USER GUIDE

Intel Corporation

Secure the Network Infrastructure - Secure Cloud Native Network Platforms User Guide

Authors

Timothy E. Knoll

David Lu

Haidong Xia

Heqing Zhu

1 Introduction

This document provides instructions on how to use Intel® Security Libraries for Data Center (Intel® SecL - DC) to set up end-to-end platform security for Kubernetes* (K8s*) clusters on an Intel network and edge platform. This technical guide contains step by step details for configuring, installing, and using Intel® SecL-DC on 2nd generation Intel® Xeon® Scalable processors (formerly codenamed Cascade Lake).

The platform attestation solution described in this guide can help telecom operators, enterprise vendors, and solution vendors build out a security-hardened infrastructure that has a seamless integration with Kubernetes.

This guide is a companion document to the *Secure the Network Infrastructure - Secure Cloud Native Network Platforms Solution Brief.*

This document is part of the Network Transformation Experience Kit, which is available at https://networkbuilders.intel.com/network-technologies/network-transformation-exp-kits.

Table of Contents

Figures

Tables

1.1 Terminology

Table 1. Terminology

ABBREVIATION	DESCRIPTION
BIOS	Basic Input / Output System
CRDs	Custom Resource Definitions
IA	Intel® Architecture
Intel® HT Technology	Intel® Hyper-Threading Technology
Intel® SecL - DC	Intel® Security Libraries for Data Center
Intel® TXT	Intel® Trusted Execution Technology
Intel® VT-d	Intel® Virtualization Technology (Intel® VT) for Directed I/O
Intel® VT-x	Intel® Virtualization Technology (Intel® VT) for IA-32, Intel® 64 and Intel® Architecture
K8s	Kubernetes*
NCCoE	National Cybersecurity Center of Excellence
NIST	National Institute of Standards and Technology
NUMA	Non-Uniform Memory Access
PCR	Platform Configuration Register
REST	Representational State Transfer
TPM	Trusted Platform Module
UEFI	Unified Extensible Firmware Interface

1.2 Reference Documentation

Table 2. Reference Documents

REFERENCE	SOURCE
Intel® Security Libraries for Data Center (Intel® SecL-DC)	https://01.org/intel-secl
Intel® SecL-DC Product Guide v1.5	https://01.org/sites/default/files/documentation/intelr_secl-dc_v1.5_product_guide_0.pdf
Intel® SecL-DC Quick Start Guide v1.5	https://01.org/sites/default/files/documentation/intelr_secl-dc_v1.5_quick_start_guide_0.pdf
Container Bare Metal for 2nd Generation Intel® Xeon® Scalable Processor Reference Architecture	https://builders.intel.com/docs/networkbuilders/container-bare-metal-for-2nd-generation-intel-xeon-scalable-processor.pdf
Secure the Network Infrastructure - Secure Cloud Native Network Platforms Solution Brief	https://builders.intel.com/docs/networkbuilders/secure-the-network-infrastructure-secure-cloud-native-network-platforms-solution-brief.pdf
NIST/NCCoE:5G CyberSecurity, Preparing a Secure Evolution to 5G	https://www.nccoe.nist.gov/sites/default/files/library/project-descriptions/5G-pse-project-description-draft.pdf

2 Overview

Containers are increasingly important for NFV in cloud computing. Unlike virtual machines (VMs), containers are lightweight, agile and portable—they can be quickly created, updated and removed. As container adoption increases, there is a corresponding rise in container security problems and solutions. One of the key requirements for container security is to secure the infrastructure.

Typically, the orchestration mechanism has a security solution to place the container workload into a security-hardened platform. This guide discusses how to use Intel® SecL-DC to provide end-to-end platform security for Kubernetes* provisioned containers.

Intel® SecL-DC is a collection of software applications and development libraries to help turn platform security features into real-world security use cases. Intel® SecL-DC provides the following software components:

- Verification service: Installed in the central control node, the verification service gathers the secure boot signatures and maintains the platform's *trusted* or *untrusted* evaluation results. It maintains the platform trust database with established "known good" values or expected measurements (which are called *flavors* in Intel® SecL – DC). If a certain firmware or Linux* kernel are found to be compromised, the policy here can change the platform trust status.

- Trust agent: Installed in every physical node that needs to be monitored by the platform security, the trust agent takes ownership of the Trusted Platform Module (TPM) on the platform and reports the platform security status to the remote management module.
- Integration hub: Connects with orchestration software such as Kubernetes or OpenStack* and contains an extension to work with K8s. The integration hub retrieves the information from the verification service and shares it with the orchestration software at a specified interval.

