Unquoted service paths: The new frontier in script kiddie security vulnerability reports

devblogs.microsoft.com/oldnewthing/20240723-00

July 23, 2024

Raymond Chen

Some time ago, my colleague Aaron Margosis wrote about how most "Unquoted Service Paths" findings are unnecessarily alarmist. But that doesn't stop people from reporting it anyway.

Usually from people who don't actually know what they're doing.

We often get unquoted service path vulnerability reports. Sometimes they go like this:

> We have identified an unquoted service path: The XYZ service has a listed service path of `C:\Program Files\Windows Xyz\XyzSvc.exe` with no quotation marks to protect the spaces.
>
> Attached find a proof of concept. Copy this program to `C:\Program.exe` or `C:\Program Files\Windows.exe`, then use the Services MMC snap-in to stop the XYZ service, then start it. The proof of concept program will run.

As with most unquoted service path vulnerabilities, this one requires that the attacker be on the other side of the airtight hatchway: Creating files in `C:\` or in `C:\Program Files` already requires administrator privilege, so this attack presupposes that the attacker has gained administrator access. It is not surprising that an attacker with administrator access can gain administrator access.

Nevertheless, when we resolve the issue as "Not exploitable, fix in next version", the finder intended to go public and sent us a preliminary copy of a blog entry they intended to publish.

The blog entry admitted that a default-configured system is not vulnerable due to the inability of non-administrative users to plant `Program.exe` in an exploitable directory, but noted that a system administrator might misconfigure the system to grant write access to those sensitive directories.

Of course, we have now wandered into the realm of creating an insecure system and then being surprised that it's insecure.

As far as I can tell, the finder never published that blog entry.

But at least this is a case where the finder actually understood the issues. Often we're not so lucky, and the finder just spits out some tool output without providing any diagnosable information.

> The XYZ service has an unquoted service path, which could allow a user to gain SYSTEM privileges. Attached please find screen shots demonstrating the issue.

The screen shots are heavily redacted captures from some unknown vulnerability scanning software.

Service name	**Vulnerable systems**
Xyz	7

That's nice, but there's nothing diagnosable here. The finder did include a screen shot of the scanning software reporting a non-vulnerable service, but that doesn't help us identify the vulnerable one.

After some back and forth with the finder, we were able to obtain the path to the vulnerable service, and it was of the form `C:\ProgramData\Microsoft\Windows Xyz\XyzSvc.exe`, which is not exploitable because it requires administrator privileges to write to the `C:\ProgramData\Microsoft\Windows` directory.

Quoting service paths is a best practice. If you forget, most of the time, other defense in depth measures prevent it from being exploitable. It's still good to fix them even though they aren't exploitable, because you don't want to rely on the kindness of others. However, you don't have to fix them with the urgency of a security vulnerability.

Another example of an alleged unquoted service path vulnerability is this one:

> The lack of proper quotation marks around the service path for the XYZ service means that this vulnerability could be exploited to achieve privilege escalation. I found this on multiple systems after running a Contoso security scan.
>
> `C:\Windows\system32\svchost.exe -k xyz`

In this case, the finder ran a commercial scanning tool with a free trial, and the tool reportedly claimed that this service path was unquoted.

While it's true that the path is unquoted, it's also true that quotation marks aren't needed because there are no spaces in the path.

The path is `C:\Windows\system32\svchost.exe`. The extra `-k xyz` are command line arguments to the program. They aren't part of the path-with-spaces. In other words, this service is not trying to run a program with the funny name `svchost.exe -k xyz.exe` in the `C:\Windows\system32` directory. The intention is to run the `C:\Windows\system32\svchost.exe` program. The lack of quotation marks is the *intended interpretation*.

Some script kiddies try to supplement their report with breathless prose cobbled together from fragments of other vulnerability reports they found on the Internet.[1]

> This unquoted path can lead the system to access resources in a parent path. A local attacker can place an executable file in the path of the service. When the service starts or restarts, the malicious file is executed instead of the intended service.

It's not clear what "place an executable file in the path of the service" means here. If they mean insert an executable file in the same directory as the service, then that doesn't work. The system will still run the intended file.

If they mean to put a file in a directory in the service's `PATH` environment variable, that still doesn't work, because the service is registered with a full path. (And even if the service were register with an unqualified path, attacking the `PATH` directories is not fruitful because all of those directories by default are writable only by administrators anyway.)

If they mean to overwrite the service executable with another executable, well, quotation marks won't do anything to block that.

In this particular report, their so-called "repro steps" didn't actually repro any attack. All they did in the repro steps was enable the service. They never planted any file to trigger unauthorized code execution. All we can do is guess what they meant; we can't try to infer it from their proof of concept.

But the clincher was the output of their alleged repro steps:

```
C:\> sc query xyz
    SERVICE_NAME: xyz
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : "C:\Program Files\Microsoft Xyz\XyzSvc.exe"
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : Microsoft XYZ Service
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem
```

The reportedly unquoted service path is quoted!

Bonus chatter: When the security vulnerability reports reach the engineering team, the identifying information about the finder has often been removed. I'm guessing that this is done to remove sources of bias that could be introduced by recognizing, "Oh no, it's this guy again," and not giving the report due consideration because of its source.

That said, it's still possible to identify that two reports came from the same person. The writing styles may match up (and sometimes the two reports are word-for-word identical, just with a service name changed). And one time, I noticed that the proof of concept video that was included with the report had exactly the same wallpaper and desktop icons as another report.

[1] Sometimes the breathless prose is outright wrong.

> An attacker could place a malicious executable in a directory whose name contains a space.

As noted above, the attack vector is not placing a malicious executable in a directory whose name contains a space. It's placing a malicious execute in a directory whose name is a *truncation* of the unquoted path.