Draft NISTIR 8320A

Hardware-Enabled Security:

Container Platform Security Prototype

Michael Bartock
Murugiah Souppaya
Jerry Wheeler
Tim Knoll
Uttam Shetty
Ryan Savino
Joseprabu Inbaraj
Stefano Righi
Karen Scarfone

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8320A-draft

National Institute of
Standards and Technology
U.S. Department of Commerce

Draft NISTIR 8320A

Hardware-Enabled Security:

Container Platform Security Prototype

Michael Bartock
Murugiah Souppaya
Computer Security Division
Information Technology Laboratory

Jerry Wheeler
Tim Knoll
Uttam Shetty
Ryan Savino
Intel Corporation
Santa Clara, California

Joseprabu Inbaraj
Stefano Righi
AMI
Atlanta, GA

Karen Scarfone
Scarfone Cybersecurity
Clifton, VA

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8320A-draft

December 2020

U.S. Department of Commerce
Wilbur L. Ross, Jr., Secretary

National Institute of Standards and Technology
Walter Copan, NIST Director and Under Secretary of Commerce for Standards and Technology

National Institute of Standards and Technology Interagency or Internal Report 8320A
49 pages (December 2020)

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8320A-draft

Public comment period: *December 7, 2020* through *January 29, 2021*

National Institute of Standards and Technology
Attn: Applied Cybersecurity Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 2000) Gaithersburg, MD 20899-2000
Email: hwsec@nist.gov

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in federal information systems.

Abstract

In today's cloud data centers and edge computing, attack surfaces have significantly increased, hacking has become industrialized, and most security control implementations are not coherent or consistent. The foundation of any data center or edge computing security strategy should be securing the platform on which data and workloads will be executed and accessed. The physical platform represents the first layer for any layered security approach and provides the initial protections to help ensure that higher-layer security controls can be trusted. This report explains an approach based on hardware-enabled security techniques and technologies for safeguarding container deployments in multi-tenant cloud environments. It also describes a proof-of-concept implementation of the approach—a prototype—that is intended to be a blueprint or template for the general security community.

Keywords

container; hardware-enabled security; hardware root of trust; platform security; trusted compute pool; virtualization.

Acknowledgments

The authors thank everyone who contributed their time and expertise to the development of this report, in particular Eric Johnson, Muthukkumaran Ramalingam, and Madhan B. Santharam from AMI.

Audience

The primary audiences for this report are security professionals, such as security engineers and architects; system administrators and other information technology (IT) professionals for cloud service providers; and hardware, firmware, and software developers who may be able to leverage hardware-enabled security techniques and technologies to improve platform security for container deployments in multi-tenant cloud environments.

Call for Patent Claims

This public review includes a call for information on essential patent claims (claims whose use would be required for compliance with the guidance or requirements in this Information Technology Laboratory (ITL) draft publication). Such guidance and/or requirements may be directly stated in this ITL Publication or by reference to another publication. This call also includes disclosure, where known, of the existence of pending U.S. or foreign patent applications relating to this ITL draft publication and of any relevant unexpired U.S. or foreign patents.

ITL may require from the patent holder, or a party authorized to make assurances on its behalf, in written or electronic form, either:

a) assurance in the form of a general disclaimer to the effect that such party does not hold and does not currently intend holding any essential patent claim(s); or

b) assurance that a license to such essential patent claim(s) will be made available to applicants desiring to utilize the license for the purpose of complying with the guidance or requirements in this ITL draft publication either:

   i. under reasonable terms and conditions that are demonstrably free of any unfair discrimination; or
   ii. without compensation and under reasonable terms and conditions that are demonstrably free of any unfair discrimination.

Such assurance shall indicate that the patent holder (or third party authorized to make assurances on its behalf) will include in any documents transferring ownership of patents subject to the assurance, provisions sufficient to ensure that the commitments in the assurance are binding on the transferee, and that the transferee will similarly include appropriate provisions in the event of future transfers with the goal of binding each successor-in-interest.

The assurance shall also indicate that it is intended to be binding on successors-in-interest regardless of whether such provisions are included in the relevant transfer documents.

Such statements should be addressed to: hwsec@nist.gov

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Purpose and Scope

The purpose of this publication is to describe an approach for safeguarding application container deployments in multi-tenant cloud environments. This publication first explains selected security challenges involving Infrastructure as a Service (IaaS) cloud computing technologies and geolocation in the form of resource asset tags. It then describes a proof-of-concept implementation—a prototype—that was designed to address those challenges. The publication provides sufficient details about the prototype implementation so that organizations can reproduce it if desired. The publication is intended to be a blueprint or template that can be used by the general security community to validate and implement the described implementation.

It is important to note that the prototype implementation presented in this publication is only one possible way to solve the security challenges. It is not intended to preclude the use of other products, services, techniques, etc. that can also solve the problem adequately, nor is it intended to preclude the use of any cloud products or services not specifically mentioned in this publication.

This publication builds upon the terminology and concepts described in the NIST white paper draft, *Hardware-Enabled Security for Server Platforms: Enabling a Layered Approach to Platform Security for Cloud and Edge Computing Use Cases* [1]. Reading that white paper is a prerequisite for reading this publication because it explains the concepts and defines key terminology used in this publication.

1.2 Document Structure

This document is organized into the following sections and appendices:

- Section 2 defines the problem (use case) to be solved.
- Sections 3, 4, and 5 describe the three stages of the prototype implementation:
  - Stage 0: Platform attestation and measured worker node launch
  - Stage 1: Trusted workload placement
  - Stage 2: Asset tagging and trusted location
- The References sections provides references for the document.
- Appendix A provides an overview of the high-level hardware architecture of the prototype implementation, as well as details on how Intel platforms implement hardware modules and enhanced hardware-based security functions.
- Appendix B contains supplementary information provided by AMI describing all the required components and steps required to set up the prototype implementation.
- Appendix C contains supplementary information describing all the required components and steps required to set up the prototype implementation for Kubernetes.
- Appendix D lists the major controls from NIST Special Publication (SP) 800-53 Revision 5, *Security and Privacy Controls for Information Systems and Organizations* that affect container platform security prototype implementation.
- Appendix E maps the major security features from the prototype implementation to the corresponding subcategories from the Cybersecurity Framework.
- Appendix F lists and defines acronyms and other abbreviations used in the document.

2 Prototype Implementation

This section defines the prototype implementation. Section 2.1 explains the basics of the objective. Section 2.2 provides more details, outlining all of the intermediate goals that must be met in order to achieve the desired prototype implementation. These requirements are grouped into three stages of the use case, each of which is examined more closely in Sections 2.2.1 through 2.2.3, respectively.

2.1 Objective

Shared cloud computing technologies are designed to be highly agile and flexible, transparently using whatever resources are available to process container deployments for their customers [2]. However, there are security and privacy concerns with allowing unrestricted container deployment orchestration. Whenever multiple container deployments are present on a single cloud server, there is a need to segregate those deployments from each other so that they do not interfere with each other, gain access to each other's sensitive data, or otherwise compromise the security or privacy of the containers. Imagine two rival companies with container deployments on the same server; each company would want to ensure that the server can be trusted to protect their information from the other company. Similarly, a single organization might have multiple container deployments that need to be kept separate because of differing security requirements and needs for each app.

Another concern with shared cloud computing is that workloads could move from cloud servers located in one country to servers located in another country. Each country has its own laws for data security, privacy, and other aspects of information technology (IT). Because the requirements of these laws may conflict with an organization's policies or mandates (e.g., laws, regulations), an organization may decide that it needs to restrict which cloud servers it uses based on their location. A common desire is to only use cloud servers physically located within the same country as the organization, or physically located in the same country as the origin of the information. Determining the approximate physical location of an object, such as a cloud computing server, is generally known as *geolocation*. Geolocation can be accomplished in many ways, with varying degrees of accuracy, but traditional geolocation methods are not secured and are enforced through management and operational controls that cannot be automated or scaled. Therefore, traditional geolocation methods cannot be trusted to meet cloud security needs.

The motivation behind this use case is to improve the security and accelerate the adoption of cloud computing technologies by establishing an automated, hardware root-of-trust method for enforcing and monitoring platform integrity and geolocation restrictions for cloud servers. A *hardware root-of-trust* is an inherently trusted combination of hardware and firmware responsible for measuring the integrity of the platform and geolocation information in the form of an asset tag. The measurements taken by the hardware root-of-trust are stored in tamper-resistant hardware, and are transmitted using cryptographic keys unique to that tamper-resistant hardware. This information is accessed by management and security tools using cryptographic protocols to assert the integrity of the platform and confirm the location of the host.

This use case builds on earlier work documented in NIST IR 7904, *Trusted Geolocation in the Cloud: Proof of Concept Implementation* [3].

2.2 Goals

Using trusted compute pools (described in Section 3) is a leading approach for aggregating trusted systems and segregating them from untrusted resources, which results in the separation of higher-value, more sensitive workloads from commodity application and data workloads. The principles of operation are to:

1. Create a part of the cloud to meet the specific and varying security requirements of users.
2. Control access to that portion of the cloud so that the right applications (workloads) get deployed there.
3. Enable audits of that portion of the cloud so that users can verify compliance.

These trusted compute pools allow IT to gain the benefits of the dynamic cloud environment while still enforcing higher levels of protections for their more critical workloads.

The ultimate goal is to be able to use “trust” as a logical boundary for deploying cloud workloads on server platforms within a cloud. This goal is dependent on smaller prerequisite goals described as stages, which can be thought of as requirements that the solution must meet. Because of the number of prerequisites, they have been grouped into three stages:

0. Platform attestation and measured worker node launch. This ensures that the integrity of the cloud server platform is measured and available for the subsequent stages.
1. Trusted workload placement**.** This stage allows container deployments to be orchestrated to launch only on trusted server platforms within a cloud.
2. Asset tagging and trusted location**.** This stage allows container deployments to be launched only on trusted server platforms within a cloud, taking into consideration qualitative asset tag restrictions (for example, location information).

The prerequisite goals for each stage, along with more general information on each stage, are explained below.

2.2.1 Stage 0: Platform attestation and measured worker node launch

A fundamental component of a solution is having some assurance that the platform the container deployment is running on can be trusted. If the platform is not trustworthy, then not only is it putting the tenant’s application and data at greater risk of compromise, but also there is no assurance that the claimed asset tag of the cloud server is accurate. Having basic assurance of trustworthiness is the initial stage in the solution.

Stage 0 includes the following prerequisite goals:

1. **Configure a cloud server platform as being trusted.** The “cloud server platform” includes the hardware configuration (e.g., BIOS integrity), operating system (OS) configuration (boot loader and OS kernel configuration and integrity), and the integrity of the container runtime. This also includes the varying hardware security technologies enabled on the server. These chain of trust (CoT) technologies provide platform integrity verification. Additional technologies and details can be found in the aforementioned NIST white paper draft [1] and are discussed in Section 3.2.
2. **Before each container worker node launch, verify (measure) the trustworthiness of the cloud server platform.** The items configured in goal 1 (BIOS, OS, container runtime) need to have their configurations verified before launching the container runtime to ensure that the assumed level of trust is still in place.
3. **During container runtime execution, periodically audit the trustworthiness of the cloud server platform.** This periodic audit is essentially the same check as that performed as goal 2, except that it is performed frequently while the container runtime is executing. Ideally this checking would be part of continuous monitoring.

Achieving all of these goals will not prevent attacks from succeeding, but will cause unauthorized changes to the cloud platform to be detected more rapidly than they otherwise would have been. This prevents new container deployments with trust requirements from being launched on the compromised platform.

For more information on the technical topics being addressed by these goals, see the following NIST publications:

- NIST SP 800-128, *Guide for Security-Focused Configuration Management of Information Systems*
  https://doi.org/10.6028/NIST.SP.800-128
- NIST SP 800-137, *Information Security Continuous Monitoring for Federal Information Systems and Organizations*
  https://doi.org/10.6028/NIST.SP.800-137
- NIST SP 800-144, *Guidelines on Security and Privacy in Public Cloud Computing*
  https://doi.org/10.6028/NIST.SP.800-144
- NIST SP 800-147B, *BIOS Protection Guidelines for Servers*
  https://doi.org/10.6028/NIST.SP.800-147B
- Draft NIST SP 800-155, *BIOS Integrity Measurement Guidelines*
  https://csrc.nist.gov/publications/detail/sp/800-155/draft
- NIST SP 800-190, *Application Container Security Guide*
  https://doi.org/10.6028/NIST.SP.800-190

2.2.2 Stage 1: Trusted workload placement

Once stage 0 has been successfully completed, the next objective is to be able to orchestrate the placement of workloads to launch only on trusted platforms. Workload placement is a key attribute of cloud computing, improving scalability and reliability. The purpose of this stage is to ensure that any server that a workload is launched on will meet the required level of security assurance based on the workload security policy.

