DANOS as a vFW

DRAFT

Work in progress documentation

The ONAP vFW VNF used for testing and demonstration was created because there wasn't a readily available opensource VNF that could be used for testing. Over time maintaining the source has been a challenge and it would be good to use a supported full featured VNF for the vFW/vLB functions. This Tutorial will show the steps to make the DANOS Virtual Router work as a vFW VNF for ONAP. It will describe both the steps to add a VES client and the steps to create the heat template artifacts, service mapping and preload data so that the DANOS vRouter can be automatically on boarded and instantiated just like the standard vFW in ONAP. Over time we may consider using the DANOS vRouter for more use cases since it is a more feature VNF than the home grown vFW.

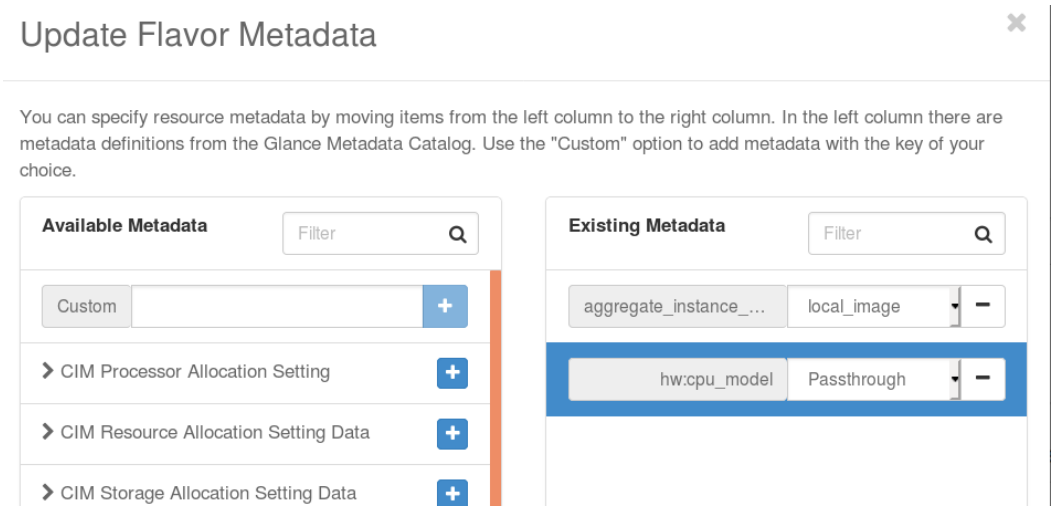
The steps below have been added into the ONAP repositories already so they do not need to be repeated but will document the steps taken so that they could be used for other VNFs as appropriate. To simplify the tutorial we will point to other wiki pages and source files in gerrit rather than showing all the steps of editing the files.

It is important to point out that while DANOS has a commercial equivalent that is deployed in production, the VES client we are using is for demonstration only and does not cover all the things needed for a production deployment. It does however support the functions we need for testing.

These steps could be followed as a general outline to add automated testing for any VNF to ONAP.

Step-by-step guide

1. Add VES to your VNF
    a. Create a debian package of the VES client for DANOS using the VESreporting_vFW5.0_DANOS sub-directory from the demo repository
    b. The DANOS build process uses debian packages so this step is creating a VES reporting Debian client that can use the DANOS API's to retrieve statistics
    c. libevel.so must be built and bundled into the package since its not available on the ONAP artifact repositories as a debian package.
2. Create a Openstack Image
    a. Download or Create a DANOS ISO image that includes the VES client
    b. RECOMMENDED: Download a DANOS ISO image that includes the VES client from here fill in url
        i. this iso image was built with dependency constraints that make sure only the correct versions of upstream debian libraries are pulled in.
        ii. DANOS 2005 release will fix this problem.
    c. ALTERNATIVE: Create an Image.
        i. This is following the existing DANOS procedures for building an ISO from binaries and the procedures for creating an Openstack image
        ii. NOTE: there is a build issue that debian libraries newer than those referenced in the build package may mistakenly be pulled in by the debian build so we recommend using the pre-built ISO image referenced above.
        iii. Follow instructions in BUILD_DANOS.md in the demo repository which has slight changes from the DANOS tutorials.
        iv. https://danosproject.atlassian.net/wiki/spaces/DAN/pages/491554/Creating+a+DANOS+ISO+using+binary+packages
        v. https://danosproject.atlassian.net/wiki/spaces/DAN/pages/79560705/Creating+a+DANOS+Virtual+Router+VNF+in+Openstack
    d. DANOS supports SNMP already but we want to use the vRouter as a replacement for the existing vFW which is VES based.
    e. DANOS could be used for both VES and SNMP based testing with the addition of this simplified VES client.
3. Create the heat template
    a. Create the heat template for DANOS under a new service vFWCLDN
    b. vFWCLDN is a short hand that makes naming of the VNFs fit within the string lengths in Openstack and ONAP.
    c. The full VNF will be the same packet generator and traffic sink as the standard vFWCL but use DANOS vRouter instead of the ONAP vFW
    d. The DANOS vRouter heat template is minimal.
4. Create the preload_data
    a. This will be the same data items as vFWCL but will demonstrate one of the steps to use for onboarding any VNF to ONAP's automated testing
5. Create the service mapping data
    a. This will add the pointers to the heat templates that are primarily just a name change from vFWCL to vFWCLDN so that the testing tools can map the service to the locations of the various artifacts
6. Add the image for your VNF to your glance repository
    a. Add the image for the DANOS with VES vRouter to your Openstack glance repository
7. Create the flavor for your VNF
    a. Create the flavor for the DANOS vRouter - 4 vcpu, 4 Gig ram, 8 GB disk with hw_options for cpu_model: passthrough
        i. cpu passthrough is the simplest way to ensure that SSSE3 is exposed as a capability in the virtualized hardware.
        ii. hw:cpu_model = passthrough



