If you're handling an out of memory exception, you probably shouldn't allocate memory

devblogs.microsoft.com/oldnewthing/20090911-00

September 11, 2009

Raymond Chen

With the assistance of Application Verifier, specifically, *low resource simulation* (also known as *fault injection*), a tester found a stack overflow condition. As we learned earlier, the important thing to look at when studying a stack overflow is the repeating section.

```
Contoso!Error::ThrowError+0x39
Contoso!Str::Set+0x35
Contoso!Win32::OpenModuleName+0x54
Contoso!StackTrace::StackEntry::FindModuleInfo+0x1b
Contoso!StackTrace::CreateTrace+0x2c
Contoso!StackTrace::StackTrace+0x4f
Contoso!Error::Error+0x1f
```

When this stack trace was shown to the development team, they instantly recognized the cause of the problem. And you also have enough information to figure it out, too.

Hint: Of the most likely reasons that a method named `Str::Set` would throw an error, which of them match the scenario?

Since we are simulating low resources, the error being thrown is most likely an out of memory error.

Reading the stack dump, the constructor for the `Error` object builds a stack trace object, and the stack trace object tries to allocate memory for a string in order to do its job. But that memory allocation fails, because we are out of memory, so an `Error` object is thrown, which builds a stack trace, which encounters an out of memory error, and so on.

Obviously, the mistake was allocating memory as part of the process of reporting an out of memory condition. You need to be careful to avoid generating the very error that caused your error handler to be called.

Related topics:

Raymond Chen

Follow