If I'm not supposed to call IsBadXxxPtr, how can I check if a pointer is bad?

devblogs.microsoft.com/oldnewthing/20100723-00

July 23, 2010

Raymond Chen

Some time ago, I opined that `IsBadXxxPtr` should really be called `CrashProgramRandomly` and you really should just let the program crash if somebody passes you a bad pointer. It is common to put pointer validation code at the start of functions for debugging purposes (as long as you don't make logic decisions based on whether the pointer is valid). But if you can't use `IsBadXxxPtr`, how can you validate the pointer?

Well, to validate a write pointer, write to it. To validate a read pointer, read from it. If the pointer is invalid, you'll crash, and at a predictable location, before the function has gotten halfway through its processing (making post-mortem debugging more difficult). Here are the functions I used:

```
// Make sure to disable compiler optimizations in these functions
// so the code won't be removed by the optimizer.
void DebugValidateWritePtr(void *p, size_t cb)
{
 memcpy(p, p, cb);
}
void DebugValidateReadPtr(void *p, size_t cb)
{
 memcmp(p, p, cb);
}
```

To verify that a buffer can be written to, we write to it by copying it to itself. Similarly, to verify that a buffer can be read, we read from it by comparing it to itself. The result of the operation is not important; we are interested in the side-effect of the memory access itself.

Note that the `DebugValidateWritePtr` function is not thread-safe: If another thread modifies the buffer while we are copying it to itself, the write may be lost. But code that does this violates one of the ground rules for programming (specifically the parameter stability requirements). Of course, if your function has specific behavior requirements beyond the ground rules, then that helper function may not work for you. I'm just putting it out there as a courtesy.

Raymond Chen

Follow