DOI:10.1145/3532631

Peter G. Neumann

Inside Risks

Toward Total-System Trustworthiness

Considering how to achieve the long-term goal to systemically reduce risks.

COMMUNICATIONS' **INSIDE RISKS** columns have long stressed the importance of total-system awareness of riskful situations, some of which may be very difficult to identify in advance. Specifically, the desired properties of the total system should be specified as requirements. Those desired properties are called emergent properties, because they often cannot be derived solely from lower-layer component properties, and appear only with respect to the total system. Unfortunately, additional behavior of the total system may arise—which either defeats the ability to satisfy the desired properties, or demonstrates that the set of required properties was improperly specified.

In this column, I consider some cases in which total-system analysis is of vital importance, but generally very difficult to achieve with adequate assurance. Relevant failures may result from one event or even a combination of problems in hardware, software, networks, operational environments, and of course actions by administrators, users, and misusers. All of the interactions among these entities need to be considered, evaluated, and if potentially deleterious, controlled by whatever means available. The problem to be confronted here is trying to analyze an entire system as a composition of its components, rather than just considering its components individually. In many cases, system failures tend to arise within the interactions and interdependencies among these components, depending on whether a system was designed modularly to minimize disruptive dependencies, with each module carefully specified.

Addressing this problem is a daunting endeavor, even for seasoned developers of critical systems. Whether explicitly defined or implicit, total-system requirements may be highly interdisciplinary, including stringent life-critical requirements for human safety; system survivability with reliability, robustness, resilience, recovery, and fault tolerance; many aspects of security and integrity; guaranteed real-time performance; forensics-worthy accountability, high-integrity evidence, and sound real-time and retrospective analysis; defenses against a wide ranges of physical, electronic, and other adversities; and coverage of numerous potential risks. Ideally, requirements should be very carefully specified at various architectural layers, preferably formally as much as possible in newly developed systems, and especially in particularly vulnerable components. Although this is usually not applicable to legacy systems, it is stated here as a farsighted goal for future developments.

Total-system architectures that must satisfy high-assurance requirements for trustworthiness may necessarily encompass much of what is described here. However, when executed on untrustworthy hardware and untrustworthy networks, the behavior of operating systems and application software should be considered with suspicion, as it suggests desirable emergent properties of the total system may have been compromised, or could easily be (resulting in adverse behavior).

An almost self-evident conclusion is that total-system trustworthiness with respect to realistic requirements under realistic assumptions is a very long-term goal that can never be completely achieved with any realistic sense of assurance. However, many efforts in that direction would be extremely valuable in attempting to withstand many adversities that are uncovered today. Several efforts currently under way are noted in this column, and seem to be small steps in that direction for new systems, although as previously mentioned, much less applicable to existing legacy systems. However, the enormity of the entire challenge should not discourage us from making structural improvements that could help overcome today's shortsightedness.

Hierarchical Layering and Formal Methods

Hierarchically layered designs have considerable potential, but today are found mostly in well-designed operating systems and layered networking protocols. The concept has often been rejected because of erroneous nested efficiency arguments that can be overcome through good design practice. Formal specification languages and formal analysis of software and hardware have become much more widely applied in recent years. Formal specifications can also exist for system requirements, high-level system architectures, hardware ISAs, and actual hardware. Here are just a few early examples (many others are omitted for brevity).

- Dijkstra's THE system[4] provided a conceptual proof that a carefully layered hierarchical locking strategy could never cause a deadlock between layers—although a deadlock within a single layer was discovered years later).
- David Parnas's seminal work on encapsulated abstraction presented advanced design considerations in the early 1970s. It has become a vital part of structured developments.[11]
- The SRI Hierarchical Development Methodology,[13] which was the basis for PSOS[9] (in which seven layers of hardware abstractions and nine layers of system software were formally specified in a non-executable language, so that proofs could have been sewn together for each layer based on their lower ones. PSOS made extensive use of Parnas's work. (Many other methodologies are also used more widely, but mostly with less rigor.)
- Virgil Gligor[5] spearheaded some contemporaneous efforts to formalize higher-layer policy issues in 1998. The Clark-Wilson paper[2] extended the notion of security requirements into an informal representation of generic application-integrity principles.
- More recently, seL4[7,8] and CertiKOS[6] provide significant advances in software hypervisors (the latter internally layered approximately similar to HDM).
- The new CHERI-Arm Morello hardware instruction-set architecture with multiple operating systems[14] includes proofs[10] the ISA satisfies several critical hardware trustworthiness properties. Hardware Morello chips and system-on-chip boards are currently being made available for experimental use by Arm Ltd. That effort is only one step toward trustworthy hardware; CHERI-RISC-V is also specified. As with all other hardware, the consistency of the actual Morello hardware with its ISA specification remains unresolved—that is, the hardware must do exactly what is specified and nothing else (for example, no supply-chain

Addressing this problem is a daunting endeavor, even for seasoned developers of critical systems.

