You can't rule out a total breakdown of normal functioning, because a total breakdown of normal functioning could manifest itself as anything

devblogs.microsoft.com/oldnewthing/20120906-00

September 6, 2012

Raymond Chen

A customer was attempting to study a problem that their analysis traced back to the `malloc` function returning `NULL`.

> Is it a valid conclusion that there is no heap corruption?

While heap corruption may not be the avenue of investigation you'd first pursue, you can't rule it out. In the presence of a total breakdown of normal functioning, anything can happen, including appearing to be some other type of failure entirely. For example, the heap corruption might have corrupted the bookkeeping data in such a way as to make the heap behave as if it were a fixed-sized heap, say by corrupting the location where the heap manager remembered the `dwMaximumSize` parameter and changing it from zero to nonzero. Now, the next time the heap manager wants to expand the heap, it sees that the heap is no longer expandable and returns `NULL`. Or maybe the heap corruption tricked the heap manager into thinking that it was operating under low resource simulation, so it returned `NULL` even though there was plenty of memory available. Remember, once you've entered the realm of undefined behavior, *anything* is possible. Heck, one possible response to heap corruption is the installation of a rootkit. After all, that's how more advanced classes of malware work. They exploit a vulnerability to nudge a process into a subtle failure mode, and then push the failure mode over the edge into a breakdown, and then exploit the breakdown to get themselves installed onto your system, and then cover their tracks so you don't realize you've been pwned. Maybe the heap was corrupted in a way that cause a rootkit to become installed, and the rootkit patched the `malloc` function so it returned `NULL`. Like I said earlier, the possibility of heap corruption is probably not the avenue I would investigate first. But you can't rule it out either.

Bonus chatter: Since heap corruption can in principle lead to *anything*, any bug that results in heap corruption automatically gets a default classification of *Arbitrary Code Execution*, and if the heap corruption can be triggered via the network, it gets an automatic default classification of *Remote Code Execution* (RCE). Even if the likelihood of transforming

the heap corruption into remote code execution is exceedingly low, you still have to classify it as RCE until you can rule out all possibility of code execution. (And it is extremely rare that one can successfully prove that a heap overflow is not exploitable under any possible conditions.)

Raymond Chen

Follow