Nest Thermostat Software Bug

By Josh Ho, Justin Nelson, Lance Aguilar, Pierre Sarabamoun, Michael Pigeon

Article Breakdown

- **Google Nest** – makes smart home products
- Dec 2015: Pushed a buggy update to internet-connected thermostats, causing malfunction
    - “It didn’t show up for about 2 weeks”
- Thermostat owners left freezing in their homes 🥶
- Nest published a nine-point plan for users to fix the issue themselves
    - “Try turning it off and on again”
- New update has been pushed to address issue

Are Bugs In Production Avoidable

Are bugs like this in “finished” and distributed programs avoidable? If not, if bugs leaking through is considered too dangerous in certain contexts should software improvements in that area be avoided all together?

Bugs like this are almost always possible, but the chance of bugs can be reduced significantly by extensive testing over an extended period and by designing methods to detect and correct when something goes bad, such as when the temperature is very low.

Software improvements in dangerous areas shouldn’t be avoided since they still improve safety. However, there should be fallbacks for when the software fails.

Are Bugs In Production Avoidable (continued)

Many companies have very strict review and testing processes already, did NEST test less than the industry standard or did they just get unlucky that a bug slips through?

In the article, the NEST team said that it took 2 weeks for the bug to appear. Since there are a lot of possible major problems when a thermostat stops working right, they should have been testing their updates for an extended period of time in a lot of different environments. They probably should have been testing processes for a longer period of time before release.

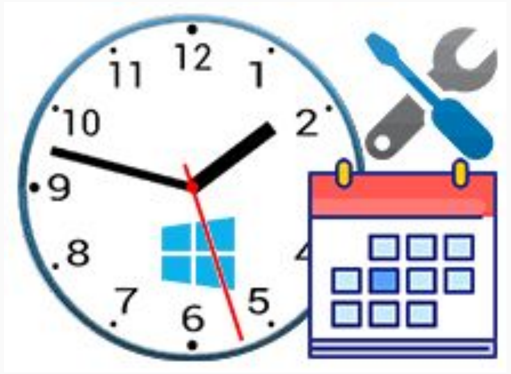
How to address this problem?

What should companies do to address this potential for critical bugs?

- The company mentioned the bug did not appear for 2 weeks, suggesting that the software may not have been tested thoroughly enough
- Better analysis of uses cases, as well as more specific software test cases could have been developed
- Software updates can be tested on a smaller scale (small user pool) before having the updates pushed to a larger consumer base
- Errors in a small sample size helps developers focus on problems quicker and more efficiently (errors in a larger sample size can be hard to keep track of)

Should they put in more rigid and strict testing protocols? What would that mean for the companies?

- Rigid and strict testing protocols should be in place
- The software error remained undetected because the software was not tested thoroughly enough
- A strict and specific testing protocol would help detect bugs that may not be readily apparent, or errors that can be undetected by normal tests
- Rigid testing protocols would require companies to spend more time in their testing phases, which could delay the release of updates
- Companies would need to assess for themselves which tests are worth update delays



Is it worth trying to address it?

Increasing the rigidity of testing protocol would require more man power and take more time, can companies afford all that time and money?

Assuming it is avoidable, these instances are very rare, so is it worth slowing down the production of software companies to try to bring that chance down as much as possible or is it not viable from a business point of view?

Do you think these bugs are avoidable? If not, is this software worth it at all?