Trusted Virtualization Platform Deployment

1. Overview

Two major components exist in our current prototype systems: the *management node*, including the Cloud Controller, Cluster Controller, Walrus and EBS, and the *computing node*, i.e. the Node Controller with the virtualization environment.

Computing Nodes. Both the virtualized environment (the guest OS) and the virtualization environment (the host OS) should be integrated with trusted computing components. In this text we focus on integrating trusted computing with the QEMU-KVM virtualization layer, as it is the currently prevalent virtualization solution and comprehensive trusted computing integration for its counterpart, XEN, has already been released and is not hard to access.

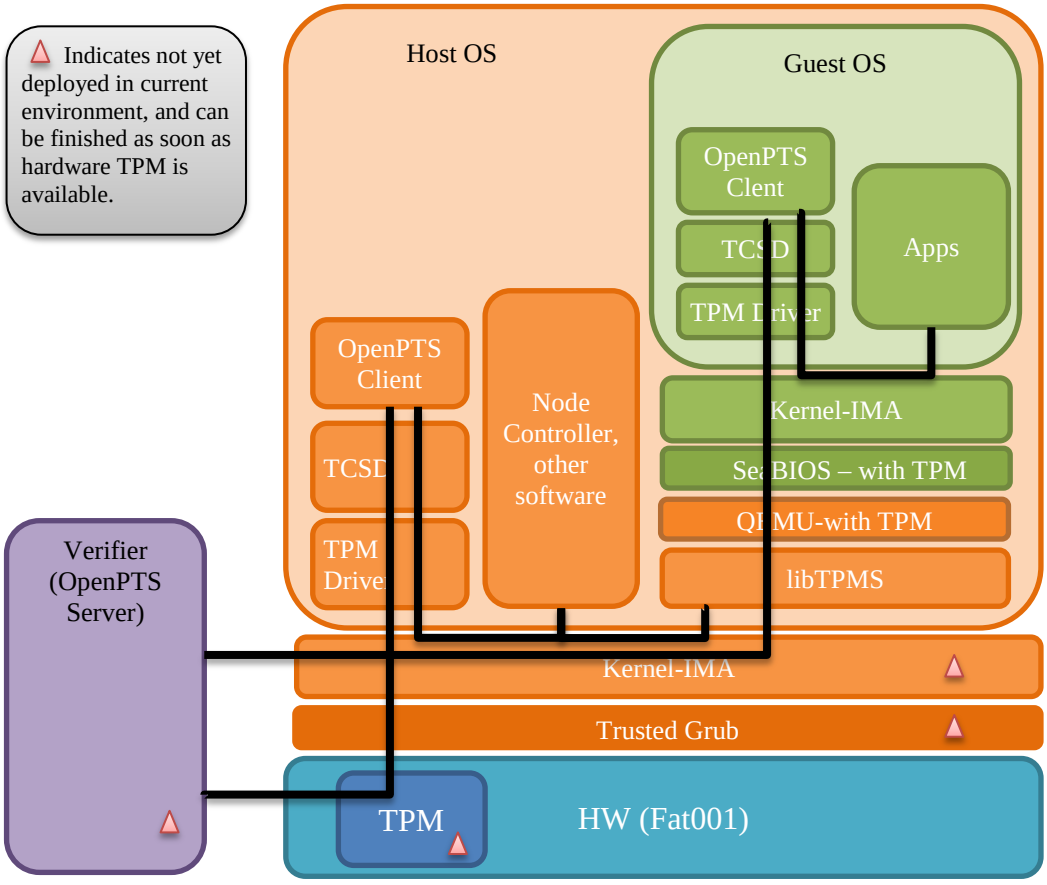

Figure 1. Computing Node

Host OS. From bottom-up, Trusted Grub should be installed to bootstrap the trusted environment. The kernel should be applied with the IBM IMA patches, in order to perform measurement for every loaded applications, kernel modules and possibly configuration files. TPM drivers should also be installed and loaded. TCSD, the

