Deliverable D3.4
5G-PPP Security Enablers Documentation (v1.0)

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	30.09.2016	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	IT INNOV: Stephen Phillips, Juri Papay, Mike Surridge NEC: Felix Klaedtke, Mihai Moraru ORANGE: Jean-Philippe Warry, Ghada Arfaoui SICS: Markus Ahlström, Simon Holmberg, Martin Svensson, Rosario Giustolisi, Nicolae Paladi TASE: Gorka Lendrino TCS: Frederic Motte TS: Theo Combe, Cyrille Martins TIIT: Madalina Baltatu, Luciana Costa, Dario Lombardo UOXF: Piers O’Hanlon VTT: Pekka Ruuska, Jani Suomalainen, Olli Mämmelä, Kimmo Ahola	

Executive summary

This document contains the manuals of the first software releases of the 5G security enablers that are developed within the 5G-ENSURE project. Each enabler has its own separate manual, which comprises the following three main parts: (1) an installation and administration guide, (2) a user and programmer guide, and (3) a description of unit tests for the enabler's software. The enablers' manuals are an important input for the enablers' deployment in the project's testbed (WP4), where the enablers will be analyzed and evaluated.

Note that the software of the project's security enablers is part of the accompanying deliverable D3.3 "5G-PPP security enablers sw release (v1.0): reference implementations for the first set of the enablers." Both deliverables D3.3 and D3.4 complete the prior WP3 deliverables of the first year of the project, namely, D3.1 "5G-PPP security enablers technical roadmap (early vision)" and D3.2 "5G-PPP security enablers open specifications (v1.0)".

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, standardisation and vision for a secure, resilient and viable 5G network. The project covers research & innovation - from technical solutions (5G security architecture and testbed with 5G security enablers) to market validation and stakeholders engagement - spanning various application domains.

This deliverable (D3.4) contains the manuals of the first software releases of the security enablers developed within WP3 of the 5G-ENSURE project. The enablers' software itself is part of the accompanying deliverable D3.3 "5G-PPP security enablers sw release (v1.0): reference implementations for the first set of the enablers."

Disclaimer

Copyright notice

Introduction

This document contains the manuals of the first software releases of the 5G security enablers that are developed within the 5G-ENSURE project. Table 1 summarizes the project's 5G security enablers for the first year with a software release as described in the project's deliverable D3.1 "5G-PPP security enablers technical roadmap (early vision)". The table provides the enablers' (code) names with the category to which they belong to (i.e., AAA, privacy, trust, security monitoring, and network management and virtualization isolation). A short description of the features of the first software release for each enabler is also provided in the table.

Note that the software of the project's security enablers is part of the accompanying deliverable D3.3 "5G-PPP security enablers sw release (v1.0): reference implementations for the first set of the enablers." Both the enablers' software releases and their manuals complete the prior deliverables of the project's "release one" security enablers, namely, D3.1 "5G-PPP security enablers technical roadmap (early vision)" and D3.2 "5G-PPP security enablers open specifications (v1.0)." Furthermore, note that new security enablers will be provided within the second year of the project, which will complement the ones of the first year of the project. Additional features for the security enablers in Table 1 are also planned for the second year of the project to mature, extend, and enrich the enablers' of the project's first year. The manuals will be updated accordingly. The enablers' manuals are an important input for the enablers' deployment in the project's testbed, where they will be analyzed and evaluated.

Furthermore, note that the feature of the security enabler "system security state repository" that is delivered in the first year of the 5G-ENSURE project is an ontology. This enabler does not have a manual in this deliverable, since it does not comprise software in the first year of the project. For the enabler's roadmap, see the project deliverable D3.1 (page 64). The accompanying software of this enabler together with its manual will be released in the project's second year.

The remainder of this document comprises the manuals of the first software releases of the 5G-ENSURE security enablers. Each enabler has its own manual, which consists of the following parts. Section 1 is an introduction to the security enabler. The Sections 2 to 4 are the main part of an enabler's manual. Section 2 contains the enabler's installation and administration guide, including the system requirements and troubleshooting. Section 3 contains the enabler's user and programmer guide. In case the enabler is not programmable, the programmer guide is empty. Section 4 describes some unit tests for the enabler's software. The abbreviations used in the enabler's manual, acknowledgements, and references are given at the end of a manual in separate sections, which are omitted if empty. Some of the enablers' manuals also comprise an appendix, which provides additional details about the enabler, its use, or its implementation.

Table 1 List of 5G security enablers of the first year of the 5G-ENSURE project with a software release.

Category	**Security enabler**	**Features of the first software release**
AAA	Internet of Things	Group-based AKA
	Fine-grained authorization	Basic authorization in satellite systems Basic distributed authorization enforcement for RCDs
Privacy	Privacy enhanced identity protection	Encryption of long term identifiers (IMSI public-key based encryption)
	Device identifier(s) privacy	Enhanced privacy for network attachment protocols
Trust	VNF certification	VNF trustworthiness evaluation
	Trust metric	Trust metric based network domain security policy management
	Trust builder	Graphical editor and 5G asset model
Security Monitoring	Generic collector interface	Collection and distribution of reports (log and events)
	Security monitor for 5G micro-segments	Complex event processing framework for security monitoring and inferencing
	Satellite network monitoring	Pseudo real-time monitoring and threat detection in 5G integrated satellite and terrestrial systems
	PulSAR: proactive security analysis and remediation	5G specific vulnerability schema
	System security state repository	Deployment model ontology
Network Management and Virtualization Isolation	Access control mechanisms	Southbound reference monitor for the ONOS controller
	Component-interaction audits	Basic OpenFlow compliance checker
	Bootstrapping trust	Integrity attestation of virtual network components
	Micro-segmentation	Creation and deletion of micro-segments, adding nodes to a micro-segment, and deleting nodes from a micro-segment

Deliverable D3.4
5G-PPP Security Enablers Documentation (v1.0)
Enabler Internet of Things

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	30.09.2016	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	SICS: Markus Ahlström, Simon Holmberg, Rosario Giustolisi, Martin Svensson	

Contents

1 Introduction

The number of machine-type communications (MTC) devices is predicted to increase enormously with 5G. This will lead to an increased signalling that may congestion and increase the network access latency. Sport events is one example use case where the consequences outlined above may happen. During sport events, a large number of sensors and tracking devices are used to increase the user experience [1]. By grouping devices on the basis of features such as ownership or location, the signalling of the Authentication and Key Agreement (AKA) can be reduced.

The Internet of things enabler has four features, release 1 includes the “Group-based authentication” feature. Section 2.1 of Deliverable 3.2 [1] describes the group-based AKA enabler feature of which a prototype implementation has been developed for release 1. Basically, the enabler consists of a new AKA protocol that decreases both network access latency and data traffic between the MME and the HSS. The new protocol allows MTC devices to perform AKA procedures without the need that the MME communicates with the HSS. This is achieved by using inverted hash trees containing authentication information. When the first MTC device attaches, *Case A* is performed: the MTC device is authenticated as in EPS AKA but the HSS includes sub-root nodes in the group trees along with the authentication vector (AV). When the remaining MTC devices in the group perform *Case B* of the protocol, individual authentication parameters are derived by the MME from the sub-root nodes so that the HSS does not have to be contacted, which is beneficial during roaming.

2 Installation and Administration Guide

The feature is implemented on top of OpenAirInterface which is a framework providing EPC and EUTRAN Open Source implementations [2]. In general, the documentation provided by OAI is useful also for the enabler. In order to make it possible for readers to take as much advantage of the OAI documentation as possible, the OAI documentation or the OAI software project in general will often be related to in the text below.

The software is divided into two different projects: one which simulates the UE and the eNodeB, Openairinterface5G, and one which implements a subset of the 3GPP Release 10 of the EPC, Openair-cn. The EPC part is further divided into two different entities that are built and that run separately: the MME and the HSS. Openairinterface5G offers a variety of emulation scenarios for UE and eNodeB deployment. The scenario used by the enabler is called “oaisim with S1”, and simulates the UE and eNodeB entities with S1 interface capabilities. In Figure 1, the OAI deployment is illustrated. The OAI platform is in continuous development and the enabler code was branched from the OAI code in February 2016.

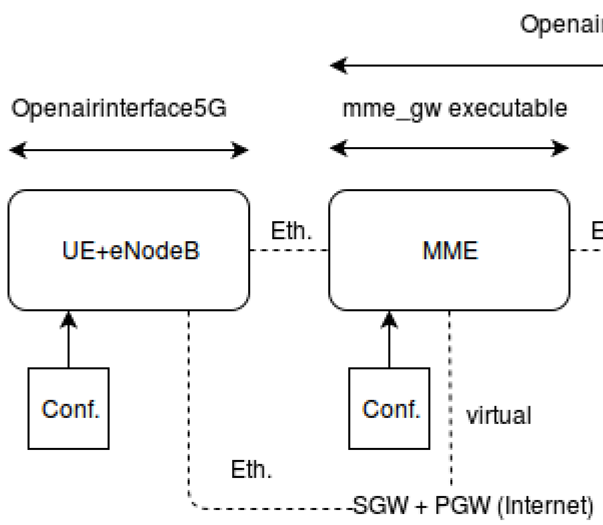


Figure 1 Illustration of the considered OAI deployment.

The installation of the enabler begins by first cloning the Openairinterface5G and Openair-cn Git projects and then applying the patch available in the 5G-ENSURE software repository. The patch file is named `groupaka-0.1.deb` and is released as a debian package. The projects should be built using the build scripts included in the OAI platform. The scripts will install all necessary dependencies.

The OAI project can be found in [2].

OAI provides documentation for the deployment and configuration of Openair-cn [3] and Openairinterface5G [4].

The parameters of the OAI UE such as IMSI and the long-term key are hardcoded, so to change a parameter one must first change it in the code and then recompile the code. The source code file with the parameters is located at `openairinterface5G/openair3/NAS/TOOLS/usim_data.c`. If changes to the OAI UE parameters are done in this file, the HSS database for subscriber data must also be updated. Changing the parameters of the HSS database is done in the MySQL server installed through the build scripts. The database is called OAI_DB and represents the AuC of the HSS. However, changing these parameters is currently discouraged since the code is only tested for the default values, and below it is assumed that they are not changed.

2.1 System Requirements

Since the enabler is a patch to the OAI Git project, Git is a prerequisite for installing the enabler.

The enabler is built on a specific version of OAI and has been tested on the operating system Ubuntu 14.04 LTS (Trusty Tahr) using kernel Linux 3.19 on Intel x86 64 bits platforms. Consequently, we recommend to use the same Ubuntu release and kernel for the enabler.

The enabler was developed and tested using the Generation 3 Intel Core i7 processor with 4GB RAM. But the OAI software has also been tested on the following processor families: Generation 3 Intel Core i5, Generation 2 Intel Xeon and Intel Atom Rangely; hence, the enabler should also work with these processor families. Note that the current OpenAirInterface software requires Intel architecture based PCs.

The entities used by the enabler can run on the same or in multiple different hosts. The MME and HSS executables can run on a single host or on two hosts, one for each entity. The UE and the eNodeB must run together on a single host.

2.2 Enabler Configuration

There are three configuration files which are used by the OAI deployment. The two configuration files which are used by the enabler are:

```
openairinterface5G/targets/PROJECTS/GENERIC_LTE_EPC/CONF/enb.band7.
generic.oaisim.local_mme.conf

openair-cn/BUILD/EPC/epc.conf.in
```

The configuration used in this document is specified in Figure 3 for Openairinterface5G and in Figure 4 for Openair-cn. Our configuration is set according to the local network architecture seen in Figure 2. However, the parameters can be changed according to the user's own network, provided that the interfaces which are used are mirrored in the configuration files. In the example deployment, virtual Ethernet interfaces were used for the interfaces between the MME, eNodeB and UE entities and the enabler was run on a single host.

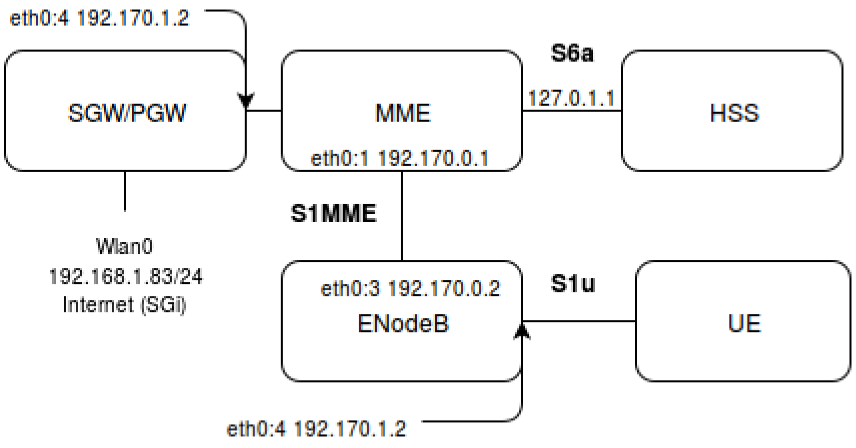


Figure 2 Illustration of example network architecture for the enabler on single host.

```
////////// MME parameters:
    mme_ip_address      = ( { ipv4       = "192.170.0.1";
                              ipv6       = "192.170.0::17";
                              active     = "yes";
                              preference = "ipv4";
                            }
                          );

    NETWORK_INTERFACES :
    {
        ENB_INTERFACE_NAME_FOR_S1_MME            = "eth0:3";
        ENB_IPV4_ADDRESS_FOR_S1_MME              = "192.170.0.2/24";

        ENB_INTERFACE_NAME_FOR_S1U               = "eth0:4";
        ENB_IPV4_ADDRESS_FOR_S1U                 = "192.170.1.2/24";
        ENB_PORT_FOR_S1U                         = 2153; # Spec 2152
    };
```

Figure 3 The network configuration of the UE and the eNodeB in the example deployment.

We recommend to keep the default parameters for OAI UE.

```
NETWORK_INTERFACES :
    {
        MME_INTERFACE_NAME_FOR_S1_MME         = "eth0:1";                        # YOUR
NETWORK CONFIG HERE
        MME_IPV4_ADDRESS_FOR_S1_MME           = "192.170.0.1/24";           # YOUR NETWORK
CONFIG HERE

        MME_INTERFACE_NAME_FOR_S11_MME        = "none";
        MME_IPV4_ADDRESS_FOR_S11_MME          = "0.0.0.0/24";
    };
};

S-GW :
{
    NETWORK_INTERFACES :
    {
        SGW_INTERFACE_NAME_FOR_S11              = "none";
        SGW_IPV4_ADDRESS_FOR_S11                = "0.0.0.0/24";

        SGW_INTERFACE_NAME_FOR_S1U_S12_S4_UP    = "eth0:2";                        # YOUR
NETWORK CONFIG HERE
        SGW_IPV4_ADDRESS_FOR_S1U_S12_S4_UP      = "192.170.1.1/24";                # YOUR
NETWORK CONFIG HERE
        SGW_IPV4_PORT_FOR_S1U_S12_S4_UP         = 2152;                            # PREFER NOT
CHANGE

        SGW_INTERFACE_NAME_FOR_S5_S8_UP         = "none";                          # DO NOT
CHANGE
        SGW_IPV4_ADDRESS_FOR_S5_S8_UP           = "0.0.0.0/24";                    # DO NOT
CHANGE
    };
};

P-GW =
{
    NETWORK_INTERFACES :
    {
        PGW_INTERFACE_NAME_FOR_S5_S8          = "none";                            # DO NOT
CHANGE
        PGW_IPV4_ADDRESS_FOR_S5_S8            = "0.0.0.0/24";                      # DO NOT
CHANGE

        PGW_INTERFACE_NAME_FOR_SGI            = "wlan0";                            # YOUR
NETWORK CONFIG HERE
        PGW_IPV4_ADDRESS_FOR_SGI              = "192.168.1.93/24";                # YOUR
NETWORK CONFIG HERE
        PGW_MASQUERADE_SGI                    = "yes";                             # YOUR
NETWORK CONFIG HERE
    };

    IP_ADDRESS_POOL :
    {
        IPV4_LIST = (
                      "192.188.0.0/24",                                            # YOUR
NETWORK CONFIG HERE
                      "192.188.1.0/24"                                             # YOUR
NETWORK CONFIG HERE
                    );
        IPV6_LIST = (
                      "2014:02:26::0/120"                                          # YOUR
NETWORK CONFIG HERE
                    );
    };

    DEFAULT_DNS_IPV4_ADDRESS     = "192.168.106.12";                                 # YOUR
NETWORK CONFIG HERE
    DEFAULT_DNS_SEC_IPV4_ADDRESS = "192.168.12.100";                                 # YOUR
NETWORK CONFIG HERE
};
```

Figure 4 The network configuration of the Openair-cn in the example deployment.

2.3 Enabler Installation

In the installation process described below, the enabler runs on a single host. We assume that MySQL and phpMyAdmin are **not** installed prior to the installation.

2.3.1 Fetching the Git projects

Install Subversion and Git:

```
sudo apt-get install subversion git
```

Configure Git with name and email address.

Add certificate in order to fetch projects. This must be done as root user:

```
sudo su root

echo -n | openssl s_client -showcerts -connect gitlab.eurecom.fr:443 2>/dev/null | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' >> /etc/ssl/certs/ca-certificates.crt

su <username>
```

Change directory to the directory where you would like the Git projects to be stored.

Clone the first project:

```
git clone https://gitlab.eurecom.fr/oai/openairinterface5g.git
```

Change directory to the `openairinterface5g` directory and branch the Openairinterface5g project to the correct commit:

```
cd openairinterface5g

git checkout -b groupaka5g 5c9b40e1bbfedd069b67f63aa933058e18b7dda4

cd ..
```

Clone the second project:

```
git clone https://gitlab.eurecom.fr/oai/openair-cn.git
```

Change directory to the `openair-cn` directory and branch the Openair-cn project to the correct commit:

```
cd openair-cn

git checkout -b groupakacn 1f905b800d8d7398f974741be9c9cf3abe75ad18
```

2.3.2 Patching the projects

Retrieve the debian package containing the group-based AKA patch `groupaka-0.1.deb` from the 5G-ENSURE software repository.

Install the debian package:

```
sudo dpkg –i groupaka-0.1.deb
```

Apply the group-based AKA patch:

```
groupaka <path1> <path2>
```

where <path1> is the full path to the `openair-cn` folder and <path2> is the full path to the `openairinterface5g` folder.

2.3.3 Build Openair-cn

Note: The default root password for MySQL chosen by OAI is "linux". If you choose another password, make sure to configure the file `openair-cn/OAI_HSS/conf/hss.conf.in` accordingly, before the build process.

Run the initial script for dependencies and kernel module installation:

```
cd openair-cn/SCRIPTS
/build_epc -i
```

Agree to install for each dependency.

When prompted for the choice of MySQL 'root' password, choose a password and enter it.

Caution: When prompted for phpMyAdmin configuration, make sure the MySQL server is running by opening a new terminal and running:

```
sudo /etc/init.d/mysql start
```

Continue with the phpMyAdmin configuration by choosing Apache for your web server.

Choose "yes" when asked to configure phpMyAdmin.

When prompted for the MySQL administrative user password, enter the password that you have chosen.

When prompted for application password enter: admin

Continue until installation is complete.

2.3.4 FQDN of MME configuration

Add a fully qualified domain name (FQDN) for the MME to `/etc/hosts`:

```
sudo nano /etc/hosts
```

Insert the following line:

```
127.0.1.1 <computername>.openair4G.eur <computername>
```

Change `openair4G.eur` with your own domain settings.

In MySQL, change the table `mmeidentity` of database `oai_db` as follows.

Login to the database:

```
mysql -u root -p
```

Enter your MySQL root password and continue:

```
use oai_db
update mmeidentity set mmehost=<computername>.openair4G.eur' where
idmmeidentity = '2';
```

Build the software by entering the following, where the -l flag means it is a local host setup:

```
./build_epc -l
```

```
./build_hss -l
```

2.3.5 Build Openairinterface5G

Enter the openairinterface5g folder and set the source:

```
cd openairinterface5g
source oaienv
```

Enter the `cmake_targets` folder and run the build script:

```
cd make_targets
./build_oai -I -g --oaisim --install-system-files
```

Agree to install for each dependency and continue until complete.

2.4 Troubleshooting

Note that for release 1, only the communications between the UE and the core network (CN) that concerns the AKA protocol are considered. Hence, any issues or errors concerning other aspects of the OAI can be ignored. In the scope of release 1, a real error occurs if the UE and the CN fail to authenticate each other or if they derive different session keys.

The command `./run_epc` will sometimes suspend its execution waiting for the superuser password even though no prompt for the password being given. If this happens, type in the superuser password and press enter. The reason that this happens is that the command runs a script that sometimes runs sudo commands.

During the installation of Openair-cn, MySQL will sometimes not start after MySQL has been installed. If this happens, open a new terminal and start it manually by entering:

```
sudo /etc/init.d/mysql start
```

3 User and Programmer Guide

3.1 User Guide

In release 1, a first prototype implementation of the group-based AKA is provided. The first time the enabler is run, Case A is performed and the group parameters are saved into a file named `group_info`. During the following runs of the enabler, the MME checks the content of the file and since exactly the same parameters are sent from the UE as the last time, the MME decides to continue the AKA in the Case B variant. All the identification and group authentication parameters are hardcoded and the code for saving the group parameters to the `group_info` file is written for those specific parameter choices. In order to let Case A be run instead of Case B, delete the file named `group_info` in the directory `openair-cn/SCRIPTS` before the next run iteration. The steps the user has to take to run the protocol are the same for Case A and Case B, and they are the following:

Open three command-line shells. Change the directory of two shells to `openair-cn/SCRIPTS`. Change the directory of the third shell to `openairinterface5g/cmake_targets/tools`. Run the following command on one of the SCRIPTS shells:

```
./run_epc -i
```

While the epc is running, run the following on the other SOURCE shell:

```
./run_hss
```

Then, go to the third shell and run the following three commands in the given order (substitute the interface names and IP addresses if you have configured them to be different than the default):

```
sudo ifconfig eth0:3 192.170.0.2/24

sudo ifconfig eth0:4 192.170.1.2/2

sudo -E ./run_enb_ue_virt_s1
```

After these commands, all the three “entities” (the UE/eNodeB, the MME and the HSS) have started. The group-based AKA should now be running. When the output on the EPC shells have been stabilised, the AKA is done. If any of the three shell executions have not exited automatically, you can exit them by using the keyboard shortcut `Control+C`.

3.2 Programmer Guide

The enabler is not programmable.

4 Unit Tests

4.1 Information about Tests

The operational procedure to run the enabler is described in the user guide above. In the first unit test the results of a Case A run are tested and in the second the results of a Case B run are tested.

4.2 Unit Test 1

If there is a file named `group_info` in `openair-cn/SCRIPTS`, then delete it. Then, run the program as described in the user guide. Check the output of the MME script and the output of the UE/eNodeB script and confirm that the following is true.

- “Success to authentify the UE” has been printed from the MME script.
- The Kasme printed from the UE/eNodeB script is identical to the one printed from the MME script.

Additionally, there should now be a file named `group_info` in `openair-cn/SCRIPTS`.

4.3 Unit Test 2

A precondition for the test is that Case A of the protocol has been run so that the file `group_info` has been created. Run the program as described in the user guide. Check the output of the MME script and the output of the UE/eNodeB script and confirm that the following is true.

- “Success to authentify the UE” has been printed from the MME script.
- The Kasme printed from the UE/eNodeB script is identical to the one printed from the MME script.

5 Abbreviations

5G-PPP	5G Infrastructure Public Private Partnership

AKA	Authentication and Key Agreement
AuC	Authentication Center
AV	Authentication Vector
CN	Core Network
EPC	Evolved Packet Core
FQDN	Fully Qualified Domain Name
HSS	Home Subscriber Server
LTE	Long-Term Evolution
MME	Mobility Management Entity
MTC	Machine-Type Communications
OAI	OpenAirInterface
eNodeB	E-UTRAN Node B
UE	User Equipment

6 References

[1] 5G-ENSURE Consortium. Deliverable D3.2: 5G-PPP security enablers open specifications (v1.0). Available online:http://www.5gensure.eu/sites/default/files/Deliverables/5G-ENSURE_D3.2-5G-PPPSecurityEnablersOpenSpecifications_v1.0.pdf 2016.

[2] Openairinterface. Open source software/hardware development for the core network (EPC) and access-network (EUTRAN) of 3GPP. Available: https://gitlab.eurecom.fr/oai

[3] Openairinterface. Documentation for the Openair-cn deployment and configuration files. Available: https://gitlab.eurecom.fr/oai/openair-cn/blob/2f301a23435b968d9a5ba4c0a06d082c63121c7a/DOCS/EPC_User_Guide.pdf.

[4] OpenAirInterface. Documentation for the Openairinterface5G deployment and configuration files. Available:https://gitlab.eurecom.fr/oai/openairinterface5g/blob/master/targets/DOCS/E-UTRAN_User_Guide.pdf

Deliverable D3.4
5G-PPP Security Enablers Documentation (v1.0)
Fine-grained Authorization

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	30.09.2016	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	TASE: Gorka Lendrino TS: Cyrille Martins	

Contents

1 Introduction

This manual describes the installation and administration guide of the “Fine-grained Authorization” enabler taking. Also, provides user and programmers guides and unit testing plans.

More and more sensitive **devices** are exposed on Internet every day. Controlling access to such devices is therefore critical, especially in industrial control systems, and requires flexible, dynamic and potentially complex access control models like **Role-Based Access Control (RBAC, see [10]) or Attribute-Based Access Control (ABAC, see [11])**. The current trend - especially on the web - is to use central authentication/authorization services from Third Parties online. This approach implies that protected resources/devices (or gateways on the resource side) have to authenticate and authorize (almost) every access requested.

To limit security **vulnerabilities** and address **technical issues** of this approach, the “Fine-grained authorization in resource-constrained devices” feature propose an alternative allowing IoT devices/resources to enforce **full offline** attribute-based **authorization** (no direct/live link to a central third-party) with ABAC capabilities, yet preserving the benefit of **central access management** from the first approach.

The main security goal of the “Fine-grained authorization in Satellite systems” feature is to support different authorization methods (RBAC/ABAC) and policies to provide basic access control to devices and services in 5G integrated satellite and terrestrial systems. This SW release covers one identity-management situation involving satellite networks in which the operator/device attaches to the satellite network to grant access to the satellite resources.

The manual is organized as follows. In Section 2, we describe enabler is installed, configured, and administrated, and in Section 3, we describe how to use the enabler. In Section 4, we provide the tests needed to cover all the features of the enabler. In Section 5, we thank people for contributions. Finally, abbreviations and references are listed in Section 6 and Section 7.

2 Installation and Administration Guide

The “Fine-grained Authorization” enabler is composed of:

- the “Fine-grained authorization in resource-constrained devices” feature, and
- the “Fine-grained authorization in Satellite systems” feature.

This section covers the system requirements, and describes how the enabler is installed and configured.

2.1 System Requirements

The “Fine-grained Authorization” enabler consists of:

- A client-side feature (e.g. resource-constrained device, satellite terminal, UE…), able to grant access control to devices and services based on an access token provided by the server-side feature. This feature shall be deployed on each device.

- A server-side feature (i.e. authorization server), able to configure the client-side features and support different authorization methods to provide access control to devices and services. This feature shall be deployed on a central server.

The basic/minimal system requirements for this enabler are that the server and the network components should be running Linux, preferably Ubuntu Server 16.04.1 LTS or newer on an x64 architecture (you can get help from the Ubuntu server guide [13]), with a network interface.

When possible, encrypt user home directory (the operative system will seamlessly mount the encrypted home directory each time the user login and automatically unmounts when the user log out of all active sessions) and set up encrypted LVM (partitions will not be readable without knowing a special key). This feature is useful to protect sensitive data in case server or hard drive gets stolen. The thief might get physical access to the hard drive, but without knowing the right passphrase, the data on the hard drive will look like random characters.

The client host(s) (i.e. 5g-fga-sat-cli01.5g-ensure.eu) in the test-bed is a vSmall virtual host with the following requirements:

- 1 CPU (x86_64)
- 2 GB RAM
- 20 GB Hard disk
- File system: ext4
- Ethernet network interface

It will act as a resource-constrained-device (e.g. IoT), satellite terminal…

The server host (i.e. 5g-fga-sat-srv01.5g-ensure.eu) in the test-bed is a vMedium virtual host with the following requirements:

- 2 CPUs (x86_64)
- 4 GB RAM
- 40 GB Hard disk
- File system: ext4
- Ethernet network interface

It will act as an authorization server.

The client-server API follows the RESTful Web Services (JAX-RS 2.0), therefore a Java Application Server is needed in both sides (any should work, but only Apache HTTP has been tested):

- Client side: to configure the client-side and to grant access to the resources.
- Server side: to provide authorization access control.

Also, the satellite feature uses:

- A LDAP instance.

Remote service access:

- Host access to Ubuntu repositories (main, universe).
- Host access to NTP server of stratum ≤ 4.
- SSH access to host (public access or restricted to specific public source IP address).
- HTTPS (TCP/443) access to host (public access or restricted to specific public source IP address).

2.2 Enabler Installation

The installation package contains two archived web applications (client-side and server-side).

Download the enabler package release from the software repository of the 5G-ENSURE project (e.g. https://<5g-ensure.git>/enablers/FineGrainedAuthorization/archive/FineGrainedAuthorization.vXX_YY_ZZ.tar.gz) later extract the package on the temporal folder **/tmp** or clone the GitHub project (e.g. https://**<5g-ensure.git>**/enablers/FineGrainedAuthorization.git) on the temporal folder **/tmp**.

As a result, all the installation files of this enabler will be located in **/tmp/FineGrainedAuthorization.vXX_YY_ZZ**.

2.2.1 Server-side

2.2.1.1 Basic distributed authorization enforcement for distributed RCDs feature

Extract /tmp/FineGrainedAuthorization.vXX_YY_ZZ/FineGrainedAuthorizationEnabler-0.1.zip content in a random directory.

Execute the following commands starting from this directory.

2.2.1.1.1 XACML PDP Installation

Install AuthZForce CE Server 5.4.1 (see [1]) and its dependencies (JDK and Apache Tomcat).

Deploy authzscope-authzforce-extension:

```
$ sudo cp authzforce/authzscope-authzforce-extension-0.0.1-SNAPSHOT.jar /opt/authzforce-ce-server/webapp/WEB-INF/lib
```

Deploy configuration:

```
$ sudo cp authzforce/authzforce-ext.xsd /opt/authzforce-ce-server/conf

$ sudo cp authzforce/catalog.xml /opt/authzforce-ce-server/conf

$ sudo cp authzforce/pdp.xml /opt/authzforce-ce-server/data/domains/A0bdIbmGEeWhFwcKrC9gSQ
```

Restart tomcat:

```
$ sudo service tomcat7 restart
```

2.2.1.1.2 OAuth2 Server Installation

2.2.1.1.3 Apache2

Install apache2

```
$ sudo apt-get install apache2
```

Install openssl

```
$ sudo apt-get install openssl
```

Activate Apache2 SSL module

```
$ sudo a2enmod ssl
```

Generate a RSA 2048 key pair and a certificate

```
$ openssl genrsa -out privkey.pem 2048
$ openssl openssl rsa -in privkey.pem -pubout -out pubkey.pem
$ openssl req -key privkey.pem -new -x509 -days 2048 -out server.cr
```

Fill the question with answers relevant in the deployment context. Caution: the Common Name must exactly match the domain name of the server.

Configure Apache2 to enable SSL

- Follow step 3 and step 4 of the following guide for ssl configuration into Apache: [2]

2.2.1.1.4 PHP 7

Install php 7

```
$ sudo apt-get install php libapache2-mod-php php-mcrypt php-mysql php-curl php-xml
```

Create symbolic link from token.php to apache2 www directory

```
$ sudo ln -s `readlink -f oauth2-server/token.php` /var/www/html/token.php
```

Restart Apache2

```
$ sudo service apache2 restart
```

2.2.1.1.5 MySQL 5.7

Install mysql 5.7

```
$ sudo apt install mysql-server mysql-client
$ sudo mysql_secure_installation
```

During the installation, set a root password and answer yes to all questions of mysql_secure_installation process.

Copy key files (generated in 2.2.1.1.3) in a directory that mysql will be able to read

```
$ sudo cp pubkey.pem privkey.pem /var/lib/mysql-files
```

Create and define database

```
$ mysql -u root -p < oauth2-server/oauth_install.sql
```

Enter the root password defined during mysql 5.7 installation

2.2.1.2 Basic authorization in Satellite systems feature

First of all install the dependencies:

```
$ sudo apt install -y openjdk-8-jre tomcat8 slapd ldap-utils
```

LDAP administrator password: secret

Then, use the following command:

```
$ /tmp/FineGrainedAuthorization.vXX_YY_ZZ/FineGrainedAuthorization-Sat.vXX_YY_ZZ/server/install-
server.sh
```

Among other things, this script allows to:

- Set time zone to UTC.
- Configure the server IP address.
- Map hostnames to IP addresses.
- Configure JAVA_HOME and PATH environment variables.
- Create the 5G_FGA_SAT environment variable.