3 Topology

The setup described in this user guide uses five server platforms, as shown in Figure 1. One system is reserved for the K8s master node, and one system is reserved for cloud imaging servers. Worker nodes 1 and 2 are used to present the distributed network/edge systems. The worker nodes must install the Trust Agent software, which talks to TPM on the worker nodes. One server is dedicated to the "Intel@ Secl_DC" service, which runs the verification service (working with all worker nodes) and the integration hub service (working with the K8s master server).

Figure 1 shows that the end-to-end platform security service is connected via out-of-band networking. In a real-world deployment, the security service can use either in-band or out-of-band networking.

For reliability and security considerations, Intel recommends that one Intel® SecL-DC server verifies <1000 servers.

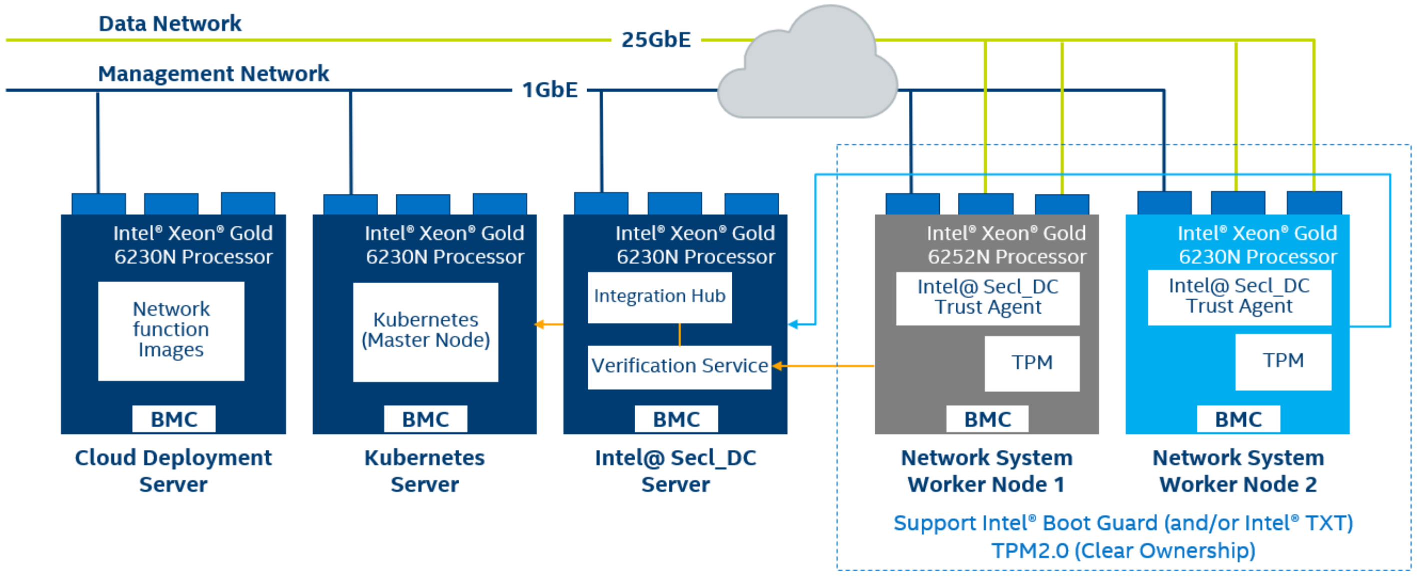
Data Network
25GbE
Management Network
1GbE
Intel® Xeon® Gold 6230N Processor
Network function Images
BMC
Cloud Deployment Server
Intel® Xeon® Gold 6230N Processor
Kubernetes (Master Node)
BMC
Kubernetes Server
Intel® Xeon® Gold 6230N Processor
Integration Hub
Verification Service
BMC
Intel@ Secl_DC Server
Intel® Xeon® Gold 6252N Processor
Intel@ Secl_DC Trust Agent
TPM
BMC
Network System Worker Node 1
Intel® Xeon® Gold 6230N Processor
Intel@ Secl_DC Trust Agent
TPM
BMC
Network System Worker Node 2
Support Intel® Boot Guard (and/or Intel® TXT)
TPM2.0 (Clear Ownership)

Figure 1. Topology Diagram

4 Ingredients

The following sections list the hardware and software bill of materials used in this release.

