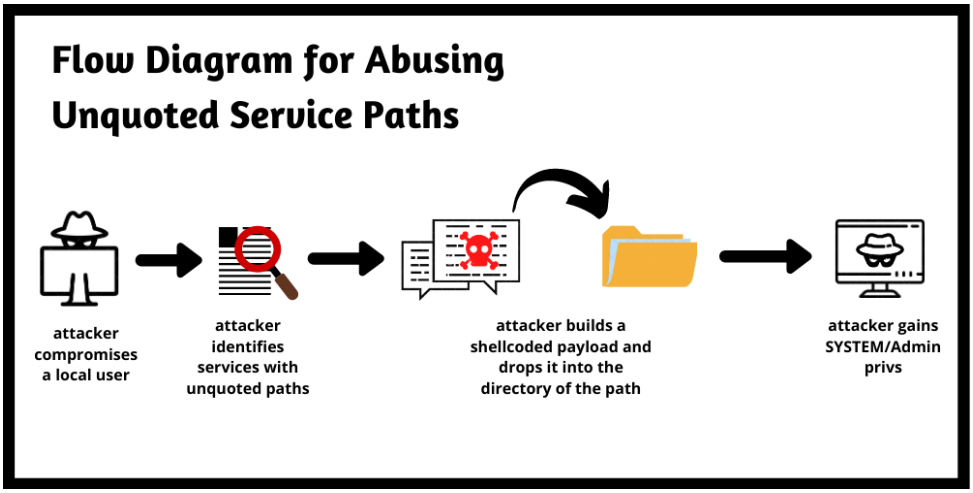
Lab - Verify Windows Privilege Escalation: Unquoted Service Path

Overview

A service whose service executable path contains spaces and is not wrapped within quotes can lead to a vulnerability known as **Unquoted Service Path.** This vulnerability allows a normal user to gain administrative access to the machine by performing privilege escalation using the local system account, which is needed to launch the service executable.

Unquoted Path or Unquoted Service path is reported as a critical vulnerability in Windows. This vulnerability allows attackers to escalate their privileges using the NT AUTHORITY/SYSTEM account.



Exam Topics!

1. Which command will allow a pentester to test for potential unquoted services paths on a host?

A. **Tip! Notice the three sets of quotes at the end of the command. That is your clue.**

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

2. A penetration tester locates a few unquoted service paths during an engagement. Which attacks can the pentester use to test for vulnerabilities?

A. Privilege escalation attacks

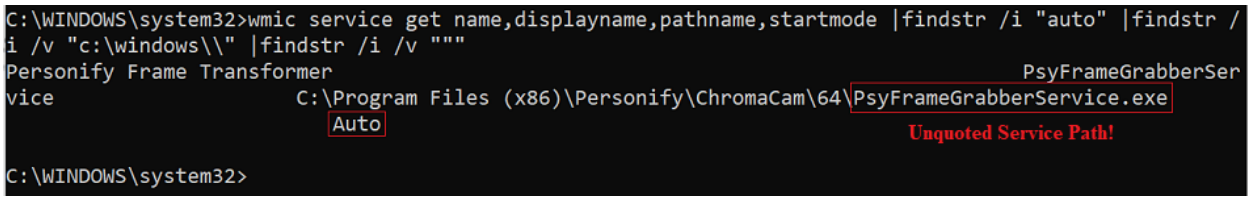
Verifying the Unquoted Service Path vulnerability exists

There is a three-step process to verify if a machine is vulnerable to an unquoted service path exploit.

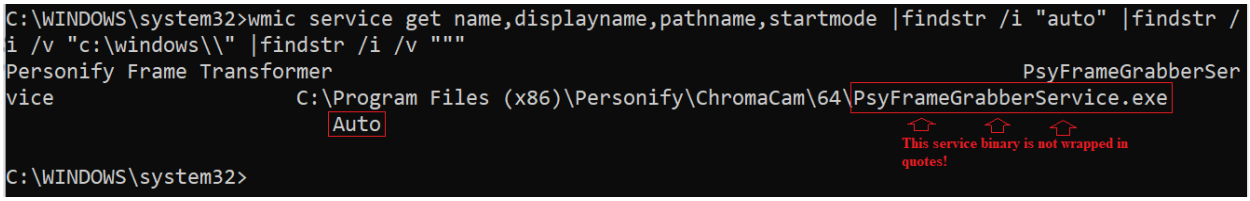
Step 1: Check to see if the machine is vulnerable to an unquoted service path exploit.

