Abstract

Crashed ! Now What ? “It works on my machine” :) Those little words that make the heart of QAs and clients skip a beat. Sometimes reproducing a crash on a developer's machine is next to impossible. Most of the time remote debugging is out of the question and all you’re left with are some scant log files and maybe a memory dump file, if you’re lucky. Wouldn’t you like to know the exact point of failure in the program and how it got there, on the client’s PC ?

How can you get your hands on a StackTrace of that crash on the client’s machine ? And how can you make any sense of it without symbols (client deployed Release build) ?

In this session, I’ll present a Windows specific technique we developed, that my team uses regularly to debug such scenarios in production. We leverage OS APIs like the Image Help Library (ImageHlp.dll), the Debug Help Library (DbgHelp.dll) to work with PE/COFF images and PDBs and reconstruct symbolicated StackTraces for Release crashes in production. The technique and APIs work all the way from Windows XP up to Windows 10, both for x86 and x64 executables.

We’ll see how symbols are loaded and how PDBs work, we’ll discuss partial/incremental PDBs and we’ll have to get comfortable with Structured Exception Handling (SEH). Did I mention Address Space Layout Randomization (ASLR) ? This is going to be fun :)

Come with me on this journey and we’ll walk the stack together, to reconstruct each frame, from a few pointers and some symbols.