INTEL® SECURITY LIBRARIES
LEVEL 200 TRAINING
Legal Notices and Disclaimers

Intel technologies’ features and benefits depend on system configuration and may require enabled hardware, software, or service activation. Learn more at intel.com, or from the OEM or retailer.

No computer system can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.

Cost reduction scenarios described are intended as examples of how a given Intel®-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications, and roadmaps.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel does not control or audit third-party benchmark data or the websites referenced in this document. You should visit the referenced website and confirm whether referenced data are accurate.

Copyright © 2019 Intel Corporation. Intel, the Intel logo, Intel. Experience What’s Inside, the Intel. Experience What’s Inside logo, Intel Atom, Intel Core, Intel Inside, the Intel Inside logo, Quark, and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others.
Course Prerequisites

Intended Audience: Technical Field Engineers who will work with potential ISVs or CSPs to integrate and enable Intel Datacenter Security features

Attendees should complete the ISecL-100 level training class before attending this class

Attendees should have at least a general understanding of Datacenter and Cloud concepts
Level 200 Course Content

1) ISecL – Why and What
   - Motivation for ISecL
   - Intel® Si Security Technologies
   - Library Integration vs Service Deployments

2) ISecL Service Deployments
   - Architecture
   - Supported Use Cases
     - Asset Tagging
     - Platform Attestation
     - Integrations with 3rd Party Services
     - Sample Integration Workflows
   - Pre-requisites
   - Manageability – Activation & Provisioning
   - Flavor Deepdive
   - Access Controls

3) ISecL Library Integration
   - Library Functional Overview
   - Sample Usages
INTEL® SECURITY LIBRARIES (ISECL)
WHAT AND WHY?
SECURITY CONTINUES TO BE #1 BARRIER FOR CLOUD ADOPTION

CLOUD ADOPTION BARRIERS

#1 General security risks
33%

#2 Lack of staff resources or expertise
28%

#3 Integration with existing IT environments
27%

#4 Data loss & leakage risks
26%

#5 Legal & regulatory compliance
24%

*Data from Cloud Research Partners

MAIN CLOUD SECURITY CONCERNS

Data loss/leakage
57%

Data privacy
49%

Confidentiality
47%

Legal and regulatory compliance
36%

Data sovereignty/control
30%
# Data Center Security Drivers & Customers Asks

## Key Drivers
- Increased multi-cloud adoption
- More regulatory controls (GDPR, HIPPA, PCI)
- Advanced cyber threats attacks
- Increased distributed data and intelligence

## Key Asks
- Platform supply chain integrity
- Platform resilience
- Platform integrity assurance
- Data protection & sovereignty
- Protect keys
- Visibility, controls & compliance
INTEL® SECURITY LIBRARIES (ISECL) SERVICE DEPLOYMENTS
Core Trusted Cloud Use-cases – Summary

**Boot Integrity based on Platform Trust**
- Trust Chain: HW->FW->BIOS->OS/VMM

**Workload Location and Boundary Control**
- Control Trusted VM/Container/App launch and migration
- Boundary can be: Geo Info, QoS, SLA, Compliance Attributes

---

Exposing and Utilizing Intel Si Security Features

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others. Copyright © 2018, Intel Corporation.*
Measurement extended to TPM Platform Configuration Registers

Measurement Phase 1 (H/W + BIOS)
- uCode evals BIOS ACM
- BIOS (evals BIOS init code)
- BIOS Option ROMs
- Option ROMs & other non-critical modules

Measurement Phase II (TBOOT, OS, Docker Engine...)
- uCode (evals SINIT ACM)
- SINIT ACM (measures OS Kernel, initrd, Tboot-xm agent in initrd) measures Docker Engine, other components

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others. Copyright © 2018, Intel Corporation.
TPM PCRs Usage (Measurements)

**BIOS**
- PCR0 – CRTM, BIOS/FW code, and Host Platform Extensions
- PCR1 – Host Platform Configuration
- PCR2 – Adapter Option ROM Code
- PCR3 – Adapter Option ROM Configuration & Data
- PCR4 – IPL Code (usually the MBR)
- PCR5 – IPL Code Configuration and Data
- PCR6 – State Transition and Wake Events
- PCR17 – Dynamic Root of Trust Measurement

**OS/VMM**
- PCR18 – Measured Launch Environment (TBOOT)
- PCR19-22 – Trusted OS/VMM (kernel code, settings, other)

Integrity of PCRs1-7 depend on integrity of PCR0 (root of trust)
Integrity of PCR18 depends on integrity of PCR17 (DRTM)
Integrity of PCR19-22 depend on integrity of PCR18 (OS/VMM)

Per TCG PC Specification

*Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others.* Copyright © 2018, Intel Corporation.
• What is ISecL?
  - Intel® ISecL is a set of software libraries and components that expose and enable Intel security features. The libraries are used by ISecL services to secure platforms and protect data.

