NISTIR 8320B

Hardware-Enabled Security:

Policy-Based Governance in Trusted Container Platforms

Michael Bartock
Murugiah Souppaya
Haidong Xia
Raghu Yeluri
Uttam Shetty
Brandon Lum
Mariusz Sabath
Harmeet Singh
Alaa Youssef
Gosia Steinder
Yu Cao
Jayashree Ramanathan

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

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

NISTIR 8320B

Hardware-Enabled Security:

Policy-Based Governance in Trusted Container Platforms

Michael Bartock
Murugiah Souppaya
Computer Security Division
Information Technology Laboratory

Haidong Xia
Raghu Yeluri
Uttam Shetty
Intel Corporation
Santa Clara, California

Brandon Lum
Mariusz Sabath
Harmeet Singh
Alaa Youssef
Gosia Steinder
IBM
Armonk, New York

Yu Cao
Jayashree Ramanathan
Red Hat
Raleigh, North Carolina

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

April 2022

U.S. Department of Commerce
Gina M. Raimondo, Secretary

National Institute of Standards and Technology
Laurie E. Locascio, NIST Director and Undersecretary of Commerce for Standards and Technology

National Institute of Standards and Technology Interagency or Internal Report 8320B
41 pages (April 2022)

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

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

There may be references in this publication to other publications currently under development by NIST in accordance with its assigned statutory responsibilities. The information in this publication, including concepts and methodologies, may be used by federal agencies even before the completion of such companion publications. Thus, until each publication is completed, current requirements, guidelines, and procedures, where they exist, remain operative. For planning and transition purposes, federal agencies may wish to closely follow the development of these new publications by NIST.

Organizations are encouraged to review all draft publications during public comment periods and provide feedback to NIST. Many NIST cybersecurity publications, other than the ones noted above, are available at https://csrc.nist.gov/publications.

Submit comments on this publication to: hwsec@nist.gov

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

All comments are subject to release under the Freedom of Information Act (FOIA).

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, cyber attacks are 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 foundation 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 prototype implementation of the approach intended to be a blueprint or template for the general security community.

Keywords

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

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 container deployment in multi-tenant cloud environments.

Trademark Information

Patent Disclosure Notice

NOTICE: ITL has requested that holders of patent claims whose use may be required for compliance with the guidance or requirements of this publication disclose such patent claims to ITL. However, holders of patents are not obligated to respond to ITL calls for patents and ITL has not undertaken a patent search in order to identify which, if any, patents may apply to this publication.

As of the date of publication and following call(s) for the identification of patent claims whose use may be required for compliance with the guidance or requirements of this publication, no such patent claims have been identified to ITL.

No representation is made or implied by ITL that licenses are not required to avoid patent infringement in the use of this publication.

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 builds upon selected security challenges involving Infrastructure as a Service (IaaS) that are discussed in NIST Interagency or Internal Report (IR) 8320A [1], which addresses cloud computing technologies and geolocation in the form of resource asset tags. Specifically, it uses the three stages of deployment described in Sections 3, 4, and 5 of NIST IR 8320A, and additionally describes two additional stages for encrypting container images and creating data access policies for containers. It then describes a prototype implementation 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 NIST IR 8320, *Hardware-Enabled Security: Enabling a Layered Approach to Platform Security for Cloud and Edge Computing Use Cases* [2]. Reading that NIST IR is a prerequisite for reading this publication because it explains the concepts and defines key terminology used in this publication.

1.2 Terminology

For consistency with related NIST reports, this report uses the following definitions for trust-related terms:

- **Trust**: "The confidence one element has in another that the second element will behave as expected." [3]
- **Trusted**: An element that another element relies upon to fulfill critical requirements on its behalf.
- **Trusted boot**: A system boot where aspects of the hardware and firmware are measured and compared against known good values to verify their integrity and thus their trustworthiness.
- **Trustworthy**: Worthy of being trusted to fulfill whatever critical requirements may be needed.[1]

[1] Based on the definition from NIST Special Publication (SP) 800-160 Volume 2 Revision 1, https://doi.org/10.6028/NIST.SP.800-160v2r1

1.3 Document Structure

This document is organized into the following sections and appendices:

- Section 2 defines the objective for the prototype implementation and the intermediate goals to be met in order to achieve the objective.
- Sections 3 through 7 describe the five stages of the prototype implementation:
    - Stage 0: Have assurance that the platform the container deployment is running on can be trusted
    - Stage 1: Orchestrate the placement of workloads to launch only on trusted platforms
    - Stage 2: Be able to continuously monitor and enforce asset tag restrictions
    - Stage 3: Enable end users to encrypt their workload images
    - Stage 4: Be able to grant workloads access to sensitive information via authentication mechanisms
- The References section lists the references cited throughout 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 IBM and Red Hat describing the components and steps required to set up the OpenShift and Multi-Cloud Manager solutions.
- Appendix C contains supplementary information describing all the required components and steps required to set up the workload encryption implementation.
- Appendix D contains supplementary information describing all the required components and steps required to set up the prototype implementation for using the Trusted Service Identity.
- Appendix E lists the major controls from NIST Special Publication (SP) 800-53 Revision 5 that affect the prototype implementation, as well as the security capabilities the prototype provides, and then maps the prototype's capabilities to the NIST SP 800-53 controls.
- Appendix F maps the major security features of the prototype to Cybersecurity Framework Subcategories.
- Appendix G contains a list of acronyms for the report.

2 Prototype Implementation

This section defines the prototype implementation. Section 2.1 presents 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 five stages of the use case, each of which is examined more closely in Sections 2.2.1 through 2.2.5, respectively.

2.1 Objective

There are security and privacy concerns with allowing unrestricted container deployment orchestration. 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. 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. NIST IR 8320A, *Hardware-Enabled Security: Container Platform Security Prototype* [1] provides an overview of challenges organizations may face when using cloud-based container workloads, as well as techniques to improve the security of cloud computing 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.

The motivation behind this use case is to build upon the stages of NIST IR 8320A and implement additional techniques that leverage hardware roots of trust in server platforms. The integrity and location data of each host are leveraged in the orchestration and protection of workloads, as well as providing workloads access to specific data. Workload orchestration can ensure that containers are instantiated only on server platforms that meet trustworthiness requirements and are in acceptable locations. Orchestration can also involve initial encryption of container images and releasing the decryption keys only to trusted servers. Additionally, the workloads themselves can be assigned identities based on these trusted attributes of the physical servers they reside on and be granted access to sensitive information based on their identities.

2.2 Goals

Using trusted compute pools, described in NIST IR 8320A Sections 3 through 5, is a leading approach to aggregate trusted systems and segregate 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 correct applications (workloads) get deployed there.
3. Enable audits of that portion of the cloud so that users can verify compliance.

Once the trusted compute pools are created, additional techniques can be used to protect the workloads that run on them, or the information that the workloads can access. These additional principles are to:

4. Encrypt workload images and ensure only specific servers can decrypt them.
5. Ensure that only specific applications with location-based restriction enforcement can access sensitive data.

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.

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.

NIST IR 8320A Section 2.2.1 describes the specific goals of Stage 0.

2.2.2 Stage 1: Trusted placement of workloads

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.

NIST IR 8320A Section 2.2.2 describes the specific goals of Stage 1.

2.2.3 Stage 2: Asset tagging and trusted location

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

NIST IR 8320A Section 2.2.3 describes the specific goals of Stage 2.

2.2.4 Stage 3: Trust-based workload encryption

This next stage builds upon stage 2 and adds the ability for end users to encrypt their workload images, which provides at-rest cryptographic isolation to help protect consumer data and intellectual property. In order for a compute node to launch a workload instance from an encrypted image, it will need to retrieve the image decryption key. The purpose of this stage is to ensure that only compute nodes with acceptable platform trustworthiness and specific asset tags will be provided the decryption keys for specific workload images.

Stage 3 includes the following prerequisite goals:

1. **Have trusted asset tag information for each trusted platform instance.** Essentially, stage 2 has been completed, and the platform trust measurements and asset tag information can be leveraged during workload deployment.
2. **Encrypt workload images and protect decryption keys in a key manager.** Decryption keys are kept in a key manager so that authorized nodes in the trusted compute pool can retrieve and launch appropriate instances of workload images.
3. **Release decryption keys for workload images only to servers with trusted platforms and in trusted locations.** Decryption keys are only released to servers that have the appropriate platform trustworthiness and are in allowed locations based on their asset tags.

