Secure the IoT Edge with Trusted Docker* Containers
Contents

Contents............................................................................................................................................................3
Authors ...............................................................................................................................................................4
Executive Summary ............................................................................................................................................4
1.0  Background ................................................................................................................................................4
2.0  Motivation and Objectives ..........................................................................................................................6
  2.1  Intel® Platform Trust Technology and a trusted platform module (TPM) ...............................................6
  2.2  Summary of Trusted Containers .............................................................................................................6
3.0  Trusted Containers ....................................................................................................................................7
  3.1  Prerequisites and Dependencies ............................................................................................................8
  3.2  Security in Transit ...................................................................................................................................9
  3.3  Security at Rest ......................................................................................................................................10
  3.4  Node Integrity .......................................................................................................................................10
  3.5  Image Integrity .......................................................................................................................................11
  3.6  Node Attestation ...................................................................................................................................12
  3.7  Registry Authentication ..........................................................................................................................12
  3.8  SELinux Integration ................................................................................................................................14
  3.9  Hardened Configuration ..........................................................................................................................14
4.0  Conclusion ................................................................................................................................................15

Figures

Figure 1. Trusted Containers ...........................................................................................................................9
Figure 2. Image Integrity ................................................................................................................................12
Figure 3. Registry Authentication ....................................................................................................................13
Figure 4. Registry Authentication p.2 .............................................................................................................13

Tables

Table 1. Summary of Trusted Containers ........................................................................................................6
Executive Summary

IoT security is increasingly important and with the rapid adoption of container and orchestration technologies it is essential to harden the overall container infrastructure to protect against malicious attacks and other anomalies that may cause disruptions to the operations of the IoT deployments. This paper presents an overview of trusted containers. It also presents several foundational steps towards the hardening of the container infrastructure running on the IoT edge device as well as the backend connectivity to the infrastructure. This is achieved using Intel® Platform Trust Technology (Intel® PTT) and a trusted platform module. The paper will present and discuss the details of the hardening steps taken and provide rationale for why those are pertinent.

1.0 Background

With the digital transformation well underway in all the IoT market verticals (connected home, industrial, manufacturing, healthcare, retail, smart buildings, smart cities etc.), security is continuing to be one of the most critical pillars to help protect data, ensure integrity, and availability of the system. Container technologies play an increasingly important role here, whether in virtualized environments or executed directly on more constrained edge devices. Workloads can be monolithic applications (especially when migrating traditional applications) but are often being developed as
microservice-based applications. These applications are managed and orchestrated with an on-premise or remote device manageability solution, in conjunction with IoT core and edge infrastructure.

This paper reflects an initiative by Intel and Docker to supply enterprise-level security objectives for trusted containers. The joint collaboration provides a solution by leveraging advanced security features in the Docker Engine - Enterprise, and Intel® Platform Trust Technology (Intel® PTT) and a trusted platform module (TPM).
2.0 Motivation and Objectives

Container technologies are instrumental to the digital transformation and provides a number of critical benefits to the IoT verticals such as:

- Self-contained, isolated and well-defined means of resource sharing
- Platform independence
- Lightweight and resource efficient
- Scaling at ease

Containers provide many critical foundational security characteristics but have not yet integrated to security enabled silicon functions such as Intel® PTT which can help to further secure the solution. In this paper, we are presenting new security hardening features provided by Docker as well as OS level services that further ensure confidentiality and integrity measures.

2.1 Intel® Platform Trust Technology and a trusted platform module (TPM)

A trusted platform module (TPM) provides a multitude of cryptographic services, provides the foundation for secure storage of assets, facilitates device onboarding, underpins remote attestation services, and general integrity and compliance of the platform through measurements and integrity checks of firmware and software running on the system. To read more about TPM, please refer to the following article:


2.2 Summary of Trusted Containers

The next section will provide more technical details of the different security hardening features enabled either by Docker or through the operating system that Docker is leveraging.

Table 1. Summary of Trusted Containers

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security in Transit</td>
<td>Docker integrates natively with a trusted platform module (TPM) subsystem to store sensitive credentials used for backend connection to enterprise infrastructure. The credentials are created in a TPM and cannot be exfiltrated.</td>
</tr>
</tbody>
</table>
Docker can leverage disk encryption to store images in an encrypted volume, which is backed by keys residing in a TPM and tied to compliance state of the platform.

Critical Docker infrastructure including configuration and other meta-data is measured and appraised through a local enforcement policy to prevent compromised files and data from being accessed and is anchored in a TPM.

Critical Docker infrastructure including configuration and other meta-data is measured by IMA (Integrity Measurement Architecture) which is anchored in a TPM and chained to the integrity of the Secure Boot flow and can be attested by a remote verifier.

Docker uses credentials stored in a trusted platform module to authenticate with a Docker Trusted Registry to securely authenticate a device identity.

With Docker Content Trust, Docker signs images and verifies their signature before pulling or running them. With Node attestation, this verification is protected by a trusted platform module (TPM) as well.

