Defeating the Scourge!

Of the Undertested Automatic Software Update

RBCS

TIME TESTED.
TESTING IMPROVED.

www.RBCS-US.com

Introduction

- You buy a car
- Six days later, in the night, someone from the dealership breaks into your garage
  - He removes the tires on the car
  - He installs "updated" tires that have holes in them
  - He leaves, taking the original tires with him
- In the morning, you have an undriveable car
- That's a crime, but it happens all the time—with software
- Let's look at some infamous examples, and then talk about how we can stop doing this

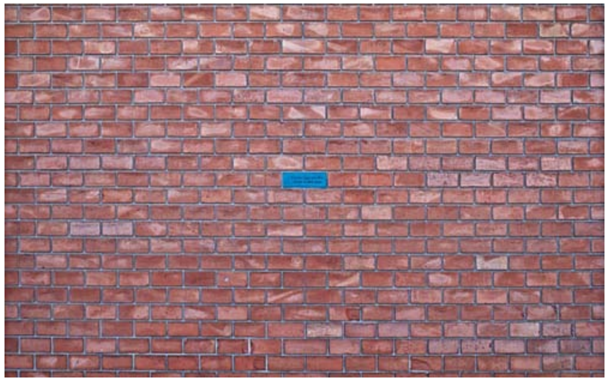
Some Background

- In late 1990s, I managed testing of a device with early automatic update features
- We ran thousands of updates to check for bricking and other bad behaviors
- We found and fixed a number of bugs that would have bricked the devices
- We respected the risks inherent in pushing updates to apps and OSes
- Recently, people have clearly become nonchalant about what can go wrong…

Full Speed Ahead, the Wrong Way

- Apple, in a fit of pique over Android, decided to drop Google Maps
- Sep 2012, iPhones automatically "upgraded" to Apple Maps when they installed iOS 6
- As Techradar wrote, "But very quickly iOS 6 upgraders began to notice problems: directions took them the wrong way, a phantom airport appeared near Dublin and Aukland's main train station appeared to be located in the middle of the sea."
- Tim Cook issued an apology, threw two people under the bus (said bus might have just been going the wrong way, though)
- I suspect Jobs would have just built the new Dublin airport and flooded downtown Auckland rather than admit the goof
- Dec 2012, Google's Google Maps free app became available for the iPhone, and was soon the most popular free app
- But hey, Apple Maps had that really cool 3D view thingee…

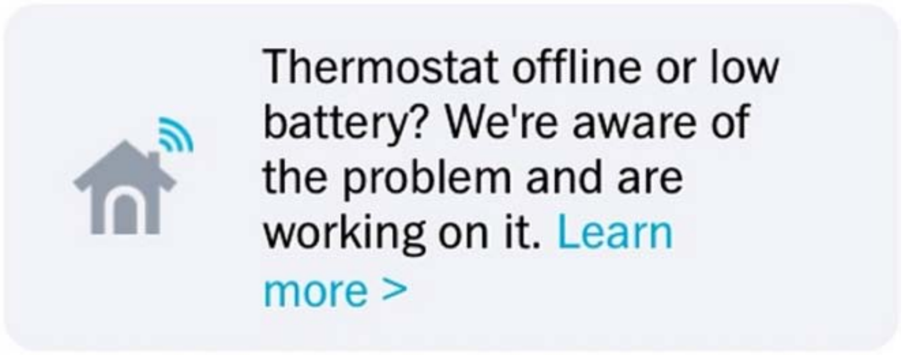
Another Brick in the Wall…



- Sept 2015, five months after initial release, Apple almost pushed bad software onto their Watch
- Bugs delayed the release by a week
- As Cnet wrote, "The delay of Watch OS 2 echoed the setback a year ago when Apple's iOS 8 mobile software…launched with numerous bugs [in] Wi-Fi, the Touch ID system and other functions. An update that quickly followed caused even more issues. Apple finally resolved the problems more than a week after iOS 8 launched."
- The iOS 8 release left about 10mm people unable to use their phones, which were effectively bricked during that period
- It might seem like I'm picking on Apple, but I'm not…

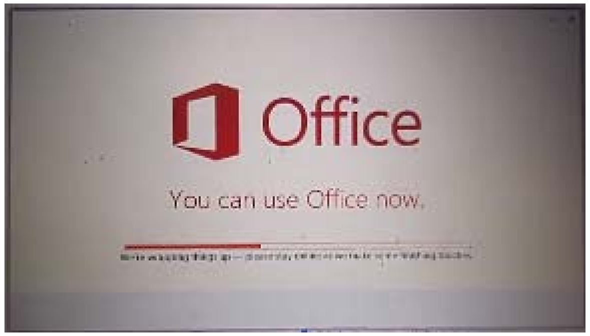
A Frozen Pipe in Your Wall?

- Dec 2015, Nest (a Google company) pushed an update for their IoT device, the Nest thermostat
- The update caused some units to drain their batteries and turn off
- Some houses got pretty cold in January
- The workaround was so confusing that Nest offered to pay for an electrician to come do it for confused customers
- While this sounds like a one-percenter problem, consider potential health impacts for infants or the elderly
- Suppose your house froze and a pipe broke?
- Tough luck, an arbitration clause prohibits you from suing for damages


Thermostat offline or low battery? We're aware of the problem and are working on it. Learn more >

How Much Is that Dog Crap on Windows?