4.1 Hardware Bill of Materials

This section lists the hardware components and systems used in the deployment. This setup features 2nd Generation Intel® Xeon® Scalable processors, which have a scalable, open architecture designed for the convergence of key workloads such as applications and services, control plane processing, high-performance packet processing, and signal processing.

Table 3. Hardware BOM

ITEM	DESCRIPTION	NOTES
Platform	Intel® Xeon® Processor Scalable Family	Intel® Xeon® processor-based dual-processor server board 2 x 25 GbE integrated LAN ports
Processors	2x Intel® Xeon® Gold 5218N Processor	16 cores, 32 threads, 2.3 GHz, 105 W, 38.5 MB L3 total cache per processor, 3 UPI Links, DDR4-2666, 6 memory channels
	2x Intel® Xeon® Gold 6230N Processor	20 cores, 40 threads, 2.0 GHz, 125 W, 27.5 MB L3 cache per processor, 3 UPI Links, DDR4-2666, 6 memory channels
	2x Intel® Xeon® Gold 6252N Processor	24 cores, 48 threads, 2.3 GHz, 150 W, 22 MB L3 cache per processor, 3 UPI Links, DDR4-2666, 6 memory channels

ITEM	DESCRIPTION	NOTES
Memory	192GB (12 x 16GB 2666MHz DDR RDIMM) or minimum all 6 memory channels populated (1 DPC) to achieve 384 GB	192GB to 384GB
Networking	2 x NICs - Required Each NIC Non-Uniform Memory Access (NUMA) aligned	2 x Dual Port 25GbE Intel® Ethernet Network Adapter XXV710 SFP28+
		2 x Dual Port 10GbE Intel® Ethernet Converged Network Adapter X710
		2 x Intel® Ethernet Server Adapter X520-DA2 SFP
Local Storage	2 x >=480GB Intel® SSD SATA or Equivalent Boot Drive. This is for the primary Boot / OS storage. These drives can be sourced by the PCH. These drives should be capable of use in a RAID1 configuration.	2 x Intel® NVMe P4510 Series 2.0TB each Drive recommended NUMA aligned - Required
Boot Guard	Intel® Boot Guard	
TPM	TPM 2.0	
BIOS	Intel Corporation SE5C620.86 B.0D.01.0241 Release Date: 11/19/2018	Intel® Hyper-Threading Technology (Intel® HT Technology) enabled Intel® Virtualization Technology (Intel® VT-x) enabled Intel® Virtualization Technology for Directed I/O (Intel® VT-d) enabled
Switches	Cisco* Catalyst 2960-XR Arista* DCS-7280QR-C36-R	Cisco 1GbE Switch Arista 25GbE Switch

4.2 Software Bill of Materials

Table 4. Software BOM

SOFTWARE FUNCTION	SOFTWARE COMPONENT	LOCATION
Host OS	Red Hat Enterprise* Linux (RHEL*) 7.6	http://www.redhat.com
Intel® SecL - DC	Intel® Security Libraries for Data Center v1.5	https://github.com/intel-secl
REST* API utility	Postman*	https://www.getpostman.com/
Bare Metal Reference Architecture 2.0 Ansible Playbook	Master Playbook v1.0	https://github.com/intel/container-experience-kits

5 System Prerequisites

5.1 Master and Worker BIOS Prerequisites

Enter the BIOS menu and update the configuration as follows:

PACKAGE	MASTER/WORKER
Intel® VT-x	Enabled
Intel® HT Technology	Enabled
Intel® VT-d (SR-IOV)	Enabled
TPM	Enabled
UEFI Secure Boot	Enabled

5.2 Check Platform Security Technology Configurations

Intel® SecL-DC supports a wide range of secure boot options and complies with the trusted computing secure boot stack. Figure 2 shows the list of options Intel recommends for different platform security technologies.

There are four recommended configurable options depending on customer requirements. Platforms may use different types of processor technology, and having a secure boot stack can help mitigate any platform incompatibilities. When a secure boot stack is implemented, platform integrity can be measured starting from the boot phase, which is the first step in the Hardware Root of Trust, and extending into the software.

The deployment described in this guide uses the first option in Figure 2, which uses Intel® Boot Guard and UEFI Secure Boot. It was selected due to its ease of use and broad ecosystem acceptance.

