If you pass enough random numbers, eventually one of them will look valid

devblogs.microsoft.com/oldnewthing/20071026-00

October 26, 2007

Raymond Chen

One customer traced a problem they were having to the way they were calling a function similar in spirit to this one:

```
HGLOBAL CopyClipboardData(UINT cf)
{
 HGLOBAL hglob = NULL;
 HANDLE h = GetClipboardData(cf);
 if (h) {
  void *p = GlobalLock(h);
  if (p) {
   SIZE_T size = GlobalSize(h);
   hglob = GlobalAlloc(GMEM_FIXED, size);
   if (hglob) {
    CopyMemory(hglob, p, size);
   }
   GlobalUnlock(h);
  }
 }
 return hglob;
}
```

This function takes a clipboard format and looks for it on the clipboard. If found, it returns a copy of the data.

Looks great, huh?

The problem is that the customer would sometimes call the function as `CopyClipboardData(CF_BITMAP)`. The `CF_BITMAP` clipboard format stores its contents in the form of a `HBITMAP`, not an `HGLOBAL`.

The question from the customer:

> This code was written in 2002, and we are wondering why it works "most" of the time and crashes sporadically. We expected that the call to `GlobalLock` would fail with an invalid parameter error, but sometimes it succeeds, and then when we call `GlobalSize` we crash. Why does it crash sometimes?

You already know the answer to this. `GlobalAlloc` works closely with `GlobalLock` so that `GlobalLock` can be fast. The bitmap handle returned by `GetClipboardData` usually fails the quick tests performed by `GlobalLock` to see whether the parameter is a fixed memory block, in which case the `GlobalLock` must go down its slow code path, and it is in this slow code path that the function recognizes that the the handle is downright invalid.

But once in a rare while, the bitmap handle happens to smell just enough like a fixed global handle that it passes the tests, and `GlobalLock` uses its highly optimized code path where it says, "Okay, this is one of those fixed global handles that `GlobalAlloc` created for me. I can just return the pointer back." Result: The call to `GlobalLock` succeeds (garbage in, garbage out), and then you crash in the `GlobalSize` function where it tries to use the `HBITMAP` as if it were a `HGLOBAL` and access some of the memory block metadata, which isn't there since the handle isn't valid after all.

The bitmap handle is basically a random number from the global heap's point of view, since it's just some number that some other component made up. It's not a global handle. If you generate enough random numbers, eventually one of them will look like a valid parameter.

Raymond Chen

Follow