2.2.2 Client-side

2.2.2.1 Basic authorization in Satellite systems feature

First of all install the dependencies:

```
$ sudo apt install -y openjdk-8-jre tomcat8
```

Then, use the following command:

```
$ /tmp/FineGrainedAuthorization-Sat.vXX_YY_ZZ/client/install-client.sh
```

Among other things, this script allows to:

- Set time zone to UTC.
- Configure the client IP address.
- Map hostnames to IP addresses.
- Configure JAVA_HOME and PATH environment variables.
- Create the 5G_FGA_SAT environment variable.

2.3 Enabler Configuration

2.3.1 Server-side

2.3.1.1 Basic distributed authorization enforcement for distributed RCDs feature

2.3.1.1.1 User provisioning

Authenticable users must be provisioned in the local “oauth” database, created during the enabler installation.

User credentials and attributes must be provisioned like follows:

```
INSERT INTO oauth_clients (client_id, client_secret) VALUES ("<login>", "<password>");
INSERT INTO oauth_users (username, first_name, last_name, role) VALUES ("<login>", "<First Name>", "<Last Name>", "<Role>");
```

See oauth2-server/oauth_users.sql as an example.

2.3.1.1.2 User attributes definition

User attributes must be defined in the local “oauth” database schema. You can simply add new attributes with the following SQL command:

```
ALTER TABLE oauth_users ADD <column_name> <column_datatype>;
```

or completely recreate the table with:

```
DROP TABLE IF EXISTS oauth_users;
CREATE TABLE oauth_users (username VARCHAR(80) NOT NULL, <attribute1> <datatype>, <attribute2> <datatype>, <attribute3> <datatype>, PRIMARY KEY (username));
```

See oauth2-server/oauth_users.sql as an example.

2.3.1.2 Basic authorization in Satellite system feature

For security of communications it is highly recommended to enable SSL/TSL using the following command:

```
$ /tmp/FineGrainedAuthorization.vXX_YY_ZZ/FineGrainedAuthorization-Sat.vXX_YY_ZZ/server/security-server.sh
```

Among other things, this script allows to:

- Generate RSA keys and self-signed certificate

Finally, add entries to the LDAP dictionary information tree using the following command:

```
$ /tmp/FineGrainedAuthorization.vXX_YY_ZZ/FineGrainedAuthorization-Sat.vXX_YY_ZZ/server/loadDIT.sh
```

The LDAP dictionary tree represent a test organization for testing purposes:

- Domain: 5g-ensure.eu
- Root: admin
- Users:
  - Christopher Carroll
  - Mildred Dunn
  - admin5g
- Groups:
  - SNO (Satellite Network Operator). Formed by Christopher Carroll
  - SVNO (Satellite Virtual Network Operator). Formed by Mildred Dunn

The contents of the loadDIT.sh file is:

```
###Initialization
DOMAIN_ROOT=5g-ensure
DOMAIN_EXTENSION=eu
DOMAIN_DC="dc=$DOMAIN_ROOT,dc=$DOMAIN_EXTENSION"
ROOTDN="cn=admin,$DOMAIN_DC"
ROOTDN_PASSWORD=secret

###Add Organizational Units
ldapadd -w $ROOTDN_PASSWORD -D $ROOTDN <<EOF
dn: ou=groups,$DOMAIN_DC
objectClass: organizationalUnit
ou: groups


dn: ou=users,$DOMAIN_DC
ou: users
objectClass: organizationalUnit
EOF
```

```
###Add users
###http://uinames.com/
ldapadd -w $ROOTDN_PASSWORD -D $ROOTDN <<EOF
dn: cn=Christopher Carroll,ou=users,$DOMAIN_DC
objectClass: inetOrgPerson
cn: Christopher Carroll
givenName: Christopher
sn: Carroll
uid: christopher.carroll
userPassword: chCA_5g
mail: christopher.carroll@5g-ensure.eu

dn: cn=Mildred Dunn,ou=users,$DOMAIN_DC
objectClass: inetOrgPerson
cn: Mildred Dunn
givenName: Mildred
sn: Dunn
uid: mildred.dunn
userPassword: miDu_5g
mail: mildred.dunn@5g-ensure.eu

dn: cn=admin5g,ou=users,$DOMAIN_DC
objectClass: inetOrgPerson
objectClass: posixAccount
cn: 5G-Ensure Administrator
sn: Administrator
uid: admin5g
userPassword: 5g-ensure
mail: admin5g@bg-ensure.eu
uidNumber: 1000
gidNumber: 1000
homeDirectory: /home/admin5g
loginShell: /bin/bash
EOF
```

```
###Add groups
ldapadd -w $ROOTDN_PASSWORD -D $ROOTDN <<EOF
dn: ou=sno,ou=groups,$DOMAIN_DC
objectClass: groupOfNames
ou: sno
cn: sno
member: cn=Christopher Carroll,ou=users,$DOMAIN_DC

dn: ou=svno,ou=groups,$DOMAIN_DC
objectClass: groupOfNames
ou: svno
cn: svno
member: cn=Mildred Dunn,ou=users,$DOMAIN_DC
EOF
```

2.3.2 Client-side

2.3.2.1 Basic authorization in Satellite systems feature

For security of communications it is highly recommended to enable SSL/TSL using the following command:

```
$ /tmp/FineGrainedAuthorization.vXX_YY_ZZ/FineGrainedAuthorization-Sat.vXX_YY_ZZ/client/security-client.sh
```

This script configures SSL/TSL accordance with the server (see 2.3.1.2)

2.4 Troubleshooting

2.4.1 Basic distributed authorization enforcement for distributed RCDs feature

For troubles during XACML PDP installation or using Policy Management API, please refer to AuthZForce 5.4.1 Diagnosis Procedures (see [3]).

For troubles during OAuth2 server installation, please refer to Apache2, MySQL 5.7 or OpenSSL documentation, according to which step the error occurs.

For troubles using Authentication API, please ensure that the debug mode is set in oauth2-server/server.php by uncommenting the following line

```
ini_set('display_errors',1);error_reporting(E_ALL);
```

Then refer to the error message appearing with PHP7 documentation.

3 User and Programmer Guide

This section describes how to use the enabler and how to “program” the enabler.

3.1 User Guide

3.1.1 Basic distributed authorization enforcement for distributed RCDs feature

3.1.1.1 Policy Management API

This API concerns the RCD owner in order to manage its RCDs access control policy.

3.1.1.1.1 Set policy

Add a new XACML PolicySet:

- Method: POST
- Path: /authzforce-ce/domains/A0bdIbmGEeWhFwcKrC9gSQ/pap/policies
- Headers:
  - Content-Type: application/xml
  - Accept: application/xml
  - Content-Length: <content_length>
- Body: XACML PolicySet as defined in the XACML 3.0 schema

Caution: the top level PolicySet element must respect the following rules:

- The PolicySetId attribute must be “root”
- The PolicyCombiningAlgId must be “urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-permit”
- The Version attribute must be lexically greater than the one of the last uploaded PolicySet
- It must contain (at least) the following AdviceExpression

```
<AdviceExpression AdviceId="urn:thalesgroup:authzforce:scope:advice" AppliesTo="Deny">
      <AttributeAssignmentExpression
AttributeId="urn:thalesgroup:authzforce:scope:assignment">
            <AttributeDesignator AttributeId="oauth2-scope"
                  Category="urn:oasis:names:tc:xacml:3.0:attribute-
category:environment"
                  DataType="urn:thalesgroup:authzforce:scope:data-type:xacml-
policyset"
                  MustBePresent="true" />
      </AttributeAssignmentExpression>
</AdviceExpression>
```

See authzforce/policytest.xml as an example

3.1.1.1.2 Get policy

Returns the last deployed XACML PolicySet

- Method: GET
- Path: /authzforce-ce/domains/A0bdIbmGEeWhFwcKrC9gSQ/pap/policies/root/latest
- Headers:
  - Accept: application/xml

3.1.1.1.3 Delete policy

Deletes a deployed XACML PolicySet

- Method: DELETE
- Path: /authzforce-ce/domains/A0bdIbmGEeWhFwcKrC9gSQ/pap/policies/root
- Headers :
    - Accept: application/xml

3.1.1.2 Authentication API

This API concerns clients in order to request for an access token using their own credentials

3.1.1.3 Request token

Returns a JSON payload with the following fields:

- access_token: the effective JWT encoded and signed access token
- expires_in : the lifetime in seconds of the access token
- token_type: the access token type
- scope: a pre-resolved XACML PolicySet (according to user attributes), synthetizing the user access rights, that must be evaluated by the policy enforcement point without requiring external connection (resource, action and environment attributes to be evaluated). If set to '*', it means that the user has all rights.

- Method: POST
- Path: /token.php
- Headers:
    - Authorization: "<login>:<password>" encoded in base64
    - Content-Type: application/x-www-form-urlencoded
    - Content-Length : 29
- Body: "grant_type=client_credentials"

3.1.2 Basic authorization in Satellite systems feature

This section is not applicable as the enabler is a programmer tool; therefore there are no different user and programmer guides.

3.2 Programmer Guide

3.2.1 Basic distributed authorization enforcement for distributed RCDs feature

3.2.1.1 How to integrate your own user directory

This guide concerns those who want to integrate their own user directory with the OAuth 2.0 authentication server provided in the enabler. Please ensure that your user directory is able to provide user credentials as well as user attributes (for access control evaluation).

If starting from a fresh install, follow the 2.2.1.1.1, 2.2.1.1.3 and 2.2.1.1.4 sections of the installation guide only.

Edit oauth2-server/server.php and replace the line

```
$storage = new OAuth2\Storage\Pdo(array('dsn' => $dsn, 'username' => $username, 'password' => $password));
```

in order to instantiate your own storage object.

You can use one of the default provided connectors (PDO, Mongo, Redis, Cassandra, DynamoDB) in following the matching guide at [4] or implement your own custom storage in following the guide at [5].

If you don't want the OAuth2 server to use your storage to store its working data, you might want to take a look at the guide to use multiple storages at [6].

Then, you should edit the scope module of the OAuth 2.0 server in order to match your user attributes. Please follow the guide at [7] in order to create your own scope module, then integrate it to oauth2-sever/server.php in replaceing the line

```
$server->setScopeUtil(new XACMLScope($storage, $pdpURL));
```

with your own scope module instance.

See 3.2.1.2 section for more information on how your scope module should behave.

3.2.1.2 How to integrate your own OAuth2 authentication service

This guide concerns those who already have their own user directory as well as OAuth 2.0 authentication service and want to integrate it with the rest of the enabler. Please ensure that your OAuth 2.0 supports the Client Credentials authentication flow, is able to provide signed and timestamped access tokens (using JWT format for example), and allows to dynamically define the token scope.

If starting from a fresh install, follow the2.2.1.1.1 section of the installation guide only.

Then, you should customize your OAuth 2.0 service in order to make it perform requests toward the XACML PDP. You might take oauth2-server/XACMLScope.php as an example.

Your scope module should request the XACML PDP at URL http://localhost:8080/authzforce-ce/domains/A0bdIbmGEeWhFwcKrC9gSQ/pdp with an XACML Request containing all user attributes, as defined in the access control policy. The request should have the following form:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
CombinedDecision="false" ReturnPolicyIdList="false">
    <Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
subject">
        <Attribute AttributeId="attribute1" IncludeInResult="false">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Value1</AttributeValue>
        </Attribute>
        <Attribute AttributeId="attribute2" IncludeInResult="false">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Value2</AttributeValue>
        </Attribute>
    </Attributes>
</Request>
```

Please ensure that all the user attributes used in the access control policy are present in the request. If there is no value for an attribute for a given user, please put the matching <Attribute> element but without any value inside the <AttributeValue> child, in order to make them disappear from the future OAuth 2.0 token scope.

Then, the PDP should answer with a XACML Response, containing a Decision (which can be 'Permit' or 'Deny'), and in the case of a Deny decision, an advice with AdviceId "urn:thalesgroup:authzforce:scope:advice" containing an AttributeAssignment containing a XACML PolicySet that should be used as token scope. In the case of a Permit decision, set the scope to "*".

3.2.2 Basic authorization in Satellite systems feature

The enabler provides a REST interface, a resource-oriented API accessed via HTTP that uses JSON-based representations for information interchange. This Programmers Guide mainly consists of an overview of these RESTful API calls.

Description	Add a new XACML policy to the PAP policies repository
HTTP Method	POST
URI	https://5g-fga-sat-srv01.5g-ensure.eu:8080/fga-sat-srv/api/v01.00.00/pap/policy
URI Parameters	N/A
Request Header	N/A
Request Body Content-type	application/xml
Request Body Content	XACML policy
HTTP status code	201 when success 400 when error
Response Header	N/A
Accept Content-type	N/A
Response Body Content	N/A

Description	Get a XACML policy from the PAP policies repository
HTTP Method	GET
URI	https://5g-fga-sat-srv01.5g-ensure.eu:8080/fga-sat-srv/api/v01.00.00/pap/policies/{policyID}
URI Parameters	{policyID}: requested policy id
Request Header	N/A
Request Body Content-type	N/A
Request Body Content	N/A
HTTP status code	200 when success 404 when not found
Response Header	N/A
Accept Content-type	application/xml
Response Body Content	N/A

Description	Request RCD or satellite resource access
HTTP Method	POST
URI	https://5g-fga-sat-srv01.5g-ensure.eu:8080/fga-sat-srv/api/v01.00.00/pdp/authorize
URI Parameters	N/A
Request Header	Authorization: Basic <userName>:<userPassword> (encoded in base64)
Request Body Content-type	application/json
Request Body Content	{"userName":<username>,"action":<action>,"resource":<resource>}
HTTP status code	200 when allow access 401 when deny access
Response Header	N/A
Accept Content-type	application/json
Response Body Content	N/A

Description	Start transmit CW Carrier
HTTP Method	PUT
URI	https://5g-fga-sat-cli01.5g-ensure.eu:8080/fga-sat-cli/api/v01.00.00/satelliteModem/mgmt/startCWCarrier where CW means continuous wave
URI Parameters	N/A
Request Header	Authorization: 5GE-HMAC-SHA256 <userName>:<userPassword>
Request Body Content-type	application/json
Request Body Content	{"frequency":" integer_Hz","power":" real_dBm","cwMaximunDuration":" integer_sec"}
HTTP status code	201 when success 404 when not found
Response Header	N/A
Accept Content-type	application/json
Response Body Content	N/A

4 Unit Tests

This section describes the tests needed to cover all the features of the enabler. These tests verify that the enabler is ready to be evaluated in WP4.

Both features are based on RESTful HTTP API, therefore any HTTP client can be used to perform the test. For testing purposes curl command line tool has been used.

4.1 Basic distributed authorization enforcement for distributed RCDs feature

4.1.1 Information about Tests

The aim of the enabler is to provide an OAuth2 access token, signed, using JWT format. This token must contain a scope that is a particular XACML PolicySet corresponding to the current user rights. This token should be evaluated to permit actions the token owner has permission to do, or deny otherwise.

The following tests will ensure that:

- a global access control policy in XACML can be uploaded

and for a given global access control policy and given user credentials:

- the enabler returns an OAuth2 access token if and only if the user credentials are correct
- the token is correctly JWT-encoded, timestamped and signed

- the token scope contains a valid XACML PolicySet corresponding to the current user rights

Important: the following tests must be performed in order on the server host, and using installation directory as current directory.

4.1.2 Unit Test 1

This test aims to initialize the user database with test users.

Load data on the server host:

```
$ mysql -u root -p < oauth2-server/oauth_users.sql
```

Enter the root password defined during mysql 5.7 installation.

Expected result is:

```
client_id       client_secret   redirect_uri    grant_types     scope   user_id username
        first_name      last_name       role

admin   adminpwd        NULL    NULL    NULL    NULL    admin   NULL    NULL    Admin

guest   guestpwd        NULL    NULL    NULL    NULL    guest   NULL    NULL    NULL

john    johnpwd         NULL    NULL    NULL    NULL    john    John    Doe     Operator
```

4.1.3 Unit Test 2

This test aims to verify that the global XACML access control policy is correctly deployed. This XACML access control policy will be used in the following tests.

Deploy the test policy on the server host:

```
$ curl -H "Content-Type: application/xml" --data "@authzforce/policytest.xml"
http://localhost:8080/authzforce-ce/domains/A0bdIbmGEeWhFwcKrC9gSQ/pap/policies
```

Retrieve the last deployed policy from the server host:

```
$ curl -H “Accept: application/xml” http://localhost:8080/authzforce-
ce/domains/A0bdIbmGEeWhFwcKrC9gSQ/pap/policies/root/latest
```

Expected result is the PolicySet as defined in authzforce/policytest.xml

4.1.4 Unit Test 3

This test aims to verify that the authentication API is functional.

We assume the following terms:

- <domain_name>: the domain name of the service as defined during 2.2.1.1.3 installation phase.
- <ca_path>: the full path of the directory containing the certificate generated during 2.2.1.1.3 installation phase.

Request for a token as user admin from the server host:

```
$ curl -u admin:adminpwd https://<domain_name>/token.php -d ‘grant_type=client_credentials’ --
capath <ca_path>
```

Expected result should be of the following form:

```
{"access_token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6IjBhOWVkZTVlMzZjM2JjOTk4ZjZjMW
MzZDA2ZWUzNjFkMTQzMTRiYzgiLCJqdGkiOiIwYTllZGU1ZTM2YzNiYzk5OGY2YzFjM2QwNmVlMzYxZDE0
MzE0YmM4IiwiaXNzIjoiIiwiYXVkIjoiYWRtaW4iLCJzdWIiOm51bGwsImV4cCI6MTQ3MzE4MTk0NywiaWF0I
joxNDczMTc4MzQ3LCJ0b2tlbl90eXBlIjoiYmVhcmVyIiwic2NvcGUiOiIqIn0.TBPmNARErLi2CxRs5hscRBuz1D
79P72t3-
sKxKLUJORbfW2fh_3XtSC9T_9SZ9bjMUDQSLTyu2aqYik3VB53Pw2QEgUwGVe5PENwislO0BpYz1tk9WZVr
YFlGya1F-qlCBwXE8SLqsmHgE6heZCSuodb7cxZ_IymT3rCwR1pHE2UjZC1i87UlvavTnwHuGHPbI-
JO4ANW1CuN8yizfBHlYmqSXvDwjBF-pJjErN3UUgq45c-
bDvhr8ZHu_E5mIpsf2MojY53OG16Rfa72NVDTZxQqBRyIjR3udPjXesvfDmTB8ot5ikCSTB_TPhfoJ9D4_K7JoE
QW241ZLEt7nLMRA","expires_in":3600,"token_type":"bearer","scope":"*"}
```

access_token field is dependent of the current time so this should be slightly different but *expires_in*, *token_type* and *scope* should be identical.

4.2 Basic authorization in Satellite systems feature

4.2.1 Information about Tests

Important: the following tests must be performed in order on the server host, and using $5G_FGA_SAT directory as current directory.

4.2.2 Unit Test 1

This test aims to initialize the PAP policies repository with test policies.

Deploy the test policy on the server host:

```
$ curl -X GET -H "Content-Type: application/xml" -H "Accept: application/json" --data
"@test/UT01/input/UT01_TestPolicy.xml" https://5g-fga-sat-srv01.5g-ensure.eu:8080/fga-sat-
srv/api/v01.00.00/pap/policy
```

Expected result is HTTP status code 201.

4.2.3 Unit Test 2

This test aims to verify the XACML policies are correctly deployed in the PAP policies repository.

Request policy from the server host:

```
$ curl -X GET -H "Accept: application/xml" https://5g-fga-sat-srv01.5g-ensure.eu:8080/fga-sat-
srv/api/v01.00.00/pap/policies/UT01_TestPolicy
```

Expected result is HTTP status code 200 and the response body with the XACML policy previously uploaded.

4.2.4 Unit Test 3

This test aims to verify that the authentication/authorization API is functional.

Request for "PUT" action on resource https://5g-fga-sat-cli01.5g-ensure.eu:8080/fga-sat-cli/api/v01.00.00/satelliteModem/mgmt/startCWCarrier.

The contents of the test/UT03/input/UT03_RequestContent.json file is:

```
{"userName":<userName>,"action":"PUT","resource":"https://5g-fga-sat-rcd01.5g-ensure.eu:8080/fga-sat-rcd/api/v01.00.00/satelliteModem/mgmt/startCWCarrier"}
```

Request resource action from the RCD using the server host as an authorization proxy:

```
$ curl -H "Authorization: Basic <userName>:<userPassword>" -H "Content-Type: application/json" -H "Accept: application/json" --data "@UT03/input/UT03_RequestContent.json" https://5g-fga-sat-srv01.5g-ensure.eu:8080/fga-sat-srv/api/v01.00.00/pdp/authorize
```

Where <userName> is "Christopher Carroll" and <userPassword> is "chCA_5g" as defined in the LDAP.

Expected result is the Satellite Terminal SW version:

```
{ "header":{
        "responseCode":"200","bodyContentType":"application/json"
    },"body":{
        "status":"OK"
    }
}
```

5 Acknowledgements

The following partners contributed to this deliverable: Thales Alenia Space and Thales Services.

Thanks Martin Svensson (WPL), 5G-ENSURE peers and 5G-ENSURE Steering Committee for their collaboration and support.

Thanks to AuthZForce Authorization PDP (FIWARE Access Control GE) which provides a XACML 3.0 Authorization engine included in this enabler implementation (see [8]).

Thanks to OAuth 2.0 Server PHP which provides a simple OAuth 2.0 authentication server with support of Client Credentials autentication flow and signed and timestamped JWT access tokens included in this enabler implementation (see [9]). Thanks as well to oauth2-server-php-mysql which helped to write initialization scripts for MySQL PDO storage support (see [10]).

6 Abbreviations

This section comprises a summary of terms and definitions used during the later sections:

5G-PPP	**5G infrastructure Public Private Partnership**
ABAC	**Attribute-Based Access Control:** an access control paradigm whereby access rights are granted according to a combination of attributes of any type (user, resource, environment, etc.)
API	**Application Programming Interface**
CE	**Community Edition**
HTTPS	**HyperText Transfer Protocol Secure**
IP	**Internet Protocol**
JDK	**Java Development Kit**
JSON	**JavaScript Object Notification**
JWT	**Java Web toolkit**
LDAP	**Lightweight Directory Access Protocol**
LTS	**Long Term Support**
LVM	**Logical Volume Manager**
NTP	**Network Time Protocol**
PAP	**Policy Administration Point**
PDO	**PHP Data Objects**
PDP	**Policy Decision Point**
PHP	**PHP: Hypertext Preprocessor**
RBAC	**Role-Based Access Control:** an access control paradigm whereby access rights are granted according to roles and privileges
RCD	**Resource-Constrained Device**
REST	**REpresentational State Transfer**
RSA	**Rivest, Shamir Adleman public-key cryptosystem**
SSH	**Secure SHell**
SSL	**Secure Sockets Layer**
TCP	**Transmission Control Protocol**
TSL	**Transport Layer Security**
UE	**User Equipment**
URI	**Uniform Resource Identifier**
URL	**Uniform Resource Locator**
UT	**Unit Test**

UTC	**Coordinated Universal Time**
XACML	**eXtensible Access Control Markup Language**

7 References

[1] C. Dangerville, "Minimal setup | AuthZForce - Installation and Administration Guide," [Online]. Available: http://authzforce-ce-fiware.readthedocs.io/en/release-5.4.1/InstallationAndAdministrationGuide.html#minimal-setup.

[2] J. Ellingwood, "How To Create a SSL Certificate on Apache for Ubuntu 14.04," 23 April 2014. [Online]. Available: https://www.digitalocean.com/community/tutorials/how-to-create-a-ssl-certificate-on-apache-for-ubuntu-14-04.

[3] C. Dangerville, "Diagnosis Procedures | AuthZForce - Installation and Administration Guide," [Online]. Available: http://authzforce-ce-fiware.readthedocs.io/en/release-5.4.1/InstallationAndAdministrationGuide.html#diagnosis-procedures.

[4] B. Shaffer, "PDO Storage | OAuth2 Server PHP," [Online]. Available: https://bshaffer.github.io/oauth2-server-php-docs/storage/pdo/.

[5] B. Shaffer, "Custom Storage | OAuth2 Server PHP," [Online]. Available: https://bshaffer.github.io/oauth2-server-php-docs/storage/custom/.

[6] B. Shaffer, "Using Multiple Storages | OAuth2 Server PHP," [Online]. Available: https://bshaffer.github.io/oauth2-server-php-docs/storage/multiple/ .

[7] B. Shaffer, "Scope | OAuth2 Server PHP," [Online]. Available: https://bshaffer.github.io/oauth2-server-php-docs/overview/scope/.

[8] C. Dangerville, "Authorization PDP - AuthZForce," 31 05 2016. [Online]. Available: http://catalogue.fiware.org/enablers/authorization-pdp-authzforce.

[9] B. Shaffer, "An OAuth2 Server Library for PHP," 2016. [Online]. Available: https://bshaffer.github.io/oauth2-server-php-docs/.

[10] dsquier, "DDL to create MySQL tables for PDO storage support of oauth2-server-php library," [Online]. Available: https://github.com/dsquier/oauth2-server-php-mysql.

[11] Role-based access control, [Online]. Available: https://en.wikipedia.org/wiki/Role-based_access_control

[12] Attribute-based access control, [Online]. Available: https://en.wikipedia.org/wiki/Attribute-Based_Access_Control

[13] Ubuntu server guide, [Online]. Available: https://help.ubuntu.com/lts/serverguide/index.html

Deliverable D3.4
5G-PPP Security Enablers Documentation (v1.0)
Enabler Privacy Enhanced Identity Protection

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	30.09.2016	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	TIIT: Madalina Baltatu, Luciana Costa, Dario Lombardo	

European Commission

Contents

1 Introduction

The Privacy Enhanced Identity Protection enabler provides protection against subscriber's identity disclosure to unauthorized parties in 5G networks. The comprehensive goal of this enabler is to offer stronger user identity protection than in current 3G and 4G networks. The enabler can be summarized in several simple concepts: 5G user long term identity, also known as International Mobile Subscriber Identity (IMSI), shall not be transferred in clear text over the network in any situation. If a user long term identity (i.e., the IMSI) has to be sent from the UE to the network, it should be sent encrypted. For an encryption to be possible in situations where shared key material is not yet available, a public key-based mechanism is proposed (namely KPABE, [1], [2]).

The final version of the present enabler due in Release 2 will implement the following (full) set of features/components as enumerated in deliverable D3.1 [1]: Encryption of Long Term Identifiers and Pseudorandom dynamic pseudonyms. The present manual covers the version of the enabler provided in Release 1, as explained in D3.2 [2], and the specific feature planned for this version, namely: Encryption of long term or permanent identifiers.

2 Installation and Administration Guide

The present enabler consists in a cryptographic C library that implements the KP-ABE encryption scheme described in [3] allowing to encrypt and decrypt sensitive information like a given long term identifier (IMSI). For the rationales behind the use of KPABE the reader is asked to refer to previous study presented in D3.1 [1] and D3.2 [2]. The library provides the main encryption and decryption functions. It provides as well the setup and key generation functions, required in order to initialize the system and produce the key material.

In order to facilitate users' understanding of the functionalities and usage of the enabler, an example use case of the enabler is also described here and illustrated in Figure 1.

Figure 1 provides a simple proof of concept (PoC) deployment architecture, illustrating where the library should be installed. It also shows where each specific enabler's function is to be called in a 5G non 3GPP access scenario with EAP-AKA full authentication (based on IMSI).

This use case architecture is based on common hardware and readily available open source software, as indicated in the following sections.

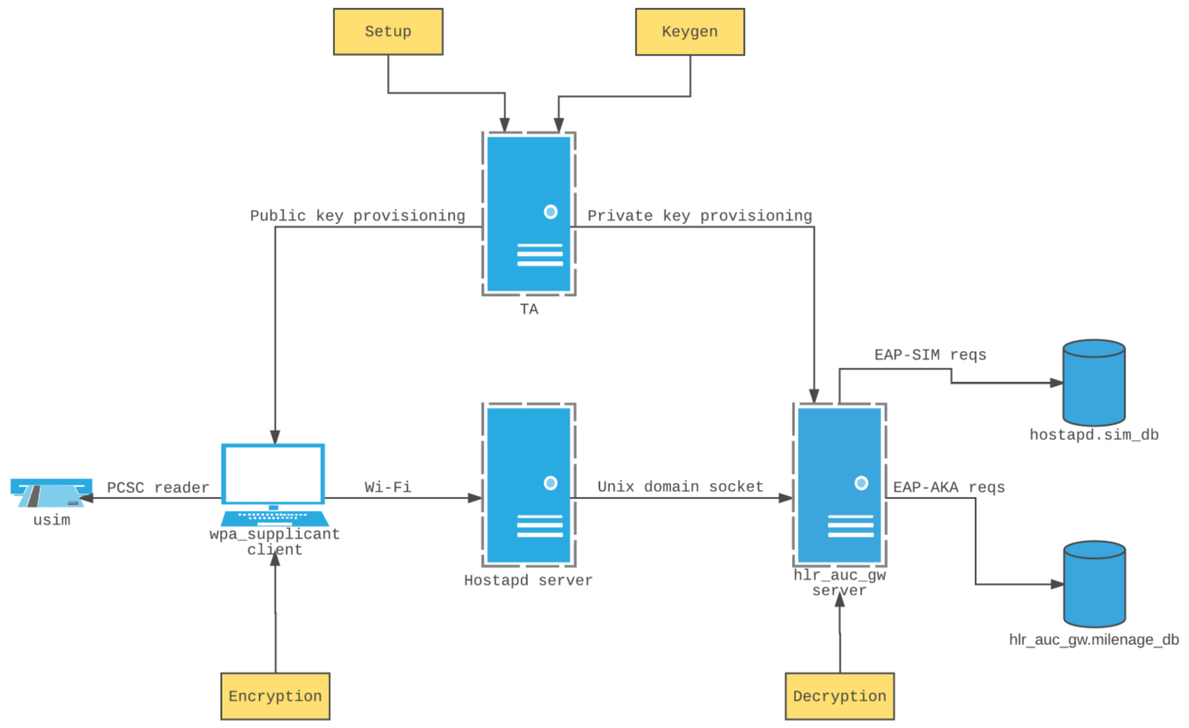


Figure 1 Example use case.

The algorithms implemented by the library are briefly described below with reference to the components illustrated in Figure 1.

- **Setup**: runs on a trusted authority server TA which is also responsible for the generation of private keys to trusted entities (e.g., network elements like authentication servers) based on the user access policy. At the cryptographic system initialization time, TA generates a public parameter (public key) and a master key. The latter is only known by the TA server.
- **Keygen:** runs on the TA server. An entity entitled to perform decryption (e.g., a network element like an authentication server) requests the provisioning of the decryption key (the private key) by presenting an access policy over an attribute or a set of attributes. The randomized key generation algorithm takes as input the public parameters (public key), the master key and the access policy. It outputs the authentication server's private key which is able to decrypt all IMSIs encrypted under the attributes that satisfy the access policy.
- **Encryption**: runs on the client components, e.g., on the UE devices (wpa_supplicant in Figure 1). The randomized encryption algorithm takes as input the user identity (i.e., the IMSI) to be encrypted, an attribute or set of attributes, and the public key. It outputs the ciphertext, i.e., the encrypted IMSI. In this way only the authentication servers which have the decryption key generated with the correct access policy (that matches the encryption attributes) will be able to decrypt the ciphertext.
- **Decryption:** the decryption algorithm runs on an authorized network element (i.e., the authentication server, i.e., the hlr_auc_gw in Figure 1). It takes as input the ciphertext, which was encrypted under the set of attributes, the public key parameter and the private key for access control. The output is the IMSI in clear text if the attributes included in the ciphertext satisfy the access policy.

The enabler delivers a C library (`libkpabe`) which implements these four functions required to perform all KPABE cryptographic operations. The library is developed by making use of some open source mathematics and cryptographic libraries that implement basic function blocks required by ABE and KPABE systems.

2.1 System Requirements

To work with the enabler, few requirements are necessary. They are listed below:

- A software for wireless communications, written in C language, that act as UE. An example of this software is wpa_supplicant.
- A software for wireless communications, that runs on the network side, written in C language. An example of this software is hlr_auc_gw (part of the hostapd project).
- 2 hardware equipment able to run the above softwares. An example of them are general purpose PCs with wifi cards.
- An operating system able to run on the above hardware and able to compile the enabler and the required libraries (see below). An example of it is Linux Ubuntu 14.04.

Additional software requirements on all systems are the prior installation of the following open source software libraries and packages:

- the PBC (Pairing Based Cryptography) C library [4],
- GMP, the GNU Multiple Precision Arithmetic C library [5],
- libcelia - a static linux C library implementing primitive kpabe operations [6],
- glib, the low-level core library used for GTK+ and GNOME and can be downloaded from [7].

2.2 Enabler Installation

Install GMP, PBC, glib and libcelia from the links provided in [5], [4], [7], and [7] by following the installation guides supplied with the corresponding packages. Note that glib and gmp can be installed on Ubuntu system from their packages by typing:

```
sudo apt-get install libglib2.0-dev
sudo apt-get install libgmp3-dev
```

Afterwards, libpkabe can be installed from its source archive. After tar gunzipping the archive, go to the libpkabe top directory and issue the following commands:

```
autoreconf -i
./configure
make
sudo make install
```

The working status of the local libkpabe installation can be verified by checking the output produced by the following command executed in the top directory of the library:

```
make check
```

The complete libkpabe documentation can be generated using:

```
make docs
```

from the top directory of the project. This command creates documentation subdirectories that contains the html and rtf versions of the manual.

The enabler is now installed on the host system. The software used to implement the cryptographic functions can now be linked against it and modified according to the desired workflow.

2.3 Enabler Configuration

The enabler is a dynamic linked library, hence it doesn't rely on configuration files, but, instead, allows dynamic configuration during the library calls. For better understanding on how to configure the options (like the attributes used for encryption or the location of the encryption keys), please see 3.1.

2.4 Troubleshooting

The library functions that compose the enabler have a very easy interface with few control on the error messages. Every function return a Boolean type that notify the caller whether the call was success or not.

Within the library the g_log() logging subsystem from glib has been used. According to glib's manual (see [10]), to raise or lower the logging some environment variables can be set. This will give the user the ability to better understand what's going wrong inside the library stack.

3 User and Programmer Guide

This enabler is a software library, hence the user guide actually can also be considered to some extent the programmer guide, since the end user of a C library is indeed a programmer. The user guide provides specific instructions for the user to deploy the example use case illustrated in Figure 1.

3.1 User Guide

There are four main steps to be undertaken in order to work with libkpabe. Herein each of them is described in details. Note that the regression test in tests/test.c shows a practical example of using the functions corresponding to each step. Wherever a variable is not listed, we consider its declaration/assignment as done in the previous steps.

3.1.1 Setup

This step is performed by a trusted party that is called Trust Authority (TA). During the setup the TA initializes the KP-ABE crypto system by exhaustively enumerating the attributes that are part of the authorized attributes universe and passing them to the setup function. Subsequently, by adding some randomness, this function creates a public key and a master key. The former is widely distributed to all the participants in the scheme, as in traditional public key encryption systems, while the latter is securely stored by the TA and it is never shared, not even with the authorized parties that are part of the ABE crypto system.

Example of use:

```
gchar universe[] = "t1 t2 t3";
GByteArray* pub;
GByteArray* msk;
if (kpabe_sys_setup(universe, &pub, &msk)) {
    print_some_error();
}
```

Attributes can be any string of letters, digits and underscores, and they must begin with a letter. The keywords “and”, “or” and “of” are reserved for the policy language and may not be used for an attribute.

An example of attributes are space separated strings, where each string identify the network authorized to request a private key from the TA. Such strings may be any identifiers such as SSID, PLMN-ID, authentication server identifier, etc.

The generated public and master keys can be accessed through pointers to their respective GByteArray variables. The GByteArray data type is provided by GLib.

The setup function returns FALSE if an uninitiated or an empty attributes string is passed as an argument.

This step is performed once when the system is booted up, and it’s made by a central trusted authority (called TA).

3.1.2 Key generation

Every authorized participant in the system asks the TA for the release of a private key. This key, paired with the common public key is used for decryption. The private key is therefore secret and must be securely stored by the authorized participant.

Example:

```
gchar policy[] = "t2";
GByteArray* prv;
if (kpabe_entity_keygen(policy, pub, msk, &prv)) {
    print_some_error();
}
```

The generation of a new private key requires the master key and the public key that were generated for the current cryptographic system. The policy parameter, which is a string passed to the keygen function, specifies what set of attributes the private key is able to decrypt from. While attributes can be any string of letters, digits and underscores, and they must begin with a letter, the keywords “and”, “or” and “of” are reserved for the policy language of the underlying library (libcelia) and may be used to logically combine authorized attributes in order to form a policy (hence they are forbidden as attributes). A policy is also known as an access structure which indicates the actual private keys that can decrypt the message encrypted under a set of authorized attributes.

This function provides the user with a pointer to the GByteArray variable that is used to store the generated private key.

The keygen function will return FALSE if the attribute(s) declared in the policy variable is/are not included in the set of the attributes universe specified in the setup function.

3.1.3 Encryption

Any UE that wants to send the network with information that need to be kept secret, performs the encryption of the secret by using the public key of the network and the related encryption attributes. Herein an example for the IMSI encryption.

Example of use:

```
char* imsi = "222011111111111";

guchar policy[] = "t2";

guchar* beimsi;


if (kpabe_imsi_encryption(pub, imsi, strlen(imsi), policy, &beimsi)) {

      print_some_error();

}
```

The input parameters of the encryption function are the public key data, a non-empty string containing the data to encrypt and the policy attributes. The list of attributes included in the encryption policy string directly determines which private keys are able to decrypt the message on the receiving system (network element).

The function stores the encrypted IMSI in a Base64-encoded string. It returns an FALSE if the policy attributes string has not been properly initiated, is empty or includes attributes that are not part of the attributes universe stored in the public key.

3.1.4 Decryption

The decryption is performed by the authorized participant that owns the right private key. Anyone participating in the KP-ABE scheme can encrypt a message with the attribute(s) bound to an authorized private key, but only that specific private key can decrypt it and obtain the original message in clear text.

Example of use:

```
guchar *decimsi;

if (kpabe_imsi_decryption(pub, prv, beimsi, &decimsi)) {

      print_some_error();

}
```

The decryption function requires the public key, a private key and the Base64-encoded encrypted data. The result of the decryption operation is saved in a string.

FALSE is returned if the selected private key is not authorized to decrypt the encrypted data because of the lack of attributes satisfying the encryption condition.

3.1.5 Deployment example

The enabler is not an end-user software, hence it has not a deployment procedure to follow. The extended software that links the enabler should have them. In this chapter we will provide an example of an extended software (hostapd) that make use of the enabler and its deployment. We don't provide the source code or patch for hostapd since it is not the enabler itself, but it's an example of the applied strategy.

All components are freely available on the Internet, while libkpabe is provided by the 5G ENSURE project under the GPL open source license.

The high level steps that a user/programmer has to perform in order to set up a functioning system for the use case illustrated in Figure 1, follow.

1. Get USIMs with known secrets (Ki). If they are not available, acquire a programmable USIM and a USB card reader. If the USIM does not come with preprogramed parameters or they have not been disclosed, it can be programmed with the desired authentication values using tools such as pySim [9]. At this step the user should know the IMSI, the Ki and the OPc (calculated from the Ki and the OP) values of his programmable USIM card. The latter approach is used for the proof of concept implemented to demonstrate the usage of the libkpabe enabler, since it's easier and more flexible.
2. Download the hostap project [8]. It ships wpa_supplicant (the component for the client system) and hostap+hlr_auc_gw (the authentication system component for the network side). Some changes are needed on those software in order to encrypt/decrypt the IMSI (just an example for any of the secrets we would like to protect).
3. Recompile the above components on 2 systems (client system and authentication system) with the libkpabe installed (see 2 for details).The aforementioned changes should allow the user to specify the details for kpabe (e.g. the attributes and the cryptographic keys paths).
4. Edit the hostapd runtime configuration file on the server. Configure the EAP-AKA milenage file that is used by the hlr_auc_gw component to authenticate the users. Using the parameters obtained in the first step, add a line for each allowed user matching the following format:

   ```
   <IMSI> <Ki> <OPc> <AMF> <SQN>
   ```

   The `AMF` and `SQN` parameters can be set respectively to `0000` and `000000000000.`

By following these instructions the user/programmer should be able to setup a running system as illustrated in Figure 1.

3.2 Programmer Guide

This section is not applicable as the enabler is not programmable.

4 Unit Tests

The libkpabe library is shipped with regression tests that run the aforementioned steps (i.e., setup, key generation, encryption, decryption). They are based on autotools test suite. To perform the tests just type:

```
make check
```

in the top directory of the libkpabe project. A short summary will illustrate the tests result. A report like the one illustrated below shows a healthy situation:

```
============================================================================
Testsuite summary for libkpabe 0.1
============================================================================
# TOTAL: 1
# PASS:  1
# SKIP:  0
# XFAIL: 0
# FAIL:  0
# XPASS: 0
# ERROR: 0
============================================================================
```

Exclusively for tests purposes, in order to obtain deterministic verifiable results, the libkpabe randomness has been disabled while running the regression tests.

The input of the regression test consists of known strings for the attributes universe and for the setup/encryption attributes policies. Every function of the library is invoked sequentially following the `setup→keygen→encryption→decryption` order. Since the test is running in a deterministic environment, the intermediate results of each function call are known a priori and can be compared to the expected data output. A regression test returns with a failure code if there is a mismatch between the produced data and the predefined one at any stage of the program.

4.1 Information about Tests

There are four unit tests available to be performed on libkpabe in order to ascertain its correct functioning, one unit for each of its main functions. The tests are run within the same executable (that's why the test report from autotools shows just 1 test).

4.1.1 Unit Test 1

This unit test checks the correct functionality of the setup function and is implemented as follows :

```
if (kpabe_sys_setup(universe, &pub, &msk)) {
        print_some_error();
}
if (compare_keys(PUB_KEY, pub->data, pub->len)) {
        print_some_error();
}
if (compare_keys(MASTER_KEY, msk->data, msk->len)) {
      print_some_error();
}
```

A known string containing the attributes universe is supplied to the setup function which produces a public key and a master key. The key generation process is deterministic since the PBC library randomness has been disabled in the regression test. This implies that the results can be matched against the expected output of the setup function, as shown by the `compare_key()` being called two times on each generated key. This unit test fails if any of the produced key is different than the expected

4.1.2 Unit Test 2

This unit test checks the correct functionality of the key generation function and is implemented as follows:

```
if (kpabe_entity_keygen(policy, pub, msk, &prv)) {
      print_some_error();
}
if (compare_keys(PRIVATE_KEY, prv->data, prv->len)) {
      print_some_error();
}
```

The `kpabe_entity_keygen()` function accepts a known string containing the chosen policy, the public key and the master key generated in Section 4.1.1. The output of the function is a private key with the input policy. Since the results are deterministic, the private key can be compared with its expected output using the `compare_key()` function. If the payloads differ, this unit test fails.

4.1.3 Unit Test 3

This unit test checks the correct functionality of the encryption function and is as follows:

```
if (kpabe_imsi_encryption(pub, message_clear, strlen((char*)message_clear),
            policy, &curr_message_enc)) {
      print_some_error();
}
if (compare_payloads(message_enc, strlen((char*)message_enc), curr_message_enc,
            strlen((char*)curr_message_enc))) {
      print_some_error();
}
```

The encryption function receives the public key generated in Section 4.1.2, a known string containing the IMSI to encrypt, the IMSI length and a known encryption policy string as input. It returns the Base64-encoded encrypted IMSI whose payload can be compared to the expected output since the encryption process is deterministic for the regression test. This unit test fails if there is a mismatch in the `compare_payloads()` function.

4.1.4 Unit Test 4

This unit test checks the correct functionality of the decryption function and is implemented as follows:

```
if (kpabe_imsi_decryption(pub, prv, curr_message_enc, &curr_message_dec)) {
      print_some_error();
}
if(compare_payloads(message_clear, strlen((char*)message_clear),
            curr_message_dec, strlen((char*)curr_message_dec))) {
      print_some_error();
}
```

The input parameters of the decryption function are the public key generated in Section 4.1.1, the private key generated in Section 4.1.2 and the Base64-encoded encrypted IMSI calculated at section 4.1.3. The decryption function returns the decrypted IMSI which is compared to the known string containing the plain IMSI passed to the encryption function as input parameter in Section 4.1.4. This unit test fails if the two strings do not match.

5 Acknowledgements

We want to acknowledge the developers and contributors of the open source projects used in the present work, especially libcelia, PBC, hostapd, and the Linux Ubuntu community.

6 Abbreviations

5G-PPP	5G Infrastructure Public Private Partnership
ABE	Attribute Base Encryption
KPABE	Key Policy Attribute Base Encryption
AKA	Authentication and Key-agreement
EAP	Extensible Authentication Protocol
EAP-AKA	Extensible Authentication Protocol Authentication Key Agreement (RFC 4187)
IMSI	International Mobile Subscriber Identity
PoC	Proof of Concept
GNOME	GNU Object Model Environment

7 References

[1] 5G ENSURE Deliverable D3.1, "5G-PPP security enablers technical roadmap (early vision)"

[2] 5G ENSURE Deliverable D3.2, "5G-PPP security enablers open specifications (v1.0))"

[3] Goyal, Pandey, Sahai, Waters, Attribute Based Encryption for Fine Grained Control of Encrypted Data, https://eprint.iacr.org/2006/309.pdf

[4] PBC, The Pairing-Based Cryptography Library, https://crypto.stanford.edu/pbc/

[5] The GMP library, the GNU Multiple Precision Arithmetic library, https://gmplib.org/

[6] Libcelia, a static Linux library implementing primitive kpabe operations,

https://github.com/gustybear/libcelia

[7] GLib, https://developer.gnome.org/glib/.

[8] hostapd, IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator,

https://w1.fi/hostapd/

[9] pySim, A python tool to program magic SIMs, http://cgit.osmocom.org/pysim/

[10] https://developer.gnome.org/glib/stable/glib-Message-Logging.html

Deliverable D3.4
5G-PPP Security Enablers Documentation (v1.0)
Enabler: Device Identifier Privacy

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	30.09.2016	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	UOXF: Piers O'Hanlon	

Contents

1 Introduction

The Device Identifier Privacy (DIP) enabler provides privacy enhanced network attachment, offering protection against unauthorized device tracking and device location disclosure. The main focus is to offer improved privacy protection of device identifiers, on IP-based networks, for access to 5G services, such as via Generic Access Network (GAN) [1], or to operator services. It also provides enhanced device identity privacy for access to third party Internet-based resources when utilising 5G-based authentication schemes (e.g. EAP-AKA [2] for WiFi). These privacy mechanisms are built upon the Detection of Network Attachment (DNA) [3] protocol which provides for reduced handover latency when moving between points of attachment.

2 Installation and Administration Guide

This section covers the system requirements, and describes how the enabler is installed, configured, and administered.

2.1 System Requirements

The system requirements for this enabler are that the target client device should be running Linux with a WiFi network interface. The client device runs the enabler which implements a privacy enhanced version of the DNA protocol in conjunction with a DHCP [4] client. The WiFi interface needs to connect via a direct, or simulated link, to a DHCP server (e.g. ISC-DHCP [5]) either running on a separate physical machine or, as outlined in the testing environment section 4.1, in a separate LXC [6] container or Virtual machine.

2.2 Enabler Installation

The enabler may be built and installed from the source code package. To install it from source, first obtain the compressed source tar file, and then follow the steps below:

```
$ tar xf dhcpcd-dip.X.XX.tgz
$ cd dhcpcd-dip.X.XX
$ ./configure
$ sudo make install
```

2.3 Enabler Configuration

This enabler provides for enhanced location privacy for a device that roams from one Internet-based network to the next, primarily in the context of non-3GPP 5G access. It provides for enhanced privacy with respect to the leakage of identifiers for previous points of attachment. Specifically, we introduce two new mechanisms to protect the privacy of the device:

- **Randomised ordering**: This mechanism will provide for randomised delivery of candidate link layer addresses in the DNA reachability tests, so as to enhance the location privacy with respect to the location and time-based analysis of the device's movement patterns.
- **Dummy addresses**: This mechanism will allow for the introduction of dummy addresses into the DNA reachability tests to enhance location privacy, with respect to location identification and time-based analysis of the device's movement patterns.

The enabler is based upon the widely used dhcpcd [7] DHCP client and utilises the same configuration file (/etc/dhcpcd.conf). For the purposes of this enabler three new configuration options are provided:

• **dna**	Enable Detection of Network Attachment (DNA) functionality
• **dna_random**	Enable randomised ordering of DNA requests
• **dna_dummy**	Enable use of dummy addresses in DNA requests

Once installed the client may be configured by editing the configuration file (/etc/dhcpcd.conf).

2.4 Troubleshooting

The enabler should normally run without user intervention as explained in the user guide section. However, if there are problems, then the enabler may be run in the foreground with console based debug logging enabled:

```
$ sudo /usr/sbin/dhcpcd -B –d [interface_name]
```

3 User and Programmer Guide

The enabler is not programmable.

3.1 User Guide

The detailed use of dhcpcd is well documented in its manual pages [7], but we will cover the aspects relevant to the operation of the enabler here. The service is configured as described in section 2 and typically it is invoked as a system service, though it also may be started directly on the command line, both procedures are outlined in the following sections.

3.1.1 Systemctl based management

With suitable system and service management tools such as `systemd`, or `system V` init scripts, the client may be launched via the system management controls.

To start the client, using `systemctl`, run the following command:

```
$ sudo systemctl start dhcpcd.service
```

To stop the client, using `systemctl`, run the following command:

```
$ sudo systemctl stop dhcpcd.service
```

3.1.2 Command line based management

The client may be launched directly via the command line, where upon it will start and attempt to acquire a DHCP lease, once acquired it will continue to run in the background.

To start the client, run the following command:

```
$ sudo /usr/sbin/dhcpcd [interface_name]
```

To stop the client, run the following command:

```
$ sudo /usr/sbin/dhcpcd -x [interface_name]
```

To stop the client, and release its DHCP leases, run the following command:

```
$ sudo /usr/sbin/dhcpcd -k [interface_name]
```

3.2 Programmer Guide

This section is not applicable as the enabler is not programmable.

4 Unit Testing

This section provides an overview of the virtualised testing environment, and number of tests that may be run to assess whether the tool is performing correctly.

4.1 Testing environment

For testing in a virtual environment, as shown in Figure 1, a number virtual WiFi devices may be instantiated using Linux's IEEE802.11 radio simulator (`mac80211_hwsim`) [8]. The client, with one simulated WiFi interface (i.e. wlan0), resides in one LXC container, whilst the server resides in a second LXC container. The server container includes multiple simulated WiFi interfaces (e.g. wlan1-4) which can be used to simulate movement by turning them on and off in a sequence using the `rfkill` [9] tool for enabling and disabling wireless devices. The mobility paths are shown to illustrate the different potential paths a client might take.

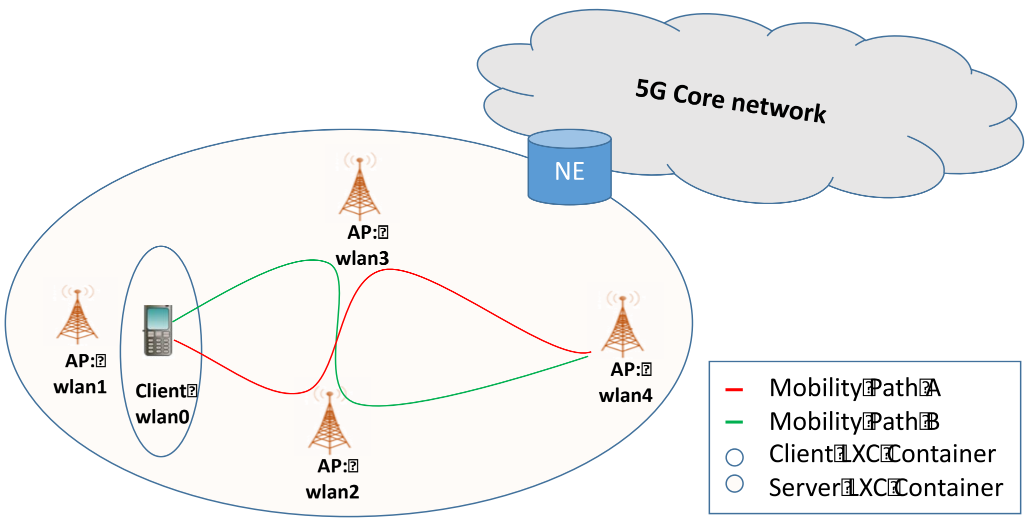


Figure 1 Test network configuration

4.2 Unit Tests

4.2.1 Unit Test 1

The dhcpcd source distribution includes a basic test suite. The test suite firstly builds the test executable, which allows one to verify that the test binary has been correctly built. Secondly the test binary allows for the confirmation of the correct operation of the security mechanisms:

```
$ tar xf dhcpcd-dip.X.XX.tgz
$ cd dhcpcd-dip.X.XX
$ ./configure
$ make test
```

The build and tests should complete with no errors reported.

4.2.2 Unit Test 2

The dhcpcd tool also has a run-time test mode which allows for testing of the tools functionality in obtaining a DHCP lease. The test mode will attempt to obtain an Internet address from a DHCP server, then perform Duplicate Address Detection (DAD), after which it exits and report the results of the interaction. The test mode is invoked by starting the tool using the –T option:

```
$ ./dhcpcd -T [interface_name]
```

Here is an example run with WiFi interface `wlan0` attached to WiFi network ‘`ssidtest1`’:

```
$ ./dhcpcd -T wlan0
wlan0: IPv6 kernel autoconf disabled
DUID 00:01:00:01:1f:07:b4:4e:08:00:27:75:f4:19
wlan0: IAID 00:00:00:00
wlan0: soliciting a DHCP lease
wlan0: soliciting an IPv6 router
wlan0: offered 10.10.1.17 from 10.10.1.1
interface='wlan0'
pid='10443'
reason='TEST'
ifcarrier='up'
ifflags='4163'
ifmtu='1500'
ifssid='ssidtest1'
ifwireless='1'
new_broadcast_address='10.10.1.255'
new_dhcp_lease_time='600'
new_dhcp_message_type='2'
new_dhcp_server_identifier='10.10.1.1'
new_ip_address='10.10.1.17'
new_network_number='10.10.1.0'
new_subnet_cidr='24'
```

```
new_subnet_mask='255.255.255.0'
dhcpcd exited
```

5 Acknowledgements

We would like to acknowledge Roy Marples, the main developer of the dhcpcd software.

6 Abbreviations

5G-PPP	5G Infrastructure Public Private Partnership
DNA	Detection of Network Attachment
DHCP	Dynamic Host Configuration Protocol
LXC	Linux Containers

7 Bibliography

[1] 3GPP, “TS44.318 Generic Access Network (GAN); Mobile GAN interface layer 3 specification (Release 13),” 3GPP, 2016.

[2] J. Arkko and H. Haverinen, “RFC4187 Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA),” IETF, 2006.

[3] B. Aboba, J. Carlson and S. Cheshire, “Detecting Network Attachment in IPv4 (DNAv4). RFC 4436.,” IETF, 2006.

[4] R. Droms, “Dynamic Host Configuration Protocol. RFC 2131 (Draft Standard),” IETF, 1997.

[5] I. S. C. (ISC), “ISC DHCP Server,” [Online]. Available: https://www.isc.org/downloads/dhcp/.

[6] “Linux LXC Containers,” [Online]. Available: https://linuxcontainers.org/.

[7] R. Marples and e. a. , “dhcpcd - DHCP client,” [Online]. Available: http://roy.marples.name/projects/dhcpcd/.

[8] J. Malinen, “Linux 802.11 radio software simulator (hwsim),” [Online]. Available: http://www.linuxwireless.org/en/users/Drivers/mac80211_hwsim/.

[9] J. Berg and M. Holtmann, “rfkill tool - to query the state of the rfkill switches,” [Online]. Available: http://www.linuxwireless.org/en/users/Documentation/rfkill/.

Deliverable D3.4
5G-PPP Security Enablers Documentation (v1.0)
Enabler VNF Certification

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	30.09.2016	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	TCS: Frederic Motte	

Contents

1 Introduction

The goal of this enabler is to provide, through a lightweight certification process, a Digital Trustworthiness Certificate (DTwC) containing trustworthiness information (evidence with metrics values) about the VNF. Using this Certificate, the infrastructure owner could decide whether to deploy this VNF into its infrastructure by comparing its trustworthy infrastructure requirements against the trustworthy information contained in the DTwC of the VNF. The certification process is intended to include testing phases, so the certification is not based solely on the claims made by the VNF owner.

The documentation here provided encompasses the installation and administration guide, the user and programmers guide and the unit testing of VNF certification enabler for Release 1 (aka V1.0) as described inD3.1 [1](Technical Roadmap) and specified in D3.2 [2](Open specifications). It is the documentation accompanying the software release of that enabler (see D3.3) for R1/v1.0 where features in scope are the following:

The first release will provide different elements coming from OPTET [1]/Fiware [2] projects with their adaptation for VNF and 5G environments:

- Format of the DTwC,
- Tools for certification workflow, (HMI and a preliminary version of the centralized certification repository)
- A certification process

The overall documentation is organized as follows. In Section 2, it is described how to install, configure and administrate the and in Section 3, we describe how to use the enabler. In Section 4, we provide the unit tests as needed to cover all the features of the enabler. Section 5contains people's acknowledgments for their contributions. Finally, abbreviations and references are listed in Section 6 and Section 7.

2 Installation and Administration Guide

The VNF certification Enabler offers a certification workflow for VNF elements. This section covers the system requirements, and describes how the enabler is installed and configured.

2.1 System Requirements

The VNF Certification enabler consists of two parts:

- A client element used by the certifier to generate DTwC. This feature is an eclipse plug-in used by each certification body to help in the VNF certification process. The feature could be deployed for each certification body.
- A server element used to collect the different certificates produced by the different certification bodies. This service offers also an API to retrieve a DTwC for a dedicated VNF

The basic/minimal system requirements for this enabler are that the server and the network components should be running Linux, preferably Ubuntu 14 LTS or later (help available at [ref [5]], with a network interface.

For the client installation requirements are

- 1 CPU (i686/x86_64)

- 4 GB RAM
- 20 GB Hard disk
- Ethernet network interface

For the server installation requirements are

- 1 CPU (i686/x86_64)
- 2 GB RAM
- 20 GB Hard disk
- Ethernet network interface

2.2 Enabler Installation

The installation package contains two archived applications (one for the client HMI and one for the repository server).

2.2.1 Client element

Two possible installation procedures could be used to get the VNF certification enabler ready to use

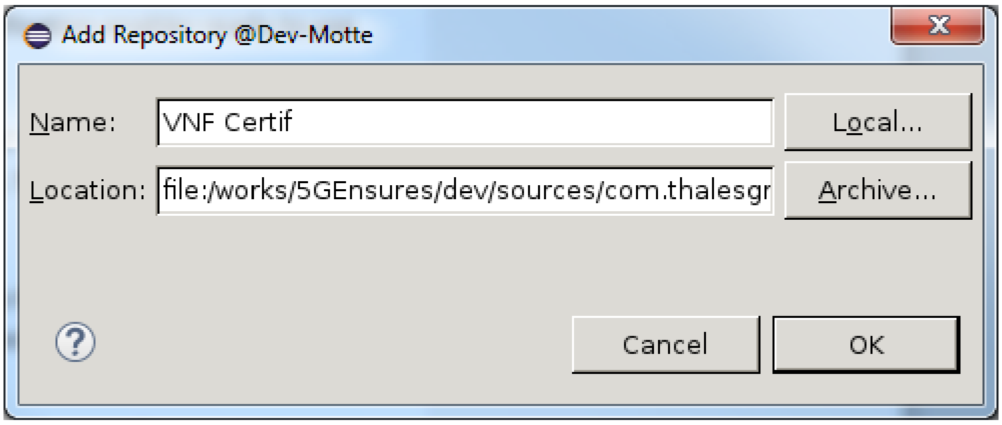
2.2.1.1 The Zip installation

The simplest solution is to take the Zip installation provided and to extract the complete VNF certification enabler already configured. For that, download the Zip archive and unzip it (The Zip could be found at this URL : https://workspace.vtt.fi/sites/5g-ensure/Site%20Assets/Forms/AllItems.aspx?RootFolder=/sites/5g-ensure/Site%20Assets/SitePages/5G%20ENSURE%20-%20Software%20Delivery%20information%20for%20R1/VNF_Certification/Release%201&). The VNF Certification tool (in the rest of the document, the name "factory" will be used) is ready to use in the eclipse directory

2.2.1.2 The eclipse update

In case of the certification body has already an eclipse environment installed, he/she could just install the VNF certification enabler by using the eclipse update facility provided by eclipse and using the VNF Certification update site.

In eclipse HMI, the software installation view must be opened: Help>Install new software. In this view, add a new local repository and point to the VNF certification update site (updatesite directory in the enabler delivery).


Add Repository @Dev-Motte

Name: VNF Certif

Location: file:/works/5GEnsures/dev/sources/com.thalesgr

Then select the feature required and complete the installation workflow by accepting the license and by restarting the eclipse installation.

At the end of the installation procedure, the eclipse.ini file must be modified. The line -product org.eclipse.platform.ide must be added in this file like this.