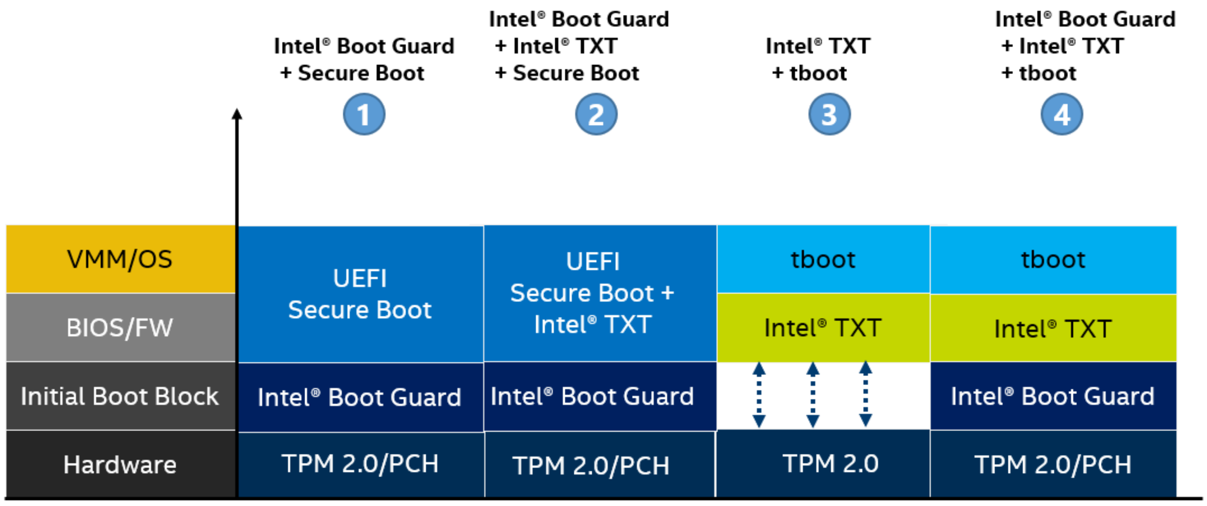

Figure 2. Recommended Platform Security Technology Configurations

5.2.1 Check Intel® Boot Guard Status

The following commands check if Intel® Boot Guard is enabled in your server system.

Note: Intel® Boot Guard is a hardware platform feature. It cannot be changed with software/firmware.

```
# git clone --depth=1 https://review.coreboot.org/coreboot
# cd util/intelmetool/
# make
# modprobe msr
# ./intelmetool –b
```

5.2.2 Check TPM Status

TPM is a must-have for any worker node, which is the server platform that installs the “Trust Agent” module of Intel® SecL-DC. During the boot phase, TPM records the platform integrity status, which can be retrieved later by the “Trust Agent” module.

```
# dmesg | grep -i tpm
[0.000000] ACPI: SSDT 000000006855c000 0044A (v02 INTEL  Tpm2Tabl 00001000 INTL 20140828)
[0.000000] ACPI: TPM2 000000006855b000 00034 (v04 INTEL  S2600WF  00000002 INTL 20091013)
[2.969478] tpm_tis IFX0762:00: 2.0 TPM (device-id 0x1B, rev-id 16)
```

5.2.3 Check UEFI and Secure Boot

1. Check for UEFI Boot:

```
# efibootmgr
```

2. Check if UEFI Secure Boot is enabled:

```
# mokutil --sb-state
SecureBoot enabled
```

5.3 Kubernetes Cluster Setup

The deployment in this guide uses a Kubernetes cluster, however, details for deploying the Kubernetes cluster are not discussed in this document. You can find more details in this document: Container Bare Metal for 2nd Generation Intel® Xeon® Scalable Processor Reference Architecture and this repository: https://github.com/intel/container-experience-kits

5.4 REST* API Utility

A Representational State Transfer (REST) API utility such as cURL* or Postman* is required to execute API requests. This deployment uses Postman because it has a good GUI interface.

6 Deployment Details

This section describes the following tasks:

- Download and Build the Open Source Software Package
- Install Verification Service on the Intel® SecL-DC Server Node
- Install the Trust Agent at each Worker Node
- Connect Trust Agent to Verification Service
- Install the Integration Hub
- Set Up the Kubernetes* Master Node
- Launch a Pod on a Trusted Worker Node