• ISecL Architecture
  - Library-centric model to expose & utilize Intel security features.
  - Includes components and services to enable key Cloud security Use Cases.
  -Aligned with platform schedules starting Cascade Lake.
  - Extensible “Flavor” based model for managing trusted configurations.
  - Support for RHEL*, Microsoft Windows* Server, VMWare* vSphere.
  - Allows for Turn Key Solutions.
  - Easy & Customized Solutions.
  - Easier Upgrades.
  - Improved Code Organization.

*Other names and brands may be claimed as the property of others. Copyright © 2018, Intel Corporation.
Level 200 Course Content

1) ISecL – Why and What
   ▪ Motivation for ISecL
   ▪ Intel® Si Security Technologies
   ▪ Library Integration vs Service Deployments

2) ISecL Service Deployments
   ▪ Architecture
   ▪ Supported Use Cases
     – Asset Tagging
     – Platform Attestation
     – Integrations with 3rd Party Services
     – Sample Integration Workflows
   ▪ Pre-requisites
   ▪ Manageability – Activation & Provisioning
   ▪ Flavor Deepdive
   ▪ Access Controls

3) ISecL Library Integration
   ▪ Library Functional Overview
   ▪ Sample Usages
Intel® Security Libraries

Platform Trust, Trusted Compute Pools

- Uses TXT/TPM to verify the integrity of a platform (BIOS, OS, VMM) against a “known good state” or “Flavor” at boot time

1. System powers on and Intel® TXT verifies system BIOS/firmware
2. Hypervisor measure matches
3. OS & applications are launched

- Hardware-based Geo and Asset Tags help control workload placement and migration

Trusted Location and Boundary Control

Intel® TXT
Intel® Trusted Execution Technology

Intel and the Intel logo are trademarks of Intel Corporation in the U. S. and/or other countries. *Other names and brands may be claimed as the property of others. Copyright © 2018, Intel Corporation.
Intel® Security Libraries v1

- Platform Integrity Assurance
- Asset Tagging for Data Sovereignty
- Integration for OpenStack* & K8S*
- Intel's End to End Solution
- Customer Turn Key Solution

Notes: Intel® architecture (IA)
Intel® Trusted Execution Technology (Intel® TXT)
ISecL - Prerequisites

Hardware Requirements (Attestation clients only)

- ISecL requires server hardware with an Intel® Xeon® or Intel® Xeon® Scalable Family processor that supports Intel® Trusted Execution Technology (Intel® TXT)
- ISecL requires server hardware that has a Trusted Platform Module (version 1.2 or 2.0) installed, and provisioned for use with Intel® Trusted Execution Technology (Intel® TXT), according to Trusted Compute Group specifications. If a version 2.0 TPM will be used, the SHA256 PCR bank must be enabled.
- TPM and Intel® TXT must be enabled in the BIOS
- Before installation, the TPM ownership must be cleared

Supported Operating Systems

- VMWare* vSphere 6.5 Update 2 or Higher (ESXi hosts must be managed by vCenter Server)
- Red Hat Enterprise Linux* 7.4 or Higher
- Microsoft Windows* Server Datacenter 2016 or Higher
TCG – Style Remote Attestation Model

1. 160 bit Nonce, NC
2. TPM Quote Request (NC, PCR list)
3. TPM Quote Response (Sig, PCR, NC)
4. \{ Sig (PCR, NC), SML, AIKcert \}
5. Integrity Verification
   a. Ver (Sig (PCR, NC), AIK) = true / false
   b. Compare (PCR, SML == Golden Measurements)

Intel and the Intel logo are trademarks of Intel Corporation in the U. S. and/or other countries.
*Other names and brands may be claimed as the property of others. Copyright © 2018, Intel Corporation.
Intel® SecL-DC: Geo/Asset-tagging – Enabling Boundary Control

- Geo/Asset descriptor (asset-tag) stored in the TPM of the Server
- Used to control placement & migration of workloads
- Broad support across bare metal OS and hypervisors (ESXi*, Windows* Server, KVM*)

**WHAT IS ASSET TAG?**

![Diagram of asset tag creation process]

- **TAG** + **UUID of Host** + **Digital Signature** = **Asset Certificate**
- **Asset Certificate** + **NVRAM Index** = **Asset Tag**

**NV Index**
- Index used: index 0x40000010
- Size of Index:
  - TPM 1.2: 20 Bytes
  - TPM 2.0 (future): 32 Bytes (for 256)
    64 Bytes (for 512)
- Data Format: 20 Bytes of Binary data

Asset Tag used with Geo-Location attributes is a Geo-Tag
ISecL – Host Management API

This API manages which hosts are added to the VS database.

