TCB-Protection

This document contains steps involved in enabling TCB-protection for trusted-hosts on Ubuntu and RHEL.

# Terminology

Following terminologies has been used in document:

* Build-host – This is the machine on which TCB-protection enabled initrd will be generated [Kernel version of build-host and trusted-hosts has to be similar]
* Trusted-hosts – These are the host machines, typically in data centers, which will be actually hosting measured VMs. These hosts will be booted using generated initrd by this procedure on build-host
* TCB-ROOT Directory – This is the root directory where all tcb-protection related code exists. This is the folder where “dcg\_security-tboot-xm/” repository is checked out

# Part 1 – Build Host

* **build-rpmmio.sh** script under **dcg\_security-tboot-xm/tbootxm-rpmmio directory** builds rpmmio.ko kernel module for all kernel devel packages available on build machine. For e.g. If on build machine 10 different version of kernel devel packages are installed on build 10 rpmmio.ko module will generated, 1 with each devel package.
* **build\_components.sh** script under **dcg\_security-tboot-xm/** directory installs required packages on host machine (if executed with provided with options) and build all component of TCB-Protection. Components include verifier (MA), and tpmextend.

# Part 2 – TCB-Protection Enabling for Host

## How TCB-Protection works for Host

TCB-Protection for host is used to make sure that root file system of the trusted host is not modified while system boots up. To achieve this initrd (for Ubuntu) or initramfs (for RHEL) of the host has been modified. In computing, initrd (initial ramdisk) is a scheme for loading a temporary root file system into memory, which may be used as part of the Linux startup process. initrd and initramfs refer to two different methods of achieving this. Both are commonly used to make preparations before the real root file system can be mounted. Brief info about working of initrd and initramfs.

* **Initrd for Ubuntu :**
  + **mkinitramfs** utility is used to generate initrd image for Ubuntu. It has to kind of scripts one Hook Scripts and Boot Scripts.
  + **Hook Scripts** are used generate initrd image and these are not included in initrd image. These scripts are stored at **/usr/share/initramfs-tools/hooks/** dir. Our script named **tcb** is also stored in this directory. In this script we specify all the dependency which we need to include in initrd image. In this script we have specified our dependency like verifier( MA binary), tpmextend, rpmmio.ko and few Linux utilities like grep, awk etc. .
  + **Boot Scripts** are scripts are executed in user space at boot time before root file system is mounted. Boot scripts are stored at **/usr/share/initramfs-tools/scripts/** and this directory has many subdirectories which control the boot stage at which the scripts are executed. Our boot script measure\_host is kept under local\_premount directory.
  + For more info mkinitramfs page[**link**](http://manpages.ubuntu.com/manpages/precise/en/man8/initramfs-tools.8.html) **.**
* **Initramfs for RHEL:**
  + **Dracut** utility is used to generate initramfs image for RHEL. It also works similar to mkinitramfs. dracut uses a modular system to build and extend the initramfs image. All modules are located in **/usr/lib/dracut/modules.d/**.
  + We have created our module directory with name of **89tcbprotection.** All module installation information is in the file **module-setup.sh.**
  + In each module we can write certain scripts named **check, depends, install, installkernel, etc**. each script has pre-defined functionality which it should satisfy. Module directory obviously contains module-setup.sh as mentioned earlier.
  + Like **check** is used as check, it is called by dracut to evaluate the inclusion of a dracut module in the initramfs, in **install** all non-kernel files should be installed in initramfs image, and in **installkernel** all kernel related files should be installed.
  + dracut modules can insert custom script at various points, to control the boot process. These hooks are plain directories containing shell scripts ending with ".sh".
  + dracut supports various BOOT PROCESS STAGES. We are using pre-mount boot stage, means at this stage of boot our **measure\_host** script will be executed.
  + Scripts in **pre-mount BOOT STAGE** are executed by dracut before the root device is mounted all scripts in the hook pre-mount are executed. In some cases (e.g. NFS) the real root device is already mounted, though.
  + **module-setup.sh** uses certain create functions to create initramfs image like inst, instmods, inst\_hook.
  + **inst** <src> [<dst>] installs one file <src> either to the same place in the initramfs or to an optional <dst>. In this script we have specified our dependency like verifier( MA binary), tpmextend, rpmmio.ko and few Linux utilities like grep, awk etc..
  + **inst\_hook** <hookdir> <prio> <src> installs an executable/script <src> in the dracut hook <hookdir> with priority <prio>. For e.g. command used by us is : **inst\_hook pre-mount 70 "$moddir"/measure\_host.sh.**
  + **instmods** <kernelmodule> [ <kernelmodule> ... ] instmods should be used only in the installkernel script. instmods installs one or more kernel modules in the initramfs.
  + For more info about dracut please refer dracut page [link](http://man7.org/linux/man-pages/man7/dracut.modules.7.html).