Stage 1 includes the following prerequisite goal:

1. **Deploy workloads only to cloud servers with trusted platforms.** This basically means that you perform stage 0, goal 3 (auditing platform trustworthiness) and only deploy a workload to the cloud server if the audit demonstrates that the platform is trustworthy.

Achieving this goal ensures that the workloads are deployed to trusted platforms, thus reducing the chance of workload compromise.

For more information on the technical topics being addressed by these goals, see the following NIST publications:

- Draft NIST Cybersecurity White Paper, *Hardware-Enabled Security for Server Platforms: Enabling a Layered Approach to Platform Security for Cloud and Edge Computing Use Cases* https://doi.org/10.6028/NIST.CSWP.04282020-draft
- NIST SP 800-137, *Information Security Continuous Monitoring for Federal Information Systems and Organizations* https://doi.org/10.6028/NIST.SP.800-137
- NIST SP 800-144, *Guidelines on Security and Privacy in Public Cloud Computing* https://doi.org/10.6028/NIST.SP.800-144
- NIST SP 800-147B, *BIOS Protection Guidelines for Servers* https://doi.org/10.6028/NIST.SP.800-147B
- Draft NIST SP 800-155, *BIOS Integrity Measurement Guidelines* https://csrc.nist.gov/publications/detail/sp/800-155/draft

2.2.3 Stage 2: Asset tagging and trusted location

The next stage builds upon stage 1 by adding the ability to continuously monitor and enforce asset tag restrictions.

Stage 2 includes the following prerequisite goals:

1. **Have trusted asset tag information for each trusted platform instance.** This information would be stored within the cloud server's cryptographic module (as a cryptographic hash within the hardware cryptographic module) so that it could be verified and audited readily.
2. **Provide configuration management and policy enforcement mechanisms for trusted platforms that include enforcement of asset tag restrictions.** This goal builds upon stage 1, goal 1 (deploy workloads only to cloud servers with trusted platforms); it enhances stage 2, goal 1 by adding an asset tag check to the server to launch the workload on.
3. **During workload orchestration, periodically audit the asset tag of the cloud server platform against asset tag policy restrictions.** This goal is built upon stage 0, goal 3 (auditing platform trustworthiness), but it is specifically auditing the asset tag information against the policies for asset tag to ensure that the server's asset tagging does not violate the policies.

Achieving these goals ensures that the workloads are not launched on a server in an unsuitable boundary location. This avoids issues caused by clouds spanning different physical locations (e.g., regulations, sensitivity levels, countries or states with different data security and privacy laws).

For more information on the technical topics being addressed by these goals, see the following NIST publications:

- NIST SP 800-128, *Guide for Security-Focused Configuration Management of Information Systems* https://doi.org/10.6028/NIST.SP.800-128
- NIST SP 800-137, *Information Security Continuous Monitoring for Federal Information Systems and Organizations* https://doi.org/10.6028/NIST.SP.800-137
- NIST SP 800-147B, *BIOS Protection Guidelines for Servers* https://doi.org/10.6028/NIST.SP.800-147B

- Draft NIST SP 800-155, *BIOS Integrity Measurement Guidelines* https://csrc.nist.gov/publications/detail/sp/800-155/draft

3 Prototyping Stage 0

This section describes stage 0 of the prototype implementation (platform attestation and measured worker node launch).

3.1 Solution Overview

This stage of the use case enables the creation of what are called *trusted compute pools*. Also known as *trusted pools*, they are physical or logical groupings of computing hardware in a data center that are tagged with specific and varying security policies, and the access and execution of apps and workloads are monitored, controlled, audited, etc. In this phase of the solution, an attested launch of the platform is deemed as a trusted node and is added to the trusted pool.

Figure 1 depicts the concept of trusted pools. The resources tagged green indicate trusted ones. Critical policies can be defined such that security-sensitive cloud services can only be launched on these trusted resources.

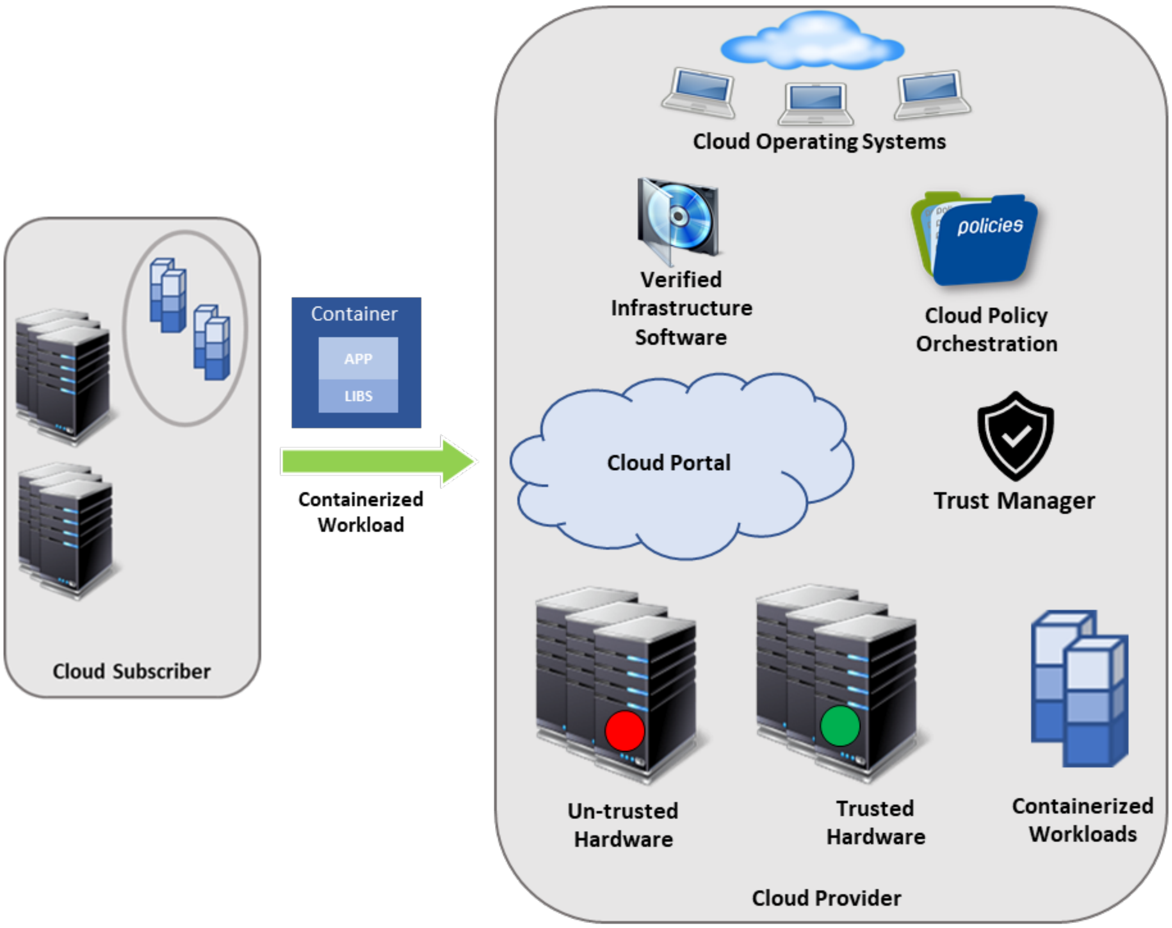

Figure 1: Concept of Trusted Compute Pools

In order to have a trusted launch of the platform, the two key questions that should be answered are:

1. How would the entity needing this information know if a specific platform has the necessary enhanced hardware-based security features enabled and if a specific platform has a defined/compliant OS/platform firmware and container runtime running on it?
2. Why should the entity requesting this information, which in a cloud environment would be a scheduler/orchestrator trying to schedule a workload on a set of available nodes/servers, believe the response from the platform?

Attestation provides the definitive answers to these questions. *Attestation* is the process of providing a digital signature of a set of measurements securely stored in hardware, then having the requestor validate the signature and the set of measurements. Attestation requires roots of trust. The platform has to have a Root-of-Trust for Measurement (RTM) that is implicitly trusted to provide an accurate measurement, and enhanced hardware-based security features provide the RTM. The platform also has to have a Root-of-Trust for Reporting (RTR) and a Root-of-Trust for Storage (RTS), and the same enhanced hardware-based security features provide these.

The entity that challenged the platform for this information now can make a determination about the trust of the launched platform by comparing the provided set of measurements with "known good/golden" measurements. Managing the "known good" for different platforms and operating systems, and various BIOS software, and ensuring they are protected from tampering and spoofing is a critical IT operations challenge. This capability can be internal to a service provider, or it could be delivered as a service by a trusted third party for service providers and enterprises to use.

3.2 Solution Architecture

Figure 2 provides a layered view of the solution system architecture. The indicated servers in the resource pool include a hardware module for storing sensitive keys and measurements. All the servers are configured by the cloud management server.

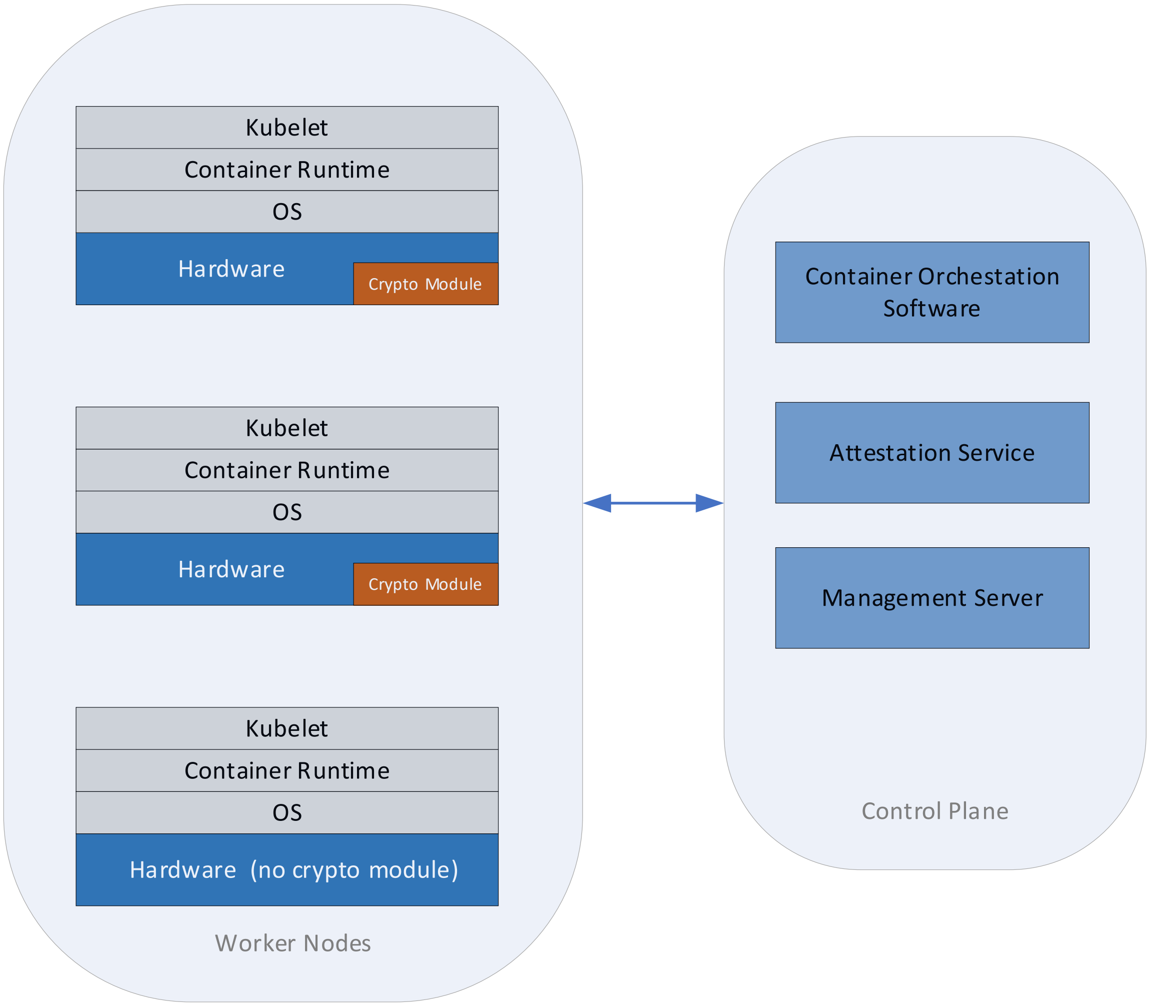

Figure 2: Stage 0 Solution System Architecture

The initial step in instantiating the architecture requires provisioning the server for enhanced hardware-based security features. This requires either physical or remote access to the server to access the BIOS, enable a set of configuration options to use the hardware module (including taking ownership of the module), and activate the enhanced hardware-based security features. This process is highly BIOS and original equipment manufacturer (OEM) dependent. This step is mandatory for a measured launch of the platform.

