

## Inside Risks

# How Might We Increase System Trustworthiness?

*Summarizing some of the changes that seem increasingly necessary to address known system and network deficiencies and anticipate currently unknown vulnerabilities.*

**T**HE ACM RISKS Forum (risks.org) is now in its 35<sup>th</sup> year, the *Communications Inside Risks* series is in its 30<sup>th</sup> year, and the book they spawned—*Computer-Related Risks*<sup>7</sup>—went to press 25 years ago. Unfortunately, the types of problems discussed in these sources are still recurring in one form or another today, in many different application areas, with new ones continually cropping up.

This seems to be an appropriate time to revisit some of the relevant underlying history, and to reflect on how we might reduce the risks for everyone involved, in part by significantly increasing the trustworthiness of our systems and networks, and also by having a better understanding of the causes of the problems. In this context, ‘trustworthy’ means having some reasonably well thought-out assurance that something is worthy of being trusted to satisfy certain well-specified system requirements (such as human safety, security, reliability, robustness and resilience, ease of use and ease of system administration, and predictable behavior in the face of adversities—such as high-probability real-time performance).

The most recent *Inside Risks* discussion of trustworthiness appeared in the November 2018 *Communications* column.<sup>2</sup> This column takes a different view of the problems, with a specific conclusion that we need some funda-

mental changes in the state of the art and practice of developing and using computer systems, rather than trying to continually make small incremental improvements on baselines that may be unworthy.

The pithy wisdom of Albert Einstein—“Everything should be made as simple as possible—*but not simpler*”—is particularly relevant in the design, modification, and configuration of computer systems and networks. Oversimplification is often a cause of failure to satisfy expectations. Indeed, this

column seriously violates that wisdom: each bullet item significantly oversimplifies the point it is intended to make. Thus, each item should be considered as a guideline that must be applied with considerable care, experience, and detailed elaboration. Consequently, given that achieving trustworthiness is inherently complex and there are typically no easy answers or quick fixes, it is with some trepidation that I offer the following ideas that might help enhance trustworthiness:

► Accept that we cannot build ade-

quately trustworthy applications on top of compromisable hardware and flawed systems of today, particularly for those with life-critical requirements. (The Common Vulnerabilities Enumerators list—[cve.mitre.org](http://cve.mitre.org)—now includes over 120,000 vulnerabilities!) For example, the best cryptography and useful artificial intelligence can be completely subverted by low-level attacks, insider misuse, and hardware failures, whereas applications are still a huge source of vulnerabilities even in the presence of stronger operating-system security. Vulnerabilities tend to be pervasive. On the other hand, building new systems or cryptography from scratch is likely to be risky. Thus, having a set of trustworthy basic system and cryptographic (for example, EverCrypt) components would be a highly desirable starting point.

- ▶ Establish a corpus of theoretical and practical approaches for predictable composition of such components—addressing both composability (requiring the preservation of local properties) and compositionality (requiring the analysis of emergent properties of compositions, some of which are vital, as in safety and security—but some of which may be dangerous or otherwise failure-prone, as in exposed crypto keys and privacy violations). Composition itself can often introduce new vulnerabilities.

- ▶ Develop and systematically use more ways to reliably increase trustworthiness through composition. Desirable approaches might include (for example) the use of error-correcting codes, cryptography, redundancy, cross-checks, architectural minimization of what has to be trusted, strict encapsulation, and hierarchical layering that avoids adverse dependencies on less-trustworthy components.

- ▶ Whenever a technology or a component might be potentially unsound, trustworthiness (for both composition and compositionality) must be independently evaluated—for example, when using machine learning in life-critical applications.

- ▶ Adopt and honor underlying principles of computer systems, especially with respect to total-system trustworthiness for safety and security.

