DEPARTMENT OF COMPUTER SCIENCE
SOFTWARE AND SYSTEMS SECURITY RESEARCH GROUP

UNIVERSITY OF
OXFORD

Keeping the bad software at bay?

or, I don't want anti-virus on my car

Andrew Martin

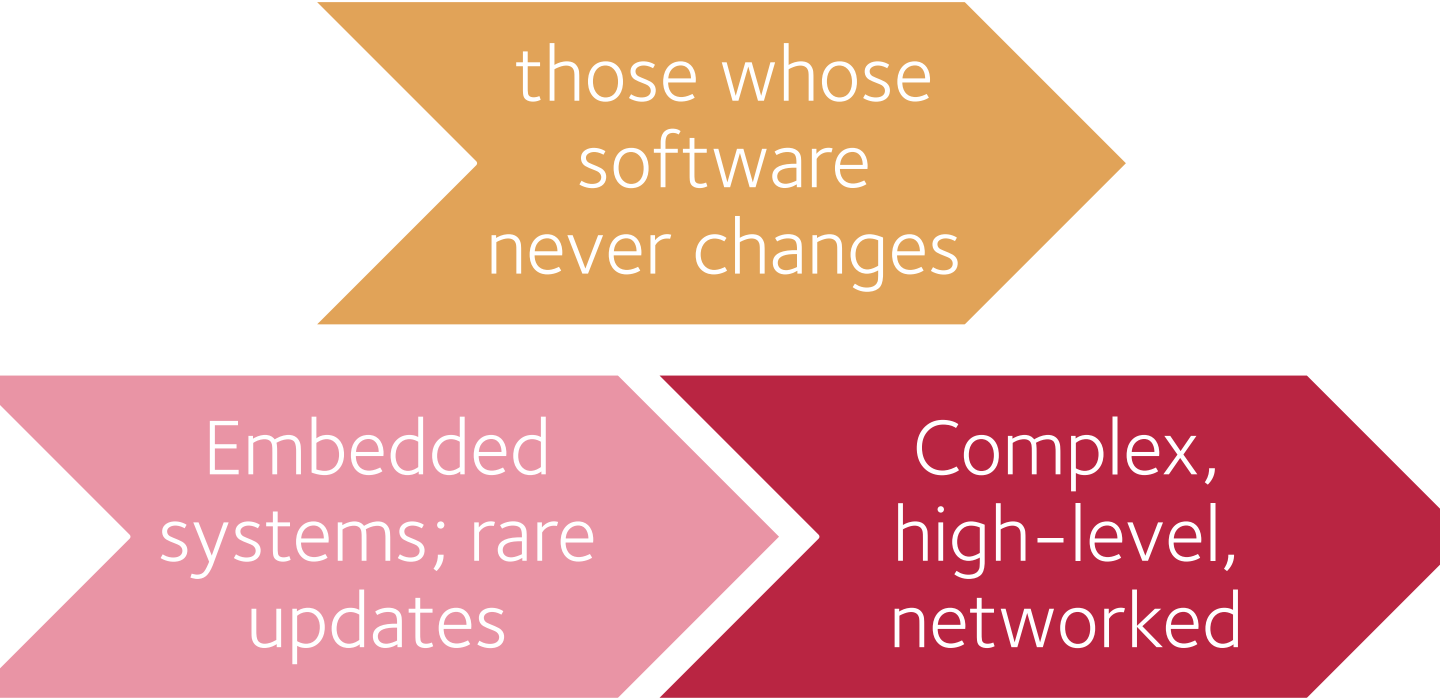
February 2016

two kinds of microprocessor-based system

those whose software never changes

those susceptible to malware

two kinds of microprocessor-based system

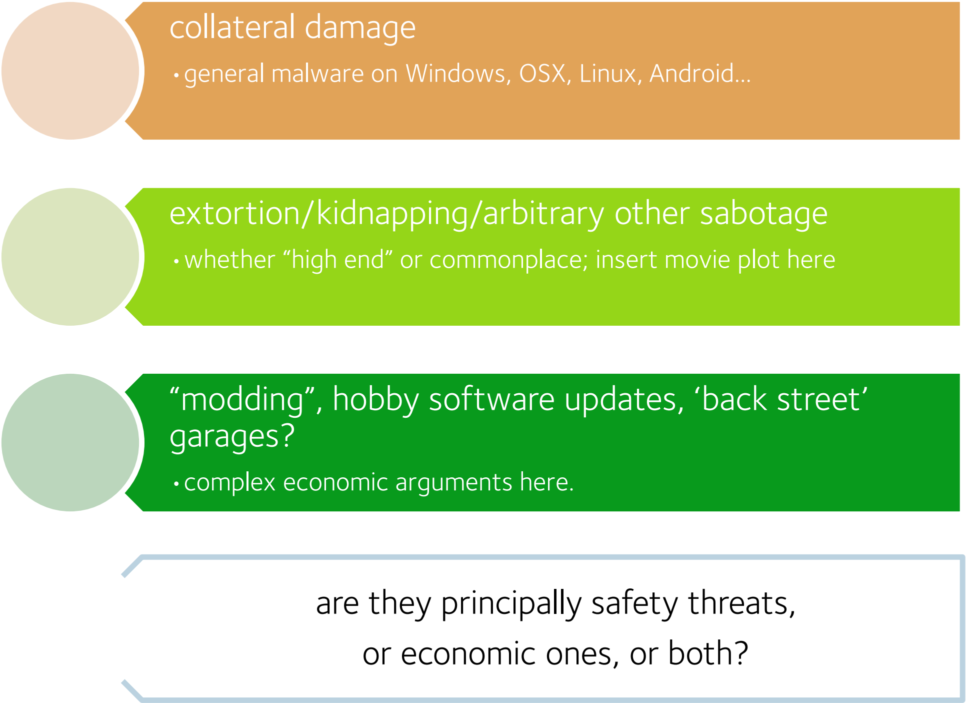


those whose software never changes

Embedded systems; rare updates

Complex, high-level, networked

malware threat classes for autonomous vehicles (?)



collateral damage

- general malware on Windows, OSX, Linux, Android…

extortion/kidnapping/arbitrary other sabotage

- whether “high end” or commonplace; insert movie plot here

“modding”, hobby software updates, ‘back street’ garages?

- complex economic arguments here.

are they principally safety threats,
or economic ones, or both?

how (not) to address the problem

- obscurantist “no one will do this”
- fatalistic *“it’s going to get hacked anyway”*
- idealistic “we can make malware-proof systems”
- anti-virus

- access control technologies
- firewalls etc.
- sandboxing
- address space randomization
- (etc.) …

- trusted computing
- secure boot
- trusted execution environments (TEEs)
- trusted enclaves; s/w partitioning
- hardware-backed key storage
- launch control
- limited/trusted software sources

secure boot, trusted execution, protected storage

Lessons from the PC platform:

- trusted boot/secure boot makes a material difference
  - Windows 8/10 step-change in control
  - ChromeOS
- trusted hardware can improve usability
  - e.g. bitlocker
- frequent software updates are the biggest protection right now
- good technology can suffer from slow adoption for diverse reasons
  - *one* is that people dislike giving up flexibility/control
  - another is FUD
  - poor ancillary design decisions play a part, also

secure boot, trusted execution

Phone OSs: a different story

- redesign of OS *and ecosystem* gives a massive security boost
- in practice, very little malware
  - not for want of trying
- rooting/jailbreaking a bit too easy
- secure enclaves/TEEs becoming common
  - making the most of this remains a huge research challenge
- update cycle is badly broken
  - particularly in Android

prospects

TEEs taking hold; capabilities commoditized

to be effective, their use need to be designed in from the bottom – radically different approach to s/w structuring.

it would be really unwise to defer this stuff till “version 2”

a lot of lessons to learn from PC and mobile platforms

the compartmentalization which comes from using TEEs can help to bound the parameters of the safety/security/freedom conundrum

- but are unlikely to resolve the argument entirely

bad outcomes would include an active community ‘rooting’ vehicles

Supply chain issues have a big impact here: but there is significant cost