vRTM Overview

# Overview

Function of vRTM is to measure VMs to be launched and based on policy allow or deny VM launch request. It also holds metadata about launched VM in its in-memory data structure.

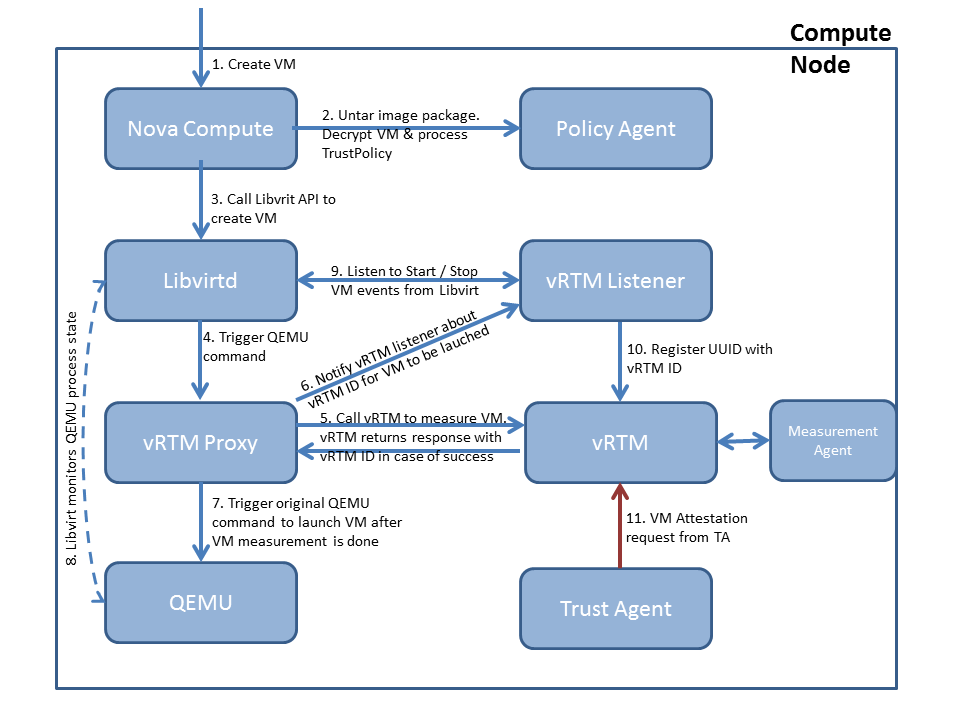
Following are the main components and their primary responsibilities:

* vRTM (or vRTM Core):
  + This is main component of vRTM, running as a service accessible over socket
  + It is responsible for getting VM measured and maintaining metadata related to measured VM
  + It provides socket based APIs to get VM metadata
* Measurement Agent:
  + This is standalone program which accesses VM disk and does measurement of all files provided in manifestlist.xml file
  + It writes cumulative hash of the VM measurement and hashes of all measured files in files under VM instance directory
* vRTM Proxy
  + vRTM proxy is a binary which works as proxy to intercept requests from Libvirt to QEMU
  + vRTM proxy replaces QEMU binary and it gets executed first whenever QEMU binary is called
  + It checks for VM launch request. As soon as it found VM launch request it checks whether its measured VM launch or not. If measure launch request it calls vRTM APIs to measure VM disk
  + Once measurement is successful it notifies vRTM Listener about vRTM ID and VM name which is going to be launched
  + Finally it executes QEMU command line which actually launches as VM
* vRTM Listener
  + This is a service doing 3 functionalities
    - Listening to Libvirt’s start / stop VM events
    - Starts socket listener to listen from vRTM proxy about VM launch request
    - Communicate with vRTM to notify it about VM state change
  + vRTM Listener registers itself with Libvirt to listen to VM start / stop VMs
  + It starts socket on which it can received notifications from vRTM Proxy about VM launch. It stores vRTM ID and VM name in its internal memory for later use for updating VM UUID and vRTM ID mapping
  + On receiving event from Libvirt, it notifies vRTM over socket about. In case of start VM, it sends vRTM ID and UUID of VM to vRTM for mapping. And for deletion of VM, it provides UUID of the VM so that vRTM can delete entry
* vRTM Channel
  + This is a library containing APIs to encode / decode request / responses for all APIs
  + It also provides APIs to read and write on socket channel to do communication with vRTM
  + This library is used by vRTM, vRTM Proxy and vRTM Listener

