Trusted Computing Platform Validation Tool

Table of Contents

Introduction

The Trusted Computing Platform Validation Tool is intended to evaluate end-to-end support for trusted computing on a given platform. This tool will automatically perform a series of tests that will determine the readiness of the underlying platform to support Trusted Computing use cases.

This tool should be run on a server with a Trusted Platform Module installed and with support for Trusted Execution Technology. The tool will work for both TPM 1.2 and TPM 2.0 versions.

The Platform Validation Tool will give a definitive "pass" for all Trusted Computing use cases, but is not intended to perform any detailed diagnostic of failures. If a test fails, the logs can be examined to gather useful information about the failure, but may not immediately provide the root cause.

High-Level Design

The Platform Validation tool works by installing Intel Cloud Integrity Technology locally on the server, and then uses CIT-related functions to verify that all of the Trusted Computing functions of the platform are working. The tool works with minimal prerequisite steps and attempts to automate as many steps as is possible.

When executed, the tool will automatically perform the following steps:

1) Reconfigure the server to boot into Trusted Boot mode
2) Install the Intel Cloud Integrity Attestation Service
3) Install the Intel Cloud Integrity Trust Agent
4) Perform a set of tests including generation of an attestation report, generation of an AIK, writing to the TPM NVRAM index, etc.

The process takes approximately 20 minutes to run, and will include several automated reboots. Upon completion, the tool will output the results of the tests:

```
Running BKC Tool for Intel(R) Cloud Integrity Technology...
=================================================>100%
cit-bkc-tool validation complete; run 'cit-bkc-tool report' to see report
txt_support: OK - TXT is supported.
txtstat_present: OK - txt-stat is present.
tpm_support: OK - TPM is supported.
tpm_version: OK - TPM 2.0
tpm_ownership: OK - TPM is owned.
aik_present: OK - AIK certificate exists.
cit_service_up: OK - CIT Attestation Service is running.
cit_agent_up: OK - CIT Trust Agent is running.
create_whitelist: OK - Create whitelist with host registration successful.
write_assettag: OK - AssetTag validated.
nvindex_defined: OK - NV index defined.
host_attestation_status: OK - Host is trusted
consistent_pcr0: OK - PCR 0 is consistent.
```

Tests

The Validation Tool performs several tests. Some of these tests are performed during the actual installation of components, and others are scripted actions that happen before or after an installation step. If all of the tests pass, the underlying server platform is verified as working for Trusted Computing use cases.

Trusted Execution Technology Support

```
txt_support: OK - TXT is supported.
txtstat_present: OK - txt-stat is present.
```

These tests verify that Intel TXT is enabled, and that the host is correctly booting using tboot into a successful trusted launch state. This is done by running the “txt-stat” command, which is installed as part of the “tboot” package.

A failure in one of these two tests indicates that TXT is not currently enabled in the server BIOS, or is otherwise not functioning (TXT support requires support in the processor, chipset, BIOS, and SINIT ACM). It can also indicate that the server is booting to a boot option other than tboot (this should be automatically configured by the tool during installation).

TPM Capabilities

```
tpm_support: OK - TPM is supported.
tpm_version: OK - TPM 2.0
tpm_ownership: OK - TPM is owned.
write_assettag: OK - AssetTag validated.
nvindex_defined: OK - NV index defined.
aik_present: OK - AIK certificate exists.
```

These tests verify various TPM-related functions. The tool will verify that a TPM is present and the TPM driver is loaded, and will check for a supported TPM version (TPM 1.2 or TPM 2.0). During the Trust Agent installation step, the Agent will take ownership of the TPM, and will generate a new Attestation Identity Key pair. Later, the tool will provision a CIT Asset Tag to the server, which will verify that the TPM commands related to defining and writing to an NVRAM index are working. These tests are performed using a combination of commands from the “tpm-tools” package and the installation of the CIT Trust Agent.

A failure in one or more of these tests indicates that the TPM is not correctly responding to TCG-related commands or is an unsupported version. This could be caused by a faulty TPM driver, a TPM that is not activated in the BIOS, a server with no actual TPM installed, or an un-provisioned TPM.

Generate Valid and Consistent TPM Attestation Report

create_whitelist: OK - Create whitelist with host registration successful.
host_attestation_status: OK - Host is trusted
consistent_pcr0: OK - PCR 0 is consistent.

