Deliverable D3.8
5G-PPP Security Enablers Documentation (v2.0)
Enabler BootstrappingTrust

Project name	5G Enablers for Network and System Security and Resilience	
Short name	5G-ENSURE	
Grant agreement	671562	
Call	H2020-ICT-2014-2	
Delivery date	31.08.2017	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	Nicolae Paladi, nicolae.paladi@ri.se Linus Karlsson, nicolae.paladi@ri.se	

This part will be removed before the final edition and publication

Document Version	Date	Change(s)	Author(s)
0.1	02.06.2017	Created template	Felix Klaedtke
0.2	01.08.2018	Draft ready for review	Nicolae Paladi
0.3	8.8.2017	Reviewed	Aleksi Dahl
0.4	14.8.2017	Fixes based on previous review	Linus Karlsson

Foreword

5G-ENSURE belongs to the first group of EU-funded projects which collaboratively develop 5G under the umbrella of the 5G Infrastructure Public Private Partnership (5G-PPP) in the Horizon 2020 Programme. The overall goal of 5G-ENSURE is to deliver strategic impact across technology and business enablement, standardization and vision for a secure, resilient and viable 5G network. The project covers research and innovation - from technical solutions (5G security architecture and testbed with 5G security enablers) to market validation and stakeholders engagement - spanning various application domains.

This manual is part of the project's deliverable D3.8. It describes how one of the security enablers that are developed within the work package 3 of the 5G-ENSURE project is installed and administrated.
Furthermore, this manual contains a user guide of the respective security enabler.

Disclaimer

Copyright notice

Contents

1 Introduction

The physical infrastructure supporting 5G networks will be multiplexed among multiple tenants with own virtual environments, similar to the cloud infrastructure model. In this model, SDN allows tenants to configure complex topologies with rich network functionality, managed by a network controller [1]. The availability of a global view of the forwarding plane enables advanced controller capabilities – from pre-calculating optimized traffic routing to managing software applications that replace hardware middleboxes with virtual network functions (VNFs). However, these capabilities also turn the controller into a valuable attack target: once compromised, it can provide the adversary with complete control over the network [2]. Furthermore, the global view itself is security sensitive: an adversary capable of impersonating network components – such as virtual switches or VNFs -- may distort the network controller's view of the forwarding plane or influence the network-wide routing policies. Virtual network components – such as virtual switches and VNFs – are security sensitive elements in SDN deployments. They operate on commodity operating systems (OS) and are often assigned the same trust level and privileges as physical switches or middleboxes – specialized, hardware components with compact embedded software. Commodity OS with large code bases are likely to contain multiple security flaws which can be exploited to compromise virtual network components. Such risks are accentuated by the extensive control a cloud provider has over the compute, storage and networking infrastructure of its tenants. It is therefore essential to appraise the trustworthiness of the virtual network components prior to enrolling them into the network infrastructure [3,7].

This enabler aims to reduce the risk to the integrity of the virtual switches and VNFs enrolled in the SDN deployment, as well as to ensure the integrity and confidentiality of the communication between SDN components and the network controller. To this end, a protocol has been implemented to attest the integrity of virtual network components and provision authentication certificates prior to enrolling them into the virtual network infrastructure. The protocol can be applied both to the elements of the forwarding plane [6] as well as to virtual network functions executed in operating system level virtualization containers [8]. We chose to implement the enabler with the Floodlight network controller, considering that this network controller is well documented and features support for TLS on the north-bound interface (i.e. between network controller and virtual network functions).

This manual describes how the enabler is installed, administered and operated. Currently, the enabler does not rely on hardware support for the isolated execution and remote integrity verification mechanisms implemented by Intel Software Guard Extensions (SGX) [4]; however, if SGX support is present on the platform, the enabler can leverage SGX functionality to offer the intended security features.