CREATE Host:
POST https://server.com:8443/mtwilson/v2/hosts
Input : { "host_name" : "RHEL-Host", "tls_policy_id" : "TRUST_FIRST_CERTIFICATE", "connection_string" : "intel:https://tagent.host.com:1443;u=tagent-admin;p=password", "flavorgroup_name" : "mtwilson_automatic", "description" : "sample TAgent host" }

DELETE Host:
ISecL – Flavor Management API

This API manages Flavors, which are the definition objects that determine expected characteristics of Trusted hosts. When a host does not match the set of Flavors required by the Flavorgroup, that host is “Untrusted.”

**IMPORT Flavors from Host:**
POST https://server.com:8443/mtwilson/v2/flavors
Input: { "connection_string": "intel:https://tagent.server.com:1443;u=tagent-admin;p=password","partial_flavor_types": ["PLATFORM","OS"],"flavorgroup_name": "mtwilson_automatic"}

**SEARCH Flavors:**
GET https://server.com:8443/mtwilson/v2/flavors?key=bios_name&value=Dell Inc

**DELETE Flavor:**
ISeCL – Report Management API

This API is used to generate and query attestation reports, which will identify whether a given host is Trusted, any Asset Tags applied, etc. Untrusted reports will show “faults” that describe why the host is untrusted.

The VS automatically generates new reports for hosts as the existing hosts expire; manually creating a new report is generally not needed, and will have a significant delay due to TPM response times.

The “Accept” header determines the format of the response; “application: samlassertion+xml” returns the most secure report with all signatures intact, but will not show specific measurement values or faults; “Application: xml” or “application: json” will return a report without signatures, but including all of the Flavor information, individual measurement values, and any Faults.

CREATE a new Report (new TPM quote):
POST https://server.com:8443/mtwilson/v2/reports
input: {"host_name":"host-1"} OR {"hardware_uuid": "6c6a9034-2708-4815-90f9-d227efb45a3b"} OR {"host_id": "6c6a9034-2708-4815-90f9-d227efb45a3b"}

SEARCH for Existing Reports:
https://server.com:8443/mtwilson/v2/reports?latestPerHost=true
Returns the most recent Report for each registered host

https://server.com:8443/mtwilson/v2/reports?hostname=host.server.com&numberOfDays=30&latestPerHost=false
Returns all Reports generated for the specified host over the last 30 days
ISecl – hostStatus API

This API is used to query the current “status” of hosts. This API does not return whether the host is “Trusted” (that would be the “reports” API), but will show which hosts are actually connected, which are in the queue, which have an error state, etc.

GET https://server.com:8443/mtwilson/v2/host-status?hostStatus=CONNECTED
Returns all connected hosts (“Connected” means a host that is actually communicating with the VS, not only added to the database)

GET https://server.com:8443/mtwilson/v2/host-status?latestPerHost=true
Returns the current status of all registered hosts

GET https://server.com:8443/mtwilson/v2/host-status?hostStatus=QUEUE
Returns all hosts currently in the QUEUE state (typically this means the host is being matched to Flavors, or an attestation report is being generated)

GET https://server.com:8443/mtwilson/v2/host-status?hostname=host.server.com&latestPerHost=false&numberOfDays=30
Returns the full list of all state changes of a specified host over the last 30 days.
ISecl – Sample Workflow
Register and Attest Hosts using Automatic Flavor Matching

IMPORT Flavors from Host:
POST https://server.com:8443/mtwilson/v2/flavors
Input: { "connection_string": "intel:https://tagent.server.com:1443;u=tagent-admin;p=password","partial_flavor_types": ["PLATFORM","OS"],"flavorgroup_name": "mtwilson_automatic"}

Be sure to use the “mtwilson_automatic” Flavorgroup for automatic matching. Note that each individual host will still need to have the HOST_UNIQUE Flavor part imported (except Windows* hosts, which have no HOST_UNIQUE part). Import OS and BIOS Flavors for all OS and BIOS versions intended to be Trusted in your datacenter.

CREATE Host:
POST https://server.com:8443/mtwilson/v2/hosts
Input : { "host_name" : "RHEL-Host", "tls_policy_id" : "TRUST_FIRST_CERTIFICATE", "connection_string" : "intel:https://tagent.host.com:1443;u=tagent-admin;p=password", "flavorgroup_name" : "mtwilson_automatic", "description" : "sample TAgent host" } 
Register each host.