```
-startup
plugins/org.eclipse.equinox.launcher_1.3.0.v20140415-2008.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.gtk.linux.x86_64_1.1.200.v20150204-1316
-product
org.eclipse.epp.package.standard.product
--launcher.defaultAction
openFile
-showsplash
org.eclipse.platform
--launcher.XXMaxPermSize
256m
-product
org.eclipse.platform.ide
--launcher.defaultAction
openFile
--launcher.appendVmargs
-vmargs
-Dosgi.requiredJavaVersion=1.7
-XX:MaxPermSize=256m
-Xms40m
-Xmx512m
```

The eclipse platform must be restarted at the end of the installation in order to activate all the options.

2.2.2 Server element

The server is a servlet which collects DTwCs. (This element was just tested with tomcat 7 and earlier. Other application servers supporting servlet and JDK8 could be used)

First of all, install the dependencies:

```
$ sudo apt-get install tomcat7
```

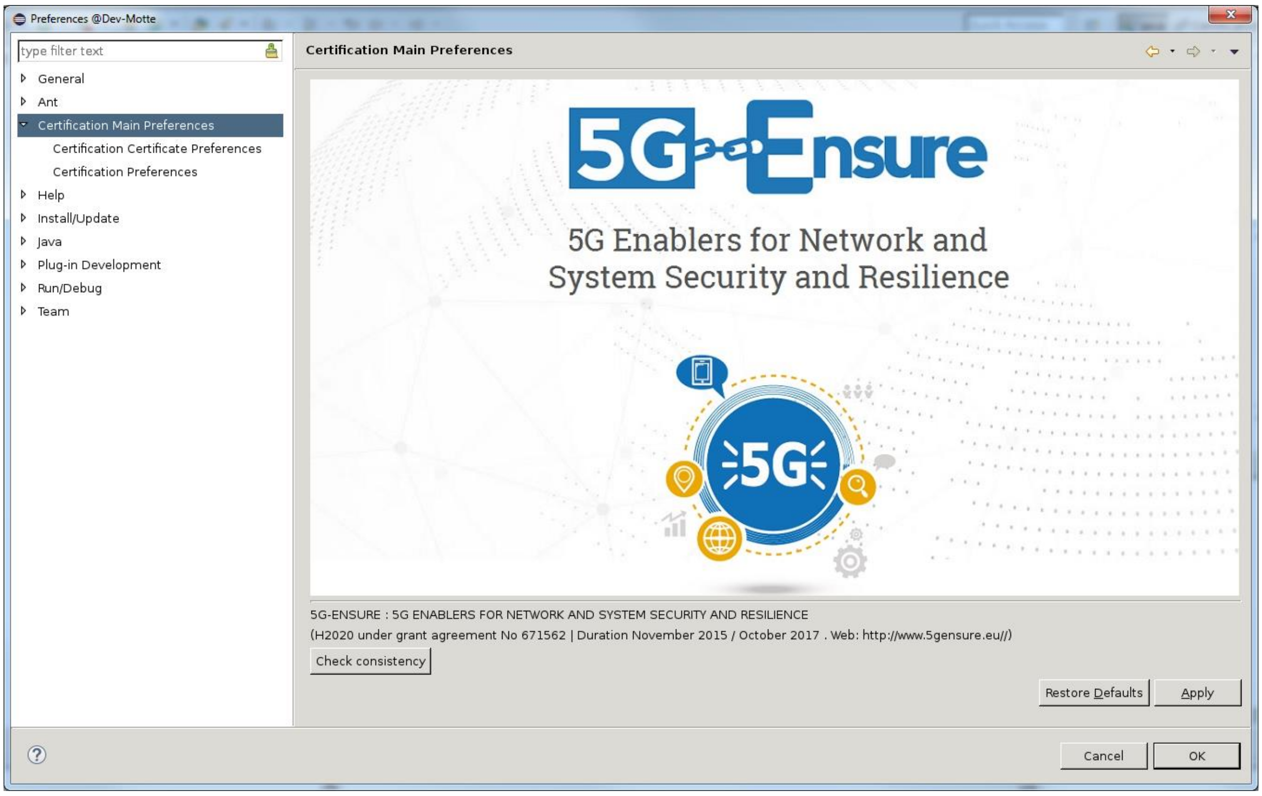
Then copy the war archive of the certification repository into the webapps directory of the tomcat installation (be careful about permissions, the war must be has the tomcat owner and group). Then, configure the tomcat server to just provide secured communication (remove the http port and just allow the https).

2.3 Enabler Configuration

Some configurations are required for the different elements of the enabler.

2.3.1 Client configuration

The configuration of the client part of the VNF certification enabler is performed through the preference pages of eclipse. To access these preference pages, click on Windows>Preferences>Optet Main Preferences The following page appears:


Preferences @Dev-Motte
type filter text
General
Ant
Certification Main Preferences
Certification Certificate Preferences
Certification Preferences
Help
Install/Update
Java
Plug-in Development
Run/Debug
Team
Certification Main Preferences
5G-Ensure
5G Enablers for Network and System Security and Resilience
5G
Check consistency
Restore Defaults
Apply
Cancel
OK

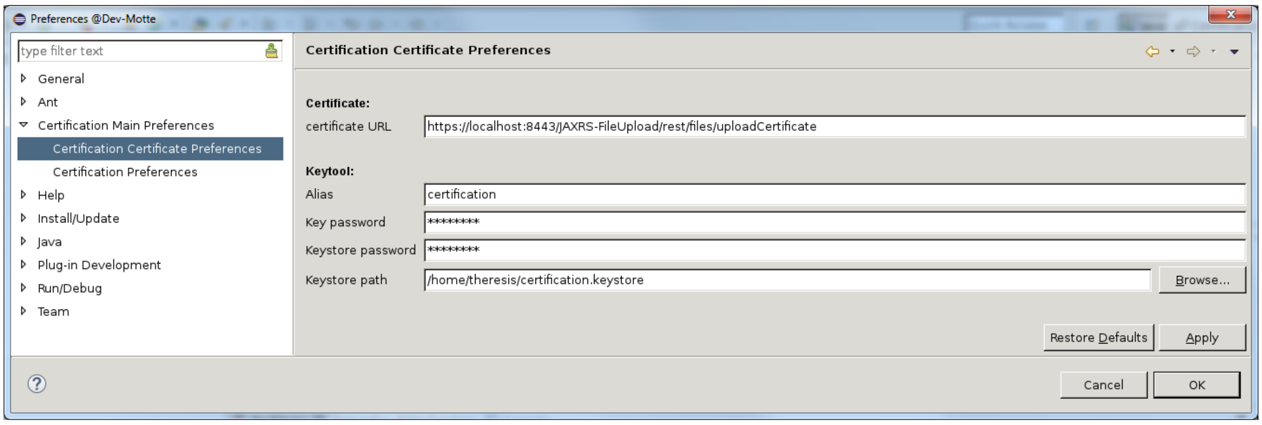
5G-ENSURE : 5G ENABLERS FOR NETWORK AND SYSTEM SECURITY AND RESILIENCE
(H2020 under grant agreement No 671562 | Duration November 2015 / October 2017 . Web: http://www.5gensure.eu//)

This main page presents some information regarding the 5G-Ensure project. Under this page, two pages are present:

- The certificate preference page
- The global preferences page

2.3.1.1 The Certificate preference page

The Certificate Preferences allows the certifier to specify some required data about the generation of the DTwC.



Certificate:

certificate URL	https://localhost:8443/JAXRS-FileUpload/rest/files/uploadCertificate

Keytool:

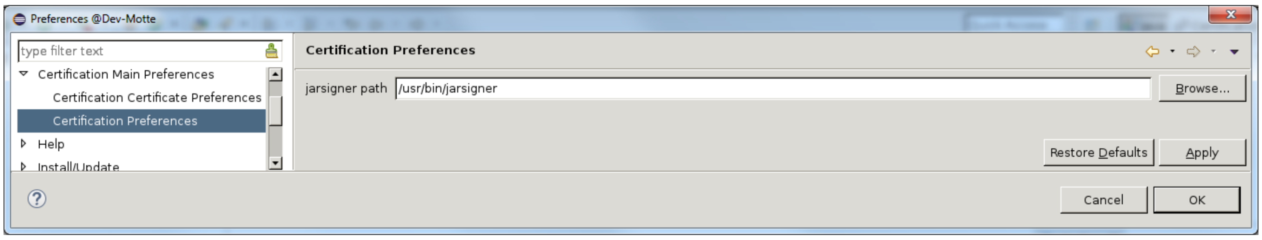
Alias	certification
Key password	********
Keystore password	********
Keystore path	/home/theresis/certification.keystore

At the end of the certification process, the certificate generated by the automatic process is signed using the certificate of the certifier. For that, the certifier must enter the element required in order to use his certificate (see the keytool part). This certificate is stored in a key store and the access to this key store is specified (the path, the alias of the certificate, the password of the certificate, the password of the key store).

When the DTwC is generated, it’s stored locally and also pushed to certificate repository in charge of storing all the certificates generated by the different certification tool.

2.3.1.2 The global preferences page

The different platforms and tools are based on different plugins combined together. Each plugin must have a signature to check the validity and consistency.
To check this, the path of the jarsigner process (provided by the JDK) must be specified into the preferences pages



Certification Preferences

jarsigner path /usr/bin/jarsigner

2.3.2 Repository configuration

The certification repository could be configured by specifying the path of the system repository into the web.inf configuration file of the servlet (This file is present into the working directory of the webapp into the tomcat directory).

2.4 Troubleshooting

The VNF certification enabler is based on Eclipse which provides error managements and diagnostic element. The factory follows this philosophy providing:

- Error Dialogs: The factory displays messages using message dialogues to display information to the user (errors or information)
- Log information: the factory has a log file where problems are recorded. The log file can be found in a couple of places:
  - Workspace log - This is the most common location for the log file. It is stored in your workspace in the meta-data directory. Check the workspace/.metadata/.log file.
  - Configuration log - Sometimes information is recorded in the configuration log file instead of the workspace log (especially if the workspace hasn’t been created yet, there isn’t one, or it cannot be created). Check your configuration area for a configuration log file (configuration/<timestamp>.log).

Regarding the certification repository, the log file is the log file of Tomcat (for example, the catalina.out file in the /var/log/tomcat7 directory).

3 User and Programmer Guide

This section describes how to use the enabler and how to program the enabler.

3.1 User Guide

3.1.1 Overview

The enabler can be used in a variety of use cases and by a variety of users, both when designing new network configurations enabled by 5G technologies and when making trust decisions during the use and operation of networks.

This certification enabler has two main objectives:

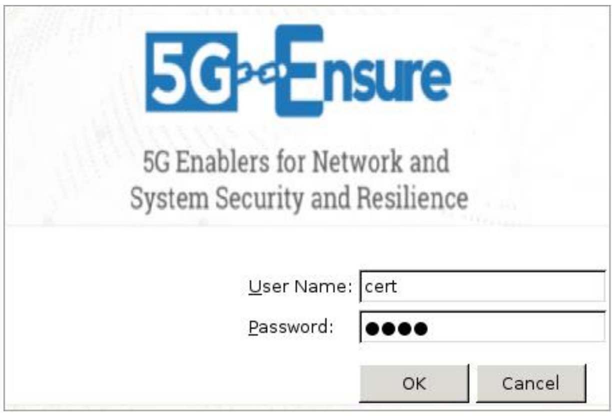
1. Help the certifier to collect evidence about the VNF and to subsequently produce a DTwC.
2. Provide a centralized service where all the DTwCs produced will be stored.

For more information, please read the Open Specification [3]

3.1.2 The VNF Certification client

3.1.2.1 Authentication into the application

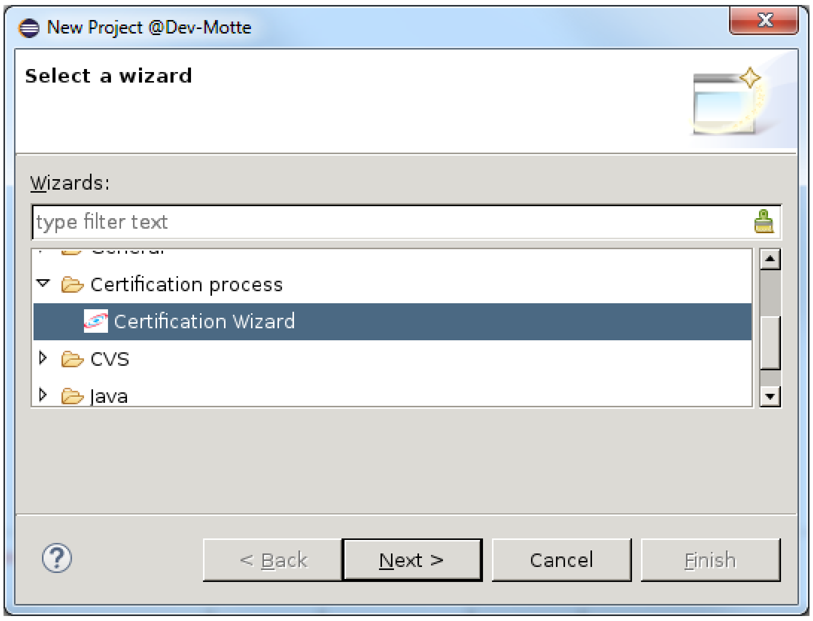
To access to the Certification tool, an authentication is required in order to know who will realize the certification. For the moment, the mechanism is present but not linked with a repository like LDAP, so, just click on OK with the default values. (This evolution will be provided for the second version of this enabler)



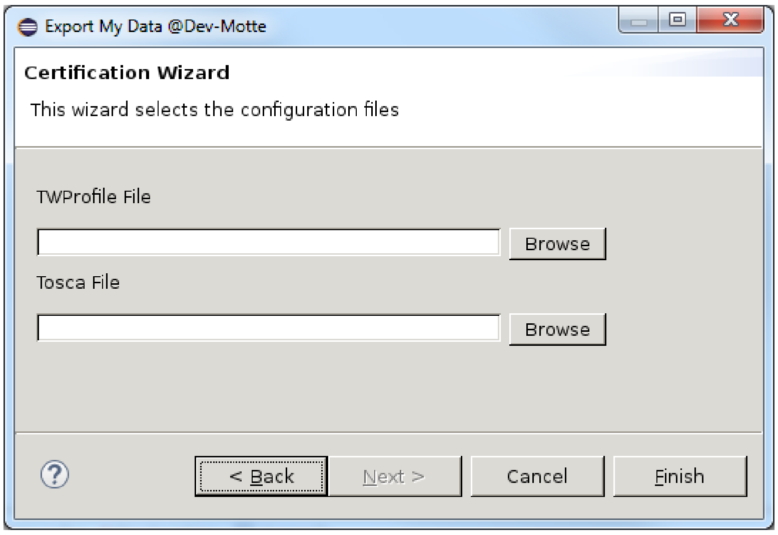
User Name: cert
Password: ●●●●

3.1.2.2 Creation of the certification project

To start a certification, the certifier must create a certification project using the Certification wizard provided by the environment. For that, File->New->Project



At this stage, the certifier must specify a project name for its certification project and click on next. Then a new page appears asking some inputs for the certification:



The "TWProfile file" element is the configuration file provided by the VNF owner. This file contains the TWAttributes expected by the VNF owner associated with metrics and expectedValue. A full version of this file is the following (This file is an example. The different values are arbitrary):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TWProfile>
        <TWAttribute name="FunctionalSuitability" attributeID="TWA001">
                <Metric name="SupportedStandard" metricID="EV0001" expectedValue="80"></Metric>
                <Metric name="Monitorable" metricID="EV0002" expectedValue="80"></Metric>
        </TWAttribute>
        <TWAttribute  name="PerformanceEfficiency" attributeID="TWA002">
                <Metric  name="ProvisionningLatency" metricID="EV0003" expectedValue="80"></Metric>
                <Metric  name="Speed" metricID="EV0004" expectedValue="80"></Metric>
                <Metric  name="Elasticity" metricID="EV0005" expectedValue="80"></Metric>
                <Metric  name="HighAvailability" metricID="EV0006" expectedValue="80"></Metric>
        </TWAttribute>
        <TWAttribute  name="Compatibility" attributeID="TWA003">
```

```
                <Metric  name="SupportedStandard" metricID="EV0007" expectedValue="80"></Metric>
                <Metric  name="HardwareIndependance" metricID="EV0008" expectedValue="80"></Metric>
                <Metric  name="VirtualisationAndContainerAwareness" metricID="EV0009" expectedValue="80"></Metric>
        </TWAttribute>
        <TWAttribute  name="Usability" attributeID="TWA004">
                <Metric  name="PolicyManagment" metricID="EV0010" expectedValue="80"></Metric>
                <Metric  name="ApplicationManagement" metricID="EV0011" expectedValue="80"></Metric>
        </TWAttribute>
        <TWAttribute  name="Reliability" attributeID="TWA005">
                <Metric  name="Reliability" metricID="EV0012" expectedValue="80"></Metric>
        </TWAttribute>
        <TWAttribute  name="Security" attributeID="TWA006">
                <Metric  name="AppropriateUseOfEncryptionTechniques" metricID="EV0013" expectedValue="80"></Metric>
                <Metric  name="Integrity" metricID="EV0014" expectedValue="80"></Metric>
                <Metric  name="SecureBootAndSecureCrash" metricID="EV0015" expectedValue="80"></Metric>
                <Metric  name="SoftwareHardened" metricID="EV0016" expectedValue="80"></Metric>
                <Metric  name="SecuredManagment" metricID="EV0017" expectedValue="80"></Metric>
                <Metric  name="SecureCommunication" metricID="EV0018" expectedValue="80"></Metric>
                <Metric  name="DataProtectionAndPrivacy" metricID="EV0019" expectedValue="80"></Metric>
        </TWAttribute>
        <TWAttribute  name="Maintainability" attributeID="TWA007">
                <Metric  name="InternalStructure" metricID="EV0020" expectedValue="80"></Metric>
                <Metric  name="PatchManagment" metricID="EV0021" expectedValue="80"></Metric>
                <Metric  name="IntegrityManagment" metricID="EV0022" expectedValue="80"></Metric>
                <Metric  name="Monitorable" metricID="EV0023" expectedValue="80"></Metric>
        </TWAttribute>
        <TWAttribute  name="Portability" attributeID="TWA008">
                <Metric  name="MigtrationOperations" metricID="EV0024" expectedValue="80"></Metric>
                <Metric  name="HardwareIndependence" metricID="EV0025" expectedValue="80"></Metric>
                <Metric  name="LocationAwareness" metricID="EV0026" expectedValue="80"></Metric>
                <Metric  name="VirtualisationAndContainerAwareness" metricID="EV0027" expectedValue="80"></Metric>
                <Metric  name="VNFState" metricID="EV0028" expectedValue="80"></Metric>
        </TWAttribute>
</TWProfile>
```

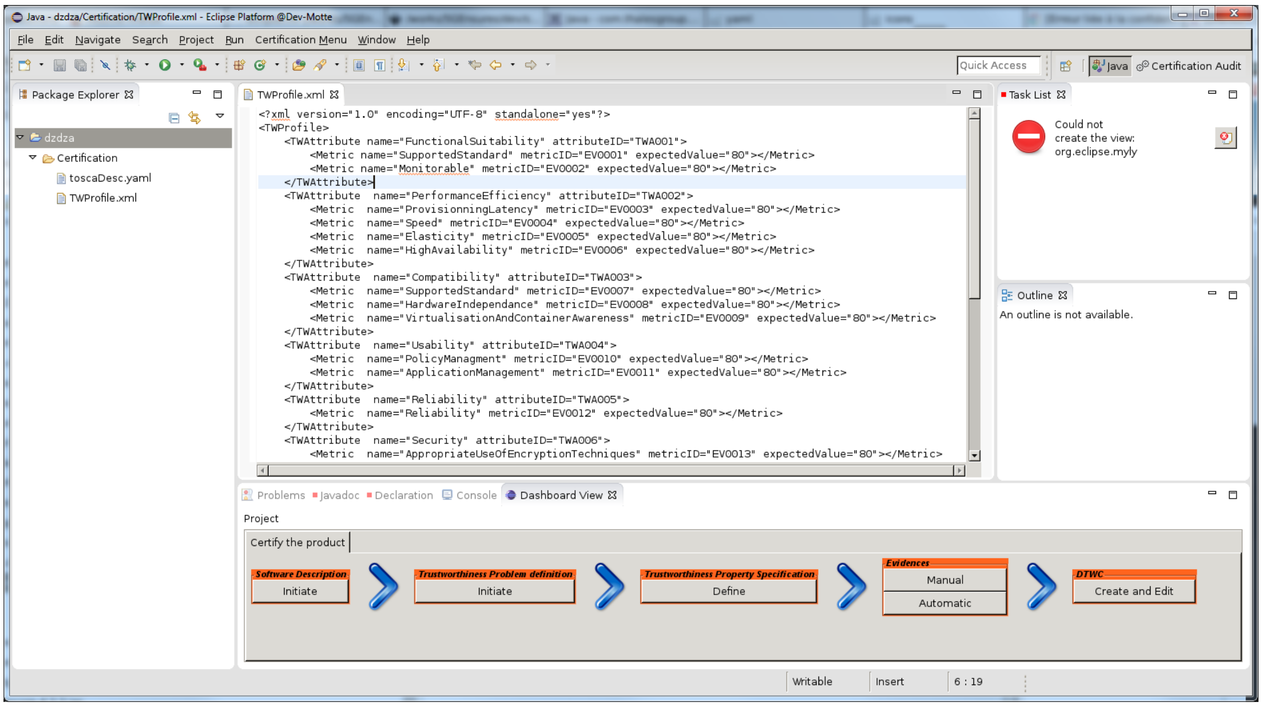
The second file is the TOSCA [4] file describing the VNF and based on the TOSCA Simple Profile in YAML [8]normalized by OASIS. This file will be used to extract evidences during the evaluation of the VNF an example of a Tosca file is present in annex Tosca file example. It’s possible to provide a CSAR file containing a TOSCA file based on the Tosca standard.

When the two files are selected, the “finish” button cloud be clicked to finish the project creation.

3.1.2.3 Start the certification process

When the certifier is ready to certify the VNF, he/she must select the project required and, using the contextual menu, click on CertificationProcess->Certification

At this stage, a specific view appears



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TWProfile>
    <TWAttribute name="FunctionalSuitability" attributeID="TWA001">
        <Metric name="SupportedStandard" metricID="EV0001" expectedValue="80"></Metric>
        <Metric name="Monitorable" metricID="EV0002" expectedValue="80"></Metric>
    </TWAttribute>
    <TWAttribute  name="PerformanceEfficiency" attributeID="TWA002">
        <Metric  name="ProvisionningLatency" metricID="EV0003" expectedValue="80"></Metric>
        <Metric  name="Speed" metricID="EV0004" expectedValue="80"></Metric>
        <Metric  name="Elasticity" metricID="EV0005" expectedValue="80"></Metric>
        <Metric  name="HighAvailability" metricID="EV0006" expectedValue="80"></Metric>
    </TWAttribute>
    <TWAttribute  name="Compatibility" attributeID="TWA003">
        <Metric  name="SupportedStandard" metricID="EV0007" expectedValue="80"></Metric>
        <Metric  name="HardwareIndependance" metricID="EV0008" expectedValue="80"></Metric>
        <Metric  name="VirtualisationAndContainerAwareness" metricID="EV0009" expectedValue="80"></Metric>
    </TWAttribute>
    <TWAttribute  name="Usability" attributeID="TWA004">
        <Metric  name="PolicyManagment" metricID="EV0010" expectedValue="80"></Metric>
        <Metric  name="ApplicationManagement" metricID="EV0011" expectedValue="80"></Metric>
    </TWAttribute>
    <TWAttribute  name="Reliability" attributeID="TWA005">
        <Metric  name="Reliability" metricID="EV0012" expectedValue="80"></Metric>
    </TWAttribute>
    <TWAttribute  name="Security" attributeID="TWA006">
        <Metric  name="AppropriateUseOfEncryptionTechniques" metricID="EV0013" expectedValue="80"></Metric>
```

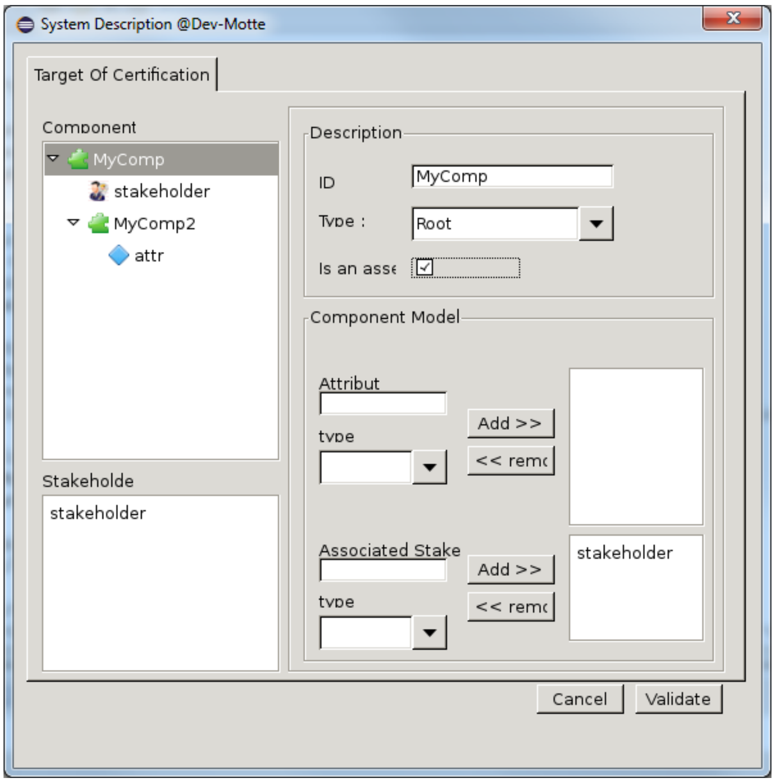
Could not create the view: org.eclipse.myly

An outline is not available.

For each step, a dedicated view will help the certifier to enter the required data. All the data is required for the successful completion of the DTwC generation.

The first step is the software description

This view helps the certifier to describe the certification environment applied to perform the certification. The description takes into account the different component linked together and the different interactions with the specified stakeholders. In the next release, this part could be defined automatically using the output of the Trust Metric Enabler:


System Description @Dev-Motte

Target Of Certification

Component
- MyComp
  - stakeholder
  - MyComp2
    - attr

Stakeholde
stakeholder

Description
ID: MyComp
Type : Root
Is an asse ☑

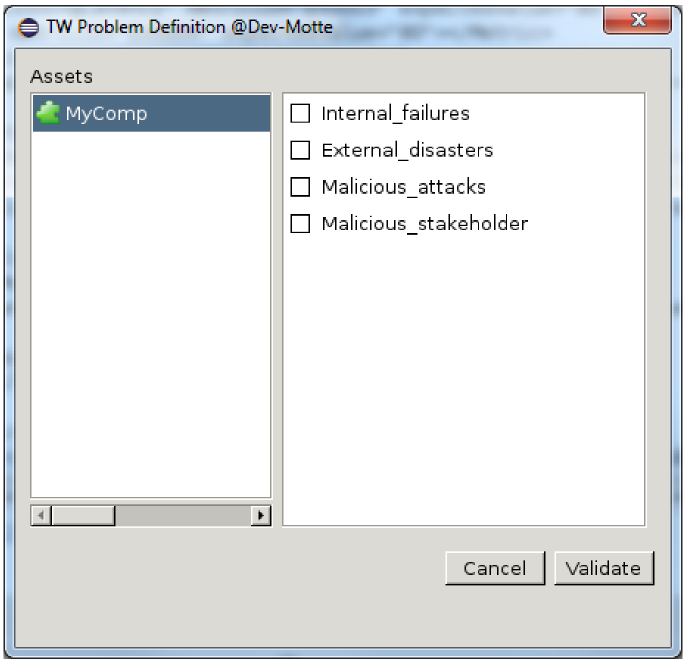
Component Model
Attribut
type
Add >>
<< remc
Associated Stake
type
Add >>
<< remc
stakeholder

Cancel
Validate

In this view, from the root element, we can create a subcomponent and associate either other components, an attribute or a stakeholder.

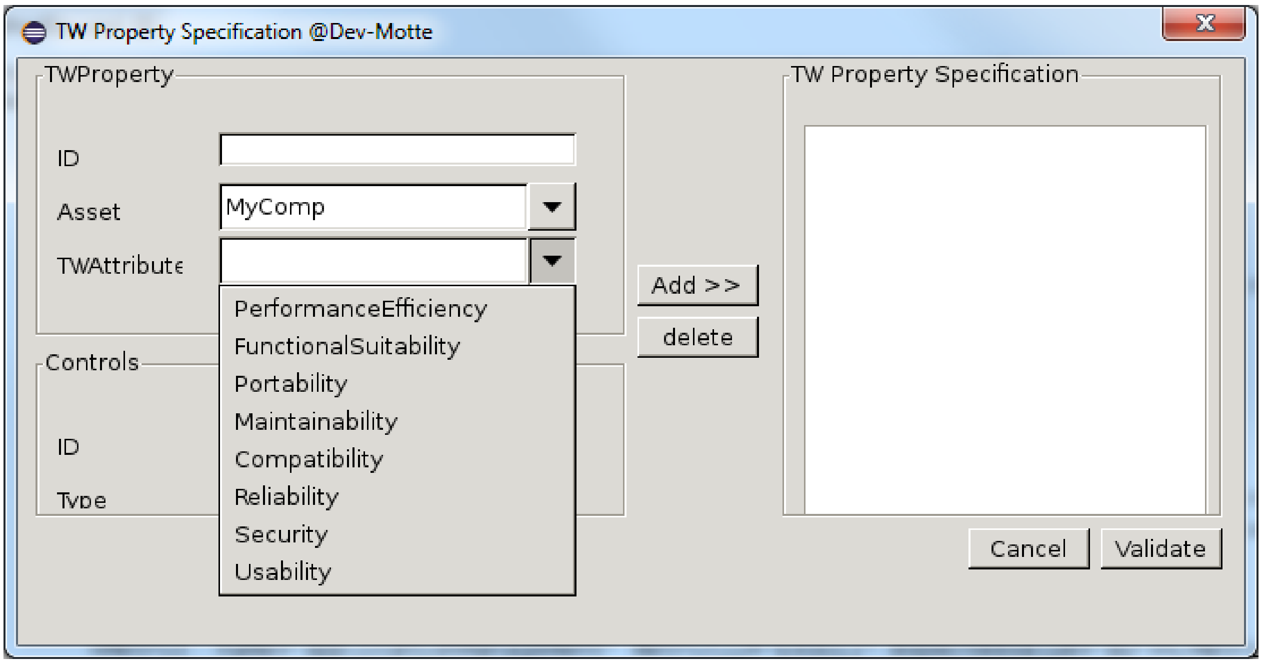
- The Component: a component could be associated with another component, some attribute and some stakeholder.
- The attributes:, the certifier may select their type as InputParameter or OutputParameter and choose if this attribute is selected for the evaluation;
- The Stakeholder:, the certifier may select the type as End-user, System or Service. All the defined stakeholders are displayed in the stakeholders list for reuse if needed.

The second step of this workflow is the definition of the possible threats applicable to the certified VNF. Just select the component and then select the associated threats.



☐ Internal_failures
☐ External_disasters
☐ Malicious_attacks
☐ Malicious_stakeholder

The third step is the definition of the TWattributes. In this step, the certifier associates the asset with a list of TWAttributes. This list could be different from the TWAtrribute list specified into the TWProfile of the VNF. It's the certifier responsibility to add the right TWAttributes. Associated with a TWAttribute, a control must be specified (for the moment, the control are not implemented. In the next release, the controls computed by the Trust Metric Editor could be used as input)

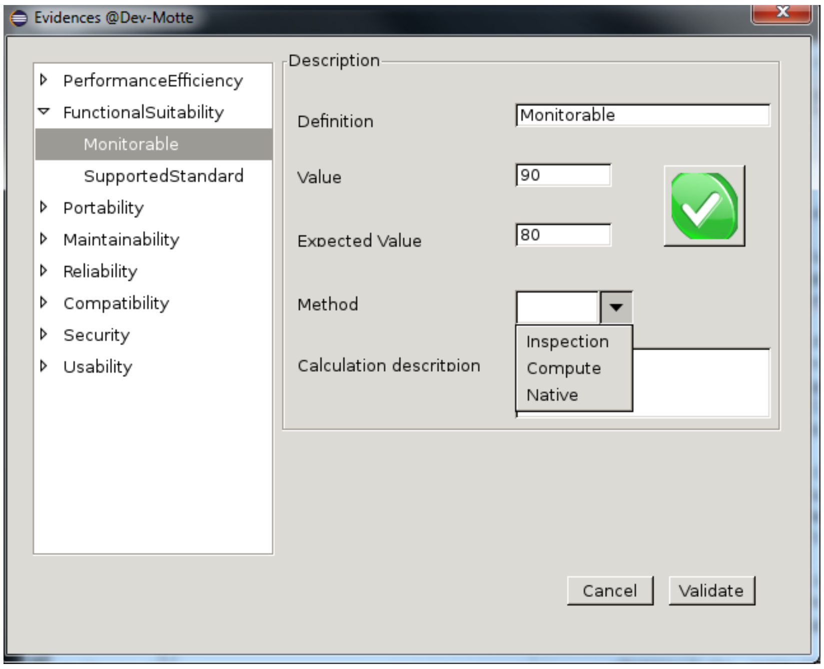


On the right side, all the TW property specification created by the certifier appears. These TWproperties can be deleted if there are not relevant for the certification.

A list of the different TWAtrributes and associated evidence are present into the annex A

The forth step is the evidence computation. At this stage, the certifier analyses the VNF information in order to find evidences to fill the data related to the trustworthiness attributes defined for the system in the evaluation report. The certifier has two possibilities:

- Manual evaluation: In this case, the certifier must enter manually all the metric values required by the certification process. When the certifier selects manual evaluation, the following view appears:



Description

Definition: Monitorable

Value: 90

Expected Value: 80

Method:

Calculation descritpion

For each TWAttribute and the associated metrics, the user sets a value and the method used to find this value. The possible methods are:

  - Compute: if the certifier uses an external tool to evaluate the metric as a Tosca parser to extract information regarding the deployment of the VNF;
  - Inspection: if the certifier explores all the files manually in order to evaluate the metric;
  - Native: if the VNF used natively some good properties (for instance, usage of an OS already hardened, etc…).

  The certifier must set all the values required.

  In this view, the expected value for the metric is displayed indicating the required value. A specific icon helps to show rapidly if the computed value is under/over the expected value.

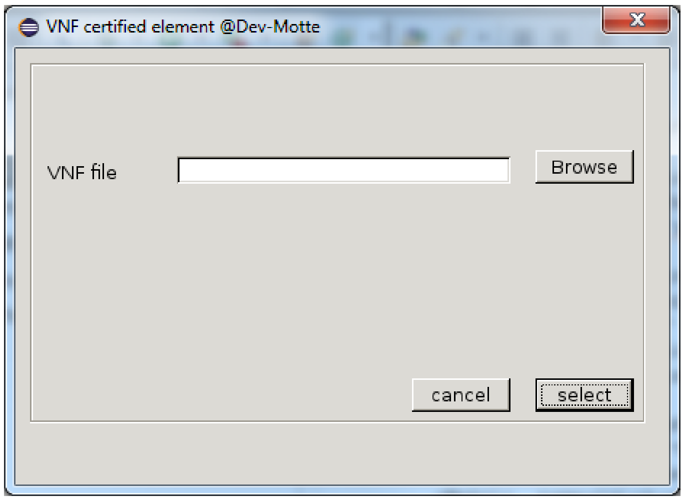
- Automatic evaluation: In this case, the certifier run automatically the static and runtime analysis in order to extract evidences. The computation of all the evidences found will use to fulfill the required TWAttributes metrics.

  In this case, the same view appears but with some not modifiable values. The values are coming from the automatic evaluation made when the certifier clicks on this “Automatic” button.

When the values are coming from the automatic evaluation, the certifier can't modify them. But, if the automatic evaluation can't fulfill some attribute due to missing information, the certifier could enter the value manually.

The automatic computation is dependent of the analysis tools known by the certification tool and the VNF element provided for the automatic computation. If an automatic computation can't be performed on the element, some values could be empty and the certifier could, in this case, realize inspection to complete the analysis.

The last step is the publication of the certificate. During this step, the certifier must select, using the view, the certified VNF element. This is used to link the VNF and the certificate by computing the hash of the VNF and insert it into the certificate.



The DTWC is generated using the data collected during the certification workflow; the hash of the product is inserted into the certificate and signed with the certifier's certificate defined in the preference pages of the IDE. Then, the DTWC is transferred to the certificate repository where all the certificate files are hosted.

3.1.3 The VNF Certification repository

This element provides an API to upload the generated DTwC and to download a specific DTwC. The APIs are:

- Push DTwC

This method is used to push the DTwC into the repository.

