PRIVACY AND SECURITY

Making Security Sustainable

By Ross Anderson

Communications of the ACM, March 2018, Vol. 61 No. 3, Pages 24-26
10.1145/3180485

Comments

As we start to connect durable goods such as cars, medical devices, and electricity meters to the Internet, there will be at least three big changes. First, security will be more about safety than privacy. Certification will no longer mean testing a car once before selling it for 10 years; safety will mean monthly software updates, and security will be an integral part of it. Second, we will have to reorganize government functions such as safety regulators, standards bodies, testing labs, and law enforcement. Finally, while you might get security upgrades for your phone for two or three years, cars will need safety and security patches for 20 years or more. We have no idea how to patch 20-year-old software; so we will need fresh thinking about compilers, verification, testing, and much else.

ARTICLE CONTENTS:

Credit: DJ Taylor

Back to Top

Privacy, Availability, or Safety?

The early security scares about the "Internet of Things" have mostly been about privacy. There have been reports of the CIA and GCHQ turning smart TVs into room bugs, while the German government banned the Cayla doll whose voice-recognition system could be abused in the same way.[3] Yet privacy may change less than we think. Your car knows your location history, sure, but your phone knows that already. It also knows where you walk, and it is already full of adware.

Denial of service has also been in the news. In October 2016, the Mirai botnet used 200,000 CCTV cameras (many of them in Brazil and Vietnam) to knock out Twitter in the Eastern U.S. for several hours. ISPs know they may have to deal with large floods of traffic from senders with whom they cannot negotiate, and are starting to get worried about the cost.

But the most important issue in the future is likely to be safety. Phones and laptops do not kill a lot of people, at least directly; cars and medical devices do.

In 2016, Éireann Leverett, Richard Clayton, and I conducted a research project for the European Commission on what happens when devices that are subject to safety regulation start to contain both computers and communications.[5] There are surprisingly many regulated industries; it is not just the obvious ones like aircraft and railway signals, but even kids' toys—they must not have lead paint, and if you pull a teddy bear's arms off, they must not leave sharp spikes.

So what is the strategic approach? We looked at three verticals—road vehicles, medical devices, and smart meters. Cars are a good example to illustrate what we learned—though the lessons apply elsewhere too.

Back to Top

Security and Safety for Cars

Car safety has been regulated for years. In the U.S., the National Highways Transportation and Safety Administration was established in the 1960s following Ralph Nader's campaigning; Europe has an even more complex regulatory ecosystem. Regulators discovered by the 1970s that simply doing crash tests and publishing safety data were not enough to change industry behavior. They had to set standards for type approval, mandate recalls when needed, and coordinate car safety with road design and driver training.

Insurers do some of the regulatory work, as do industry bodies; but governments provide the foundation.

This ecosystem faces a big change of tempo. At present, a carmaker builds a few hundred prototypes, sends some for crash testing, has the software inspected, and gets certification under more than 100 regulations. Then it ramps up production and sells millions of cars. Occasionally carmakers have to change things quickly. When a Swedish journalist found that avoiding an elk could cause an A-class car model to roll over, Mercedes had to redesign its suspension and fit a stability control system, which delayed the product launch at a cost of $200 million.[2] But most of the safety case is a large up-front capital cost, while the time constant for the design, approval, and testing cycle is five years or so.

In the future, a vulnerability in a car will not need a skillful automotive journalist to exploit it. Malware can do that. So if a car can be crashed by commands issued remotely over the Internet, it will have to be fixed. Although we have known for years that car software could be hacked, the first widely publicized public demonstration that a Jeep Cherokee could actually be run off the road—by Charlie Miller and Chris Valasek, in 2015—showed that the public will not tolerate the risk.[4] While previous academic papers on car hacking had been greeted with a shrug, press photos of the Cherokee in a ditch forced Chrysler to recall 1.4 million vehicles.

Cars, like phones and laptops, will get monthly software upgrades. Tesla has started over-the-air upgrades, and other vendors are following suit. This will move safety regulation from pre-market testing to a safety case maintained in real time—a challenge for both makers and regulators. We will need better, faster, and more transparent reporting of safety and security incidents. We will need responsible disclosure—so people who report problems are not afraid of lawsuits. We will need to shake up existing regulators, test labs, and standards bodies. Over two dozen European agencies have a role in car safety, and none of them have any cybersecurity expertise yet. We will need to figure out where the security engineers are going to sit.

We may need to revisit the argument between intelligence agencies who want "exceptional access" to systems for surveillance, and security experts who warn that this is hazardous.[1] The Director of the FBI and the U.K. Home Secretary both argue that they should be able to defeat encryption; they want a golden master key to your phone so they can listen in. But how many would agree to a golden master key that would let government agents crash their car?

Over two dozen European agencies have a role in car safety, and none of them have any cybersecurity expertise yet.

