Virtual Network Broker Brief

Version 1.0

16/06/2017

Table of Contents

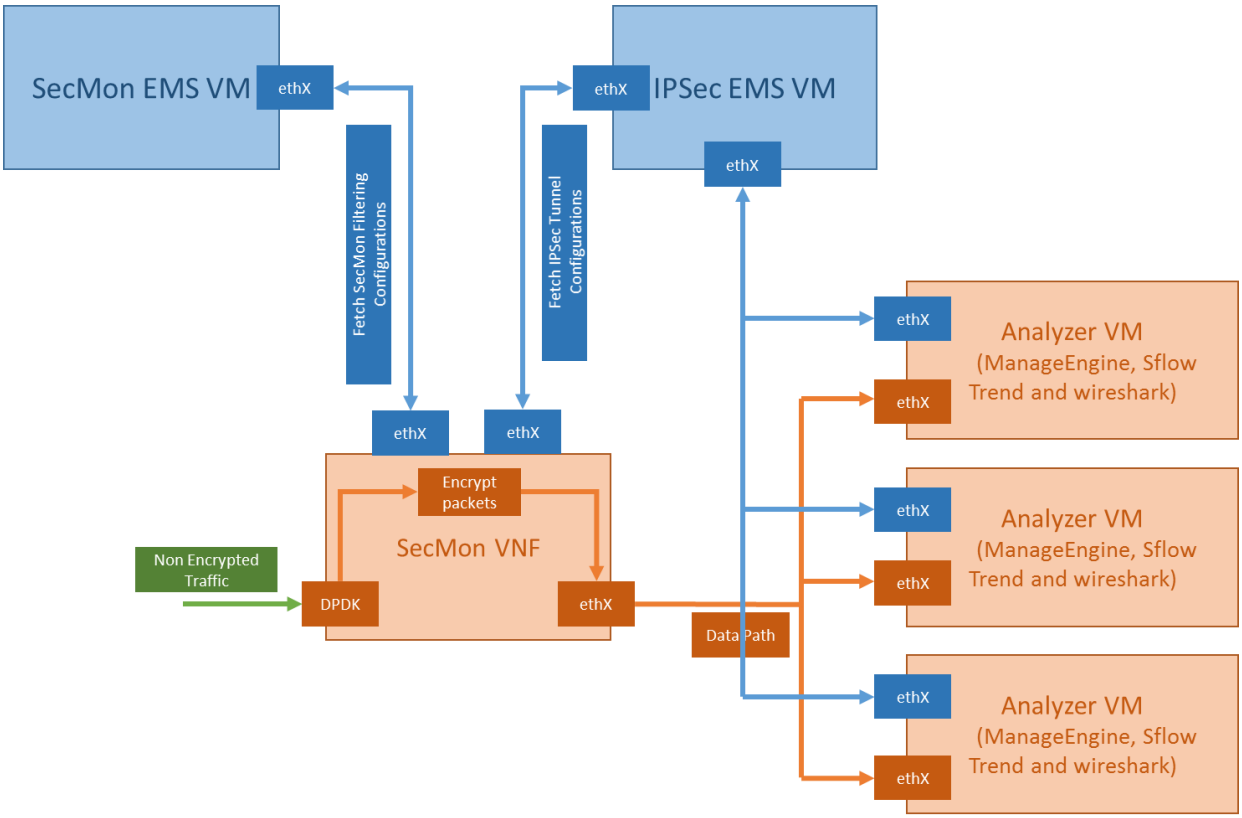
Table of Contents 1

1. Overview 2
2. Monitoring as a Service 2
   - 2.1 SecMon VNF 2
     - 2.1.1 SecMon Agent 3
       - 2.1.1.1 Interface between SecMon Agent and DPDK library 3
     - 2.1.2 SecMon Plugin 3
       - 2.1.2.1 Interface expected by SecMon in plugins 3
   - 2.2 SecMon EMS VM 3
     - 2.2.1 Components of SecMon EMS 4
3. IPSec Function - Architecture 4
   - 3.1 IPSec EMS 4
   - 3.2 IPSec Enforcers 5

1. Overview

Virtual Network Broker or Secured Monitoring comprises of following:

- **Monitoring as a Service** provided by components SecMon EMS and SecMon
- **IPsec function** provided by components IPSec EMS and IPSec Enforcers



Secure Monitoring Architecture with IPSec Secure Tunneling

2. Monitoring as a Service

2.1 SecMon VNF

SecMon has two parts SecMon Agent and SecMon Plugins. Following pluggable Filtering and Monitoring Services can be used Monitoring as a Service (SecMon EMS):

- Netflow
- Sflow
- Rawforward

Note: Support for Sflow is present only in SecMon EMS (currently disabled). An implementation of Sflow can be used as plugin by implementing the interfaces mentioned in section **Interface expected by SecMon in plugins**

2.1.1 SecMon Agent

It is a DPDK based application which loads filtering plugins and receive packets from network interface through DPDK library. Received packets then are provided to all the plugins which are registered.

