Application crash reported as security vulnerability, but you never crossed the airtight hatchway

devblogs.microsoft.com/oldnewthing/20170421-00

April 21, 2017

Raymond Chen

A security vulnerability report came in that said roughly this:

> There is a use-after-free vulnerability in the XYZ component which can be triggered as follows:
>
> - Run the XYZ application.
> - From the X menu, select Y.
> - In the resulting dialog, check Z, clear the checkbox on W, and twiddle your nose.
> - When you click OK, the application crashes because it is using a pointer to freed memory.

Thanks for finding a bug in the XYZ application, but is it a security vulnerability?

We look at the usual questions: Who is the attacker? Who is the victim? What privileges have been gained?

There is no remote aspect to this attack. In order to launch this attack, the user needs to run the XYZ application and manually go through a series of steps. It's not like a bad guy can send a specially-crafted file and lure a victim to encounter this crash. A bad guy would have to send instructions to a victim and socially-engineer them into following them, and even then, the bad guy doesn't gain anything yet. The bad guy would also have to socially-engineer the user into doing whatever steps are necessary in order to drop a special vtable into the freed memory. If you are a bad guy who has this much power over a victim that you can get them to type anything you want, then you don't need the XYZ application at all. You can tell the victim, "Download this executable and run it."

Okay, so maybe the attacker isn't remote? Maybe the attacker is malware that is already running at medium integrity, and it's going to try to use this attack to gain additional privileges from an instance of the XYZ application that is running elevated.[1] But that doesn't work because User Interface Privilege Isolation will not let a medium-integrity process manipulate the UI of a high-integrity process. In order for this to work, the malware would

have to socially-engineer the user into setting up the vtable on the heap, at which point you may as well just socially-engineer the user into running your malware with administrative privileges.

As far as we can tell, it looks like what you have there is a simple bug. This bug was introduced in a Windows Insider build; it never reached general availability. And the bug was already identified by the team and fixed in the next Windows Insider build.

The finder submitted a lengthy document explaining the alleged attack and attempting to identify the root cause. It seems that this is their bread and butter, because their document appeared to follow a template. The chapter after the bug was identified and root-caused was titled “Security implications”.

That chapter was blank.

[1] The finder never mentioned such a scenario, but we try to give the finder the benefit of the doubt. For example, if you misspell a file name, we will assume that you simply had a typo in your report, and we will fix the typo for you.

Raymond Chen

Follow