These tests verify that the TPM generates a valid attestation report, including checking to verify the signature using the previously generated Attestation identity Key. The CIT Attestation Service imports the values from an initial attestation report to establish them as the "whitelist" values for future attestations. The tool then triggers a reboot, and performs a new attestation when the server boots back up. This is done to verify the consistency of key measurements (BIOS, OS kernel, etc) between reboots. These tests are performed via REST API requests made to the locally installed CIT Attestation Service.

A failure in one or more of these tests indicates that the attestation reports generated by the TPM are unreliable. This could indicate an Attestation Identity Key signature mismatch, or could indicate that one or more of the measured components in the trusted launch is changing between reboots.

Building the Validation Tool

Preparing the Build Environment

It is recommended to build the Validation Tool on a separate environment. The following procedures were tested on an Ubuntu 14.04 host. Update your proxy settings if it is required.

Installing the Required Packages

To install the required packages, execute the following commands:

```
cd /root

apt-get update

apt-get install ssh ant makeself git make gcc g++ openssl libssl-dev unzip nsis zip -
y
```

Installing the Java Development Kit (JDK)

To install the JDK, follow these steps:

1. Get the JDK from the following URL:
2. http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html#jdk-7u51-oth-JPR
3. Download the jdk-7u51-linux-x64.tar.gz file.
4. Systems usually come with a default Java. You must update at least the ones used by CIT compilation. Run the following commands to set it up correctly on the system:

```
mv jdk-7u51-linux-x64.tar.gz jdk-1.7.0_51-linux-x64.tar.gz
gzip -dc jdk-1.7.0_51-linux-x64.tar.gz | tar xf -
mv jdk1.7.0_51 /usr/lib/jvm #Create the jvm directory if it doesn’t exist
already
cd /etc/alternatives/
ln -s /usr/lib/jvm/jdk1.7.0_51/bin/java #remove them first if they already
exist
ln -s /usr/lib/jvm/jdk1.7.0_51/bin/javac #remove them first if they already
exist
cd /usr/bin/
ln -s /etc/alternatives/java #remove them first if they already exist
ln -s /etc/alternatives/javac #remove them first if they already exist
```

Installing Apache Maven 3.3.3

To install Apache Maven 3.3.3, follow these steps:

1. Download the binary from repository with the following command:

```
wget http://archive.apache.org/dist/maven/maven-3/3.3.1/binaries/apache-maven-
3.3.1-bin.zip
```

2. Unzip the file.

```
unzip apache-maven-3.3.1-bin.zip
```

3. Move the application directory to /usr/local.

```
mv apache-maven-3.3.1 /usr/local
```

Editing setting.xml

If you need proxy settings, do so in the /usr/local/apache-maven-3.3.1/conf/settings.xml file. It is located in the conf directory of your maven app directory:

```
<proxy>
    <id>YourProxyId</id>
    <active>true</active>
    <protocol>http</protocol>
    <host>YourProxyIP</host>
    <port>YourProxyPort</port>
</proxy>
```

Modifying Environment Files

To modify the environment files, follow these steps:

1. Add environment variables to ~/.bashrc.

```
2. JAVA_HOME=/usr/lib/jvm/jdk1.7.0_51
3. export JAVA_HOME PATH=$PATH:$JAVA_HOME
4. export M2_HOME=/usr/local/apache-maven-3.3.1
5. export M2=$M2_HOME/bin
6. export MAVEN_OPTS="-Xmx1024m -XX:MaxPermSize=1024m" PATH=$PATH:$M2
```

7. Restart your session so the environment variables are loaded correctly.

8. Once you login again, verify maven installation:

```
9. mvn -version
```

Downloading the Source Code

The source code must be downloaded at /root. When you clone, the default branch is master. Use this if the latest published code is required.

```
cd /root

git clone https://github.com/opencit/opencit-external-artifacts/

git clone https://github.com/opencit/opencit-util/

git clone https://github.com/opencit/opencit-contrib/

git clone https://github.com/opencit/opencit/
```

Building the Source Code

The source code must be built in the following order:

1. External Artifacts opencit-external-artifacts
2. Util Project opencit-util
3. TPM Patched Tools opencit-contrib
4. Open CIT opencit

Note: External Artifacts requires steps to build see the next section 'Building External Artifacts' You can also see the Readme.md file for more information on how to build this project.

To build the source code, follow these steps:

1. Run ant to build each project.

```
cd /root/opencit-external-artifacts
git checkout release-cit-2.2
ant
cd ..
cd /root/opencit-util
git checkout v1.0+cit-2.2+tc-pvt
ant
cd ..
cd /root/opencit-contrib
git checkout release-cit-2.2
ant
cd ..
cd /root/opencit
git checkout v2.2+tc-pvt
```

```
ant
```

2. Verify that all 3 builds are successful.

   Note: The openCIT APIs are documented in a Javadoc, which is generated automatically at /root/opencit/integration/mtwilson-client-java7/target/mtwilson-client-java7-2.2-SNAPSHOT-javadoc.zip when you run ant to build the code. Or can be generated separated by running ant Javadoc.