The management console provides remote monitoring and management of all servers in this solution architecture. It allows remote configuration of BIOS that are required for a server to be measured and secured. It periodically checks the measurements of all monitored servers and compares them against golden measurements that were taken in pristine condition. When any such measurements do not match, indicating a platform security compromise, it notifies the administrator through email and/or text message. The administrator can then use the management console to take remediation actions, which could include powering down the server or reconfiguring or updating the firmware of the server.

The platform undergoes a measured launch, and the BIOS and OS components are measured (cryptographically) and placed into the server hardware security module. These measurement values are accessible through the cloud management server via the application programming interface (API). When the hosts are initially configured with the cloud management server, the relevant measurement values are cached in the cloud management database.

In addition to the measured launch, this solution architecture also provides provisions to assign a secure asset tag to each of the servers during the provisioning process. The asset tag is provisioned to a non-volatile index in the hardware module via an out-of-band mechanism, and on a platform launch the contents of the index are inserted/extended into the hardware module. Enhanced hardware-based security features provide the interface and attestation to the asset tag information, including the asset tag lookup and user-readable/presentable string/description.

4 Prototyping Stage 1

This section discusses stage 1 of the prototype implementation (trusted workload placement), which is based on the stage 0 work and adds components that orchestrate the placement of workloads to launch on trusted platforms.

4.1 Solution Overview

Figure 3 shows the operation of the stage 1 solution. It assumes that Server A and Server B are two servers within the same cloud.

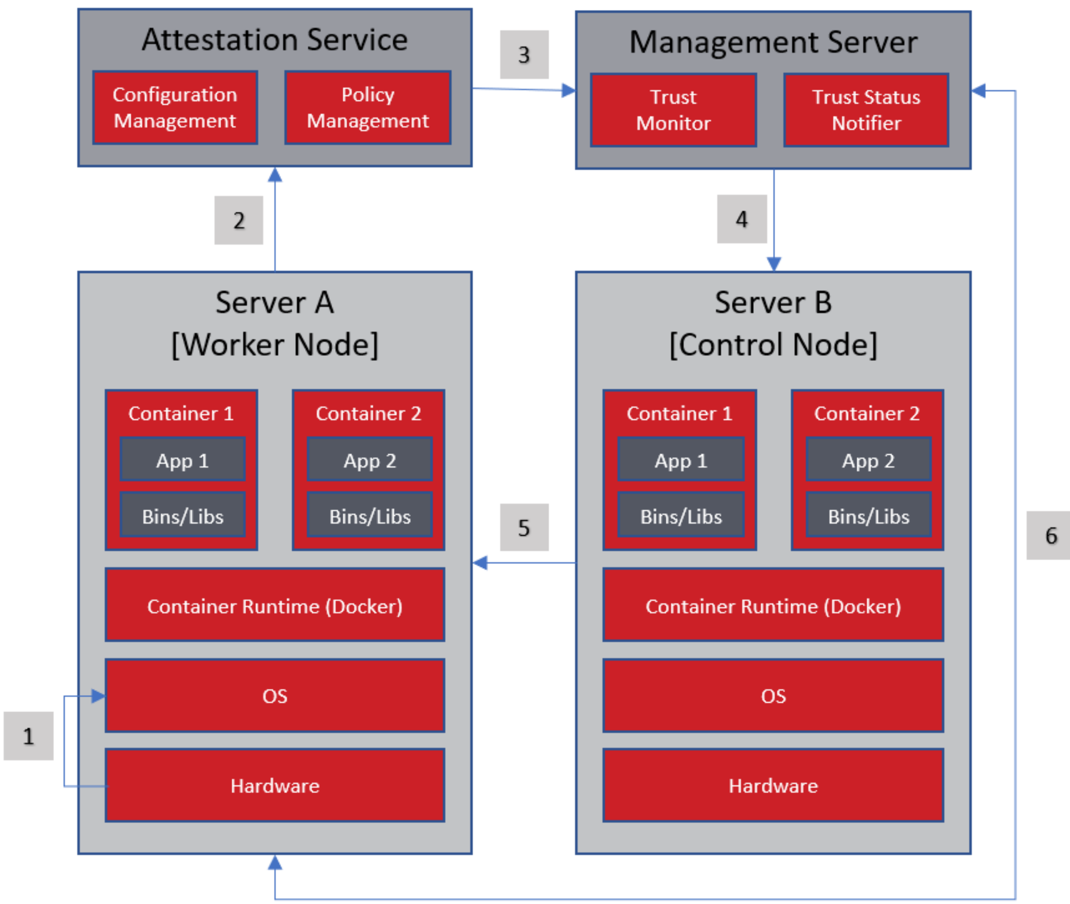

Figure 3: Stage 1 Solution Overview

There are six generic steps performed in the operation of the stage 1 prototype, as outlined below and reflected by the numbers in Figure 3:

1. Server A performs a measured launch, with the enhanced hardware-based security features populating the measurements in the hardware module.
2. Server A sends a quote to the Trust Authority. The quote includes signed hashes of various platform firmware and OS components.
3. The Trust Authority verifies the signature and hash values, and sends the attestation of the platform's integrity state to the management server.

4. The management server enforces workload policy requirements on Server B based on user requirements.
5. Server B launches workloads that require trusted infrastructure only on server platforms that have been attested to be trusted.
6. Each server platform gets audited periodically based on its measurement values.

4.2 Solution Architecture

The stage 1 architecture is identical to the stage 0 architecture (see Figure 2), with additional measurement occurring related to the orchestration of workload placement among trusted hosts.

5 Prototyping Stage 2

This section discusses stage 2 of the prototype implementation (trust-based and asset tag-based secure workload placement), which is based on the stage 1 work and adds components that take into account asset tag restrictions.

5.1 Solution Overview

Stage 2 adds the monitoring of measurements in a governance, risk, and compliance dashboard. One chart that might appear in such a dashboard could reflect the relative size of the pools of trusted and untrusted cloud servers. This could be displayed by percentage and/or count.

Table 1 is a drill-down page from the high-level dashboard view. It provides more details on all the servers within the cloud. In this example, there are three servers. Information listed for each server includes the server's IP address and universally unique identifier (UUID), and the status of the measurements (trusted boot validation and system health validation).

Table 1: Trusted Boot Compliance View

Cloud Host IP	Hardware UUID	Trusted Boot Validation	System Health Validation
<Host 1>	<UUID 1>	Yes/No	Yes/No
<Host 2>	<UUID 2>	Yes/No	Yes/No
<Host 3>	<UUID 3>	Yes/No	Yes/No

Figure 4 shows a drill-down from Table 1 for an individual server. It includes a detailed measurement data for the trusted boot validation, alongside the connection status and asset tag list which may include asset tag value. It also shows when the server was measured and when the validity of this measurement expires. Measuring each server's characteristics frequently (such as every five minutes) helps to achieve a continuous monitoring solution for the servers.

General Information			
Host Info:	<IP Address or Host Name>	**UUID:**	<Unique ID>
Trust Report Created On:	<Time Stamp>	**Trust Report Expires On:**	<Time Stamp>
Asset Tag Status:	<Deployed/Not Deployed>	**Asset Tag List:**	<Name-1/Value-1> <Name-N/Value-N>
Flavor Group Name:	<Name>	**Connection Status:**	<Connected / Not connected>

Trust Information			
Overall System Trust:	<Trusted/Untrusted>		
Software Trust:	<Trusted/Untrusted>	**Platform Trust:**	<Trusted/Untrusted>
Asset Tag Trust:	<Trusted/Untrusted>	**Host Unique Trust:**	<Trusted/Untrusted>

Figure 4: Single Server Overview

References

References for this publication are listed below.

[1] Bartock MJ, Souppaya MP, Savino R, Knoll T, Shetty U, Cherfaoui M, Yeluri R, Scarfone KA (2020) Hardware-Enabled Security for Server Platforms: Enabling a Layered Approach to Platform Security for Cloud and Edge Computing Use Cases. (National Institute of Standards and Technology, Gaithersburg, MD), Draft NIST Cybersecurity White Paper. https://doi.org/10.6028/NIST.CSWP.04282020-draft

[2] Souppaya MP, Scarfone KA, Morello J (2017) Application Container Security Guide. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-190. https://doi.org/10.6028/NIST.SP.800-190

[3] Bartock MJ, Souppaya MP, Yeluri R, Shetty U, Greene J, Orrin S, Prafullchandra H, McLeese J, Scarfone KA (2015) Trusted Geolocation in the Cloud: Proof of Concept Implementation. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Interagency or Internal Report (IR) 7904. https://doi.org/10.6028/NIST.IR.7904

[4] Greene J (2012) Intel Trusted Execution Technology: Hardware-based Technology for Enhancing Server Platform Security. (Intel Corporation). Available at http://www.intel.com/content/www/us/en/architecture-and-technology/trusted-execution-technology/trusted-execution-technology-security-paper.html

[5] AMI (2020) AMI TruE Trusted Environment. Available at https://ami.com/en/products/security-services-and-solutions/ami-true-trusted-environment/

[6] Joint Task Force (2020) Security and Privacy Controls for Information Systems and Organizations. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-53, Rev. 5. https://doi.org/10.6028/NIST.SP.800-53r5

[7] National Institute of Standards and Technology (2018) Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1. (National Institute of Standards and Technology, Gaithersburg, MD). https://doi.org/10.6028/NIST.CSWP.04162018

Appendix A—Hardware Architecture and Prerequisites

This appendix provides an overview of the high-level hardware architecture of the prototype implementation, as well as details on how Intel platforms implement hardware modules and enhanced hardware-based security functions [4].

A.1 High-Level Implementation Architecture

The prototype implementation network is a flat management network for the AMI components and Intel compute servers with a generic laptop to use as a workstation. The Intel servers utilize an overlay network for communication between the containers that run on top of them. Each Intel server has a socketed BIOS and Baseboard Management Controller (BMC) slot so that a BIOS and BMC chip could be flashed with AMI firmware and installed on the server. There are more technical details regarding the AMI components and Intel server configurations and installation in Appendices B and C. Table 2 displays the hostname and IP configuration of each system, and Figure 5 shows the logical network architecture of the implementation.

