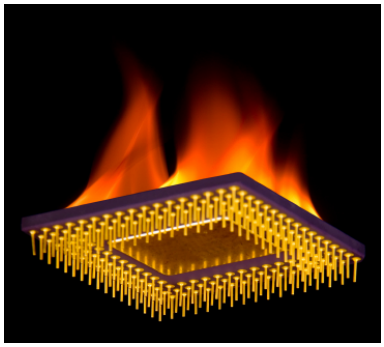
Pipeline
Technology for Service Providers.
www.pipelinepub.com Volume 12, Issue 5

Tamper-proof Computing

By Chris Piedmonte



Ask the vast majority of information technology professionals today if it is possible to completely secure a computing system against cybercrime and the answer you will get is a resounding "no." Through a series of events too long and convoluted to address in this article, we have established a global computing infrastructure that is fundamentally incapable of protecting itself, creating both a playground and a candy store for cybercriminals.

Our current global computing infrastructure is based on technology first conceived almost forty years ago, long before the global network of billions of computers was created to manage our banking, communications, defense, entertainment … even the everyday mundane facts of our lives, like the Instagram of the yogurt parfait we had for breakfast. The open communication protocols of the Internet were designed to insure that data can be transferred and shared under the most adverse of circumstances. The computer architectures in vogue today are the great, great grandchildren of the original IBM PC, which was best secured by locking your office door at night. These technologies were simply not designed with the forethought of their then far future applications in secure and trusted computing.

We are now faced with the consequences of this legacy – the loss of billions of dollars annually due to cybercrime and the social and political ramifications of not being able to secure knowledge that we would prefer to keep private. To date, we have attempted to address this problem by creating firewalls, scatter-shot use of cryptography, anti-virus software, intrusion detection systems, two-phase authentication and a variety of other methods attempting to Band-Aid our flawed infrastructure. What is needed is a fundamental overhaul of the architecture of our computing infrastructure and the introduction of better security from the ground up. What is needed is tamper-proof computing[1].

Search the web for information related to the term "tamper-proof computing" and you will find plenty of material on tamper-evident and tamper-resistant computing, but you will find very little on tamper-proof computing. So, perhaps we should best start with a definition of what exactly is tamper-proof computing. To be tamper-proof, the computing system must not permit itself to be altered while running in a production environment through means readily available to those with access to the system. That requires that it have the following characteristics:

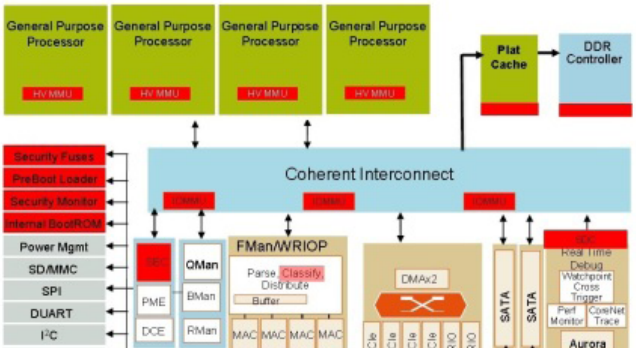
1. hardware comprising the computing system cannot be added, changed or removed,
2. software cannot be added, changed or deleted by a single individual,
3. software cannot be injected and executed from a remote source or via data,
4. proprietary "secrets" such as private keys and checksum values are undiscoverable,
5. hardware and software can always be verified to be known and trusted, and
6. attempts to compromise the system are prevented or result in protective actions.

These six characteristics must be enforced from the time power is applied to the computing system until it is shut down and decommissioned from production use. To accomplish this requires new hardware and software only now becoming available. Leading the way toward the future of tamper-proof computing are innovative computer technology companies like Freescale Semiconductor (soon to be part of NXP).

System-on-a chip

Freescale has been working for many years to develop their Trust Architecture now available in their QorIQ (pronounced core-IQ) and Layerscape line of system-on-a-chip (SoC) products (see Figure 1). These SoCs are available in both POWER and ARM compatible versions and are designed to run with exceptional performance and compatibility within a modern computing infrastructure.