Docker infrastructure is leveraging (Security Enhanced Linux*) to provide further authorization and access control policies beyond DACL.

The Docker infrastructure and configuration ensures that only enterprise-approved TLS versions and ciphers suites are being used in addition to preventing any type of insecure fallback of protocols or otherwise.

While the above-mentioned capabilities provide necessary foundational capabilities for many IoT deployments, it should be pointed out that they may not be sufficient in every situation. Further steps may be required depending on Enterprise IT/OT objectives, requirements and adversary models.

3.0 Trusted Containers

This section describes an overview of the joint collaboration between Intel and Docker to provide secure container infrastructure. As mentioned in the prior section, security is chained and rooted in a trusted platform module (TPM). However, there are also other critical ecosystem and OS services needed in order to provide the functionality described in this chapter.
3.1 Prerequisites and Dependencies

There are several software components and frameworks that integrates in between the silicon technology and Docker's infrastructure. Here we will take a closer look at what some of those platform dependencies are.

**TPM 2.0 software stack.** A trusted platform module software consists of several distinct components necessary to leverage a trusted platform module (TPM) functionality. The tpm2-tss stack provides an implementation of the TCG TPM 2.0 specification. The tpm2-abrmd (Access Broker and Resource Manager) provides a co-existence function and marshalling of TPM related commands from user space applications down to the tpm2-tss stack and ultimately a trusted platform module itself. The tpm2-tools provides a high-level set of tools that provides a good reference for essential trusted module platform commands required to perform common and critical trusted platform module operations.

TPM 2.0 PKCS#11 API provides a cohesive PKCS#11 API to a trusted platform module for more HSM (Hardware Security Module) agnostic access and use of a trusted platform module (TPM).

TPM 2.0 integration utilities provide a high level convenience framework to setup and configure LUKS (Linux Unified Key Service) based partitions with support for integration with TPM 2.0 where the credentials needed to mount and decrypt the LUKS volume resides in a trusted platform module (TPM) optionally also tied to the compliance state of the platform (e.g. a static or flexible PCR policy that measures various aspects of the platform).

IMA (Integrated Measurement Architecture) is an open source framework that provides several important aspects including measurements of select parts of the filesystem, integrated with HW (Hardware) Root of Trust (i.e. TPM 2.0 in this case), appraisal functionality that provides a local policy enforcement function to prevent access to file system entities (executables, binary, data) that fails previously mentioned integrity checks, and lastly an extended verification mechanism that also provides protection against file system meta-data and offline attacks.

SELinux (Security Enhanced Linux) builds on top of the DACL access control mechanisms defined natively and establishes more fine-grained access control through policy files.

All of the abovementioned software components would ideally be enabled, provided and supported natively by the OSV (Operating System Vendor). The Docker security hardening is leveraging the above.
In order for the IT/OT enterprise backend to verify attestation, the device management function of the platform is presumed to exist as well, and integrated with necessary OS provided infrastructure to collect and sign measurements by the a trusted platform module for backend verification.

3.2 Security in Transit

It is critical that communication between the device and the infrastructure backend is secure. This is done using standard TLS 1.x protocol. Traditionally the credentials (i.e. the private key corresponding to a certificate) are stored on the filesystem, sometimes encrypted and with access control mechanism provided by the filesystem. To provide additional security here, the private key resides in a trusted platform module, which makes disclosure of this sensitive asset more difficult for adversaries. Docker infrastructure integrates with a TPM and provides the following functionality and characteristics:

- Generation of public private key pair inside a TPM
• The private key never leaves a TPM
• The certificate signing request (CSR) flow by the device to the backend is integrated with a TPM
• Authentication flow when establishing the TLS connection is offloaded to a trusted platform module (TPM) (e.g., signing operations etc.) through the TLS/SSL library.

Note that with a deployment that ensures a TPM backed certificate is issued to client nodes, this much more tightly couples node identity to authentication against a Docker Trusted Registry than the use of filesystem-stored credentials would. Trusted platform module-backed credentials can be used similarly by the Docker engine TLS server. First verifying the certificate against a trusted CA makes it so any remote client communicating with a Docker engine over TLS will have more assurance of that host’s identity, and therefore, security of their communication.

3.3 Security at Rest

To provide security of data at rest, encrypted volumes using LUKS (Linux* Unified Key Service) is commonly used. A credential is generated and typically stored on the filesystem, relying on the protection of that key by the authorization and access control functionality provided natively by the filesystem on Linux*. Sensitive data and code from a container when at rest can be stored on an encrypted LUKS volume to prevent disclosure and access by an authorized subject.

To provide additional hardening of an encrypted volume based on LUKS, the credential needed to access (decrypt and mount) the volume is stored on a trusted platform module (TPM).