Table 2: System Hostnames and IP Configurations

Hostname	Management IP Address	BMC IP Address
AMI-TruE	172.16.100.50	N/A
AMI-HVS	172.16.100.51	N/A
Purley1	172.16.100.61	172.16.100.161
Purley2	172.16.100.62	172.16.100.162
Purley3	172.16.100.63	172.16.100.163
Workstation	172.16.100.21	N/A

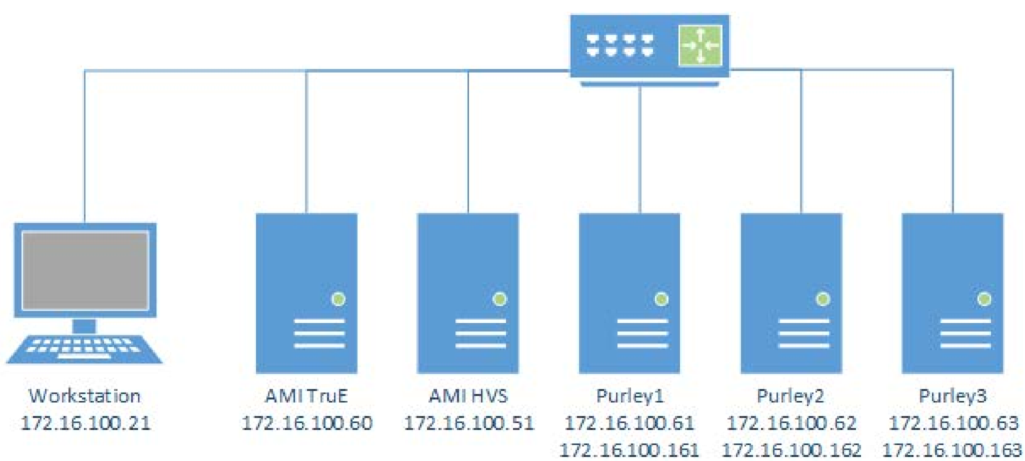

Figure 5: Logical Network Architecture of the Prototype Implementation

A.2 Intel Trusted Execution Technology (Intel TXT) & Trusted Platform Module (TPM)

Hardware-based root-of-trust, when coupled with an enabled BIOS, OS, and components, constitutes the foundation for a more secure computing platform. This secure platform ensures BIOS and OS integrity at boot from rootkits and other low-level attacks. It establishes the trustworthiness of the server and host platforms.

There are three roots of trust in a trusted platform: RTM, RTR, and RTS. They are the foundational elements of a single platform. These are the system elements that must be trusted because misbehavior in these normally would not be detectable in the higher layers. In an Intel TXT-enabled platform the RTM is the Intel microcode: the Core-RTM (CRTM). An RTM is the first component to send integrity-relevant information (measurements) to the RTS. Trust in this component is the basis for trust in all the other measurements. RTS contains the component identities (measurements) and other sensitive information. A trusted platform module (TPM) provides the RTS and RTR capabilities in a trusted computing platform.

Intel® Trusted Execution Technology (Intel® TXT) is the RTM, and it is a mechanism to enable visibility, trust, and control in the cloud. Intel TXT is a set of enhanced hardware components designed to protect sensitive information from software-based attacks. Intel TXT features include capabilities in the microprocessor, chipset, I/O subsystems, and other platform components. When coupled with an enabled OS and enabled applications, these capabilities provide confidentiality and integrity of data in the face of increasingly hostile environments.

Intel TXT incorporates a number of secure processing innovations, including:

- **Protected execution:** Lets applications run in isolated environments so that no unauthorized software on the platform can observe or tamper with the operational information. Each of these isolated environments executes with the use of dedicated resources managed by the platform.
- **Sealed storage:** Provides the ability to encrypt and store keys, data, and other sensitive information within the hardware. This can only be decrypted by the same environment that encrypted it.
- **Attestation:** Enables a system to provide assurance that the protected environment has been correctly invoked and to take a measurement of the software running in the protected space. The information exchanged during this process is known as the attestation identity key credential and is used to establish mutual trust between parties.
- **Protected launch:** Provides the controlled launch and registration of critical system software components in a protected execution environment.

Intel® Xeon® Platinum Scalable processor series and the previous generation Xeon Processor E3, Xeon Processor E5, and Xeon Processor E7 series processors support Intel TXT.

Intel TXT works through the creation of a measured launch environment (MLE) enabling an accurate comparison of all the critical elements of the launch environment against a known good source. Intel TXT creates a cryptographically unique identifier for each approved launch-enabled component and then provides a hardware-based enforcement mechanism to block the launch of any code that does not match or, alternately, indicate when an expected trusted launch has not happened through a process of secure remote attestation. In the latter case, when an attestation indicates that one or more measured components in the MLE do not match expectations, orchestration of workloads can be prevented on the suspect platform, even though the platform itself still launches. This hardware-based solution provides the foundation on which IT administrators can build trusted platform solutions to protect against aggressive software-based attacks and to better control their virtualized or cloud environments. For additional information on TXT and other RTM technologies, see the NIST white paper draft, *Hardware-Enabled Security for Server Platforms: Enabling a Layered Approach to Platform Security for Cloud and Edge Computing Use Cases* [1].

A.3 Attestation

There are two main considerations for use cases to be instantiated and delivered in a cloud:

- How would the entity needing this information know if a specific platform has Intel TXT enabled or if a specific server has a defined or compliant BIOS or OS running on it (i.e., can it be trusted)?
- Why should the entity requesting this information (which, in a cloud environment, could be a resource scheduler or orchestrator trying to schedule a service on a set of available nodes or servers) trust the response from the platform?

An attestation authority provides the definitive answers to these questions. Attestation provides cryptographic proof of compliance, utilizing the root of trust concept to provide actionable security controls by making the information from various roots of trust visible and usable by other entities. Figure 6 illustrates the attestation protocol providing the means for conveying measurements to the challenger. The endpoint attesting device must have a means of measuring the BIOS firmware, low-level device drivers, and OS and other measured components, and forwarding those measurements to the attestation authority. The attesting device must do this while protecting the integrity, authenticity, nonrepudiation, and in some cases, confidentiality of those measurements.

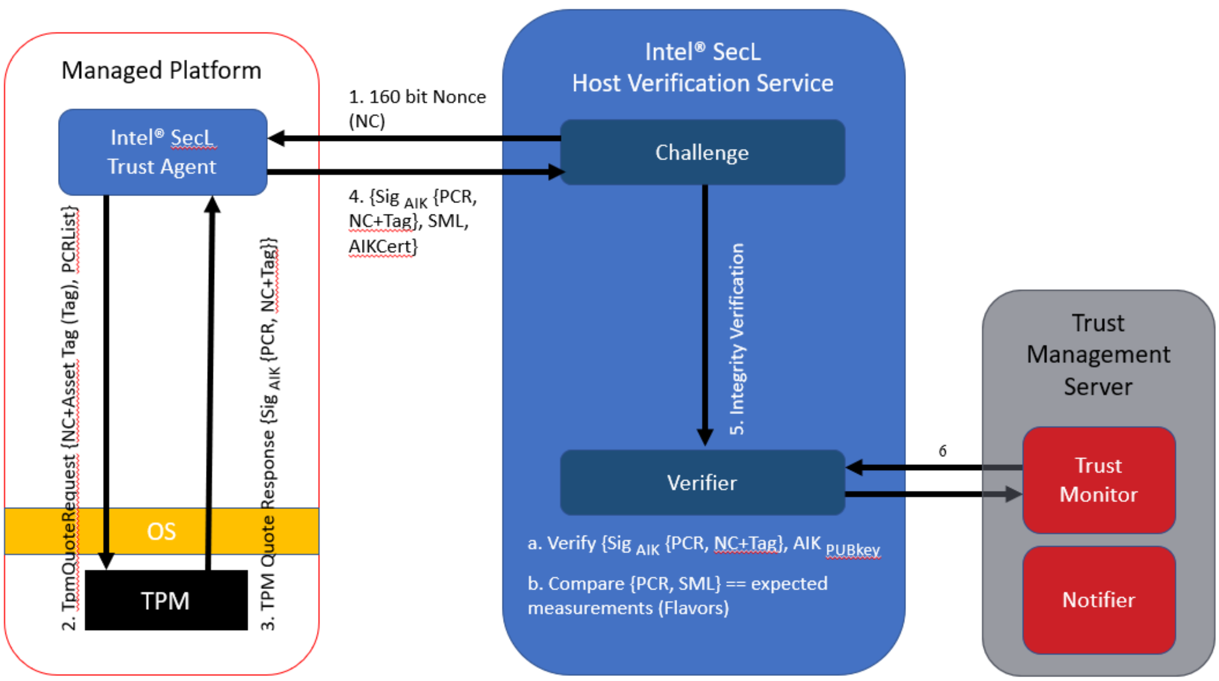

Figure 6: Remote Attestation Protocol

Here are the steps shown in Figure 6 for the remote attestation protocol:

1. The challenger, at the request of a requester, creates a non-predictable nonce (NC) and sends it to the attestation agent on the attesting node, along with the selected list of Platform Configuration Registers (PCRs).
2. The attestation agent sends that request to the TPM as a TPMQuoteRequest with the nonce and the PCR List.

3. In response to the TPMQuote request, the TPM loads the attestation identity key from protected storage in the TPM by using the storage root key (SRK), and performs a *TPM Quote* command, which is used to sign the selected PCRs and the NC with the private key *AIKpriv*. Additionally, the attesting agent retrieves the stored measurement log (SML).
4. In the *integrity response* step, the attesting agent sends the response consisting of the signed quote, signed NC, and the SML to the challenger. The attesting agent also delivers the Attestation Identity Key (AIK) credential, which consists of the AIKpub that was signed by a privacy certificate authority (CA).
5. For the *integrity verification* step:
    a. The challenger validates if the AIK credential was signed by a trusted Privacy-CA, thus belonging to a genuine TPM. The challenger also verifies whether AIKpub is still valid by checking the certificate revocation list of the trusted issuing party.
    b. The challenger verifies the signature of the quote and checks the freshness of the quote.
    c. Based on the received SML and the PCR values, the challenger processes the SML, compares the individual module hashes that are extended to the PCRs against the "good known or golden values," and recomputes the received PCR values. If the individual values match the golden values and if the computed values match the signed aggregate, the remote node is asserted to be in a trusted state.
6. The verifier informs the trust state of the remote node to the manager. The manager records the trust state in its management database and uses it for any individual or aggregated device status requests. If an administrator subscribed to trust-related events, the manager will also send email notifications when a managed remote node is detected as being untrusted.

This protocol can help mitigate replay attacks, tampering, and masquerading.

Appendix B—Platform Implementation: AMI TruE

This section contains supplementary information provided by AMI Trusted Environment (AMI TruE) describing all the components and steps required to set up the prototype implementation [5].

B.1 Solution Architecture

Figure 7 shows the architecture depicted in Appendix A, but with the specific products used in the AMI TruE platform implementation.

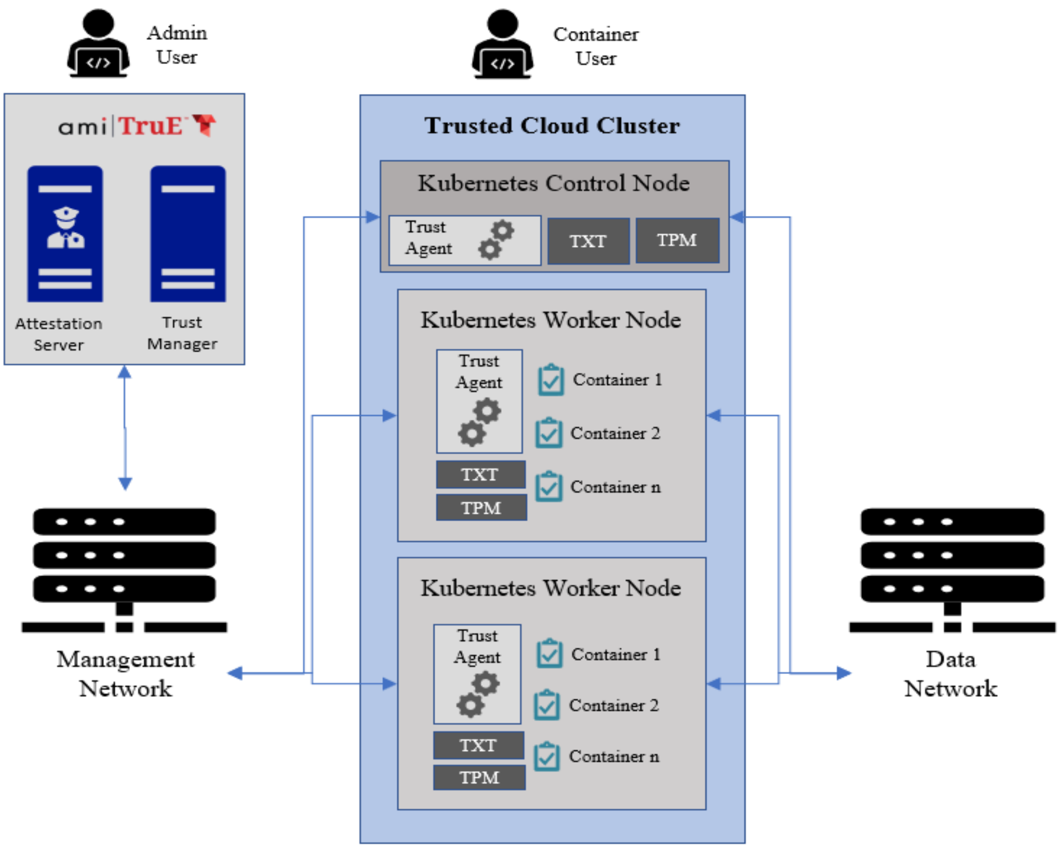

Figure 7: AMI TruE Prototype Implementation

B.2 Hardware Description

The implemented architecture is composed of two Intel Next Units of Computing (NUCs) acting as Management Nodes and one Intel NUC together with two Intel TXT-enabled Server Platforms serving as the Trusted Cloud Cluster. Their hardware is detailed in Table 3.

