A trio of dubious denial-of-service security vulnerability reports which are just style points piled on top of nothing

devblogs.microsoft.com/oldnewthing/20230117-00

January 17, 2023

Raymond Chen

A security vulnerability report arrived that showed that if you loaded a specially-crafted file into a file viewer, the file viewer crashed. This was filed as a denial of service against the file viewer.

And indeed, the crafted file did cause the viewer to crash: The file format internally consists of several recursive structures, and in the crafted file, those structures contained loops. The viewer didn't have any detector for infinite loops, so its recursive parser crashed with a stack overflow.

Let's run through the usual questions.

Who is the attacker? The attacker is presumably somebody who sends you a crafted file and tricks you into viewing it.

Who is the victim? The victim is the person running the viewer and is unable to view the file.

What has the attacker gained? The attacker prevented the victim from viewing the file.

Now, another way to prevent the victim from being able to use the viewer to view a file is to *send them a file full of garbage*, or even simpler: *Don't send them the file in the first place*. But that's so obviously pointless that it's presumably not what the finder was reporting.

Maybe what the attacker gained is that the victim's file viewer program crashed, and the victim therefore lost any unsaved data. But this is a file viewer program. There is no unsaved data at all, since it never modifies the file. All the victim has to do to recover is to relaunch the file viewer program. The crash is not persistent, and it's certainly not unrecoverable. You recover by just running the program again!

Therefore, all the attacker really accomplished was to annoy the user briefly. They could also have done that by tricking the user into viewing a file that contained the message "Ha ha, I annoyed you!"

A similar security vulnerability report came in that claimed to have identified a denial of service attack against a file editor: Loading a crafted file into the file editor caused the file editor to crash.

What happened is that the file is corrupted in a specific way that causes a helper function to say, "Nope, I can't parse this," and return a null pointer to represent the error. But the caller of the helper function didn't check whether the call succeeded and tried to use the resulting null pointer. Therefore, this is a guaranteed null pointer crash, with no ability to control the read-from pointer. No chance for memory modification or information disclosure. Worst case is data loss and denial of service.

But does it even get you that much?

Before you load a new file, this particular program will prompt you to save your changes to the old file before it closes the old file and opens the new one. So there is no loss of data here: The user was given a chance to save their unsaved data. Any unsaved data would have been lost anyway even if the new file were not corrupted.

That leaves denial of service. The program crashes trying to load the crafted file, but the user can just launch it again. There is no permanent corruption, the relaunch of the program proceeds as usual. This is just a bug in the file editor program. The user will say, "Oh, well. I guess I'm not opening that file any more." All you managed to do was annoy the user momentarily.

A third security vulnerability report came in that claimed to have identified a denial of service attack against Windows logon. The finder claimed that if a user set up their user profile with a crafted image with a crafted file name, then the image would come out garbled on the logon screen. The finder included a copy of the problematic image, as well as instructions on how to set it as the profile picture.

What the finder didn't explain was why this was a denial of service, or why there was really any issue at all.

I guess an attacker can set a crafted image as their profile image, thereby preventing their image from displaying correctly on the logon screen. But so what? Just create an image filled with static or other garbage, and set it as your profile image. But why stop at garbage? If you don't want people to know what you look like, then upload a blank picture, or a picture of a cow. You can even try to impersonate somebody else: Upload a picture of a celebrity, or your boss!

Who is the attacker? The attacker is presumbly the person setting the crafted profile picture.

Who is the victim? The victim is, I guess, everybody else who is looking at the profile picture.

What has the attacker gained? Unclear. This doesn't let them do anything they couldn't already do by much more conventional means.

Nothing has been gained here. Nobody is even slightly annoyed![1]

In all of these cases, what the finder identified was a bug. Thanks for reporting the bug, and we've assigned it to the component owners. But these bugs have no security consequences. You could have gotten the same result by just using a file full of random garbage.

[1] Okay, if you upload a picture of your boss, you might annoy people who mistake you for your boss. And you might annoy your boss.