• Generation of a high entropy symmetric key for the LUKS volume by a TPM to make brute force attacks infeasible.
• Storage of the symmetric disc encryption key inside a TPM.
• Sealing of the key to a TPM which ties it to a specific state of the system based on measurements extended into PCR (Platform Configuration Registers). The encryption key can only be unsealed if the system is in an authorized state (i.e. the set of PCRs measured meets the policy - through a previously defined static or flexible policy).

3.4 Node Integrity

The Docker runtime as well as the containers themselves have to be integrity protected and require a mechanism in place that allows a remote verifier to validate the claimed attestation state of the node (i.e. remote attestation). Runtime measurements of select
parts of the file system is done using IMA (Integrity Measurement Architecture) which will establish a baseline integrity measurement using a TPM of the static and relevant parts of the Docker infrastructure installation on the device. IMA additionally provides a mechanism that prevents loading of components that fails the integrity (appraisal) check on the device. This feature has to be enabled at kernel build time by the OSV. Measurements taken from SRTM (Static Root of Trust Measurements) are extended into the TPM’2 PCRs (0-9) throughout the boot sequence.

- Enablement of IMA measurements at the OS level
- Definition of IMA measurement policies that defines what should be included and excluded from the filesystem (e.g. immutable files included and mutable files such as logs, caches and alike are excluded).
- IMA uses PCR 10 and measures select parts of the file system, starting by extending the values from the boot sequence (PCR 0-9) into PCR 10 as well as the initramfs effectively creating a cryptographic binding to the trust of SRTM and subsequently HW RoT.
- Runtime measurements, including support for software updates on compatible filesystems with i_version
- Optionally the appraisal features and EVM (extended verifications) can be enabled to ensure that files where integrity verification fails cannot be accessed/loaded, and verifications of file properties and attributes.

### 3.5 Image Integrity

The Docker infrastructure also verifies the integrity of the containers on the platform. Using Docker Content Trust [https://docs.docker.com/engine/security/trust/content_trust/](https://docs.docker.com/engine/security/trust/content_trust/), whereby the Docker runtime performs a verification of images signed by Docker Notary [https://docs.docker.com/notary/](https://docs.docker.com/notary/). A container image that is not signed by the appropriate signer, tampered with, not signed recently (freshness), or not whitelisted by a trust policy will not be started, restarted, or even pulled to the host. Since the Docker runtime, its configuration, and the trust policy is protected by IMA policy, the chain of trust from a TPM can be further extended to the container images.
3.6 Node Attestation

The measurements taken by IMA and those collected through the boot sequence have to be made available to a remote verifier that can verify the claimed attestation state of the node. The node sends an attestation report to the verifier that can authenticate and validate the contents of this report.

- The device management service issues an attestation request to the device that may target a specific set of TPM PCRs.
- The node has a collector which requests a TPM to provide a PCR report based on the request, and signs it.
- The attestation protocol flow contains mechanisms to prevent playback of old attestation data through a nonce.
- The attestation response provides details from the boot log and event log that contains information about the specific measurements that were extended into a specific PCR.

3.7 Registry Authentication

As of Docker Enterprise 3.0, Docker can be configured to use credentials stored in a trusted platform module (TPM) to authenticate with Docker Trusted Registry to securely authenticate a device identity. This provides enhanced security as outlined here.

- Credentials are generated and stored in a TPM.
- Client identity is pinned to a TPM and cannot be cloned or copied.
- Coupled with Roles Based Access Control that Docker Trusted Registry provides, fine-grained access control can be defined whereby users, teams and organizations are limited in their interactions with given applications/repositories.

**Figure 3. Registry Authentication**

**Figure 4. Registry Authentication p.2**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>mTLS handshake with Trusted Registry</td>
</tr>
<tr>
<td>1</td>
<td>Certificate forwarded to Control Plane</td>
</tr>
<tr>
<td>2</td>
<td>Authentication and authorization</td>
</tr>
<tr>
<td>3</td>
<td>Encryption and authorization</td>
</tr>
<tr>
<td>4</td>
<td>Encrypted image sent to node</td>
</tr>
</tbody>
</table>
3.8 SELinux Integration

SELinux can be enabled in the Docker daemon, allowing for the definition of policies that specify processes, actions, and targets that are whitelisted to interact with each other.

To enable, SELinux must be enabled on the host itself and then the following configuration is to be specified in daemon.json.

"selinux-enabled": true

This will label containers to restrict their access to the host filesystem and further restrict any files written by containers to their own category isolated from all other processes.

3.9 Hardened Configuration

Other security hardening of interest here is to eliminate the potentially insecure operations of the Docker infrastructure that may be permissible in some deployment of for sandboxed debug/pre-production deployments.

4.0  Conclusion

Docker and Intel are offering a comprehensive end to end security posture, backed by the most secure hardware mechanisms available. Users, operators, owners, admins, and developers can make use of this security technology stack on their Intel® based boards by leveraging Docker through building, sharing, and running Docker containerized applications.

https://success.docker.com/article/security-best-practices