There are opportunities too. Your monthly upgrade to your car software will not just fix the latest format string vulnerability, but safety flaws as well. The move to self-driving cars will lead to rapid innovation with real safety consequences. At present, product recalls cost billions, and manufacturers fight hard to avoid them; in the future, software patches will provide a much cheaper recall mechanism, so we can remove the causes of many accidents with software, just as we now fix dangerous road junctions physically.

But cars will still be more difficult to upgrade than phones. A modern car has dozens of processors, in everything from engine control and navigation through the entertainment system to the seats, side mirrors, and tire-pressure sensors. The manufacturer will have to coordinate and drive the process of updating subsystems and liaising with all the different suppliers. Its "lab car"—the rig that lets test engineers make sure everything works together—is already complex and expensive, and the process is about to get more complex still.

Back to Top

Sustainable Safety and Security

Perhaps the biggest challenge will be durability. At present most vendors won't even patch a three-year-old phone. Yet the average age of a U.K. car at scrappage is 14.8 years, and rising all the time; cars used to last 100,000 miles in the 1980s but now keep going for nearer 200,000. As the embedded carbon cost of a car is about equal to that of the fuel it will burn over its lifetime, a significant reduction in vehicle durability will be unacceptable on environmental grounds.

As we build more complex artifacts, which last longer and are more safety critical, the long-term maintenance cost may become the limiting factor. Two things follow. First, software sustainability will be a big research challenge for computer scientists. Second, it will also be a major business opportunity for firms who can cut the cost.

On the technical side, at present it is hard to patch even five-year-old software. The toolchain usually will not compile on a modern platform, leaving options such as keeping the original development environment of computers and test rigs, but not connecting it to the Internet. Could we develop on virtual platforms that would support multiple versions?

Perhaps the biggest challenge will be durability.

That can be more difficult than it initially appears. Toolchain upgrades already break perfectly functional software. A bugbear of security developers is that new compilers may realize that the instructions you inserted to make cryptographic algorithms execute in constant time, or to zeroise cryptographic keys, do not affect the output. So they optimize them away, leaving your code suddenly open to side-channel attacks. (In separate work, Laurent Simon, David Chisnall, and I have worked on compiler annotations that enable a security developer's intent to be made explicit.)

Carmakers currently think their liability for upgrades ends five years after the last car is sold. But their legal obligation to provide spare parts lasts for 10 years in Europe; and most of the cars in Africa arrive in the country secondhand, and are repaired for as long as possible to keep them operable. Once security patches become necessary for safety, who is going to be writing the patches for today's cars in Africa in 25 years' time?

This brings us to the business side—to the question of who will pay for it all. Markets will provide part of the answer; insurance premiums are now rising because low-speed impacts now damage cameras, lidars, and ultrasonic sensors, so that a damaged side mirror can cost $1,000 rather than $100. The firms that earn money from these components have an incentive to help maintain the software that uses them. And part of the answer will be legal; there have been regulations in Europe since 2010 that force carmakers to provide technical information to independent garages and spare-parts manufacturers. It is tempting to hope that a free/open source approach might do some of the heavy lifting, but many critical components are proprietary, and need specialist test equipment for software development. We also need incentives for minimalism rather than toolchain bloat. We do not really know how to allocate long-term ownership costs between the different stakeholders so as to get the socially optimal outcome, and we can expect some serious policy arguments. But whoever pays for it, dangerous bugs have to be fixed.

Once software becomes pervasive in devices that surround us, that are online, and that can kill us, the software industry will have to come of age. As security becomes ever more about safety rather than just privacy, we will have sharper policy debates about surveillance, competition, and consumer protection. The notion that software engineers are not responsible for things that go wrong will be put to rest for good, and we will have to work out how to develop and maintain code that will go on working dependably for decades in environments that change and evolve.

Back to Top

References

1. Abelson, H. et al. Keys under doormats: Mandating insecurity by requiring government access to all data and communications. *Commun. ACM 58*, 10 (Oct. 2015).

2. Andrews, E.L. Mercedes-Benz tries to put a persistent Moore problem to rest. *New York Times* (Dec. 11,

1997).

3. German parents told to destroy Cayla dolls over hacking fears. *BBC News* (Feb. 17, 2017).

4. Greenberg, A. Hackers remotely kill a jeep on the highway—with me in it. *Wired* (July 21, 2015).

5. Leverett, É., Clayton, R., and Anderson, R. Standardisation and certification of the Internet of Things. In *Proceedings of WEIS 2017*; http://weis2017.econinfosec.org/program/.

Back to Top

Author

Ross Anderson (Ross.Anderson@cl.cam.ac.uk) is Professor of Security Engineering at Cambridge University, U.K. He is a Fellow of the Royal Society and the Royal Academy of Engineering, and author of *Security Engineering—A Guide to Building Dependable Distributed Systems*.

No entries found