- ▶ Eschew the idea of inserting back doors (or not patching existing ones) in computer and communication sys-

## We need some fundamental changes in the state of the art and practice of developing and using computer systems.

tems.<sup>1</sup> It should be intuitively obvious that if back doors in systems have exploitable vulnerabilities, they would be exploited by people and programs supposedly not authorized to use them. Nevertheless, governments repeatedly fantasize that there can be bypasses that would be securely accessible only to ‘authorized’ entities. (Consider again the first bullet item in this column.)

- ▶ Recognize that trustworthiness in the “Internet of Things” may always be suspect (for example, regarding security, integrity, human safety, and privacy). Various hardware-software and operational approaches must be developed (perhaps easily securable and locally maintainable firewalls?) that can help control and monitor how various classes of devices can more soundly be connected to the Internet. Self-driving vehicles, fully autonomous highways, and totally interconnected smart cities imply the components and the total systems must be significantly more trustworthy. However, the risks of ubiquitously putting your crown jewels on the IoT would seem to be excessively unwise.

- ▶ Accept the fact that the use of remote processors and storage (for example, cloud computing) will not necessarily make your computer systems more trustworthy, although there are clearly considerable cost and operational savings that can result from not having to manage local hardware and software. Nevertheless, trusting trustworthy third-party cloud providers would be more desirable than attempting to create one’s own. As in other cases, there are many trade-offs to be considered.

- ▶ Accept the reality that we cannot build operational election systems that are trustworthy enough to withstand hacking of registration databases, in-

sider misuse, rampant disinformation and disruptions—especially using components without substantive audit trails or paper records that should be able to make forensics-worthy analysis possible. Although greater system trustworthiness would be helpful, many of the existing problems are not technological and must also be addressed.

- ▶ Recast software engineering and system engineering as engineering disciplines, with more focus on hardware and software vulnerabilities, aspects of system trustworthiness, importance of well-defined system requirements, proactive design, system usability, risk assessment, computer-science theory and practice.

- ▶ Revamp software-engineering educational programs to ensure graduates have the necessary abilities and resources.<sup>3</sup>

- ▶ Recognize there are no one-size-fits-all solutions, and that many potential trade-offs must be considered. Furthermore, technology by itself is not enough, and many other factors must be considered—especially critical systems.

- ▶ Stress learning, not just teaching, to instill an awareness of the issues discussed here from elementary school on, dealing with complexity, principles, abstraction, respecting holistic long-term thinking rather than just short-term premature optimization, logical reasoning, altruism, and much more. Encourage rational and logical thinking from the outset, and later on, the use of practical formal methods to improve the quality of our computer systems. Formal methods have come a long way in recent years (for example, DeepSpec) and are increasingly finding their way into practice.

- ▶ Pervasively respect the importance of human issues (for example, with greater emphasis on usability, personal privacy, and people-tolerant interfaces) as well as issues that are less technological (for example, compromises of supply-chain integrity, environmental hazards, and disinformation). Also, independent oversight is often desirable, as for example is the case in aircraft safety, business accountability, and elections.

- ▶ Respect history, study the literature, learn from past mistakes, and benefit from constructive experiences of yours and others.

- ▶ Recognize this list is incomplete and only a beginning. For example, I have not

even mentioned the risks of side channels, speculative execution, direct-memory access from embedded microcontrollers and input-output, and tampering.

As a reminder, some important mantras have been repeated in the Inside Risks archives. Here are just a few:

- ▶ Characterizing potential vulnerabilities inherent in various types of computer-related systems and operational environments, such as automation,<sup>11</sup> clouds,<sup>9</sup> IoT,<sup>5</sup> and AI.<sup>16</sup>


- ▶ System engineering and software engineering as a discipline.<sup>2,15</sup>

- ▶ Theoretically based practice.<sup>4,14,15</sup>

- ▶ Foresighted planning for achieving long-term benefits rather than just short-term gains.<sup>6,8,10</sup>

