LPE Lab 03 [**Unquoted Service path**]

Table of Contents

Setup requirements

Setup requirements

Use the Tools below to conduct our service enumeration and identify misconfigurations that we can leverage later for the privilege escalation:

- windows 10 host (Victim)
- Kali Linux host (Attacker)
- Process Monitor

"Unquoted service path" is when a Windows service's executable path (the *ImagePath* in the Service Control Manager) contains spaces **and** is not wrapped in double quotes. When the Service Control Manager starts that service, Windows may misinterpret where the executable actually lives and will try a sequence of "shorter" paths. If any directory along that path is writable by a low-privileged user, they can drop a fake executable that Windows will happily start **with the service's privileges** (often LocalSystem). That's a tidy local privilege escalation.

What is an unquoted service path?

Every Windows service has a command that tells Windows what to start. You can see it in the registry under:

HKLM\SYSTEM\CurrentControlSet\Services\<ServiceName>\ImagePath

A *safe* ImagePath looks like this (note the quotes):

"C:\Program Files\Acme Corp\Service\acmesvc.exe" -k run

An *unsafe* (unquoted) ImagePath with spaces looks like this:

C:\Program Files\Acme Corp\Service\acmesvc.exe -k run

Both look similar, but the second one lacks quotes around the path that contains spaces. That tiny omission changes how Windows parses the string when launching the service.

How it works under the hood

When the Service Control Manager (SCM) starts a service, it ultimately calls into the standard process-creation logic (CreateProcess). If the executable path contains spaces and isn't quoted, Windows doesn't know where the filename ends and where the arguments begin. So it tries a search strategy, testing progressively longer substrings ending at each space until it finds an executable that exists.

Take this unquoted line:

C:\Program Files\Acme Corp\Service\acmesvc.exe -k run

Windows will attempt something like this, in order:

1. **C:\Program.exe**
2. **C:\Program Files\Acme.exe**
3. **C:\Program Files\Acme Corp\Service\acmesvc.exe** ← the intended target

If **any earlier candidate** exists, Windows will launch *that* instead. If a low-privileged user can write to one of those locations (for example, a misconfigured folder like **C:\ or C:\Program Files\Acme Corp),** they can plant a binary named Program.exe or Acme.exe. On the next service start, Windows runs the attacker's file **with the service's account** (often LocalSystem), turning a standard user into an administrator (or worse).

That's the core of the issue:

- **Ambiguous parsing** of a path with spaces + **missing quotes** => **search order side effect**
- **Writable directory** along that path + **elevated service start** => **privilege escalation**

In typical, well-hardened systems, places like C:\ and C:\Program Files are **not** writable by normal users, which mitigates many cases. But real environments are messy: third-party products sometimes install to custom folders with permissive ACLs, admins relax permissions for support tasks, or software creates subfolders under C:\ with "Everyone: Modify." One mistake meets one missing quote—and you've got a local escalation path.

Why this is considered a security vulnerability

Unquoted service paths are a classic **local privilege escalation** vector:

- **Impact:** An unprivileged user can execute code as the service account (often LocalSystem). That means full control of the box—install drivers, dump secrets, disable EDR, pivot, you name it.
- **Stealth:** The "service" started legitimately. EDR may only see "a service started" unless it inspects the *executed image* path closely.
- **Reliability:** The behavior is deterministic. If the attacker can create the right filename in the right writable directory, the service will load it on next start (boot, crash/restart, or manual start).

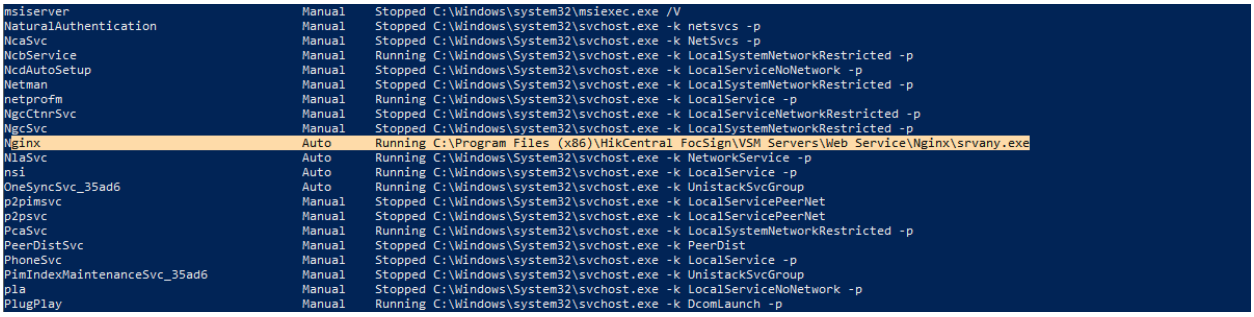
Security teams treat these as high-priority findings because they're easy to miss in configuration reviews and can unravel your privilege boundaries in one move.