2.2.5 Stage 4: Trust-based workload access to information

The last stage builds upon stage 3 and adds the ability to grant workloads access to sensitive information. The majority of workloads running in the cloud need some access to data sources or other services, authenticating themselves using a password, application programming interface (API) key, or certificate. Today, this is typically done through secrets that are designed to be stored securely. The purpose of this stage is to ensure that only specific workloads running within a trusted compute pool can use these authentication mechanisms to access sensitive information.

Stage 4 includes the following prerequisite goals:

1. **Deploy workloads only to cloud servers with trusted platforms and in trusted locations.** Essentially, stage 2 has been completed and workloads are running on an appropriate host in the trusted compute pool.
2. **Create an identity for the workload that is signed by its compute node's root of trust.** Each instance of the workload that is instantiated on a compute node will have a unique identity created for it, which is signed by the root of trust on the compute node to prove where it is running.
3. **Grant workloads appropriate access to sensitivity information based on their identity.** When accessing sensitive information, the workload will present its identity from goal 2 and will be granted the appropriate level of access to sensitive information. The level of access is predefined and is determined by the function of the workload, the platform trustworthiness, and the location of the compute node it is running on.

2.3 Additional Resources

For more information on the technical topics being addressed by these stages, 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 (ISCM) 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 IR 8320, *Hardware-Enabled Security: Enabling a Layered Approach to Platform Security for Cloud and Edge Computing Use Cases*
https://doi.org/10.6028/NIST.IR.8320
- NIST IR 8320A, *Hardware-Enabled Security: Container Platform Security Prototype*
https://doi.org/10.6028/NIST.IR.8320A

3 Prototyping Stage 0

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

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 including the container runtime 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. For more detailed information and the solution architecture, please refer to Section 3 of NIST IR 8320A.

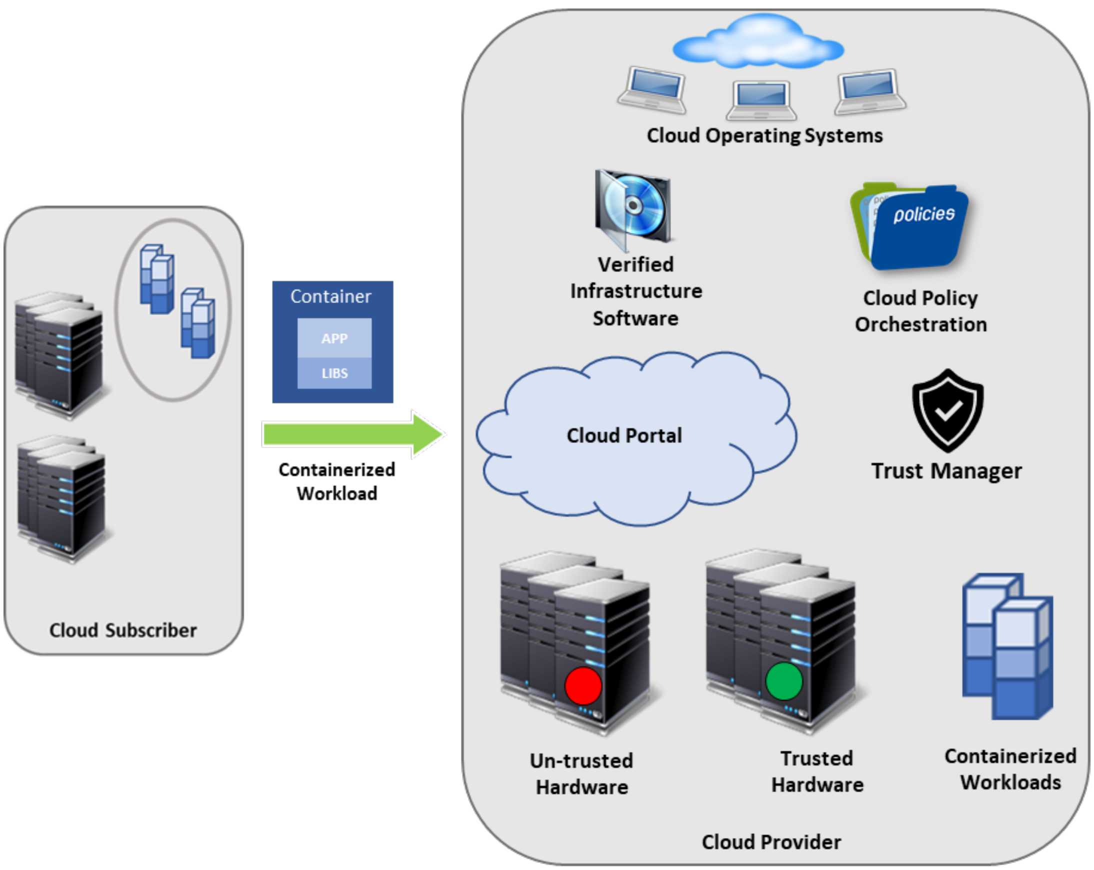

Figure 1: Concept of Trusted Pools

4 Prototyping Stage 1

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

Figure 2 shows the components of the stage 1 solution. It assumes that Server A and Server B are two servers within the same cloud.

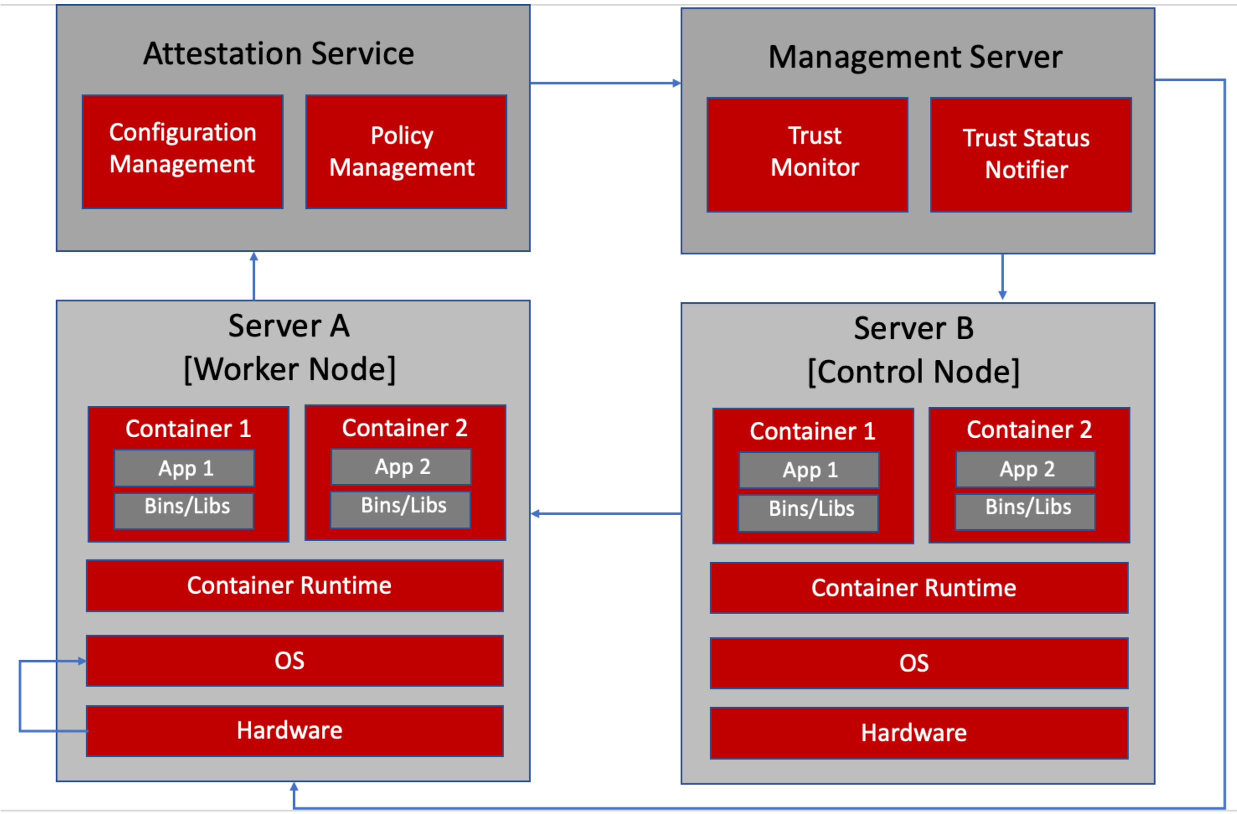

Figure 2: Stage 1 Solution Overview

The solution is comprised of four main components: a control node, a worker node, an attestation service, and a management server. They all work together to deploy container workloads only to worker nodes in a trusted compute pool. For detailed information about the solution overview and the interaction of its components, please refer to Section 4 of NIST IR 8320A.