Table 3: Hardware for Implemented Architecture

Hardware	Processor	Memory	Disk Space	OS
One Intel NUC acting as 'Kubernetes Control Node'	Intel i5-7300U @ 2.60 gigahertz (GHz)	8 gigabytes (GB)	250 GB	Red Hat Enterprise Linux (RHEL) 8.1

Hardware	Processor	Memory	Disk Space	OS
Two Server Platforms (with Intel TXT enabled) acting as 'Kubernetes Worker'	Intel Xeon Platinum 8260L @ 2.40 GHz	96 GB	250 GB	RHEL 8.1
Intel NUC acting as Trust Manager	Intel i5-7300U @ 2.60 GHz	8 GB	250 GB	RHEL 7.6
Intel NUC acting as Attestation Server	Intel i5-7300U @ 2.60 GHz	8 GB	250 GB	RHEL 8.1

B.3 AMI TruE Installation and Configuration

AMI TruE provides datacenter security solutions using Intel security technologies and libraries. With AMI TruE, datacenters can achieve a level of trust and security. The following is a list of high-level features offered by AMI TruE to manage and secure servers. Some of these features are discussed in more detail later in this section.

- Automatic discovery of servers
- Asset inventory information collection
- Server health monitoring
- Trust status monitoring for all discovered servers
- TruE agent provisioning
- Remediation of untrusted servers
- Alert emails for health or trust events
- Remote power control
- Remote console (keyboard, video, mouse [KVM] redirection)
- BIOS/BMC firmware configuration and update
- OS and software provisioning
- Hypertext Markup Language version 5 (HTML5) based web interface
- Representational State Transfer (REST) APIs for automation or integration

AMI TruE has two components, Trust Manager and Attestation Server, so it requires two physical or virtual systems to deploy AMI TruE. Table 4 specifies the minimum requirements for those systems.

Table 4: Minimum System Requirements to Install AMI TruE

System Element	Trust Manager	Attestation Server
Processor	4-core 2.66 GHz central processing unit (CPU)	4-core 2.66 GHz CPU
Memory	8 GB	8 GB
Disk Space	100 GB	100 GB
OS	RHEL 7.6, 64-bit	RHEL 8.1, 64-bit

B.3.1 Installing AMI TruE Trust Manager

To install the Trust Manager onto its system, perform the following steps:

1. Log into the Trust Manager system as a root user.

2. Download and extract the “amitrue_trustmanager_artifacts.zip” file into the “/root” folder.
3. Run the commands below as root user:
    a. Set execution permission for the install script:
    ```
    # chmod +x ./install.sh
    ```
    b. Install AMI TruE Trust Manager by running the following install script:
    ```
    #./install.sh
    ```

B.3.2 Installing AMI TruE Attestation Server

To install the Attestation Server onto its system, perform the following steps:

1. Log into the Attestation Server system as a root user.
2. Download and extract the “amitrue_attestationserver_artifacts.zip” file into the “/root” folder.
3. Edit the “amitrue_security.env” file to configure the following:
    ```
    HOSTNAME
    IP_HOSTNAME_ARRAY
    ```
4. Run the commands below as root user:
    a. Set execution permission for the install script:
    ```
    # chmod +x ./install.sh
    ```
    b. Install AMI TruE Attestation Server by running the following install script:
    ```
    #./install.sh
    ```

B.3.3 Configuring Firewall for AMI True

AMI True uses several network ports for managing and securing platforms. The install script automatically configures the firewall to allow these ports. Ensure that no other software or utility disables any of the ports listed in Table 5 and Table 6.

Table 5: Network Ports for Trust Manager

Port	Purpose	Direction
9090	HTTP port for NGINX	Inbound/outbound
9443	HTTPS port for NGINX	Inbound/outbound
6379	Redis Database	Internal
5432	PostgreSQL Database	Internal
1900	Simple Service Discovery Protocol (SSDP) Discovery Module	Inbound/outbound
25/625	Core Notification Service (Simple Mail Transfer Protocol [SMTP])	Outbound
9089	Core Service Manager	Internal
9075	Core Discovery	Internal
9065	Core Platform Security	Internal
9055	Core API Server	Internal
9080	Redfish Server	Internal
9091	Server Manager – KVM	Inbound/outbound

Table 6: Network Ports for Attestation Server

Port	Purpose	Direction
5432	PostgreSQL Database	Internal
8443	Host Verification Service (HVS)	Inbound/outbound
8444	Authentication and Authorization Service (AAS)	Inbound/outbound
8445	Certificate Management Service (CMS)	Inbound/outbound
5000	Workload Service (WLS)	Inbound/outbound
9443	Key Management Service (KMS)	Inbound/outbound
1443	Trust Agent (TA)	Inbound/outbound
19082	Integration Hub (HUB)	Inbound/outbound
19445	Integration Hub (HUB)	Inbound/outbound

B.3.4 Configuring Device Access Keys

AMI TruE needs credentials in order to securely communicate with discovered and manageable devices. To configure these access keys, follow these steps:

1. Under the “Settings” submenu in the main menu, choose “Authentication Keys.”
2. On the Keys page, use “Add” or “Edit” to add access credentials for different types of resources.

B.3.5 Configuring Discovery Range and Manageability Range

To enable AMI TruE to scout and discover devices in a network, it needs to be configured with IP address ranges. Use the following steps to configure the discovery range:

1. Click on the hamburger menu icon  in the top left corner.
2. Under the “Settings” menu group, click “Discovery Settings.”
3. Select the “Global” tab under the “Discovery Ranges” section.
4. Click the “Add” button on the right side of the page to add a new discovery range.

You may not want to manage all discovered devices. A manageable device range can be configured so that AMI TruE will manage only devices that fall within that range. Use the following steps to configure a manageability range:

1. On the “Discovery Settings” page, under the “Discovery Ranges” section, select the “Manageability” tab.
2. Click the “Add” button to add a manageability range. Figure 8 shows a sample set of ranges.

Range	Range Type	Effective Range	Manageability
10.2.0.0/15	CIDR	10.2.0.0 - 10.3.255.255	rmm
10.2.0.0/15	CIDR	10.2.0.0 - 10.3.255.255	rss
10.2.1.0/15	CIDR	10.2.0.0 - 10.3.255.255	fpx
10.2.1.0/15	CIDR	10.2.0.0 - 10.3.255.255	psme
10.2.1.0/24	CIDR	10.2.1.0 - 10.2.1.255	osm
10.2.3.3	Static	10.2.3.3	redfish
10.2.3.4	Static	10.2.3.4	redfish

Figure 8: Examples of Manageability Ranges

B.4 Trusted Cloud Cluster Installation and Configuration

All three nodes in the Trusted Cloud Cluster need to be configured to be managed and secured by AMI TruE. This includes configuring BIOS settings, installing the TruE agent, and registering those agents with AMI TruE. You can do these operations either remotely using AMI TruE or manually. See Appendix B.4.1 for the remote instructions and Appendix B.4.2 for the manual instructions.

Prerequisites for being secured by AMI TruE

To be attested and be monitored for trust status, managed platforms should have:

- Intel® Xeon® or Intel® Xeon® Scalable Family processor that supports Intel TXT
- TPM (version 1.2 or 2.0) installed and provisioned for use with Intel TXT, according to Trusted Compute Group specifications. If a version 2.0 TPM will be used, the Secure Hash Algorithm 256-bit (SHA256) PCR bank must be enabled.
- TPM and Intel TXT enabled in the BIOS
- TPM ownership cleared before installation

To be remediated or recovered from trust compromises, managed platforms should have:

- BMC with Redfish support
- BIOS with Redfish Host Interface support
- Secure shell (SSH) enabled in the host OS

B.4.1 Provisioning TruE Agent Remotely

To provision the TruE agent remotely using AMI TruE, follow these steps:

1. Log into the web interface of AMI TruE.
2. Under the “Provisions” menu, use the “Images” option to go to the “Images List” page.
3. Click on the “Add Image” option in the top menu. Enter the details about the image as depicted in Figure 9. When finished, click the “Save” button.

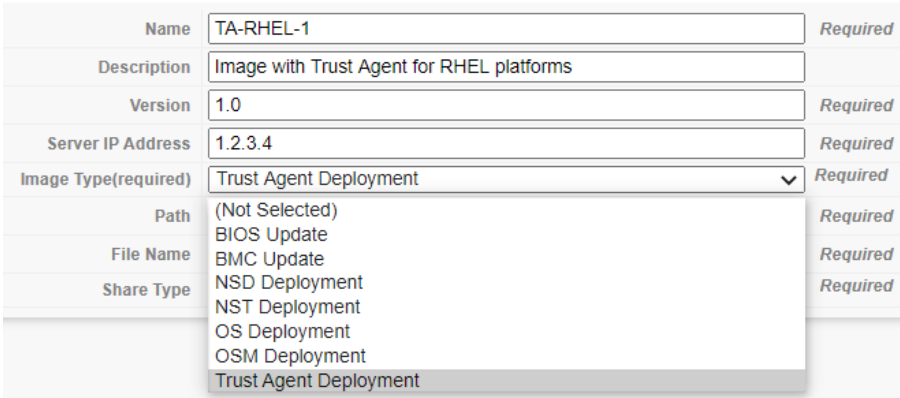

Figure 9: Example of Adding an Image

4. Go to the “Create Job Wizard” page by choosing the “Create Job” option under the “Provisions” menu. Once there, enter a name and description for the job that is getting created, and choose “TA Deployment” as the job type.
5. Click “Next” to move to the “Select Image” page. This page lists all images that were previously uploaded by the administrator. Select the image that needs to be used for deployment.
6. Click “Next” to go to the “Targets” page. Select one or more platforms from the list of target platforms where the trust agent needs to be deployed.
7. Go to the “Schedule” page by clicking “Next”. You can either opt to perform the deployment now or schedule it to be performed at a future date and time. Set this to the desired option for this deployment.
8. Click the “Next” button to go to the “Review Job Settings” page. This page summarizes the information entered for this specific job. Use the “Previous” button to go back to any of the pages in the wizard and make changes if required. When you are satisfied with the settings, click “Start” to initiate the deployment process.
9. To view the status of the job, click the “Jobs” option in the “Provisioning” menu. This page lists all scheduled jobs with their current statuses, as the example in Figure 10 shows. If needed, a scheduled job can be canceled by using the “Cancel Job” button.