```
Method: POST
Path: /uploadDTwCs
Headers:
    o  Content-Type:multipart/form-data
    o  Accept: text/html,application/xhtml+xml,application/xml
```

- Get a DTwC

To search for a specific DTwC, a query using the hash of the VNF

The research by hash

```
Method: GET
```

```
Path : /downloadDTwCByHash/{hash}")
```

The complete path of the service is dependent of the installation. By default, it's https://hostname:XXXX/CertificationRepository/rest/files/downloadDTwCByHash/YYYYYY with

- Hostname = the name of the host where the repository is deployed
- XXXX = the open port of tomcat
- YYYY = the hash of the VNF researched

3.2 Programmer Guide

This section is not applicable because the enabler is not programmable

4 Unit Tests

This section describes unit tests that validate the different functionalities of the enabler.

4.1 Information about Unit Tests

Tests are manual tests with visual inspections. To perform the tests, some input files are provided with the enabler.

4.2 Unit Test 1

This test is relevant for the certification HMI

The goal of this test is to check the good creation of the certification project using the certification wizard provided by the eclipse VNF certification plug-in. for that, we must follow the user guide and create a new certification project.

- If we select a valid TWProfile file (TWProfile.xml) and a valid Tosca file (tosca.yaml) (The files are provided in test repository of the deliverable), the project is created and the previous files are copied into the Certification directory of the project.
- If we select wrong files (use the toscaBad.yaml for instance), the certification wizard failed and the project is not created.

This test could be performed using a csar archive (the csartest.zip file) instead of the tosca.yaml file.

4.3 Unit Test 2

This test is relevant for the certification HMI

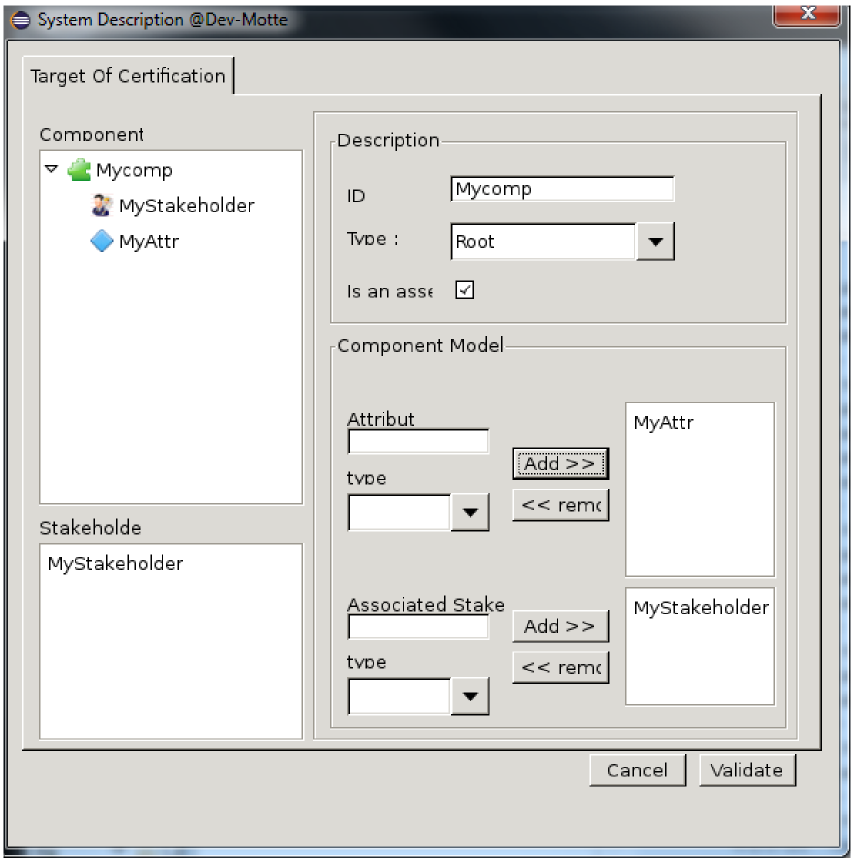
The goal of this test is to check the correct generation of the DTwC. For that, using the project previously created, start the certification by selecting the project and, into the contextual menu, select the certification action.

4.3.1 The software description

- Add a new component MyComp which is a root component. This component is an asset

- Add a new attribute MyAttr which is a Host.
- Add a new stakeholder MyStakeholder which is a Human

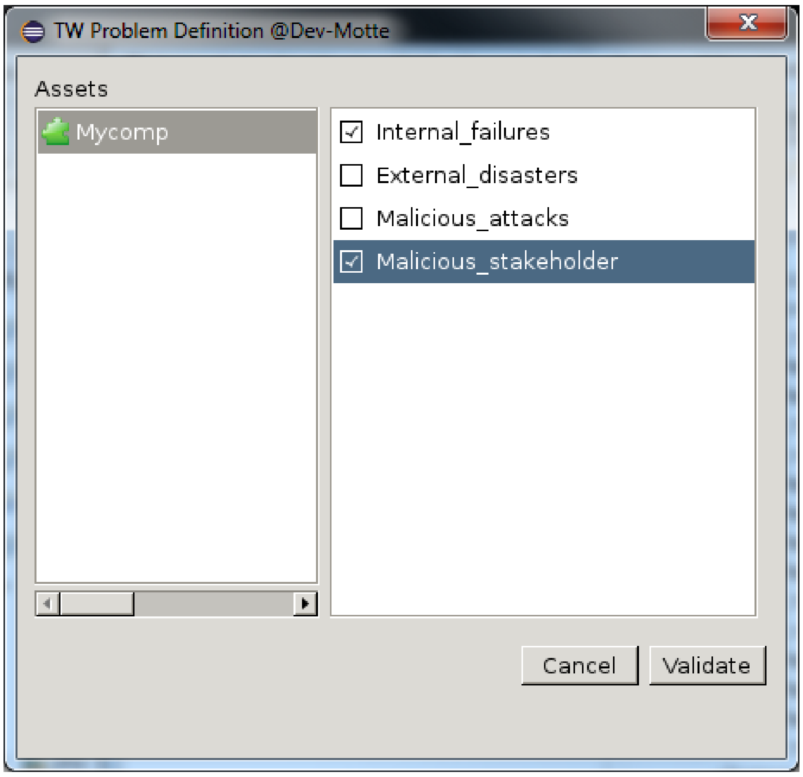
The result must be the following:



At the end of this phase, validate the input. The view must be closed.

4.3.2 The Trustworthiness Problem definition

Select the Asset "MyComp" and check the elements "Internal_failures" and "Malicious_stakeholder"



- [✓] Internal_failures
- [ ] External_disasters
- [ ] Malicious_attacks
- [✓] Malicious_stakeholder

This list of threats is one of the results of the D2.2 Trust Model [5] document

At the end of this phase, validate the input.

4.3.3 The Trustworthiness Property Specification

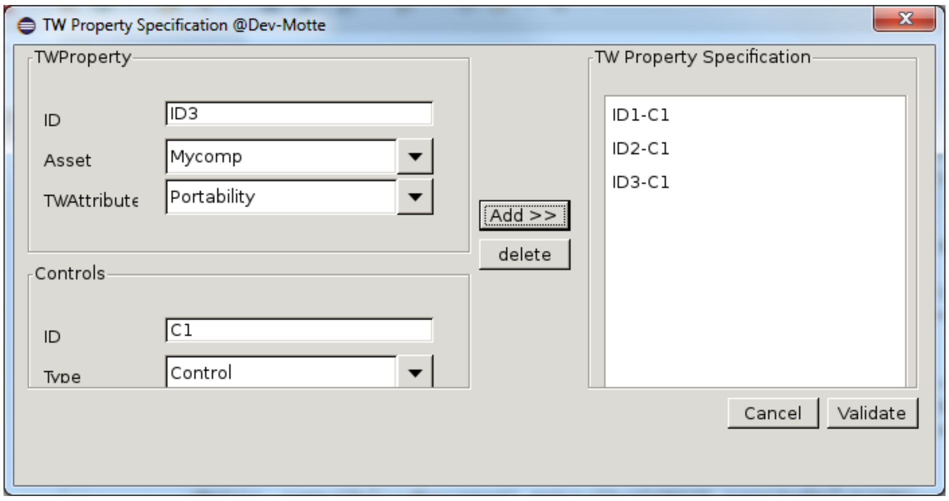
In this step, we must assign the required TWAttributes to the selected asset. So, for each TWAttribute:

- Set an ID for the TWProperty
- Select the asset MyComp
- Select the TWAttribute required
- Set an ID for the control (not used but needed)
- Set the Type Control (not used but needed)
- Click on the add button

This previous list of action must be performed for the following required TWAttributes:

- PerformanceEfficiency
- FunctionalSuitability
- Portablility

The result is the following:


TW Property Specification @Dev-Motte
TWProperty
Add >>
delete
Controls
TW Property Specification
Cancel
Validate

ID: ID3
Asset: Mycomp
TWAttribute: Portability

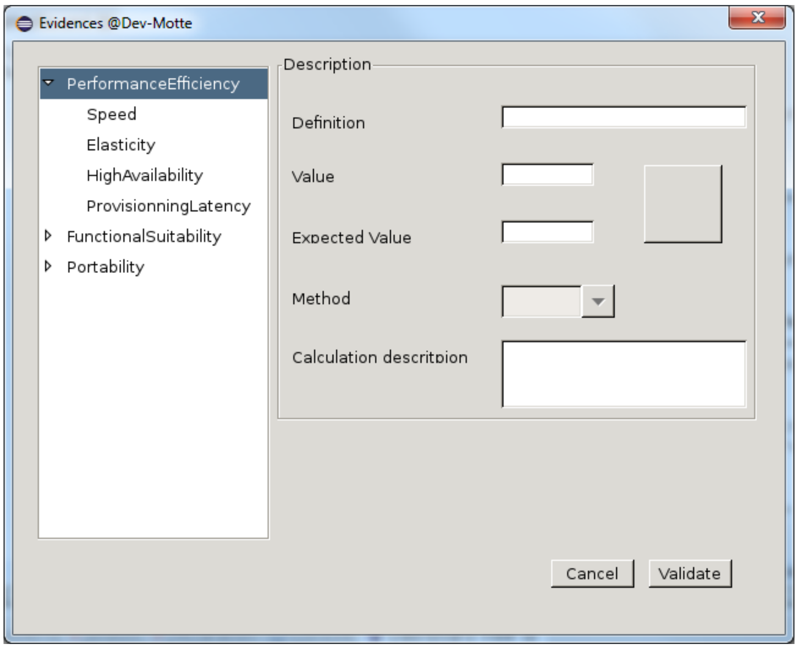
ID: C1
Type: Control

ID1-C1
ID2-C1
ID3-C1

At the end of this phase, validate the input using the validate button.

4.3.4 The Evidences

Select the manual button. The following view must appear

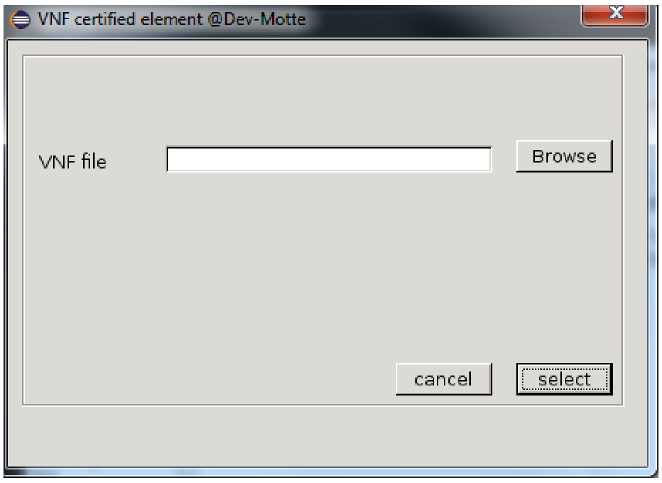


Select the speed element and enter the Value 90 with the method "inspection"

4.3.5 The DTWC

This phase is the generation of the DTwC.

The following view appears:



To link the VNF and the certificate, the hash of the VNF is inserted into the DTwC file. For that, using this view, select the VNF file tested during the certification (The VNF file is the image of the VNF used to deploy the functionality). Click on the select button and wait the confirmation.

In the certification repository, a directory based on the date and time of the certification is created. Into this directory, the certification file must be present

Into this file:

- check if the system description is present

```
<!-- SystemDescription -->

<owl:NamedIndividual rdf:about="SystemDescription">
  <rdf:type
rdf:resource="http://www.optet.eu/ns/DigitalTrustworthinessCertificate#SystemDescription"/>
  <hasStakeholder rdf:resource="MyStakeholder"/>
  <hasAsset rdf:resource="Mycomp"/>
  <hasPart rdf:resource="Mycomp"/>
</owl:NamedIndividual>

<!-- MyAttr -->
<owl:NamedIndividual rdf:about="MyAttr">
  <rdf:type rdf:resource="http://www.optet.eu/ns/DigitalTrustworthinessCertificate#Attribute"/>
  <type rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Host</type>
  <ID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">MyAttr</ID>
</owl:NamedIndividual>

<!-- MyStakeholder -->
<owl:NamedIndividual rdf:about="MyStakeholder">
  <rdf:type rdf:resource="http://www.optet.eu/ns/DigitalTrustworthinessCertificate#Stakeholder"/>
  <type rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Host</type>
  <ID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">MyStakeholder</ID>
</owl:NamedIndividual>
```

```
  <!-- Mycomp -->
  <owl:NamedIndividual rdf:about="Mycomp">
    <rdf:type rdf:resource="http://www.optet.eu/ns/DigitalTrustworthinessCertificate#Component"/>
    <ID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Mycomp</ID>
    <type rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Root</type>
    <inTargetOfEvaluation
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">true</inTargetOfEvaluation>
    <hasAssociatedStakeholder rdf:resource="MyStakeholder"/>
    <hasPart rdf:resource="MycompComponentName"/>
  </owl:NamedIndividual>
```

- Check if the Trustworthiness Problem definition is present

```
  <!-- Mycomp-Internal_failures -->

  <owl:NamedIndividual rdf:about="Mycomp-Internal_failures">
    <rdf:type
rdf:resource="http://www.optet.eu/ns/DigitalTrustworthinessCertificate#TWProblemDefinition"/>
    <ID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Mycomp-Internal_failures</ID>
    <hasThreat rdf:resource="Internal_failures"/>
    <affectsAsset rdf:resource="Mycomp"/>
  </owl:NamedIndividual>



  <!-- Mycomp-Malicious_stakeholder -->

  <owl:NamedIndividual rdf:about="Mycomp-Malicious_stakeholder">
    <rdf:type
rdf:resource="http://www.optet.eu/ns/DigitalTrustworthinessCertificate#TWProblemDefinition"/>
    <ID                      rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Mycomp-
Malicious_stakeholder</ID>
    <hasThreat rdf:resource="Malicious_stakeholder"/>
    <affectsAsset rdf:resource="Mycomp"/>
  </owl:NamedIndividual>
```

- Check if the Trustworthiness Problem Specification is present

```
  <!-- ID1 -->
  <owl:NamedIndividual rdf:about="ID1">
    <rdf:type rdf:resource="http://www.optet.eu/ns/DigitalTrustworthinessCertificate#TWProperty"/>
    <ID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">ID1</ID>
    <hasAsset rdf:resource="Mycomp"/>
    <hasTWAttribute rdf:resource="PerformanceEfficiency"/>
  </owl:NamedIndividual>
  <!-- ID1-C1 -->
  <owl:NamedIndividual rdf:about="ID1-C1">
    <rdf:type
rdf:resource="http://www.optet.eu/ns/DigitalTrustworthinessCertificate#TWPropertySpecification"/>
    <ID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">ID1-C1</ID>
    <hasControl rdf:resource="C1"/>
    <hasTWProperty rdf:resource="ID1"/>
  </owl:NamedIndividual>

  <!-- ID2 -->
  <owl:NamedIndividual rdf:about="ID2">
    <rdf:type rdf:resource="http://www.optet.eu/ns/DigitalTrustworthinessCertificate#TWProperty"/>
    <ID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">ID2</ID>
    <hasTWAttribute rdf:resource="FunctionalSuitability"/>
    <hasAsset rdf:resource="Mycomp"/>
  </owl:NamedIndividual>

  <!-- ID2-C1 -->
  <owl:NamedIndividual rdf:about="ID2-C1">
    <rdf:type
rdf:resource="http://www.optet.eu/ns/DigitalTrustworthinessCertificate#TWPropertySpecification"/>
    <ID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">ID2-C1</ID>
    <hasControl rdf:resource="C1"/>
    <hasTWProperty rdf:resource="ID2"/>
  </owl:NamedIndividual>

  <!-- ID3 -->
  <owl:NamedIndividual rdf:about="ID3">
    <rdf:type rdf:resource="http://www.optet.eu/ns/DigitalTrustworthinessCertificate#TWProperty"/>
    <ID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">ID3</ID>
    <hasAsset rdf:resource="Mycomp"/>
    <hasTWAttribute rdf:resource="Portability"/>
  </owl:NamedIndividual>

  <!-- ID3-C1 -->
  <owl:NamedIndividual rdf:about="ID3-C1">
    <rdf:type
rdf:resource="http://www.optet.eu/ns/DigitalTrustworthinessCertificate#TWPropertySpecification"/>
    <ID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">ID3-C1</ID>
    <hasControl rdf:resource="C1"/>
    <hasTWProperty rdf:resource="ID3"/>
  </owl:NamedIndividual>
```

- Check if the Evidences are present. In the context of the Evidences, check the values tagged as inspection and the values added manually during the certification process

```
<!-- EV0004-ID1 -->
<owl:NamedIndividual rdf:about="EV0004-ID1">
  <rdf:type rdf:resource="http://www.optet.eu/ns/DigitalTrustworthinessCertificate#Metric"/>
  <hasUnit rdf:datatype="http://www.w3.org/2001/XMLSchema#string"/>
  <hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string">90</hasValue>
  <ID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EV0004</ID>
  <type rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Inspection</type>
</owl:NamedIndividual>
```

- Check if the hash is present

```
<owl:NamedIndividual rdf:about="dzdza">
  <rdf:type rdf:resource="http://www.optet.eu/ns/DigitalTrustworthinessCertificate#DTWC"/>
  <hasHash
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">1336d3ee3bb217f321a717095445ae1d6aec
ad2</hasHash>
  <hasEvidence rdf:resource="FunctionalSuitabilityEv"/>
  <hasTWPropertySpecification rdf:resource="ID1-C1"/>
  <hasTWPropertySpecification rdf:resource="ID2-C1"/>
  <hasTWPropertySpecification rdf:resource="ID3-C1"/>
  <hasTWProblemDefinition rdf:resource="Mycomp-Internal_failures"/>
  <hasTWProblemDefinition rdf:resource="Mycomp-Malicious_stakeholder"/>
  <hasEvidence rdf:resource="PerformanceEfficiencyEv"/>
  <hasEvidence rdf:resource="PortabilityEv"/>
  <hasPart rdf:resource="SystemDescription"/>
</owl:NamedIndividual>
```

- Check if the file is signed

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#"><SignedInfo><CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315#WithComments"/><SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/><Reference
URI=""><Transforms><Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/></Transforms><DigestMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/><DigestValue>JMbJGk/mGHOzYWEHnVFU
4nJn5t/xqh4VBCGZCUwpfxs=</DigestValue></Reference></SignedInfo><SignatureValue>FWcZhH++ry6
XZMTkDJ2d0TprN4+pTYwrPXp0LK5JY7MEB5dUsxtY54dvMvdwu5oR5IfXnZwDbeHA
iJ8OCPJ52yHfjg8zi0wrggiITXAC5dTBzKPrvKEc4u9tRAq0gxAIx/aRfEgkyJrXceokmrMD+brN
g+4RsQPEJ+kYp0oQSbEKkqPUw7yAzZcWMe9IuXdkQyQ1H9hiYlAyT7pubGcz+qHuQ6HcrTN4+b/s
qLBq71z1/ByBh3aApMi+JicdLlNUwKOXGb4ZocAWdQiivky218RKONPD0NtOIGDwzvbY5udFr+pI
YEUTA16L6DTjmQSc4yevMm55CNQEHytx7ToJ4g==</SignatureValue><KeyInfo><KeyValue><RSAKeyVal
ue><Modulus>lJgmuFl0ofNP94RzOKKhjUG4monext81Bkuu2lizKkwKReUKPFAl5SFGaWyhJEwbiFB+S4Or
Uysc
GnmMR+VD5oBzh/jSpj9nCFaRb72QW0N/AJqr9je/LvpDyc5FzYAfl4OIGeCexHv6exoQgUXevSjq
1GKp+Lr1Ec4zPAJhAESj1BhcBCgCGomcFDnKpHUvjwWc/PIoEy8S7s16N2gYr6LMJaYSxLjUu66W
```

p72M9MTQtNF/TYfMIYbt7UI1+B637IMAhKxy5XR4Pp0rmFtZZY2A74RVx3jCTIdyh6/FskYqXLZM
dSSUp+3E2xb6On71OXLParFVdnFOgTeCwn1ncQ==</Modulus><Exponent>AQAB</Exponent></RSAKey
Value></KeyValue></KeyInfo></Signature>

If all the verification are Ok, the test is succeed.

4.4 Unit Test 3

This test is relevant for the certification repository. The files used are present into the test directory of the delivery.

1. Push a valid DTwC into the repository.

Take the DTWC.xml file and push it into the repository using the POST request.

```
curl -i -k -X POST -H "Content-type:multipart/form-data" -F "file=@DTWC.xml"
https://localhost:8444/CertificationRepositorCls^Crest/files/uploadDTwCs
```

The result is a valid request (Return code: 200) and the creation off the file into the repository.

2. Push an invalid DTwC into the repository

Take the EmptyDTWC.xml file and push it into the repository using a POST request

```
curl -i -k -X POST -H "Content-type:multipart/form-data" -F "file=@EmptyDTWC.xml"
https://localhost:8444/CertificationRepository/rest/files/uploadDTwCs
```

The result is an invalid request (Return cod : 400 Bad Request)

3. Retrieve a DTwC using a valid hash

Using a browser, enter the following URL (Adapt the path following the tomcat configuration)

https://localhost:8444/CertificationRepository/rest/files/downloadDTwCByHash/985ae4b2e9682c685a7452f1696cb11d1c214

The browser must offer the possibility to download the DTwC.

The same could be realized using the command

```
Wget --no-check-certificate
https://localhost:8444/CertificationRepository/rest/files/downloadDTwCByHash/985ae4b2e9682c685a745
2f1696cb11d1c214
```

Retrieve a DTwC using an invalid hash (Adapt the path following the tomcat configuration)

Perform the following command:

```
Wget --no-check-certificate
https://localhost:8444/CertificationRepository/rest/files/downloadDTwCByHash/985ae4b2e9682c685a745
2f1696cb11d1c214XXXXXX
```

The result is an invalid request (Return code: 400 Bad Request)

5 Acknowledgements

We'd like to thank the Optet partners (especially the WP4) for its contribution on the certification methodology aspects and also NEXU for his contribution around the certification process definition.

6 Abbreviations

5G-PPP	5G Infrastructure Public Private Partnership
DTwC	Digital Trustworthiness Certificate
TWAttribute	Trustworthiness Attribute
VNF	Virtualized network function
VMNO	Virtual Mobile Network Operator
OAM	Operations, administration and maintenance
VNFC	Virtualized network functions Component
VDU	Virtual Deployment Unit
NF	Network function
COTS	Commercial off-the-shelf
OS	Operating system
Certification Body	legal entity or a defined part of a legal entity that certify element based on test and documentation

7 References

[1] Optet consortium, "https://www.optet.eu/," [Online].

[2] Fiware consortium, "http://catalogue.fiware.org/," [Online].

[3] Consortium, 5G-Ensure. D3.2 : Open Specification.

[4] OASIS, "Tosca specification," [Online]. Available: http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html.

[5] 5G-Ensure , D2.2 Trsut Model.

[6] 3GPPP consortium, [Online]. Available: http://www.3gpp.org/.

[7] ETSI, [Online]. Available: http://www.etsi.org/.

[8] ISO, [Online]. Available: http://www.iso.org/iso/home/standards/management-standards/iso27001.htm?=.

[9] W. Rudin, Functional Analysis, McGraw-Hill, 1973.

A Tosca file example

This following file represents a VNF description based on the Tosca representation. This VNF file represents a virtual function composed by two unitary functions. Each unitary function is composed by three components. The different components are linked together using Virtual links (Two in this example)

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0

 node_templates:
  VDU1:
    type: tosca.nodes.nfv.VDU.Tacker

    capabilities:
      high_availability: tosca.capabilities.nfv.HA
      monitoring_parameter: tosca.capabilities.nfv.Metric

    properties:
      image: cirros-0.3.4-x86_64-uec
      flavor: m1.tiny
      availability_zone: nova
      mgmt_driver: noop
      config: |
        param0: key1
        param1: key2
  CP11:
    type: tosca.nodes.nfv.CP.Tacker
    properties:
      management: true
      anti_spoofing_protection: false
    requirements:
      - virtualLink:
          node: VL1
      - virtualBinding:
          node: VDU1

  CP12:
    type: tosca.nodes.nfv.CP.Tacker
    properties:
      anti_spoofing_protection: false
    requirements:
```

```
    - virtualLink:
        node: VL2
    - virtualBinding:
        node: VDU1

CP13:
  type: tosca.nodes.nfv.CP.Tacker
  properties:
    anti_spoofing_protection: false
  requirements:
    - virtualLink:
        node: VL3
    - virtualBinding:
        node: VDU1

VDU2:
  type: tosca.nodes.nfv.VDU.Tacker
  capabilities:
    high_availability:
      type : tosca.capabilities.nfv.HA
    monitoring_parameter:
      type: tosca.capabilities.nfv.Metric

  properties:
    image: cirros-0.3.4-x86_64-uec
    flavor: m1.tiny
    availability_zone: nova
    mgmt_driver: noop
    config: |
      param0: key1
      param1: key2


CP21:
  type: tosca.nodes.nfv.CP.Tacker
  properties:
    management: true
    anti_spoofing_protection: false
  requirements:
    - virtualLink:
        node: VL1
    - virtualBinding:
        node: VDU2

CP22:
  type: tosca.nodes.nfv.CP.Tacker
```

```
   properties:
     anti_spoofing_protection: false
   requirements:
     - virtualLink:
         node: VL2
     - virtualBinding:
         node: VDU2

 CP23:
   type: tosca.nodes.nfv.CP.Tacker
   properties:
     anti_spoofing_protection: false
   requirements:
     - virtualLink:
         node: VL3
     - virtualBinding:
         node: VDU2

 VL1:
   type: tosca.nodes.nfv.VL
   properties:
     network_name: net_mgmt
     vendor: Tacker

 VL2:
   type: tosca.nodes.nfv.VL
   properties:
     network_name: net0
     vendor: Tacker