When combined with operating systems and tools designed specifically to take advantage of the Freescale Trust Architecture, such as the Suvola LINUX Distribution, we now have the basis for tamper-proof computing, or what Freescale calls a trustworthy system. Freescale defines a trustworthy system as a system which does what its stakeholders expect it to do, resisting attackers with both remote and physical access, else it fails safe. Using the Freescale Trust Architecture, each of the six requirements for tamper-proof computing can be achieved. Let's revisit each of those six requirements and examine how a tamper-proof platform developed on top of the Freescale Trust Architecture achieves them.



Hardware cannot be added, changed or removed: During the booting up of the system, a secure boot process can detect unauthorized modifications to the system configuration, such as device trees or hardware certificates, and prevent the continuation of the boot process. Attempts to add, remove or substitute hardware prior to booting are therefore prevented. After the system is up and running, the external attachment of other devices, via USB, Ethernet or other interfaces is also detectable and these devices can be ignored or disabled to prevent the unauthorized transfer of data to or from these devices.

> *To be tamper-proof, the computing system must not permit itself to be altered while running in a production environment through means readily available to those with access to the system.*

Software cannot be added, changed or deleted (by a single individual): To add, change or delete software requires that a user have access and permission within the computing file system. The Suvola LINUX distribution includes a file system that requires a two-party authentication system to install executable code of any type, thereby preventing the modification of executable software by a single individual. Further, all software authorized to run on a tamper-proof computing platform must be signed by the manufacturer (author), the user of the platform (stakeholder in Freescale terminology), and keyed to a secret within the SoC to guarantee its authenticity. The algorithms for software installation are known only to the Suvola software installation toolset and are rotated frequently to provide additional security.

Software cannot be injected from a remote source or via data: The computing platform provides for a run-time integrity checker, which can determine if the software about to be submitted to a CPU for execution is known to the system or not. Unknown software is simply not permitted to execute. Known software must match a series of signature and validation checks to ensure that it has not been tampered with prior to execution. While in memory, code is continuously monitored to ensure that it has not been tampered with since it was originally loaded. The SoC can also be configured (by a

permanent fusing mechanism within the SoC) to disable debug interrupts, physically partition cores and memory, and close other "back doors" that can be used to alter code or execution when in production by a sophisticated attacker.

Secrets are undiscoverable: The Freescale Trust Architecture provides a write-only storage area that is only accessible by components of the SoC Trust Architecture, such as the hardware-based cryptography and boot subsystems. This storage area is used to secure cryptography private keys and other "secrets" so that they cannot be discovered (short of destroying the chip and attempting to read the molecular state of the storage area via scanning electron microscope), but are readily usable by the SoC itself.

Hardware and software can always be verified: Given that the configuration of the hardware and software are known to the SoC and verifiable via its secrets, the system is capable of continuously monitoring both hardware and software for unauthorized modifications. This provides a means of ensuring that the hardware and software comprising the system has not been tampered with.

Attempts to compromise the system are prevented: Other advanced features designed to protect the secrets of the system include the ability to zero-out all secrets if the computing platform is physically tampered with, operated out of voltage, temperature or AC power frequency ranges in attempts to compromise the SoC. The cryptographic subsystem also provides for timing equalization to prevent attempts to "guess" the algorithms via careful analysis of cryptographic processing times.

But how do we know that a system's built around these six tenants really is tamper-proof? As always, proving a negative is really not readily possible. It's the equivalent of trying to prove that the airbags in your car will go off if you drive into a brick wall. We all expect that they will, but how do we prove that short of totaling our new Tesla Model S? The same is true of tamper-proof computing. We can't prove it is tamper-proof, only prove that it is not if and when at some point in the future someone figures out how to get around the six tenants we've set out herein. But do we really require proof to use the clearly advantageous capabilities of a tamper-proof computing system? Clearly, as with the air bags in our modern vehicles, it's better to have and trust that they will work, rather than not have them at all. As more and more tamper-proof computing platforms are deployed, time will surely indicate that having them in place is better than not, and that the value they deliver in thwarting cybercrime is both measurable and significant.

As with the air bags in our modern vehicles, it's better to have and trust that they will work, rather than not have them at all.

---

[1] Most of the major publicly known breaches (Target, Home Depot, etc.) could have been prevented had this technology existed, as they all required that the systems be tampered with to achieve the criminal objective.