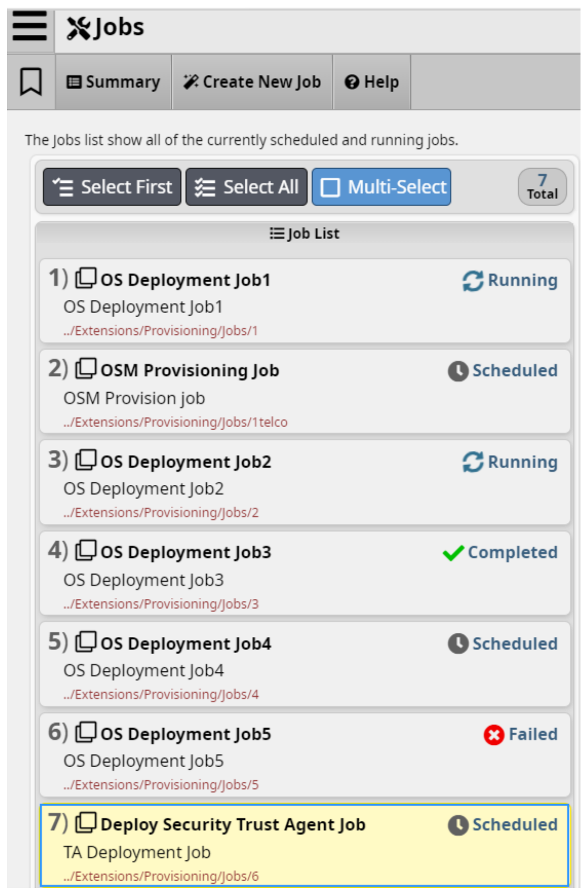

Figure 10: Example of Job List

B.4.2 Provisioning TruE Agent Manually

To provision the TruE agent manually, follow these steps:

1. It is mandatory to register the RedHat subscription manager. Use the following instructions to register:

   a. Create a RedHat account.
   b. Make sure the system has access to the internet.
   c. Start the terminal access by logging in as root user.
   d. Type “subscription-manager-gui” and the Subscription Manager window will pop open.
   e. Click on the “Register” button and provide access credentials to register.
2. Disable the firewall on the target system by running the following commands:
```
$ sudo systemctl stop firewalld
$ sudo systemctl disable firewalld
```
3. Go to the BIOS settings. Enable TPM2 and clear ownership.
4. If you want to use the Unified Extensible Firmware Interface (UEFI) SecureBoot option for trusted boot, enable it in the BIOS settings and skip the next step, otherwise you will use tboot and should <u>not</u> enable SecureBoot.
5. If you want to use tboot, perform these steps:
   a. Run this command to install tboot:
   ```
   # yum install tboot
   ```
   b. Make a backup of the current “grub.cfg” file:
   ```
   # cp /boot/grub2/grub.cfg /boot/grub2/redhat/grub.bak
   ```
   c. Generate a new “grub.cfg” file with the tboot boot option:
   ```
   # grub2-mkconfig -o /boot/grub2/grub.cfg
   ```
   d. Update the default boot option. Ensure that the GRUB_DEFAULT value is set to the tboot option. The tboot boot option can be found by looking in the “/boot/redhat/grub.cfg” file.
6. Reboot the system. Because measurement happens at system boot, a reboot is needed to boot to the tboot boot option and populate measurements in the TPM.
7. To verify a successful trusted boot with tboot, run the `txt-stat` command to show the tboot log. Verify if the output shows “TRUE” for both “TXT measured launch” and “secrets flag set.”
   ```
   ***********************************************************
   TXT measured launch: TRUE
   secrets flag set: TRUE
   ***********************************************************
   ```
8. Install AMI TruE agents by following these steps:
   a. Log in as a root user, and run all the commands below as root user.
   b. Copy the “pkgs/platform-security-agent-artifacts.zip” file into the “/root” folder.
   c. Extract the artifacts into the “/root” folder.
   d. From “platform-security-agent-artifacts,” copy the “install_agent.sh” and “install_agent.env” files into the “/root” folder.
   e. Configure the “install_agent.env” file as follows:

i. HOSTNAME should not be empty. This variable is to set up the hostname for the system (e.g., demo, demo-name-one, demo2).

Note: Do not use an underscore as part of HOSTNAME.

ii. IP_HOSTNAME_ARRAY should not be empty. Users need to provide the IP and HOSTNAME pairs with space separation. This variable will edit the "/etc/hosts" file with given IPs and hostnames.

For example: `IP_HOSTNAME_ARRAY=(10.0.0.0 demo 10.0.0.1 demo-name-one 10.0.0.2 demo2)`

Note: First give the IP and then give the hostname for that IP. Be careful not to mismatch the IPs and hostnames.

iii. Replace all the instances of "127.0.0.1" with the system IP, all instances of "localhost" in URLs by the system IP, and all other instances of "localhost" by the system hostname.

iv. Generate BEARER_TOKEN and CMS_TLS_CERT_SHA384 by using the service installation script with the "populate-users" option on the machine where all the services are running. Then copy and paste the values in the env file.

v. For "NO_PROXY" and "HTTPS_PROXY", provide registry_ip and proxy_url, respectively.

Note: Do not use unnecessary spaces in the env file.

Note: These are basic guidelines. If users have the proper knowledge about these env variables, then they can modify the env variables according to their need.

f. Set execution permission for the install script.

```
# chmod +x install_agent.sh
```

g. Run the install script.

```
# ./install_agent.sh install
```

h. The script will show some dependency messages and ask whether or not to continue installation. Enter "yes" to continue installation.

i. After TA installation, the script will restart the system. After restart, log in as a root user and run the script again.

```
# ./install_agent.sh install
```

j. After the Workload Agent installation, the script will restart the system again to complete the installation.

B.5 Using AMI TruE

Once installed and configured, AMI TruE starts discovering and monitoring the health and trust status of all managed platforms. This section explains the features offered by AMI TruE to monitor and secure the platforms.

B.5.1 Monitoring Trust Status using AMI TruE

Connect to AMI TruE from any standard web browser and use the credentials to log into its web interface. You will see a dashboard with several widgets. The dashboard can also be reached from any other page in the web interface by clicking on the hamburger menu icon in the top left corner and selecting the “Dashboard” menu.

Figure 11 shows a dashboard with a chart reflecting the relative size of the pools of trusted (green), untrusted (red), and unknown (gray) cloud servers. In this example, there are eight servers in the trusted pool and two servers in the untrusted pool. More detailed trust information, including the security state of nodes, flavor-wise trust status, etc., is also depicted in this dashboard.

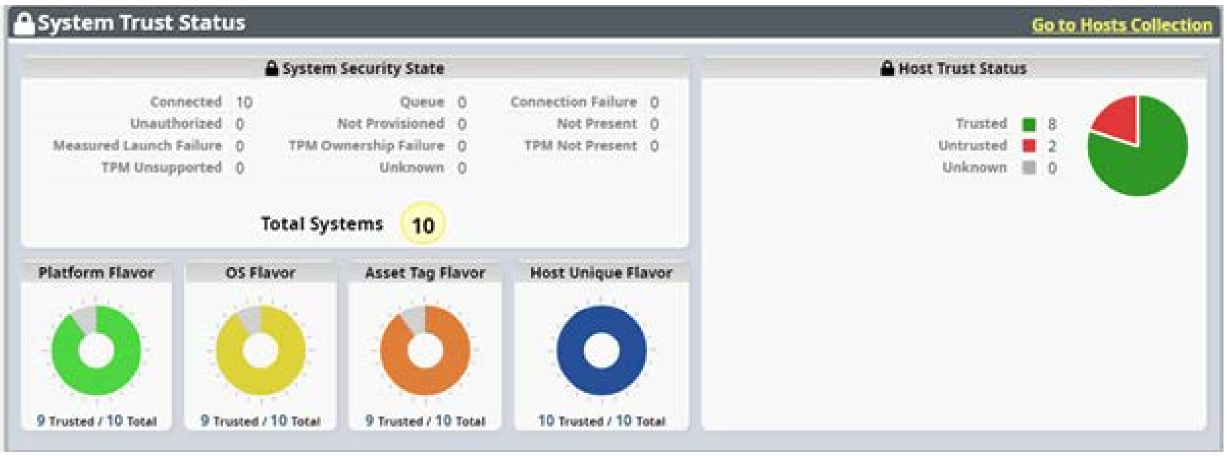

Figure 11: Example of AMI TruE Report Dashboard

Clicking on the “Go to Hosts Collection” link in the top right corner of the dashboard switches to a more detailed management page. Clicking on one of the servers listed in the “Hosts” page displays trust information for that selected server, as Figure 12 depicts. Details provided include the hostname, hardware UUID, flavor details, policy tags, connection URL, and component trust status.

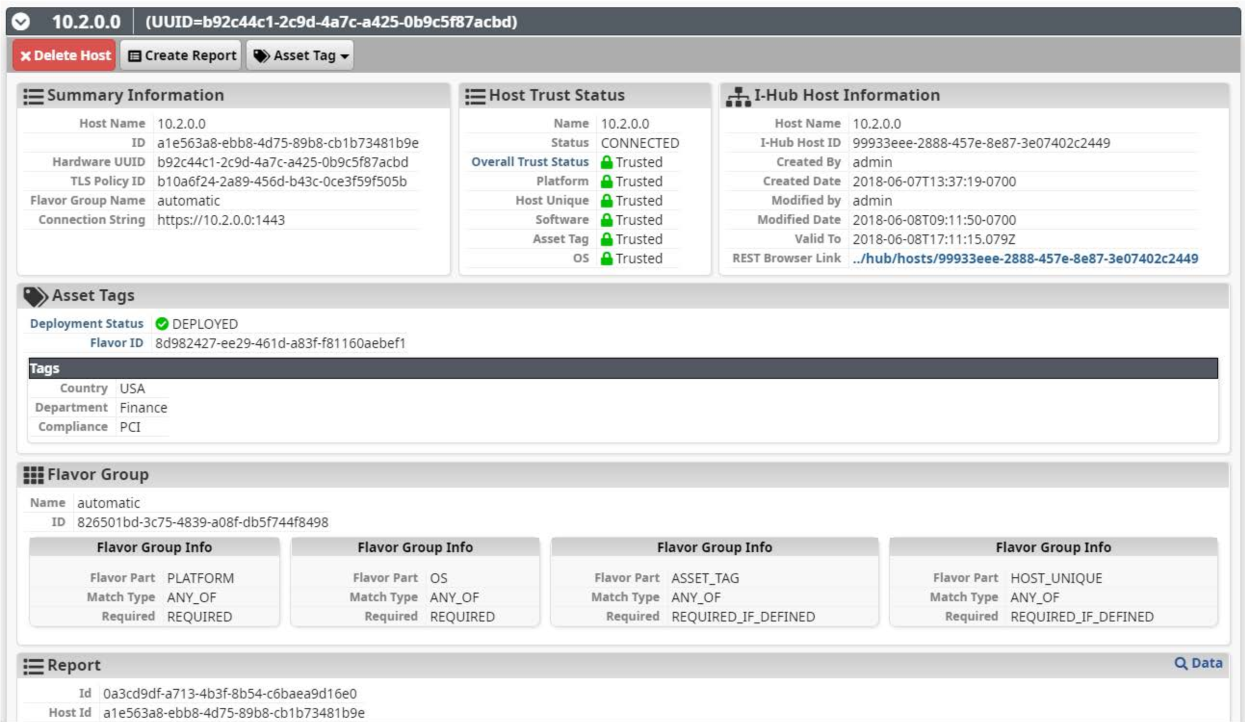

Figure 12: Example of Summary Page for Server Trust Information

B.5.2 Generating Trust Reports

To generate a new report for a platform, select that platform from the host list and click the "Create Report" option. The retrieved report is presented as part of the host information, as shown in Figure 13.

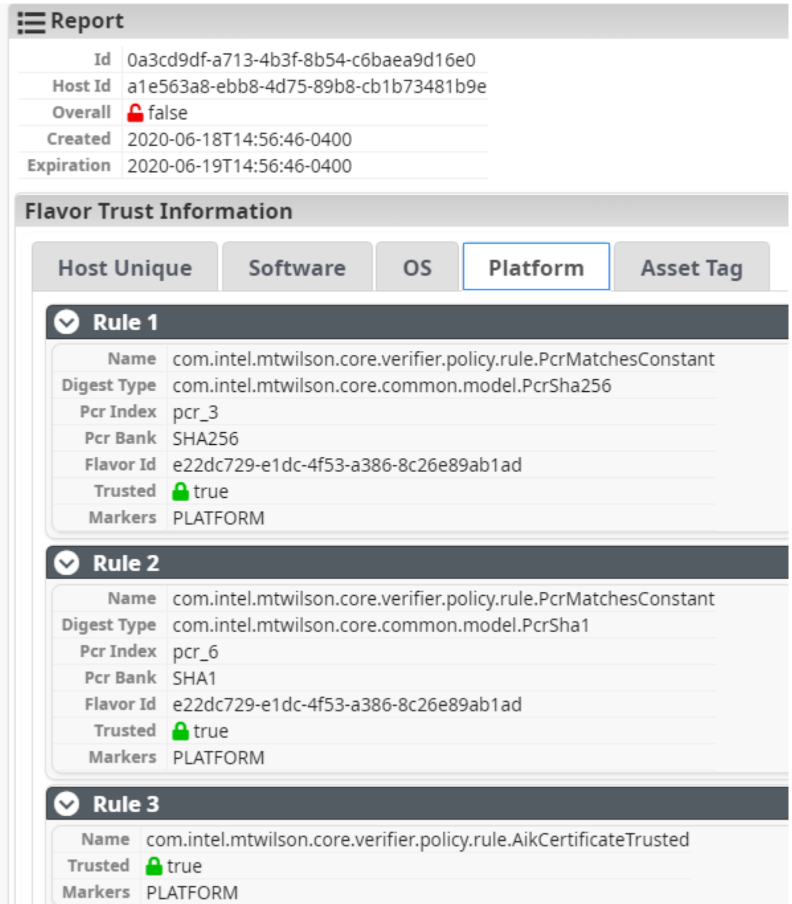

Figure 13: Example of Trust Report

To view the raw JavaScript Object Notation (JSON) data for any analytic needs, click the “Data” option in the top right corner of the “Report” window.

