A Vendor-Agnostic Root of Trust for Measurement

Authors: Jonathan McCune, Rick Altherr

Abstract

We report the success of a project that Google performed as a proof-of-concept for increasing confidence in first-instruction integrity across a variety of server and peripheral environments. We begin by motivating the problem of first-instruction integrity and share the lessons learned from our proof-of-concept implementation. Our goal in sharing this information is to increase industry support and engagement for similar designs. Notable features include a vendor-agnostic capability to interpose on the SPI peripheral bus (from which bootstrap firmware is loaded upon power-on in a wide variety of devices today) without negatively impacting the efficacy of any existing vendor- or device-specific integrity mechanisms, thereby providing additional defense-in-depth.

Original Publication 2017-09-08
Last revised 2017-09-08
Version Number: 1.0
Link: https://research.google.com/pubs/pub46352.html

Introduction

A security primitive becomes more useful as it can integrate more fully into essential board functions. This particularly includes the ability to verify and deny execution of start-up code, along with release of cryptographic capabilities. Google's Titan security chip [3,4] design incorporates two means to make it essential to board initialization and operation:

- A physical requirement that it mediate all board accesses to bootstrap code; and
- The ability to pull the board into reset and hold it there at any time.

It aims to do this in the least expensive and least destabilizing way, by interposing on the SPI bus and exposing a simple reset connection. This materially advances Titan's ability to help secure a board design, beyond the classic "side car" style of operation that can be bypassed arbitrarily until a release of cryptographic capabilities is required.

Recent years have given rise to varied new hardware security primitives, such as the Trusted Platform Module, AMD SEV, Intel SGX, and Intel Boot Guard. Each of these depends on a silicon root of trust that is controlled by its manufacturer, is tightly integrated with the desired functionality of the system, and is difficult to separate out. By adding a physically distinct chip with an easily understood and controlled root of trust, a design can achieve defense in depth against single-point compromise of boot integrity. This is particularly true if the chip design is open sourced and available for inspection and third party implementation.

We report the success of a project that we performed for increasing our confidence in first-instruction integrity across a variety of environments, in the hope of increasing industry support and engagement for similar designs.

Problem Statement

"First Instruction" Integrity

Modern devices are complex and identifying the "first instruction" is itself a subtle and challenging proposition. For the purposes of this discussion we consider the first instruction to be the first data loaded from mutable non-volatile media. This could be a flash chip containing Coreboot firmware, OpenPOWER firmware, UEFI firmware, or even CPU/chipset-specific content such as microcode patches or out-of-band manageability functionality such as the Intel ME.

Some CPU and chipset vendors try to address the first-instruction integrity problem with features such as Intel Boot Guard. While such solutions offer an interesting layer of protection, they are by no means panaceas. Some problems seem to be fundamental:

- They are vendor- or product-specific and are not readily portable to new platforms.
- They are inextricably bound to the CPU/chipset. Critical system components such as the firmware image for a BMC or NIC are not covered.

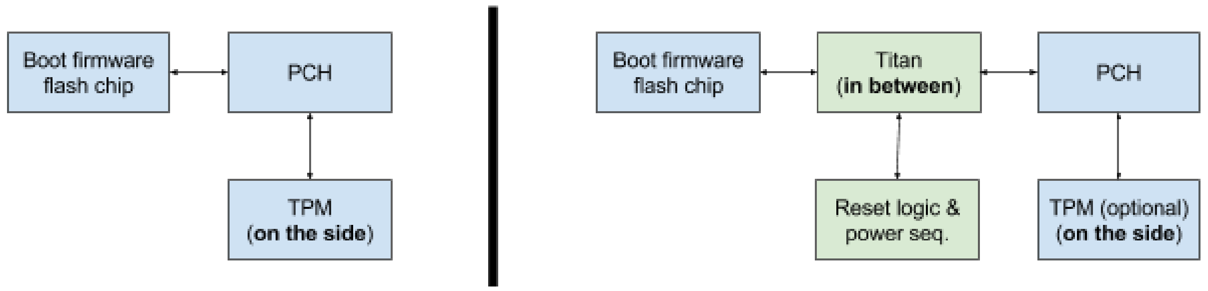
Other problems arise mostly in the context of specific implementations:

- They are opaque, so it is difficult to reason about the level of protection truly offered or their ability to recover from discovered vulnerabilities.
- They require a business arrangement between organizations. This puts them out of reach for smaller organizations and (perhaps most importantly) complicates the efforts of independent security researchers to study and evaluate their effectiveness.
- If the physical owner of a device is not able to execute code of their choice, they may encounter issues of device ownership.