Trusted Core Service daemon, is the major component of the TSS (TCG Software Stack). It exports the trusted services and also the cryptographic algorithms from the TPM to the upper layer applications. The OpenPTS (Open Platform Trust Services) is the software set implementing the attestation procedure. Its client part report the measurements stored in and certified by the TPM to its server part, which is usually hosted in the Verifier Platform, as depicted in the left in Figure 1. The OpenPTS server maintains a trusted measurement list to help examining whether the reported measurements and related SMLs (Stored Measurement Logs) are trustworthy. All the software components loaded on the Host OS, including the Node Controller and the virtualized software (QEMU) are measured and reported.

Guest OS. To support trusted computing in QEMU-KVM virtualization environment, the QEMU should be patched with TPM supports, which contains a backend TPM driver to invoke the functionalities in libTPMs and exports a frontend TPM to the Guest OS. LibTPMs implements emulated TPMs for each virtualized instance. The backend TPM driver in the patched QEMU can also be modified to link to other software TPM emulators or to communicate directly with underlying TPM hardware. SeaBIOS should also be patched with TPM supports. No bootloader is needed in this environment, and the upper layer software components are the same with in the host OS, including the Kernel-IMA patches, TPM drivers, TCSD and the OpenPTS client.

Verification. As in this deployment architecture, no direct communication is made from the software TPM to the hardware TPM, the Guest OS and the Host OS is attested to and verified separately. As depicted in Figure 1, two set of OpenPTS clients are deployed and talk to the OpenPTS server in the verifier respectively. However, as the Guest OS is actually a loaded software component in the Host OS, attestation to the Host OS can also report the trustworthiness of the virtualization software, the libTPMs and the QEMU, and whether the expected images and configuration files are loaded. The internal running state is verified by the attestation to the Guest OS. To implement a one-step attestation, libTPMs can be modified to quote the PCR values in the hardware PCR.

Management Node. The trusted computing integration architecture of the management node is depicted in Figure 2. Most added components are the same as in the Host OS of the computing node, including the Trusted Grub, Kernel-IMA, TCSD and the OpenPTS. The management components, mainly the CLC, CC, Walrus and EBS are measured by Kernel-IMA, and with the measurement values reported by the OpenPTS client. The policies of each of these components are important in this scenario and may result in the damage of the users' benefits, so they should also be attested to. Moreover, as in our use case, access control to important files, mainly the virtual disk images, are also critical. SELinux can be active and with the access control policies attested to by clients.