SEARCH for Existing Reports:
https://server.com:8443/mtwilson/v2/reports?latestPerHost=true
Reports are automatically generated as registered Hosts are matched to created Flavors. This request will return the current Reports for all attested Hosts.
I SecL – Sample Workflow

Register and Attest Hosts using Host-Based Flavors

**IMPORT Flavors from Host:**

POST https://server.com:8443/mtwilson/v2/flavors

Input: {
   "connection_string": "intel:https://tagent.server.com:1443;u=tagent-admin;p=password",
   "partial_flavor_types": ["PLATFORM", "OS"],
   "flavorgroup_name": "flavor_group_hostname"
}

Instead of using the mtwilson_automatic Flavorgroup, we create a new custom Flavorgroup using the host’s hostname. By adding Flavors and registering Hosts to host-specific Flavorgroups, we create a direct mapping for Hosts to Flavors.

**CREATE Host:**

POST https://server.com:8443/mtwilson/v2/hosts

Input: {
   "host_name": "RHEL-Host",
   "tls_policy_id": "TRUST_FIRST_CERTIFICATE",
   "connection_string": "intel:https://tagent.host.com:1443;u=tagent-admin;p=password",
   "flavorgroup_name": "flavor_group_hostname",
   "description": "sample TAgent host"
}

Register each host. Again note that we use the Flavorgroup name based on the host’s hostname.

**SEARCH for Existing Reports:**

https://server.com:8443/mtwilson/v2/reports?latestPerHost=true

Reports are automatically generated as registered Hosts are matched to created Flavors. This request will return the current Reports for all attested Hosts.
ISecL – Sample Workflow
Asset Tags and Data Sovereignty

Prerequisite: Hosts are already Registered, and attest as Trusted with OS/BIOS/HOST_UNIQUE Flavors

1) Create Asset Tag
POST https://verification.server.com:8443/mtwilson/v2/tag-certificates
{"hardware_uuid": "<hardware UUID of host to be tagged>",
 "selection_content": [ 
 { 
     "name": "<key>",
     "value": "<value>"
 },
 { 
     "name": "<key>",
     "value": "<value>"
 },
 ...
 ]
}

1) Provision Asset Tag (Linux*/Windows*)
POST https://verification.server.com:8443/mtwilson/v2/rpc/deploy-tag-certificate
{
 "certificate_id": "<certificate ID>",
 "host": "<Hostname of host to be tagged>"
}
ISecL – Sample Workflow
Provisioning Asset Tags for ESXi*

1) After creating the Asset Tag certificate, create the ASSET_TAG Flavor (this is automatic during the Provisioning process for Linux* and Windows*, but is manual for ESXi*)

```plaintext
POST https://verification.server.com:8443/mtwilson/v2/flavors
{  "connection_string": "<connection string to vCenter specifying the host>",
  "tls_policy_id": "TRUST_FIRST_CERTIFICATE",
  "partial_flavor_types": ["ASSET_TAG"]
}
```

2) The ASSET_TAG Flavor creation request will return the Asset Tag certificate content. Use this content to calculate the Tag hash using the following:

```plaintext
cat <Certificate content> | base64 --decode | openssl dgst -sha1 | awk -F " " '{print $2}
```