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. The solution architecture is the same as stage 1; however, the additional asset tag measurement is leveraged in the server hardware roots of trust and is taken into account during deployment of workloads.

Additionally, the capability of monitoring measurements in a governance, risk, and compliance dashboard is introduced in stage 2. For detailed information about the solution overview and a high-level example of what the dashboard may look like, please refer to Section 4 of NIST IR 8320A.

6 Prototyping Stage 3

This section discusses stage 3 of the prototype implementation (trust-based workload decryption), which is based on the stage 2 work and adds components that allow encrypted container images to be decrypted by servers with trusted platform measurements and asset tags.

6.1 Solution Overview

Consumers who place their workloads in the cloud or the edge are typically forced to accept that their workloads are secured by their service providers without insight or knowledge as to what security mechanisms are in place. The ability for end users to encrypt their workload images can provide at-rest cryptographic isolation to help protect consumer data and intellectual property.

When the runtime node service receives the launch request, it can detect that the image is encrypted and make a request to retrieve the decryption key. This request can be passed through an attestation service so that an internal trust evaluation for the platform can be performed. The key request is forwarded to the key broker with proof that the platform has been attested. The key broker can then verify the attested platform report and release the key back to the cloud service provider and node runtime services. At that time the node runtime can decrypt the image and proceed with the normal workload orchestration. The disk encryption kernel subsystem can provide at-rest encryption for the workload on the platform.

6.2 Solution Architecture

Figure 3 shows the operation of the stage 3 solution. It assumes that Server A and Server B are two servers within the same cloud. It uses the same base architecture as stages 1 and 2, but it introduces two additional components: container registry and key broker. The container registry is where encrypted container images are stored, and the key broker stores their decryption keys and can provide trusted servers with access to the keys.

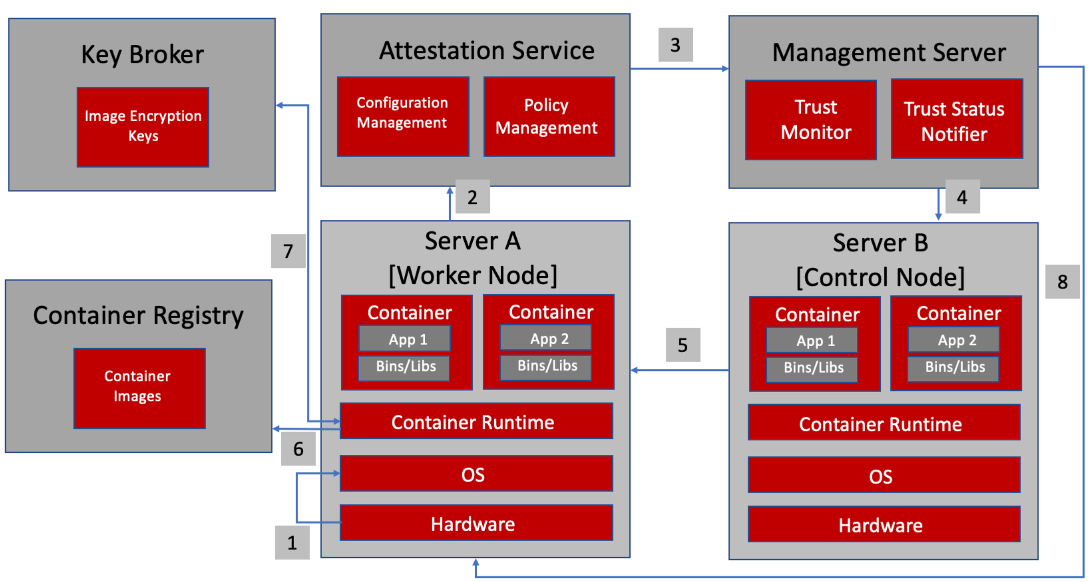

Figure 3: Stage 3 Solution Architecture

There are eight generic steps performed in the operation of the stage 3 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 Attestation Service. The quote includes signed hashes of various platform firmware and operating system (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. Server A pulls the encrypted workload image from the Container Registry so that it can launch an instance of the workload.
7. The Key Broker releases the workload decryption key to Server A only if it has a trusted attestation report, and Server A launches an instance of the workload.
8. Each server platform gets audited periodically based on its measurement values.

7 Prototyping Stage 4

This section discusses stage 4 of the prototype implementation (trust-based workload access to information), which is based on the stage 3 work and adds components that create identities for individual workloads so that they can be granted appropriate access to sensitivity information.

7.1 Solution Overview

The majority of workloads running in the cloud need some access to data sources or other services. To do this, they must authenticate themselves using a password, API key, or certificate. Today, this is typically done through secrets. Even though secrets are designed to be stored securely (for example, encrypted at rest) by the orchestration, they can simply be mounted to an arbitrary container and read by anyone who has access to the namespace, including cloud administrators. Those knowing the secret can also access the sensitive data that needs to be protected. The problem with the secrets is that once they are stored, they are also available to administrators, cloud operators, or anyone else with access to the namespace, whether or not they are authorized to access the data that the secrets protect.

Trust-based workload access to information protects sensitive data access by ensuring only attested services with specific location-based restrictions can access the secrets. This is done through the use of workload identity, which is composed of the trusted hardware identity data that has been fully attested, including the data center location and region, and various runtime measurements to identify the application. These measurements are securely signed by a service running on each worker node, using a chain of trust created during the secure bootstrapping of the environment, then continuously attested and validated. The bootstrapping of the environment requires configuring a secret store that runs a root Certificate Authority (CA), and installing an intermediate CA and token signing service on each worker node. Each worker node with the token signing service uses its hardware root of trust to protect its individual private key.

7.2 Solution Architecture

It is assumed that all the steps from stage 3 have been completed, and that a workload is successfully deployed to a worker node before any steps of this stage begin. Figure 4 shows the operation of the stage 4 solution, with the assumptions that the workload deployed is on top of a trusted worker node and policies have been defined for the measurements of the application that can access secrets. These measurements represent the identity of the application.