Building External Artifacts

To build this project, you must download a couple of artifacts and place them in specific directories so we can help you install to your local maven repo.

1. After you clone the project, **cd** to the root directory of external artifacts.

```
cd opencit-external-artifacts
git checkout release-cit-2.2
```

2. Download each artifact and prepare them to build the opencit-external-artifacts project.

Monit

Execute the following commands:

```
wget https://mmonit.com/monit/dist/monit-5.5.tar.gz
mv monit-5.5.tar.gz monit/monit-5.5-linux-src.tgz
```

Tomcat

Execute the following commands:

```
wget https://archive.apache.org/dist/tomcat/tomcat-7/v7.0.34/bin/apache-tomcat-
7.0.34.tar.gz
mv apache-tomcat-7.0.34.tar.gz apache-tomcat/apache-tomcat-7.0.34.tar.gz
```

Glassfish

Execute the following commands:

```
wget http://download.java.net/glassfish/4.0/release/glassfish-4.0.zip
mv glassfish-4.0.zip glassfish/glassfish-4.0.zip
```

vijava

Execute the following commands:

```
wget
https://sourceforge.net/projects/vijava/files/vijava/VI%20Java%20API%205.5%20Beta/vij
ava55b20130927.zip
unzip vijava55b20130927.zip
mv vijava55b20130927.jar vijava/vijava-5.5.jar
```

JDK

1. Use the JDK downloaded in Section 3.1.2 . If you do not have it, you can download it from:

```
2. http://www.oracle.com/technetwork/java/javase/downloads/java-archive-
   downloads-javase7-521261.html#jdk-7u51-oth-JPR
```

3. Download the jdk-7u51-linux-x64.tar.gz file.
4. Once you have the file, place it in the jdk directory of the opencit-external-artifacts project

```
5. mv jdk-7u51-linux-x64.tar.gz jdk/jdk-1.7.0_51-linux-x64.tar.gz
```

6. Now clean up.

```
7. rm -rf *.zip
8. rm -rf *.jar
```

TPM Tools

Execute the following commands:

```
wget https://sourceforge.net/projects/trousers/files/tpm-tools/1.3.8/tpm-tools-
1.3.8.tar.gz
mv tpm-tools-1.3.8.tar.gz tpm-tools/tpm-tools-1.3.8.tar.gz
```

Build using:

```
ant
```

That’s the end of the build process.

Installation

Prerequisites

- Red Hat Enterprise Linux (RHEL) 7.2 or 7.3
- The Validation Tool must be run on a physical server
- TXT must be activated in the server BIOS
- A TPM version 1.2 or 2.0 must be installed in the server
- The TPM to be activated in the server BIOS
- The following packages are required (these will require a valid RHEL subscription):
    - grub2-efi-modules-2.02-0.44.el7.x86_64.rpm
    - pgdg-redhat93-9.3-2.noarch.rpm
    - epel-release-latest-7.noarch.rpm
    - postgresql93-libs-9.3.14-1PGDG.rhel7.x86_64.rpm
    - postgresql93-9.3.14-1PGDG.rhel7.x86_64.rpm
    - postgresql93-contrib-9.3.14-1PGDG.rhel7.x86_64.rpm
    - postgresql93-server-9.3.14-1PGDG.rhel7.x86_64.rpm
    - wxBase-2.8.12-8.el7.x86_64.rpm
    - wxGTK-2.8.12-8.el7.x86_64.rpm
    - pgadmin3_93-1.22.1-1.rhel7.x86_64.rpm
    - xmlstarlet-1.6.1-1.el7.x86_64.rpm
- Tboot 1.9.4 must be compiled from the latest source code and installed

Installation Steps

Once all of the prerequisites are configured, the Validation Tool can be executed. It will automatically perform all installations and configurations, and then execute the tests. The server will reboot several times during the process.

1) Build the Validation Tool from source
2) Copy the Validation Tool to the /root/ directory
3) Configure any needed proxies for internet access
4) Install prerequisite packages
5) Execute the tool binary

After approximately 20 minutes and several automated reboots, the tool will report back the status of all tests. This will automatically be displayed upon login.

Installing the Prerequisite Packages

The Validation Tool will automatically attempt to install most packages on its own. Some installations and configurations, however, must be done prior to running the tool.

