NEWSANALYSIS

The Linux Voice view on what's going on in the world of Free Software.

Opinion

The code audit mirage

What does it take to keep open source code secure?

Simon Phipps is president of the Open Source Initiative and a board member of the Open Rights Group and of Open Source for America.

In the wake of Heartbleed and other serious bugs found in open source code, I keep hearing calls for "audit". For example, the European Commission is considering whether to budget for security audits as part of its encouragement of open source software adoption. That's something I never heard in all my years at technology corporations. We discussed testing strategies, deployed "pen testing", did automated code scans, used assertions in code to back up several of those test strategies, and so on. But "audit"? I never remember hearing it mentioned.

An audit is exactly the wrong strategy for open source. It spends money on finding fault without fixing anything. It reflects a worldview that open source software is a product someone else is responsible for, rather than a shared resource everyone is responsible for. If you don't trust the code, you should be pointing your finger at whoever deployed it, not at the code. The only time to fall back to an audit is when the collaborative developers refuse to co-operate, as happened with TrueCrypt.

All code has bugs. Secret code can also have intentional back-doors and undocumented functions. Audit is a reasonable way to give customers of this proprietary code confidence that it behaves in the way the vendor describes. But open source code is not a product until a vendor packages it as one. Until then it is a tool, waiting to be deployed by a suitably knowledgeable person.

What should we do?

First, we should be demanding of suppliers who deploy open source code as part of a deliverable with security aspects that *they* audit the code. If your supplier does not employ committers on the code they are trying to sell you, or at very least have a paid business relationship with someone who does, don't buy.

Second, as community developers we should be performing regular scans of the code we share using a tool like *Coverity Scan*. It's a free-of-charge proprietary offering that analyses even the largest codebase and identifies common programming errors like uninitialised pointers, buffer over-runs, unreachable code and so on. It produces a detailed, actionable report that community programmers can then cherry-pick to fix the serious issues identified. *Coverity Scan* also produces a useful code quality metric, which allows tracking of code improvement trends as well as highlighting regressions.

Third, we should keep an eye on the diversity of communities. A project that has very few different sources of external motivation for its committers – all the same employer, for example – can easily neglect the steps necessary to keep quality high.

Open source code is neither more nor less secure intrinsically -- all code has bugs, and sometimes they lead to security exposures and subsequently to exploits. But open source, properly maintained in a diverse community, can have them fixed faster and can use tools that publicly expose issues. Let's make the code better and not allow open source to be misrepresented.

> "An audit is exactly the wrong strategy – it spends money on finding fault without fixing anything."

One of the major reasons Heartbleed happened was a lack of new eyes coming into the OpenSLL project