   a. Standard vFW is m1.medium but DANOS is a smaller footprint and an on disk image that is only 8 GB instead of a full Ubuntu volume.
2. Add a tag for running tests with your VNF
   a. Add instantiateVFWCLDN tag and data for an instantiate VNF testcase to demo.robot
   b. The command line "./ete-k8s.sh onap instantiateVFWCLDN" would now onboard the vFWCLDN models and instantiate the VNF as a DANOS vFW and a traffic sink and a traffic generator without having to use the GUI's directly so it could be used for regression.
3. Document any post instantiation steps
   a. DANOS Post Install Configuration
      i. Use curl/postman to make Netconf configuration changes to the vRouter to configure the ports - this demonstrates post instantiation configuration through SDNC
      ii. Update the DACE collector ip addres and port in the vRouter if not provided on your VNF by cloud-init (also works for PNFs)
   b. Use horizon to remove the "Port Security" on the ports for the 3 virtual machines in case your version of openstack defaults to activate port security.
4. Test that the VNF works with ONAP
   a. Closed Loop telemetry from DANOS to DCAE/VES can be confirmed indicting both correct traffic flow from the packet generator to the traffic sink through the DANOS vRouter but also that VES telemetry is properly going to DCAE and being processed as events.
   b. Use POSTMAN to get events from the VES collector output
   c. GET https://{{dmaap_ssl_port}}/events/unauthenticated.VES_MEASUREMENT_OUTPUT/g1/c3?timeout=5000

dmaap VES event


```
"{\"event\":{\"commonEventHeader\":{\"startEpochMicrosec\":1588882076723273,\"eventId\":\"
mvfs00000001\",\"sequence\":0,\"domain\":\"measurementsForVfScaling\",\"lastEpochMicrosec\":
1588882086723273,\"eventName\":\"vFirewallBroadcastPackets\",\"reportingEntityId\":\"No UUID
available\",\"internalHeaderFields\":{\"collectorTimeStamp\":\"Thu, 05 07 2020 08:07:27 UTC\"},\"
sourceName\":\"vofwl01fwleccf\",\"priority\":\"Normal\",\"version\":3,\"reportingEntityName\":\"
vyatta\"},\"measurementsForVfScalingFields\":{\"measurementInterval\":10,\"
measurementsForVfScalingVersion\":2,\"vNicPerformanceArray\":[{\"transmittedOctetsDelta\":0,\"
receivedTotalPacketsDelta\":1003,\"vNicIdentifier\":\"dp0s4\",\"valuesAreSuspect\":\"true\",\"
transmittedTotalPacketsDelta\":0,\"receivedOctetsDelta\":60180}]}}}"
```


   d. Notice the sourceName is the DANOS firewall and the receivedTotalPacketsDelta is 10003 representing the stream:10 setting on the packet generator.
   e. Use POSTMAN to get the TCA EVENT output
   f. GET https://{{dmaap_ssl_port}}/events/unauthenticated.DCAE_CL_OUTPUT/g1/c3?timeout=5000

DCAE TCA Event Output


```
"{\"closedLoopEventClient\":\"DCAE_INSTANCE_ID.dcae-tca\",\"policyVersion\":\"1.0.0\",\"
policyName\":\"DCAE.Config_tca-hi-lo\",\"policyScope\":\"DCAE\",\"target_type\":\"VM\",\"AAI\":{\"
vserver.prov-status\":\"ACTIVE\",\"vserver.resource-version\":\"1588875887013\",\"vserver.is-
closed-loop-disabled\":false,\"vserver.vserver-name2\":\"vofwl01fwleccf\",\"vserver.vserver-id\":\"
25413ae3-11ed-408d-9bdd-c2ce3926097c\",\"vserver.vserver-selflink\":\"http://10.12.25.2:8774/v2.1
/712b6016580e410b9abfec9ca34953ce/servers/25413ae3-11ed-408d-9bdd-c2ce3926097c\",\"vserver.in-
maint\":false,\"vserver.vserver-name\":\"vofwl01fwleccf\"},\"closedLoopAlarmStart\":
1588877023057584,\"closedLoopEventStatus\":\"ONSET\",\"closedLoopControlName\":\"ControlLoop-
vFirewall-e713e960-8dd2-4b7e-9c8d-e439bdb30bc3\",\"version\":\"1.0.2\",\"target\":\"vserver.
vserver-name\",\"requestID\":\"914a94eb-90c0-4b2a-baf3-fbc309385041\",\"from\":\"DCAE\"}",
```

g. Notice the closedLoopEventStatus is ONSET since the traffic is above the 700 packets threshold in the policy.

5. We can also use the netconf interface from SDNC to the DANOS virtual router to create and change the firewall rulesets.
    a. Configure DANOS Firewall

Related articles

- Committer Promotion Request for CCSDK - John Keeney - DRAFT
- NCMP De-registration Performance test guide
- Jeff van Dam - Committer Promotion Request for [SDC]
- Francisco Javier Paradela - Committer Promotion Request for [SDC]
- Committer Promotion Request for [all DCAE]: Shuting Qing