The Box is Beautiful, but...

Michael Barwise

Richard Halton, chief executive of YouView was interviewed today on the BBC by Steve Hewlett (Radio4 Media Show 16/02/11). The company is developing a set-top box that combines digital TV with internet connectivity - potentially allowing all sorts of new services to be offered . But it appears from the radio programme that the cunning box currently keeps crashing. Now this is not a huge surprise - or necessarily even a huge problem - for a product that's still effectively in beta. What did strike me as a problem though - and a serious one - was Halton's response to Mr. Hewlett's questions about the state of development. He made two remarks: first, that the user interface is fantastic - it's only the underlying technology that still needs work, and second, that the product is the important thing - the technology is only an enabler. Mr. Halton is far from alone in taking this position - I've met it almost universally among the software developer community over the last few years. In fact it's the order of the day - but it scares me stiff.

No engineering discipline other than software development could get away with issuing so many products that don't work as promised, or with such a constant stream of fixes for thousands of design and implementation errors. Imagine the builders returning to your house every month for a period of several years to underpin a wall, refit the windows or replace a rafter or two. Notwithstanding the recent A380 super jumbo engine problems - an exceptionally rare occurrence - aircraft are not raining from the skies, as they would be if they were designed and assembled using the same sloppy engineering practice that's taken for granted in the consumer software industry.

Of course neither your house nor the plane is "internet enabled" - yet. But that's not the real issue. They are both designed from first principles and engineered by people who understand those principles. Software, on the other hand, is mostly developed rather on the same lines as assembling a flat-pack wardrobe - blindly plugging together highly abstracted components supplied by the development system vendor. The person coding the application never sees the machine level executable or even the internals of the high-level methods being used. That means a wholly different concept of the development process - broad brush strokes in place of attention to fine detail. But the real devil is always in the detail. Most of the dangerous exploitable bugs in software are very near the machine - mismanaged array indices, incorrect use of pointers, variable typing conflicts. And despite the emergence of development environments that automate or conceal such low-level stuff from the developer, these bugs keep occurring with appalling frequency.

I may be old-fashioned, but when I was taught engineering, functionality and robustness came first. No longer, it seems - the interface is everything. And of course, security comes last. There are already numerous reports of vulnerabilities in internet-connected TVs. We must hope that this offering - for which "developers can create apps that can make your TV do all sorts of things" - will be robust against such threats. They can backfire on service providers as well as on consumers, so it's probably worth the extra effort. As I said, the attitude of the consumer software developer community scares me. But the really scary thing is that systems developed on this same basis are now widely deployed in support of retail banking, critical national infrastructure and the military. Maybe we need to think again.

Originally appeared on the Infosecurity Network, February 2011