1. **Installation of TCB-Protection on host:**

* TCB-Protection have script named **generate\_initrd.sh** under **tcb\_protection** directory. It first create initrd (for Ubuntu) or initramfs (for RHEL).
* **generate\_initrd.sh** checks all pre-requisite on host machine like, availability of verifier, tpmextend, rpmmio.ko for current kernel version in which system is booted, measure\_host script. If these pre-requisite are met then it will create initrd or initramfs image which contains all these.
* After creating image it creates grub entry for TCB-Protection which will use this recently created initrd or initramfs image.
* To crate grub entry it looks for grub entry for current kernel version in /boot/grub/grub.cfg ( file name is specific to Ubuntu, RHEL), and using this entry it creates entry for TCB-Protection and append it to grub file present under /boot/grub/ directory.
* While creating grub entry it also found out info about all devices available on host and their mount point, and add it to file **tboot-xm.conf at /**. In tbootxm.conf it also add the location of **manifest.xml** which contains list of files to be measured. At last location for this file is added in grub entry as **CONFIG\_FILE\_PATH="/tbootxm.conf**.
* At this stage Host is ready to be booted under TCB-Protection.

1. **Execution of TCB-Protection on host:**

* So, Commonly on both Ubuntu and RHEL our additional binaries are added in initrd for host measurement and extending PCR values.
* A script (*measure\_host*) has been added which is executed by initrd before it does chroot to root filesystem of the host.
* When System boots mkinitramfs or dracut executes our measure\_host script at specified hook point at local\_premount in case of Ubuntu and pre-mount in case of RHEL.
* *measure\_host* script mounts all devices available on the host at */tmp/root* so that measurement of the required files is possible by reading info from **tbootxm.conf** (information related to devices and their mount points on host is provided in this file).
* Once partitions are mounted verifier available in initrd gets called and it measures all files mentioned in manifest.xml file (location of manifest.xml file on host file system has been provided in **tbooxm.conf**).
* Result of the verification is used to extend PCR-19 value using **tpmextend** binary present in initrd image so that later Trust Agent and Mt Wilson can use this value to attest host
* Once all this operations are done *measure\_host*script unmounts all partitions and control goes back to initrd and normal booting operation continues.

## Part 3: How to handle failure scenarios

All TCB-Protection related changes are part of the initrd and get executed while system boots up. So, any failure in this could result in halting of booting process.

There are some failures, about which we could not do much like, using corrupted initrd, wrongly configured grub entry, etc.

But there are some cases where we can handle error gracefully as part of *measure\_host* script. Examples of such errors are:

* Wrong manifest path
* Wrong mount point information
* Failure while executing verifier, rpmmio or tpmextend binaries
* Etc.

If such errors occur which we could handle those in 2 different ways:

1. Halt booting process: On error halt booting process by sending kernel panic call
2. Stop host measurement process but continue with booting: On error stop measurement process (means exit from *measure\_host* script) so that PCR-19 value will not get extended and post booting PCR-19 value will not match with the one registered with Mt Wilson and it will be able to detect that host measurement is not done properly

**<TBD: Which approach to go with. Depending on that small changes needs to be done in *measure\_host* script.>**

Current implementation is works as per approach #2.

## Part 4: Step-by-step Procedure to Enable TCB-Protection for Host

**NOTE: Please run the following scripts from current directory only where scripts exists.**

Following are the steps involved in host TCB protection enabling.

1. Generating initrd with TCB protection enabled in it
2. Creation of measurement using Trust Director (Creation of TCB-manifest.xml file)
3. Configuring trusted-hosts using TCB-protection enabled initrd

### 1. Generating initrd with TCB protection enabled in it

##### Pre-requisites

Please make sure that following pre-requisites are fulfilled before you proceed:

1. Build and copy following binaries in <TCB-ROOT>/tcb\_protection/bin directory.
   * tpmextend
   * verifier
   * rpmmio.ko

If want to build code on the same machine then execute *build\_components.sh* script. It will build and copy above mentioned binaries in required folder.