A Dedicated Boot Flash Interposer

We designed a security chip for Google with a goal of verifying integrity from the first instruction. It is a single-chip SPI-bus interposer that is installed between a slave SPI-flash chip and the SPI master. This is one of the features of Google's Titan security chip [3,4]. We have explored its use in a variety of environments, including the SPI flash chip connected to an Intel PCH providing boot firmware, a BMC's boot firmware, and boot firmware for the microcontroller on a NIC.

Beyond interposing on the SPI bus, Titan is also integrated with our boards' power and reset logic. This makes it an ongoing essential element to the boot sequence and allows it to disable the system at any time. By contrast, a TPM typically sits **on the side**. The following diagram describes the distinction:



This design allows Titan to participate in power-sequencing and platform reset. This ensures that the order in which components begin operating is consistent with the security model, and reduces the opportunity for time-of-check time-of-use attacks in the face of malicious peripheral devices. It can also communicate with any BMC that may be present. Indeed, in some designs Titan may authenticate the BMC firmware responsible for authenticating host boot firmware (in a transitive fashion).

An early design where Titan authenticates BMC firmware is available in a publicly available board design, with complete schematics, in the POWER9 Zaius OpenPOWER / Open Compute Server [5]. We continue to evolve the design and look forward to contributing further updates in the near future.

Titan can authenticate the full contents of the slave flash chip, serve as a root of trust for measurement and provide a CPU architecture-independent "first-instruction" integrity property. This includes mutable and opaque regions for chipsets or CPUs whose contents may not be completely understood by Titan but for which the intended values are known [6]. Examples include:

- Intel ME
- Intel GbE firmware
- UEFI NVRAM variable store
- Boot device eligibility and ordering

We have found that this functionality can peacefully coexist with - and in fact be complementary to - vendor-specific technologies such as Intel Boot Guard. This is an opportunity to provide defense-in-depth. An attacker would need to circumvent both technologies, or an insider in one but not the other organization would be unable to circumvent both mechanisms.

As Titan can serve as a SPI bus master to the boot firmware flash chip, it is able to manage the contents of the slave flash chip without higher-level software support.

- Titan can facilitate A/B updates of firmware images to minimize system unavailability during updates, without needing to trust host-side execution (e.g., SMM) or perform the update while the previous firmware is actively executing (e.g., early in the boot process, slowing system boot). Existing system software and flash chip contents are trusted only for the availability of the update mechanism. (Titan can cryptographically attest to the flash chip's contents, again only trusting the host system to provide availability.)

- Titan can leverage the larger capacity of non-volatile media offered by the slave flash chip. This is of interest for Titan's more traditional role as a security-focused chip. For example, it can offer more and higher-QPS monotonic counters for uses such as attested audit logging and storing wrapped secrets without depending on the OS or the complexity of disk I/O.

Titan is also a generally programmable tamper-resistant cryptographic device. We can update and adjust the security policy as experience and conditions warrant. While the preceding text has focused on Titan's application on server motherboards, we also employ Titan on in-house peripheral devices such as Google's purpose-built network interface card. In addition to yielding the aforementioned benefits for the NIC itself (considering that a modern NIC is itself a powerful computing device), we can begin to bind a complete platform into a single entity using cryptographic protocols. This further encumbers attacks that may attempt the introduction of unknown peripherals.