CNM / AD TK

compromises such as added Trojan horses resulting from inserted analog circuitry[15]). In addition, Nirav Dave's Ph.D. thesis[3] extended the Bluespec executable specification language (BluespecSystem Verilog BSV) to BSL, which could enable an extended compilation into both hardware (the lower layers) and software (the upper layers).

A long-term goal for the future would be to have hierarchical proofs (from the hardware up through hypervisors, operating systems, and application code) to prove that specified total-system requirements (with stated assumptions) could be satisfied with some desired measure of assurance. There are still many potential pitfalls (incomplete requirements and specifications, inadequate development and assurance tools, sloppy programming, unreliable systems, malicious attacks, and so forth). However, assurance is necessary for each step along the way to this goal, as well as better analyses of entire systems. Unfortunately, that approach is not applicable to most legacy hardware-software systems, which suggests the long-term approach must be injected early into future technology developments.

Perhaps as an indication that more R&D is needed, DARPA is currently planning a new program called PROVERS: Pipelined Reasoning of Verifiers Enabling Robust Systems for extending formal methods to work at the scale of real systems.

Illustrative Applications

Several relevant application areas in which the compromise of total-system attributes may be of great concern are considered here. In each case, there are difficulties in analyzing the relevant components, but also their embeddings into total systems; proving anything convincingly could be very difficult—if not generally impossible. Furthermore, even if a particular component could somehow be shown to be logically sound by itself (which often seems not to be the case), its compliant total-system behavior may be compromised by exploitation of hardware and operating system flaws that can undermine its integrity, or by poor application programs.

▸ *Cryptography*. Cryptography is sometimes thought of as a panacea. However, overreliance on the very best cryptographic implementations and their applications is especially worrisome, particularly when embedded in hardware or operating systems that can themselves be compromised.

▸ *Real-time systems*. The design of real-time systems with guaranteed performance and fail-safe/fail-soft/fail-secure requirements must anticipate a much wider range of faults and failure modes than laptops. The same is true of some analog-digital and mixed-signal cyberphysical systems. Again, low-level failures can compromise the ability to satisfy those requirements, as can simple application-specific code.

▸ *Election integrity*. Previous Inside Risks columns on election integrity have stressed that every part of the election process is a potential weak link. Existing commercial systems are seriously flawed, and many of the overall systemic weaknesses are external to computer systems and can make the technology more or less irrelevant if the results have been compromised.

▸ *Quantum computing*. In the future, quantum computing and its integration into networking with conventional computing are likely to be fraught with unanticipated problems. Also, the necessary error-correcting coding required in quantum computers may miscorrect results whenever the errors exceed the limits of the coding system. Thus, the choice of the coding system to fit the actual range of hardware failures becomes critical.

The enormity of the entire challenge should not discourage us from making structural improvements that could help overcome today's shortsightedness.

▸ *Multilevel security*. One of the most demanding areas of trustworthy computing and communications involves being able to concurrently deal with different levels of critical security (for example, top secret to unclassified). With very few exceptions, most of the efforts in the 1970s and 1980s assumed implementing the required separation in a software kernel would be good enough. Unfortunately, the available hardware was (and still is) inadequate. Notable exceptions to the software-only approach included Butler Lampson's BCC-500 computer (Berkeley Computing Corp.) in the late 1960s, the hardware-software MLS retrofit for Multics in the early 1970s, and PSOS (which sought to ensure MLS as a strongly typed hardware capability extension) in the mid-1970s.

▸ *Artificial intelligence*. The trustworthiness of systems based on deep learning, neural networks, and many other aspects of what is generally referred to as artificial intelligence is typically difficult to prove or otherwise evaluate, for all possible circumstances. Also, AI elements that require self-adaptation or training may have not been programmed or trained properly for their intended use; also, algorithms and training data may be intentionally or inadvertently biased. Certainly, a trained neural network can do no better than the data it is fed. In generally, the use of AI would seem very risky in life-critical and other systems with stringent requirements—especially where deterministic or demonstrably sound results would be essential (see for example, Parnas[12,a]). Nevertheless, AI is very popular, and is finding many diverse useful applications.

These examples expose just tips of multiple icebergs, but are intended to be suggestive of the difficulties that must be overcome.

In each of these cases, there is also a desirability of having some independent sanity checks ensure the total-system results are correct—or at least within realistic bounds—with respect to the stated requirements. An analogy in formal theorem proving is to use trustworthy proof checkers to check the proofs, although that still assumes the underlying assumptions and the proof checkers are correct and unbiased.