3) Write the Asset Tag hash to the TPM
   1) Enable SSH on the host from vCenter*
   2) Connect to the ESXi host to be tagged via SSH
   3) Run the following command:
       ```plaintext
esxcli hardware tpm set -d <hash value>
```

To verify the Tag content in the TPM:
```plaintext
esxcli hardware tpm get
```

4) Disable SSH and reboot the ESXi host. The Tag value will be measured and extended to PCR22 during boot, and will be included as part of the attestation process
SAML Assertions

Available for every registered host

XML format “Security Assertion Markup Language”

Provide PLATFORM, OS, HOST_UNIQUE, ASSET_TAG, and Overall trust status determined by Verification Service

Provide BIOS and OS details (name, version) when trusted

Signed by Verification Service private SAML signing key

Can be verified using Verification Service SAML Certificate
TPM Measurements
Measurement Event Log

When a server boots using the TCG Measured Boot process, different system components are individually hashed, and those hashes are extended to a set of PCRs. Multiple measurements may extend to a single PCR.

During the remote attestation process, in some cases the raw PCR value is directly compared to a corresponding Flavor.

For several measurements, however, a measurement event log is created that details, in order, all of the individual hash measurements taken and extended to the PCR.

In the case of a PCR that contains a measurement event log, ISecL will verify the integrity of the log by replaying all of the measurements and ensuring that they “add up” to the actual PCR value reported by the TPM. The PCR value is not actually contained in the Flavor.

If the measurement event log replay matches the parent PCR value, the log integrity is considered to be verified. Then, each measurement event is compared to a corresponding measurement event expected value in the Flavor.

If the log integrity is verified and all of the measurement events match expectations, the host is considered Trusted. If either of these conditions is not true, the host will be considered Untrusted, and an attestation report will contain Faults detailing which measurements were wrong/missing.
Flavor Deepdive
Flavor Definition

A “Flavor” is a set of expectations that will be used to determine whether a given server is “Trusted.”

Flavors contain descriptive metadata defining the applicability of the Flavor, any version information, and then a set of expected attributes. The Flavor is then compared to an actual report from a host to determine whether the host report matches the requirements specified in the Flavor.

Typically these expectations include TPM PCR values, or specific measurement event values.
Flavor Deepdive (Cont’d)
Flavor Parts

Flavors are made up of different “Flavor Parts.”

- **OS Flavor**
  - An OS Flavor defines expectations regarding the Operating System of a given host. Typically this includes measurements extended by tboot or a Windows* boot driver to PCRs 17, 18, 19, and/or 20 in a TPM.

- **PLATFORM Flavor**
  - A PLATFORM Flavor defines expectations regarding platform-level measurements, including the BIOS. These measurements are typically extended to PCRs 0, 17, and/or 18.

- **HOST_UNIQUE Flavor**
  - VMWare* ESXi and Red Had Enterprise Linux* hosts each have specific measurement events that will always be unique for every host, and so cannot be shared across all hosts using the same OS or BIOS versions (in the way that PLATFORM and OS Flavors can). For these host types, the HOST_UNIQUE Flavor must be imported individually, for each host.

- **ASSET_TAG Flavor**
  - The ASSET_TAG Flavor is used for attestation of Asset Tags. ASSET_TAG Flavors are unique to specific host, since all Asset Tag certificates are tied to the host by the hardware UUID.
Flavor Deepdive (Cont’d)
Flavorgroups

“Flavorgroups” are logical collections of Flavors and Hosts, with “Flavorgroup Policies” that define which Flavor parts are required and how Flavors will be matched to Hosts.

A Flavor can only be used to attest a Host if both the Flavor and the Host are in the same Flavorgroup.
Flavor Deepdive (Cont’d)
Flavor Group Policies

Flavor Match Policies are used to define how the Flavor Match engine will match Flavors to hosts for attestation for a given Flavor Group. Each Flavor part can have defined Flavor Match Policies within a given Flavor Group.

That is:
“PLATFORM”: { “any_of”, “required” }

“OS”: { “all_of”, “required_if_defined” }

“HOST_UNIQUE”: {“latest”,“required_if_defined”}

“ASSET_TAG”:{“latest”,“required_if_defined”}

The sample Policy above would require that a PLATFORM Flavor part be matched, but any Bios Flavor part in the Flavor Group may be matched.

The OS Flavor Part will only be required if there is an OS Flavor part in the Flavor Group; if there are no OS Flavor parts in the Group, the match will not be required. If more than one OS Flavor part exists in the Group, all of those OS parts will be required to match for a host to be Trusted.
Flavor Deepdive (Cont’d)
Flavorgroup Policies

**ANY_OF**

The ANY_OF Policy allows any Flavor of the specified Flavor part to be matched. If the Flavor Group contains OS Flavor 1 and OS Flavor 2, a host will be Trusted if it matches either OS Flavor 1 or OS Flavor 2.

**ALL_OF**

The ALL_OF Policy requires all Flavors of the specified Flavor Part in the Flavor Group to be matched. For example, if Flavor Group X contains PLATFORM Flavor Part 1 and PLATFORM Flavor Part 2, a host in Flavor Group X will need to match both PLATFORM Flavor 1 and PLATFORM Flavor 2 to attest as Trusted. If the host matches only one of the Flavors, or neither of them, the host will be attested as Untrusted.

**LATEST**

The LATEST Policy requires that the most recently created Flavor of the specified Flavor part be used when matching to a host. For example:

```
“ASSET_TAG”: { “latest”, “required_if_defined” }
```

ASSET_TAG Flavor parts by default use the previous Policy. This means that if Asset Tag Flavors are in the Flavor Group, the most recently created Asset Tag Flavor will be used. If no Asset Tag Flavors are present in the Flavor Group, then this Flavor part will be ignored.
Flavor Deepdive (Cont’d)

Flavorgroup Policies

**REQUIRED**

The REQUIRED Policy requires a Flavor of the specified part to be matched. For example:

```json
"PLATFORM": { "any_of", "required" }
```

This policy means that a PLATFORM Flavor part must be used; if the Flavor Group contains no PLATFORM Flavor parts, hosts in this Flavor Group will always count as Untrusted.

**REQUIRED_IF_DEFINED**

The REQUIRED_IF_DEFINED Policy requires that a Flavor part be used if a Flavor of that part exists. If no Flavor part of this type exists in the Flavor Group, the Flavor part will not be required.