In this example, I open a command prompt on my host machine, and I copy and paste the following command at the prompt.

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



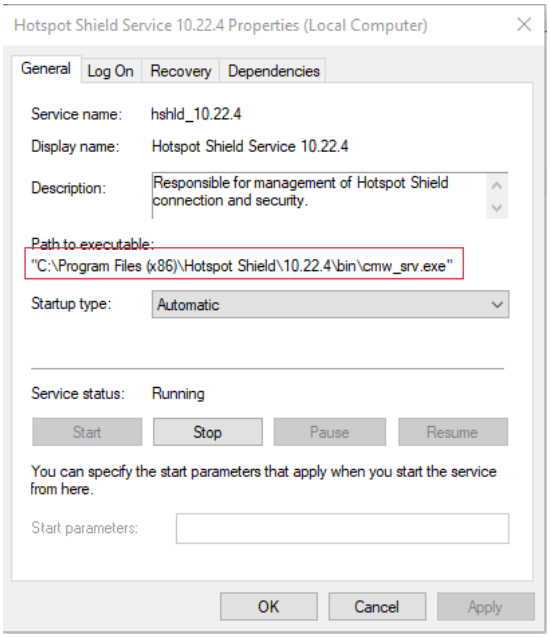
When I press enter, it comes back and tells me the following service binary has the potential for being vulnerable.



Caveat!

This is my host machine. You will not have the same results unless you have the same version of ChromaCam installed on your machine. If you run the above command and you have a vulnerable service, adjust.

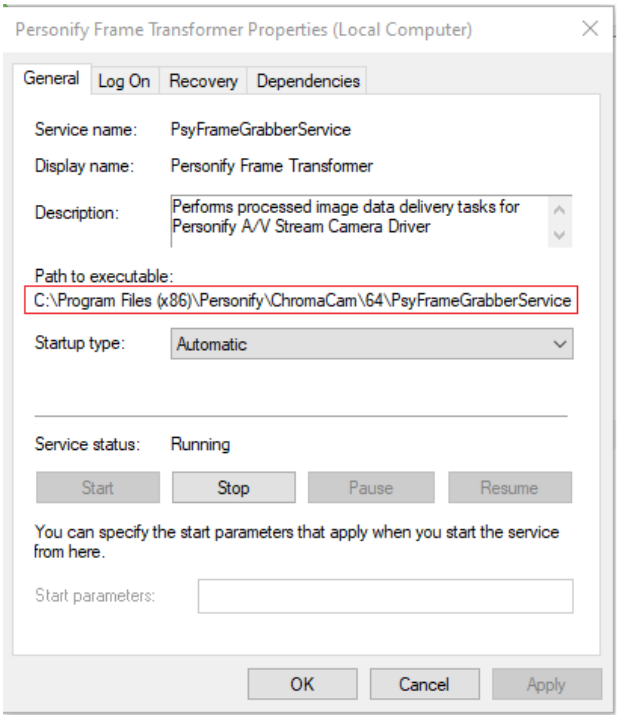
Let's look at what this means. When a Windows service attempts to start, the Windows API must locate the service executable. Usually, the path is well-defined and contained within double quotation marks. Examine the service executable path of my commercial VPN service:



Windows has no question about where to find this executable.

Not wrapping an executable's path in quotes with blank spaces or characters forces the Windows API to search inside the entire path stopping at every folder in the path looking for the service executable.

If we examine the path for my vulnerable service, the path to the service executable is not wrapped in quotes and contains blank spaces.



The Windows API must break apart the path to the executable and search the entire path one folder at a time until it finds the referenced service executable.

Vulnerable Service: C:\Program Files (x86)\Personify\ChromaCam\64\PsyFrameGrabberService

Starting at the root directory, the attacker, as a normal user, could place a different piece of malware inside each subfolder along the path. In addition, the API would see that the user has permission to launch executables.

Step 2. Verify the identity and level of privilege the service is running under

Look to the right of the vulnerable service. In this example, the service is running as SYSTEM.

