Software Fail Watch: The Most Interesting Bugs of Q1, 2018

by Chelsea Frischknecht MVB </>CORE · Apr. 13, 18 · Performance Zone · Analysis

We wrapped up 2017 with a collection of 606 recorded software bugs, impacting half of the world's population (3.7 billion people), $1.7 trillion in assets, and 314 companies. (Check out our final report in the Software Fail Watch: 5th Edition). We barely had time to process how staggering those numbers were before the clocks reset themselves and 2018's first bugs started making the headlines.

And one thing is for certain: Q1 hasn't disappointed. One single bug has already impacted an estimated *3 billion devices*, indicating that 2018 will make for yet another record-breaking Software Fail Watch.

Some of the most interesting software fails of Q1, however, have fallen into unexpected categories. As with any form of data collection and sorting, certain patterns expose themselves fairly quickly: the Consumer Tech industry experiences bugs with startling frequency, software security vulnerabilities and hacks are on the rise, etc.

That is why it always catches our attention to find a new bug that doesn't quite fit the standard mold. Here are a few of the unusual bugs in Q1—and why we find them so noteworthy.

Hawaii Sends Out a State-Wide False Alarm About a Missile Strike

On January 13, the citizens of Hawaii were notified to take immediate cover in the face of an inbound ballistic missile strike. It turned out to be a false alarm, although it took over 30 minutes (and, presumably, several thousand heart attacks) before the alert was retracted. Investigations found that while the problem was largely due to human error, there were "troubling" design flaws in the Hawaii Emergency Management Agency's alert origination software.

Why it's interesting: The Hawaii Emergency Management Agency was chastised for having no discernible differences between their testing and live alert environments, making it extremely easy to mistake which environment they were using in the moment. While software *bugs* (defined

easy to mistake which environment they were using in the moment. While software bugs (defined as a software failing to perform as designed) are the most common types of fails in the Software Fail Watch, it is dangerous to underestimate the damage poorly *designed* software can incur.

Uber's Self-Driving Car Causes a Fatal Accident

A tragic accident occurred on March 21, when a pedestrian was struck by one of Uber's self-driving cars. The victim ultimately died of her injuries in the hospital. Self-driving cars are famously marketed as the "safer" alternative to driving, utilizing cutting-edge technology and sensors to prevent such accidents. In this case, however, the software failed to detect the pedestrian, and the car's controller (who reportedly was not watching the road at the time) could not prevent the accident from happening.

Why it's interesting: Complex autonomous software like self-driving cars promise to effectually replace mankind's need to think or make decisions in certain situations. That begs the question then: if something *does* go wrong, who takes the responsibility? In this particular use case, it is unclear whether Uber or the car's controller is at fault for the accident. The more this type of software evolves and becomes available on the market, the more the legislation around these types of scenarios will need to evolve. Technology typically changes faster than law, however, which means we could be facing some interesting years ahead as government races to catch up with the new technology at our disposal.

"Spectre" Security Flaw Found in all Intel Processing Chips Made in Last Decade

Security researchers discovered a security flaw that exists in virtually every Intel processing chip made in the past 10 years. The vulnerability makes it possible for commonly used programs to partially discern the layout and content of information stored in protected memory kernels. Companies using the chips in their hardware scrambled to start patching the vulnerability—no small feat when there are an estimated 3 BILLION chips in production.

Why it's interesting: We're all familiar with the household brand names like Apple, Google Android, and BMW. What is often less familiar, however, are the names of the production giants that provide the parts that make up a recognizable product. For many of these companies (such as Intel), their hardware and software products are so widely used that a software fail has global repercussions which can take years to fix. The Takata Corporation, for instance, is an automotive parts company that gained notoriety in 2013, after a software defect was discovered that caused airbags to explode in a cloud of shrapnel. Five years later, 42 million cars have been recalled, over 100 injuries and 8 deaths have been reported, and brands like BMW, Chrysler, Ford, Honda, and Toyota are *still* trying to recall affected vehicles. The same risk occurs with open-source code, used to underpin hundreds-of-thousands (if not millions) of software and application instances.

This fact begs the question whether diversifying the source of our parts (hardware or software) would positively impact the safety of our software as a whole.

Like This Article? Read More From DZone

DZone Article
How to Avoid the Tricentis Software Fail Watch

DZone Article
The Most Shocking Software Failure Award Goes To…

DZone Article
How to Become the Sherlock Holmes of Bug Searching: Concentration on the Details

Free DZone Refcard
Visual Testing

Topics: PERFORMANCE, PERFORMANCE 2018, SOFTWARE BUGS, SOFTWARE FAIL WATCH

Let's be friends:

DZone.com is powered by