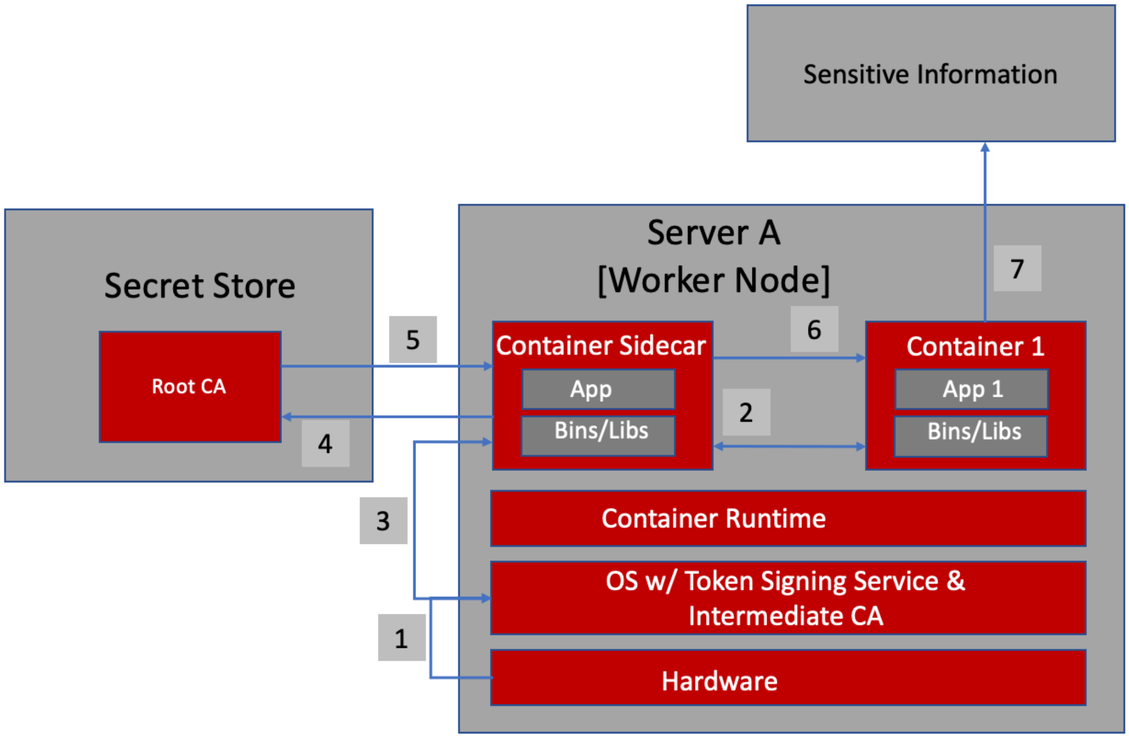

Figure 4: Stage 4 Solution Architecture

There are seven generic steps performed in the operation of the stage 4 prototype, as outlined below and reflected by the numbers in Figure 4:

1. Server A has been bootstrapped by installing an intermediate CA that contains a signing authority which uses the hardware root of trust to protect its private key.
2. When Server A instantiates a workload instance with its sidecar, the sidecar collects the measurements called *claims* that define the identity of this application.
3. The workload sidecar sends the measurements securely to the token signing service on Server A, and the signing service signs these claims using the intermediate CA, then returns the token to the sidecar.
4. The sidecar requests the annotated secrets from the secret store by passing the signed token along with the request.
5. The secret store validates the signature and the expiration date on the token, and if everything is valid, it uses the provided claims against the policies to retrieve the secret. If the measurements match the policy, the secret is released to the application.
6. The sidecar injects the secret into the running workload instance.
7. The running workload instance can easily access the secret locally and use it to obtain sensitive data.

References

[1] Bartock M, Souppaya M, Wheeler J, Knoll T, Shetty U, Savino R, Inbaraj J, Righi S, Scarfone K (2021) Hardware-Enabled Security: Container Platform Security Prototype. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Interagency or Internal Report (IR) 8320A. https://doi.org/10.6028/NIST.IR.8320A

[2] Bartock M, Souppaya M, Savino R, Knoll T, Shetty U, Cherfaoui M, Yeluri R, Malhotra A, Banks D, Jordan M, Pendarakis D, Rao JR, Romness P, Scarfone KA (2022) Hardware-Enabled Security: Enabling a Layered Approach to Platform Security for Cloud and Edge Computing Use Cases. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Interagency or Internal Report (IR) 8320. https://doi.org/10.6028/NIST.IR.8320

[3] Polydys ML, Wisseman S (2009) Software Assurance in Acquisition: Mitigating Risks to the Enterprise. https://apps.dtic.mil/dtic/tr/fulltext/u2/a495389.pdf

[4] 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. Includes updates as of December 10, 2020. https://doi.org/10.6028/NIST.SP.800-53r5

[5] 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 Root of Trust Implementation

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.

A.1 High-Level Implementation Architecture

Figure 5 shows the high-level implementation architecture. The Intel Security Libraries for Data Center (ISecL-DC) server (in the upper left corner) contains the key broker service, attestation service, and utilities for attesting the hardware root of trust and host measurements. Descriptions of each component and installation steps are in the ISecL-DC product guide.

There are two OpenShift clusters, one comprised of virtual machines (VMs) running on a VMware cluster, and another comprised of VMs running on kernel-based virtual machines (KVMs) plus one bare metal host. The first cluster is used as a management cluster, while the second is a managed cluster.

- The managed cluster is the cluster in which the trusted workloads will run. There can be multiple managed clusters governed by the IBM Cloud Pak for Multicloud Management (MCM).
- The management cluster contains the control plane for MCM, as well as DevOps-related tooling.

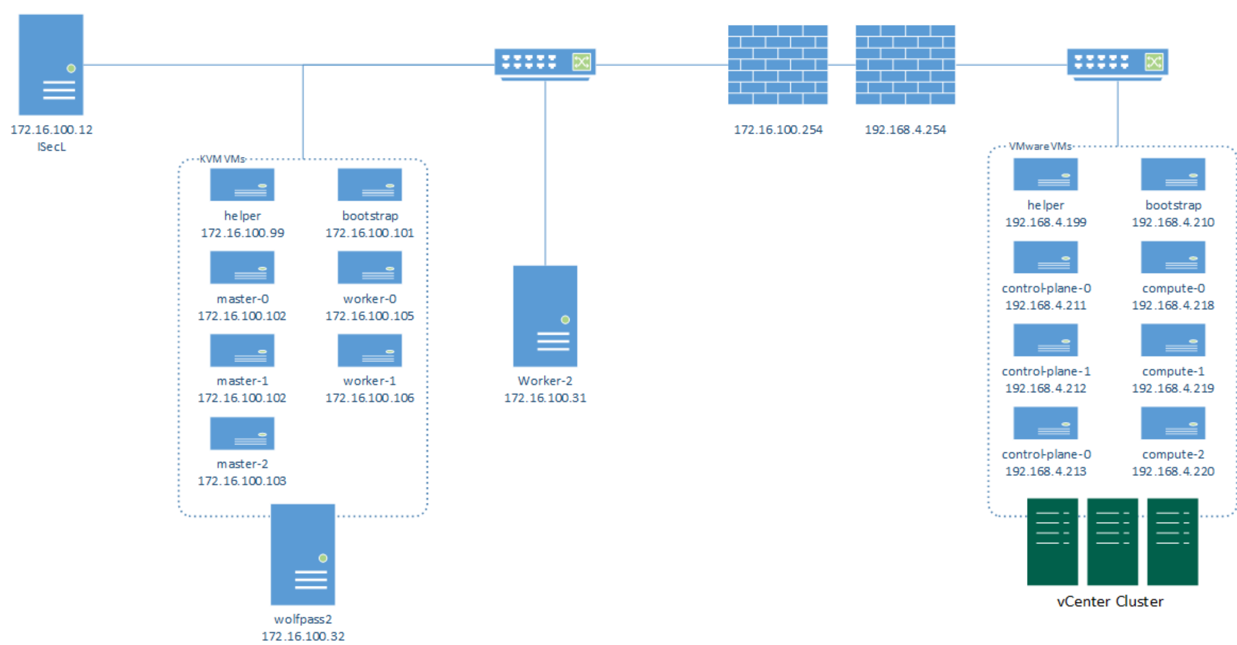

Figure 5: Prototype Implementation Architecture

A.2 Hardware Root of Trust: Intel TXT and 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: root of trust for measurement (RTM), root of trust for reporting (RTR), and root of trust for storage (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 Trusted Execution Technology (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 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 safeguard the 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. This is achieved by the attestation process defined in the next subsection. 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.

A.3 Attestation: Intel Security Libraries (ISecL)

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.