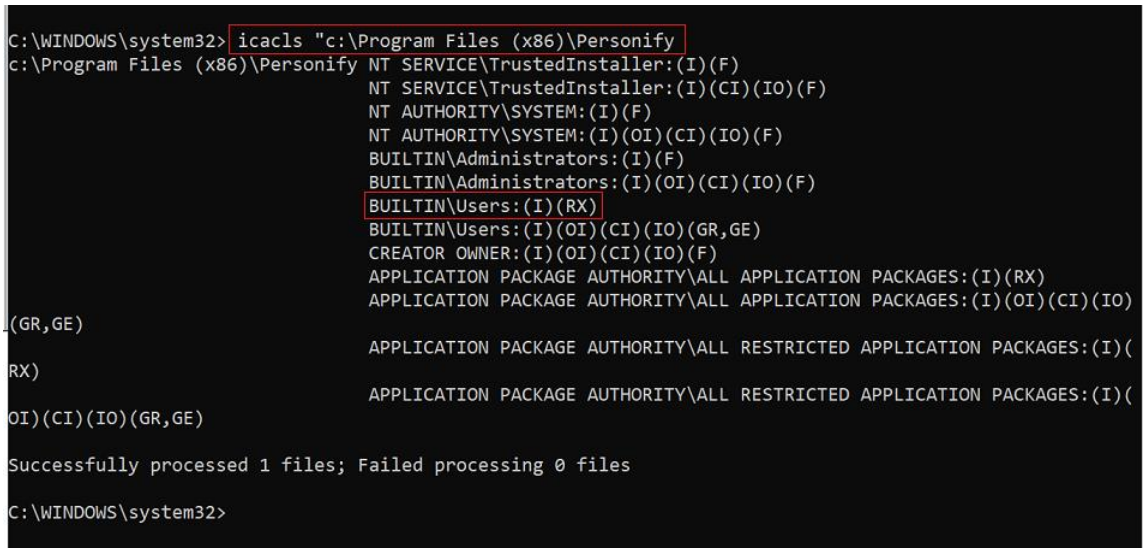
Step 3. Check the permissions of BUILTIN\users for write access to the service executable.

So far, we have discovered the service is running as SYSTEM, and the service path is not wrapped in double quotes. The final check is to determine if the BUILTIN\ users have "Write" access in the directory where the service executable is located or in any directory in the path such as the root, C:\ or C:\Program Files (x86).

Folder permissions can be identified using a Windows built-in tool called **icacls (Integrity Control Access Control Lists)**.

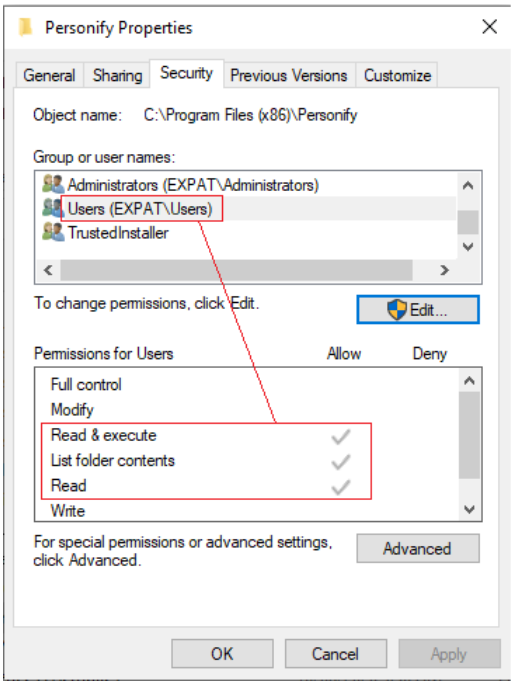
At the command prompt, we type the following command:

```
icacls "c:\Program Files (x86)\Personify"
```



The BUILTIN\Users group assigned to the Personify folder can read, traverse the directory, view data files, and run applications inside the directory (RX). This allows the attacker to generate a malicious binary and plant the executable inside Personify folder or any folder along the service executable path. The write permission ensures that any low-level user account has the right to launch the bogus service each time the machine is restarted. Additionally, this provides that Windows will launch the bogus executable instead of the legitimate one by giving SYSTEM privileges to the user.

We can verify this by viewing the permissions assigned to the BUILTIN\USERS group for the Personify folder.



My host machine meets the criteria of being vulnerable for the Windows Privilege Escalation: Unquoted Service Path.

Mitigation:

1. Vulnerability Solution: Ensure that any service binary with a space in its path is wrapped in quotes.
2. Restrict File and Directory Permissions: Restrict access by setting directory and file permissions that are not specific to users or privileged accounts
3. Execution Prevention: Block code execution on a system through application control and script blocking.

Summary

In this short lab, you learned how to check for the Windows Privilege Escalation: Unquoted Service Path vulnerability.