6.1 Download and Build the Open Source Software Package

1. Prepare the build environment:

```
# subscription-manager repos --disable=* --enable=rhel-7-server-rpms --enable=rhel-7-server-
extras-rpms --enable=rhel-7-server-optional-rpms
# yum install epel-release
# yum install wget git zip unzip ant gcc patch gcc-c++ trousers-devel openssl-devel java-1.8.0-
openjdk.x86_64 rpm-build deltarpm makeself

# curl -O https://dl.google.com/go/go1.12.7.linux-amd64.tar.gz
# tar -C /usr/local -xvzf go1.12.7.linux-amd64.tar.gz
# export GOPATH=<path_to_go>
# export PATH=$GOPATH/bin:$PATH

# wget https://www-us.apache.org/dist/maven/maven-3/3.6.1/binaries/apache-maven-3.6.1-
bin.tar.gz
# tar -xf apache-maven-3.6.1-bin.tar.gz
```

2. Download it from the website:

```
# wget https://01.org/sites/default/files/documentation/auto-scripts-intel_secl-dc_0.zip
# unzip auto-scripts-intel_secl-dc_0.zip
# cd BuildScripts/
```

3. Add the following profile element under the `<profiles>` section in `apache-maven-3.6.1/conf/settings.xml`:

```
       <profile>
            <id>artifacts</id>
            <repositories>
            <repository>
                <id>mulesoft-releases</id>
                <name>MuleSoft Repository</name>
                <url>http://repository.mulesoft.org/releases/</url>
                <layout>default</layout>
            </repository>
            <repository>
                <id>maven-central</id>
                <snapshots><enabled>false</enabled></snapshots>
                <url>http://central.maven.org/maven2</url>
            </repository>
            </repositories>
        </profile>
```

4. Enable `<activeProfiles>` to include the above profile:

```
        <activeProfiles>
        <activeProfile>artifacts</activeProfile>
        </activeProfiles>
```

5. Build the source code to get the binary:

```
# chmod +x isecl_bootstrap.sh
# ./isecl_bootstrap.sh -a -r repos_v15.conf
```

6. After a successful build, you should see the packages listed below:
   - **trust agent:** `` `trustagent/packages/trustagent-linux/target/` ``
   - **verification service:** `` `verification-service/packages/host-verification-service-linux/target/` ``
   - **integration hub:** `` `attestation-hub/packages/attestation-hub/target/` ``
   - **k8s-extensions:** `` `k8s-extensions/out/` ``

6.2 Install Verification Service on the Intel® SecL-DC Server Node

The verification service can be installed on a physical server or a virtual machine. In this guide, a server node is used. This server has administrative and security control privileges, so a certain level of security protection is required. The verification service performs platform integrity monitoring by acting as a remote attestation authority. The verification service attests whether the trust agent (running in the worker nodes) is booted into a known-good state. The detailed installation steps can be found in the Intel® SecL-DC Quick Start Guide.

To install the verification service, follow these steps:

1. Copy the verification service installation binary to the /root/ directory.
2. Create the `mtwilson.env` installation answer file and execute the installer binary :
```
#cat mtwilson.env
### User credentials
export MC_FIRST_USERNAME=administrator
export MC_FIRST_PASSWORD=password

###Database Configuration
export DATABASE_USERNAME=root
export DATABASE_PASSWORD=password

### Service IP or Hostname definition
export MTWILSON_SERVER=XX.XX.XX.XX

# ./host-verification-service-linux-4.5-SNAPSHOT.bin
```
3. Validate the installation status. As shown below, `mtwilson` is the name of the verification service.
```
# mtwilson status
Host Verification Service is running
```

6.3 Install the Trust Agent at each Worker Node

The Intel®-SecL integration hub is used to “push” security attributes from attestation reports into orchestration and cloud management services, such as OpenStack* or Kubernetes. The cloud orchestrators can then use the attributes in conjunction with workload policies to schedule protected workloads only on compliant hardware.

The next step is to install the Intel® SecL-DC trust agent on each worker node in the Kubernetes cluster. The trust agent resides on the worker node and takes the ownership of the TPM on the worker nodes, which allows secure attestation quotes to be sent to the verification service. The trust agent reports on platform integrity measurements.

Each worker node must have TPM installed. It is highly recommended that each worker node is set up with the Hardware Root of Trust configuration (Intel® Boot Guard and UEFI secure boot). The detailed “Trust Agent” installation steps are available in the Intel® SecL-DC Quick Start Guide.