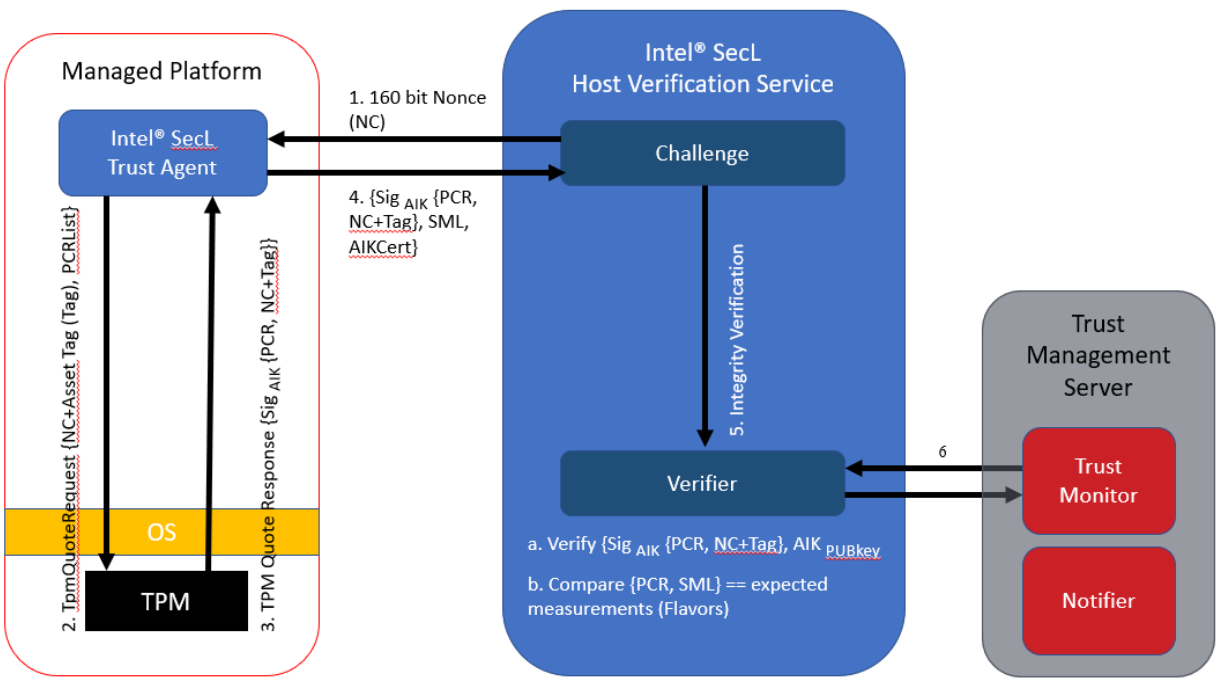

Figure 6: Remote Attestation Protocol

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 TPMQuoteRequest, the TPM loads the attestation identity key (AIK) from protected storage in the TPM by using the storage root key (SRK), then 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 AIK credential, which consists of the AIKpub that was signed by a Privacy-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 manager of the trust state of the remote node. The manager records the trust state in its management database and uses it for any individual or aggregated device status requests. If an administrator is 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.

Once the ISecL trust agent and host verification service are installed successfully, asset tags can be created and provisioned to each managed server. Section 6.4 in the ISecL-DC v1.6 product guide describes the steps of creating and provisioning asset tags.

Appendix B—Workload Orchestration Implementation: OpenShift

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

B.1 Prototype Architecture

Kubernetes has become a popular source for building large web-scale technologies that enable the enterprise to take advantage of innovations like analytics, artificial intelligence, machine learning, and cloud services. Supporting advanced technologies and building cloud-native applications requires enterprise-grade platforms such as Red Hat OpenShift. This section describes the provisioning and configuration of OpenShift clusters. Figure 7 shows how the nodes from Figure 5 (in Appendix A) in the architecture are logically implemented into two OpenShift clusters and the Remote Attestation Server.

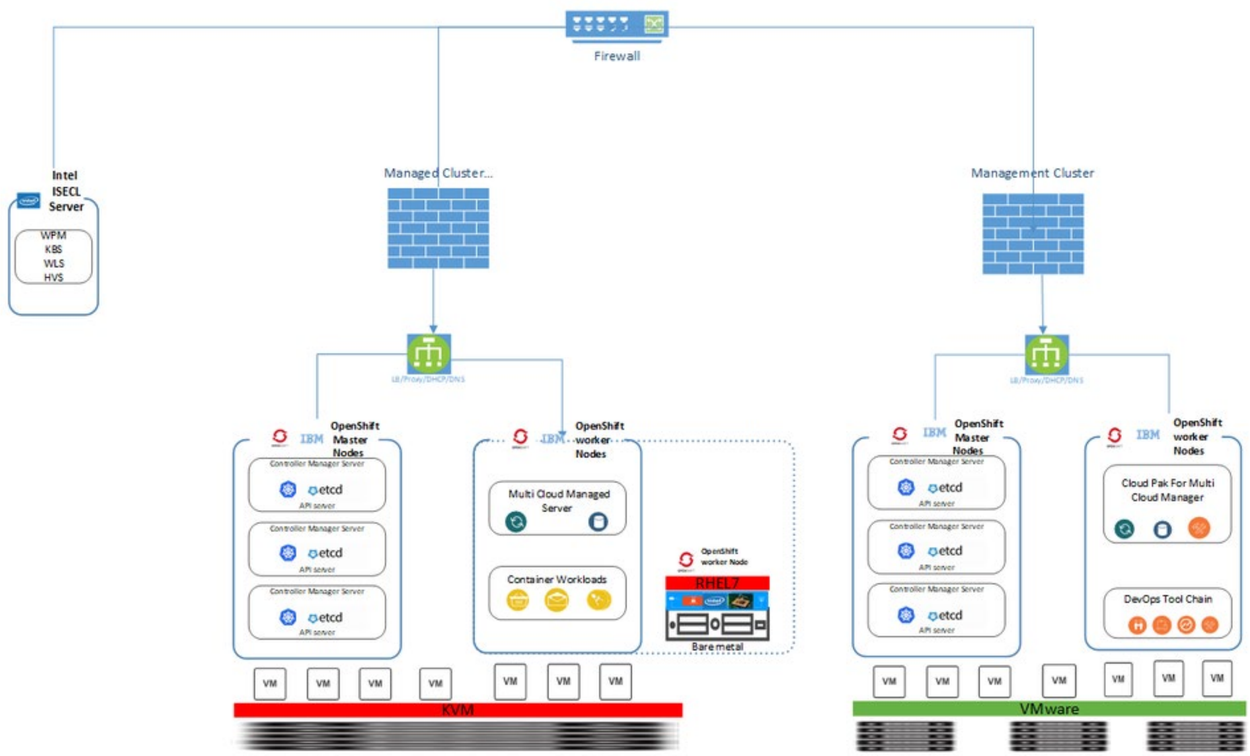

Figure 7: Prototype Architecture

To implement use cases, OpenShift is deployed in two separate clusters: a Management Cluster based on VMWare, and a Managed Cluster based on KVM infrastructure. Each cluster has three control-plane and three worker nodes, and a VM with load balancer and Domain Name System (DNS) to simulate the trusted container workload environment. MCM components are deployed on both OpenShift clusters. The Managed Cluster has an MCM Hub component, which aggregates information from multiple clusters using an asynchronous work request model. The hub cluster (Management Cluster) maintains the status of clusters and applications, and provides a set of APIs for the various functions to support as a central controller. The Managed Cluster has an MCM managed-cluster component used to define the cluster, with the MCM Klusterlet and other resources configured to initiate a connection to the hub cluster. The Managed Cluster receives work requests and applies them, then returns the results.

B.2 OpenShift Installation and Configuration

Managing the OpenShift platform in multiple places comes with challenges like complexity, governance, and cost. For example, how do you gain visibility into all the clusters to see where the application's components are running? How do you know which systems are failing? How can you monitor usage across the clouds and clusters? How do you govern the configuration and changes to this environment? IBM Cloud Pak for Multicloud Management/Red Hat Advanced Cluster Management for Kubernetes supports the use cases. It is based on the Kubernetes community direction and includes advanced functions important to running enterprise-grade environments.

The Policies repository, which is part of the Manager hub component, resides on the Management Cluster. The repository comes with default compliance and policies templates, but our use cases developed new policies that reflected our environment to integrate with the Intel attestation hub. The repository holds the policy defined for the Managed Cluster, and the policy document is applied by using placement bind.

B.2.1 VMware-Based Management Cluster (Cluster A)

Hardware Requirement: For deploying OpenShift Container Platform (OCP) 4.3 on VMware, the minimum recommendation is to provision one ESXi server and one Centos/Red Hat VM on the same virtual local area network (VLAN) in the local datacenter. For this deployment, the setup was an ESXi bare-metal server with 48 CPUs, 256 GB random-access memory (RAM), and 2 TB storage. The Centos/Red Hat VM is only required for a few hours and can be de-provisioned after the install is complete.

