in use.

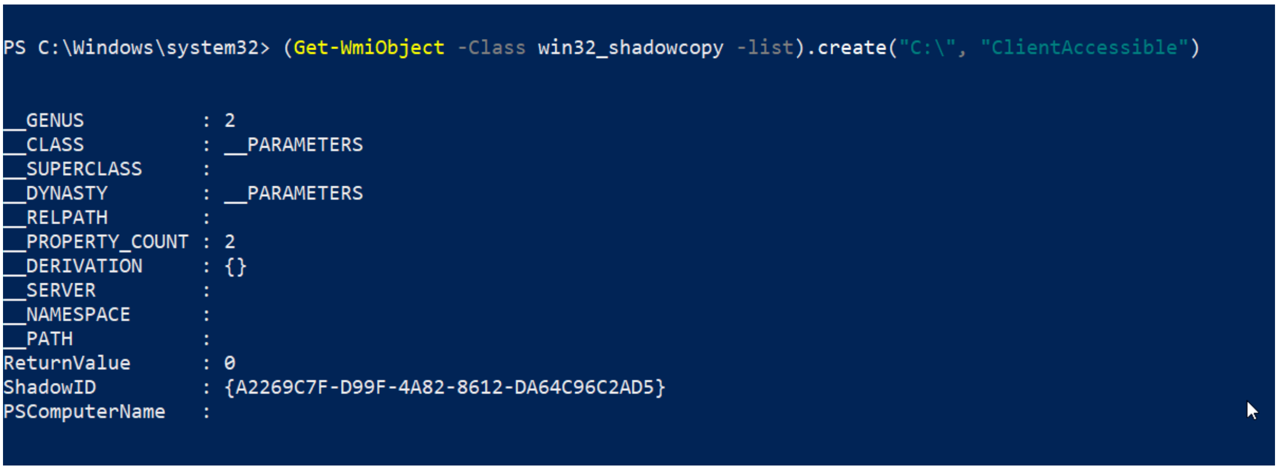
To interact with the shadow copies, we have 2 available methods as seen in the picture below:



```
PS C:\Users\pew> Get-WmiObject -Class win32_shadowcopy -list | select -ExpandProperty methods


Name          : Create
InParameters  : System.Management.ManagementBaseObject
OutParameters : System.Management.ManagementBaseObject
Origin        : Win32_ShadowCopy
Qualifiers    : {constructor, implemented, static}

Name          : Revert
InParameters  : System.Management.ManagementBaseObject
OutParameters : System.Management.ManagementBaseObject
Origin        : Win32_ShadowCopy
Qualifiers    : {implemented}
```

Creating a quick shadow copy is easy, we just need to specify the volume and the context of the copy creation:

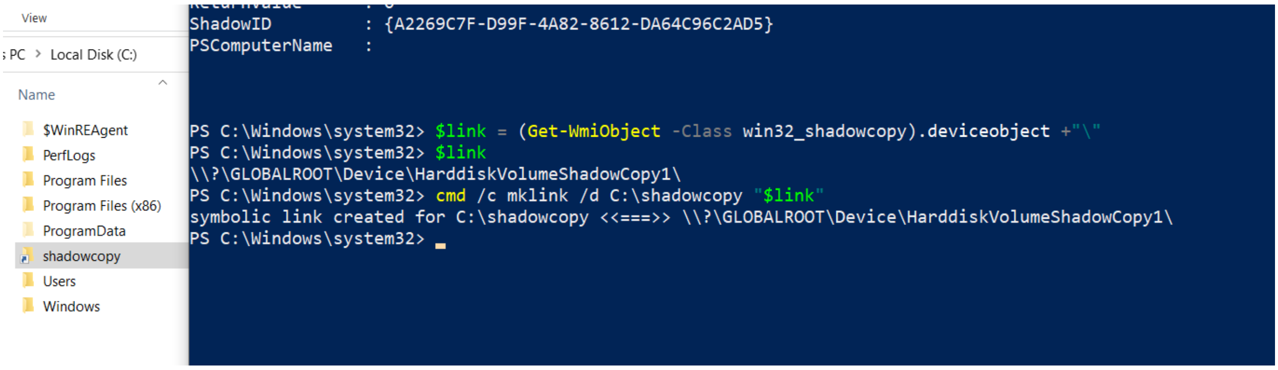
```
(Get-WmiObject -Class win32_shadowcopy -List).create("C:\", "ClientAccessible")
```



```
PS C:\Windows\system32> (Get-WmiObject -Class win32_shadowcopy -list).create("C:\", "ClientAccessible")


__GENUS          : 2
__CLASS          : __PARAMETERS
__SUPERCLASS     :
__DYNASTY        : __PARAMETERS
__RELPATH        :
__PROPERTY_COUNT : 2
__DERIVATION     : {}
__SERVER         :
__NAMESPACE      :
__PATH           :
ReturnValue      : 0
ShadowID         : {A2269C7F-D99F-4A82-8612-DA64C96C2AD5}
PSComputerName   :
```