```json
"ASSET_TAG": { "latest", "required_if_defined" }
```

ASSET_TAG Flavor parts by default use the previous Policy. This means that if Asset Tag Flavors are in the Flavor Group, the most recently created Asset Tag Flavor will be used. If no Asset Tag Flavors are present in the Flavor Group, then this Flavor part will be ignored.
Flavor Deepdive (Cont’d)

By default the Verification Service includes a Flavor Group named “mtwilson_automatic” and another named “mtwilson_unique.” During host registration, the “mtwilson_automatic” Flavor Group is used as a default selection if no other Flavor Group is specified.

**mtwilson_automatic**

The mtwilson_automatic Flavor Group is used as the default Flavor Group for all hosts and all Flavor parts. If no other Flavor Groups are specified when creating Flavors or Hosts, all Hosts and Flavors will be added to this group. This is useful for datacenters that want to manage a single set of acceptable configurations for all hosts.

The mtwilson_automatic Flagorgroup uses the following Flavorgroup Policies:

```
“PLATFORM”: { “any_of”, “required” }
“OS”: { “all_of”, “required_if_defined” }
“HOST_UNIQUE”: {“latest”,“required_if_defined”}
“ASSET_TAG”:{“latest”,“required_if_defined”}
```

**mtwilson_unique**

The mtwilson_unique Flavor Group is used to contain the HOST_UNIQUE Flavor parts.
I SecL- Boundary Control With OpenStack*

1. Upload Workload A to Glance with Launch Policy
2. Launch VM A
3. Workload A
   - Launch Policy
   - Trust Verified. Geo=France
4. Placement Service
5. Request Location Attestation
   - Attestation Report
6. Workload A launched with appropriate policy

Intel® SecL-DC

OOB: Provision Geo-Tag on to Server TPMs

Intel® recommends following published Openstack* security best practices. Intel and the Intel logo are trademarks of Intel Corporation in the U. S. and/or other countries. *Other names and brands may be claimed as the property of others. Copyright © 2018, Intel Corporation.
Smart Scheduling – Integrating Trust to Schedulers

Principles of Operation

- Cluster Manager determines best hosts in the cluster, based on utilization, type, location compliance, etc.
- (For this host list) Cluster Manager verifies Host Integrity with the Attestation Authority
- Attestation Authority responds with Attestation Reports for the Hosts
- Cluster Manager picks best Server that has the Integrity and instantiates Containers

Examples of schedulers
- OpenStack
- Kubernetes

Scheduler/Cluster Manager

ISeCL Plugins

Image Registry

Containersized NFV Apps

Agents

OS, Initrd

TPM

Agents

OS, Initrd

TPM

Attestation Authority

Trust Not Verified.
Smart Orchestration/Scheduling – Plugin Based ‘Push’ Model

REST APIs

- SysLog/Custom Plug-in
- OpenStack Plug-in
- K8s Plug-in

ISecl Integration Hub

- ACL: hosts allowed for each tenant & Scheduler
- Get host trust status

ISecl Verification Service

 COMPUTE NODES

- Windows
- RHEL
- ESXi
- RHEL

Schedule VMs/Containers

Store host trust status

OpenStack Scheduler Tenant A

OpenStack Scheduler Tenant B

K8s Scheduler

SIEM/Consoles

*Other names and brands may be claimed as the property of others. Copyright © 2018, Intel Corporation.
Smart Orchestration with Kubernetes*

1. ISeCL Integration Hub pushes Reports info to Master.

2. Trust Plugin assigns nodeSelector labels based off the ISeCL host attestation Reports.

3. Kubernetes scheduler launches containers on hosts where nodeSelector matches pod configuration.

pod configuration
apiVersion: v1
kind: Pod
metadata:
  name: app1