If build dependencies are not installed then install those. For this you can execute script with option --installpkg.

*Note : Ensure the rpmmio.ko is build for the same kernel version as the bare-metal machine where we wish to deploy TCB protection.*

1. Kernel version of build-host should be exactly same as trusted-hosts

After fulfilling the above mentioned Pre-requisites, generate new initrd for host OS protection by following the steps:

1. Go to the <TCB-ROOT> directory and run the “generate\_initrd.sh” script.

***e.g.***

***# cd /home/intel/dcg\_security-tboot-xm/tcb\_protection***

***# ./generate\_initrd.sh***

This script removes any pre-existing initrd from “<TCB-ROOT>/tcb\_protection/generated\_files” directory and creates a new initrd named “initrd.img-<KERNEL-VERSION>-generic-measurement” in the “<TCB-ROOT>/tcb\_protection/generated\_files” directory.

### 2. Creation of measurement using Trust Director (Creation of TCB-manifest.xml file)

Using Trust Director create measurement of the trusted-host and copy it over the bare-metal. This location will be asked by the script which configures the TCB protection.

<FOR STEPS REFER TRUST DIRECTOR GUIDE>

### 3. Configuring trusted-hosts using generated TCB-protection enabled initrd

For **Ubuntu OS with grub 2.0**, follow below mentioned steps:

1. Execute *<TCB-ROOT>/tcb\_protection/configure\_host.sh* script
2. Script will ask for manifest file location. Provide complete path of the manifest file
3. Script will create menuentry from existing menuentry available in /boot/grub/grub.cfg file and will populate it in /etc/grub.d/40\_custom. Read logs carefully. In case of error follow suggestions
4. Once grub is updated (either by script or by following suggested steps manually) check /boot/grub/grub.cfg file and look for menuentry whose name starts with prefix “TCB-Protection” (See sample menuentry shown below and look for highlighted values in menuentry)
5. Reboot system and select above grub option

Here is the sample menuentry from /etc/grub.d/40\_custom or /boot/grub/grub.cfg. Modified values are highlighted in RED.

Menuentry ‘***TCB-Protection* Ubuntu – tboot**’ --class ubuntu --class gnu-linux --class gnu --class os --class tboot {

insmod part\_msdos

insmod ext2

set root='hd0,msdos1'

if [ x$feature\_platform\_search\_hint = xy ]; then

search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 e9b96055-6bbf-45fb-9040-e7f28d84e91b

else

search --no-floppy --fs-uuid --set=root e9b96055-6bbf-45fb-9040-e7f28d84e91b

fi

echo 'Loading tboot 1.8.1 ...'

multiboot /boot/tboot.gz /boot/tboot.gz logging=serial,vga,memory

echo 'Loading Linux 3.13.0-32-generic ...'

module /boot/vmlinuz-3.13.0-32-generic /boot/vmlinuz-3.13.0-32-generic root=UUID=e9b96055-6bbf-45fb-9040-e7f28d84e91b ro biosdevname=0 intel\_iommu=on ***MANIFEST\_PATH=”<path to manifest file>” PARTITION\_INFO=”<partition\_information>”***

echo 'Loading initial ramdisk ...'

***module <path to newly generated initrd present boot device> <path to newly generated initrd present in boot device>***

echo 'Loading sinit 3rd\_gen\_i5\_i7\_SINIT\_67.BIN ...'

module /boot/3rd\_gen\_i5\_i7\_SINIT\_67.BIN /boot/3rd\_gen\_i5\_i7\_SINIT\_67.BIN

}

<START: NEEDS TO BE UPDATED>

For **other OSs (apart from Ubuntu or with grub 1.0)** follow below mentioned steps on trusted-hosts to enable TCB-Protection:

1. Copy “initrd.img-<KERNEL-VERSION>-generic-measurement” file from <TCB-ROOT>/generated\_files folder on build-host to “/boot” of trusted-host
2. Configure grub entry on trusted host
   1. Create a grub entry for TCB protection in file */boot/grub/grub.cfg* (as sample menuentry shown below) along with the kernel parameters mentioned below. (TBD: This is a manual step, which will be automated in future)
   2. Make sure that following entities has been correctly modified in newly created grub menuentry:
      1. Modify name of the menuentry
      2. Initrd module – Path of the copied initrd file “/boot/initrd.img-<KERNEL-VERSION>-generic-measurement”
      3. Additional kernel argument as follows :
         1. “MANIFEST\_PATH” – Path to the manifest file required for host verification
         2. “PARTITION\_INFO ” – This will be the information of disk partition of bare-metal. For e.g {/dev/sda1:/boot , /dev/sda2:/root, /dev/sda3:/var}.

