Welcome to OPNFV Security Guide

Release arno.2015.1.0 (50ad84d)

OPNFV

August 10, 2016

This guide seeks to inform operators who to secure and maintain the security of the OPNFV Platform and its components.

Contents:

CHAPTER

ONE

INTRODUCTION

The OPNFV Security Guide is the collaborative work of many individuals, involved in both the OPNFV Security Group and the wider OPNFV community.

The purpose of this guide is to provide the best practice security guidelines for deploying the OPNFV platformm. It is a living document that is updated as new changes are merged into it's repository.

1.1 Background

Pre-virtualization security protection was largely centered on the network. Malicious attacks from hostile machines, would seek to exploit network based operating systems and applications, with the goal of compromising their target node.

Physical security had always been a much simpler business, with most focus on the secure access of the data center hardware. In-turn security was built up in layers (defense in depth) where machines would be daisy chained with network cables via security appliances to provide controlled segmentation and isolation. This form of security was built upon the principle of an 'air gap' being present, whereby machines were separate physical units, joined largely by the network stack.

With the advent of virtualization (namely the hypervisor), new attack vectors have surfaced as the 'air-gap' is no longer key design aspect for security. Further to this elements orchestation nodes and network controllers lead to an even wider attack surface:

- Guests breaking isolation of the hypervisor.
- Unauthorized access and control of supporting orchestration nodes.
- Unauthorized access and control of supporting overlay network control systems.

The hypervisor and the overlay network have now become the 'Achilles heel' whereby all tenant data isolation is enforced within the hypervisor and its abstraction of hardware and the virtualized overlay network.

This guide has been formulated, in order to assist users of the OPNFV platform in securing an Telco NFV / SDN environment.

1.2 Acknowledgements

CHAPTER

TWO

COMPUTE SECURITY

2.1 DAC & MAC Controls

2.2 Trusted Compute

Trusted compute is centered on insuring the complete lifecycle of a VM, and the VM's underlying infrastructure is of a 'trustful' state.

Trusted computing in a cloud environment

To ensure overall security in an OPNFV deployment, both the launch and the operation of virtualized resources need to be secure. To build a trusted computing in a cloud environment the following core features are essential:

- boot integrity - the hardware platform can guarantee a trustworthy RoT for the overall cloud environment
- secure management of VMs – to secure the launch and migration of VMs in the cloud environment

In this section we will cover some aspects of what is considered compute security, such as secure/trusted boot, although of course these can be extended to other actors such as neutron networking nodes.

2.2.1 Secure Boot

Secure boot, a UEFI-based feature that has become controversial lately, ensures that nodes in an OPNFV deployment boot only software that is trusted by the admin or end user.

In order to understand the secure boot procedure, we need to explain the related technology and specification.

Unified Extensible Firmware Interface (UEFI)

UEFI is a specification intended to be the replacement and improvement on the old BIOS (Basic Input/Output System).

One UEFI-based feature that has become controversial lately is the secure boot feature.

The UEFI specification is a standard that's handled by a non-profit organization with representatives of Intel, AMD, Microsoft, Apple, Dell, HP, IBM and others, called the Unified EFI Forum.

UEFI supports 32 and 64 bit processors and can be used with Itanium, x86, x64 and ARM processors.

Trusted Execution Environment (TEE) vs Trusted Platform Mobile (TPM)

Two main components of platform security:

- Trusted Execution Environment
- Trusted Platform Module

These are not designed as a replacement of the other. TEE is the bulletproof safe, while TPM is the 128-digit combination lock for the safe. Both are needed to ensure the safe is protected.

TPM is a dependency of TEE but not the other way around.

The TPM is where TEE will store the measurements - hash of components - of the platform.

If TEE is not supported by a platform but a TPM is still present you still have all these features:

- Integrity measurement – securely measure the platform's components (hashes stored within the TPM)
- Authenticated boot – a process by which a platform's state (the sum of its components) is reliably measured and stored
- SRTM - Static Root of Trust for Measurements
- Sealed Storage - encrypt data based on the current state of the platform or in other words, what has been measured (the PCR hash values stored in the TPM) - seal operation
- Attestation - securely report to other parties the state of the platform

2.2.2 Trusted Compute Pools

Trusted Boot

Trusted boot (tboot) is an open source, pre- kernel/VMM module that uses Intel(R) Trusted Execution Technology (Intel(R) TXT) to perform a measured and verified launch of an OS kernel/VMM. The root of trust is in the hardware and a TPM is required. Compute nodes in an OPNFV deployment boot with Intel TXT technology enabled.

Read more about Trusted Boot and Trusted Computing.