The manual is organized as follows. In Section 2 we describe the steps to install and administer the enabler. In Section 3 we describe the operation of the enabler. In Section 4 we describe two basic tests for the Bootstrapping Trust enabler. In Section 5 we list the abbreviations used throughout the manual of the Bootstrapping Trust enabler.

2 Installation and Administration Guide

2.1 System Requirements

2.1.1 Host infrastructure:

- 2 virtual instances
  - Instance 1 flavor: vMedium
  - Instance 2 flavor: vMedium

2.1.2 Library support:

The following packages should be installed in all instances: floodlight-rise, intelsgxpsw, intelsgxsdk, rise-bootstrappingtrust

2.1.3 Network architecture:

The two VMs should be connected and reachable from each other. The IP addresses should be static and known.

2.1.4 Miscellaneous

The IP addresses need to be known since configuration files in the rise-bootstrappingtrust package must be updated with the IP addresses before running the deployed software.

2.2 Enabler Configuration

The enabler configuration consists of two main steps: integrity measurement configuration and enabler configuration. We describe the installation and configuration procedures in the below sequence. We assume that the underlying Operating System is Linux Ubuntu 16.04

2.2.1 Integrity Measurement Configuration

The *Integrity Measurement Architecture* must be enabled and configured in order to obtain the initial measurement:

```
mount -t securityfs security/sys/kernel/security

ls /sys/kernel/security/integrity/ima
```

If the '`ima`' directory exists, then the IMA is enabled.

Next, the 'ima_tcb' flag must be added to the to the boot options in order to collect IMA measurements:

```
vi /etc/default/grub > GRUB_CMDLINE_LINUX_DEFAULT="quiet splash ima_tcb
ima=on"
```

The default IMA policy will measure all executed programs (files for which the `exec` system call has been invoked), as well as files or devices that have been mapped to the memory for execution (using the `mmap` system call), and all files opened for read by uid=0.

The SMACK mandatory access control (MAC) framework can be used to measure and monitor the integrity of individual files in Linux.

To verify whether SMACK is enabled, execute:

```
grep CONFIG_SECURITY_SMACK /boot/config-`uname -r`
```

The expected outcome is:

```
CONFIG_SECURITY_SMACK=y
#CONFIG_SECURITY_SMACK_BRINGUP is not set
CONFIG_SECURITY_SMACK_NETFILTER=y
```

Next, the operating system must be configured to mount SMACK at startup.

```
vi /etc/fstab
smackfs /sys/fs/smackfs smackfs defaults 0 0
```

Next, the boot parameters must be updated:

```
vi /etc/default/grub

security=smack ima_policy rootflags=i_version

update-grub
```

Parameter explanation:

- `smack` – instruct IMA to use the Smack security module;
- `ima_policy` – instruct IMA to use the specified Ima policy;
- `rootflags=i_version` – mount the root directory with i_version support, allow IMA to re-measure files to detect changes.

Enable file measurements by defining a SMACK label as follows:

```
echo "_ M rwxa" > /sys/fs/smackfs/load2
```

The files to be measured can be marked by setting the "M" attribute:

```
setfattr -n security.SMACK64 -v M <file>
```

Linux IMA re-measures the files each time they are accessed and (if needed) extends the measurements in:

```
/sys/kernel/security/ima/ascii_runtime_measurements
```

2.2.2 Enabler Configuration

Two configuration files on each of the hosts involved on the communication, should contain the IP address of the other host:

```
/opt/bootstrappingtrust/Certs/rh_host
/opt/bootstrappingtrust/Certs/container_host
```

2.3 Enabler Installation

Installing the following packages deploys the enabler:

```
floodlight-rise, rise-bootstrappingtrust
```

3 User and Programmer Guide

The goal of the Bootstrapping Trust enabler is to verify the integrity of the virtual network components and provision the authentication certificates to allow their enrollment into the deployment.

3.1 User Guide

The enabler should be used prior to the operation of the SDN deployment.

