Evaluating the security consequences of an instance of reading past the end of a buffer

devblogs.microsoft.com/oldnewthing/20171005-00

October 5, 2017

Raymond Chen

A security vulnerability report came in that laid out the following scenario: By performing the correct sequence of operations in a program's UI, you can cause code to miscalculate the location of a string and display a string in an error dialog box where the string points past the end of a buffer. The finder reported this as a remote code execution vulnerability.

Let's evaluate the severity of this issue.

First of all, there is no writing going on. The string is read from memory and displayed on the screen in a dialog box. The string is not written to, and the contents of the string are not used to control where data gets written. Furthermore, the contents of the string are not used to determine what code gets executed.[1] There is no opportunity to mutate values in memory or cause unusual code to be executed. The claim of remote code execution seems to be unjustified. (They never did explain how this could lead to remote code execution; they just reported it as such.)

What kind of vulnerability do we have, then? Well, if the miscalculated pointer happens to point to a memory block that does not contain any null characters before reaching an invalid page, then the code that renders the string will take an access violation trying to read the string. That's a denial of service.

The miscalculated string will print a garbage string from the process's memory. If the attacker can arrange for the miscalculation to point to some memory of interest, they can extract data from the process by reading it from the dialog box. That's information disclosure.

Okay, so how bad are these issues?

Recall that in order to trigger the issue, the user needs to interact with the program in just the right way. This means either that the attacker has to socially-engineer the victim into performing those unusual operations, or the attacker has sufficient privileges to automate those operations. But if you have sufficient charisma to socially-engineer the victim into

performing those operations, or you have sufficient privileges to perform automation, then why are you wasting your time attacking this program? Just trick the user into typing or automate `Win`+`R` `\\hacker- site\ exploit\ pwnz0rd.exe` `Enter` and start counting your money.

How bad is the information disclosure? Well, who is the information being disclosed to? The information is displayed on the screen in a dialog box. It's not copied to another location within the program that might be disclosed further, nor is it sent over the network or written to a file. You're disclosing the information to the user who is running the program. This is in general not particularly interesting in the case where you are showing the user information that they already have access to.

In order for the information to leave the computer, somebody would have to screen-scrape it. But if somebody has the ability to screen-scrape your computer as you're using it, they are getting far more valuable information than some bytes of memory from this one program.

In other words, in order for these issues to become security issues, the attacker must already have significant powers on the computer under attack, so much so that they could use those powers to do far more valuable things than get a program to display garbage on the screen.

We thanked the finder for their report but indicated that what they had found was a bug, not a meaningful security vulnerability.

[1] Well, technically it controls what the font rendering engine does, because you're printing different characters. Theoretically, if there's a bug in the font rendering engine where, say, something bad happens if you ask it to draw a particular character, you could try to arrange for that character to be in the garbage string. Of course, an easier way would be to use that character in, say, the name of your file, so the font rendering engine crashes when it tries to put the name of the file in the title bar of the window.

Raymond Chen

Follow