Trusted Execution Environments (TEE)

The Trusted Execution Environment is an isolated execution environment which provides higher level of security such as isolated execution, integrity of Trusted Applications along with confidentiality of their assets.

Goals of a Trusted Execution Environments:

- Isolated Execution
- Secure Storage
- Remote Attestation
- Secure Provisioning
- Trusted Path

TEE platforms/implementations

- Intel's TXT (Trusted Execution Technology)
- AMD Secure Execution Environment
- ARM TrustZone

All three of these TEE implementations provide a virtualized Execution Environment for the secure OS and applications.

To switch between the secure world and the normal world, Intel provides SMX Instructions, while ARM uses SMC. Programmatically, they all achieve very similar results.

Read more about Trusted Execution Environments here.

NIST SP800-147 , is a guidelines for firmware security, to ensure that the firmware itself is secure.

Read more about "Trusted compute pools", in the OpenStack Security Guide.

CHAPTER

THREE

NETWORK SECURITY

3.1 Neutron Security

CHAPTER

FOUR

HOW TO CONTRIBUTE

Anyone is welcome to make additions, raise bugs, and fix issues within this Documentation. To do so, you will however need to first get an enviroment set up.

4.1 Development Environment

All project data such as formatting guidelines, and upstream mapping is documented via sphinx which uses reStructuredText

It is recommended that you use a python virtualenv to keep things clean and contained.

4.1.1 VirtualEnv

Use of a virtual environment is recommended, as not only is it a quick easy form of getting the needed modules in place, it isolates the module versions to a project.

From within your inspector directory, set up a new virtualenv:

```
virtualenv venv
```

Activate the new virtual environment:

```
source venv/bin/activate
```

Install requirements:

```
pip install -r requirements.txt
```

4.1.2 Sphinx Basics

To get started with sphinx, visit the main tutorial which will provide a primer *http://sphinx-doc.org/tutorial.html*

Hack your changes into opnfv-security-guide/source

To compile changes:

    make html

From here you can run a basic python web server or just navigate to the file:///<repo>/opnfv-security-guide/build/html/index.html in your browser

CHAPTER

FIVE

AUDIT

Requirements references related to OPNFV Audit

5.1 Source information

http://www.etsi.org/deliver/etsi_gs/NFV-INF/001_099/003/01.01.01_60/gs_NFV-INF003v010101p.pdf
http://www.etsi.org/deliver/etsi_gs/NFV-INF/001_099/004/01.01.01_60/gs_NFV-INF004v010101p.pdf

- ETSI GS NFV-SEC 003 V1.1.1 (2014-12)
  - Network Functions Virtualisation NFV);
  - NFV Security; Security and Trust Guidance
  - NFV-SEC-003.
- ETSI GS NFV 004 V1.1.1 (2013-10)
  - Network Functions Virtualisation (NFV);
  - Virtualisation Requirements
  - NFV-SEC-004.

5.1.1 Requirements on Auditing framework

Audit records shall be maintained within protected binary logs so that the record of malicious actions cannot be deleted from the logs.

5.1.2 Necessary auditable events

- access control management
  - Adding a user account
  - Modifying user account
  - Deleting a user account
  - login event
  - logout event
  - IP whitelisting update
  - IP blacklisting update

- VNFC Creation
  - The instantiation of a newly-defined VNFC
  - The instantiation of a VNFC with pre-configured state
  - The cloning of an existing VNFC
- VNFC Deletion
  - The deletion of VNFC and of all of its instances (e.g. snapshots, backups, archives, cloned images)
- Software management
  - patching e.g. opreating system, drivers, VM components
  - dynamic updates to the configuration e.g. DNS, DHCP
  - application software updates
  - software component updates
- Data management
  - Root level access to NFVI file system
  - User level access to NFVI file system
  - Secured wipe, disk and memory
  - Verified destruction
  - Certificate revocation
- VNFC Migration
  - VNFC original host identity
  - VNFC target host identity
  - high availability
  - recovery
  - data-in-motion changes
- Other VNFC Operational State Changes
  - Hibernation, sleep, resumption, abort, restore, suspension
  - Power-on and power-off (either physical or virtual)
  - Integrity verification failure, crash and OS compromise
- VNFC Topology Changes
  - Network IP address and VLAN updates
  - Service chaining
  - Failover and disaster recovery
- traffic inspection
  - enabling virtual port mirroring
  - enabling hypervisor introspection
  - enabling in-line traffic inspection
  - application insertion

- initial provisioning of a public/private key pair
  - Self-generation of key pairs for later validation by an external party:
    - Certificate Authority
    - VNFM
  - Provision by trusted party
    - network
    - storage
  - Injection by hypervisor