Dubious security vulnerability: Reading the files in the WindowsApps folder

devblogs.microsoft.com/oldnewthing/20221129-00

November 29, 2022

Raymond Chen

A security vulnerability report arrived that went roughly like this:

> I have discovered a simple way to access protected system files from a non-administrative account. You can use this to hack application data and steal private sensitive administrator data.

The report showed that by a clever series of operations, you can access files in the `C:\Program Files\WindowsApps` folder. In the video they attached, they were able to open a file in that directory and claimed that they had gained the ability to modify application data and steal information from the administrator.

But accessing those files doesn't show either of those privileges. All you got was the ability to read the files that came with the app. You don't have write permission (so you can't tamper with the files), and the administrator's private data is not kept there, so you didn't get any of that either.

I suspect the finder though that they had stumbled across some treasure trove of information because these files are inconvenient to reach. If you go to Explorer or a command prompt, you aren't given access to the `WindowsApps` subdirectory. But the reason for blocking access isn't for any security purpose. It's just to discourage users from accidentally wandering into those directories and trying to run the programs by double-clicking them. These programs run in a special execution environment, and just double-clicking the executable (or typing their name into the command prompt) is not sufficient to get them to run correctly. Rather than lead the user into a pit of failure, we put a *No user-serviceable parts inside* sign at the entrance so users will look elsewhere, and hopefully wind up at the Start menu.

But putting up a sign doesn't mean that anybody who manages to get onto the property has broken through security. It's just a sign. The files themselves are still readable by standard users. They have to be: The program needs to be able to run! If a standard user didn't have access to the files, then a standard user wouldn't be able to run the app, since the system needs to be able to read the files in order to load them into memory, and the app needs to be able to read its own files so it can (say) draw its own icons onto the screen.

The clever series of operations is just style points. You can get the paths to all of the files in the `C:\Program Files\WindowsApps` folder from PowerShell:

```
PS> Get-AppxPackage
```

Grab all of the `InstallLocation`s and start the reading files in them. You always had access to them. The sign was just to discourage you, not to stop you.

Raymond Chen

Follow