spec:
  containers:
    - name: app1
      image: "ubuntu:14.04"
      nodeSelector:
        ISeCL - Trusted: true

Intel and the Intel logo are trademarks of Intel Corporation in the U. S. and/or other countries. *Other names and brands may be claimed as the property of others. Copyright © 2018, Intel Corporation.
Components in TPM 2.0 Solution Stack – Vertical View

Algorithm Support Currently Available

<table>
<thead>
<tr>
<th>Platform</th>
<th>Purley (Current scope)</th>
<th>Purley-R (Target SMX support)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBoot</td>
<td>SHA1, SHA256, SM3-256</td>
<td>SHA1, SHA256, SM3-256</td>
</tr>
<tr>
<td>BIOS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRB</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>PCSD</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>OEM</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>TPM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTT</td>
<td>SHA1, SHA256</td>
<td>SHA1, SHA256, SMx (RCR)</td>
</tr>
<tr>
<td>dTPM (nationZ)</td>
<td>SHA1, SHA256, SMx</td>
<td>SHA1, SHA2, SMx</td>
</tr>
<tr>
<td>CPU ACM</td>
<td>SHA1, SHA256, SHA384,</td>
<td>SHA1, SHA256, SHA384,</td>
</tr>
<tr>
<td></td>
<td>SHA512, SM3-256 SM2</td>
<td>SHA512, SM3-256 SM2</td>
</tr>
<tr>
<td></td>
<td>(in plan)</td>
<td></td>
</tr>
</tbody>
</table>

Current Status
- TSS and TPM tools ready in open source (Q1’16)
- ESXi* support for TPM2.0 only partially implemented (ISecL supports VMWare* with TPM 1.2 only)

SHA1/SHA256 in Purley

Intel and the Intel logo are trademarks of Intel Corporation in the U. S. and/or other countries.
*Other names and brands may be claimed as the property of others. Copyright © 2018, Intel Corporation.
Server Procurement - Check for Intel® TXT supported platforms from all OEMs. Make sure you pick a SKU that has TPM or order it as optional module – IMPORTANT!!

Check if TPM is provided. Else, you will need to order TPM and retrofit using OEM specific instructions. Ex. Shown is for Cisco* To avoid this step, make sure Step 1 is correctly done.

To avoid this step, make sure Step 1 is correctly done.

For Linux* distribution, install TBOOT, modify Grub config to do measured launch with Intel® TXT in OS, and validate activation is successful

Activate Intel® TXT and TPM in BIOS and validate activation is successful
Intel® TXT and TPM Management now made easier with **One-Touch Activation**

- **Remote**
- **Rapid**
- **Provisioning**

**CLI**

**GUI**

**IPMI**

**TXT/TPM/PTT**

- **Discover**
- **Activate**
- **Deactivate**
- **Clear**

**Scale out**

**OEM Independent**

DC Admin (Remote)
Level 200 Course Content

1) ISecL – Why and What
   - Motivation for ISecL
   - Intel® Si Security Technologies
   - Library Integration vs Service Deployments

2) ISecL Service Deployments
   - Architecture
   - Supported Use Cases
     - Asset Tagging
     - Platform Attestation
     - Integrations with 3rd Party Services
     - Sample Integration Workflows
   - Pre-requisites
   - Manageability – Activation & Provisioning
   - Flavor Deepdive
   - Access Controls

3) ISecL Library Integration
   - Library Functional Overview
   - Sample Usages
Authentication for API Requests

By default, API requests are authenticated via HTTP Basic authentication

Shiro* IP whitelisting can also be used to allow all requests from a specific IP address

1) Edit /opt/<service>/configuration/shiro.ini

2) Uncomment the following line, and edit to include only those IPs or hostnames from which to allow all requests

