

A Commercial Bug Case Study

Here's a case of finding and identifying a bug in commercial software, reporting it, and having it fixed in a way that seems odd at first but is in fact appropriate to address backwards compatibility issues.

I wrote a number of Flash-based tools and games recently using open source tools (the Mtasc compiler and the swfmill tool for building SWF files), and was dismayed when a new release of Adobe's Flash Player (version 10,0,12,36 on Windows XP, for example) caused my programs to occasionally hang.

My first thought was that the open source tools had made assumptions based on reverse engineering and that these assumptions had simply failed with the new version of the Flash player. Unfortunate for me.

I started to track down the problem and found that stored objects (SOL, or shared object library format) were losing data. Using a graphical editing tool, I tried to minimize an example and found that loss only occurred for a certain number of drawn items. The file size corresponding to that boundary was around 64kB. Odd.

A little poking around the web turned up various reverse-engineered versions of the SOL format, so I took a look at the broken files. In the format, bytes 2-5 contain the file size as a 32-bit big-endian integer. In the corrupted files, bytes 2 and 3 (the high 16 bits) were always 0, while bytes 4 and 5 were accurate. That is, the size was being truncated—perhaps someone had converted it to a 16-bit integer by accident?

I didn't know much about how my tools interacted with the Flash player, and I didn't really expect a response from Adobe if the bug were in an open source tool. I also noticed that a new version of the compiler was available, so I installed it, but it didn't solve the problem. I contacted the author of the compiler with a description of the bug, thinking that perhaps one of the support libraries for the compiler might be the source of the problem.

Then I took a look at the bug reports for Flash online, which Adobe keeps in an open bug-tracking interface. I found one report that seemed similar: SOLs "corrupted" on update. The report was fairly useless: copies of SOL files before some unspecified update as well as after, not including source code or anything that might help identify the problem. There was a response listed along with the report in the bug-tracking tools: a developer at Adobe had taken a look and had no idea what to do with such a report.

You might wonder how such a bug could escape notice and make its way into a release. An SOL file is a serialized list of named hierarchical objects. Only one sanity check seemed to be broken: when searching for a name, the Flash library stopped looking both when it reached the projected end of the file (from the size field) as well as the actual end of the file (from the file system). A lot of Flash code probably encapsulates stored objects in a single name before storing them in an SOL file, while other code might put all small objects at the start of a file and one large object at the end. In either case, false negatives—that is, not finding a name because the start of the object lies beyond the bound imposed by the corrupted size field—do not occur. One workaround that I considered was to rewrite all of my code to encapsulate data into a single object. However, doing so would invalidate all SOL files created by previous versions of my tools, and I was reluctant to commit myself to writing version identification and conversion code or to breaking backwards compatibility for my tools by changing the naming hierarchies in such a manner.

Rather than creating an Adobe account (required for participation in the bug tracker), I sent a detailed account of the bug to what seemed like the most likely e-mail address for the Adobe developer. I also sent a follow-up mail to the Mtasc author, telling him that it probably wasn't worth his time, since the bug was present in SWF code that didn't use Mtasc. He confirmed that such a bug couldn't be in Mtasc.

About a week later, I got a response from the Adobe developer. The response was thankful but non-committal, saying that they'd have to do more investigation. That was on 16 January, to give you an idea of how long things take given commercial release schedules.

This morning, I got an update notice on Flash (for Windows XP version 10,0,22,87—maybe I missed releases in between? I haven't been putting off updates) and decided to check for the bug. I created a large SOL file and re-loaded it. Success! The bug was gone.

Looking at the SOL file, though, the size field was still corrupt, in exactly the same way. Rather than fixing the size field, Adobe had removed the file format check that had caused my programs to crash!

Thinking about it a little more carefully, Adobe certainly had to remove the file format check. How many corrupt SOL files exist now due to these releases? How many more will be created before everyone updates their Flash players? Should all of those files simply be discarded? Customers might get unhappy.

Not fixing the SOL output is more difficult to understand. Was it really so hard to find and eliminate? Or did someone want to make sure that plenty of broken files were around so as to provide better regression testing? Hard to say.