Misinterpreting the misleadingly-named STATUS_STACK_BUFFER_OVERRUN

devblogs.microsoft.com/oldnewthing/20230731-00

July 31, 2023

Raymond Chen

I noted some time ago that `STATUS_STACK_BUFFER_OVERRUN` doesn't mean that there was a stack buffer overrun, although that's what it meant at first. Later, the status code was broadened to mean "Program self-triggered abnormal termination", but it was too late to change the name.

A security vulnerability report came in that went like this:

> I have found a stack overflow bug in Explorer. This stack overflow occurs in ucrtbase.dll and Windows.UI.FileExplorer.dll. Since it occurs in Explorer, this can be exploited to escalate privileges. To reproduce, download the attached ZIP file and right-click it. I have also attached Explorer crash dumps for analysis.
>
> ```
> EXCEPTION_RECORD: (.exr -1)
> ExceptionAddress: 00007ffcaa19dd7e (ucrtbase!abort+0x000000000000004e)
> ExceptionCode: c0000409 (Security check failure or stack buffer overrun)
> ExceptionFlags: 00000001
> NumberParameters: 1
> Parameter[0]: 0000000000000007
> Subcode: 0x7 FAST_FAIL_FATAL_APP_EXIT
>
> STACK_TEXT:
> ucrtbase!abort+0x4e
> ucrtbase!terminate+0x29
> ucrtbase!__crt_state_management::wrapped_invoke<void (__cdecl*)(void)
> noexcept,void>+0xf
> explorer!_scrt_unhandled_exception_filter+0x5a
> KERNELBASE!UnhandledExceptionFilter+0x1f1
> ntdll!LdrpLogFatalUserCallbackException+0xa2
> ntdll!KiUserCallbackDispatcherHandler+0x20
> ntdll!RtlpExecuteHandlerForException+0xf
> ntdll!RtlDispatchException+0x25a
> ntdll!RtlRaiseException+0x163
> KERNELBASE!RaiseException+0x6c
> msvcr90!_CxxThrowException+0x86
> contoso+0x104d
> contososhellext!OnInvokeCommand+0x548
> contososhellext!OnInvokeCommand+0x24d2a
> contososhellext!OnGetCommandString+0x4f
> contoso+0x2abd
> contoso+0x27f0
> shell32!CDefFolderMenu::GetCommandString+0x1f6
> shell32!CDefFolderMenu::_UnduplicateVerbs+0x31f
> shell32!CDefFolderMenu::QueryContextMenu+0x7d2
> shell32!CDefView::_DoContextMenuPopup+0x2f7
> shell32!CDefView::OnSelectionContextMenu+0x85
> explorerframe!UIItemsView::ShowContextMenu+0x378
> explorerframe!CItemsView::ShowContextMenu+0x17
> shell32!CDefView::_DoContextMenu+0x92
> shell32!CDefView::_OnContextMenu+0xec
> shell32!CDefView::WndProc+0x718
> shell32!CDefView::s_WndProc+0x5c
> user32!UserCallWinProcCheckWow+0x33c
> user32!CallWindowProcW+0x8e
> ```

From the exception record, we see that this was a fast-fail: `FAST_FAIL_FATAL_APP_EXIT`.

From the stack trace we can see that this was due to an unhandled C++ exception thrown by Contoso: Contoso called `_CxxThrowException`, and this ended up reaching the `_scrt_unhandled_exception_filter`, which decided to `terminate` the process.

The problem is therefore with the Contoso shell extension: When you right-click this file, the Contoso shell extension throws a C++ exception which it does not handle.

C++ exceptions cannot be thrown across the ABI boundary because there's no requirement in the ABI that the calling code even be written in C++ at all![1] And certainly unhandled C++ exceptions are really bad ideas.

There is a bug here, but the bug is in the Contoso shell extension, not in Explorer. Such is the punishment for allowing third party extensibility: Any bug in a third party extension manifests itself as a bug in Explorer.

[1] For example, the Windows 95 shell was written in C.