# Adventures in application compatibility: Calling an internal function

devblogs.microsoft.com/oldnewthing/20190724-00

July 24, 2019

Raymond Chen

We try hard to make sure applications continue to work, but some things that applications do are so egregious that there's no practical way of getting them to work.

Today, we'll learn about one such.

The program bills itself as "the most advanced Windows optimization toolkit in the universe!"

If you say so.

One of their awesome optimizations, it appears, is to reset file associations to match their concept of what file associations should be in an ideal world. This ideal world probably is one in which their application is the default handler for a lot of popular and contentious file types.

The application compatibility team reported that this program crashed when you asked it to reset file associations. Windows goes to some lengths to make it hard for programs to change file associations programmatically, and instead of trying to reverse-engineer how Windows protects the settings in the registry, they instead opted to reverse-engineer the code that manages the settings.

Specifically, they scanned memory looking for the internal function that sets the file associations, and then called it.

Now, searching all of memory is a daunting task, but they were able to take a shortcut: They got their hands on an `IApplicationAssociationRegistration` object, which is the documented interface for managing application defaults. They used the vtable as a clue as to where the application defaults management code is, and focused their search on that region of memory. I'm not quite sure exactly how they found the internal function; perhaps they disassembled the code looking for `call` instructions, and assumed that the third `call` (say) was to a handy function, and then they disassembled the handy function and assumed that the second `call` (say) was to the secret internal function.

Of course, searching memory for a function to call is not exactly something documented and supported. Windows made some changes to how these functions operate, and that threw off their code that grovels the binary, and they ended up calling the wrong function.

Instead of creating a decoy that keeps their crazy algorithm working, the team opted to let the program crash when you pushed the button to reset file associations to their ideal state. This was an older version of a program still under active development, and the failure mode made it rather clear to the user that the program was at fault: It crashes when you press a specific button. The initial inclination is to blame that button. Therefore, the user will contact the vendor for an update.

Now that everything is online, shifting the cost of a vendor's mistake to the vendor's support infrastructure has become a viable alternative to patching the operating system to work around a single program.

Raymond Chen

**Follow**