The "Patching" Mentality

Jeffrey Voas, IEEE Fellow

Software patches—we can't live without them, but it would be good if we could.

When I was growing up, "patching" was something a person did to physical entities, such as clothing or pipes, whenever holes manifested. I also knew that cloth patches were used to sew quilts.

In the IT community, the term *software patch* has been employed to define software modifications (fixes) that mitigate code security vulnerabilities or other reliability/performance problems after software has been released. Generally speaking, patches are bandages applied after release and during maintenance, and they are often continuous over the lifetime of a product. A software patch should mitigate the situation in which there was an original weakness, or at some point, software behavioral decay occurred (possibly created by a modification of the operational environment).

Although this all sounds simple enough, the concept of software patching still remains somewhat mysterious to me. In fact, I'm not even sure if the term patch is used much anymore. After all, not all software vendors wish to acknowledge original weaknesses and defects. Instead, it seems that terms like *update* and *upgrade* are now the new, preferred terms. So, have update and upgrade become replacement terms to move away from the uglier term of patch?

I've also always wondered about ideas, such as "building quality in" or "building security in." Is there a true business model for these concepts in a time-to-market world? After all, no one wants to be last to market. I believe that when it affects a brand, the answer is yes. But other than brand and stockholder protection, does it really matter? If the customer is willing to accept patch number 1, then patch number 2, and then patch number 3, and so on, why would an organization sacrifice time to market to offer excellence on day infinity (that is, the software product never gets released)? One might argue "yes" due to liability, but that argument is not strong. (And I don't even want to get into discussing patching patches.)

I ask these questions because I've been troubled for many years by an actual situation from 30 years ago in which a request for proposals was set up in such a way that the winning vendor was encouraged to bid on a job where the cost to perform the contract was greater than the revenue from the contract. So, who would bid on such a job? Well, as it turns out, many vendors because the profits were built into the maintenance phase to fix the problems built into the original deliverable. The lower the quality on the front end, the bigger the revenue/profits on the back end. That never sat well with me.

Digital Object Identifier 10.1109/MC.2020.2991277
Date of current version: 1 July 2020

 0018-9162/20©2020IEEE

EDITOR IN CHIEF **JEFFREY VOAS**
IEEE Fellow; j.voas@ieee.org

Patches are bandages applied after release and during maintenance, and they are often continuous over the lifetime of a product.

Automobile companies are an industry that understands the costs of recalls, not just in terms of fixing the vehicles that come back but in brand degradation. Does the software industry feel similar or even need to care?

To consider this question, just start from the simple fact that software is nonphysical. Physical is not easily malleable. Software is easily malleable. In my opinion, software's malleability is the grand enabler of the "patching mentality." Whether good or bad, it is what it is, and it's here to stay.

In summary, installing patches, updates, upgrades, or whatever you prefer to call it is a normalized practice today. I believe we all agree that defect-free software is not an option. Zero trust, as an operational philosophy, seems to be gaining momentum, but it has its own limits. In terms of patches to improve security, the release of new patches exacerbates successful attacks for those who do not install them immediately.

So, patches, we can't live without them, but it would great if we could. It's a strange business model. Something to ponder.

JEFFREY VOAS is the editor in chief of *Computer*. He is a Fellow of the IEEE. Contact him at j.voas@ieee.org.

Digital Object Identifier 10.1109/MC.2020.3000399