Trusted ⇐ Trustworthy ⇐ Proof
Position Paper

Gernot Heiser

Open Kernel Labs and NICTA and University of New South Wales
Sydney, Australia
gernot@nicta.com.au

July 16, 2008

Abstract

Trusted computing is important, but we argue that it remains an illusion as long as the underlying trusted computing base (TCB) is not trustworthy. We observe that present approaches to trusted computing do not really address this issue, but are trusting TCBs which have not been shown to deserve this trust. We argue that only mathematical proof can ensure the trustworthiness of the TCB. In short: trust requires trustworthiness, which in turn requires proof. We also show that this is achievable.

1 The Security Challenge

There can be little doubt that security, safety and reliability issues in computer systems are becoming increasingly important, even outside the traditional domain of national security uses. One of the reasons is that computer systems, especially embedded systems, are increasingly used in mission-critical, even life-critical scenarios.

Examples where lives are at stake are aeroplanes, cars and medical devices. System reliability and safety are paramount there, and significant effort is generally invested into ensuring this, including certification requirements that focus on software processes and in some cases a degree of formal-method use.

Other devices are treated in a far more nonchalant fashion, yet security violations can have quite significant consequences. Baseband processing in mobile-phone handsets requires approval by certification authorities. Yet, the complete baseband stack may contain millions of lines of code (LOC), and typically all that code executes at the same privilege level in a single, flat address space without memory protection.

A million LOC can safely be assumed to have hundreds (more likely thousands) of bugs, and no-one can seriously expect that the certification process has any significant impact on that amount of defects. Given the programming model, any single bug can, in the worst case, arbitrarily subvert the software. In the case of the phone handset, this could mean jamming the network. If the bug can be triggered externally (or from the user-interface part of the handset software), a distributed denial-of-service attack on the network becomes conceivable, which could within minutes force down the cellular network country-wide. Recovery from such an attack would be very expensive and time consuming.

Evaluation Level	Requirements	Functional Specification	High-Level Design	Low-Level Design	Implementation
EAL1	Informal	Informal	Informal	Informal	Informal
EAL2	Informal	Informal	Informal	Informal	Informal
EAL3	Informal	Informal	Informal	Informal	Informal
EAL4	Informal	Informal	Informal	Informal	Informal
EAL5	*Formal*	Semiformal	Semiformal	Informal	Informal
EAL6	*Formal*	Semiformal	Semiformal	Semiformal	Informal
EAL7	*Formal*	*Formal*	*Formal*	Semiformal	Informal
Trustworthy	*Formal*	*Formal*	*Formal*	*Formal*	*Formal*

Table 1: Evaluation methods used at the different CC evaluation levels.

If this is not disconcerting enough, there are more security issues associated with phone terminals. They are increasingly used to store and access highly-sensitive data and services. Smartphones are used to access the enterprise computing system of the owner's employer, and compromising the device could compromise the corporate IT system. Phones are used for financial transactions (presently mostly small, but there is a tendency to turn them into more full-blown banking terminals), and phones increasingly store sensitive personal information. This implies an increasing reliance on the (even bigger and even less scrutinised) user-interface software stack on the handset. The potential for corporate or personal financial damage and identity theft is clearly mounting.

2 Trust and Trustworthiness

Obviously, as a society we are putting an increasing amount of *trust* in computing systems. Hence, trusted computing is highly relevant for the main stream, not just for specialised "highly secure" systems.

Yet, by and large, the computing systems we routinely trust are far from being *trustworthy* in any real sense. In general they massively violate fundamental security principles, especially the *principle of least authority* (POLA)—their *trusted computing base* (TCB) is far too large. In particular, the operating systems are large (hundreds of kLOC) complex, buggy and impossible to get defect free.

> Trusted computing without a trustworthy TCB is a phantasy.

Initiatives such as the TCG's trusted platform module aim at providing a *trust anchor*, which enable secure boot and secure execution of trusted code. This is a (necessary) start, but it does not solve the fundamental problem of the lacking trustworthiness of the TCB.

Having observed the sorry state of the vast majority of systems that are (like it or not) *trusted* to perform security- or safety-critical operations, we would hope that at least our national security is in good hands.