 VL3:
   type: tosca.nodes.nfv.VL
   properties:
     network_name: net1
     vendor: Tacker
```

B TWAttributes and Evidences for VNF

This list is the consolidation of the works performed by the 3GPPP consortium [6]and the ETSI [7]I around the principal VNF characteristics. The different evidences are classified into families' based on the ISO 27001 classification [8] .

Functional Suitability (Functional Completeness, Functional Correctness, Functional Appropriateness)	
supported standard	The VNF could support some standards
Monitorable	The VNF could provide monitoring capabilities (memory-consumption, CPU,

	bandwidth, etc…)

Performance efficiency (Time-behavior, Resource utilization, Capacity)	
provisioning latency	This metric measures the time taken to on-board a VNF, provision it and create a VNF. This impacts service latency, and in the quality of service experienced by end users.
Speed	The time between the start of the VNF and is availability.
Elasticity	This property describes the ability of a VNF to define and perform elastic operations as well as the type of scaling it can use (up/down, out/in). Every VNF will belong to exactly one of these categories. - No elasticity: The VNF requires a fixed set of resources that cannot be changed. - Elasticity by scaling up/down only: The NFV framework can increase or decrease the size, performance or bandwidth of the virtual resources. - Elasticity by scaling out/in only: The VNF is able to perform scaling operations by separately adding/removing instances of specified VNFCs. - Elasticity in either dimension: The VNF has VNFCs that may be scaled out/in, up/down or both.
High availability	Defines redundancy model to ensure high availability examples include: • ActiveActive: Implies that two instance of the same VDU will co-exists with continuous data synchronization. • ActivePassive: Implies that two instance of the same VDU will co-exists without any data synchronization. • Nothing

Compatibility (Co-existence, interoperability)	
supported standard	The VNF could support some standard
Hardware Independence	Some NFs might not fully realize hardware independence for various reasons (implementation history, performance/latency aspects, special requirements). Therefore this clause categorizes the NFs in the way they can realize hardware independence - COTS-Ready: The NF fully realizes independence from the physical infrastructure, allowing the network function to be run on any appropriate physical infrastructure. - Partly COTS-Ready: The NF in some of its VNFCs fully realizes independence from the physical infrastructure, while some VNFCs can run only on a specified type of hardware - Hardware dependent: The NF requires a specified type of hardware for all its components
Virtualization and Container Awareness	Some NFs might not be able to run in any virtualized environment for various reasons (implementation history, performance/latency aspects, special

	requirements). Some NFs that are able to run in a virtualized environment in the form of a VNF are designed for OS containers, while other are designed for hypervisor virtualization. These categories describe the NF's software relation to the various container and virtualization technologies. In most cases a NF will belong to exactly one category, but some categories can also be combined. • Hypervisor agnostic: The network function is able to run in a virtual machine/hypervisor environment. Its software has no dependency to the virtual machine/hypervisor environment, so it can run unchanged on different hypervisors or also on native hardware (typically COTS hardware). • Hypervisor dependent: The network function is able to run in a virtual machine or some other abstraction mechanism between the logical environment seen by the network function and the actual physical infrastructure. But the VNF implementation is dependent on the virtual machine/hypervisor environment it is running in. • Operating System Containers: The NF software is designed for use with OS container technology for its deployment. Note that there is always some dependency between application and OS in this technology. • Higher layer container technologies: The NF software is able to run on some higher layer container technology such as Java virtual machines. • Not virtualized and no container technology: The network function cannot run in a virtual machine/hypervisor environment and does not use other container technologies. Therefore it could be seen outside the scope of NFV. I • Partly virtualized: Some components of the NF can be virtualized, some may be virtualization aware, some not virtualized.

usability (Appropriateness recognisability, learnability, operability, user error protection, user interface aesthetics, accessibility)	
Policy Management - Fully policy based VNF - Not policy based VNF	This property describes the ability of a VNF to support dynamic rule-based provisioning and management needed for automated operational tasks (e.g. scale-in, scale-out, or migration based on threshold crossing). There are some or more policies for each automatic operation task of VNF. Every VNF will belong to exactly one of the categories below. - Fully policy based VNF: NFV orchestration and management provides full set of provisioning and management policies for this VNF. - Not policy based VNF: NFV orchestration and management will not provide any provisioning and management VNF policies for this VNF. VNF will provide policy management by itself.
Application Management - No management - Standardized management interface - Proprietary management	This property describes what type of application management needs to be used for the VNF. A VNF can belong to the first category indicated below or allow a combination of the other categories. - No management: The VNF does not require management operations during its lifetime. All configuration shall be done during installation. - Standardized management interface: The VNF provides a standard

interface - Multiple service providers' management interfaces	management interface to be used by a separate Element Management (EM). - Proprietary management interface: The VNF provides a non-standard OAM interface to be used by a separate Element Management by the same VNF Provider. - Multiple service providers' management interfaces.

Reliability (Maturity, availability, Fault tolerance, recoverability)	
Reliability	VNF Fault Management is somehow related to reliability in the sense that reliability mechanisms are dependent on fault detection and fault reporting mechanisms

Security (Confidentiality, integrity, non-repudiation, accountability, authenticity)	
Appropriate use of encryption techniques	The VNF could ensure that he use : - the right trusted algorithms - the right key length - correctly the certificates
Integrity	The intent behind this principle is that VNFs are signed and that the static component of the VNF can be verified during boot, run-time, suspension and transfer. Additionally, data used by the VNFs is properly structured and hence its integrity can be checked
Secure boot and secure crash	Boot only in the right image avoid information leakage by VM crashing
Software hardened	Check that the VNF is already hardened at software level using for example: - Best practice development for the VNF - Operation System hardening (access file, permission, etc…) - Hardening of the communication elements
Secured management (Authentication, authorization)	Support a secure management of VNF sets by other third-party entities
Secure communication (TLS, etc…)	Ensure that end-user data that is transiting a network element or communication link is not diverted or intercepted as it flows between the endpoints (without an authorized access).
Data protection & privacy	Ensure that the VNF uses mechanisms to protect data like: - usage of secure storage - The operator of NFV must guarantee that no third party can access the user's VNFs or any information of the VNFs without the user's permission (access control)

Maintainability (reusability, analysability, Modifiability, testability)

Internal Structure	The internal structure of the VNF could be more or less complex depending of the number of VDU used and interconnections.
Patch management	The VNF could provide mechanism to manage the security patch all along his life.
Integrity management	
Monitorable	The VNF could provide monitoring capabilities (memory-consumption, CPU, bandwidth, etc…)

Portability (adaptability, instability, replaceability)	
Migration operations	This property describes the ability of a VNF to perform migration operations of its resources. A VNF can belong to the first category or allow any of the other attributes in combination. - No live migration supported: The VNF does not support that its resources are migrated during their time of assignment. - Live migration supported: The VNF allows its resources to be migrated to a different NFVI resources using hypervisor technology. Typically this includes specifying live migration policies, e.g. performance parameters. - Migration partially supported: Some VNFC support migration while others do not. - Other migration mechanisms: The VNF allows other mechanisms (e.g. shutdown and restart on different hardware) to change the resource assignments dynamically. These mechanisms might make use of the VNF's internal redundancy schemes
Hardware Independence	Some NFs might not fully realize hardware independence for various reasons (implementation history, performance/latency aspects, special requirements). Therefore this clause categorizes the NFs in the way they can realize hardware independence - COTS-Ready: The NF fully realizes independence from the physical infrastructure, allowing the network function to be run on any appropriate physical infrastructure. - Partly COTS-Ready: The NF in some of its VNFCs fully realizes independence from the physical infrastructure, while some VNFCs can run only on a specified type of hardware - Hardware dependent: The NF requires a specified type of hardware for all its components (it is still a PNF, so not fully in scope of NFV).
Location Awareness	This property describes dependencies of the VNF or some of its components on a position in the topology or geography
Virtualization and Container Awareness	Some NFs might not be able to run in any virtualized environment for various reasons (implementation history, performance/latency aspects, special requirements). Some NFs that are able to run in a virtualized environment in the form of a VNF are designed for OS containers, while other are designed for hypervisor virtualization. These categories describe the NF's software relation to the various container and virtualization technologies. In most cases a NF will belong to exactly one category, but some categories can also

	be combined. • Hypervisor agnostic: The network function is able to run in a virtual machine/hypervisor environment. Its software has no dependency to the virtual machine/hypervisor environment, so it can run unchanged on different hypervisors or also on native hardware (typically COTS hardware). • Hypervisor dependent: The network function is able to run in a virtual machine or some other abstraction mechanism between the logical environment seen by the network function and the actual physical infrastructure. But the VNF implementation is dependent on the virtual machine/hypervisor environment it is running in. • Operating System Containers: The NF software is designed for use with OS container technology for its deployment. Note that there is always some dependency between application and OS in this technology. • Higher layer container technologies: The NF software is able to run on some higher layer container technology such as Java virtual machines. • Not virtualized and no container technology: The network function cannot run in a virtual machine/hypervisor environment and does not use other container technologies. Therefore it could be seen outside the scope of NFV. • Partly virtualized: Some components of the NF can be virtualized, some may be virtualization aware, some not virtualized.
VNF State	This property describes whether the VNF needs to keep state information. Every VNF will belong to exactly one of these categories. - Statefull operation: The definition of the VNF's task or interfaces requires that the VNF needs to keep state information. - Stateless operation: The VNF does not require keeping state information.

Deliverable D3.4
5G-PPP Security Enablers Documentation (v1.0)
Trust Metric Enabler

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	30.09.2016	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	VTT: Pekka Ruuska	

Contents

1 Introduction

As explained in the manuals of the Micro-Segmentation Security Enabler and the Security Monitor for 5G Micro-Segments micro-segmentation may effectively improve security of the 5G networks [1] [2]. The Trust Metric Enabler supports micro-segmentation and it is implemented as an integrated part of the Security Monitor. It should produce a trust metric value which indicates in real-time how a 5G service provider's trust requirements are met in a 5G system.

The Security Monitor for 5G Micro-Segments provides an advanced software framework which is composed of 'big data' technologies: Apache Kafka and Apache Spark. This Complex Event Processing (CEP) Framework provides a tool chain and input for the Trust Metric Enabler. The CEP system can handle large amounts of event streams in near real-time. The first prototype of the Security Monitor should collect traffic statistics available from switches [1]. In the future releases, more event information from 5G specific metrics, Key Performance Indicators (KPI), counters and deeper packet analysis are planned. The Trust Metric Enabler requires some critical KPIs and counter data in real-time as input. The first prototype version of Trust Metric Enabler is implemented as Python scripts that will be integrated to Sparks' API (Application Programming Interface) in Release 2. Since the Release 1 version of the Security Monitor cannot provide the input data as streams which the final version of Trust Metric Enabler needs, test runs with the Release 1 version of Trust Metric Enabler are limited to running with text file input.

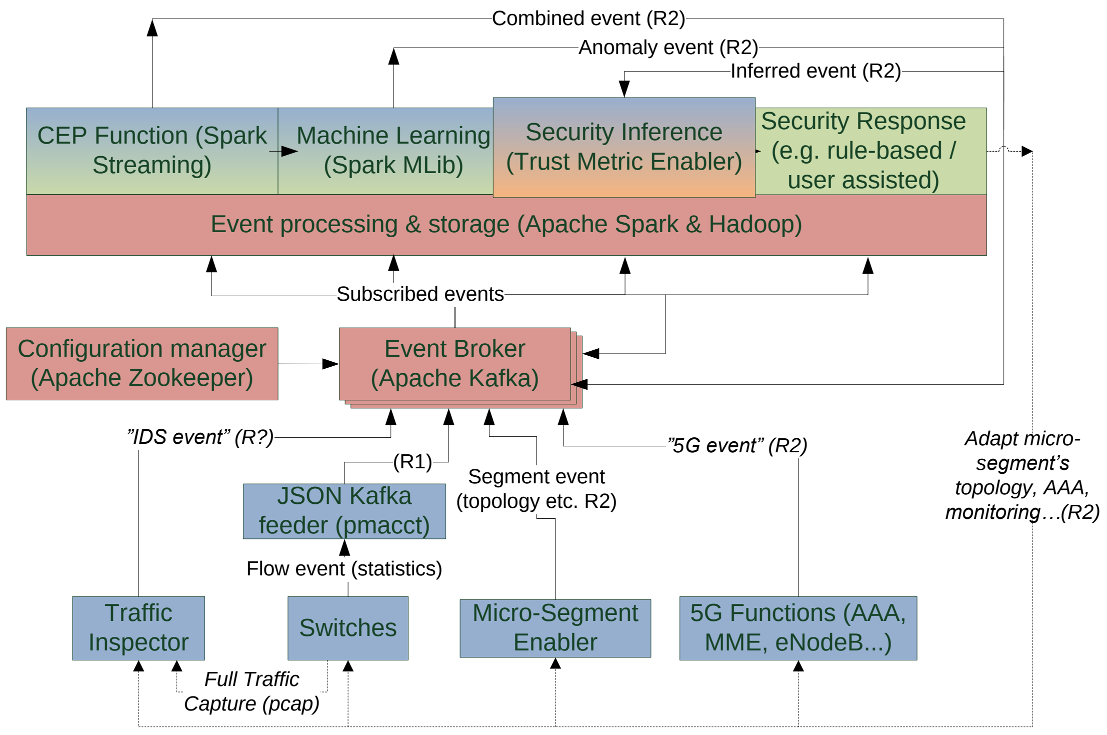
The manual is organized as follows: In Section 2 is described the framework architecture and provided quick configuration and installation instructions. Section 3 describes how the enabler is used In Section 4, some basic tests are introduced. Abbreviations are listed in Section 5.

2 Installation and Administration Guide

The Trust Metric Enabler consists of Python Scripts which are implemented through **Apache Sparks'** API (Application Program Interface).

2.1 System Requirements

The Trust Metric Enabler is implemented as an integrated part of the Security Monitor. Therefore the Trust Metric Enabler requires the host environment specified in [1] and an installed Security Monitor for 5G Micro-Segments. Figure 1 (first shown in [1]) depicts the architecture of the Security Monitor. In this edited version of the figure Trust Metric Enabler is enhanced to indicate its role more clearly. However, the Figure 1 gives only a rough overview of the monitoring system while the Trust Metric Enabler may also utilize straight input data from the 5G functions, such as MME or eNodeB.



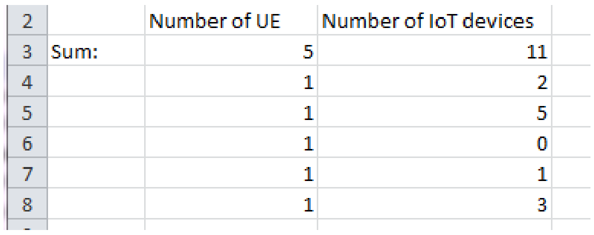
Combined event (R2)
Anomaly event (R2)
Inferred event (R2)
CEP Function (Spark Streaming)
Machine Learning (Spark MLib)
Security Inference (Trust Metric Enabler)
Security Response (e.g. rule-based / user assisted)
Event processing & storage (Apache Spark & Hadoop)
Subscribed events
Configuration manager (Apache Zookeeper)
Event Broker (Apache Kafka)
"IDS event" (R?)
(R1)
Segment event (topology etc. R2)
"5G event" (R2)
Adapt micro-segment's topology, AAA, monitoring…(R2)
JSON Kafka feeder (pmacct)
Flow event (statistics)
Traffic Inspector
Switches
Micro-Segment Enabler
5G Functions (AAA, MME, eNodeB...)
Full Traffic Capture (pcap)

Figure 1: Architecture and technology selections for security event distribution [1]

2.2 Enabler Configuration

Apache Kafka and Apache Spark are distributed with example configurations that may be utilized in simple testing scenarios with one host. Configuration guidelines on more complex scenarios - using e.g. large amount of clustered servers - can be found from the Apache web-site.

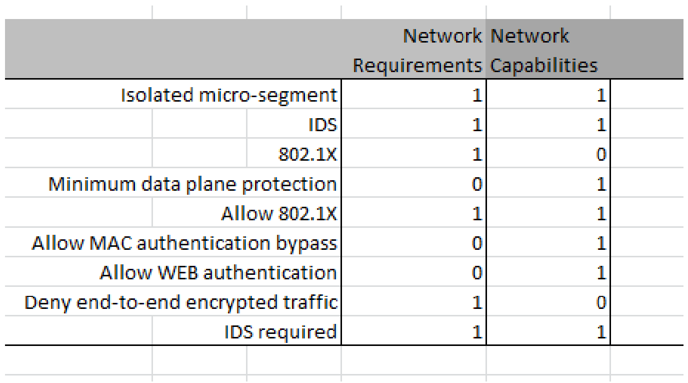
In Release 1, input data for the Trust Metric Enabler's prototype version is presented as .csv-files, in other words Excel-files, which are named `number_of_devices.csv` and `trust_requirements.csv`. The file `number_of_devices.csv` must present a table with two columns and at least one row, as Excel it may look as shown in Figure 2.



2		Number of UE	Number of IoT devices
3	Sum:	5	11
4		1	2
5		1	5
6		1	0
7		1	1
8		1	3

Figure 2: An example of a "number_of_devices.csv" in Excel-format

The file **`trust_requirements.csv`** must present a table with three columns and at least one row. The network requirements must be presented as Boolean variables as shown in Figure 3.



	Network Requirements	Network Capabilities
Isolated micro-segment	1	1
IDS	1	1
802.1X	1	0
Minimum data plane protection	0	1
Allow 802.1X	1	1
Allow MAC authentication bypass	0	1
Allow WEB authentication	0	1
Deny end-to-end encrypted traffic	1	0
IDS required	1	1

Figure 3: An example of a "`trust_requirements.csv`" as an Excel-file

In Release 2 these Excel-files are replaced with input streams from MME, Kafka and the other event consumers which are developed for the Security Monitor for 5G Micro-Segments [1].

2.3 Enabler Installation

The Trust Metric Enabler can be installed by following the procedures described in [1]. Anyway, if not already done for installing any other enabler, an interactive Python shell should be launched. This is done in Spark by installing script bin/pyspark. We also need to import some Spark classes into our Python program [5], or the following line:

```
from pyspark import SparkContext, SparkConf
```

2.4 Troubleshooting

Kafka [4], Spark [5], and Pmacct [3] documentation and web-sites provide component specific troubleshooting information. In Release 1, the Trust Metric Enabler sends an error if it cannot open any of the two input files. That can be solved by checking the path to the csv-files. The enablers output is collected to a log-file. If the log-file cannot be found or is empty, you should check first the Spark and Kafka issues.

3 User and Programmer Guide

3.1 User's Guide

As described in [1], the framework requires a working broker for distributing the monitoring information. The Apache broker and Zookeeper providing configurations for the broker can be launched as shown below:

```
$ cd kafka_2.10-0.9.0.0
$ bin/zookeeper-server-start.sh config/zookeeper.properties
$ bin/kafka-server-start.sh config/server.properties
```

In Release 1 version, the user should first create the Excel-files (specified in Section 2.2)and save them in csv-format into the Hadoop's directory:

```
~/5g/spark/spark-1.6.1-bin-hadoop2.6
```

After this is done the enabler can be activated as any Spark-based event consumer. The program should output data that has been published through Kafka topic called "pmacct.acct". Activation of the Trust Metric Enabler is shown in the following:

```
$ cd ~/5g/spark/spark-1.6.1-bin-hadoop2.6/
$ bin/spark-submit --packages org.apache.spark:spark-streaming-kafka_2.10:1.6.1
myenablers/src/main/python/streaming/trust_metric_enabler.py localhost:9092 pmacct.acct
```

Output from the enabler is logged into file `Trust_Metric_Log.csv`.

3.2 Programmer's Guide

The Trust Metric Enabler is configured in the Hadoop's directory:

```
~/5g/spark/spark-1.6.1-bin-hadoop2.6
```

As the Trust Metric Enabler is an integrated part of the Security Monitor, the programmer's guide presented in Section 3.2 in [1] can be used. However, the Trust Metric Enabler (named here as TrustMetricE) must be initialized to Spark, as follows:

```
conf = SparkConf().setAppName(TrustMetricE).setMaster(local)
sc = SparkContext(conf=conf)
```

In the above Master URL is set as local, which suffices for our Release 1 tests, but when running in a cluster another method is used. After this the PySpark shell is activated. However, in Release 2, the TrustMetricE is linked with other applications and that requires a more complicated approach to submit the bundled applications to Spark's cluster managers [5].

4 Unit Tests

4.1 Unit Test 1

This test is to prove, if Trust Metric Enabler follows its specifications and returns `True` when it should allow a traffic flow as trusted traffic.

To start the test, create the input files (specified in Section 2.2) with Excel and save them in csv-format in the Hadoop's directory (~/5g/spark/spark-1.6.1-bin-hadoop2.6), with parameters set as follows:

In the file **`number_of_devices.csv`** which is shown in Figure 2 set:

```
Number of UE = 1

Number of IOT devices = 0
```

In the file **`trust_requirements.csv`** shown in Figure 3 set:

```
                        Network Requirements       Network Capabilities

Isolated micro-segment           1                      1

IDS                              1                      1

802.1X                           1                      1

Minimum data plane protection    0                      0

Allow 802.1X                     1                      1

Allow MAC authentication bypass  1                      1

Allow WEB authentication         1                      1

Deny end-to-end encrypted traffic0                      0

IDS required                     0                      1
```

Run the test by activating the Trust Metric Enabler as explained in Section 3.1. After the test run, the output file should be found from the same directory as the input files.

From the above input data, the enabler should return `True` into its output file `Trust_Metric_Log.csv`.

4.2 Unit Test 2

After successfully running the Unit Test 1, you should test that the enabler turns its output to `Untrue` when the requirements for trusted traffic in the file **`number_of_devices.csv`** are not satisfied.

First save the input files and the results of the previous tests to another directory. Delete the output files of the previous tests from the Hadoop's directory.

Start this test by changing the parameters in the file **`number_of_devices.csv`** to higher values, e.g.

```
Number of UE = 1

Number of IOT devices = 99
```

And save the new file as csv into file **`number_of_devices.csv`**. Do not change the input file **`trust_requirements.csv`** from the Unit Test 1.

Activate the Trust Metric Enabler as explained in Section 3.1. After the test run, the output file should be found from the same directory as the input files.

From the above input data, the enabler should return `Untrue` into its output file `Trust_Metric_Log.csv`.

You can continue this test to find out when the `True` turns to `Untrue` in the output file. You can insert more lines and varying parameter values on each row in the file **`number_of_devices.csv`** for example as follows:

```
Number of UE = 1

Number of IOT devices = 1

Number of UE = 1

Number of IOT devices = 2

Number of UE = 1

Number of IOT devices = 3

---
```

From the above input data, the enabler should return “`True True True` …” into its output file `Trust_Metric_Log.csv` as long as both “Number of UE” and “Number of IOT devices” do not reach their limits.

4.3 Unit Test 3

After successfully running the Unit Test 2, check that the enabler turns its output to `Untrue` when any of the requirements for trusted traffic presented in the file **`trust_requirements.csv`** are not met.

In this test you should first set the parameters in the files **`number_of_devices.csv`** and **`trust_requirements.csv`** to accepted values (shown in Unit Test 1) and run the test as explained for the Unit Test 2. From the above input data, the enabler should return `True` into its output file `Trust_Metric_Log.csv`.

After each test run you may change the file **`trust_requirements.csv`** while you should keep the file **`number_of_devices.csv`** unchanged. You should check the results from the output file `Trust_Metric_Log.csv`. Checking the trust requirements against the network's capabilities is much more sophisticated than tests which were done in Unit Test 1 and 2. As some of the network requirements are dependent on each other, you should test the enabler by comparing its output with the enabler's specifications presented in Deliverable D3.2.

In this test session you should first keep the network's capabilities unchanged and change the network's trust requirements by one parameter at a time in the file **`trust_requirements.csv`**. You should carefully track the changes that you make in the input files. You should notice the critical requirements, such as when IDS is used, end-to-end encrypted traffic is not allowed.

After successful testing of the changing trust requirements, you should keep the requirements as they are and change the network's capabilities one by one.

When all potential combinations of input parameters are tested, this test is done.

5 Abbreviations

5G-PPP	5G Infrastructure Public Private Partnership
CEP	Complex Event Processing
JSON	JavaScript Object Notation
PCAP	Packet CAPture – application interface to captured packets that are available from libpcap library

	implementations
IDS	Intrusion Detection System
802.1X	IEEE 802.1X Extensible Authentication Protocol over LAN (IEEE8 02)
MME	Mobility Management Entity

6 References

[1] D34 Manual Security Monitor for 5G Micro-Segments

[2] D35 Enabler 5G Micro-Segmentation

[3] The pmacct project. http://www.pmacct.net.

[4] Apache Kafka project. http://kafka.apache.org/

[5] Apache Spark project. http://spark.apache.org/

[6] The Apache Hadoop project. http://hadoop.apache.org

[7] Apache ZooKeeper. https://zookeeper.apache.org/

Deliverable D3.4
5G-PPP Security Enablers Documentation (v1.0)
Trust Builder

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	30.09.2016	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	IT INNOV: Stephen Phillips, Juri Papay, Mike Surridge	

Contents

1 Introduction

The notion of trust in 5G mobile networks if one of the hot topics of research. In this respect we may consider trust between the end users and network operators or trust between the network operators only. Over the years numerous trust models have been suggested, however they must be updated so that these models reflect the requirements of 5G. One of the motivations of our work was to deliver a tool that enables to construct trust networks, to reason about these networks and perform "*what if studies*". The presented Trust Builder uses ontology-based reasoning for analyzing dynamic complex systems. This tool enables the user to identify security threats and take mitigation actions.

Trust Builder is a graphical tool that enables to construct trust networks for representing 5G networks, generate potential threats and validate the model. The output of validation is a modified trust network enriched by features that were not captured by the initial design.

For describing the Trust Builder we use a set of terms for explaining various features if the tool:

- *Core Model* – the core ontology, defining common vocabulary and relationships used in all higher level models.
- *Generic Model* – an ontology defining the typology of Assets, Threats and Controls (security measures) for a given domain (e.g. 5G networks).
- *Design-Time System Model* – an abstract model of a particular system, described in terms of relationships between system specific Asset classes. The design time model can be enriched by specifying which Security Controls. These controls allow to protect the assets, and generate a set of system-specific Threat Classes for describing potential threats to the system.
- *Runtime System Model* – a model using instances of Assets, Threats and Controls for describing what is known about the current state of the system.
- *Domain Modeler* - a software tool for defining a generic domain model.
- *System Modeler* - a software tool for defining a design-time system model in terms of assets and other elements from a suitable generic model.

The presented document is structured as follows:

- Section 2 is the Installation and Administration Guide that describes the system requirements, configuration, installation and troubleshooting.
- Section 3 is the User and the Programmer Guide for Trust Builder. The User Guide represents the bulk of this document and provides a detailed account of the system's functionality.
- Section 4 describes a case study that illustrates a typical usage of Trust Builder.

The Trust Builder is released as a confidential project output to partners under the terms of the 5G-ENSURE consortium agreement.

2 Installation and Administration Guide

This section describes the system requirements, configuration, installation and administration of "*Trust Builder*" software. Trust Builder is released as a confidential project output to partners under the terms of the 5G-ENSURE consortium agreement.

2.1 System Requirements

For running the software requires a Java 8 installation, Tomcat server and a MongoDB server. The dependencies are Java libraries that are managed by Gradle tool. The software dependencies of Trust Builder are summarised in Table 1.

Group	Artefact	Version
org.springframework.boot	spring-boot-starter-data-mongodb	1.4.0
	spring-boot-starter-data-rest	1.4.0
	spring-boot-starter-web	1.4.0
	spring-boot-starter-security	1.4.0
	spring-boot-starter-thymeleaf	1.4.0
	spring-boot-starter-tomcat	1.4.0
org.springframework.mobile	spring-mobile-device	1.1.5
com.googlecode.json-simple	json-simple	1.1
io.jsonwebtoken	jjwt	0.2

Table 1 - System requirements

2.2 Enabler Configuration

The default configuration provides a deployment onto a single machine. The pre-condition is that both Tomcat server and MongoDB are running with the default settings on the same machine. In case the MongoDB is deployed on a remote host then we need to update the *application.properties* file so that it refers to MongoDB's address. The configurable settings are as follows:

```
#tomcat server settings
server.port=8080
server.contextPath=/system-modeller

#MongoDB connection URL
spring.data.mongodb.uri=mongodb://localhost:27017/system-modeller

#MongoDB Users collection names
mongo.user.collection=Users

#Logging levels
logging.level.org.springframework.web=DEBUG

# default users path
# note: users file contains test users (all of them will be stored in the database in the future)
load.users.from.file=true
user.path=src/main/resources/users.json

#Json serialization settings
spring.jackson.serialization.INDENT_OUTPUT=true

#Java Web Token authorization settings
jwt.header=Authorization
jwt.secret=secret
jwt.expiration=604800
jwt.route.authentication.path=auth
jwt.route.authentication.refresh=refresh

# Default admin user
create.default.admin=true
admin.email=admin
admin.password=admin@server.org

# Default guest user
create.default.guest=true
guest.email=guest@server.org
guest.password=guest

# cookies settings
trust.builder.cookies=JSESSIONID,Authorization
```

2.3 Enabler Installation

The installation procedure consists in copying the war file into the Tomcat *webapps* root folder and restarting the Tomcat server. The Tomcat manager app can also be used for deploying the war file and check if the installation was successful. The software was installed on Ubuntu 14.04 Operating System, the installation steps is represented by a sequence of "apt-get" commands.