B.5.3 Tagging Platforms Using AMI TruE

Asset tags are used to tag managed platforms with one or more user-defined attributes, such as asset tag, compliance information, or customer type. This enables policy-based workload placement and orchestration.

AMI TruE provides options to create and deploy new asset tags to one or more managed platforms. It also has provisions to revoke any previously created asset tag. To create and deploy an asset tag:

1. Go to the “Hosts” page by opening the main menu and choosing the “Hosts” menu item under the “Security” menu group.
2. Select one or more servers in the host list to choose the platforms for which the asset tag needs to be deployed.
3. Click on the “Asset Tag” menu on the right side of the page, and choose the “Create and Deploy” option as shown in Figure 14.

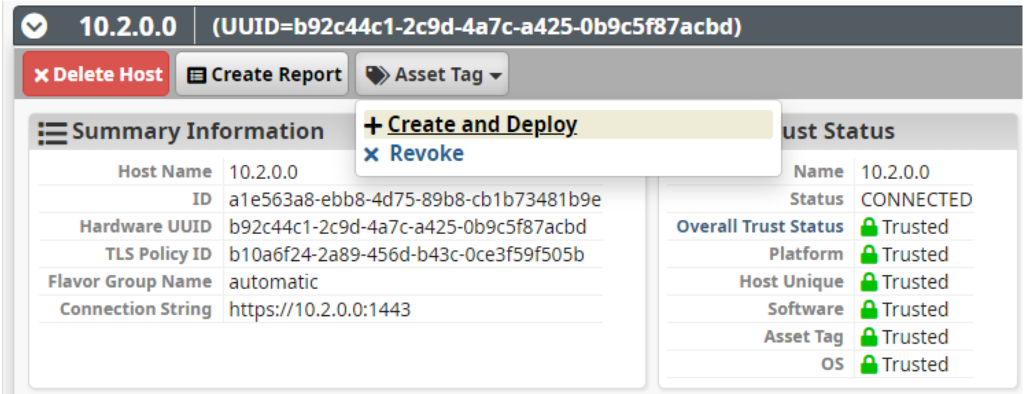

Figure 14: Example of Creating and Deploying an Asset Tag

4. An “Asset Tag Creation” window should open.
   a. To add an asset tag to the list, enter the asset tag name and value in the “Key Name” and “Key Value” fields, then click the “Add” button.
   b. To remove a specific tag from the list, click on the tag’s “X” button in the list of asset tags.
   c. After adding one or more asset tags, click the “Deploy” button to deploy all listed asset tags to the selected platforms.

Alternatively, you can click on the “Asset Tag” menu item in the top menu while being on the “Hosts” page to launch the “Asset Tag” page. That page has options to filter platforms using the “Search” option, and you could then use the filtered list of platforms to deploy an asset tag, as explained above.

B.5.4 Receiving Trust Event Alert Notification

Being notified when a platform turns untrusted allows administrators to quickly take remediation actions. AMI TruE provides two modes of notifications for any security events: email alerts and event log entries.

B.5.4.1 Email Alerts

Administrators can opt to receive email alerts on security events. This starts with adding information about one or more email servers that can be used to email alert information.

1. To add an email server, select the main menu (hamburger icon) icon in the top left corner and choose “Email Notifications” under the “Notifications” menu group.
2. Select “Configure Email Servers” in the resulting menu. This presents a list of configured email servers and provides options to add a new email server configuration, modify an existing email server configuration, or delete a previously configured email server.
3. Click “Add” to add a new entry, or select a row and click “Edit” to edit an existing entry. Enter the details of the email server and choose “Save.”
4. Once at least one email server is configured, the next step is to add email addresses of administrators or support engineers who need to be notified on any trust events. To view the email address book, click the main menu (hamburger icon) icon in the top left corner and select “Email Notifications” under the “Notifications” menu group. Choose “Email Address Book” in the resulting menu. Use the “Add,” “Edit,” and “Delete” options in the “Email Addresses” tab to

manage email addresses. Also, email groups can be used to notify a group of administrators on any specific event. Use the “Email Groups” tab in the “Email Address Book” page to manage email groups.

5. After adding email addresses or groups, the next step is to configure notification subscriptions. From the main menu, select “Notification Subscriptions” under the “Notifications” menu group.
6. Use the “Add New Subscriptions” option on the “Notification Subscriptions” page to configure event subscriptions. From the “Add New Subscription” page, as shown in Figure 15, choose the types of events and resources (event sources) for which notifications need to be sent. You can choose “Security” as the resource to receive any trust-related events.

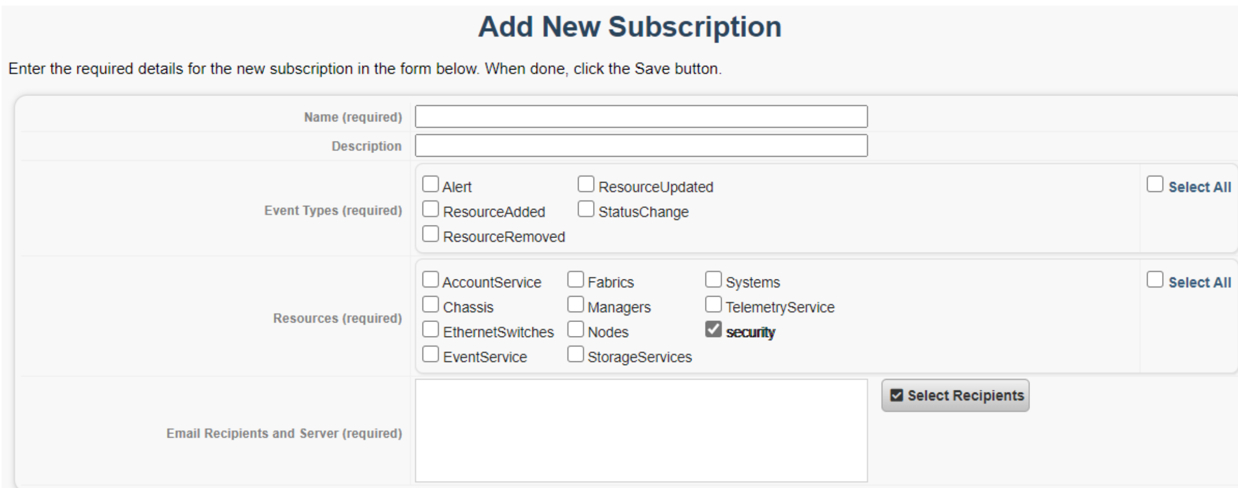

Figure 15: Example of the "Add New Subscription" Page

7. Next, click “Select Recipients” to add one or more email addresses or groups that need to be notified. When done, click the “Save” button to add the subscription.

B.5.4.2 Event Logs

AMI TruE records all platform-related events, including security events, into an event log. Administrators can view those events through a web interface.

1. To view event logs, select the main menu (hamburger icon) icon in the top left corner and choose “Global Event Log” under the “Logs” menu group.
2. Once event logs are viewed and acted upon, administrators can delete the events using the “Clear Log” option on the “Global Event Log” page.

B.5.5 Using AMI TruE for Remediation

Being able to remediate and recover completely is one of the key needs for platform resilience. AMI TruE offers multiple options to recover a compromised platform.

B.5.5.1 Remote Power Control

AMI TruE provides remote power management features to either shut down/power off or reset/restart the platform as part of remediation efforts.

1. Click the main menu icon in the top left corner and select “Server Summary” under the “Server Manager” menu. This page lists all managed servers in the left pane with icons depicting their trust, health, and power state. Select a server that needs to shut down or powered off. On the right pane, click on the “Actions” button and select the “Power Reset” option, as shown in Figure 16.

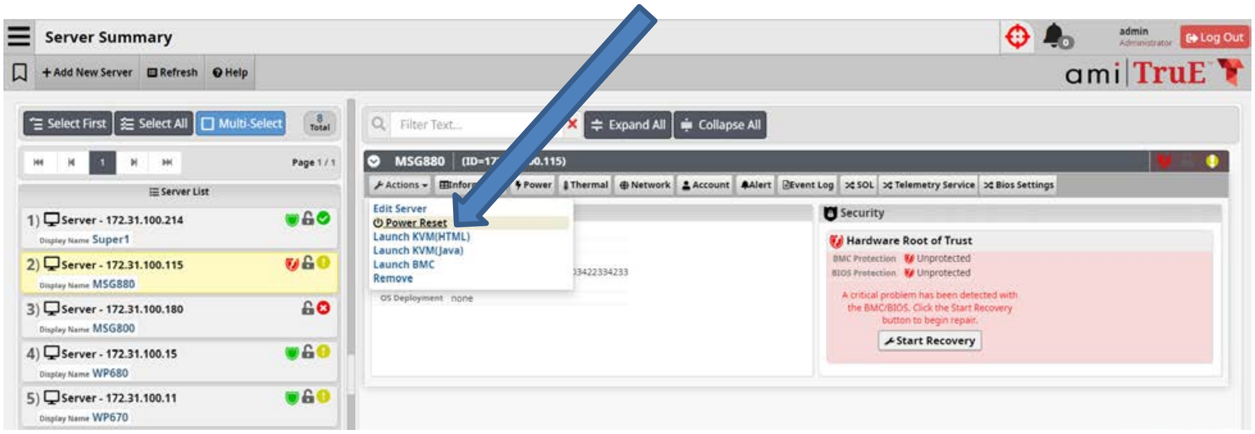

Figure 16: Example of Remote Power Control

2. A popup window with options to choose the type of power operation to be performed is presented. Select the appropriate power control operation and click the “OK” button to proceed.

B.5.5.2 Remote Firmware Update

Using AMI TruE, BIOS/BMC configurations can be made, or the entire BIOS/BMC firmware can be updated if the firmware layer becomes untrusted.

1. To update either BIOS or BMC firmware, the first step is to upload the firmware images. Select the “Images” option under the “Provisions” menu view.
2. Click on the “Add Image” button in the top menu to add a new image for any provisioning need. Enter details on where the firmware image is located, version details, etc. to allow AMI TruE to filter and present matching images for a specific provisioning task.
3. Once a BMC or BIOS firmware image is uploaded, create a job to update the firmware either immediately or at a scheduled time. AMI TruE provides a wizard for creating a job to choose images, select target nodes, and either update them immediately or schedule the update for a future date and time. To create a job, click on “Create New Job” on the “Jobs” page.
4. Enter a name for the job, add a description, and choose either “BIOS Update” or “BMC Update” as the job type. Click “Next” to go the “Image” tab.
5. Select an image that needs to be used and click “Next” to go to the “Targets” page. This page lists all target platforms with a BIOS or BMC that can be updated, depending on the Job Type selected.
6. Choose one or more target platforms that need to be updated and select “Next” to go to the “Schedule” tab. This tab provides options to either run the job immediately or schedule it to be run at a future date and time. After configuring this, click “Next” to go to the “Review Job Settings” tab.
7. Review the information entered for this update job. Use the “Previous” button to navigate to other tabs in this wizard to change any data, if needed. When ready, click “Start” to start or schedule the update job.

8. To know the status of a scheduled task, choose the “Jobs” option under the “Provisioning” menu. This lists all running or scheduled jobs and provides the capability to cancel them if needed.

