29

Reflections on Trusting Trust Revisited

Security is often described as a weak-link phenomenon. Ken Thompson in his 1983 Turing Award Lecture [3] described how a compiler could be modified to plant a Trojan horse into the system's login authentication program so that it would accept a known password. In addition, the C compiler could be altered to propagate this change when it was recompiled from its (unmodified) source code. The system Thompson described was seriously compromised and could never be trusted: even a recompilation from clean source code would yield a Trojaned compiler and login program.

Twenty years later we find efforts such as the Trusted Computing Group (the retooled Trusted Computing Platform Alliance, a 190-company industry work group), Intel's LaGrande, and Microsoft's NGSCB (Next Generation Secure Computing Base, previously known as Palladium) aiming to create secure systems from the ground up [4]. "Trusted Computing" (TC) platforms include specialized hardware or a processor that can monitor a system's boot process to ensure the computer is based on appropriately certified hardware and software. After verifying the machine's hardware state and firmware, the platform can check that the operating system is certified, then load it and transfer control to it. The operating system (for example Microsoft's NGSCB) can similarly verify that only secure untampered applications are loaded and executed—no more doctored C compilers or unauthorized descramblers. Thus, a TC platform can be used to rigorously enforce third-party-mandated security policies such as those needed for digital rights management (DRM) and mandatory access control [1].

Given our nearly unbroken track record of failed security technologies, we should view claims regarding a system's trustworthiness with skepticism. Recently a group managed to run Linux on a Microsoft Xbox—without any hardware modifications [2]. The Xbox, in common with many other game consoles, mobile phones, and even printer cartridges, can be considered an instance of a special-purpose TC platform. Although based on commodity hardware, the Xbox is designed in a way that allows only certified applications (games) to run, thus protecting the licensing revenue stream of its vendor. Earlier attempts to run unauthorized software (such as the Linux kernel) on it required hardware modifications, a prospect that will not be realistic once TC features are part of the CPU (as might be the case with Intel's LaGrande design). The recent attack modifies the saved data of a particular game in a way that renders the trusted game into an untrusted agent that can then be used to boot Linux.

The two attacks, set apart by 20 years, share an interesting parallel structure. Thompson showed us that one cannot trust an application's security policy by examining its source code if the platform's compiler (and presumably also its execution environment) were not trusted. The recent Xbox attack demonstrated that one cannot trust a platform's security policy if the applications running on it cannot be trusted. The moral of the Xbox attack is that implementing on a TC platform a robust DRM, or mandatory access control, or even a more sinister security policy involving outright censorship will not be easy. It is not enough to certify the hardware and have a secure operating system; even a single carelessly written but certified application can be enough to undermine a system's security policy. As an example, a media player could be tricked into saving encrypted content in an unprotected format by exploiting a buffer overflow in its (unrelated) GUI customization (so-called skin) code. Capability machines built in the 1970s used strong typing and a finer granularity object classification and access control schema that would prevent such an attack. However, as Needham and Wilkes concluded from their work on the CAP system, the resultant operating system was too complex and therefore hard to trust and maintain [5]. Those of us who distrust the centralized control over our data and programs that TC platforms and operating systems may enforce can rest assured that the war for total control of computing devices cannot be won. C

References

1. Anderson, R. Cryptography and competition policy—Issues with trusted computing. In *Proceedings of the Workshop on Economics and Information Security* (2003); www.cl.cam.ac.uk/ftp/users/rja14/tcpa.pdf
2. Malda, R. Linux running on Xbox without Modchip!; slashdot.org/article.pl?sid=03/03/30/1337234
3. Thompson, K.L. Reflections on trusting trust. *Commun. ACM 27*, 8 (Aug. 1984), 761–763.
4. Vaughan-Nichols, S.J. How trustworthy is trusted computing? *Computer 36*, 3 (Mar. 2003), 18–20.
5. Wilkes, M.V. and Needham, R.M. *The Cambridge CAP Computer and its Operating System.* Elsevier, London, 1978.

Diomidis Spinellis (dds@aueb.gr) is an assistant professor at the Athens University of Economics and Business and author of *Code Reading* (Addison-Wesley, 2003).

PAUL WATSON