Security and Privacy Viewpoint

Making Security Sustainable

Can there be an Internet of durable goods?

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 ten years; safety will mean monthly software updates, and security will be an integral part of it. Second, we will have to reorganise 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 twenty years or more. We have no idea how to patch 20-year-old software; so we'll need fresh thinking about compilers, verification, testing and much else.

Privacy, availability or safety?

The early security scares about the "Internet of Things" have mostly been about privacy. We've heard 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[i]. 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's already full of adware.

Denial of service has also been in the news. Last October, the Mirai botnet used 200,000 CCTV cameras (many of them in Brazil and Vietnam) to knock out Twitter in the Eastern USA for several hours. ISPs know they may have to deal with large floods of traffic from senders with whom they can't 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 don't kill a lot of people, at least directly; cars and medical devices do.

Last year Éireann Leverett, Richard Clayton and I did 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[ii]. There are surprisingly many regulated industries; it's not just the obvious ones like aircraft and railway signals, but even kids' toys – they mustn't have lead paint, and if you pull a teddy bear's arms off, they mustn't leave sharp spikes.

So what's 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.

Security and safety for cars

Car safety has been regulated for years. In the USA, the National Highways Transportation and Safety Administration was set up 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 was not enough to change industry behaviour. 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 roll a new A-class, Mercedes had to redesign its suspension and electronic stabilization, and delay the product launch at a cost of $200m[iii]. But most of the safety case is a large upfront 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 won't need a skilful motoring 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-publicised 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[iv]. While previous academic papers on car hacking had had limited impact, press photos of the Cherokee in a ditch forced Chrysler to recall 1.4m 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 aren't 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[v]. The Director of the FBI and the UK 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?

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 future, software patches will be give 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 harder 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, wing mirrors and tyre 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.

Sustainable safety and security

Perhaps the biggest challenge will be durability. At present we have a hard time patching a phone that’s three years old. Yet the average age of a UK car at scrappage is about 14 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, durability is good. Any reduction has grave consequences for the environment.

As we build more complex artefacts, 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’s hard work to patch even five-year-old software. The toolchain usually won’t 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?

That can be harder than it looks. Toolchain upgrades already break perfectly functional software. A bugbear of security developers is that new compilers may realise that the instructions you inserted to make cryptographic algorithms execute in constant time, or to zeroise cryptographic keys, don’t affect the output. So they optimise 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.)

At present, carmakers think their liability for upgrades ends five years after the last car is sold. But most of the cars in Africa arrive in the country second-hand, and are repaired for as long as possible. Once security patches become necessary for safety, who’s going to be writing the patches for today’s cars in Africa in 25 years’ time?

This brings us to the business side: to 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 clipped wing mirror can cost $1000 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; we have had regulations in Europe since 2010 that force carmakers to provide technical information to independent garages and spare parts makers. It is tempting to hope that a free / open source approach might do some of the heavy lifting, but we also need incentives for minimalism rather than toolchain bloat. We don't 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 tussles. But whoever pays for it, the maintenance will have to be done.

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 laid 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.

– – – –

Ross Anderson is Professor of Security Engineering at Cambridge University. 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'

[i] "German parents told to destroy Cayla dolls over hacking fears", BBC News, February 17 2017
[ii] Éireann Leverett, Richard Clayton, Ross Anderson, "Standardisation and Certification of the Internet of Things", WEIS 2017 (in submission)
[iii] Edmund L Andrews, "Mercedes-Benz tries to Put a Persistent Moore Problem to Rest", New York Times, December 11 1997
[iv] Andy Greenberg, "Hackers remotely kill a jeep on the highway – with me in it," Wired, July 21 2015
[v] Hal Abelson, Ross Anderson, Steve Bellovin, Josh Benaloh, Matt Blaze, Whit Diffie, John Gilmore, Matt Green, Susan Landau, Peter Neumann, Ron Rivest, Jeff Schiller, Bruce Schneier, Michael Specter and Danny Weitzner, "Keys Under Doormats: Mandating insecurity by requiring government access to all data and communications" in Communications of the ACM v 58 no 10 (Oct 2015)