Why is my x64 process getting heap address above 4GB on Windows 8?

devblogs.microsoft.com/oldnewthing/20150709-00

July 9, 2015

Raymond Chen

A customer noticed that when they ran their program on Windows 8, memory allocations were being returned above the 4GB boundary. They included a simple test program:

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv)
{
    void *testbuffer = malloc(256);
    printf("Allocated address = %p\n", testbuffer);
    return 0;
}
```

When run on Windows 7, the function prints addresses like `0000000000179B00`, but on Windows 8, it prints addresses like `00000086E60EA410`.

The customer added that they care about this difference because pointers above 4GB will be corrupted when the value is truncated to a 32-bit value. As part of their experimentation, they found that they could force pointers above 4GB to occur even on Windows 7 by allocating very large chunks of memory, but on Windows 8, it's happening right off the bat.

The memory management team explained that this is expected for applications linked with the `/HIGHENTROPYVA` flag, which the Visual Studio linker enables by default for 64-bit programs.

High-entropy virtual address space is more commonly known as Address Space Layout Randomization (ASLR). ASLR is a feature that makes addresses in your program less predictable, which significantly improves its resiliance to many categories of security attacks. Windows 8 expands the scope of ASLR beyond just the code pages in your process so that it also randomizes where the heap goes.

The customer accepted that answer, and that was the end of the conversation, but there was something in this exchange that bothered me: The bit about truncating to a 32-bit value.

Why are they truncating 64-bit pointers to 32-bit values? That's the bug right there. And they even admit that they can trigger the bug by forcing the program to allocate a lot of memory. They need to stop truncating pointers! Once they do that, all the problems will go away, and it won't matter where the memory gets allocated.

If there is some fundamental reason that they have to truncate pointers to 32-bit values, then they should build without `/LARGEADDRESSAWARE` so that the process will be given an address space of only 2GB, and then they can truncate their pointers all they want.

(Of course, if you're going to do that, then you probably should just compile the program as a 32-bit program, since you're not really gaining much from being a 64-bit program any more.)

Raymond Chen

Follow