2019 year-end link clearance: The different kinds of DLL planting

devblogs.microsoft.com/oldnewthing/20191231-00

December 31, 2019

Raymond Chen

This time, the link clearance is just one link:

> Triaging a DLL planting vulnerability.

That link summarizes various causes of alleged DLL planting vulnerabilities and how the Microsoft Security Response Center assesses them.

I've discussed many of these scenarios in the past.

In my experience, people tend to mistake application directory planting for current directory planting because they run the proof-of-concept in a scenario where the current directory is the same as the application directory. They plant a file in the current directory, and it gets loaded: Aha, a current directory vulnerabilily!

Nope. The DLL is being loaded from the directory because it is the application directory, not because it is the current directory. To be sure that it's a current directory attack, you need to make the current directory different from the application directory.

Consider this scenario:

```
C:\appdir> copy \\badplace\files\attack.dll
        1 file(s) copied.

C:\appdir> C:\appdir\app.exe
```

In this case, the application directory is `C:\appdir`. Copying a file into the application directory means that you have already crossed the airtight hatchway and entered the app's "safe zone". If you weren't supposed to be able to get into the app's safe zone, then whoever set up the application directory messed up.

To make this a current directory attack, you need to do this:

```
C:\appdir> mkdir C:\attack

C:\attack> copy \\badplace\files\attack.dll
        1 file(s) copied.

C:\attack> C:\appdir\app.exe
```

If this loads the `attack.dll`, then you've found something.

As for path planting, there tend to be three categories of false reports.

The first is the blatant "other side of the airtight hatchway", where the proof of concept requires planting a DLL into administrator-only directories.

The second is a case where an application running with full user permissions alters the path for the current user, and then uses it to attack that same user. Note that no elevation occurred: The user is attacking himself. It is not a security vulnerability that users can make their own lives miserable.

Besides, if you have a bad actor running with full user permissions, there's no point going through all this path attack nonsense. The bad actor already has full user permissions, so it can just do the bad thing directly. Doesn't need to be sneaky about it.

The third case is where some application's installer put an insecure directory onto the global system path. This is a security vulnerability in the application's installer, where they create an insecure system and then are shocked that the resulting system is insecure.[1]

This third case is frustrating to diagnose because the finder typically doesn't realize that they have installed some program that made changes to the system that introduced the vulnerability. The bug goes through a few rounds of "Not repro", "Is too", "Is not" until we realize that the finder is probably testing on their own machine rather than in a freshly-installed system with all settings at their factory defaults.

Bonus planting: Exploits that require planting `C:\Program.exe` are also invalid, because the default permissions for `C:\` require administrator privileges to create a file. (You do not need administrator privileges to create a new directory in the root, but you do need administrator privileges to create a new file in the root.)

[1] Occasionally, somebody will just come right out and create the insecure system directly in their proof of concept.

1. Create the `C:\Sucker` directory and put a bad thing in it.
2. Add `C:\Sucker` to the `PATH` for the user (or for the system).
3. User (or system) component does bad thing.

Yup, that's a problem. A problem that you created right there in step 2.

Sometimes the finder doesn't quite understand that the problem was of their own doing, and it takes a few more rounds of back-and-forth to get it through to them that all they did was show that users have permission to make their own lives miserable, and that administrators have permission to make everybody's lives miserable.

Raymond Chen

Follow