Not even getting to the airtight hatchway: Creating a process with a different parent

devblogs.microsoft.com/oldnewthing/20200122-00

January 22, 2020

Raymond Chen

A security vulnerability report breathlessly claimed that “By using this technique, a process can create a child process whose parent is not the current process.”

That’s nice. What’s the vulnerability?

There are any number of ways that you can request that a process be created in such a way that the parent of the resulting process will not be you.

- Request an out-of-process activation of an in-process object. COM will create a surrogate process whose parent is the COM subsystem.
- Request an out-of-process activation of an out-of-process object. COM will launch the local server, and its parent is the COM subsystem.
- Create a scheduled task and trigger it. The parent will be the task scheduler.
- Use shell automation to launch a process from Explorer. The parent will be Explorer.
- Use the `PROCESS_CREATE_PROCESS` privilege to make a specific process become the parent of the created process.
- Register a hook on a target process, trigger the hook, then create the child process from inside the hook handler.
- Inject a DLL into a target process, and have the DLL create the child process.

I’m sure there are plenty more ways of doing this.

None of these operations result in elevation of privilege. In the COM surrogate scenario, the resulting process runs at the same security level as the invoker. In the COM local server and task scheduler scenarios, the resulting process runs based on how the server or task was configured, and configuring a process to run elevated itself requires elevation, so gaining elevation requires you already to be on the other side of the airtight hatchway. In the Explorer and `PROCESS_CREATE_PROCESS` cases, you are accessing a process that you already have access to (probably because it is running as you). In the last two cases, you are accessing processes that you already have code injection privileges to.

So none of these mechanisms give you access to anything you didn't already have access to. No security boundary was crossed.

Windows doesn't really care about parent processes much. The fact that you can obtain the parent process ID for a child process gives you some information that could be incorrect by the time you receive it. Windows permits the parent process to exit before its children, and process IDs are recycled, so when you ask a process for its parent, you may be the ID of a totally unrelated process.

The parent process ID is not a security feature. It's just a random piece of data that some people find interesting. The identity of the parent is used for some things, like handle inheritance and job object accounting, but that information is used only at child process creation time. After that, nobody cares who the parent was. If Windows doesn't use it, it's hard to say that manipulating it is a security vulnerability. You're fiddling with a knob that isn't connected to anything.

Raymond Chen

Follow