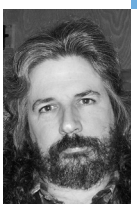
ROBERT G. FERRELL

/dev/random



Robert G. Ferrell is an information security geek biding his time until that genius grant finally comes through.

rgferrell@gmail.com

Patching the roof and pitching the hay
Is not my idea of a perfect day

—*Stephen Schwartz, "Pippin" (one of my all-time favorite musicals)*

IN THE BPC ERA (BEFORE PERSONAL computers), patching was something you did to squeeze a little more use out of a punctured, torn, or split thingamajig. It was no more a normal stage in the life cycle of an item than open-heart surgery can be said to constitute part of the normal life cycle of a human being. How, then, did periodic patching come to be accepted as part and parcel of the routine software experience? It's like a bonus we get unexpectedly days or weeks after the product is in place and functioning: "Look what came today, honey—the rest of the spokes for little Johnny's tricycle! Now he won't have to drag it around behind him anymore."

In virtually any other manufacturing arena (home office furniture and toys notably excepted), the repeated release to the public of products that they hadn't really finished assembling yet would be detrimental to the company's bottom line at some point. For reasons I've never fully fathomed, the market has not seen fit to apply this rather fundamental economic principle to software firms.

Let's examine this phenomenon a little more closely. In effect, a patch is an admission that the software you bought wasn't really well and thoroughly tested. It was rushed out the door with flaws the manufacturer felt motivated by their lawyers and public relations janks to correct at a later date. The weird part is that we the sheeple just accept this malfeasance as though it were a perfectly natural way of doing business, instead of stringing the perpetrators up in the mall food court for all to see and taunt.

Imagine if you got a package in the mail once a month that contained parts for your new car, the installation of which were necessary for it to continue to operate without, say, blowing up when you accelerate to a certain speed. Or what if every song you snagged off iTunes required you to download regular fixes for bad notes or missing lyrics? That adorable pedigreed puppy you just brought home from the breeder? They'll be sending you ointment you'll want to administer every so often or poochie will moo instead of bark. And then there are the shots you'll need to give her to keep that precious little tail from falling off or becoming dislocated when wagged.

A fair number of these patches are issued for security, or more accurately, insecurity reasons. The flaws they correct are for the most part well-known to the software engineering community, however, and should have been expunged during the quality review process. Part of the reason for this sorry state of affairs, pundits sympathetic to the software industry will be happy to expound in your general direction, is that modern software packages are so incredibly complex that no one could reasonably be expected to catch all the little nitpicking mistakes like, oh, I don't know, buffer overflows and null pointer dereferences. This argument is a steaming flagon of septic wallaby lymph because secure engineering starts with educating the coders themselves and auditing their code as it's produced, before it gets too deeply entwined with the rest of the application and has to be tweezed out like errant ear hair. Programmers are, or at least should be, taught not to make errors of this sort. It's as simple as that. Apologists try to make it sound as though insecure coding is a sort of congenital Tourette's Syndrome that affects most software engineers. We should pity them for their affliction and be supportive. If that means pushing a few dozen patches to a hundred million systems worldwide at a total cost of a couple thousand (wo) man-years of potential lost productivity and who knows how many terabits of wasted bandwidth, so be it. After all, other professionals aren't expected to learn their trades properly. Look at investment bankers. (But not too long: You're gazing into the abyss.)

To release yet another cacodaemon from the lurking horror of my metaphor petting zoo: ever watch one of those edutainment documentaries where they take you on a tour of the factory that makes, like, dismembered squid tentacle slices coated in a vaguely chocolateoid substance? Notice that there's always at least one or two hairnetted workers whose job it is to yank the moldy, scabrous, and otherwise obviously substandard appendage pieces off the conveyor before they get covered in brown trans-fat-laden goo and packaged up to be shipped to your child's school as a healthy snack alternative. That's called "quality control," and most experts agree it should be accomplished prior to the product actually leaving the place of manufacture. If certain software companies were in charge, they'd bide their time until some kids got food poisoning, then mail out little cups of disinfectant for consumers to dip their CalimariBars® in to kill any putative bacteria. The resulting taste bud damage? That's a feature, little lady.

To add insult to injury, a great many patches get foisted on the unsuspecting user via some insidious auto-update mechanism without so much as a by-your-leave and then break things that were working just fine before, thank you. Your car (probably) won't now blow up on the freeway entrance ramp, but the headlights switch on and off at random and the radio will only play easy-listening stations or talk shows on the Esperanto network. No worries, though: The next patch will make it impossible to roll the windows up, so you won't be able to hear the radio, anyway.

Having endured this tirade, you'd be excused for thinking that I have nothing good to say about the practice of patching. You'd be wrong, as it so frequently turns out. It saved my mother a lot of money on new jeans when I was a boy.