Corrupted file causes application to crash; is that a security vulnerability?

devblogs.microsoft.com/oldnewthing/20150723-00

July 23, 2015

Raymond Chen

A security vulnerability report came in that went something like this:

> We have found a vulnerability in the XYZ application when it opens the attached corrupted file. The error message says, "Unhandled exception: System.OverflowException. Value was either too large or too small for an Int16." For a nominal subscription fee, you can learn about similar vulnerabilities in Microsoft products in the future.

Okay, so there is a flaw in the XYZ application where a file that is corrupted in a specific way causes it to suffer an unhandled exception trying to load the file.

That's definitely a bug, and thanks for reporting it, but is it a security vulnerability?

The attack here is that you create one of these corrupted files and you trick somebody into opening it. And then when they open it, the XYZ application crashes. The fact that an `OverflowException` was raised strongly suggests that the application was diligent enough to do its file parsing under the `checked` keyword, or that the entire module was compiled with the `/checked` compiler option, so that any overflow or out-of-range errors raise an exception rather than being ignored. That way, the overflow cannot be used as a vector to another attack.

What is missing from the story is that nobody was set up to catch the overflow exception, so the corrupted file resulted in the entire application crashing rather than some sort of "Sorry, this file is corrupted" error being displayed.

Okay, so let's assess the situation. What have you gained by tricking somebody into opening the file? The program detects that the file is corrupted and crashes instead of using the values in it. There is no code injection because the overflow is detected at the point it occurs, before any decisions are made based on the overflowed value. Consequently, there is no elevation of privilege. All you got was a denial of service against the XYZ application. (The overflow checking did its job and stopped processing as soon as corruption was detected.)

There isn't even data loss, because the problem occurred while loading up the corrupted file. It's not like the XYZ application had any old unsaved data.

At the end of the day, the worst you can do with this crash is annoy somebody.

Here's another way you can annoy somebody: Send them a copy of onestop.mid.

Raymond Chen

Follow