

Re: Two questions

Source: <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2005-06/msg00351.html>

- *From:* Peter Hansen <peter@xxxxxxxxxxx>
 - *Date:* Thu, 02 Jun 2005 12:46:17 -0400
-

rbt wrote:

Peter Hansen wrote:

Philosophy not entirely aside, you should note that object code in any language can "easily" be reverse-engineered in the same way, with the only difference being the degree of ease involved. If the code is worth enough to someone that they are willing to risk violating your license terms, they *will* be able to recover enough source code (whether it was Python, C, or assembly) to do what they need.

Don't intend to hijack this thread, but this bit interests me. I know several accomplished C/assembly programmers who have told me that reverse engineering object code from either of these two languages is anything but trivial. Yet, I *hear* and *read* the opposite all of the time. Can anyone actually demonstrate a decompile that mirrors the original source?

It all depends on the definition of "reverse engineering".

In my opinion and experience, very little code in the world is so sophisticated that it is not roughly as easy to write equivalent code from scratch (with the original, working program as a guide of what the code actually does) as it would be to convert the object code back into source. (Exceptions such as "decompyle" which may make the job near trivial aside.)

If that's true, it leaves us with a very small subset of the code in any given program, that might actually be worth the effort of converting back to source. That bit of code will generally turn out to be so small that once again an automated conversion to source is not really necessary, since analysis of the object code would with relatively little effort allow one to "reverse engineer" some equivalent source, in whatever

Re: Two questions

language (or pseudo-code) one chose.

So, for languages like C, where the compilation process is fairly "destructive" to things like the variable names and control structures used, "reverse engineering" in the sense of "automated conversion back to equivalent source code" is rather difficult, probably infeasibly so for most non-trivial programs. I personally don't understand the need for this, other than in the case of "I lost my source", and the correct answer there is a session of pummelling, followed by tarring and feathering with the pages of the Subversion Book.

On the other hand, "reverse engineering" in the sense of "creating source code capable of reproducing the effects of the valuable and interesting parts of the object code" is not that difficult, and in the sense of "understanding how this code works" is even easier, being just the first part of that step.

Software development is far more about choosing the right problems to solve and the right ways to solve them than it is about writing source code for the program that will do the job.

And if I had an automated tool to reproduce source code for a given program, I'd still be very concerned that I didn't end up with any of its automated test cases. ;-)

-Peter

.