Code Location : <base location>/SecMon/secmon_agent

2.1.1.1 Interface between SecMon Agent and DPDK library

SecMon application primarily uses following DPDK interfaces to read packets:

- **rte_mempool_create**
- **rte_eth_dev_configure**
- **rte_eal_remote_launch**

2.1.2 SecMon Plugin

Each plugin implemented as a dynamic loadable library is loaded by DPDK packet filter application. Plugin from /opt/secmon/plugins. Its architecture can be configured to use software rings or not for packets buffering according to requirement.

Code Location : <base location>/SecMon/secmon_plugin

2.1.2.1 Interface expected by SecMon in plugins

SecMon expects some interface functions to be present in plugins. The interfaces are listed below with their significance:

- **int init()**
- **int deinit()**
- **int config()**
- **Int receive_data()**
- **Int receive_from_secmon(struct rte_mbuf)**

.

For more details, refer **Virtual_Network_Broker_Functional_Specification.pdf**

2.2 SecMon EMS VM

SecMon EMS is python based server implemented using Django framework. It interfaces with EMS policy enforcer i.e. Horizon GUI / CLI and SecMon EMS VM over REST Based Interface.

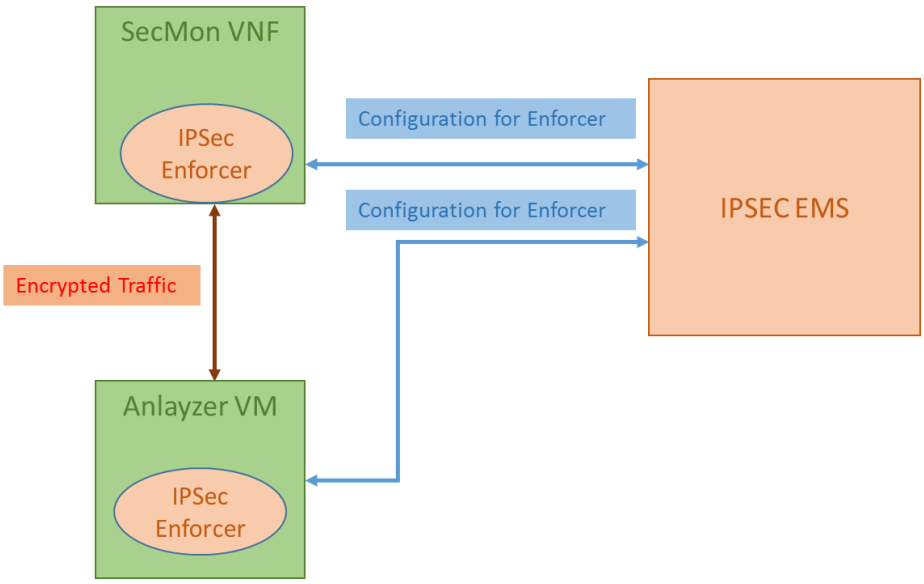
Code Location : <base location>/SecMonEMS/

2.2.1 Components of SecMon EMS

- Django Server
  Server handle requests done by client and respond with proper response. Server also inform EMS notifier about the changes which in turn inform SecMon.
- Consul DB Database
  This is distributed database in which data is stored as key/value pairs.
  SecMon EMS Notifier
  Whenever user changes the plugins configurations in Django Server. Django server notify the SecMon plugins of configurations changes through EMS notifier.

All SecMon EMS server APIs are described in document **SecMon_EMS_Rest_Documentation.pdf**

3. IPSec Function - Architecture



IPSec Function Overview

3.1 IPSec EMS

It provides User Interface for the easy configuration of IPSec and exposes REST APIs towards Enforcers.

Location : <base location>/ipsec_ems

3.2 IPSec Enforcers

Enforcers fetches IPsec configurations from IPsec EMS using RESTful API's exposed by IPsec EMS. These enforcers use the configuration for setting up IPSec between SecMon VNF (s) and Analyser(s) by configuring ipsec.secrets and ipsec.conf files.

Location : <base location>/ipsec_enforcer