It rather involved being on the other side of this airtight hatchway: Invalid parameters from one security level crashing code at the same security level (again)

devblogs.microsoft.com/oldnewthing/20140402-00

April 2, 2014

Raymond Chen

A few years after I posted this story, the security team received something very similar.

> If have found that if you call the XYZ function (whose last parameter is supposed to be a pointer to a `DWORD`) and instead of passing a value pointer to a `DWORD`, you pass `NULL`, then you can trigger an access violation in the XYZ function. The XYZ function does not check whether the input parameter is `NULL`. This is a denial of service attack against the system.

Okay, first of all, even if the XYZ function checked that the final parameter is non-`NULL`, that wouldn't prevent a caller from passing an invalid non-`NULL` pointer, so adding a `NULL` check doesn't accomplish much from a security-theoretical standpoint.

The problem with this vulnerability report is that there is no elevation. The attack code and the code that crashes are on the same side of the airtight hatchway. If your goal was to make the process crash, then instead of passing invalid parameters to the XYZ function, you can just trigger the crash yourself.

```
int __cdecl main(int, char**)
{
    return *(DWORD*)NULL = 0;
}
```

In other words, in order to trigger an access violation in the XYZ function, you must already have had enough privilege to run code, which means that you already have enough privilege to trigger an access violation without even needing the help of the XYZ function.

This dubious vulnerability falls into the category *Code execution results in code execution*.

Raymond Chen

Follow