Unsafe Code at Any Speed?

This article was originally published on Fox Business on March 4, 2015.

by Brian E. Finch

Brian E. Finch
Public Policy
+1.202.663.8062
brian.finch@pillsburylaw.com

Brian E. Finch is a partner in Pillsbury's Public Policy practice in Washington, DC. He is a recognized authority on global security matters and is co-leader of the firm's Global Security practice.

Though I am loathe to associate myself to Ralph Nader, a cybersecurity discussion recently brought to mind his seminal work, *Unsafe at Any Speed*.

For those of you unfamiliar with Nader's book, it documented serious if not fatal design flaws in the Chevrolet Corvair, as well as serving as an indictment generally of the automobile industry. The main thrust of the book was that auto makers put driver and pedestrian safety at the bottom of their priority list, focusing more on costs and appearance.

What triggered memories of *Unsafe* was a recent article by Politico writer David Perara about a brewing software industry fight over "Zero Days." There is an interesting parallel between the two works that bears investigation, namely whether the software industry is also more focused on cost and style than safety/security.

In his article "*Google sparks Zero Day disclosure ethics debate*," Perera delves into detail about a decision by Google to reveal previously undiscovered software security vulnerabilities (known as "Zero Days") that its own researchers find in other companies' products.

According to Perara, Google (GOOGL) will notify companies of the newly discovered Zero Day, and give software companies 90 days to write and distribute a patch. At the end of the 90 days, regardless of whether a patch has been released, Google will publish the aforementioned software bug.

I want to highlight a key point here up front: it is not as if software developers simply release their product and ignore flaws, particularly ones related to security. In fact, software developers regularly issue fixes for problems they discover. Microsoft, for instance, has "Patch Tuesday." Typically falling on the second Tuesday of a month, Microsoft will release a variety of patches for flaws it (or others) have discovered in its software.

Still many argue that software developers are often slow in releasing patches, with cited instances where months passed between the time a Zero Day was identified and a patch was released. Perara noted an instance where a software vulnerability was discovered and the developer notified, yet seemingly no action was taken until the flaw was made public.

There are all sorts of issues surrounding the discovery of Zero Days, including whether individuals who discover them should simply notify the developer of the problem or ask for some form of payment in return for information. Some have also raised ethical concerns associated with the sale of Zero Days, as some individuals sell them to the highest bidder (which may include cyber criminals).

The U.S. government has occasionally been drawn into the debate, with questions surrounding whether it stores certain Zero Days for use in the event of a cyber conflict.

To me, the issue of Zero Days raises an equally compelling set of questions, focused not necessarily on "name and shame" issues but rather the obligations of the developer to design and release software with minimal flaws.

That particular question is at the heart of a fight most people don't hear about: Internet service providers (ISPs) versus software developers. One camp argues that ISPs should be actively monitoring their networks for malware, and taking affirmative action to cut down on the amount of "bad" traffic it sees flowing across its connections (and thus to its customers).

The other camp argues that the software developers need to take greater action because, well, they release a lot of bad, bug-ridden software. This camp, obviously backed by the ISPs, argues that it isn't fair to ask network providers to be constantly checking for malicious behavior, especially when so much of that bad behavior could be curtailed through better designed software.

It's a fascinating debate for sure. Since Internet traffic passes through their pipes, doesn't it make sense that ISPs should practice something akin to "See something, Say something"? On the other hand, why should ISPs be shouldered with the burden of inspecting traffic as it comes by, looking for potentially damaging packets of information. At the same time, what happens if they miss something or don't report it in a timely fashion—are they liable for not being such a "good Samaritan"?

It's a difficult debate, and that's before we even get to a key issue in all of this—ultimately software users have to actively install any fixes issued by the developers. It's not as if the developers can or should install those patches by themselves. Especially for larger and more complicated information technology systems, companies have to make sure that the patch will not interfere with other critical systems. Failure to make such checks could lead to problems with the overall system, and cause disruption of vital business processes.

What makes it all the more interesting is that it seems to fly in the face of many other product liability trends we are seeing.

We are all familiar with automobile recalls. If there is a problem, whether it be with the brakes, engine, airbags or other components that could lead to safety hazards, the National Highway Traffic Safety Administration (NHTSA) has the authority to issue recalls. As the NHTSA website notes:

"The National Traffic and Motor Vehicle Safety Act ... gives ... NHTSA the authority to issue vehicle safety standards and to require manufacturers to recall vehicles that have safety-related defects or do not meet Federal safety standards ... Manufacturers voluntarily initiate many of these recalls, while others are either influenced by NHTSA investigations or ordered by NHTSA via the courts. If a safety defect is discovered, the manufacturer must notify NHTSA, as well as vehicle or equipment owners, dealers, and distributors. The manufacturer is then required to remedy the problem at no charge to the owner. NHTSA is responsible for monitoring the manufacturer's corrective action to ensure successful completion of the recall campaign."

That indeed offers an interesting model. Perhaps we should consider a National Software Safety Administration, which has the authority to recall software and order patches installed to protect the nation writ large. There could be some benefits to that, including the obvious benefit of having a formalized process through which software flaws are identified and fixes pushed out to consumers.

Personally, while that is an interesting idea, I don't really know that it would be helpful. Frankly, I don't think we need government rooting around code—we already have enough problems with the fallout from the NSA/Snowden situation and the associated allegations of government-created software "backdoors." Moreover, it is a short leap between the government discovering and ordering a fix on a critical software flaw to merely directing remedial action for imperfect but not very harmful glitches.

Ultimately, and sadly, this will likely be resolved through the court system. Some creative plaintiffs lawyer will file suit against a software developer, saying but for their failure to create good software or their failure to rapidly fix a software flaw, the cyberattack would not have occurred. And of course because of that failure their client is entitled to $200 billion in damages. (Minus fees, of course.)

I am no fan of tort litigation by any measure, especially here. I think a blizzard of lawsuits against software companies would be wasteful and serve as a huge disincentive to innovation. Of course, developers can take heart knowing that software defects rarely if ever are considered actionable, but you never know how courts will react to a well-argued case.

Still, at some point the software industry has to ponder its role in the cyber-onslaught. I have no doubt that they are doing their best to churn out reliable and useful software programs, but at the same time perhaps more attention needs to be paid to finding and closing vulnerabilities or glitches before those same programs are brought to market. If nothing else, if a developer gains a reputation as a company that is seen as ripe for exploit by hackers, ultimately it will impact what matters most to them—sales.

Pillsbury Winthrop Shaw Pittman LLP