# Request Flow

## Launch VM Request

When new VM instance is started Launch VM Request comes. Below is the request flow diagram of the launch VM request on Compute node:



Below is the detailed description of the steps mentioned in the above diagram:

1. User will trigger VM launch request from Horizon UI, which will propagate to controller and scheduler as per normal Openstack flow [Not shown in diagram]
2. Controller will send create VM request to Nova Compute on the selected node by scheduler
3. While processing launch VM request, Nova compute will call Policy Agent (patches has been applied in Nova Compute code to achieve this) to get trustpolicy.xml, manifestlist.xml, decrypt VM image, create encrypted disk and copy required artifacts at appropriate locations
4. Once image and trustpolicy are at desired location, Nova compute calls Libvirt API to launch VM
5. Libvirt processes request and forms command line to start QEMU process and executes it. But as original QEMU binary is replaced by vRTM Proxy, vRTM proxy process starts
6. vRTM proxy parses command and if it’s for VM launch, it calls vRTM API to measure VM.
   * vRTM then calls Measurement agent to measure all files mentioned in manifestlist.xml file and their cumulative hash.
   * Depending on policy and outcome of measurement it decides and sends response to vRTM proxy about whether to launch VM or not
   * vRTM also holds VM measurement and other metadata in its in-memory data structure
7. If vRTM decision is deny then VM launch is denied at this point and process stops. Else vRTM Proxy notifies to vRTM Listener about vRTM ID (generated by vRTM for this VM) and its name which it holds in its internal memory for later use
8. vRTM Proxy starts QEMU process from its process so that Libvirt can monitor it
9. Libvirt keeps monitoring QEMU process (running VM process) and generates events on state change of VM
10. vRTM Listener has registered itself with Libvirt process for VM Start / Stop events. So, as soon as QEMU process starts and VM image booting starts it send Start VM event, which is detected by vRTM Listener
11. vRTM Listener finds out vRTM ID of the VM from VM related data received as part of event and mapping stored in its memory (in step 6). It sends request to vRTM over socket to update VM UUID against the vRTM ID of the VM

Here actual VM launch process is completed.

In diagram addition step is shown which is used to show VM attestation status on Horizon UI.

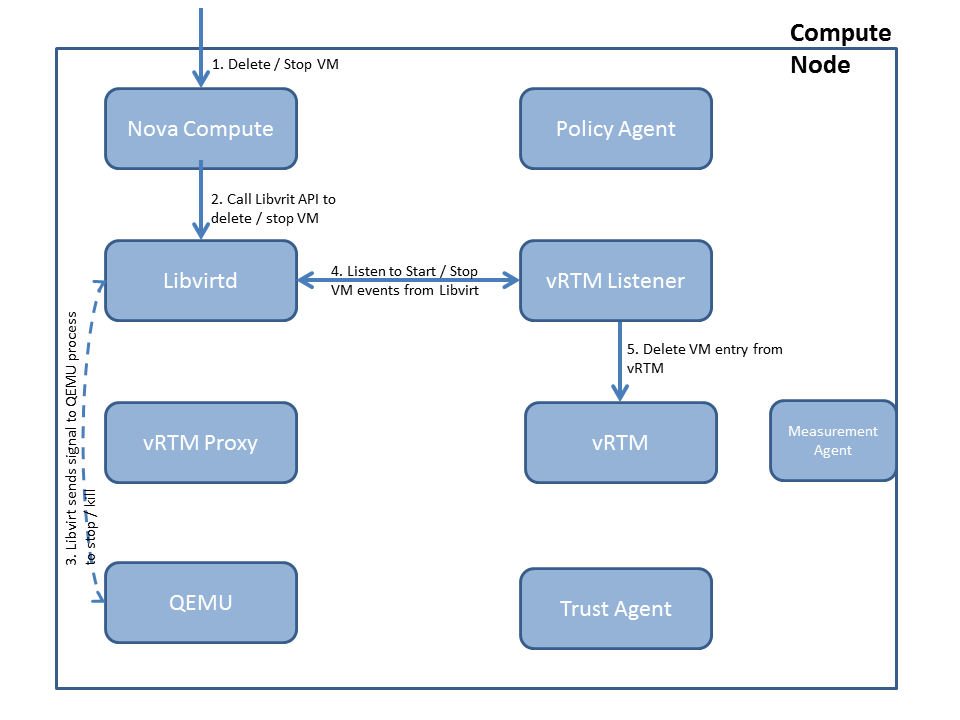
1. TA asks for VM Attestation status from vRTM using VM’s UUID over the socket