In fact, in the defence sector we can observe that serious attempts are made to establish the trustworthiness of the system (or at least its TCB). Systems used in defence generally require certification under higher evaluation levels of the Common Criteria [NIS99], which appears a very stringent requirement. But is this really true?

First we note that no operating system kernel (always a critical part of the TCB) has been certified at an evaluation level of more than EAL5 (although it seems that at Green Hills Integrity is close to EAL6 certification) [NSA]. The highest level of Common Criteria (CC) evaluation is EAL7, and that has definitely not yet been reached. Hence, even systems used in the most sensitive defence applications are not considered trustworthy at the highest certification level.

Things become even more worrisome if we look at what CC certification really means. For the purpose of security evaluation, the CC distinguish between the *security requirements* of a system, its *functional specification*, its *high-level design*, its *low-level design*, and its *implementation*. The higher evaluation levels (EAL5 and higher) require formal descriptions of the security requirements, as shown in Table 1. However, only a semi-formal representation of the functional specification and the high-level design is required, with a semi-formal argumentation that they meet the formal security requirements. At EAL6, a semi-formal low-level design is also required.

Only at the (so far unachieved) EAL7 do CC require formal functional specification and high-level design, and formal proofs of their correspondence. The requirement for the low-level design and its correspondence to the high-level design is only semi-formal. *No formal reasoning whatsoever is required for assessing the actual implementation!* Instead, informal arguments of correspondence between design and implementation are used, and in the end, the CC rely on software processes and testing.

But, as Dijkstra famously remarked, testing can only show the presence, not the absence of bugs. It may therefore not be overly surprising (but is nevertheless scary) that the CEO of a leading vendor of operating systems for military use seems to believe in security by obscurity [O'D].

> Real trustworthiness cannot be achieved by testing or stringent software processes. Trustworthiness requires proof—mathematical proof is the only way to gain certainty.

This is our core assertion on the future of trust in computing: we need to enable trusted computing in a real sense, which means that we need real trustworthiness at least in our TCBs. *This can only be achieved by proof.*

And TCBs proved (rather than hand-waved) to be trustworthy are needed just about everywhere, not just in defence. It is needed in everyday devices, such as cars and mobile phones.

3 Requirements

We need to understand what is needed to achieve this level of trustworthiness.

The effort required to build and prove a trustworthy TCB is obviously high, and it must be possible to re-use the results of that effort. This implies that a *general-purpose base* is required on which arbitrary systems can be built, whether for defence use or for commodity articles like mobile phones.

That base must be completely formally verified, providing a complete proof chain from requirements to implementation, as shown in the bottom row of Table 1. The MILS approach [Alves-Foss06] can then be used to build trustworthy systems on top. But we need to keep in mind that *MILS without complete formal verification of the kernel is a fortress built on sand.*

The generality requirement has an important implication: the base must be performant and support the development of well-performing systems on top. This is because it must support

systems with tight energy budgets, as many mobile devices are battery powered. Since battery capacity is improving only slowly, the performance requirement will not go away in the foreseeable future. This means that there is very limited freedom to make the security-vs-performance trade offs which are prominent in security-oriented systems available today.

4 Can it be Achieved?

We claim that such a trustworthy base is possible, and the existence proof of such a system is almost complete. It is the essence of a project conducted at NICTA since the beginning of 2004. The project has developed a new operating-system microkernel called *seL4*. seL4 has been formally specified in Isabelle/HOL [NPW02], and its performance is at par with implementations of the L4 microkernel, which since more than ten years is the performance benchmark for small kernels [LES+97].

Formal, machine-checked proofs have been developed which show that seL4 can satisfy strict isolation requirements [EKE07]. These proofs do not yet cover a complete set of requirements, such as the CC Separation Kernel Protection Profile [IAD07], but this is now only a matter of time.

seL4 has formal high-level and low-level designs, the latter being a formalisation (in Isabelle/HOL) of an executable specification written in the Haskell programming language. Thanks to its executable nature, the low-level design can simulate the actual implementation and can therefore be used to port and test higher-level software components.

