Firmware Integrity in the Cloud Data Center

cloud security alliance®

Acknowledgements

Lead Organizations

Cloud Security Industry Summit

The Cloud Security Industry Summit (CSIS) is a group of Cloud Service Providers (CSPs) and stakeholders in cloud, with a mission to evolve faith in cloud computing for the broad benefit of Enterprise and Cloud Service Providers, partnering an industry team evolving a coordinated approach for cloud security. The group includes members from top cloud service providers including 1&1, IBM, Cloud Security Alliance, Microsoft, Oracle, Rackspace, Swisscom and others. Intel Corporation serves as facilitator for the group.

Cloud Security Alliance

The CSA is the world's leading organization dedicated to defining and raising awareness of best practices to help ensure a secure cloud computing environment. CSA harnesses the subject matter expertise of industry practitioners, associations, governments, and its corporate and individual members to offer cloud security-specific research, education, certification, events, and products. CSA's activities, knowledge, and extensive network benefit the entire community impacted by cloud — from providers and customers, to governments, entrepreneurs, and the assurance industry — and provide a forum through which diverse parties can work together to create and maintain a trusted cloud ecosystem.

Usage of Position Papers

The position papers are not designed as prescriptive requirements but rather as input for industry organizations and hardware vendors developing roadmaps and/or requirements relating to cloud adoption. These are freely available, however please reference CSIS and CSA if you utilize these in RFPs, RFQs, publications, etc.

Table of Contents

Overview

The CSIS has dedicated a technical working group to focus on the cloud supply chain. The purpose of the research initiative is to:

- Bring together leading cloud service provider experts who are facing similar external security threats at global scale
- Share threat intelligence and Best Known Methods (BKMs) that provide value to all
- Define and test mechanisms for incident response, containment and recovery
- Enable members to deploy these techniques more broadly to build end user confidence in public cloud and reduce cost/credibility risk/impact for CSPs in the event of a breach.

The objective of the research is to focus on supply chain assurance in a cloud environment. With the rise in server firmware level threats, such as UEFI (Unified Extensible Firmware Interface) rootkits and other server firmware backdoors (e.g. the Equation Group backdoor in HDD firmware[1]), and looking at the current state of industry regarding server firmware protections, combined with the evolving new standardization and regulations in this space, the working group determined that there is a need and an opportunity to advance this area.

The technical work group has worked towards identifying mechanisms to verify supply chain security from component to system to solution. Due to the increasing level of sophistication of attackers and nation state threat mitigations, it is critical to build a new generation of servers that are more secure than ever.

The research papers from this working group are considered living documents and will be iterated accordingly as needed.

Introduction / Problem Space

This paper presents the point of view from key stakeholders in datacenter development regarding how to build cloud infrastructure using secure servers and in order to enable customers to trust the cloud provider’s infrastructure at the hardware/firmware level.

In general, security of a cloud server at the firmware level is comprised of two equally important aspects – integrity and quality of the firmware code, as defined below:

1. Integrity – having a cloud server that can be verifiably trusted to execute known firmware. “Known” in this context means full integrity – the server runs the firmware that the cloud provider intended it to run and the cloud provider can verify that with full trust.
2. Quality – having the cloud server run high quality firmware, which does what it was meant to do, and was developed and maintained in compliance with industry security best practices, to minimize chances of vulnerabilities at the firmware level.

[1] https://www.kaspersky.com/blog/equation-hdd-malware/7623/

This document focuses on the aspect of server firmware integrity. Building firmware with high code quality and minimizing potential for vulnerabilities at the firmware level is an issue that the hardware/firmware industry needs to address better than it does today, but is left as a potential topic for a future publication.

The scope in the term “firmware” for this document refers to any code, micro-sequencing, or operating instructions that are located on programmable storage elements. This includes all firmware on a system, including boot firmware (e.g. UEFI firmware, UEFI firmware drivers that are loaded by the UEFI and executed by the main processor), as well as firmware running on symbiont devices that rely on the main system as their root of trust. Differences in approach and solutions to making such firmware secure will be called out specifically in the relevant context.

Scope and Approach of this paper

This paper takes the newly drafted NIST 800-193[2], Platform Firmware Resiliency Guidelines, as a good base set of integrity requirements which, when met, will help raise expectations in terms of building secure servers.