Once the bootstrapping trust application is started:

```
sudo ./app
```

it will listen to incoming connections and follow the protocol to

1. Verify the integrity of the labeled files on the host;
2. Verify the integrity of the VNF container.
3. Provision the authentication credentials.

3.2 Programmer Guide

The enabler is not programmable.

4 Unit Tests

4.1 Unit Test 1

4.1.1 Summary:

This unit tests verifies that Floodlight has been correctly installed, and can listen for connections.

4.1.2 Preconditions:

The floodlight-rise package is installed on the host.

4.1.3 Test actions:

Launch a terminal window and issue the following commands:

```
1. cd /opt/Floodlight
2. sudo java -jar target/floodlight.jar
```

4.1.4 Expected outcome:

Floodlight starts, and displays among the few lines the following output:

"HTTPS enabled; only trusted clients permitted. Allowing secure access to REST API on port 8081"

4.2 Unit Test 2

4.2.1 Summary:

This test verifies that selected parts of the enabler can be started and listen for connections.

4.2.2 Preconditions:

The following packages are installed on the hosts:

intelsgxpsw, intelsgxsdk, rise-bootstrappingtrust

The configuration files are updated with the correct IP-addresses of hosts, as per the user manual (or currently; separately provided installation steps in the ticket for enabler deployment).

4.2.3 Test actions:

On any of the VMs, launch the Application and ensure that it can listen for incoming connections:

1. cd /opt/bootstrappingtrust/Application
2. sudo ./app

4.2.4 Expected outcome:

Expect the following line in the output

Server running on port 22222

5 Abbreviations

5G-PPP	5G Infrastructure Public Private Partnership
OVS	Open vSwitch
TEE	Trusted Execution Environment
IMA	Integrity Measurement Architecture
SGX	Software Guard Extensions
SDK	Software Development Kit
SDN	Software Defined Networking
TLS	Transport Layer Security
OS	Operating System

VNF	Virtual Network Function

6 References

[1] Open Networking Foundation (ONF). OpenFlow switch specification – version 1.3.0 (wire protocol 0x04). 2012.

[2] Porras, P., Cheung, S., Fong, M., Skinner, K., Yegneswaran, V.: Securing the software-defined network control layer. In: Proceedings of the 2015 Network and Distributed System Security Symposium (NDSS), San Diego, California (2015)

[3] N. Paladi, C. Gehrmann: Towards Secure Multi-tenant Virtualized Networks. In: Trustcom/BigDataSE/ISPA, 2015 IEEE. vol. 1, pp. 1180–1185. IEEE (2015)

[4] Anati, I., Gueron, S., Johnson, S., Scarlata, V.: Innovative technology for CPU based attestation and sealing. In: Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy. p. 10 (2013)

[5] 5G-Ensure Consortium, Deliverable 2.4 Security Architecture (draft), [Online]. Available: http://www.5gensure.eu/sites/default/files/Deliverables/5G-ENSURE_D2.4-SecurityArchitectureDraft.pdf, 2016.

[6] N. Paladi and C. Gehrmann, "TruSDN: Bootstrapping Trust in Cloud Network Infrastructure", in Proc. 12th International Conference on Security and Privacy in Communication Networks, SecureComm'16, Springer, October 2016

[7] M. Bursell, A. Dutta, H.-L. Lu, M.-P. Odini, K. Roemer, K. Sood, M. Wong, and P. Wörndle, "Network Functions Virtualisation (NFV), NFV Security, Security and Trust Guidance, v.1.1.1." http://www.etsi.org/deliver/etsi_gs/NFV-SEC/001_099/003/01. 01.01_60/gs_NFV-SEC003v010101p.pdf, 2014. Accessed: 2016-11-20.

[8] N. Paladi, L. Karlsson, "Safeguarding VNF Credentials with Intel SGX", in *Proc. 2017 conference on ACM SIGCOMM*, ACM, 2017 *(In press)*

.