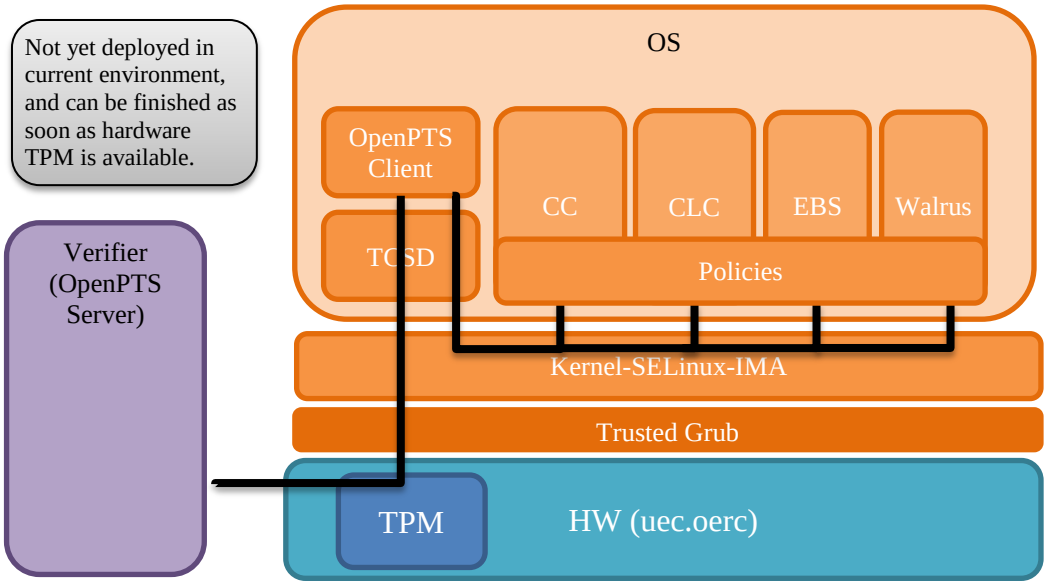

Figure 2. Management Node (CLC, CC, Walrus, EBS)

Verifications. Attestations to both the management node and the computing node can also be implemented in two ways: separated attestation and integrated attestation. For the separated attestation, clients first attest to the computing node, to verify the whether right images have been loaded and executed by genuine underlying infrastructure. They then attest to the management node, to verify whether the genuine management components are loaded with expected policies enforced, which can indicate that the expected virtual disk images are loaded and well protected. For the integrated attestation, the verification to the management node can be enforced by the software components in the Guest OS, and with the trustworthiness reflected in the integrity values of the Guest OS. Hence the clients only need to attest to the Guest OS and then make sure whether its applications, the underlying virtualization enforcement components (Host OS) and the cloud computing enforcement components (Management Node) are trustworthy.

2. Components Deployment

- Kernel-IMA

The IBM IMA patches have been integrated into the main kernel stream since 2.6.30. In our current environment, Ubuntu Maverick (10.10), the kernel version is 2.6.35, hence re-compiling the kernel with the IMA-related configurations can enable the IBM IMA supports.

```
$ sudo apt-get source linux-image-`uname -r`

$ cd $KERNEL-SRC-PATH

$ cp -vi /boot/config-`uname -r` .config

$ make oldconfig
```

```
set following in .config

CONFIG_IMA=y

CONFIG IMA MEASURE PCR IDX=10

CONFIG_IMA_AUDIT=y

CONFIG_IMA_LSM_RULES=y
```

```
$ fakeroot make-kpkg --initrd --append-to-version=-some-string-here
kernel-image kernel-headers

$ sudo dpkg -i ../linux-image-XXX.deb

$ sudo dpkg -i ../linux-headers-XXX.deb
```

- TPM Driver

TPM drivers are compiled by default in current Ubuntu versions as kernel modules. However, for the current Ubuntu Guest OS, where linux-image-2.6.35-virtual is used as kernel, no TPM driver module exists. But this is fixed by default when installing the Kernel-IMA to it.

- libTPMS

```
$ tar zxvf libtpms-0.5.1.tgz

$ cd $LIBTPMS-SRC

$ make –f makefile-libtpms

$ make –f makefile-libtpms install
```

- SeaBIOS

```
$ git clone git://git.seabios.org/seabios.git

$ cd $SEABIOS-SRC

$ patch –p0 <SeaBIOS-TPM.patch

$ make
```

- Qemu-TPM

```
$ git clone git://git.qemu.org/qemu.git

$ cd $QEMU-SRC

$ patch –p0 <qemu-tpm.patch

$ ./configure –-enable-kvm –-enable-tpm
```

```
$ make; sudo make install
```

- TCSD

The IBM implement of TCSD , the TrouSers, is included in current Ubuntu versions, and can be easily installed.

```
$ sudo apt-get trousers
```

- OpenPTS

```
$ sudo apt-get install libcommons-codec-java libcommons-logging-
java libpg-java liblog4j1.2-java libibatis-java

$ sudo apt-get install libcommons-discovery-java libaxis-java

$ sudo apt-get install liblog4j1.2-java-gcj libaxis-java-gcj

$ sudo apt-get install cdbs debhelper default-jdk-builddep


$ sudo ln -s /usr/lib/jvm/java-1.6.0-openjdk /usr/lib/jvm/java-6-
sun


$ git clone git://git.sourceforge.jp/gitroot/openpts/tools.git

$ cd tools

$ make dpkg-buildpackage

$ sudo dpkg -i ../openpts-tools_XXX.deb


$ git clone git://git.sourceforge.jp/gitroot/openpts/core.git

$ cd core

$ make dpkg-buildpackage

$ sudo dpkg -i ../openpts-core_XXX.deb

$ sudo dpkg -i ../openpts-core-gcj_XXX.deb
```