The paper’s scope is limited to what is applicable to building servers that power the cloud. This helps focus the requirements to cloud specific use-cases.

Also, this document represents the working group’s opinion on how the industry could meet those requirements without cloud vendors having to design and build specialized hardware. Requiring specially designed hardware is not always achievable by the industry as a whole (the specific gaps are called out in the next section), so we would like to encourage the industry to standardize and incorporate the necessary features into commodity hardware, which in turn will help raise the industry-wide level of security at the firmware level.

Within this scope, this document identifies:

- Gaps in the industry which makes it difficult to meet the NIST requirements with ‘standard’ commodity servers
- Ways to build servers designed to meet the NIST requirements, including calling out missing technology when applicable.
- Additional requirements that could further strengthen the level of security of servers

Out of Scope

As mentioned above, security is also dependent on firmware code quality. The NIST document includes very little information about coding best practices and building high quality firmware, including, for example, static and dynamic code analysis, firmware build procedures, updates, patching SLA’s, and use of third party libraries.

While it is left out of scope of this paper, we would like to encourage the industry and NIST to find ways to provide cloud providers with better assurances on this front, which in turn can be used to provide better assurances to cloud customers.

[2] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-193.pdf

The NIST Approach for Implementing a Secure Cloud Server

This section summarizes the NIST requirements as described in the newly evolving 800-193. The next section highlights the gaps in the industry in meeting those requirements and suggests ways to address these challenges to help close the gaps.

The following list is a short summary of the key areas addressed in the NIST document:

- **Core principles of protection, detection, and recovery**
    - Protection – prevent firmware from being corrupted.
    - Detection – know when a corruption occurred.
    - Recovery – ability to regain integrity after corruption.
- Systems are classified into three levels: **protected, recoverable, and resilient**. Resilient is the highest level and implies all protect, detection, and recovery requirements are supported.
- **Roots of trust and Chains of trusts** – are the core mechanisms used to ensure integrity by linking system integrity verification all the way back to a trusted component/entity in the system (the root component that is always in a known state)
- **Symbiont devices** – devices on a system that relies wholly or partially on another device (the host device, such as the parent system) to establish the roots and chains of trust.
- **Firmware update mechanisms** – considered as a central tenet required to achieve protection given the ability to change firmware. To be considered secure, all firmware updates need to be authenticated (i.e. performed by a known entity) and authorized.

Some concepts relevant for cloud servers are mentioned but left open for vendor consideration in the NIST paper:

- **Management** - how management is performed can impact the security of a system. However, the NIST paper leaves the impact of management open due to different requirements by customers on how to be able to manage their systems.
- **Authorization mechanisms** – how authorization is performed (e.g. remotely or by requiring physical presence) is left open due to the complexity of the requirements that would be needed to enforce strong and well-defined authorization.
- **Network vs. Local-Assisted recovery** – both options are allowed, but NIST does not define how to securely enable network-based recovery.
- **Automated vs. Manual recovery** – again, NIST 800-193 allows for both but does not define how to perform automated or manual recovery in a secure way.
- **Event Logging** – recognized as useful but not mandated or standardized.

The next section discusses where the working group stands toward current state of the industry compared to these requirements, and how scoping the problem to cloud server helps define more specific requirements for the areas that were left open for vendor consideration by NIST.

Industry-Level Gaps for Implementing Secure Cloud Servers

We believe that the industry is relatively mature when it comes to building “protected” systems as defined by NIST 800-193. Protection can be generally achieved by having all firmware signed and by validating firmware integrity during boot and during updates.

As of 2017, there are notable industry gaps in the following fronts, which we consider high priority issues that need to be addressed by the industry of hardware manufacturers:

1. **First-instruction integrity** – Ability to ensure integrity of the first instruction (the first code or data loaded from mutable non-volatile media), in a way that is verifiable by the cloud provider and not just by the manufacturer
2. **Chain-of-Trust for peripherals** – Ability to leverage the host root of trust and other roots of trust to create a chain of trust to peripherals (e.g. for PCIe devices or other symbiont devices)
3. **Automatable Recovery** – Ability to perform automated recovery back to a known boot-time state upon detection of corrupted firmware (after initial boot)

For recovery, and related to the NIST approach to this area, the working group members believe that in the context of cloud servers, the goal must be to enable Cloud providers with the ability to **control and automate** the recovery process at scale. Manual recovery simply doesn’t scale to the cloud. In addition, depending on the implementation of the critical-to-boot firmware (mutable or not), a cloud server might also need to have the ability to perform fully **local** (non-network dependent) and **automatic** self-recovery of the critical-to-boot portion(s) of the firmware.