B.5.5.3 Remote OS Installation

If an OS is compromised, AMI TruE provides options to remotely re-install a version of the OS that is trusted by your enterprise or datacenter. Follow the steps in Appendix B.5.5.2, but choose “OS Deployment” in the “Job Type” field.

Appendix C—Platform Implementation: Kubernetes

This section contains supplementary information describing all the required components and steps required to set up the prototype implementation for Kubernetes.

C.1 Prototype Architecture

Refer to Figure 7 from Appendix B.1 for the relevant architecture diagram.

C.2 Hardware Description

Refer to the hardware descriptions from Appendix B.2 that are used for the Kubernetes setup.

C.3 Kubernetes Installation and Configuration

Kubernetes deployments minimally consist of master node(s) and worker node(s), which utilize a specific container runtime. There is a common set of prerequisites that both types of nodes need, and there are unique configurations for each type of node as well. This implementation installs Docker 19.3.5 as the container runtime, and is running kubelet version 1.17.0 as its prerequisite.

Prerequisite installation:

The following commands enable the network traffic overlays for communications between Kubernetes pods within the cluster.

```
# cat > /etc/modules-load.d/containerd.conf <<EOF
overlay
br_netfilter
EOF

# modprobe overlay
# modprobe br_netfilter
#  cat > /etc/sysctl.d/99-kubernetes-cri.conf <<EOF
net.bridge.bridge-nf-call-iptables  = 1
net.ipv4.ip_forward                 = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF

# sysctl --system
# systemctl enable containerd

```

The following commands add the Kubernetes repository needed for the software package installations.

```
# cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
```

```
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOF
```

The following commands install and start the necessary Kubernetes software packages.

```
# dnf install -y kubeadm-1.17.0 kubelet-1.17.0 kubectl-1.17.0
# systemctl enable kubelet
# echo 'KUBELET_EXTRA_ARGS="--fail-swap-on=false"' > /etc/sysconfig/kubelet
# systemctl start kubelet
```

Running the following command will produce the output shown in Figure 17:

```
# kubectl describe nodes
```

```
System Info:
  Machine ID:                 8a2900f0e5ff4c1fa525e609cf9f7514
  System UUID:                237accad-ab12-03ce-1000-bfde6a6dc4e0
  Boot ID:                    24a26a03-d512-4cd4-85fd-e2e8a35a985a
  Kernel Version:             4.18.0-193.14.3.el8_2.x86_64
  OS Image:                   Red Hat Enterprise Linux 8.2 (Ootpa)
  Operating System:           linux
  Architecture:               amd64
  Container Runtime Version:  docker://19.3.5
  Kubelet Version:            v1.17.0
  Kube-Proxy Version:         v1.17.0
PodCIDR:                      10.10.20.0/24
PodCIDRs:                     10.10.20.0/24
```

Figure 17: Kubelet and Docker Versions

C.3.1 Kubernetes Control Node Configuration

The following command is executed to complete the configuration of the Kubernetes master node in this implementation by establishing the control-plane overlay network.

```
# kubeadm init --pod-network-cidr=10.10.20.0/24
```

Run the following commands to enable the current user, root in this case, to use the cluster.

```
# mkdir -p $HOME/.kube
# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
# sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Run the following command to enable Calico as the Container Network Interface (CNI).

```
# kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
```

C.3.2 Kubernetes Worker Configuration

Once the master node is up and running, worker nodes can be joined to the Kubernetes deployment. A token is required to join the worker nodes to the Kubernetes cluster. Run the following on the master node to obtain the token.

```
# kubeadm token create --print-join-command
```

The following command is executed to join worker nodes to the Kubernetes deployment, which is obtained from the previous command.

```
# kubeadm join 172.16.100.61:6443 --token <token redacted> --discovery-
token-ca-cert-hash \
             sha256:<hash redacted>
```

After the worker node is joined to the Kubernetes cluster, run the following command on the master to verify the nodes in the cluster are ready:

```
# kubectl get nodes
```

C.3.3 Kubernetes Orchestration

In order for the Kubernetes cluster to use the trust measurements and asset tags of hosts in its scheduling policies, the Kubernetes master must be configured to communicate with the attestation hub service in the AMI installation. There is an installation binary on the AMI host verification server at /root/binaries/k8s/isecl-k8s-extensions-v2.0.0.bin that needs to be copied to and run on the Kubernetes control node. This will create Custom Resource Definitions (CRDs) that allow the ISecL trust measurements and asset tags to be leveraged by the Kubernetes scheduler.

A tenant must be created for the Kubernetes hosts, which they must be added to, in the AMI TruE web client. When the tenant is created, the user can choose to associate hosts that are already in the host verification service. After the hosts have been added into the tenant, their trust measurements and asset tags can be used for Kubernetes scheduling policies. In order to enable the trust measurements being used by the Kubernetes scheduler, a few modifications need to be made to the Kubernetes scheduler configurations, and access keys need to be shared between the Kubernetes control node and the AMI TruE host verification server. The AMI TruE host verification server is built from the ISecL code base; the full steps can be found in Section 6.13.3.2 in the ISecL product documentation at https://01.org/sites/default/files/documentation/intelr_secl-dc_v1.6_ga_product_guide_1.pdf.

Appendix D—Supporting NIST SP 800-53 Security Controls

The major controls in the NIST SP 800-53 Revision 5, *Security and Privacy Controls for Information Systems and Organizations* [6] control catalog that affect the container platform security prototype implementation are:

- AU-2, Event Logging
- CA-2, Control Assessments
- CA-7, Continuous Monitoring
- CM-2, Baseline Configuration
- CM-3, Configuration Change Control
- CM-8, System Component Inventory
- IR-4, Incident Handling
- SA-9, External System Services
- SC-1, Policy and Procedures [for System and Communications Protection Family]
- SC-7, Boundary Protection
- SC-29, Heterogeneity
- SC-32, System Partitioning
- SC-36, Distributed Processing and Storage
- SI-2, Flaw Remediation
- SI-3, Malicious Code Protection
- SI-4, System Monitoring
- SI-6, Security and Privacy Function Verification
- SI-7, Software, Firmware, and Information Integrity

Table 7 lists the security capabilities provided by the trusted asset tag prototype:

Table 7: Security Capabilities Provided by the Trusted Asset Tag Prototype

Capability Category	Capability Number	Capability Name
IC1 – Measurements	IC1.1	Measured Boot of BIOS
	IC1.2	Baseline for BIOS measurement (allowed list)
	IC1.3	Remote Attestation of Boot Measurements
	IC1.4	Security Capability & Config Discovery
IC2 – Tag Verification	IC2.1	Asset Tag Verification
IC3 – Policy Enforcement	IC3.1	Policy-Based Workload Provisioning
	IC3.2	Policy-Based Workload Migration
IC4 – Reporting	IC4.1	Support for Continuous Monitoring
	IC4.2	Support for On-Demand Reports

Capability Category	Capability Number	Capability Name
	IC4.3	Support for Notification of Trust Events
IC5 – Remediation	IC5.1	Remotely Powering Down a Compromised Platform
	IC5.2	Updating a Compromised BIOS Firmware
	IC5.3	Reinstalling a Compromised OS

Table 8 maps the security capabilities from Table 7 to the NIST SP 800-53 controls in the list at the beginning of this appendix.

Table 8: Mapping of Security Capabilities to NIST SP 800-53 Controls

NIST SP 800-53 Control	Measurements				Tag Verifi-cation	Policy Enforcement		Reporting			Remediation		
	IC1.1	IC1.2	IC1.3	IC1.4	IC2.1	IC3.1	IC3.2	IC4.1	IC4.2	IC4.3	IC5.1	IC5.2	IC5.3
AU-2								X	X	X			
CA-2				X				X	X				
CA-7								X	X				
CM-2		X		X	X								
CM-3	X		X		X								
CM-8				X	X								
IR-4										X	X	X	X
SA-9						X	X						
SC-1						X	X						
SC-7	X			X		X	X						
SC-29						X	X						
SC-32					X	X	X						
SC-36					X	X	X						
SI-2											X	X	X
SI-3	X	X		X				X	X				
SI-4		X	X	X				X	X				
SI-6	X	X	X	X									
SI-7	X	X	X			X	X				X	X	X

Appendix E—Cybersecurity Framework Subcategory Mappings

This appendix maps the major security features of the container platform security prototype implementation to the following Subcategories from the Cybersecurity Framework [7]:

- DE.CM-4: Malicious code is detected
- DE.CM-6: External service provider activity is monitored to detect potential cybersecurity events
- ID.GV-1: Organizational cybersecurity policy is established and communicated
- ID.GV-3: Legal and regulatory requirements regarding cybersecurity, including privacy and civil liberties obligations, are understood and managed
- ID.RA-1: Asset vulnerabilities are identified and documented
- PR.DS-6: Integrity checking mechanisms are used to verify software, firmware, and information integrity
- PR.IP-3: Configuration change control processes are in place
- PR.IP-5: Policy and regulations regarding the physical operating environment for organizational assets are met
- PR.IP-12: A vulnerability management plan is developed and implemented
- PR.PT-1: Audit/log records are determined, documented, implemented, and reviewed in accordance with policy
- RS.MI-1: Incidents are contained
- RS.MI-2: Incidents are mitigated

Table 9 indicates the mappings from the security capabilities in Table 7 in the previous appendix to the Cybersecurity Framework Subcategories listed above.

Table 9: Mapping of Security Capabilities to NIST Cybersecurity Framework Subcategories

Cybersecurity Framework Subcategory	Measurements				Tag Verifi-cation	Policy Enforcement		Reporting			Remediation		
	IC1.1	IC1.2	IC1.3	IC1.4	IC2.1	IC3.1	IC3.2	IC4.1	IC4.2	IC4.3	IC5.1	IC5.2	IC5.3
DE.CM-4	X							X					
DE.CM-6	X		X			X	X	X	X	X			
ID.GV-1						X	X						
ID.GV-3					X	X	X						
ID.RA-1											X	X	X
PR.DS-6	X	X	X			X	X				X	X	X
PR.IP-3	X		X	X	X								
PR.IP-5					X	X	X						
PR.IP-12											X	X	X
PR.PT-1	X		X					X	X	X			
RS.MI-1											X	X	X
RS.MI-2												X	X

Appendix F—Acronyms and Other Abbreviations

Selected acronyms and abbreviations used in the report are defined below.

AAS	Authentication and Authorization Service
AIK	Attestation Identity Key
AMI TruE	AMI Trusted Environment
API	Application Programming Interface
BIOS	Basic Input/Output System
BMC	Baseboard Management Controller
CA	Certificate Authority
CMS	Certificate Management Service
CNI	Container Network Interface
CoT	Chain of Trust
CPU	Central Processing Unit
CRD	Custom Resource Definition
CRTM	Core Root of Trust for Measurement
FOIA	Freedom of Information Act
GB	Gigabyte
GHz	Gigahertz
HTML5	Hypertext Markup Language (version 5)
HVS	Host Verification Service
IaaS	Infrastructure as a Service
Intel TXT	Intel Trusted Execution Technology
I/O	Input/Output
IP	Internet Protocol
IR	Interagency or Internal Report
IT	Information Technology
ITL	Information Technology Laboratory
JSON	JavaScript Object Notation
KMS	Key Management Service
KVM	Keyboard, Video, Mouse
MLE	Measured Launch Environment
NC	Nonce
NIST	National Institute of Standards and Technology
NISTIR	National Institute of Standards and Technology Interagency or Internal Report
NUC	Next Unit of Computing
OEM	Original Equipment Manufacturer
OS	Operating System
PCR	Platform Configuration Register
REST	Representational State Transfer
RHEL	Red Hat Enterprise Linux

RTM	Root of Trust for Measurement
RTR	Root of Trust for Reporting
RTS	Root of Trust for Storage
SHA256	Secure Hash Algorithm 256-bit
SML	Stored Measurement Log
SMTP	Simple Mail Transfer Protocol
SP	Special Publication
SRK	Storage Root Key
SSDP	Simple Service Discovery Protocol
SSH	Secure Shell
TA	Trust Agent
TPM	Trusted Platform Module
UEFI	Unified Extensible Firmware Interface
URL	Uniform Resource Locator
UUID	Universally Unique Identifier
WLS	Workload Service