Networking: The IP addresses used in this process and the configuration files came from our National Cybersecurity Center of Excellence (NCCoE) environment. They are used here for illustration purposes only. Besides setting up your ESXi and vCenter server, you need to have a minimum of 16 IP addresses to assign to the VMs. Each VM node takes one IP address. The recommended minimum of 16 IP addresses is determined by: 1 helper node + 1 boot node + 3 control-plane nodes + 3 worker nodes = 8 nodes. The extra IP addresses are available in case additional worker nodes are required in the future. This installation provisioned the vCenter on the same IP subnet, so a total of 9 IP addresses were used.

VMware OCP VM Requirements: Table 1 lists the VMs that are instantiated on the VMware server, along with their virtual hardware requirements and the roles they serve in the cluster.

Table 1: VMs Instantiated on the VMware-Based Management Cluster

Node Name	vCPU	Mem (GB)	HDD (GB)	Role
Helper Node	4	16	150	LB/DNS/Proxy/DHCP/OCP Installer
Bootstrap-0	4	16	150	Bootstrap OCP
Control-plane-0	4	16	150	Controller OCP
Control-plane-1	4	16	150	Controller OCP
Control-plane-2	4	16	150	Controller OCP
compute-0	4	16	150	Compute OCP
compute-1	4	16	150	Compute OCP
compute-2	4	16	150	Compute OCP

OCP VMware Deployment Playbooks: To deploy OCP 4.3 on VMware, download the following Git repository: https://github.com/fctoibm/ocpvmware4.3 and follow the steps to run the playbooks. Make sure to change the `vars.yaml` and `host.yaml` files to match the networking information for your environment.

B.2.2 KVM-Based Managed Cluster (Cluster B)

The second OCP cluster is the managed cluster. It contains an MCM Klusterlet, which ensures that each managed cluster adheres to the policy in place.

Hardware Requirement: For this deployment, the lab setup was on a CentOS bare-metal server with 48 CPUs, 256 GB RAM, and 1 TB storage. KVM will be used to create and manage virtual machines. The KVM command line tool is `virt-install` and the graphical user interface (GUI) tool is `virt-manager`. To use the KVM GUI tool, install Gnome desktop and VNC on the CentOS bare-metal server. All of the VMs for this managed cluster are deployed on this single KVM host, which has hostname `wolfpass2` in the table and image in Figure 5 (in Appendix A).

Networking: The IP addresses used in this process and the configuration files came from our NCCoE environment. They are used here for illustration purposes only. When you install OCP in the KVM host environment, you will need a minimum of 16 portable IP addresses. Each VM node takes up one IP address. The recommended minimum of 16 portable IP addresses is determined by: 1 helper node + 1 boot node + 3 control-plane nodes + 3 worker nodes = 8 nodes. The extra IP addresses are available for additional worker nodes required in the future. You should plan your IP address space accordingly.

KVM OCP VM Requirements: Table 2 lists the VMs that are instantiated on the KVM server, along with their virtual hardware requirements and the roles they serve in the managed cluster.

Table 2: VMs Instantiated on the KVM-Based Managed Cluster

Node Name	vCPU	Mem (GB)	HDD (GB)	Role
Helper Node	4	16	150	DNS/Proxy/DHCP/OCP Installer
Bootstrap	4	16	150	Bootstrap OCP
Master0	4	16	150	Controller OCP
Master1	4	16	150	Controller OCP
Master2	4	16	150	Controller OCP
Worker0	4	16	150	Compute OCP
Worker1	4	16	150	Compute OCP

Note: The OpenShift cluster requires three worker nodes; however, since this deployment uses an additional physical server for the third worker node, only two worker node VMs are deployed.

OCP KVM Deployment Playbooks: To deploy OCP 4.3 on KVM, download the following Git repository: https://github.com/fctoibm/ocpkvm4.3 and follow the steps to run the playbooks. Make sure to change the `vars.yaml` and `host.yaml` files to match the networking information of your environment.

The OCP KVM deployment playbook creates all of the worker nodes as virtual machines. In order to create policies bases on hardware roots of trust, a physical server with the Intel TXT and TPM capabilities must be added as an additional worker node to the cluster. This server needs to have the corresponding Red Hat Enterprise Linux (RHEL) OS installed, as well as the Intel Trust Agent as described in Appendix A.2. For the physical server, the OpenShift documentation details how to add an RHEL compute node to an existing cluster.

B.2.3 Installing MCM Pak 1.3 (MCM HUB - VMware)

To install MCM Pak 1.3 on OCP 4.3, the setup assumes OCP 4.3 is already installed and administrator-level access is available to deploy the MCM Pak. The guide assumes OCP 4.3 was installed using the same GitHub repository and the same `vars.yaml` file.

Deploying MCM Pak: The installation Git repository supports two options, VMware or KVM install, and both will deploy a VM guest if required. The VM guest called PakHelper node will act as a client to install MCM Pak. There is no reason to deploy a VM guest client if you already have a VM guest Centos 7 OS available in the same network. If the Centos 7 VM is already in place, please skip options 1 and 2, but if there is no VM guest available, please execute both options 1 or 2 and 3 from the following Git repository: https://github.com/fctoibm/mcmpak1.3 and follow the steps to run the playbooks.

Adding KVM OCP as Managed Cluster in MCM: Once the MCM Pak has been deployed, the KVM OCP cluster can be imported into, and managed by, the IBM MCM. To do so, browse to the web user interface (UI) of MCM and navigate to the Clusters management page. As shown in Figure 8, there is an option to import an existing cluster. To import the existing KVM OCP cluster, perform the steps in this IBM knowledge center article.

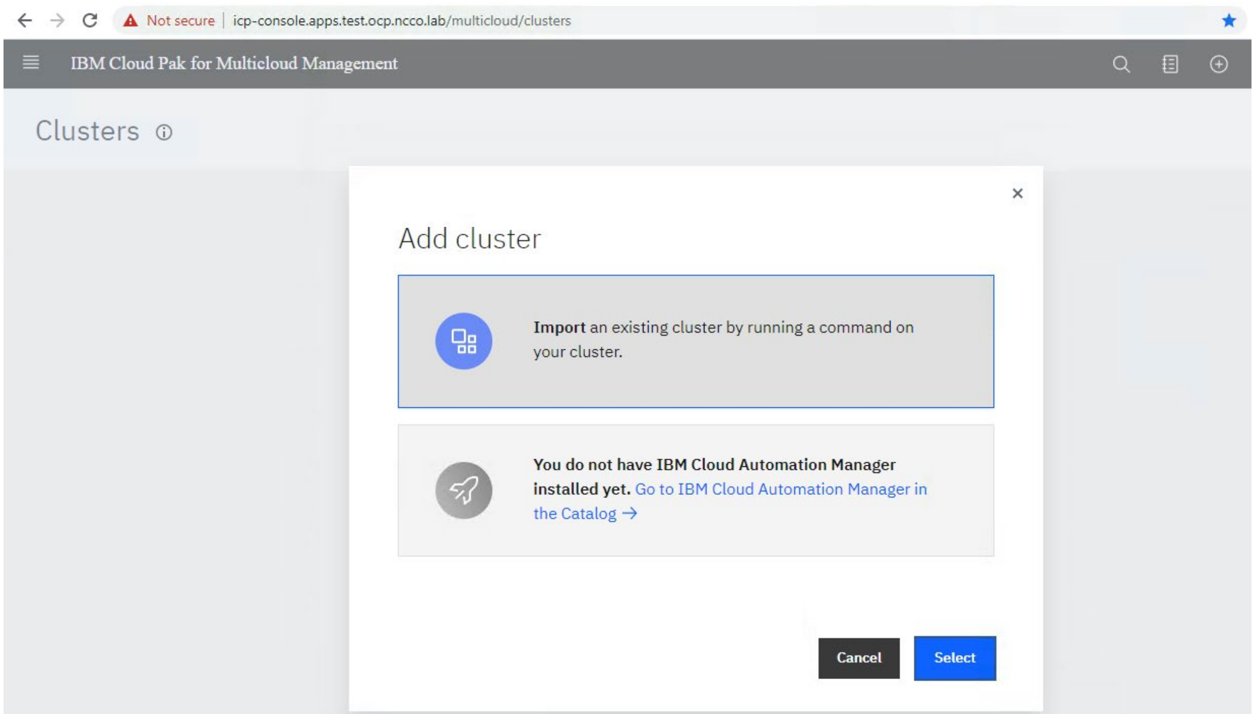

Figure 8: MCM Console to Import a Cluster

Creating Policies for Managed Cluster: Once the KVM OCP cluster has been successfully imported, specific polices that govern workload orchestration can be created and applied. Policies are created in the MCM Hub and these policies are propagated to each managed cluster, where they are enforced. Two policies were put in place for our prototype implementation:

1. The "Trusted Node Policy" ensures that all nodes in the cluster are trusted and attested. In the inform mode, the policy logs whenever the trust status of a node has been violated. In the enforce mode, the policy drains and removes the node from the cluster.
2. The "Trusted Container Policy" ensures that all workloads run within a namespace are using a set of images from a particular registry path. The infrastructure is set up so only encrypted container images are in that path. This makes it so only encrypted images are run within the namespace.