To install the trust agent for Linux:

1. Copy the Trust Agent installer binary to the /root/ directory.
2. Create the `trustagent.env` answer file in the /root/ directory:
```
# cat trustagent.env
MTWILSON_API_URL=https://XX.XX.XX.XX:8443/mtwilson/v2
MTWILSON_TLS_CERT_SHA384=610d58e344611747bf3377e0ad9f155d5ece9242004dc1048c430e47e817cfdac5f1c5
1410e35c93056b959455bad58c
MTWILSON_API_USERNAME=administrator
MTWILSON_API_PASSWORD=password
REGISTER_TPM_PASSWORD=y
TRUSTAGENT_LOGIN_REGISTER=true
PROVISION_ATTESTATION=y
CURRENT_IP=XX.XX.XX.XX
TRUSTAGENT_ADMIN_USERNAME=tagentadmin
TRUSTAGENT_ADMIN_PASSWORD=password
TPM_OWNER_SECRET=5d3044b5fb98899f4887af6d7e82874b4024faba
```
3. Execute the trust agent installer and wait for the installation to complete:
```
# ./trustagent-linux-4.5-SNAPSHOT.bin
```
4. After installation is complete, the worker node must be rebooted so that the actual platform integrity measurements can be performed and extended to the TPM.

6.4 Connect Trust Agent to Verification Service

After installing the trust agent on the worker nodes, you must connect the worker nodes (trust agent) with the server that has the “verification service” running. This step is called a *registration procedure*.

To complete the registration, create a host (worker node) record with connectivity details in the verification service database. The host record will be used by the verification service, retrieving an attestation report from the trust agent, which takes TPM attestation quotes. Each trust agent host is known as a worker node, which must be registered with a separate call under Postman, as shown in the example below:

```
POST https://verification.service.com:8443/mtwilson/v2/hosts
{
   "host_name": "<hostname of host to be registered>",
   "tls_policy_id" : "TRUST_FIRST_CERTIFICATE",
   "connection_string": "https://trustagent.server.com:1443;u=tagentadmin;p=password",
   "flavorgroup_name" : "",
   "description" : "<description>"
}
```

6.4.1 Import Flavors to Verification Service

Intel® SecL-DC uses the term *flavor* to represent a whitelist of expected platform measurements, defined in a JSON* format. The *flavor* can include different platform components represented by TPM Platform Configuration Register (PCR) values. Technically speaking, only the HOST_UNIQUE part must come from each host. The PLATFORM and OS flavors can be created one time per version. For example, if all of your trust agent based hosts use BIOS v1.23, you only need to import the PLATFORM flavor for BIOS version 1.23 one time, and all other hosts using the same BIOS version will be matched to the same flavor.

For simplicity, you can import all three flavor parts (PLATFORM, OS, and HOST_UNIQUE) from each worker node using one call under Postman, as shown below:

```
POST https://verification.server.com:8443/mtwilson/v2/flavors
{
   "connection_string": "https://trustagent.server.com:1443;u=tagentadmin;p=password",
   "partial_flavor_types": ["PLATFORM", "OS", "HOST_UNIQUE"],
   "flavorgroup_name": "",
   "tls_policy_id":"TRUST_FIRST_CERTIFICATE"
}
```

6.4.2 Import SOFTWARE Flavor to Verification Service

The SOFTWARE flavor part includes measurements for all the Linux trust agent static files and folders. The SOFTWARE flavor part should be created separately. Only one default SOFTWARE flavor part needs to be created for each version of the trust agent.

To import the SOFTWARE flavor part from a host:

```
POST https://verification.server.com:8443/mtwilson/v2/flavors
{
   "connection_string": "https://trustagent.server.com:1443;u=tagentadmin;p=password",
   "partial_flavor_types": ["SOFTWARE"],
   "flavorgroup_name": "",
   "tls_policy_id":"TRUST_FIRST_CERTIFICATE"
}
```

6.5 Install the Integration Hub

The integration hub integrates Intel® SecL–DC with Kubernetes. It can assign Kubernetes-based environment configurations to specific tenants and it handles pushing the required attributes to Kubernetes.

To install the integration hub, follow these steps:

1. Copy the `integration` hub software binary to the /root/ directory.
2. Create the `attestation-hub.env` file in /root/:

```
# cat attestation-hub.env
MTWILSON_API_URL="https://10.250.250.170:8443/mtwilson/v2"
MTWILSON_SERVER=10.250.250.170
MTWILSON_TLS=610d58e344611747bf3377e0ad9f155d5ece9242004dc1048c430e47e817cfdac5f1c51410e35c9305
6b959455bad58c
MTWILSON_USERNAME=administrator
MTWILSON_PASSWORD=password
ATTESTATION_HUB_DB_NAME="attestation_hub_pu"
ATTESTATION_HUB_DB_HOSTNAME="localhost"
ATTESTATION_HUB_DB_PORTNUM="5432"
ATTESTATION_HUB_DB_DRIVER="org.postgresql.Driver"
ATTESTATION_HUB_DB_USERNAME=root
ATTESTATION_HUB_DB_PASSWORD=password
ATTESTATION_HUB_PORT_HTTP=19082
ATTESTATION_HUB_PORT_HTTPS=19445
```