Challenges

SPI was not designed to be interposed. To provide time for Titan to make decisions based on the incoming SPI data, the SPI bus is run at a lower frequency than normal. This has implications on boot time as an initial read of firmware will take longer. Thanks to Titan's interposing, we can use an A/B update scheme to take the slow write and verification operations off the critical path for reboot, though this costs roughly twice the space on the slave flash chip. Future variants may support eSPI to enable broader applicability and faster boot times.

On a motherboard, Titan's integration with the board-level reset logic gives us the option of holding off reboot to give Titan time to complete any remaining operations (e.g., transitioning which A/B image is active). Engineering such board-level reset integration was a unique challenge and deviates from existing board designs to a larger extent than just the SPI bus wiring.

On a peripheral, the slower SPI bus speeds can make meeting PCIe bus enumeration timings challenging. For example, a peripheral may need to boot its own internal microcontroller from a SPI flash chip before it becomes fully operational. To date we have been able to meet the deadlines without proprietary logic to delay bus enumeration, but the margins are tight, and required additional engineering effort during peripheral bring-up.

As an exploratory and ambitious project, we were not certain that Titan's interposing would be sufficiently performant or reliable to use in production. The schedule risk for a production server project was a significant concern. This added complexity to our board designs since we needed an alternative where Titan was either absent or present but not interposing on the SPI bus. We expect this situation to improve in the future now that there are fewer unknowns. By testing in a production environment, we have gained confidence in the design and no longer anticipate requiring removal as a failure mitigation strategy. Instead, removal has shown its value as a solution for providing open server designs that allow the end-user to supply their own security module.

Additional Security Functionality

While Titan was not specifically designed to be a TPM, it addresses a similar set of issues that motivate inclusion of TPMs in other platforms. Titan is complementary to a TPM and can work in concert with a TPM. Titan utilizes manufacturing, supply chain, and provisioning processes that would be unsurprising to those familiar with hardware security devices. Titan employs a silicon root of trust, and a robust rollback protection and TCB recovery scheme for its own internal firmware. It employs the defense-in-depth mechanisms reasonably expected of any modern tamper-resistant Secure Element.

Titan includes a suite of well-known symmetric and asymmetric cryptographic primitives. It is capable of generating and serving as a good steward of cryptographic secrets and supporting values, such as counters or other small data. These are passive functions in the sense that these operations are executed in response to commands from code executing elsewhere on the platform.

Platform Ownership

Any cryptographic root of trust gives rise to issues of control. We posit that an independent component with this responsibility is more transparent, and more likely to lead to the development of an ecosystem with more distributed control. Large, well-resourced organizations such as cloud infrastructure providers can invest in their own components. OEMs, smaller organizations, or motivated individuals can elect for Titan-like devices that allow provisioning of trust during board configuration. Modulo resistance to physical tampering, this enables capable individuals to enjoy the majority of the same security benefits as large organizations, with fewer concerns over who owns the platform.

Conclusions

Proprietary and bespoke solutions to firmware integrity and verified boot exist today. They have shown their utility in resisting attack. However, the existing level of incompatibility and requirement for extensive per-platform work puts these technologies out of reach for all but the most well-resourced organizations. Even for those organizations, these mechanisms are expensive in that they have not historically amortized well across platform generations.

This whitepaper has introduced some conceptual detail about the Titan project at Google that we believe may be of interest to the industry. We have tried to compare Titan with alternative mechanisms available in the industry, and show why the additional work towards a Titan-like device is warranted. While Google does derive some security benefit from the fact that Titan is “in-house”, our main goal is to motivate the general utility of such devices to the server industry as a whole. Infrastructure security is important for everyone, and we wish to encourage the industry to pursue many of these same ideas, while remaining focused on component interoperability.

References

[1] http://www.opencompute.org/wiki/Server/SpecsAndDesigns

[2] https://cloud.google.com/security/security-design/

[3] https://blog.google/topics/google-cloud/bolstering-security-across-google-cloud/

[4] https://cloudplatform.googleblog.com/2017/08/Titan-in-depth-security-in-plaintext.html

[5] https://github.com/opencomputeproject/zaius-barreleye-g2

[6] https://www.chromium.org/chromium-os/firmware-porting-guide/fmap