Figure 9 shows the two policies in the web user interface that have been created for the managed clusters.

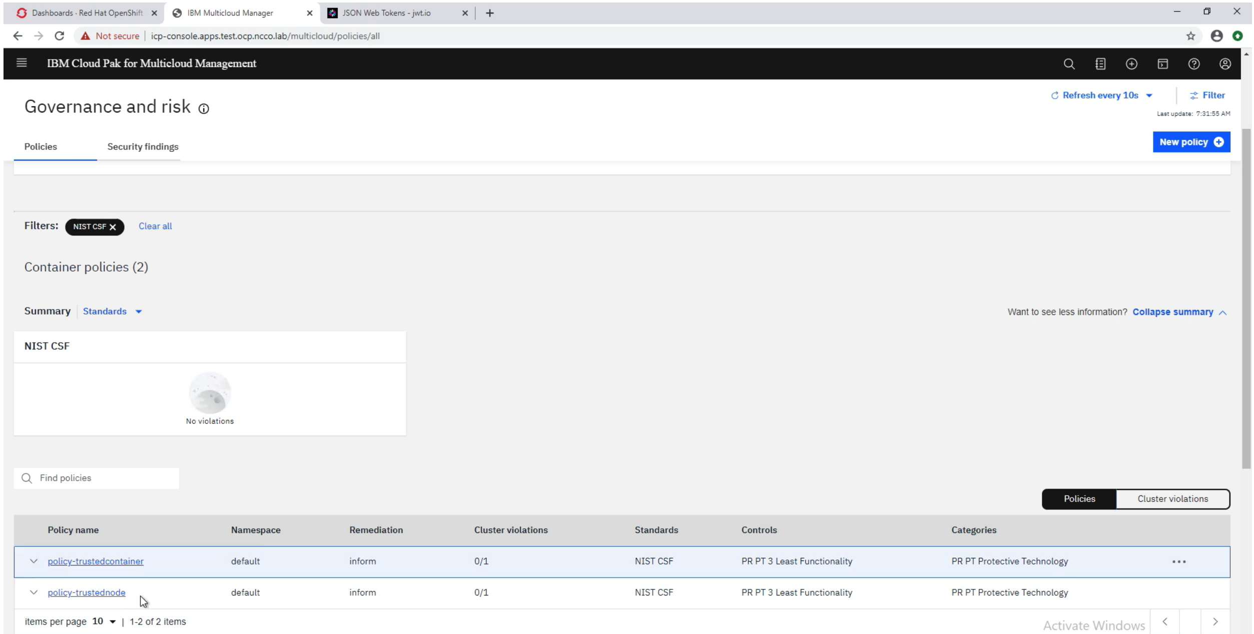

Figure 9: Managed Cluster Policies

In addition to these two policies, there is a Tekton task set up as part of the OpenShift pipeline that does a set of checks and encrypts the image. This secure pipeline does building, vulnerability scanning, and encryption. More details on this pipeline are provided in Appendix C.

Appendix C— Workload Encryption Implementation

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

C.1 Prototype Architecture

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

C.2 Workload Encryption Configuration

Various parts of the container ecosystem allow the enablement of workload encryption via container image encryption. The technology is based on the Open Container Initiative (OCI) container image specification. The components that support the use of this are:

- **Build**: The Skopeo tool is used to encrypt the container images and push them to the registry.
- **Runtime**: The Cri-o container runtime came as part of OpenShift and was configured to decrypt the images. It is the default runtime of OpenShift 4.3 worker nodes and supports decrypting OCI container images.
- **Registry**: The Docker Distribution registry is used to push, pull, and store the encrypted images. The version used was v2.7.1.

These are the core components of workload encryption. Several integrations were required with the ISecL Attestation Hub's APIs to showcase the workload and image encryption with hardware attestation. A custom container encryption metadata scheme was defined to work with the ISecL key broker. The reference implementation code and the document are located at https://github.com/lumjjb/seclkeywrap.

Key integration points are:

- A custom container encryption metadata scheme was defined to work with the ISecL key broker.
- The core components CRI-0 and Skopeo were patched to enable the use of the custom ISecL protocol. The patches are available at https://github.com/lumjjb/cri-o/tree/sample_integration and https://github.com/lumjjb/skopeo/tree/sample_integration, respectively.

As part of the DevSecOps cycle, it was integrated with the development flow with the Tekton pipeline to perform builds, security checks, and image encryption. The pipeline workflow can be reached through the OpenShift dashboard by toggling into the Developer role and selecting the "Pipelines" menu, as shown in Figure 10. The definitions of the Tekton objects are available here: https://gist.github.com/lumjjb/22191008f849f240851aec8a1ee0304d

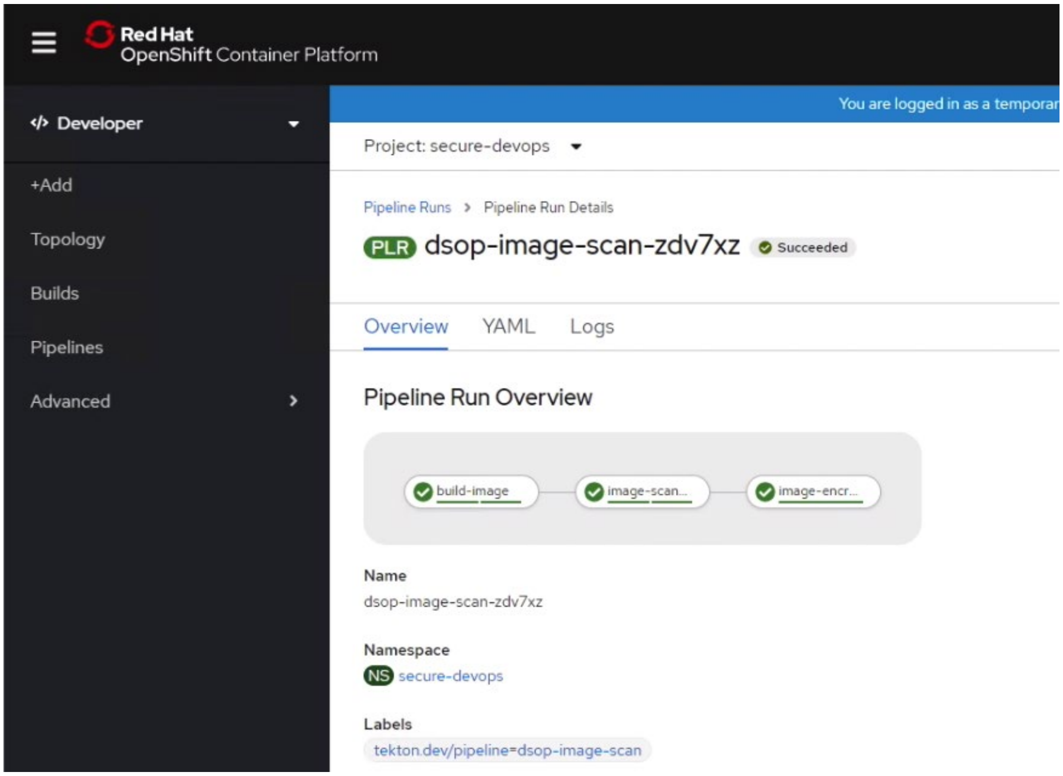

Figure 10: Creating Pipeline for Image Decryption

Appendix D—Trusted Service Identity (TSI)

This section contains supplementary information describing all the required components and steps required to set up the prototype implementation for the Trusted Service Identity (TSI).