There is also a desirability of having some independent sanity checks ensure the total-system results are correct.

What Is Needed?

The desiderata were established many years ago, but are still not used widely in practice. A grossly oversimplified set might include something like this:

▸ Consider established principles for total-system development and trustworthiness, and invoke those that are most relevant.

▸ Establish well-defined total-system requirements against which evaluations can be made, and specify them formally where possible.

▸ Establish well-defined system architectures, hierarchically defining accessible interfaces at each major interface, from hardware to operating systems and applications (for example, Robinson-Levitt,[13] which was applied to conceptual hardware and software in PSOS, and to the Ford Aerospace KSOS1 MLS kernel in software.

▸ Use formal specifications and formal methods by which formal analysis is possible, particularly in systems with particularly critical requirements. Analysis might include dependency analysis (seeking dependence only on less-trustworthy entities, and avoiding circular dependences that might cause deadlocks), and proofs of essential properties. Hierarchical proofs from the ground up are theoretically supported,[13] but still lurking in the future if they were to span hardware, operating systems, applications, and total-system requirements as much as possible—leaving out-of-scope assumptions clearly stated via itemization of unaddressed or missing requirements, and enumerating threats that remain uncovered.

▸ Address myriad other problems proactively throughout.

Conclusion

Significant progress is being made with some of the steps toward the desired long-term goal of total-system trustworthiness. Of course, all of this is still nowhere near enough, considering all the extrinsic problems we face. However, the goal is nevertheless worth pursuing for new critical systems—to the extent it is realistic. This suggests we must begin now to recognize the relevance of the overall long-term goal. C

a This article refers to many additional references that deserve to be included here, such as Parnas's remarkably prescient early papers.

References

1. Berson, T.A. and Barksdale, G.L., Jr. KSOS: Development Methodology for a Secure Operating System, National Computer Conference, *AFIPS Conference Proceedings 48* (1979), 365–371.
2. Clark, D. and Wilson, D.R. A comparison of commercial and military computer security policies. In *Proceedings of the 1987 Symposium on Security and Privacy*. IEEE Computer Society, Oakland, CA (Apr. 1987), 184–194.
3. Dave, N. A Unified Model for Hardware/Software Co-design, MIT Ph.D. thesis. 2011.
4. Dijkstra, E.W. The structure of the THE multiprogramming system. *Commun. ACM 11*, 5 (May 1968), 341–346.
5. Gligor, V.D. and Gavrila, S.I. Application-oriented security policies and their composition. In *Proceedings of the 1998 Workshop on Security Paradigms*, Cambridge, England, 1998.
6. Gu, R. et al. CertiKOS: An extensible architecture for building certified concurrent OS kernels. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI'16)*. (Nov. 2016), 653–669.
7. Heiser, G., Klein, G., and Andronick, J. seL4 in Australia: From research to real-world trustworthy systems. *Commun. ACM 63*, 4 (Apr. 2020), 72–75.
8. Klein, G. et al. Comprehensive formal verification of an OS microkernel. *ACM Transactions on Computer Systems 32*, 1 (Feb. 2014).
9. Neumann, P.G. et al. A Provably Secure Operating System: The System, Its Applications, and Proofs. SRI International. 1980.
10. Neumann, P.G. Fundamental trustworthiness principles in CHERI. In A. Shrobe, D. Shrier, and A. Pentland, Eds. *New Solutions for Cybersecurity*. MIT Press/Connection Science (Jan. 2018); https://bit.ly/3kgxyt6
11. Nienhuis, K. et al. Rigorous engineering for hardware security: Formal modeling and proof in the CHERI design and implementation process. In *Proceedings of the 36th IEEE Symposium on Security and Privacy*, May 2020.
12. Parnas, D.L., Clements, P.C., and Weiss, D.M. The modular structure of complex systems. *IEEE Transactions on Software Engineering SE-11*, 3 (Mar. 1985), 259–266.
13. Parnas, D.L. The real risks of artificial intelligence. *Commun. ACM* (Oct. 2017).
14. Robinson, L. and Levitt, K.N. Proof techniques for hierarchically structured programs. *Commun. ACM 20*, 4 (Apr. 1977), 271–283.
15. Watson, R.N.M. et al. Cambridge-SRI CHERI-ARM Morello and CHERI-RISC-V; https://www.cl.cam.ac.uk/research/security/ctsrd/cheri/
16. Yang et al. A2: Analog malicious hardware. In *Proceedings of the 2016 IEEE Symposium on Security and Privacy*. IEEE Computer Society.

Peter G. Neumann (neumann@csl.sri.com) is Chief Scientist of the SRI International Computer Science Lab, and has moderated the ACM Risks Forum since its beginning in 1985.