Note that old wisdom may still be very relevant and insightful, as in the Einstein quote, Norbert Wiener's prescient *Human Use of Human Beings*,<sup>18</sup> and Don Norman's *The Design of Everyday Things*.<sup>13</sup> *Computer-Related Risks*<sup>7</sup> is no exception. Furthermore, there is considerable hope in some recent advances. For example, the CHERI hardware-software architecture and its intra-process compartmentalization<sup>17</sup>—together with its ongoing for-

mal analysis of the hardware specifications—can provide some guidance on how many of the aforementioned desiderata and principles<sup>12</sup> can actually be constructively applied in practice. CertiKos, seL4, and the Green Hills separation kernel are other examples of formal analysis of real system components—albeit just for operating-system microkernels.

Each bulleted item is oversimplified, and the problems that must be faced are complex and far-reaching. System engineers, academics, computer users, and others might wish to reflect on the history of how we reached where we are today, and how theoretical and practical research and development experience might help achieve the desired goals, as well as avoiding known shortcomings and as-yet-unrecognized vulnerabilities. However, the bottom line is that we still have a long way to go toward achieving trustworthy systems. 

#### References

1. Abelson, H. et al. Keys Under Doormats: Mandating Insecurity by Requiring Government Access to All Data and Communications. July 6, 2015. <https://dspace.mit.edu/handle/1721.1/97690>
2. Bellovin, S.M. and Neumann, P.G. The big picture. *Commun. ACM* 61, 11 (Nov. 2018).
3. Landwehr, C.J. et al. Software systems engineering

programmes: A capability approach. In *Journal of Systems and Software* 125 (Mar. 2017), 354–364; Article: JSS9898 doi 10.1016/j.jss.2016.12.016

4. Leveson, N. and Young, W. An integrated approach to safety and security based on system theory. *Commun. ACM* 57, 2 (Feb. 2014).
5. Lindqvist, U. and Neumann, P.G. Risks in the emerging Internet of Things. *Commun. ACM* 55, 2 (Feb. 2017).
6. Neumann, P.G. The foresight saga, redux. *Commun. ACM* 55, 10 (Oct. 2012).
7. Neumann, P.G. *Computer-Related Risks*. Addison-Wesley and ACM Press, 1995.
8. Neumann, P.G. More sight on foresight. *Commun. ACM* 56, 2 (Feb. 2013).
9. Neumann, P.G. Risks and myths of cloud computing and cloud storage. *Commun. ACM* 57, 10 (Oct. 2014).
10. Neumann, P.G. Far-sighted planning for deleterious computer-related events. *Commun. ACM* 58, 2 (Feb. 2015).
11. Neumann, P.G. Risks of automation. *Commun. ACM* 59, 10 (Oct. 2016).
12. Neumann, P.G. Fundamental trustworthiness principles in CHERI. A. Shrobe, D. Shrier, and A. Pentland, Eds. In *New Solutions for Cybersecurity*, MIT Press/Connection Science, 2018, chapter 6.
13. Norman, D. *The Design of Everyday Things*, 2002; revised and expanded edition, 2013.
14. Parnas, D.L. Software engineering: An unconsummated marriage. *Commun. ACM* 40, 9 (Sept. 1997).
15. Parnas, D.L. Risks of undisciplined development. *Commun. ACM* 53, 10 (Oct. 2010).
16. Parnas, D.L. The real risks of artificial intelligence. *Commun. ACM* 60, 10 (Oct. 2017).
17. Watson, R.N.M. Capability Hardware Enhanced RISC Instructions: CHERI Instruction-Set Architecture, Version 7, University of Cambridge, June 2019; <https://www.cl.cam.ac.uk/research/security/ctsrds/cheri/>
18. Wiener, N. *The Human Use of Human Beings*. Houghton Mifflin, 1950, revised 1954.

**Peter G. Neumann** (neumann@csl.sri.com) is Chief Scientist of the SRI International Computer Science Lab, and moderator of the ACM Risks Forum.

Copyright held by author.

AD TK

AD TK