Help eliminate unquoted path vulnerabilities - SANS ISC

By SANS Internet Storm Center

Archived: 2026-04-05 17:03:36 UTC

Metasploit's "Service Trusted Path Privilege Escalation" exploit takes advantage of unquoted service paths vulnerability outline in CVE-2005-1185, CVE=2005-2938 and CVE-2000-1128. The vulnerability takes advantage of the way Windows parses directory paths to execute code. Consider the following command line.

C:\windows\system32\notepad \temp\file.txt

This tells windows to launch notepad.exe from the c:\windows\system32\ directory and pass it the argument \temp\file.txt. The result is notepad.exe will execute and begin editing file.txt from the temp directory. How does Window's differentiate between the program and the arguments? The SPACE is used as a delimiter between the program to execute and the arguments. Now consider this command line.

C:\program files\Microsoft Office\Winword.exe

If space is used as a delimiter, wouldn't Windows think you are trying to execute the program C:\PROGRAM.EXE and pass it the argument "files\Microsoft Office\Winword.exe"? Or maybe you are trying to execute "C:\Program files\Microsoft.exe" and pass it the argument "Office\Winword.exe"? So how does it know what you are trying to do? If the software developer places quotation marks around the path then Windows knows the spaces are spaces and not delimiters. If the software developer fails to put the path in quotes then Windows just doesn't know. If Windows doesn't know then it tries to execute all the possible programs in the path. First it tries "**C:\Program.exe**", Then, it tries "**C:\Program files\Microsoft.exe**" and finally the path we intended for it to execute.

This programming error is very common because when a developer is addressing paths on the file system they are usually stored in strings. Because they are in strings the developer has used quotes once already and they often fail to consider that they need two sets of quotes. For example, the following line would incorrectly assign the path variable.

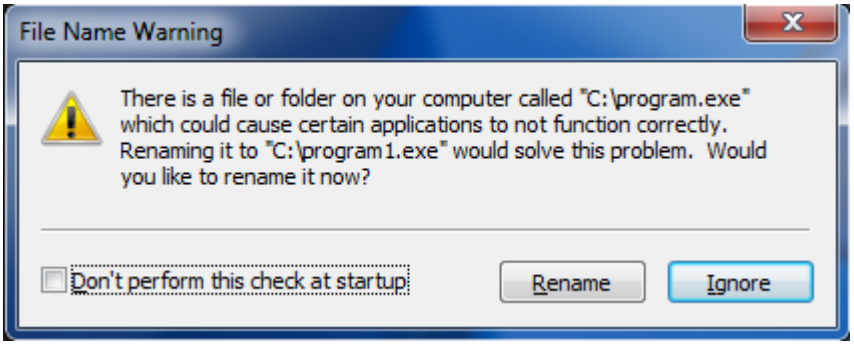
pathvariable = "C:\Program Files\Common Files\Java\Java Update\jusched.exe"

Really, the developer needs to double quote it because they need the path to contain quotes. So they should have assigned their variable by doing something like this:

pathvariable = "\\"C:\Program Files\Common Files\Java\Java Update\jusched.exe\\""

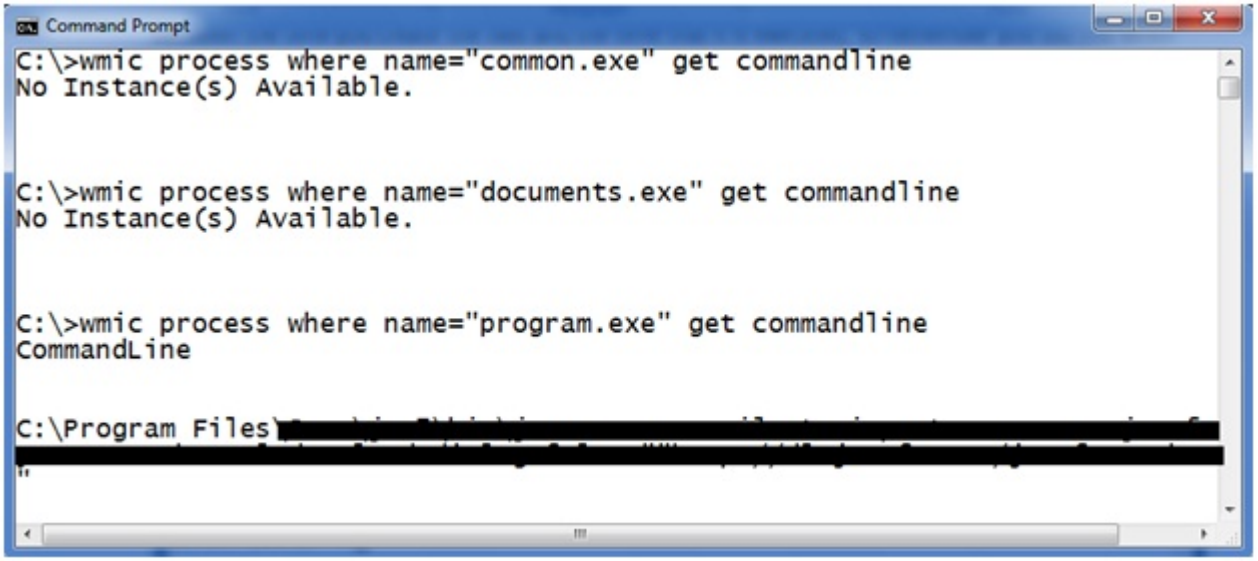
In the first case, an attacker can strategically place a program in the path and his program will be executed instead of the intended program. If the process runs under administrative privileges or some account other than the attacker it can be used to cause code to execute under a different set of privileges.

