OCT 2017

Towards **Trustable Software**

A SYSTEMATIC APPROACH TO ESTABLISHING TRUST IN SOFTWARE

INTRODUCTION TO THE WHITE PAPER

Rt Hon. Lord Reid of Cardowan
Executive Chairman, ISRS

"We very much hope that this paper will serve to stimulate discussion of the first principles and steps towards consensus as to how software should be designed, constructed and operated so as to be trustable."

While software has become critical to virtually all aspects of modern life, processes for determining whether we can trust it are conspicuously absent.

The goal of this paper is to stimulate discussion of the urgent need, potential solutions and proposed next steps to address the systemic risks posed by that gap.

In **Towards Trustable Software**, the Institute for Strategy, Resilience and Security (ISRS) at UCL, in association with software development specialist Codethink, explores the potential for a more secure foundation for societal resilience, analogous to existing trust mechanisms in key industries such as finance, healthcare and construction.

Among stakeholder groups – vendors, purchasers, software engineers, computer scientists, government and regulators – there exists little, if any, consensus as to how software should be designed, constructed and operated to achieve this.

We examine current approaches and deficiencies within the software industry towards the issue of trust and propose the concept of a *trustable software engineering process* as a necessary and appropriate underpinning platform to ensure solid foundations for the trust of software going forward.

The principles of how that process might work are outlined, by establishing software engineering practices that generate audit information at all stages of creation, deployment, change and use, to enable the continual assessment of trust, just as this is done in other industries.

Download your copy of this free white paper at:
www.trustablesoftware.com

About Codethink

Codethink is a leading provider of effective software engineering solutions, particularly in the infrastructure critical space. The company develops and maintains system and device-level software supporting advanced technical applications for its international corporate clients, across a range of industries including aerospace, automotive, finance, medical and telecoms. Codethink has pioneered software industry thinking around the concepts of trustable software, with a view to improving the quality of software engineering for societal good.

About the Institute for Strategy, Resilience & Security (ISRS) at UCL

Over the last decade the Institute for Strategy Resilience & Security (ISRS) at UCL has served as a pioneer and forum for next generation thinking. Founded by the Rt Hon. Lord Reid of Cardowan, ISRS provides analysis and assessment of the major issues of resilience with respect to national and global infrastructure and the ability of governments, regulators and businesses to respond to them. The Institute advises industry and the public sector on the persistent challenges to their agility, stamina and capacity in strategic decision making, so as to better face existential threats, risks, and disruptive innovation that are not addressed by conventional strategy and forecasting.

For industry and public sector queries in relation to this paper, please email: info@isrs.org.uk

For media enquiries about this paper, please email: press@isrs.org.uk

Institute for Strategy Resilience & Security
University College London

in association with

Codethink

EXECUTIVE SUMMARY

Towards Trustable Software

Within the space of a few decades, human civilisation has become highly dependent upon computer software, a trend that shows no signs of reversing, as our industries, homes, transportation and a plethora of mechanical and electronic objects – the "Internet of Things" – become increasingly software controlled and interconnected.

However, while improved regulation of construction, healthcare, law, financial services and other fundamental building blocks of human society has achieved ever greater levels of trust, the reverse is true of software. Engineering practices within the software industry, which would be considered irresponsible in construction or mechanical engineering, have led to the creation of systems that are not simply unworthy of trust, but incapable of having their level of trustworthiness assessed. As software has evolved from basic, highly deterministic electronic circuits and simple control logic to levels of unfathomable complexity contained within billions of lines of code, this blind spot has crept up upon us and it should be of major concern to governments, regulators, the software industry and the general public. When these systems fail in unexpected ways, as they inevitably must, they risk not only far reaching and potentially systemic consequences, but also triggering future crises of confidence in the products and services that they support.

Existing initiatives to improve the *trustworthiness* of software have focused largely on how to build better software by improving safety, reliability, availablility, resilience and security. Complementary to these, the concept of trustable software proposes a general solution that adds auditability to the software development process, enabling parties in the value chain to assess the degree to which they can trust a particular piece of computer code, in the same way that audit trails provide confidence in other industries. Auditability does not *de facto* enforce trust, rather it strongly incentivises behaviours along the value chain that lead to it through increased transparency of the process.