```
> apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv EA312927
> echo "deb http://repo.mongodb.org/apt/ubuntu "$(lsb_release -sc)"/mongodb-org/3.2
multiverse" | sudo tee /etc/apt/sources.list.d/mongodb.list
> apt-get update
> apt-get install mongodb-org
> service mongod start
> add-apt-repository ppa:webupd8team/java
> apt-get update
> apt-get install oracle-java8-installer
> apt-get install tomcat7
> service tomcat7 start
> mv system-modeller.war /var/lib/tomcat7/webapps
> service tomcat7 restart
```

2.4 Troubleshooting

The installation procedure declares explicitly the dependencies of the software as standard packages and libraries. The application is composed of a standard MongoDB installation and a web application running on a Tomcat server. Any errors are captured in the standard log files (e.g. /var/log/tomcat7/catalina.out) after the application war file has been deployed.

3 User and Programmer Guide

3.1 User Guide

In this section we describe the functionality of the Trust Builder. For a better understanding we provide definitions of the main concepts used in this document. An Asset is an element of the network that can be the following: Stakeholder, Logical asset or Physical asset. Stakeholder is a person or organisation, i.e. an entity that can carry out actions. Logical asset is a process, usually represented by a software. A physical asset is an element of the infrastructure or environment. For example it can be a Host, a Network or an Interface (a point of control). A Misbehaviour represents different ways in which assets may be compromised as a consequence of an active threat. An Involved asset is an asset whose presence is necessary for a threat to occur or to be managed (where a control can be located). A Control Strategy is a set of controls located at different assets that block or mitigate a threat.

The following sections provide details of the system's functionality covering:

- User management
- Model management
- Model editing
    a. Stage 1: defining assets and relationships which provide the initial model of a network
    b. Stage 2: validation and auto-generation of threats
    c. Stage 3: defining threat management strategy (selecting controls for assets or control strategies for threats)
- Model outputs

Trust modelling in essence is a three stage process, with the stages used repeatedly in an iterative fashion. In the first stage the user constructs the trust model by putting assets into the modelling panel and establishing links between the assets. An "asserted asset" is one defined by the user in this way during Stage 1. There are Physical, Logical, Stakeholder and Connection assets that can be used for constructing the Trust model. An "inferred asset" is one generated automatically in the model validation at Stage 2, whose presence can be inferred from the asserted assets and their relationships.

The validation process in stage 2 automatically generates inferred assets and also threats and possible security controls to counter act threats. The validation process also determines whether the information provided about assets and their relationships is consistent and complete. If this is not the case, the validation still runs but the model is marked as 'invalid'. In this case the uses should go back to Stage 1 and updates the model so that it contains sufficient information for a successful validation. Once the validation step is successful, the model is 'valid', but it becomes invalid again if any Stage 1 changes are made to the assets or their relationships.

In Stage 3 the user addresses threats by selecting or modifying the set of security controls that should protect the assets in the system. The aim is to eliminate threats by addressing them, or at least reduce the list of threats to an acceptable level.

3.1.1 User Management

3.1.1.1 Main page

On the main page of Trust Builder (Figure 1) there are several links in the upper right hand corner of this page, these are: *Home*, *View Models* and a dropdown menu under the *Person* icon. The *Home* link takes the user to the main page of "*Trust Builder*". The *View Models* presents a list of models previously defined by the user. The dropdown menu under the person icon provides the following functionality:

a) Sign in
b) Register
c) Forgot Password
d) Manage Account
e) Sign out

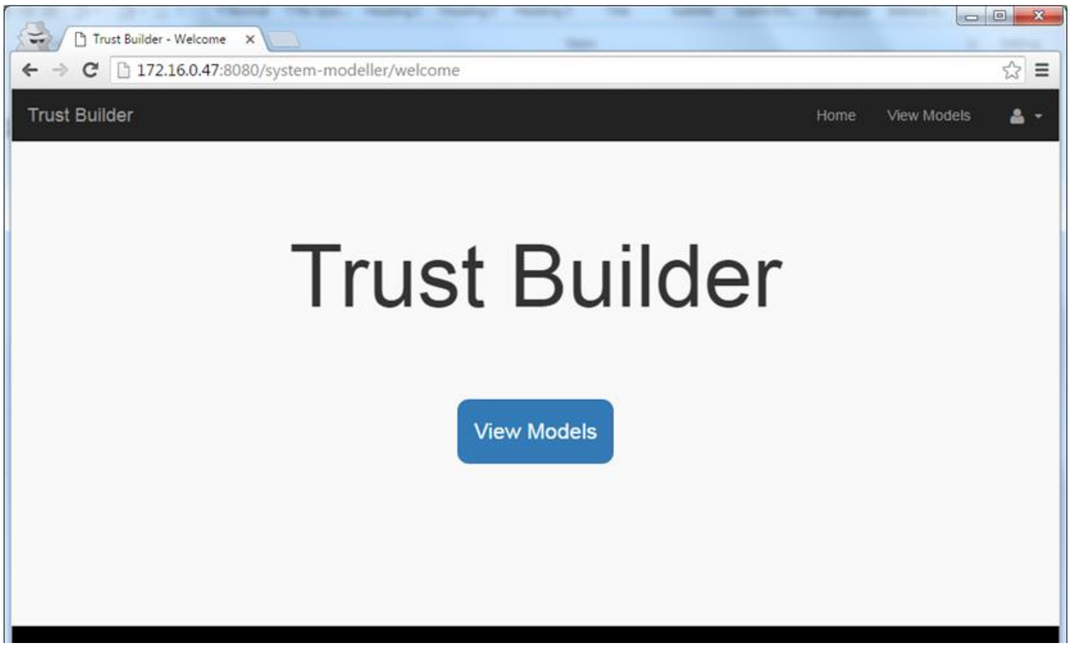


Figure 1- Trust Builder main page

3.1.1.2 User login

The login page of Trust Builder is activated by clicking the *Sign In* link in the dropdown menu under the person icon. If there is no login and password the user needs to register for the service by clicking on a link *Sign up here* (see Figure 2).

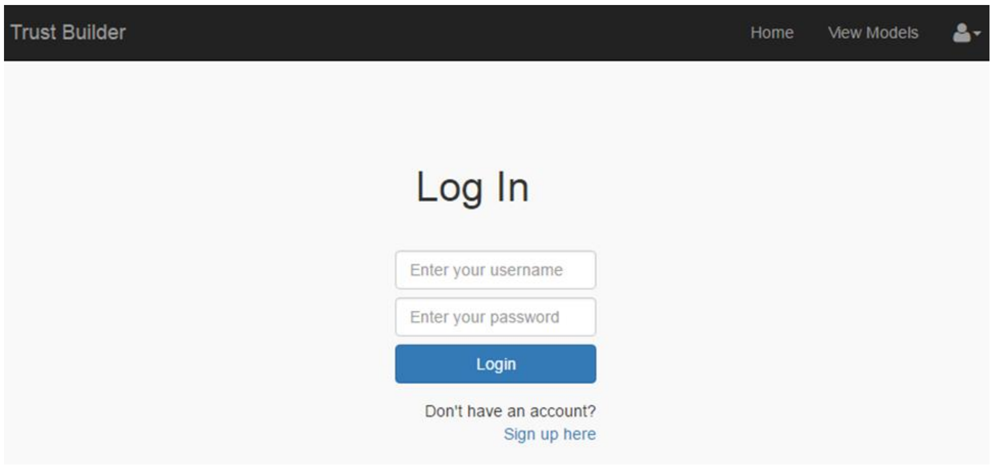


Figure 2 – User login page

3.1.1.3 User registration

The user can register by providing email address and password (see Figure 3). On registration the user's account is still inactive. The system administrator needs to activate the account before the user is able to login. The user is notified via email about activating the account.

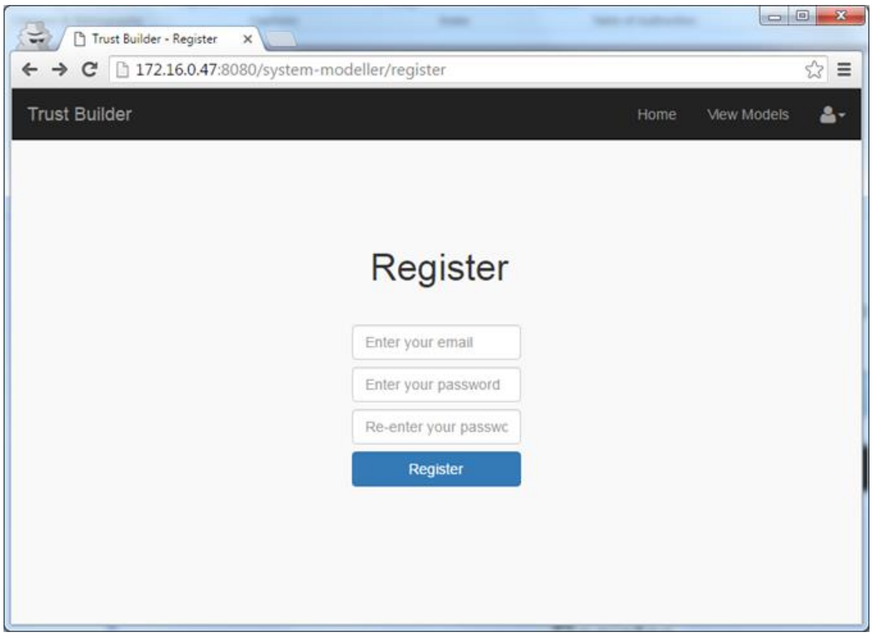


Figure 3- User registration

3.1.1.4 Password management

The user can reset the password by clicking on the "Forgot Password" link in the dropdown menu (see Figure 4).

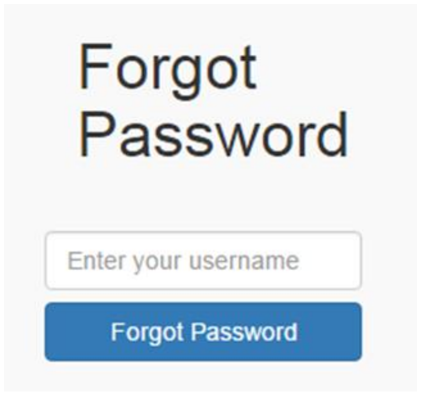


Figure 4 – Resetting the password

After typing in the user id and clinking on the "Forgot Password" button the user is presented with the "Reset Password" page (see Figure 5).

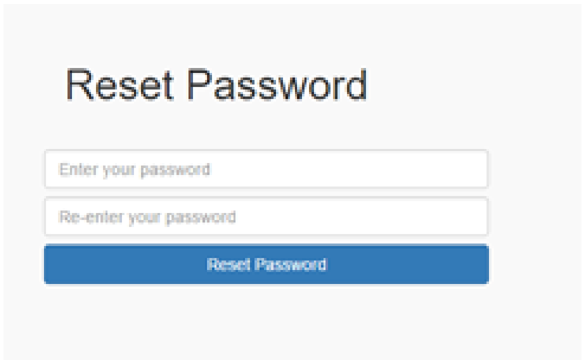


Figure 5 - Password reset

3.1.1.5 Logout

Logout is activated by clicking on the *Sign Out* link in the dropdown menu under the person icon.

3.1.2 Model management

The term Model management incorporates several functions such as:

a) Listing models created by the user or shared by others
b) Creating models
c) Importing/Exporting Models
d) Managing access to models
e) Deleting models
f) Checking in/out models

In the first release of the Trust Builder only a subset of the above functions has been implemented.

3.1.2.1 List models

By clicking on the "*View Models*" button on the main page the user can list the models that the user either owns or has read/write access to (see Figure 6).

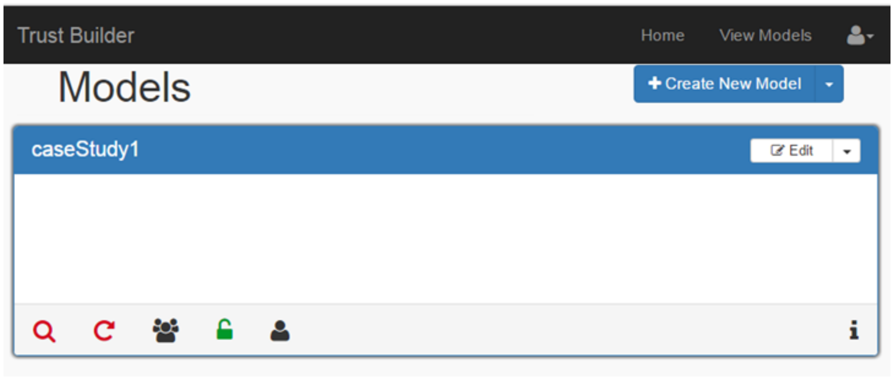


Figure 6- List models

In Figure 6 we can see one model called "caseStudy1". At the bottom left corner of the model window there are five icons. These are indicators reflecting the status of the model. The colour of these indicators reflects whether the corresponding action has taken place or not. For example if the "magnifying glass" indicator is read it means that for the given model no report was generated. These indicators are described in Table 2.

Icon	Description
🔍	indicates if a report was generated (green colour) on not (red colour)
↻	indicates if the model was validated (green colour) on not (red colour)
👥	indicates if the model was shared (not yet implemented)
🔓	indicates if the model is not locked (green) or locked (red)
👤	last modification of the model
ℹ	when the model was created

Table 2 - Model indicators

The dropdown menu in the top right corner of the model window offers several functionalities, these are: Edit, Share, Rename, Copy and Lock (see Figure 7).

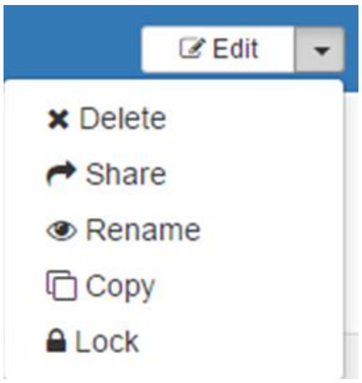
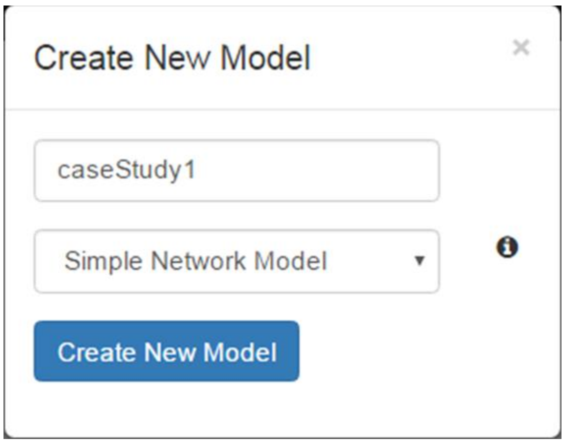


Figure 7- Model dropdown menu

3.1.2.2 *Create model*

By clicking on the "*Create New Model*" the user can create an empty model (see Figure 8). The drop-down selection allows the user to choose which generic model (version) to use to construct the model. The user automatically becomes an owner of the newly created model with read/write access and with the ability to of granting/revoking access rights for other users. The new model is added to the model list and it can be edited.


Create New Model

caseStudy1

Simple Network Model

Create New Model

Figure 8 - Creating a new model

The “*Import Existing Model*” and “*Create Model from Template*” options are not implemented for the first release.

3.1.2.3 Delete model

The delete action removes the model from the list along with the contained resources (e.g. assets, relationships) and all information about the access rights to the model for other users. The precondition for the delete operation is that the model should not be checked out by another user.

3.1.3 Asset definition

Clicking the *Edit* button opens the editing panel that consists of three parts. On the left side there is the “*Select Assets*” panel. In the middle there is the Model Construction Canvas. The right side panel contains various categories related to an item selected on the canvas, such as *Incoming/Outgoing Relations*, *Control Sets*, *Roles*, *Inferred Relations and Threats* (see Figure 9).

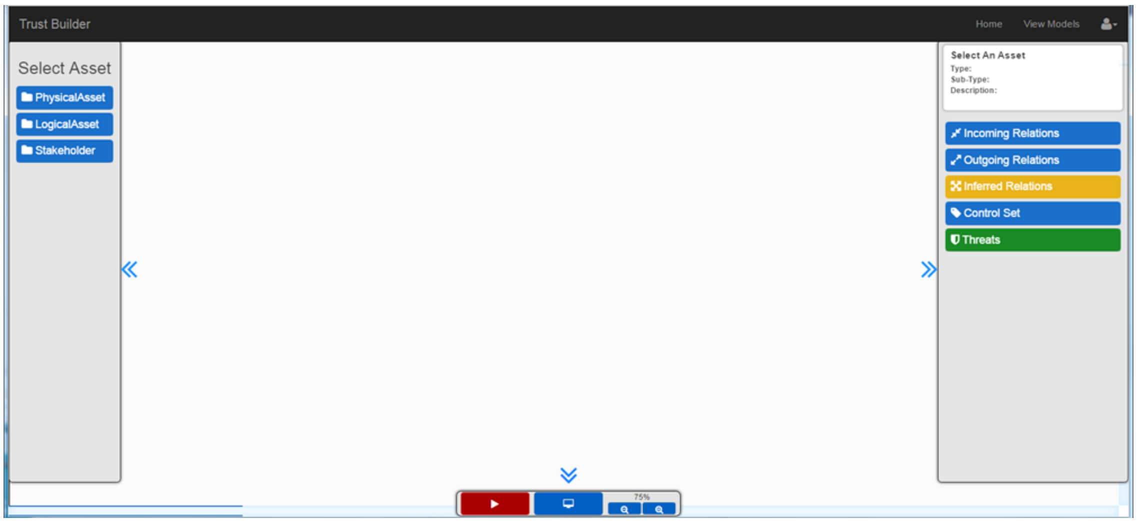


Figure 9 - Model editing

The main panel also contains four buttons, these are in Table 3.

Icon	Description
▶	Process (validate) the model
🖥	Upload a screenshot of the model
100% 🔍 🔍	Zoom controls

Table 3 – Model editing controls

3.1.3.1 Select and add asserted asset

The left panel of the model editor contains various assets, these fall into three categories (see Figure 10):

a) Physical assets
b) Logical assets
c) Stakeholders

The choices available depend on the generic model (version) chosen to construct the model. Here we give examples from the built-in Simple Network Model.

The Simple Network Model includes the following entities: Cellular Network, Cellular Router, Cloud Host, Host, Server, Smart Phone, Wifi Router, Wired Router, Wireless Network, Work Station.

The Logical Assets in the Simple Network Model are: DB Server, File Server, Mail Agent, Mail Store, Remote Login, Web Server.

In this model there are two entities in the Stakeholder category, these are Human and Organisation.

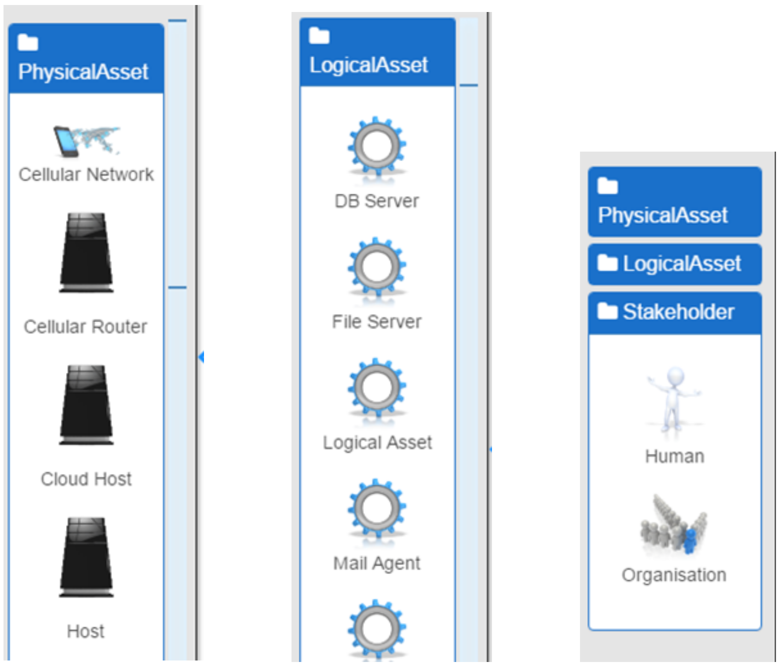


Figure 10 – Selecting assets

For inserting an asset in the model the user selects an asset by clicking on the asset's icon in left panel and dragging the asset into the model canvas (see Figure 11). By clicking on the asset we can view/edit the properties, these are:

a) name of the asset
b) incoming relations
c) outgoing relations
d) inferred relations
e) control set
f) threats (once the model has been processed)

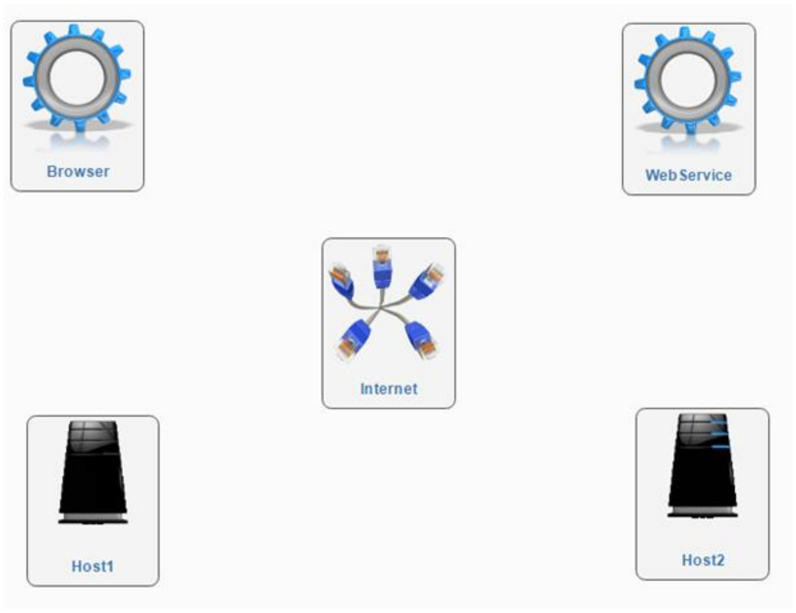


Figure 11 - Inserting assets into model canvas

3.1.3.2 Add relationship between assets

Once the assets have been put on the modelling canvas the user can connect two assets by establishing a relationship between them. By clicking on the asset a green cross appears in the left corner, this indicates that the asset can be connected to other assets (Figure 12).

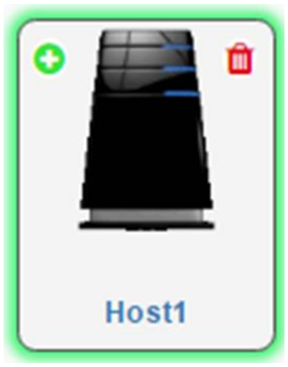


Figure 12 - Connecting assets

By clicking on the green + sign of the asset (in our example Host1) appropriate assets on the canvas are indicated with a blue tick, showing that a connection can be made between the assets (see Figure 13).

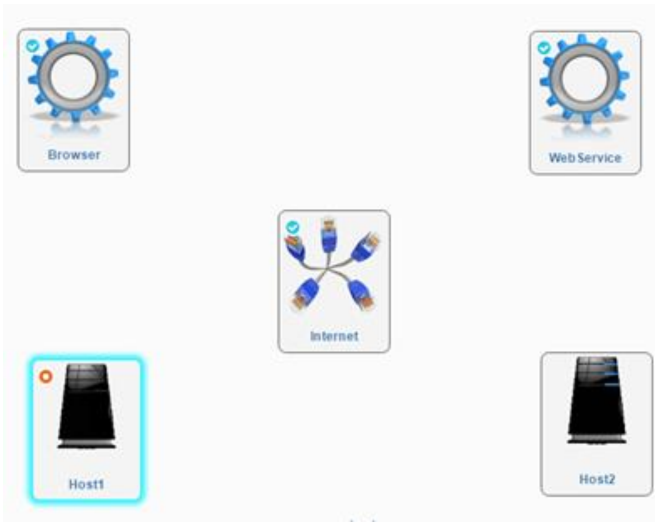


Figure 13 – Target assets for making connections

By clicking on the blue tick icons we can establish connections between assets (Figure 14).

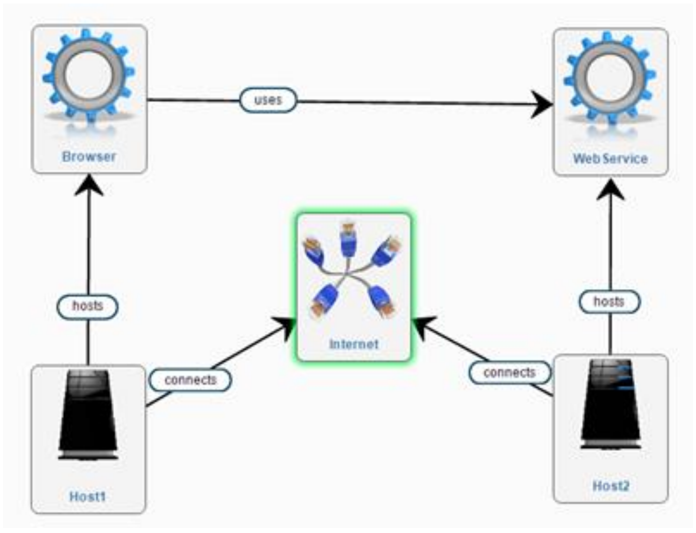


Figure 14 – Connecting assets

3.1.3.3 *Remove asset*

Assets can be removed by clicking on the red trash icon of the asset in the top right corner (see Figure 12). The delete operation removes all relationships between the selected asset class and other assets.

3.1.3.4 *Remove relationship*

By right clicking on the connection between two assets an option with a delete button comes up (see Figure 15). The delete applies only on the relationship the assets remain on the canvas.

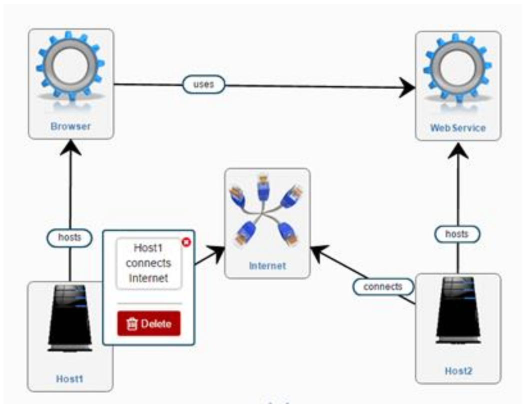


Figure 15 - Remove relationship between assets

3.1.3.5 Rename asset

The user can rename an existing asset by editing the asset's name under the corresponding icon. NB by changing the name the asset's connections will stay unaffected. All asset names must be unique.

3.1.4 Validation

Once the model is constructed it can be processed. This operation is activated by clicking on the red "play" button (see Table 3). The validation operation activates semantic reasoning that generates inferred assets that are added to the model and also a List of Threats that can be associated with the given model. This operation guarantees that the inferred assets are consistent with the asserted assets and relationships. On completion of the validation operation the updated model is presented to the user. Figure 16 illustrates the model after the validation operation.

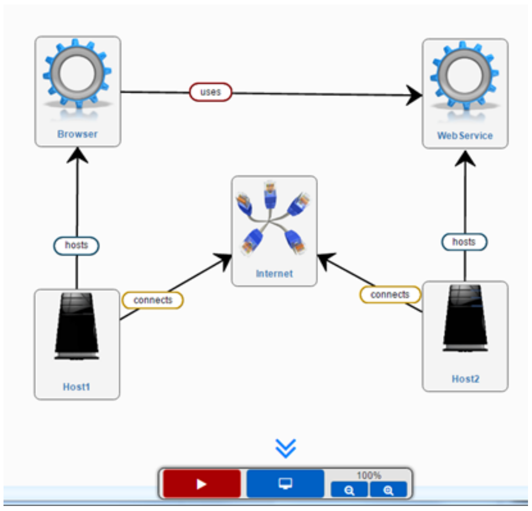


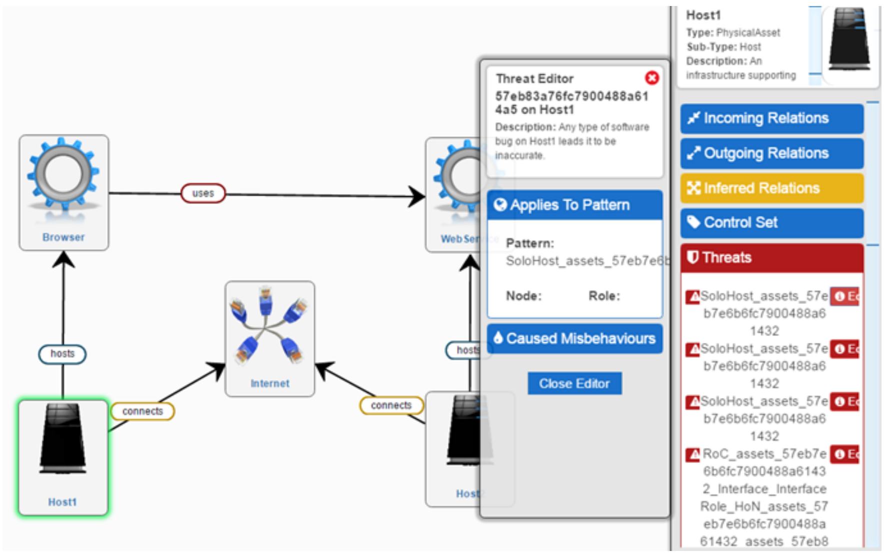
Figure 16 – Model after validation

The validation in essence generates entities that were missing from the initial model. Then the user needs to configure the new entities (e.g. inferred assets and inferred relationships). In our case there are some issues with the "*uses*" connection between the *Browser* and *Web Service*. The "play" button being red at the bottom in Figure 16 indicates that the model is not yet correct and needs to be updated and validated again. When the validation succeeds, the play button will turn green.

3.1.5 Threat management

3.1.5.1 Selecting a threat

The user can view the threats associated with the given asset. First the user needs to select an asset then click on the "*Threats*" button on the right panel (see Figure 17). This panel also provides options for editing the incoming/outgoing connections, control strategies to address a specific threat and control sets that make up the control strategy. Clicking the Edit button brings up the *Threat Editor* that allows to configure various parameters for the give threat.



Host1
Type: PhysicalAsset
Sub-Type: Host
Description: An infrastructure supporting

Threat Editor
57eb83a76fc7900488a614a5 on Host1
Description: Any type of software bug on Host1 leads it to be inaccurate.

Pattern: SoloHost_assets_57eb7e6b

Node: Role:

SoloHost_assets_57eb7e6b6fc7900488a61432
SoloHost_assets_57eb7e6b6fc7900488a61432
SoloHost_assets_57eb7e6b6fc7900488a61432
RoC_assets_57eb7e6b6fc7900488a61432_Interface_InterfaceRole_HoN_assets_57eb7e6b6fc7900488a61432_assets_57eb8

Figure 17 – List of threats associated with Host1

3.1.5.2 Set/unset control set

The user can select a control set within the asset properties panel marking it for implementation or removing the marker if already present. The control sets that are marked determine which control strategies will be in place, and hence the status of threats. The example on Figure 18 illustrates the Control Set for Host1.

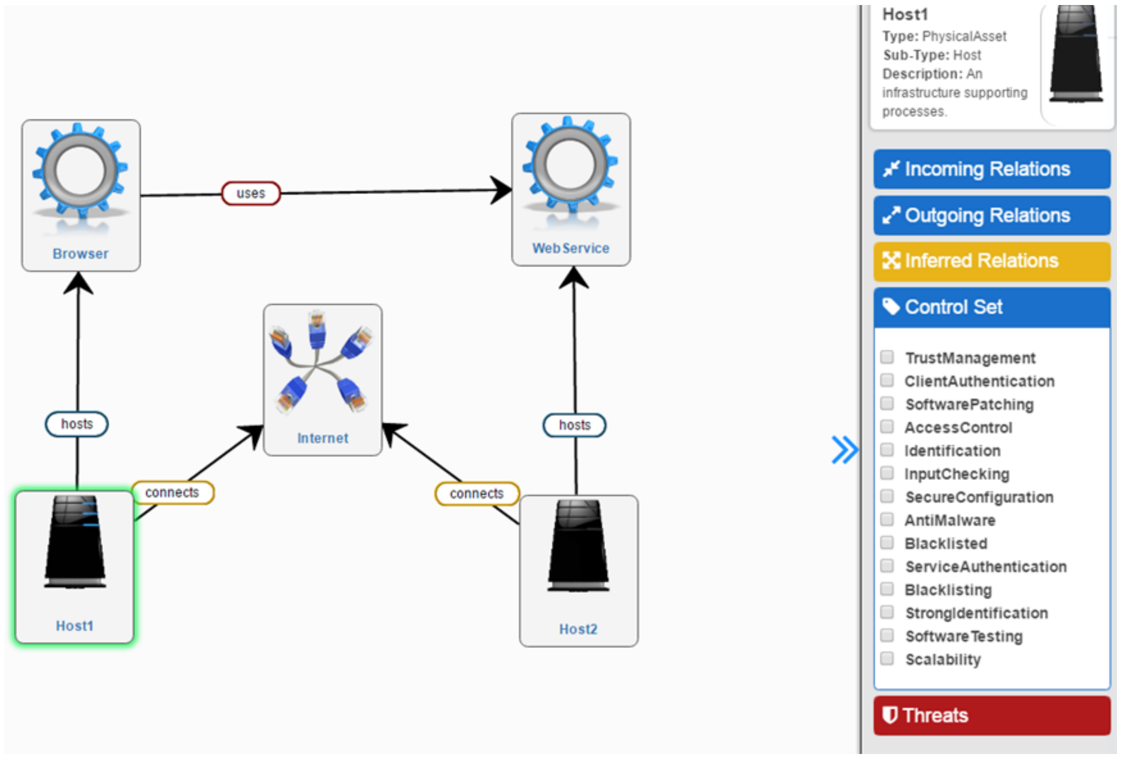


Figure 18 - Selecting Control Set properties for Host1

3.1.6 Model outputs

3.1.6.1 Generate report

The *Generate Report* function is activated by clicking on the "blue report" button (see Table 3).The report contains information about the current system setup. This is feature is not implemented in the current release.

3.1.6.2 Obtain network diagram

Allows retrieval of the network diagram as it would appear in the editor canvas. This feature is not implemented in the current release.

3.2 Programmer Guide

N.B. The presented *Trust Builder* is not programmable: there is no API that can drive the system programmatically.

4 Unit Tests

4.1 Information about Tests

In this section we describe a case study represent simple scenario demonstrating the use of Trust Builder tool. The main tasks that we need to complete for each case study are:

a) Constructing the model
b) Validating the model
c) Addressing the threats
d) Interpreting the outcome of threat elimination

The interpretation of the output is beyond the scope of this test.

4.2 Case Study

4.2.1 Constructing a model

In this section we analyse a typical case for ICT systems representing the threats associated with accessing a web page hosted on a remote server. Constructing a security model involves placing assets on the canvas and establishing connections between them. First step we open a new model by clicking on the “Create New Model” link (see Figure 6). Then we drag the assets in the modelling panel and make connections (see Figure 14). The model itself consist of two hosts connected to the internet. The user on Host1 accesses a Web Service deployed on Host2. This is a simple model, in essence downloading a web page from a remote web server. The reasons for selecting this model are as follows:

a) Frequently occurring case, typical for all web applications
b) Simplicity
c) All types of inference can be well demonstrated, these are:
    - Inferred assets
    - Inferred relationships
    - Mandatory relationships
d) The threats inherent to the model are well understood
e) The effect of controls and threats can be easily interpreted

The model is shown below in Figure 19. The model consists of

- two physical asset “Host” assets, re-labelled to be “Host1” and “Host2”;
- the “Wired Network” physical asset, labelled “Internet”;
- the generic “Logical Asset” logical asset, labelled “Browser”;
- the “Web Server” logical asset, labelled “Web Service”.

The assets are linked together as shown.

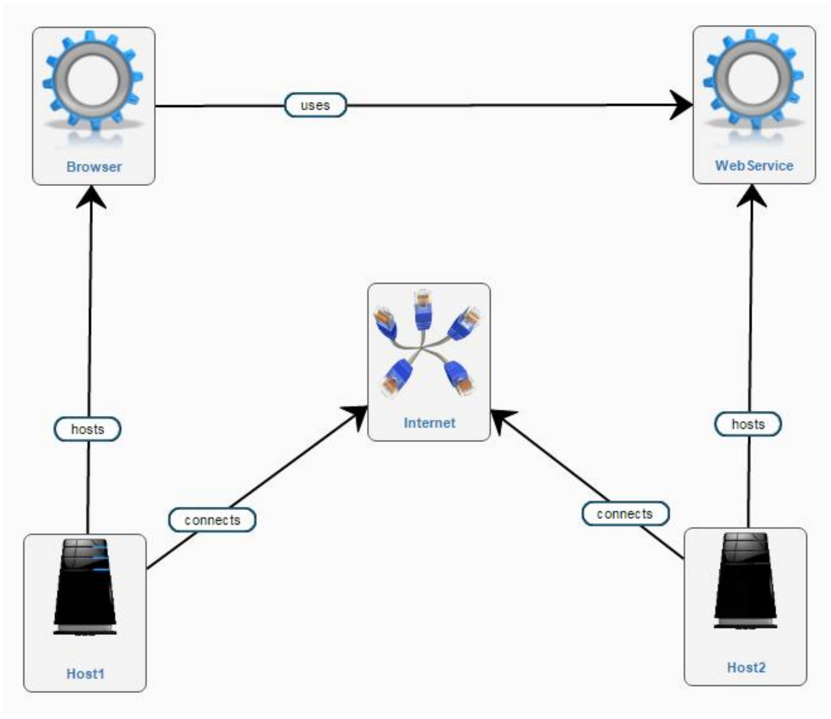


Figure 19 – Test model

4.2.2 Validating the model

The validation operation checks the model and generates inferred assets and relationships if they were missing from the initial model. For example if there are two assets connected and the user had missed out an asset between them, this inferred asset will be automatically added by the validator.

The red “play” button must be pressed. The validation operation can take some seconds. The outcome of the validation operation is indicated by changing the colour of the boundary around the assets. A green colour indicates the validation has succeeded, otherwise the colour is red. All issues identified by the validation operation need to be addressed and the model validated again.

In this case we need to pay attention to the connection between the Browser and Web service (the “uses” label is framed by red line). Clicking on the “uses” relation reveals an inferred “Service Pool” asset which must have its relations to the asserted assets (Browser and Web Service) defined. This is indicated by the red background of Incoming Relations in Figure 20.

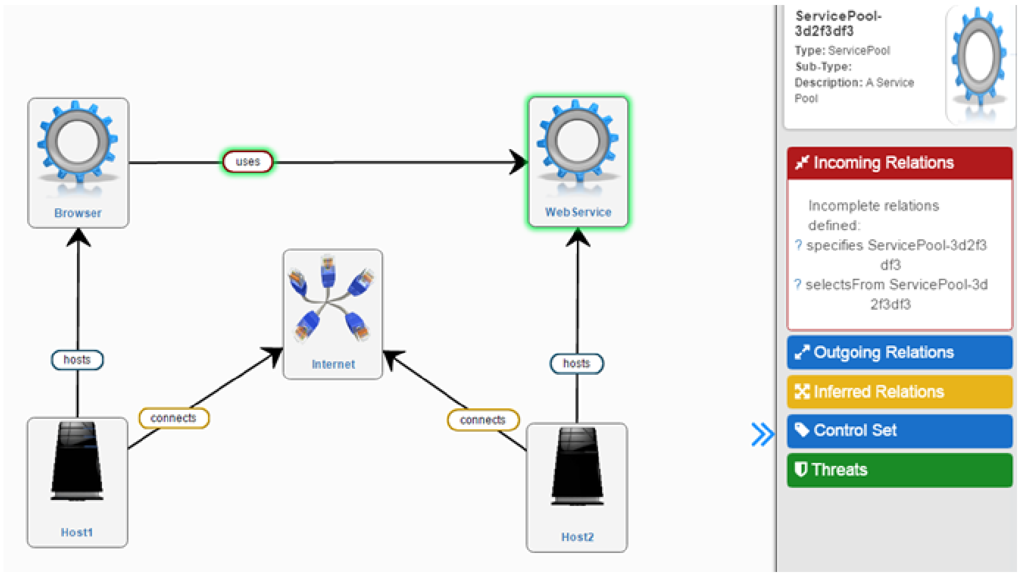


Figure 20 - Incoming relations must be defined on the inferred asset

The blue question marks indicate missing information. By clicking on the first blue question mark we can specify that it is the Browser which "specifies" the Service pool (inferred asset), see Figure 21.

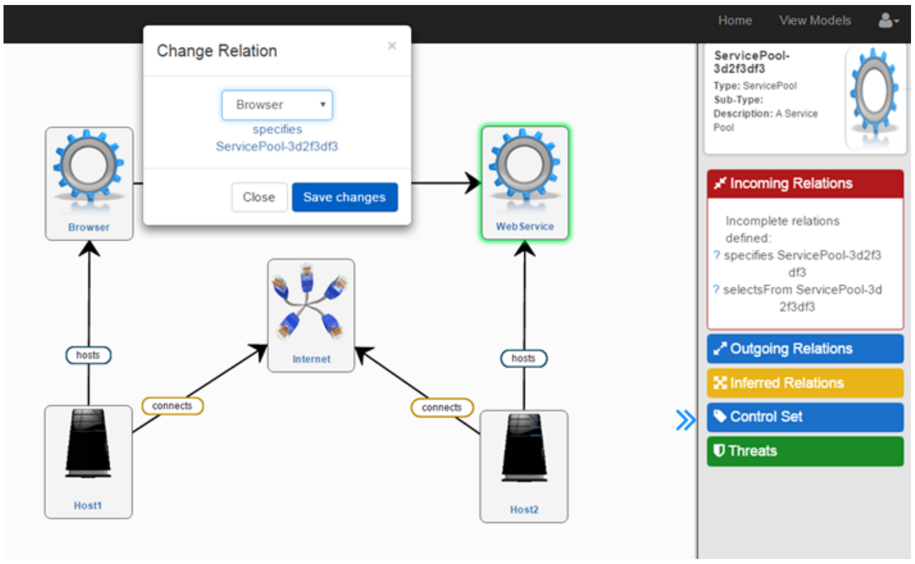
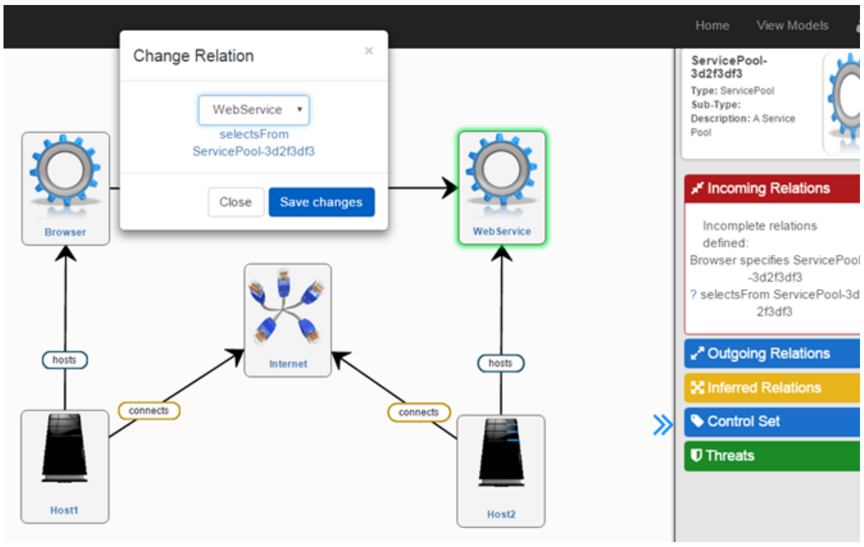


Figure 21 – Establish connection between Browser and Service Pool

By clicking on the second blue questions marks we can specify a relationship between the Web Server and Service pool (inferred asset), see Figure 22.



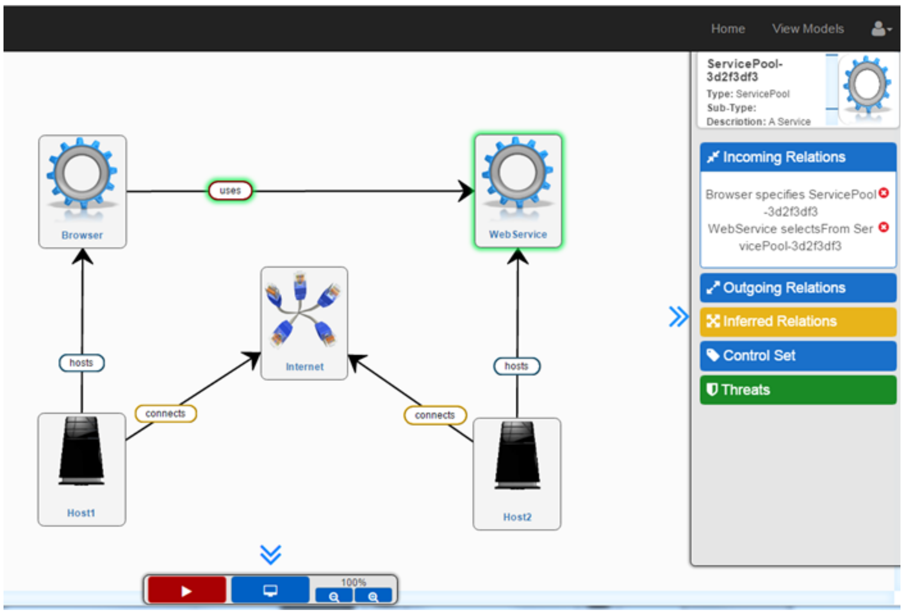
ServicePool-3d2f3df3
Type: ServicePool
Sub-Type:
Description: A Service Pool

selectsFrom ServicePool-3d2f3df3

Incomplete relations defined:
Browser specifies ServicePool-3d2f3df3
? selectsFrom ServicePool-3d2f3df3

Figure 22 – Establish connection between Web Service and Service Pool

After establishing connections to the Service Pool (inferred asset) the background colour of Incoming Relations should turn blue and the “uses” relation label should change from red to yellow to indicate that an inferred asset can be found at that relationship but that the inferred asset is fully specified. In the following step we validate the model again by clicking on the red play button at the bottom of the screen (see Figure 23).



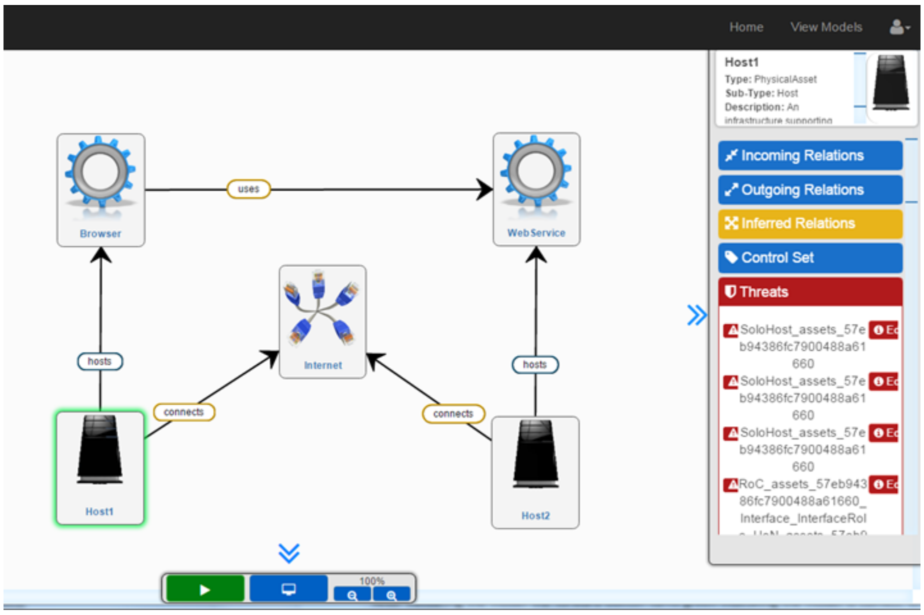
ServicePool-3d2f3df3
Type: ServicePool
Sub-Type:
Description: A Service

Browser specifies ServicePool-3d2f3df3
WebService selectsFrom ServicePool-3d2f3df3

Figure 23 - Setting all Incoming Relations

4.2.3 Addressing the threats

The validation step also generates a list of threats that need to be resolved. Resolving threats is achieved by specifying the controls to use. The effect of specifying a control is that the threat becomes either eliminated or its likelihood of occurring significantly reduced. After validating the model the forward button turns green indicating the model is correct. In the following step we need to resolve the threats for each asset. For example by clicking on Host1 and the Threats tab we can see a list of threats associated with Host1 (see Figure 24).

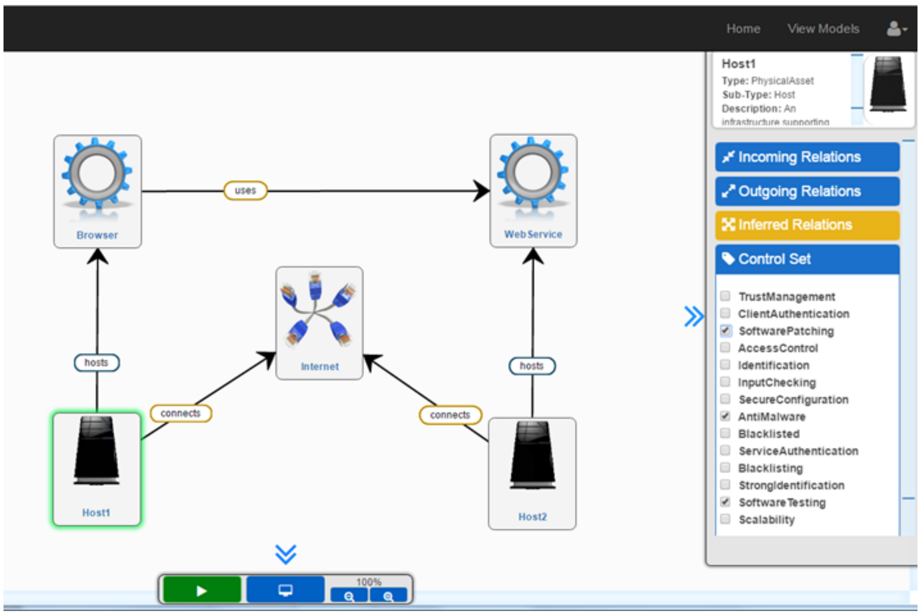


Host1
Type: PhysicalAsset
Sub-Type: Host
Description: An infrastructure supporting

SoloHost_assets_57e b94386fc7900488a61 660
SoloHost_assets_57e b94386fc7900488a61 660
SoloHost_assets_57e b94386fc7900488a61 660
RoC_assets_57eb943 86fc7900488a61660_ Interface_InterfaceRol

Figure 24 - Threats associated with Host1

By clicking on the red Edit button to the right of a threat, the user can see more details about the selected threat. The threats can be resolved by selecting options under the Control Set tab. A list of controls available for Host 1 is given in Figure 25.

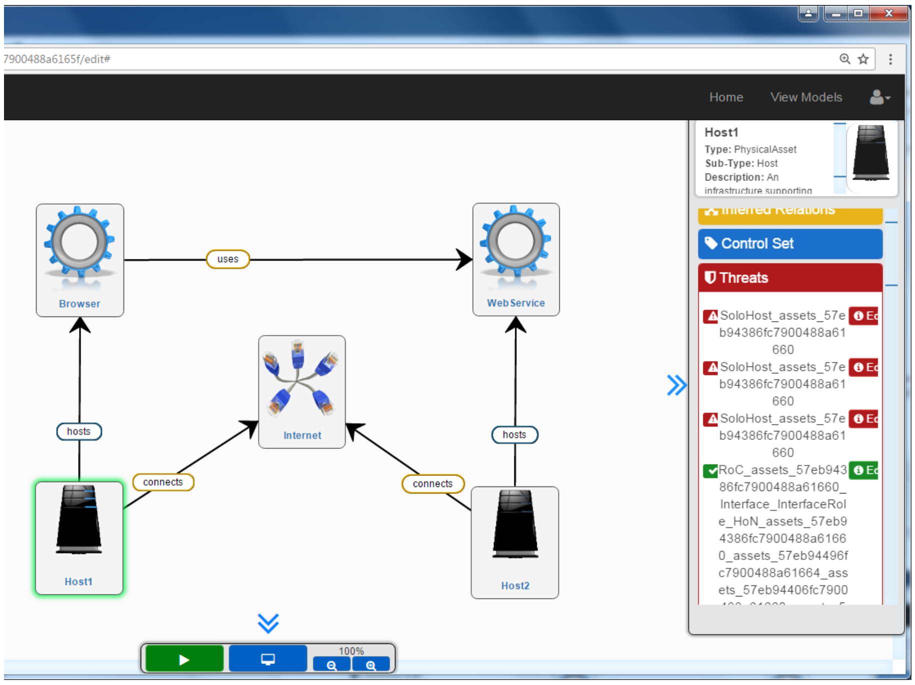


Host1
Type: PhysicalAsset
Sub-Type: Host
Description: An infrastructure supporting

- ☐ TrustManagement
- ☐ ClientAuthentication
- ☑ SoftwarePatching
- ☐ AccessControl
- ☐ Identification
- ☐ InputChecking
- ☐ SecureConfiguration
- ☑ AntiMalware
- ☐ Blacklisted
- ☐ ServiceAuthentication
- ☐ Blacklisting
- ☐ StrongIdentification
- ☑ SoftwareTesting
- ☐ Scalability

Figure 25 - Control set for Host1

For the purpose of demonstration we can select three options (Software Patching, Anti Malware and Software Testing) and see which threats will be resolved. These threats are indicated by green colour (see Figure 26).



Host1
Type: PhysicalAsset
Sub-Type: Host
Description: An infrastructure supporting

SoloHost_assets_57eb94386fc7900488a61660
SoloHost_assets_57eb94386fc7900488a61660
SoloHost_assets_57eb94386fc7900488a61660
RoC_assets_57eb94386fc7900488a61660_Interface_InterfaceRole_HoN_assets_57eb94386fc7900488a61660_assets_57eb94496fc7900488a61664_assets_57eb94406fc7900

Figure 26 - Resolving threats

The screenshot in Figure 26 shows only one threat resolved. By scrolling down the list, one can see that there are four threats resolved as a result of selecting Software Patching, Anti Malware and Software Testing from the Control set.

Resolving the threats is an iterative process, the user needs to go through the assets one by one, selecting the options from the Control Set and checking which threats have been eliminated. In this User Guide due to the limited space we have described the threat resolution steps for one asset (Host1) but these steps are applicable to all assets. By following these steps the user should be able to resolve or at least mitigate all threats associated with the given Trust Model.

5 Acknowledgements

The work described in this deliverable was sponsored by 5G-ENSURE Project (Grant Agreement number: 671562 — 5G-ENSURE — H2020-ICT-2014/H2020-ICT-2014-2).

6 Abbreviations

5G-IPPP	5G Infrastructure Public Private Partnership
DoS	Denial of Service

Deliverable D3.4
5G-PPP Security Enablers Documentation (v1.0)
Generic Collector Interface Enabler

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	30.09.2016	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	ORANGE: Jean-Philippe Warry, Ghada Arfaoui	

Contents

1 Introduction

In this chapter, we present the “Generic Collector interface” enabler documentation. This enabler mainly aims at collecting data from the 5G network elements / components. The “Generic Collector Interface” will provide to authorized parties / actors large amount of data namely: logs, events and incidents related to virtualization, identity management, communication protocols / layers / stacks and some specific privileges escalation. This enabler leverages also the implementation of efficient FastData [1] inside 5G Networks, in order to detect as soon as possible security and efficiency issues of the network.

At the writing time of this document, the Generic Collector interface is still under research. In the scope of R1, we plan to develop some features (namely the report exchange), of the open specifications described in D3.2 [2], as a proof of concept. Our proof of concept mainly includes three components: a monitoring server that centralizes the collected data of a given network, the monitoring client(s) that send(s) collected data to the monitoring server and the monitoring service that uses the collected data.

2 Installation and Administration Guide

This section describes how the enabler is installed, configured, and administrated.

2.1 System Requirements

In a nutshell, we recall a simplified functional overview of the “Generic Collector interface”. The Client engine sends XML reports to the server engine. Depending on the security policies, the server engine transmits these reports to the designated service provider.

As shown in Figure 1 , the “Generic Collector Interface” consists of three main software blocs namely the client engine software (cf. Open specification in D3.2), the server engine software (Open specification D3.2) and the service software. The first software, so called “monitoringClient.py”, should be installed on all the network components. This software should run with the root privileges in order to be able later to collect necessary information. The second software, so called “monitoringServer.py”, should run in an external server (i.e., the collect server). The third software, so called “monitoringService.py” should run on the service provider side.

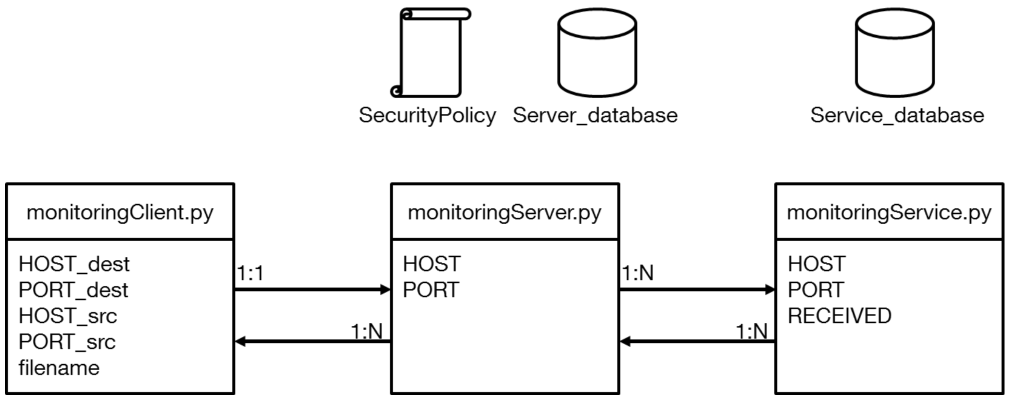


Figure 1: Generic Collector Interface diagram

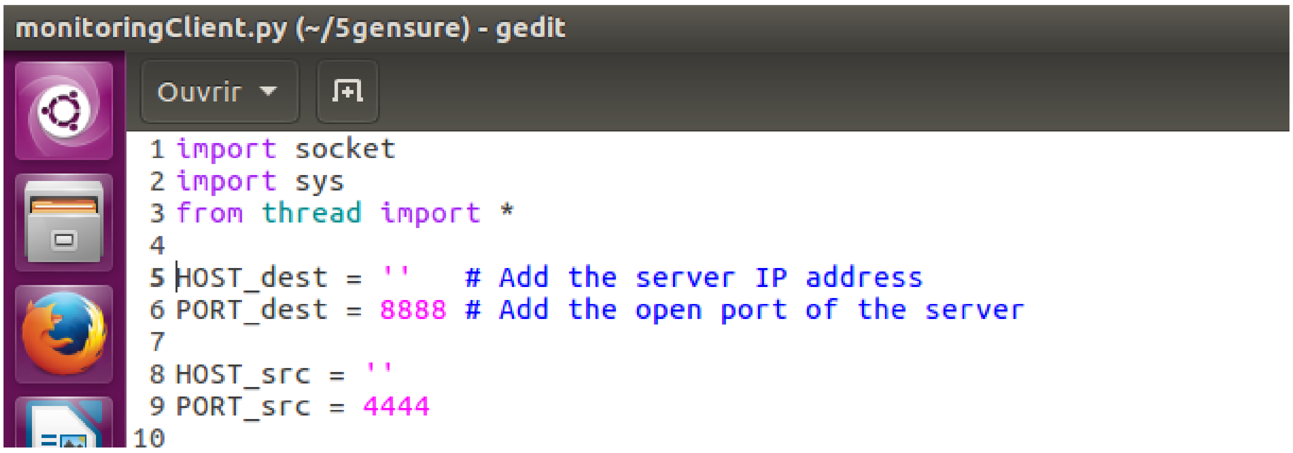
2.2 Enabler Configuration

The “Generic Collector Interface” has a so called “securityPolicy” file (cf. Figure 1) and two database folders, i.e., “Server_database” and “Service_database” (cf. Figure 1). The “securityPolicy” file should be accessible only in “read” mode for this enabler, namely “monitoringServer.py”. This file contains the network configuration, .i.e., “who can get which information”. For instance, this file can contain information like: data originated from the network components *X* must be transmitted only to the service *A.* This file is organized in four fields per line, as follows: <source IP address> <source port> <target IP address> <target port>. The “Server_database” folder should be available in “read” and “write” modes only for the server engine software. This folder will temporary contain the collected data. Once the data is correctly transmitted to the designated service, it is deleted from this folder. The “Service_database” folder should be available in “read” and “write” modes only for the service software.

The current main configurations of this enabler are

- The setup of the IP addresses and ports for the different running software (Figure 2, Figure 3, Figure 4).
- The setup of the storage folders and its access rights
- The setup of the security policy file and its access rights

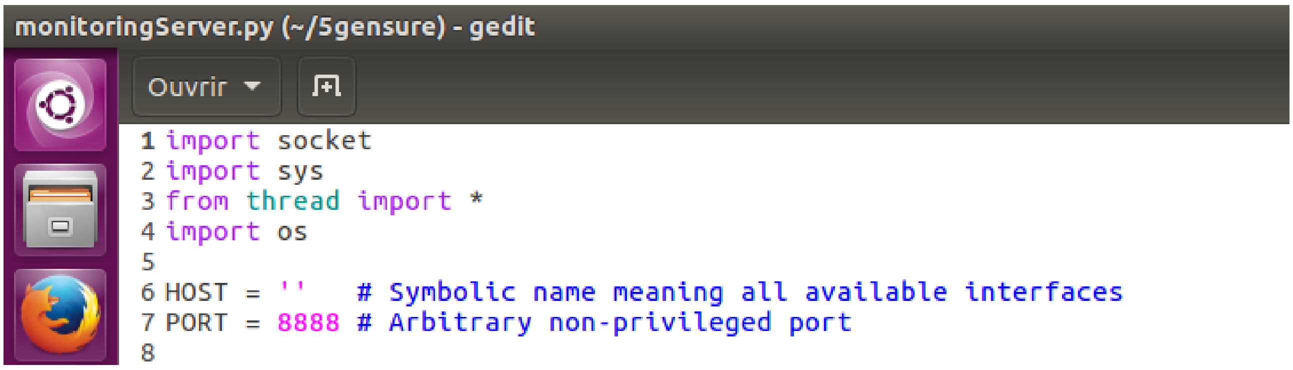
For “monitoringClient.py”, it is required to precise the available IP address and port of the host running “monitoringServer.py” (lines 5 and 6 in Figure 2). Lines 8 and 9 specify the IP address and port from with data will be sent. HOST_src = `''`means that ‘monitoringClient.py” will send data on the available interface. To force using a given IP address, replace `''` by the given IP address.