We have known about these types of vulnerabilities for 12 years now. So much so, that if you create a file called c:\program.exe Windows will generate this pop up when you reboot the machine.



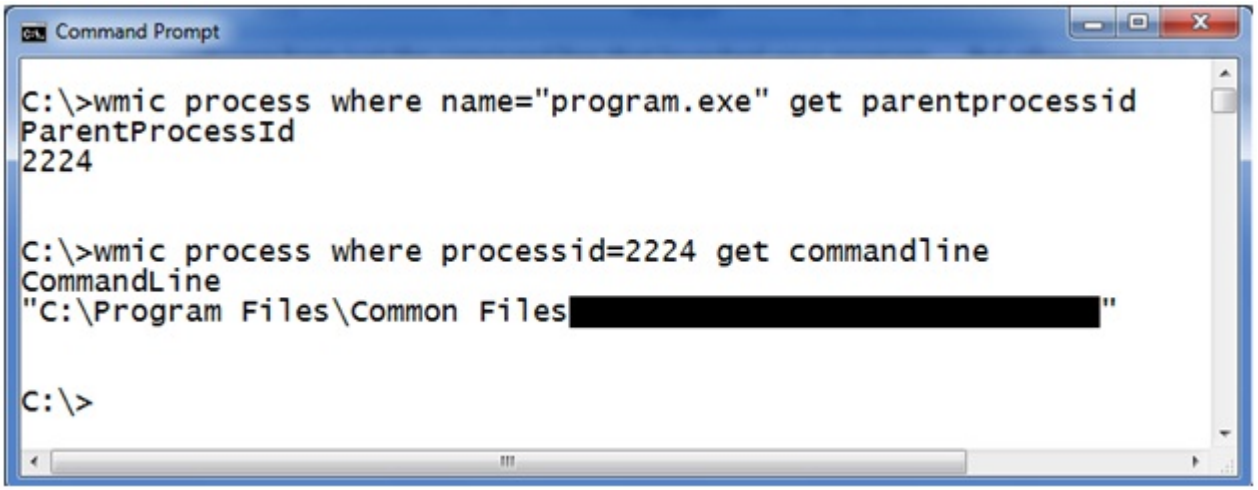
With such an old vulnerability, surely very few programs suffer from this problem, right? You might be surprised at how often this vulnerability occurs. So let's start fixing it! This is an easy problem to identify. Here are some steps you can follow to identify applications that fail to quote their file paths directly. Then you can help fix this by contacting the vendor to have them fix these issues.

First you need to copy any existing executable and create a program named c:\program.exe. For example, take a copy of calc.exe and name it c:\program.exe. Then make a copy of calc.exe named c:\program files\common.exe. Last, create a copy of calc.exe called c:\documents.exe. Then go about your business and use your computer as you normally would. Sometime while you are running normal applications they will accidently launch the renamed calc.exe. System reboots, services and scheduled tasks may trigger calculator. Whatever the cause, eventually you will likely run a vulnerable program and it will launch the application on your computer. Several days may transpire between the time you create the files and the time they are executed. Remember you did this when you have strange copies of calc.exe spontaneously launching on your computer. Once one your copy of calculator executes, first find out which one of your calc.exe programs launched and who launched it. Use WMIC to query which copy of your calculator is running like this.



Here you can see this query for "program.exe" returns the command line that was executed when our executable launched. This program failed to properly quote the system paths and launched your renamed program. Often

times you may be able to identify the vendor responsible for the software from just the command line that launched your program. But often times it is also useful to know the parent process that launched the command line that started your program. To find out what program launched the unquoted path we use the following commands.



There you go bug hunter. You found a 0-Day vulnerability! Now notify the vendor so they can patch it and keep everyone safe. While you are at it you can notify us if you would like.

Join me in San Antonio Texas November 27th for SANS 504 Hacker Techniques, Exploits and Incident Response! Register Today!!

Follow me on Twitter @MarkBaggett
Mark Baggett

Update: Here are some additional references. Thanks to altonius for the linksblakhal0 for the post that inspired this.

http://blakhal0.blogspot.com/2012/08/hiding-files-by-exploiting-spaces-in.html

http://xato.net/hardening/the-programexe-problem/

http://www.commonexploits.com/?p=658