A formal proof of the correspondence between functional specification, high-level design and low-level design has been completed [CKS08]. As such, seL4 goes already well beyond the CC requirements even at EAL7, and it is already the most formally analysed general-purpose operating-system kernel in history.

The final step, the formal proof of the correspondence between low-level design and implementation, is in progress. A formalisation of the implementation (in Isabelle/HOL) exists, and the correspondence proof is to be completed later this year. This will result in the first OS kernel that can really support trusted computing.

5 Cost

While it a demonstration that full verification of a high-performance general-purpose OS kernel seems a worthwhile achievement, the issue of cost cannot be ignored. Verification is likely to remain irrelevant if it is unreasonable expensive.

The NICTA project provides a good data point for cost as well. We estimate that by the end of the project (December 2008), the project will have cost around $4–5M. We estimate that on the back of the first project, taking another 10kLOC kernel through full verification will cost no more than $2M. This is to be compared to the industry estimate of $10k/LOC just for CC EAL6 certification, or $100M for 10 10kLOC microkernel! That cost is dominated by the extensive documentation that needs to be created and maintained for CC evaluation, and which is basically irrelevant if the code is formally verified.

In other words, formal verification can be one to two orders of magnitude less expensive than traditional assurance schemes! This experience clearly shows that extending CC by another evaluation level or two, leading up to complete verification, cannot be the right approach. Formal verification must be the basis of an alternative assurance scheme which strips away the

need for expensive processes which are irrelevant if the implementation is proved to satisfy its requirements.

6 Conclusions

We observed that the security and safety challenges facing modern computing systems are massive, yet poorly addressed to date. Security assurance even for the the most sensitive military systems is woefully insufficient, and cannot deliver true trustworthiness.

We claim that real trustworthiness is not only required, it is actually *achievable* and cost-effective, and seL4 is a case in point. Real trustworthiness will, in our view, become a central piece of trusted computing.

References

[Alves-Foss06] Jim Alves-Foss, Paul W. Oman, Carol Taylor, and Scott Harrison. The MILS architecture for high-assurance embedded systems. *International Journal on Embedded Systems*, 2:239–247, 2006.

[CKS08] David Cock, Gerwin Klein, and Thomas Sewell. Secure microkernels, state monads and scalable refinement. In Cesar Munoz and Otmane Ait, editors, *Proceedings of the 21st International Conference on Theorem Proving in Higher Order Logics*, Lecture Notes in Computer Science, Montreal, Canada, August 2008. Springer. To appear.

[EKE07] Dhammika Elkaduwe, Gerwin Klein, and Kevin Elphinstone. Verified protection model of the seL4 microkernel. Technical report, NICTA, October 2007. Available from http://ertos.nicta.com.au/publications/papers/Elkaduwe_GE_07.pdf.

[IAD07] Information Assurance Directorate. *U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness*, June 2007. Version 1.03. http://www.niap-ccevs.org/cc-scheme/pp/pp.cfm/id/pp_skpp_hr_v1.03/.

[LES+97] Jochen Liedtke, Kevin Elphinstone, Sebastian Schönberg, Herrman Härtig, Gernot Heiser, Nayeem Islam, and Trent Jaeger. Achieved IPC performance (still the foundation for extensibility). In *Proceedings of the 6th Workshop on Hot Topics in Operating Systems*, pages 28–31, Cape Cod, MA, USA, May 1997.

[NIS99] US National Institute of Standards. *Common Criteria for IT Security Evaluation*, 1999. ISO Standard 15408. http://csrc.nist.gov/cc/.

[NPW02] Tobias Nipkow, Lawrence Paulson, and Markus Wenzel. Isabelle/HOL–a proof assistant for higher-order logic. In *Volume 2283 of LNCS*. Springer, 2002.

[NSA] The Common Criteria evaluation and validation scheme. http://www.niap-ccevs.org/cc-scheme/in_evaluation/. Accessed May 2008.

[O'D] Dan O'Dowd. Linux security controversy. http://www.ghs.com/linux/unfit.html. Accessed May 2008.

Keywords

trusted computing, trusted computing base, operating systems, assurance, formal verification