Cloning a Windows Runtime vector in the face of possible concurrent modification, denial of service?

devblogs.microsoft.com/oldnewthing/20230717-00

July 17, 2023

Raymond Chen

Last time, we cloned a Windows Runtime vector in the face of possible concurrent modification, adding support for the case where the underlying type is `bool`, and we deferred the question of whether our loop offered the opportunity for a denial of service attack.

If the attacker is running in the same process, then the issue is not particularly interesting: The in-process attacker is *already in the process*. so they can hang the program by just going into an infinite loop themselves.[1]

If the attacker is out-of-process, the next thing to study is who implemented the `IVector` that is being cloned. If the attacker implemented it, then they can give you a pathological vector, like one that changes size each time you ask it. If you don't know who implemented the vector, you shouldn't go into a loop. You should just give up at the first sign of trouble and propagate the error back to the caller for them to deal with.[2]

If the attacker is out-of-process, but they are using a vector that you provided (so you know it is not pathological), then the worst they can do is keep changing the size of the vector so you loop forever. This requires them to have the ability to burn 100% CPU in their attack process. So what they have done is leverage 100% CPU in one process into 100% CPU in another process.

Is that a problem?

Maybe.

If your process exists solely to provide services to that other process, then the attacker is just denying service to themselves. It's like buying a book and then tearing the pages out. "Haha, now nobody can read this book!" Dude, that's your book.

If your process provides services to multiple other processes, then you may have a concern, because it means that one bad client can prevent other clients from making progress.

On the other hand, the rogue client could also deny service to others by simply saturating your process with valid requests, so arguably they haven't really gained anything. The real thing to worry about is whether they can trigger a large problem with relatively small effort. If creating the problem requires as much effort as the problem itself, then they haven't really gained much.

[1] If the in-process component is externally controllable, say because it's running a script that was downloaded from the Internet, then you'll have to do a security analysis on the script engine, or you can just treat it as an out-of-process attacker.

[2] In many cases, the caller is the out-of-process code that provided the suspicious vector in the first place, so you can just propagate a "state changed" error back to the caller. Give them a taste of their own medicine.