**NOTE:** Additional kernel argument could be created manually by gathering required information from trusted host machine or else you can execute *<TCB-ROOT>/tcb\_protection/configure\_host.sh* script on trusted host. It will ask for manifest file location and will generate required kernel arguments by gathering information from system which could be copied in menuentry to avoid manual mistakes.

<TBD: SAMPLE GRUB ENTRY>

1. Execute update-grub entry. It will add the entry from /etc/grub.d/40\_custom to /boot/grub/grub.cfg
2. Reboot the system using the TCB grub entry created

<END: NEEDS TO BE UPDATED>

After reboot TCB protection related log will be available at */var/log/tcb.log*. Cumulative hash of the host measurement will be available at */var/log/cumulative\_hash.txt*. Host measurement log will be available at */root/MA\_Hash.xml*.

# Part 2 – VM storage encryption:

Requirements for VM storage encryption are as below:

* Create loopback mount point for each encrypted VM storage
* Encrypt it using dm-crypt
* Key for encryption would be the same as VM encryption key received from KMS
* Encryption key will be stored on disk (this will be seal using TPM)

This will require changes in Policy Agent (MH Agent) and Nova compute.

Following are some of investigation results and approaches to move ahead to achieve VM encryption:

1. **Encrypted loop-device per measured VM**
   1. For a VM launch, its corresponding image and its DEK will be gathered (from Policy Agent). The DEK will be sealed and saved along with the base image.
   2. When the disk is being created for the VM, we will initiate create encrypted loop-mounted device. The key for this device will be the same DEK for corresponding image. The encrypted device will be used to store the image-disk.
      1. PROS :
         1. Each VM will have its own encrypted device, with a key shared with images having same base image.
      2. CONS :
         1. Hypervisor will not be able to do thin provisioning of the disk resource. For e.g, in current OpenStack version, if we create VM disk, the disk is created using qemu-img –o backing\_file option. By using this option, only the differences to be stored in the newly created VM disk, which helps save the disk resource. This flexibility is not available in dm-crypt and hence we will have to create disk with maximum possible size. As a result, we will lose out over-committing of resources.
         2. Nova compute will need more changes. Might need to handle different image formats like qcow2, raw etc differently. Also will need changes in Openstack for handling for startup and shutdown of all VMs.
         3. We will have to provision the number of loop devices for bare-metal. Which will limit the number of encrypted VMs that can be launched simultaneously.
      3. RISK:
         1. Base images under /var/lib/nova/instances/\_base would be in plain format [Some more investigation required here as this might be used as backing file for launched VMs]
            1. One possible solution to deal with: Create a loop device for storing base images at boot time. All the base images will be downloaded and decrypted in this device. A random key for this device will be created at boot time (as in Approach 2) and preserved across reboots.
            2. Deciding upon the size of the loop device will be very critical. Once decided it will be difficult to change without bare-metal downtime.
2. **One Encrypted loop-device per host**
   1. Create a random key at a well-known location, if not present. Once created, seal key using TPM.
   2. Create a loop device if not already created. The name of device can be well-known. If the device is already present, we will use the existing device.
   3. Unseal the key and use it to crypt mount a loop-device
   4. All the base images and VM images will reside in this encrypted partition.
      1. PROS :
         1. Minimal or no change in nova-compute
         2. Encrypted partition mounting using dm-crypt will be one time activity on host startup [Is it OK to do mount while startup or should do it via initrd?]
         3. VM Disk thin provisioning will work seamlessly ( pl refer to approach below for more information )
      2. CONS:
         1. Single encrypted partition for all VMs
         2. There will be one key for the entire partition.
         3. Size of encrypted partition will be fixed. [TBD: How to provide / decide size of the partition]
         4. Deciding upon the size of partition is critical, but is less risky as in previous approach. Since the bare-metal will be a dedicate hypervisor we can allot maximum possible space for encrypted loop device.

**Following are the flows needs to be taken care:**

* Launch VM
* Reboot VM
* Shutdown VM
* Migrate VM
* Snapshot of VM
* Reboot Host (when to mount encrypted VM partitions of previously launched VMs)