3. Execute the installer binary:
   ```
   # ./attestation-hub-4.5-SNAPSHOT.bin
   ```
4. Create an administrative user (requires root permission):
   ```
   # attestation-hub password hubadmin password --permissions *:*
   ```

A new hub user (`hubadmin`) is created, who can issue RESTful API requests to the integration hub.

6.6 Set Up the Kubernetes* Master Node

Intel® SecL-DC uses Custom Resource Definitions (CRDs) to expand security attributes to increase Kubernetes capabilties. CRDs allow Kubernetes administrators to configure pods with certain security attributes, which enable the Kubernetes master node to schedule those pods (a set of workload instances) to run on the trusted worker nodes or untrusted node. The specified security attributes are matched to indicate whether a node is trusted or untrusted.

The following sections describe steps to be performed on the Kubernetes master node.

6.6.1 Configure the K8s Master Node

1. Add a mount path to the `kube-scheduler.yaml` file for the Intel® SecL-DC scheduler extension:
   ```
   -  mountPath: /opt/isecl-k8s-extensions/bin/
      name: extendedsched
      readOnly: true
   ```
2. Add a volume path to the `kube-scheduler.yaml` file for the Intel® SecL-DC scheduler extension:
   ```
   -  hostPath:
        path: /opt/isecl-k8s-extensions/bin/
        type: “”
      name: extendedsched
   ```
3. Copy the `isecl-k8s-extensions.bin` installer to the Kubernetes master node and execute the installer:
   ```
   # ./isecl-k8s-extensions.bin
   ```
4. The installer outputs a set of keystores upon completion, which are located in the `attestation-hub-keystores` folder. These keys will be used by the integration hub to communicate with the K8s master node. This directory must be copied to the integration hub running in the other server:
   ```
   # scp -r /root/attestation-hub-keystores/* root@integration-hub.server.com:/opt/attestation-
   hub/configuration/
   ```
   Note: The integration hub can work with multiple K8s master environments at the same time, but the keystores must be kept separate between them. To do this, Intel recommends that you create subfolders in the integration hub configuration directory for separate Kubernetes environments, and copy the appropriate keystores to the matching subfolder.
5. Create the secret channel between the hub and master node, and copy the integration hub public key to the Kubernetes master node:
   ```
   # scp attestation-hub.server.com:/opt/attestation-hub/configuration/hub_public_key.pem
   /etc/kubernetes/pki/
   ```
6. Run the command `systemctl restart kubelet` to restart all the Kubernetes control plane container services, including the scheduler. When completed, the security attributes are expected to be in effect.
7. Verify that the Intel® SecL-DC Custom Resource Definitions have been made ready on the Kubernetes master node:
   ```
   # kubectl get crds
   # kubectl get -o json hostattributes.crd.isecl.intel.com
   ```

6.6.2 Configure the Integration Hub for K8s

The integration hub can work with multiple types of orchestration software. Within the hub service module, a tenant defines the connection and its authentication details to work with a specific K8s master node. At least one tenant must be created here. As described previously, it is possible that multiple tenants can be created to work with multiple K8s master nodes.

```
POST https://hub.server.com:19445/v1/tenants
{
    "name": "DemoTenant",
    "plugins": [
        {
            "name": "kubernetes",
            "properties": [
                {
                    "key": "api.endpoint",
                    "value": "https://kubernetes-master.server.com:6443"
                },
                {
                    "key": "tenant.name",
                    "value": "DemoTenant"
                },
                {
                    "key": "plugin.provider",
```

```
                    "value": "com.intel.attestationhub.plugin.kubernetes.KubernetesPluginImpl"
                },
                {
                    "key": "kubernetes.client.keystore",
                    "value": "/opt/attestation-hub/configuration/root_k8s_client.jks"
                },
                {
                    "key": "kubernetes.server.keystore",
                    "value": "/opt/attestation-hub/configuration/root_k8s_trust.jks"
                },
                {
                    "key": "kubernetes.server.keystore.password",
                    "value": "<Keystore password>"
                },
                {
                    "key": "kubernetes.client.keystore.password",
                    "value": "<Keystore Password>"
                }
            ]
        }
    ]
}
```

Note: The `kubernetes.client.keystore` and `kubernetes.server.keystore` values must be the filesystem path on the integration hub that contains the K8s master keystores output from the scheduler extensions for this tenant. The `kubernetes.server.keystore.password` and `kubernetes.server.keystore.password` values must be the keystore passwords output by the scheduler extensions installer.