```
iniHostRealm.allow=127.0.0.1
```

*WARNING* - No authentication at all will be used for requests that come from an IP listed when this configuration is used. No granularity of permissions is possible. Use with caution.
Authentication for CLI

CLI commands (mtwilson restart; tagent restart; etc) can be executed by the Root user by default.
API Access Controls
Permissions

A “Permission” is a specific action within a privilege domain

hosts:create allows a user to make POST requests to the /v2/hosts resource

hosts:* allows a user to use all supported methods for the /v2/hosts resource

See the Javadoc descriptions for each API request and method for information on required permissions
## API Access Controls (Cont’d)

### Roles

<table>
<thead>
<tr>
<th>Role</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>trustagent_provisioner</td>
<td>host_aiks:certify</td>
</tr>
<tr>
<td></td>
<td>tpm_endorsements:create</td>
</tr>
<tr>
<td></td>
<td>tpm_endorsements:search</td>
</tr>
<tr>
<td></td>
<td>tpm_passwords:create</td>
</tr>
<tr>
<td></td>
<td>tpm_passwords:retrieve</td>
</tr>
<tr>
<td></td>
<td>tpm_passwords:search</td>
</tr>
<tr>
<td></td>
<td>tpm_passwords:store</td>
</tr>
<tr>
<td></td>
<td>tpm_passwords:store</td>
</tr>
<tr>
<td></td>
<td>host_signing_key_certificates:create</td>
</tr>
<tr>
<td></td>
<td>store_host_pre_registration_details:create</td>
</tr>
<tr>
<td>administrator</td>
<td><em>:</em></td>
</tr>
<tr>
<td>auditor</td>
<td>*:*search,retrieve</td>
</tr>
<tr>
<td>asset_tag_manager</td>
<td>tag_certificate_requests:*</td>
</tr>
<tr>
<td></td>
<td>tag_selection_kv_attributes:*</td>
</tr>
<tr>
<td></td>
<td>tag_certificates:*</td>
</tr>
<tr>
<td></td>
<td>tag_kv_attributes:*</td>
</tr>
<tr>
<td></td>
<td>tag_selections:*</td>
</tr>
<tr>
<td>flavor_manager</td>
<td>host_tls_policies:create, search, retrieve</td>
</tr>
<tr>
<td></td>
<td>flavorgroup:*</td>
</tr>
<tr>
<td></td>
<td>flavors:*</td>
</tr>
<tr>
<td>host_manager</td>
<td>hosts:*</td>
</tr>
<tr>
<td></td>
<td>host_attestations:*</td>
</tr>
<tr>
<td></td>
<td>host_tls_policies:search,retrieve</td>
</tr>
<tr>
<td></td>
<td>host_status:search,retrieve</td>
</tr>
<tr>
<td>reports_manager</td>
<td>reports:*</td>
</tr>
<tr>
<td>host_unique_flavor_creator</td>
<td>host_unique_flavor:create</td>
</tr>
<tr>
<td>host_based_flavor_creator</td>
<td>host_based_flavor:create</td>
</tr>
</tbody>
</table>

A “Role” is a collection of permissions that can be assigned to a user

Users can create custom Roles

Verification Service installation creates several default Roles
User Creation (API)

1) Create new user request

POST https://server.com:8181/mtwilson/v2/users

Input: { "username" : "Developer1", "locale" : "en-US", "comment" : "Access needed for Project1" }

2) Create user LoginPassword

POST https://server.com:8181/mtwilson/v2/users/<user ID>/login-passwords

Input: { "password_hash" : "RZMrrSt/PvKvdqs10qR0id0bDE0dvF4XbPKV7sF+oDg=", "salt" : "a9gDma0hUF8=", "iterations" : 1, "algorithm" : "SHA256", "comment" : "Access needed for development" }

3) ApproveUserLoginPassword with assigned permissions (requires administrator)

PUT https://server.com:8181/mtwilson/v2/users/<User ID>/login-passwords/<Password ID>

Input: { "status" : "APPROVED", "enabled" : true, "roles" : ["security","whitelist"] }
User Creation (API)

Password creation rules

- **Password Hash**
  This is the Base64-encoded hash of the password using the specified algorithm, concatenated with the salt, over the specified number of iterations
  
  
  "password_hash":"RZMrrSt/PvKvdqs1OgR0id0bDE0dvF4XbPKV7sF+oDg="

- **Salt**
  The "salt" is a base64-encoded string added to the end of the password before hashing
  
  "salt":"a9gDma0hUF8="

- **Iterations**
  The number of iterations defines the number of times the password+salt is hashed to produce the final hash
  
  "iterations":1

- **Algorithm**
  The algorithm used to hash the specified password. Currently only SAH256 is supported
  
  "algorithm":"SHA256"
User Creation (CLI)

Verification Service:

mtwilson login-password <username> <password> --permissions <permission1> <permission2>...

Trust Agent:

tagent password <username> <password> --permission *:*

Integration Hub:

attestation-hub password <username> <password> --permission *:*

NOTE – The Trust Agent and Integration Hub do not have any permission granularity. These services must use administrator access (*:*)
## Library Integration

### Verification Service Libraries

<table>
<thead>
<tr>
<th>Library Name</th>
<th>Library Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>HostConnector</td>
<td>Connects to different types of host (Linux*, Windows*, ESXi*) to retrieve host and TPM quote information</td>
</tr>
<tr>
<td>Flavor</td>
<td>Manages flavors</td>
</tr>
<tr>
<td>Verifier</td>
<td>Verifies if a host is trusted by comparing <code>if host report</code> against a policy</td>
</tr>
<tr>
<td>AssetTagCreator</td>
<td>Generates asset tag and certificate</td>
</tr>
<tr>
<td>AssetTagProvisioner</td>
<td>Provisions the asset tag for managed hosts</td>
</tr>
<tr>
<td>PrivacyCA</td>
<td>Generates and manage certificates for AIK certificate, binding and signing certificates</td>
</tr>
<tr>
<td>SAML generator</td>
<td>Generates attestation reports for host or VM attestation status, either in SAML format</td>
</tr>
</tbody>
</table>
## Library Integration

### Agent Libraries

<table>
<thead>
<tr>
<th>Library Name</th>
<th>Library Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlatformInfo*</td>
<td>Collect host hardware information (Intel® TXT, TPM type or version, BIOS info, etc.) and OS information</td>
</tr>
<tr>
<td>TpmProvider*</td>
<td>Provides a coherent API for basic TPM functions to support host attestation, regardless of TPM version (PTT), or type dTPM, and OS</td>
</tr>
</tbody>
</table>

* Not used for VMWare* ESXi clients
 LIBRARY INTEGRATION SCENARIOS WITH ISECL

Scenario 1: Verification Service + Trust Agent
Platform Integrity & Data Sovereignty for Windows*/RHEL*

Scenario 2: Verification Service + VMWare* vSphere
Platform Integrity & Data Sovereignty for VMWare

Scenario 3: PlatformInfo + TPM Provider
Secure Discovery of Intel Security Features

Scenario 4: Trust Agent + Customer Verification Service
Custom Implementation of Platform Integrity

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others. Copyright © 2018, Intel Corporation.