## Start VM Request

If VM was launched previously and now user wants to start it Start VM request comes.

Flow of the start VM is exactly same as mentioned above except that step 2 doesn’t get executed. Means, Policy Agent doesn’t get executed as VM image and all required artifacts are already in place as this is not a new VM launch request.

## Stop VM Request

Request flow diagram of the stop VM request on Compute node:

Below is the detailed description of the steps mentioned in the above diagram:

1. User will trigger stop VM request from Horizon UI, which will propagate to controller as per normal Openstack flow [Not shown in diagram]
2. Controller sends stop VM request to Libvirt
3. Libvirt sends signal to QEMU process to stop
4. Libvirt keeps monitoring its state and generate event as soon as process stops
5. vRTM Listener is listening to Libvirt events
6. On stop VM event, vRTM listener notifies vRTM to remove VM entry from its internal in-memory data structure for given VM UUID. And vRTM removes entry from that VM

## Reboot VM Request

Reboot VM is basically Stop VM followed by Start VM for Libvirt.

# Code Structure

Here is the code structure and explanation of each file:

dcg\_security-vrtm

|-- blueprints : Contains documents

|-- build : Contains scripts for building vRTM

| -- vRTM\_KVM\_build.sh : Build script for building all vRTM components

| -- vRTM\_libvirt\_build.sh : Script to build Libvirt – Not in use. May needed if Libvirt needs to be modified

|-- install : Contains scripts for installing vRTM components

| -- vRTM\_KVM\_install.sh : Installation script to install vRTM components

|-- packages : installer related code

| -- test\_util : Test code to test vRTM

| -- vrtm : vRTM code base

| -- configuration : Contains configuration files and log properties

| -- scripts : Scripts used by vRTM components while execution

| -- src : CPP code for vRTM components

| -- Makefile : Master make file

| -- util : Contains utility code

| -- base64.h / .cpp : Functionalities related to base64 encoding decoding

| -- logging.h / .cpp : Logging related code

| -- vrtmCommon.h : Contains definitions

| -- vrtmchannel : vRTM channel code used by all vRTM components

| -- channelcoding.h / .cpp : Functions to parse req / resp for all vRTM APIs

| -- log\_vrtmchannel.h / .cpp : Function to set logger from parent components

| -- parser.h / .cpp : Code to parse XML

| -- tcpchan.h / .cpp : Functionalities related to reading / writing over channel

| -- vrtmchannel-g.mak : Make file

| -- vrtmcore : vRTM core service related code

| -- modtcService.h / .cpp : Contains business logic for all APIs and internal data structure used by vRTM

| -- vrtm\_api\_code.h : Definitions of vRTM API codes

| -- vrtmcoremain.cpp : Main class to start vRTM core service and initializing logger and properties, etc.

| -- vrtminterface.h / .cpp : Code to start vRTM socket and processing requests

| -- vrtmcore-g.mak : Make file

| -- vrtmproxy : Code base for vRTM Proxys

| -- kvm\_proxy : Code base for vRTM KVM proxy

| -- vrtm\_proxy.cpp : Code for KVM proxy

| -- vrtm-proxy.mak : Make file for KVM proxy

| -- vrtmlistener : Code base for vRTM listeners

| -- kvm\_listener : Code base for vRTM Libvirt listener

| -- vrtm\_listener.cpp : Code for Libvirt listener

| -- vrtm-listener.mak : Make file for Libvirt listener

| -- vrtm-maven-root : Maven build and installer related code