6.6.3 Assign Worker Nodes to Tenant in Integration Hub

The distributed deployed network nodes are referred as the *worker nodes* in this setup. They must be assigned to a tenant in the integration hub, so that Intel® SecL-DC can decide which platform's security attributes need be pushed to a specific K8s master node. Multiple worker nodes can be assigned to a tenant in a single request by using a comma-separated list of `hardware_uuids`.

```
POST https://hub.server.com:19445/v1/host-assignments
{
   "tenant_id": "<Tenant ID>",
   "hardware_uuids":  [
          "<Host 1 Hardware UUID>", “<Host 2 Hardware UUID>”
          ]
}
```

6.6.4 Verify the Integration Hub Working Status

The integration hub periodically queries the verification service to retrieve the list of all new reports. Only the reports generated after the timestamp of a recent query will be returned. The list of hosts known to the integration hub can be retrieved using the following sample API:

```
GET https://hub.server.com:19445/v1/hosts
```

This command returns a list of all the worker nodes seen by the integration hub with their most recent report status.

By default, the hub will:

- Poll the verification service for new reports every 2 minutes.
- Refresh the list.
- Send updates to all tenants associated with the working nodes.

6.7 Launch a Pod on a Trusted Worker Node

1. From the K8s master node, run the following command to verify the worker nodes status. The *trusted* status is expected to be true, after the worker node has been attested successfully.

```
# kubectl label node --list --all
Listing labels for Node./master:
 node-role.kubernetes.io/master=
 beta.kubernetes.io/arch=amd64
 beta.kubernetes.io/os=linux
 kubernetes.io/arch=amd64
 kubernetes.io/hostname=master
 kubernetes.io/os=linux
Listing labels for Node./test1:
 node.alpha.kubernetes-incubator.io/nfd-rdt-RDTMON=true
 node.alpha.kubernetes-incubator.io/node-feature-discovery.version=v0.3.0
 kubernetes.io/arch=amd64
```

```
……………………………………………
 node.alpha.kubernetes-incubator.io/nfd-rdt-RDTL3CA=true
 isecl.trusted=true
……………………………………………
```

2. K8s orchestration can deploy the pod on the worker node with `isecl.trusted=true`. This can be done by creating a configuration file called `sample.yaml`, which must contain the **matchExpressions** policy for platform security.

```
apiVersion: v1
kind: Pod
metadata:
name: samplepod
spec:
affinity:
 nodeAffinity:
  requiredDuringSchedulingIgnoredDuringExecution:
   nodeSelectorTerms:
    - matchExpressions:
       -
       key: isecl.trusted
       operator: In
       values:
         - "true"
containers:
 -
  image: nginx
  imagePullPolicy: IfNotPresent
  name: nginx
```

3. Run the following command from the master node to launch a pod:

```
# kubectl create -f sample.yaml
```

Finally, you can list the running pods to confirm that the pod launched and is running on a secured worker node.

7 Summary

The working group National Cybersecurity Center of Excellence (NCCoE) is part of the National Institute of Standards and Technology (NIST). The working group has released a 5G Cybersecurity project paper, which recommends that “The supporting infrastructure will utilize hardware roots of trust for platform measurement and attestation to ensure that certain workloads run on hardware in a good known state and within a well-defined logical boundary.” This security guidance highlights the importance of platform attestation, which is a foundational requirement to secure the 5G infrastructure.

This guide contains detailed steps for installing, configuring, and using Intel® SecL-DC, a platform attestation solution that can help telecom operators, enterprise vendors, and solution vendors build out an end-to-end platform security-hardened infrastructure that has a seamless integration with Kubernetes. Intel® SecL-DC is an open source software project. It provides a complete platform attestation solution reference together with Intel hardware Root of Trust, TPM, and Secure Boot technologies.

Intel is committed to innovate with its global partners and use open source software to create easy, affordable and secure solutions and address key adoption barriers with optimized costs in mind. Intel is committed to secure the cloud native networking platform and uncover the power of Intel Architecture-based servers with hardened platform security.

0420/DN/PTI/PDF 621828-001US