1) Install grub2-efi-modules
2) Create a script named create-grub-efi.sh:

```
GRUB_MODULES="all_video boot btrfs cat chain configfile echo efifwsetup \
efinet ext2 fat font gfxmenu gfxterm gzio halt hfsplus iso9660 \
jpeg loadenv lvm mdraid09 mdraid1x minicmd normal part_apple \
part_msdos part_gpt password_pbkdf2 png reboot search \
search_fs_uuid search_fs_file search_label sleep syslinuxcfg \
test tftp regexp video xfs linuxefi multiboot multiboot2"
grub2-mkimage -O x86_64-efi -o grubx64.efi.new -p /EFI/redhat $GRUB_MODULES
```

3) Make a backup of the existing grubx64.efi

```
cp /boot/efi/EFI/redhat/grubx64.efi /boot/efi/EFI/redhat/grubx64.efi.bak
```

4) Execute the create-grub-efi.sh script

```
chmod 777 create-grub-efi.sh
./create-grub-efi.sh
```

5) Copy the resulting grubx64.efi.new to replace the old grubx64.efi

```
cp grubx64.efi.new /boot/efi/EFI/redhat/grubx64.efi
```

6) Install tboot

```
yum localinstall –y tboot-1.9.4-x86_64.rpm
```

7) (**REQUIRED** for RHEL 7.3; **OPTIONAL** for RHEL 7.2) Install the remaining prerequisite packages.

```
yum localinstall -y pgdg-redhat93-9.3-2.noarch.rpm
yum localinstall -y epel-release-latest-7.noarch.rpm
yum makecache
yum localinstall -y postgresql93-libs-9.3.14-1PGDG.rhel7.x86_64.rpm
yum localinstall -y postgresql93-9.3.14-1PGDG.rhel7.x86_64.rpm
yum localinstall -y postgresql93-contrib-9.3.14-1PGDG.rhel7.x86_64.rpm
yum localinstall -y postgresql93-server-9.3.14-1PGDG.rhel7.x86_64.rpm
yum localinstall -y wxBase-2.8.12-8.el7.x86_64.rpm
yum localinstall -y wxGTK-2.8.12-8.el7.x86_64.rpm
yum localinstall -y pgadmin3_93-1.22.1-1.rhel7.x86_64.rpm
yum localinstall -y xmlstarlet-1.6.1-1.el7.x86_64.rpm
```

8) (**REQUIRED** for RHEL 7.3; **OPTIONAL** for RHEL 7.2) Initialize Postgres

```
/usr/pgsql-9.3/bin/postgresql93-setup initdb
systemctl enable postgresql-9.3
systemctl start postgresql-9.3
ln -s /usr/lib/systemd/system/postgresql-9.3.service
/usr/lib/systemd/system/postgresql.service
```

9) Execute the Validation Tool

Sample test output

```
Running BKC Tool for Intel(R) Cloud Integrity Technology...
==================================================>100%
cit-bkc-tool validation complete; run 'cit-bkc-tool report' to see report
txt_support: OK - TXT is supported.
txtstat_present: OK - txt-stat is present.
tpm_support: OK - TPM is supported.
tpm_version: OK - TPM 2.0
tpm_ownership: OK - TPM is owned.
aik_present: OK - AIK certificate exists.
cit_service_up: OK - CIT Attestation Service is running.
cit_agent_up: OK - CIT Trust Agent is running.
create_whitelist: OK - Create whitelist with host registration successful.
write_assettag: OK - AssetTag validated.
nvindex_defined: OK - NV index defined.
host_attestation_status: OK - Host is trusted
consistent_pcr0: OK - PCR 0 is consistent.
```

Repeating the Tests

The tests performed by the Validation Tool can be re-executed without re-installing.

1) Clear any existing test results using the following command:

```
cit-bkc-tool clear
```

2) Uninstall the CIT Trust Agent

```
tagent uninstall
```

3) Repeat the tests

```
cit-bkc-tool
```

The tool will re-run the tests and report the output. To repeat the display of the results at any time, run the following command:

```
cit-bkc-tool report
```

Logs

Logs for the installations can be found in the following locations:

```
/usr/local/var/cit-bkc-tool/monitor/install_cit_service/stdout
/usr/local/var/cit-bkc-tool/monitor/install_cit_service/stderr
/usr/local/var/cit-bkc-tool/monitor/install_cit_agent/stdout
```

```
/usr/local/var/cit-bkc-tool/monitor/install_cit_agent/stderr
```