Lab Setup

install HikCentral FocSign (any build between v1.4.0 and v2.2.0), and launch it once so its services are registered. Then pivot into an elevated PowerShell session. We'll query the service catalog and flag any service whose executable path contains spaces and isn't properly quoted—classic unquoted service path exposure.

```
# List services whose ImagePath contains spaces and lacks quotes
Get-CimInstance Win32_Service |
  Where-Object { $_.PathName -match '\s' -and $_.PathName -notmatch '^".*"$' } |
  Select-Object Name, StartMode, State, PathName |
  Sort-Object Name
```

After Running the Command, you will find an output like below image. now you can identify this is an unquoted service path



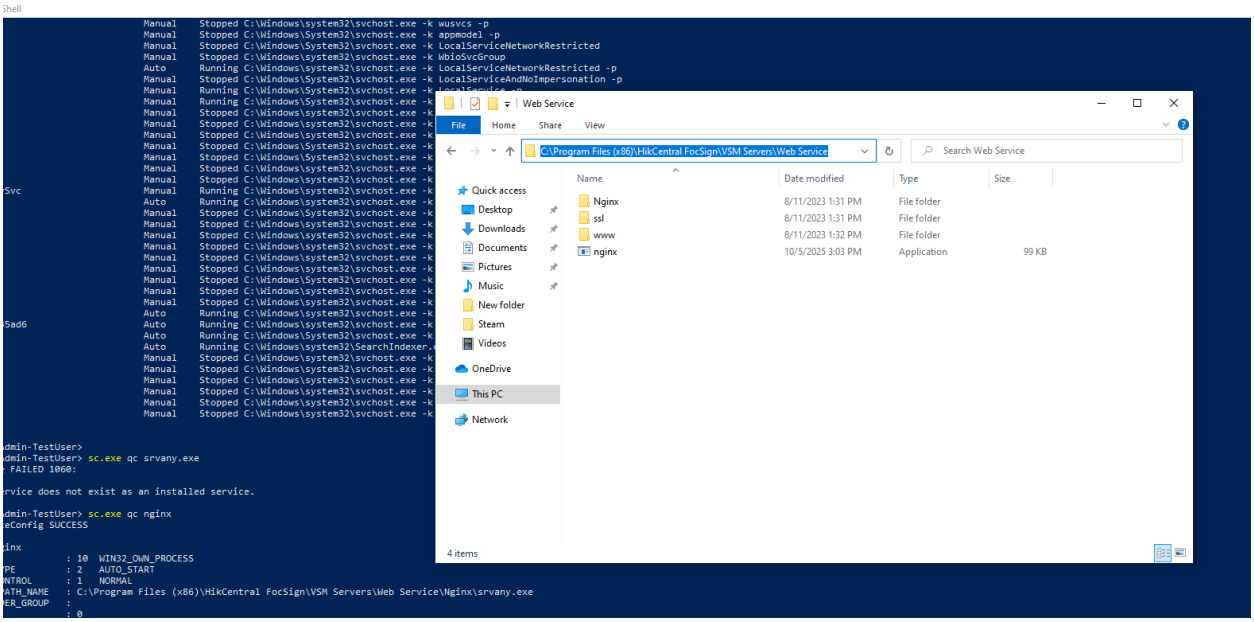
Note: You can use the tool like WinPeas for this instead of the PS Command.

Using **sc.exe qc service name** we can get better view of the service

```
PS C:\Users\NonAdmin-TestUser> sc.exe qc nginx
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: nginx
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\Program Files (x86)\HikCentral FocSign\VSM Servers\Web Service\Nginx\srvany.exe
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : Nginx
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem
PS C:\Users\NonAdmin-TestUser>
```

So now if we were to add our malicious exe to the path we can get it executed when the programs next start so for this lab I will create a exe that open up a cmd and add it to the above Hikvision path.



So like this adding the malicious code to the path we can run it with the same privileges as the Programme running .