All critical products and services upon which human health, safety and security depend, have, of necessity, evolved recognisable processes to provide transparency and allow assessment of the degree to which to which that product or service is capable of being trusted. We refer to these as trustable processes because they generate the ability to trust. These vary from industry to industry, but generally take the form of laws, regulations, standards and audit practices. They provide confidence that e.g. a pill may be swallowed, that is safe to board an aircraft - conversely, that the risk of using a product or service is worth accepting.

However, there exists an important exception – software. In an age of increasing reliance upon software and ever more complex, interconnected and interdependent systems, we must address the question: to what extent can we trust this software?

Unlike physical construction, software does not have to conform to a set of building standards; unlike the pharmaceutical industry, there are no notified bodies or regulators; unlike the legal profession there is not a single body upholding standards of practice, and unlike accounting, software is unaudited. There is currently no recognisable process, regulatory framework, set of standards or audit trail by which, at any stage, it is possible to assess the degree to which software is capable of being trusted. Instead, software use remains largely an act of faith, built upon a stack of unverified assumptions, as most computer code is written informally and evaluated based on whether or not it works. Little software is formally verified to be error free and it is generally supplied in an opaque manner to its users. Even open source software, while in principle visible in its entirety, is in practice often so large and complex that fully understanding its operation is unfeasible.

We are on the cusp of further dramatic increases in the capabilities and complexity of software, and the issues of trust that are raised by its use are set to increase exponentially with the advent of evolving technologies such as robotics and artificial intelligence. Virtually every aspect of human life will involve or be controlled entirely by devices incorporating software. Software is now used to deliver many essential public services and to support critical national infrastructure. The loss or denial of service for any reason, accidental or deliberate, has potential consequences that range from mere inconvenience and reputational damage, to financial loss and ultimately loss of life.

The advent of the driverless car will finalise an ongoing engineering paradigm inversion, whereby a vehicle that is currently considered primarily as a mechanical object supported by software will become viewed as primarily software encapsulated within mechanical components. Concerns of safety and security will shift from trust in mechanical components to trust in the software, for example, can a cyber attacker take control of the vehicle?

A Critical Issue to Address Now

In the wake of the global financial crisis of 2007-2008, it became clear that the crisis was avoidable and was caused by widespread failures in regulation and supervision, poor management of accumulated systemic risk, lack of transparency, breakdown in accountability and ethics and failures to correctly price risk.

Analogously, despite an urgent unmet need, the software industry is inexorably drawn towards fuelling growth and will de facto ignore and resist a "push" towards a systematic approach to trust in software. An equivalent "pull" is required by governments and regulators in recognising the problem and encouraging the adoption of trustablility as standard practice, before a series of events or a particular disaster forces this issue into the wider public domain and Government is required to compel industry post hoc to address the issue of trust in software.

By 2020, at least 20 billion devices will be connected to the Internet, each more complex, interconnected and interdependent than ever before. Ignoring the systemic risks, lack of transparency, breakdown in accountability and failure of regulatory supervision holds the potential to accumulate a crisis as potent as any previously experienced.

Operating in an environment with software supplied 'as safe as possible', as it currently is, but without an auditable process for verifying the provenance and testing of that code, is no longer appropriate. Without adopting a process by which the trustability of software can be determined, society will increasingly stumble from one problem to the next. Whether this is experienced as failure in use, increased cyberattacks, or financial loss, the result will inevitably lead to an erosion of public confidence with repercussions for governments and regulators. Despite the challenges of adopting this approach in an industry that has historically been relatively free of constraints, the opportunity for nation states that are early adopters is competitive advantage through the creation of a safer society and a safer place to do business.

Trustability: An Established Key to Trust

A trustable process can be defined as "auditable in such a way that, at any point in the process, one can assess the degree to which it can be trusted". Although this term may be unfamiliar in everyday language, examples in use are immediately recognisable e.g. financial auditing is an established process that evolved over centuries in response to the need for trust in finance. The handling of evidence in the criminal justice system also follows a strict process so that a jury can have confidence in the provenance of evidence and that it has not been tampered with.

The requirements and steps of these trustable processes may at first appear to have little in common. However, all such processes share a set of features that enable trustability: those providing a product, service or information are required to present detailed evidence on the provenance, manufacture, testing and validity of what is being supplied. The evidence required, its format and the standards for preparing that evidence are specified by a regulator or agency, and it is then made available to a nominated body to inspect and audit to certify its accuracy.

To find out more please visit:
www.trustablesoftware.com

Towards **Trustable Software** ■