It's not a security vulnerability that users can access files that they have access to, even if the file is a little hard to find

devblogs.microsoft.com/oldnewthing/20200113-00

January 13, 2020

Raymond Chen

If a user has access to a file, then they can access the file. That seems self-evident.

Putting that accessible thing in a hard-to-find place doesn't make it inaccessible. It just makes it hard to find.

In Windows, you can put a file in a directory that the user does not have access to, but if the user can produce the name of the file, they can still access it. This works because Windows by default enables "bypass traversal checks", which means that you can access anything you can name.

This technique is a useful trick if you want the user to be able to access something by name, but you don't want to reveal what other nearby things are present but not accessible. You can read more about this scenario in this old *Windows Confidential* article.

In Windows, we sometimes make a directory non-enumerable despite having accessible contents in order to create an obstable to keep casual users away from places that they probably shouldn't be going. For example, the `%USERPROFILE%\ Start Menu` directory is a non-enumerable link to `%USERPROFILE%\AppData\ Roaming\ Microsoft\ Windows\ Start Menu`. That link exists for backward compability, so that programs which hard-coded the string `Start Menu` (instead of using a function like `SHGetSpecialFolderPath`) would still land in the right place. But the directory is non-enumerable because we don't want users browsing around in Explorer and clicking into that directory, thinking that it's the recommended way to find their Start menu.

Similarly, the `%PROGRAMFILES%\ WindowsApps` directory is non-enumerable to prevent users from browsing into it with Explorer and trying to double-click the programs in it. Double-clicking those programs won't work, because they are UWP apps which execute in a special environment that needs to be set up first. Blocking enumeration prevents users from

getting themselves into trouble. The programs themselves and their associated data files must still be readable, because when the program runs, it needs to be able to read those data files.

This means that if you can guess[1] the name of a subdirectory inside the `WindowsApps` directory, you can dive in and snoop around. There's no security hole here. You always had access to those files. You just had to know where they were.

[1] And really, you don't need to guess. PowerShell will tell you.

```
Get-AppxPackage | Select InstallLocation
```

Raymond Chen

Follow