It rather involved being on the other side of this airtight hatchway: Planting files onto a custom PATH

devblogs.microsoft.com/oldnewthing/20200420-00

April 20, 2020

Raymond Chen

For some reason, we get lots of reports about DLL planting that basically boil down to this:

> Program X is susceptible to a DLL planting attack because it loads the DLL `TOTALLYSAFE.DLL` without a full path. If I put a rogue `TOTALLYSAFE.DLL` on the system `PATH` ahead of its actual location, then the rogue copy is loaded into the service, and I have gained elevation of privilege.

When we dig into the report, we find that the directory into which `TOTALLYSAFE.DLL` was planted is not one of the directories that are on the `PATH` by default. It's some custom directory that was added by a third-party program's installer. And that third-party program added a directory that granted write access to non-administrators.

So what we have here is a case of creating an insecure system and then being surprised that it's insecure.

Creating this insecure system was done by editing the global `PATH`, which requires administrator permission. Therefore, we are already on the other side of the airtight hatchway. There is no elevation of privilege, because you need to have administrator privileges to create the insecure system in the first place.

The third-party program decided to install itself into a directory directly off the root of the `C:` drive. If you create your own subdirectory as a direct child of the root, the default security grants Modify access to all authenticated users, and that's dangerous if you're going to add that directory to the `PATH`.

This is one of the reasons why the long-standing recommendation has been to install programs into a subdirectory of `%ProgramFiles%`. The security for `%ProgramFiles%` is set so that only administrators have write access, which means that if you install into a subdirectory of `%ProgramFiles%`, you will get a directory that by default grants write access only to administrators. You can then safely add that directory to the `PATH`.

In many of the cases I've seen, the rogue unsafe directory on the `PATH` belongs to a variety of popular developer tools. My guess is that the finders install these programs by habit into all of their systems, and when they find an issue, it never occurs to them that it was their insecure customizations that was the source of the vulnerability.

Bonus chatter: In one of the cases, the developer tool indeed protects its directories by limiting write access only to administrators. That didn't stop the finder from "planting" a DLL in that protected directory and then "discovering" a vulnerability. So not only did they require elevation to install the developer tool, they also required elevation in order to "plant" the DLL into the protected directory. I guess that puts them on the other side of *two* airtight hatchways.

Raymond Chen

Follow