The next few sections call out some specific comments to the NIST requirements, and follow the same structure of that doc to allow easier mapping to it.

Roots of trusts / Chains of trusts (NIST Section 4.1)

Within cloud servers, having an immutable RoT for reporting (RTR) is usually addressed by having a TPM, or devices with similar capabilities, on the system board. This has become a common practice.

Industry challenges still exist on the following fronts:

- There is no standard way or common practice to have RoT built into peripherals (e.g. PCIe cards, externally connected devices such as JBODs)
- There is no standard way to anchor peripherals’ CoT in the servers’ RoT in a manner that crosses device boundaries (applicable to 4.1.1.(8))

Protection (NIST Section 4.2)

We agree with the NIST statement regarding UEFI that protecting firmware has become common practice and is readily achievable.

Challenges still exist on how cloud providers can verify the first-instruction other than UEFI-related firmware. The NIST paper addresses firmware being loaded beyond UEFI, but cloud providers still need to "blindly" trust the hardware manufacturer on this front. It would be preferred if there were a certification program or if vendors published more transparent and frequent information about the firmware that is loaded and executed independently and not under the control of UEFI/Platform Firmware.

Detection (NIST Section 4.3)

As stated above, we believe this is indeed an area of notable gaps in the industry and the NIST set of requirements are calling out the right set of requirements.

Scoping the challenge to building clouds makes this problem more achievable than in the generic case of any platform:

- wrt/ NIST section 4.3.1 subsection 2.d and section 4.3.2 subsection 2 – In cloud environments, the number of expected configurations is only in the range of 10's of configurations, which makes it possible to verify not just code but also the expected configurations. Platforms should be aware of correct boot flow and expected configurations, and any deviation from it should trigger additional checks and possibly a recovery action.
- Signature checking is insufficient as a detection mechanism because it does not distinguish between different versions of signed firmware (e.g. a signed version with known vulnerabilities). The RTD/CTD should compute hashes of the firmware, or in the case of a symbiont devices, provide a mechanism whereby the host RTD/CTD can directly read the firmware and compute its hash. This mechanism should be documented such that it does not depend on any OS/EFI drivers provided by the manufacturer.

Recovery (NIST Section 4.4)

It is critical that recovery actions can be performed to restore all the components in a system to a consistent, known state. For example, recovering just a corrupted BIOS image without recovering other components to compatible revisions could lead to a PDoS due to possible FW incompatibility issues.

- The platform administrator must be able to initiate recovery on-demand in order to ensure that all devices on the platform reach a consistent state.
- As mentioned above, it is required that this recovery can be automated (i.e. it does not require physical presence to initiate).
- The RTRec must be protected independently of the running firmware (e.g. by being implemented in an immutable component, such as ROM) in order to ensure recovery is possible even when the firmware has been corrupted.
- Symbiont devices with a non-local recovery mechanism must ensure that the recovery mechanism is available regardless of any modification or corruption of the existing firmware.

Final Thoughts on UEFI

Option ROMs and trust in them has been a long standing issue. Simply put, it is too easy to sign almost any Option ROM, and UEFI implementations still support loading even unsigned Option ROMs and execute it by the host processor.

Working group members believe that:

- Un-signed Option ROMs represent an unnecessary risk to modern cloud servers. In secure servers, all firmware that is put on Option ROMs must be signed and needs to be measured and verified by the host before executing it.
- In addition, a cloud server should support the ability to only load Option ROMs that were pre-validated/approved by the cloud service provider. This adds a layer of security which prevents new hardware from being added to a server, even when signed, unless it was also deemed necessary for the operation of that server. Running less Option ROM code helps reduce the attack surface and helps mitigate some threats related to physical access to a server and the inserting of unwanted hardware into it. One way to achieve the secure addition of new hardware to a server is to support dual-signing and enable the cloud service provider to sign the desired Option ROMs for its own servers.