3. Experiments

```
$ mkdir seabios

$ cp /usr/share/qemu/*  seabios/ -R

$ cp $SEABIOS-SRC/out/bios.bin seabios/

$ qemu-img create qcow2 tpm.nvram 63k
```

```
$ ln –s $NEW-QEMU qemu-tpm

$ sudo qemu-tpm –kernel $KERNEL –initrd $INITRD –append $ARGS -
drive if=virtio,file=$DISK -tpm type=builtin,path=tpm.nvram -
nographic -L seabios -enable-kvm




……

tpm_tis:  read(00000018) = 000fef88

tpm_tis: write(00000024) = 00000000

tpm_tis: Byte to send to TPM: 00

tpm_tis:  read(00000018) = 000fee80

tpm_tis:  read(00000018) = 000fee80

tpm_tis: write(00000018) = 00000020

tpm_tis: tpm_tis: To TPM length = 18

00 C1 00 00 00 12 00 00 00 65 00 00 00 07 00 00

00 00

...

...

tpm_tis:  read(00000024) = 00000000

tpm tis:  read(00000018) = 00100080

tpm_tis:  read(00000018) = 00100080

tpm_tis: write(00000018) = 00000040

tpm_tis:  read(00000000) = 000000a1


Ubuntu 10.10 ubuntuhost ttyS0


ubuntuhost login:


ubuntu@ubuntuhost:~$ ls /dev/tpm*

/dev/tpm0

ubuntu@ubuntuhost:~$ tpm_version

  TPM 1.2 Version Info:

  Chip Version:        1.2.17.128

  Spec Level:          2
```

```
  Errata Revision:     3

  TPM Vendor ID:       IBM

  TPM Version:         01010000

  Manufacturer Info:   49424d00


ubuntu@ubuntuhost:~$ sudo cat
/sys/kernel/security/tpm0/ascii_bios_measurements

 1 1a6a0303dabdcbc104fdfd4302c90499a377ded4 06 [SMBIOS]

 2 4f4ca75d3a876437faa96e4d35d64110090faba6 06 [Option ROM]

 2 9dbd87163112e5670378abe4510491259a61f411 05 [Start Option ROM
Scan]

 4 c1e25c3f6b0dc78d57296aa2870ca6f782ccf80f 05 [Calling INT 19h]

 0 d9be6524a5f5047db5866813acf3277892a7a30a 04 []

 1 d9be6524a5f5047db5866813acf3277892a7a30a 04 []

 2 d9be6524a5f5047db5866813acf3277892a7a30a 04 []

 3 d9be6524a5f5047db5866813acf3277892a7a30a 04 []

 4 d9be6524a5f5047db5866813acf3277892a7a30a 04 []

 5 d9be6524a5f5047db5866813acf3277892a7a30a 04 []

 6 d9be6524a5f5047db5866813acf3277892a7a30a 04 []

 7 d9be6524a5f5047db5866813acf3277892a7a30a 04 []

 4 be38484ddd911f42aed970ccd2c42dfb6fa4350f 05 [Booting BCV device
00h (Floppy)]

 4 2daa7010226b59cee4227de5636b68767627acfc 0d [IPL]

 5 806d1fc2d901d2e0b498f5b1bd8b4964733ff9ec 0e [IPL Partition Data]


s untu@ubuntuhost:~$ sudo cat
/sys/kernel/security/ima/ascii_runtime_measurements

10 2032055c24ebfc2b3fa8c4268bbf7f21a9ef8507 ima
fca82a97f8741008a8613a0189812e9eb29d209b boot_aggregate
```