D.1 TSI Overview

TSI protects sensitive data access by ensuring only attested services with specific location-based restrictions can obtain credentials. This is done through the use of workload identity, composed of the trusted hardware identity data that has been fully attested by Intel TXT, including the data center location and region, and various runtime measurements like the image and cluster name, unique pod IDs, and namespace to identify the application. These measurements are securely signed by a service running on every hosting node, using a chain of trust created during the secure bootstrapping of the environment, then continuously attested and validated.

Every container that requires a secret for accessing sensitive data is assigned a short-lived measured identity, in the form of a JSON Web Token (JWT) token, that is signed with the root of trust by the attested process. This measured identity is a form of an *ephemeral digital biometric* with a short time to live.

In this implementation, there is a Kubernetes cluster running on OpenShift, extended with TSI, where each node has an intermediate CA signed by the root CA during the secure bootstrapping of the cluster.

During this bootstrapping, the installation process obtains an attestation report (Security Assertion Markup Language [SAML]) from the Attestation Server for every worker node. This report is checked to verify if all the components are trusted (OS, platform, and software), and the bootstrapping process retrieves worker identity fields from the asset tag of this report. Each worker node also has a JWT Signing Service (JSS) that contains a signing authority that uses the hardware TPM to protect its individual private key.

The root CA is securely stored in a vault, which is extended with the TSI Authentication Vault plugin.

D.2 TSI Installation and Configuration

TSI requires an attestation process to accurately define the identity of the worker nodes hosting the application containers. In this implementation, TSI relies on the ISecL server that has been deployed to provide the identity of the worker nodes. Steps detailing the integration of TSI with ISecL can be found here: https://github.com/IBM/trusted-service-identity/blob/intel-asset/README.md#attestation.

This process requires two independent phases:

- **Asset Registration with Intel Verification Server** – The trusted bootstrapping process responsible for installing the environment must properly set the identity attributes of every worker node. These identity values in the form of asset tags are securely stored in their corresponding TPMs on hosts. As a result, they are included in the SAML attestation report that also includes all the attestation results (OS, platform, software trusted). This process was performed in the steps outlined in Appendix A.

- **TSI Deployment with Attestation** – The implementation outlined in this document allows for the use of the Intel Attestation Server to obtain the identity of the worker nodes. There are several changes required to configure the TSI installation to support Intel Attestation Server. Additionally, a hardware TPM device is shared between the Intel Trust Agent and the TSI JWT Signing Service, and it requires the use of TPM proxy. For details outlining the suggested configuration changes, visit https://github.com/IBM/trusted-service-identity/blob/intel-asset/README.md#attestation.

As a result of these changes, TSI will be installed in the cluster, using an attestation report from the Intel Attestation Service to provide the identities of the workers and to keep the attestation going.

Before secrets can be injected into the application container, first they need to be created in the Secret Store (Vault). Follow these steps for injecting secrets into the Vault: https://github.com/IBM/trusted-service-identity/blob/master/examples/vault/README.md#secrets.

Once the application is started, secrets will be injected based on the application identity, including the workload environment and location. As a result, the secret will be delivered to the container runtime memory without ever being stored anywhere in Kubernetes—but from the point of view of the application, no additional changes were needed.

Figure 11 shows a sample JWT created by TSI. Notice its three parts: the header, the payload containing the actual claims, and a signature for validation.

Figure 11: Sample JWT Created by TSI

The claims included here represent the measured identity of the application. They contain some static values like `cluster-region` (e.g., Germany, eu-de), `cluster-name`, individual `machineid` which is a unique worker node ID, and several runtime measurements like a namespace, unique pod ID, and the list of images making up the pod. These images include the image signature, so you can validate the image and guarantee that the application is running the code that you want to be running and it was not tampered with. The `hd-trusted` value determines whether all the attested elements are trusted.

Values for `cluster-name`, `cluster-region`, and `hd-trusted` are essential for defining the identity of the compute resources, and they are read from Intel's Attestation Server. There is also a token expiration timestamp, typically set to one minute, to make these tokens ephemeral and short-lived, protecting the security from leaking. These are the runtime measurements that represent the identity of the application, signed with the root of trust, and used for evaluation against policies controlling the secrets.

The secrets stored in the Vault are protected by policies. Policies are composed of the policy-type and the attributes, the same that are used for building claims, and they represent the path to the secret. If the claims provided in the request match the policy attribute path, the secret will be released to the application.

Appendix E—Supporting NIST SP 800-53 Security Controls and Publications

The major controls in the NIST SP 800-53 Revision 5, Security and Privacy Controls for Information Systems and Organizations [4] 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-3, Malicious Code Protection
- SI-4, System Monitoring
- SI-6, Security and Privacy Function Verification
- SI-7, Software, Firmware, and Information Integrity

Table 3 lists the security capabilities provided by the prototype:

Table 3: Security Capabilities Provided by the 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
	IC3.3	Policy-Based Workload Decryption
	IC3.4	Policy-Based Workload Access

Capability Category	Capability Number	Capability Name
IC4 – Reporting	IC4.1	Support for Continuous Monitoring
	IC4.2	Support for On-Demand Reports
	IC4.3	Support for Notification of Trust Events

Table 4 maps the security capabilities from Table 3 to the NIST SP 800-53 controls in the list at the beginning of this appendix.

Table 4: Mapping of Security Capabilities to NIST SP 800-53 Controls

NIST SP 800-53 Control	Measurements				Tag Verifi-cation	Policy Enforcement				Reporting		
	IC1.1	IC1.2	IC1.3	IC1.4	IC2.1	IC3.1	IC3.2	IC3.3	IC3.4	IC4.1	IC4.2	IC4.3
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
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-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					

Appendix F—Cybersecurity Framework Subcategory Mappings

This appendix maps the major security features of the trusted geolocation prototype implementation to the following subcategories from the Cybersecurity Framework [5]:

- ID.GV-1: Organizational information security policy is established
- ID.GV-3: Legal and regulatory requirements regarding cybersecurity, including privacy and civil liberties obligations, are understood and managed
- PR.DS-6: Integrity checking mechanisms are used to verify software, firmware, and information integrity
- PR.IP-5: Policy and regulations regarding the physical operating environment for organizational assets are met

Appendix G—Acronyms and Other Abbreviations

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

AIK	Attestation Identity Key
API	Application Programming Interface
BIOS	Basic Input/Output System
CA	Certificate Authority
CPU	Central Processing Unit
CRTM	Core Root of Trust for Measurement
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
FOIA	Freedom of Information Act
GB	Gigabyte
HD	Hard Drive
HDD	Hard Disk Drive
IaaS	Infrastructure as a Service
Intel TXT	Intel Trusted Execution Technology
I/O	Input/Output
IP	Internet Protocol
IR	Interagency or Internal Report
ISecL	Intel Security Libraries
ISecL-DC	Intel Security Libraries for Data Center
IT	Information Technology
ITL	Information Technology Laboratory
JSON	JavaScript Object Notation
JSS	JWT Signing Service
JWT	JSON Web Token
KVM	Kernel-Based Virtual Machine
MCM	Multicloud Management
MLE	Measured Launch Environment
NC	Nonce
NCCoE	National Cybersecurity Center of Excellence
NIST	National Institute of Standards and Technology
OCI	Open Container Initiative
OCP	OpenShift Container Platform
OS	Operating System
PCR	Platform Configuration Register
RAM	Random-Access Memory
RHEL	Red Hat Enterprise Linux
RTM	Root of Trust for Measurement
RTR	Root of Trust for Reporting
RTS	Root of Trust for Storage
SAML	Security Assertion Markup Language
SML	Stored Measurement Log
SP	Special Publication
SRK	Storage Root Key

TB	Terabyte
TPM	Trusted Platform Module
TSI	Trusted Service Identity
VLAN	Virtual Local Area Network
VM	Virtual Machine