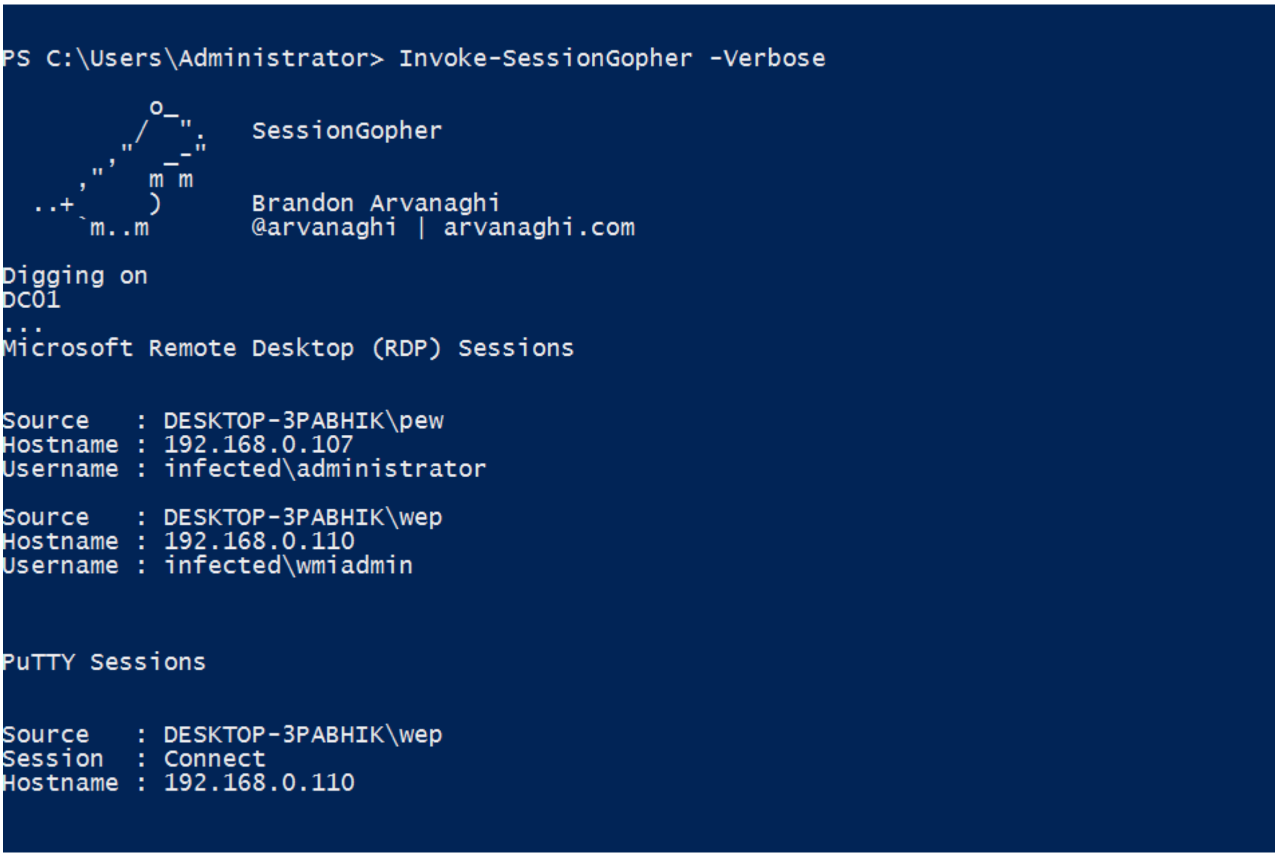
To add to this, we can create a symlink to easily access the shadow copy from our local explorer:

```
$link = (Get-WmiObject -Class win32_shadowcopy).deviceobject + "/"
cmd /c mklink /d C:\shadowcopy "$link"
```



```
ReturnValue     : 0
ShadowID        : {A2269C7F-D99F-4A82-8612-DA64C96C2AD5}
PSComputerName  :


PS C:\Windows\system32> $link = (Get-WmiObject -Class win32_shadowcopy).deviceobject +"\"
PS C:\Windows\system32> $link
\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\
PS C:\Windows\system32> cmd /c mklink /d C:\shadowcopy "$link"
symbolic link created for C:\shadowcopy <<===>> \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\
PS C:\Windows\system32> _
```

Once we have the shadow copy ready to use, we can simply run tools like `Invoke-SessionGopher.ps1` with the `-Thorough` switch to search for secrets on the filesystem. This would yield saved session information for PuTTY, WinSCP, FileZilla, SuperPuTTY, RDP, etc. In my case, I found a few saved RDP sessions and PuTTY sessions using the script.



```
PS C:\Users\Administrator> Invoke-SessionGopher -Verbose

          o_
         /  ".      SessionGopher
       ,"  _-"
     ,"   m m
  ..+     )        Brandon Arvanaghi
     `m..m         @arvanaghi | arvanaghi.com

Digging on
DC01
...
Microsoft Remote Desktop (RDP) Sessions


Source   : DESKTOP-3PABHIK\pew
Hostname : 192.168.0.107
Username : infected\administrator

Source   : DESKTOP-3PABHIK\wep
Hostname : 192.168.0.110
Username : infected\wmiadmin



PuTTY Sessions


Source   : DESKTOP-3PABHIK\wep
Session  : Connect
Hostname : 192.168.0.110
```

Conclusion

So this was all about information gathering over WMI for a single blog post. We saw how we can gather so much useful data in just a few key taps so conveniently. Of course, the information presented above is not exhaustive and there are endless possibilities to consider when it comes to reconnaissance.

That's it for now folks and I'll meet you in our next blog that will focus on Active Directory enumeration via WMI. Sláinte! 🥂