How can I debug a function that has been subjected to COMDAT folding?

devblogs.microsoft.com/oldnewthing/20160815-00

August 15, 2016

Raymond Chen

Suppose you want to set a breakpoint on a function, but you find that the function has been subjected to COMDAT folding, so your attempt to set a breakpoint on the function ends up setting a breakpoint on some other function, and your breakpoint ends up firing when either your desired function or the other identical function gets called. How can you get your breakpoint to fire only on the function you are debugging?

One way to do this is to disable COMDAT folding temporarily and rebuild. Mind you, this may result in your binary size exploding, but since you're just debugging, this probably doesn't bother you that much. On the other hand, there may be parts of the program that are relying on COMDAT folding, or it may be hard to find the build setting that controls COMDAT folding, and you run the risk of forgetting to change the setting back and accidentally committing a change that disables COMDAT folding!

The easy way is to mutate the function. Add a call to a harmless function like `GetTickCount()`. Note that the harmless function must be something the compiler can't optimize out, so don't try `free(nullptr)` because the compiler is allowed to take advantage of the fact that `free(nullptr)` is required by the language standard to have no effect and can consequently optimize the call out entirely.

Then again, if you're going to mutate the function, you may as well mutate it in a way that makes debugging easier. For example, you might add

```
bool breakpoint = false;

void TheFunction()
{
 if (breakpoint) DebugBreak();
 ... rest of function ...
}
```

Then you can patch the breakpoint variable to `true` and boom, there's your breakpoint.

If you can't recompile the binary, then your options are more limited. If the set of callers is manageable, you could try setting a breakpoint on each of the callers. Or if there is something in the function that lets you detect which identical function you're in, you can use that. For example, maybe the two functions are

```
class Circle
{
public:
   virtual int GetRadius() { return m_radius; }
private:
  int m_radius;
  int m_xcenter;
  int m_ycenter;
};

class Channel
{
public:
   int GetId() { return m_id; }
private:
  HANDLE m_signal;
  int m_id;
};
```

Since `Circle::GetRadius` and `Channel::GetId` compile to the same code, they will get COMDAT-folded. But you can still figure out which method you're in by looking at other parts of the `this` pointer. In the example above, you see that `Circle` has a virtual method, hence a vtable, so you can use a conditional breakpoint to check whether the vtable matches.

If you use a boring debugger, it might be something like this:

```
0:001> bp Circle::GetRadius "j poi(ecx)==0x10014270 r;g"
```

If you use a fancy debugger, then use your fancy debugger's conditional breakpoint facility.

Raymond Chen

Follow