- Nov 2015, Microsoft pushed a Windows 7 "security" update, which didn't give Outlook 2013 or me a waggly tail
- I spent over 3 hours fixing problem
- Microsoft knew for days yet said nothing
- Microsoft reports that 1bn people use Office, so assume half of them use Outlook
- Over half of people use Windows 7, so figure about 250mm people were affected
- If everyone suffered the same productivity hit I did, over 300,000 person-years lost
- Did Microsoft have to repay businesses and individuals who lost time this way?



Actually, no, I couldn't, not for two more days, at least not Outlook

40 min call, no fix, lousy sound, tech hung up on me

"Houston, We Have a Problem. Houston?"

- Even when updates are tested thoroughly, bad stuff can happen
- Feb 2013, an annual software update killed safety-critical, mission-critical, and comms software on the International Space Station
- According to Marcy Kerr, SW Dev Manager, two years of development and testing didn't catch the underlying data allocation bug in one line (out of 4.5mm) of code
- It took a couple hours to fix, a week to upload
- Lessons
  - Software is not linear—small changes can have huge effects
  - Even when best practices are followed, there is still some risk
  - This is not an excuse not to follow best practices, as NASA software bugs have never killed anyone, though they have caused mission failures
  - Ultimately, software engineering must evolve to true engineering

"Dude, I Pwned your Jeep!"

- The risks posed by auto-update channels go beyond functionality and reliability
- Auto manufacturers are starting to do over-the-air updates to cars
- Tesla already updates core systems, not just telematics and maps
- Jul 2015, Wired published an article where two hackers miles away high jacked a moving Jeep
- Similar weaknesses exist for implanted medical devices
- Jun 2013, Vice published an interview with Barnaby Jack, who said he could hack into pacemakers and insulin pumps and cause fatal events
- Jack died, aged 35, one week before he was to present his findings at the Black Hat conference for hackers
- It turns out he had ODed on cocaine, heroin, Xanax, and alcohol – unless you believe the conspiracy theories

Not Updating? Not an Option!

- So, if updating is so risky, how about we don’t do it?
- Two historic software failures resulted from not *enough* updating
- Therac-25 nuclear medicine device reused code from previous versions
- Previous Therac devices had hardware safety interlocks not present on the Therac-25
- A bug lead to software-initiated overdosing that the safety systems could have prevented, killing three people
- Ariane 5 re-used software from a previous rocket, but the data sizes for integers and decimal numbers were different
- An overflow failure during decimal-to-integer conversion caused the rocket to tip over, requiring its destruction

Ariane 501: $500mm bottle rocket

Ways to Mitigate the Risk

- In addition to careful adherence to known best practices related to quality and testing, some more ideas…
    - Thorough automated regression testing at all levels, including SIT tests through APIs and shared data
    - Wider use of A/B testing, beta testing, and other forms of staged releases
    - Testing skills for programmers and vice versa
    - Less-frequent, larger releases wherever possible
    - Enabling true end-user control over updates
    - Order-of-magnitude improvements in software security, so updates aren't urgently needed so often
- Put not your trust in lifecycles nor in languages, from which we have not gotten salvation…

Some (im)Modest Legislative Suggestions

- Public reporting of all regression bugs
- Banning of arbitration clauses in contracts of adhesion
- Implied warranty of fitness for software, with legal recourse for damages
- SOX-like requirement for executives to approve all software releases
- An SEI 2.0 initiative that goes beyond just looking at software development process

How Likely Is That Stuff?

- On the non-legislative side, employee and consumer pressure can drive those improvements
- This is especially true in hyper-competitive markets
- On the legislative side, tech companies have too much money for those to happen
- In the 1990s, we barely avoided the opposite situation, due to tech efforts to pass UCITA
- A tester named Cem Kaner helped lead the fight against UCITA
- Sadly, people may have to start dying (in larger numbers) from software bugs before sufficient public pressure will drive legislative changes

Stuff My Fellow Software Professionals Say

- In addition, let's take our role seriously
- Let's stop saying stupid stuff that encourages lackadaisical quality attitudes, such as…
  - "If you are not embarrassed by the first version of your product, you've launched too late"
  - Even if you have only "one hour to test" an application, you can be confident in your work
  - Given a set of TDD-based automated unit tests, programmers can change software without incurring regression risk
- These statements, and many others like them, push our profession in exactly the wrong direction

Conclusions

- Software updates carry significant risks to the quality of the updated systems
- The frequency of problems indicates inadequate care in managing these risks
- The very ability to do automated software updates also creates significant security risks
- Solutions are necessary, as not updating software is not an option
- Until software engineering becomes true engineering, the application of known best practices can help manage the risk
- If we as professionals don’t manage the risks, we’ll be regulated into managing them
- At the very least, let’s start talking like we really care about these risks

To Contact RBCS

For over twenty years, RBCS has delivered consulting, outsourcing, and training services to clients, helping them with software and hardware testing. Employing the industry's most experienced and recognized consultants, RBCS advises its clients, trains their employees, conducts product testing, builds and improves testing groups, and hires testing staff for hundreds of clients worldwide. Ranging from Fortune 20 companies to start-ups, RBCS clients save time and money through improved product development, decreased tech support calls, improved corporate reputation and more. To learn more about RBCS, visit www.rbcs-us.com.

Address:	RBCS, Inc. 31520 Beck Road Bulverde, TX 78163-3911 USA
Phone:	+1 (830) 438-4830
E-mail:	info@rbcs-us.com
Web:	www.rbcs-us.com
Twitter:	@RBCS, @LaikaTestDog
Facebook:	RBCS-Inc.