```
 1 import socket
 2 import sys
 3 from thread import *
 4
 5 HOST_dest = ''    # Add the server IP address
 6 PORT_dest = 8888 # Add the open port of the server
 7
 8 HOST_src = ''
 9 PORT_src = 4444
10
```

Figure 2: Configuration of "monitoringClient.py"

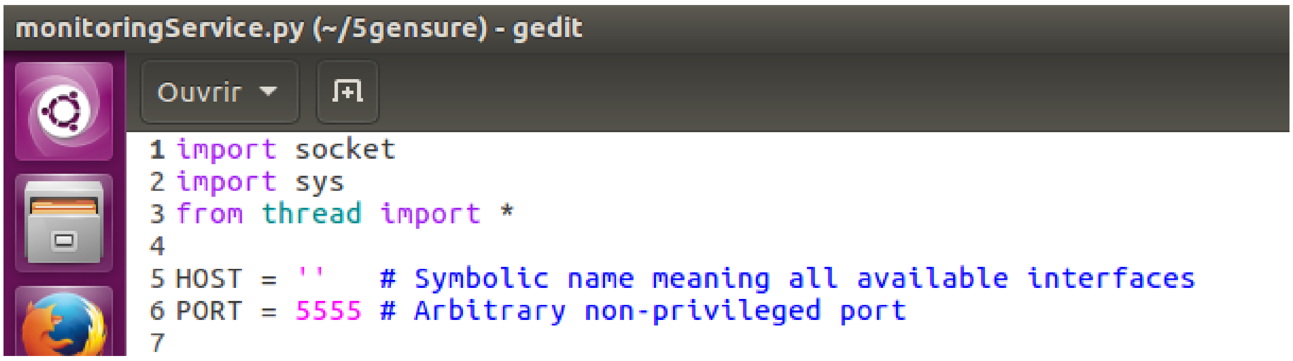
For “monitoringServer.py”, it also required to specify the IP address and port where to wait for inputs from the client engines (lines 6 and 7 in Figure 3). HOST = `''`means that “monitoringServer.py” will wait for input on all the available interfaces. As previously mentioned, to force “monitoringServer.py” to listen only to a given IP address, replace `''` by the given IP address.



```
1 import socket
2 import sys
3 from thread import *
4 import os
5
6 HOST = ''    # Symbolic name meaning all available interfaces
7 PORT = 8888 # Arbitrary non-privileged port
8
```

Figure 3: Configuration of "monitoringServer.py"

“monitoringService.py” requires almost the same configuration as “monitoringServer.py”: specify the IP address and port where to wait for inputs from the server engines (lines 5 and 6 in Figure 4). HOST = `''`means that “monitoringService.py” will wait for input on all the available interfaces. As previously mentioned, to force “monitoringService.py” to listen only to a given IP address, replace `''` by the given IP address.



```
1 import socket
2 import sys
3 from thread import *
4
5 HOST = ''    # Symbolic name meaning all available interfaces
6 PORT = 5555 # Arbitrary non-privileged port
7
```

Figure 4: Configuration of "monitoringService.py"

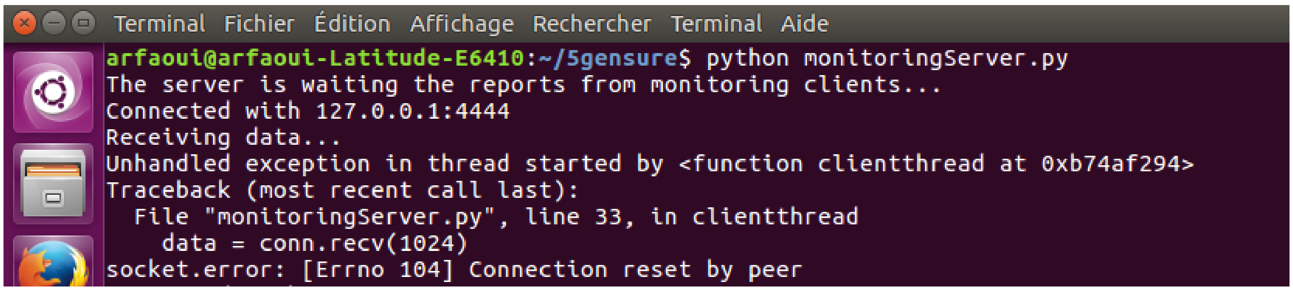
2.3 Enabler Installation

In this link [3], you can find compressed folder “GenericCollectorInterface.zip” containing three python scripts, i.e., “monitoringClient.py”, “monitoringServer.py” and “moitoringService.py”, an example of a security policy file “securityPolicy.txt”, and an example of an XML report “report.xml”.

The python script “monitoringClient.py” should be installed in all the network components that are supposed to collect and send reports to the Server engine. The python script “monitoringSrever.py” should be installed in a host that is accessible by the entire network component and that can access the host where installed the “moitoringService.py” script. “securityPolicy.txt” should be in the same host as the “monitoringSever.py”. “report.xml” can be duplicated in all the network components (This example can be used to facilitate the tests). Afterwards, “Server_database” should be created in the machine hosting “monitoringServer.py”. Similarly, “service_database” should be created in the machine hosting “monitoringService.py”.

2.4 Troubleshooting

The message error in Figure 5, can occur in the server engine side ("monitoringServer.py"). It is due to socket issues. This problem does not require any solution. It will be solved by the following report transmission process.



```
arfaoui@arfaoui-Latitude-E6410:~/5gensure$ python monitoringServer.py
The server is waiting the reports from monitoring clients...
Connected with 127.0.0.1:4444
Receiving data...
Unhandled exception in thread started by <function clientthread at 0xb74af294>
Traceback (most recent call last):
  File "monitoringServer.py", line 33, in clientthread
    data = conn.recv(1024)
socket.error: [Errno 104] Connection reset by peer
```

Figure 5: Socket error

3 User and Programmer Guide

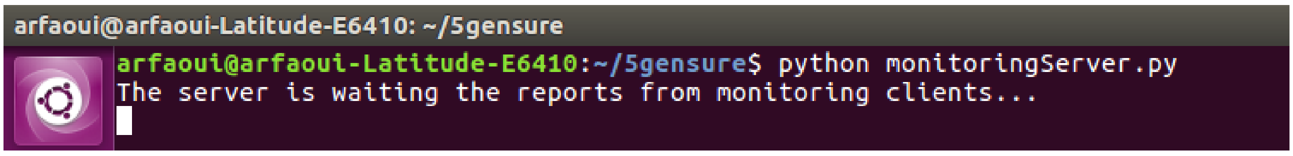
This section describes how to use and "program" the enabler.

3.1 User Guide

After installing and configuring the applications

1- Start the server engine (Figure 6):

python monitoringServer.py



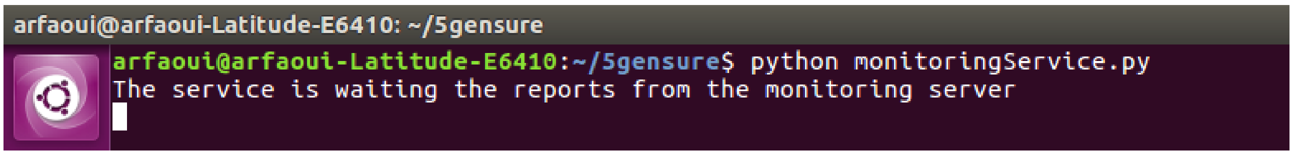
```
arfaoui@arfaoui-Latitude-E6410:~/5gensure$ python monitoringServer.py
The server is waiting the reports from monitoring clients...
```

Figure 6: Start the server engine

At this moment, the server engine is ready to get any report from any client and then transmit it to the corresponding Service.

2- Start the service (Figure 7):

python monitoringService.py



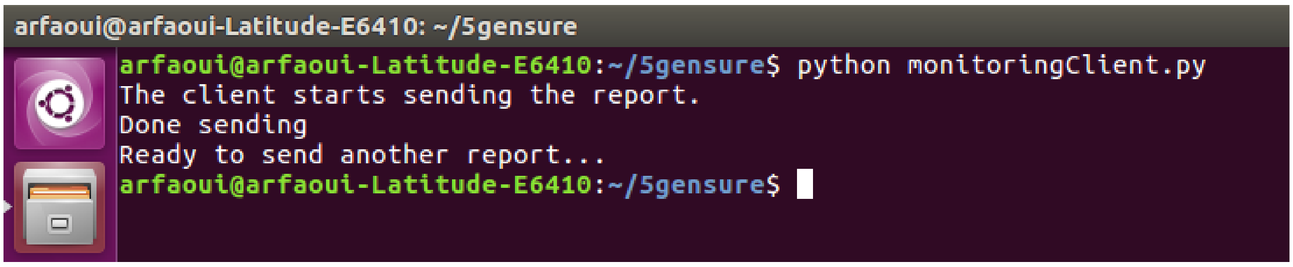
```
arfaoui@arfaoui-Latitude-E6410:~/5gensure$ python monitoringService.py
The service is waiting the reports from the monitoring server
```

Figure 7: Start the service

At this moment, the service is ready to receive any report from any client.

3- Start a report transmission by the client engine (Figure 8):

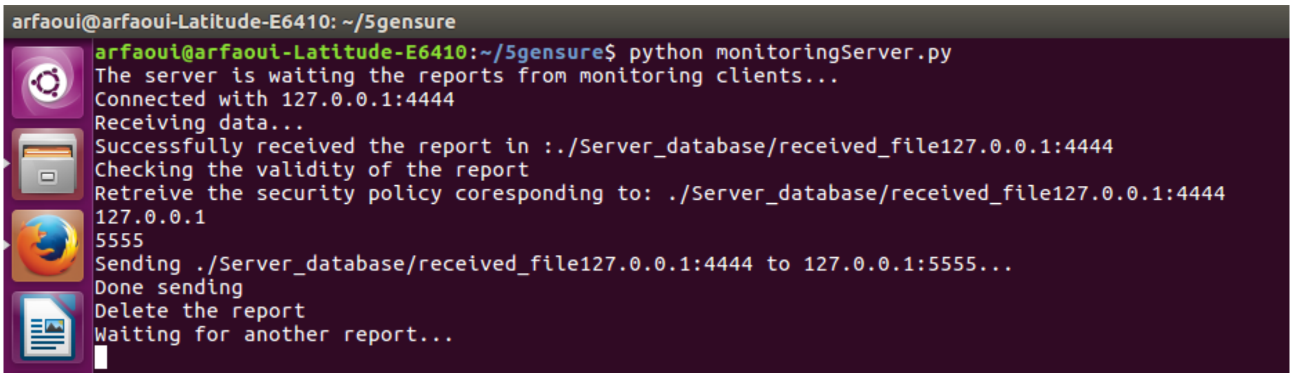
python monitoringClient.py



```
arfaoui@arfaoui-Latitude-E6410: ~/5gensure
arfaoui@arfaoui-Latitude-E6410:~/5gensure$ python monitoringClient.py
The client starts sending the report.
Done sending
Ready to send another report...
arfaoui@arfaoui-Latitude-E6410:~/5gensure$ 
```

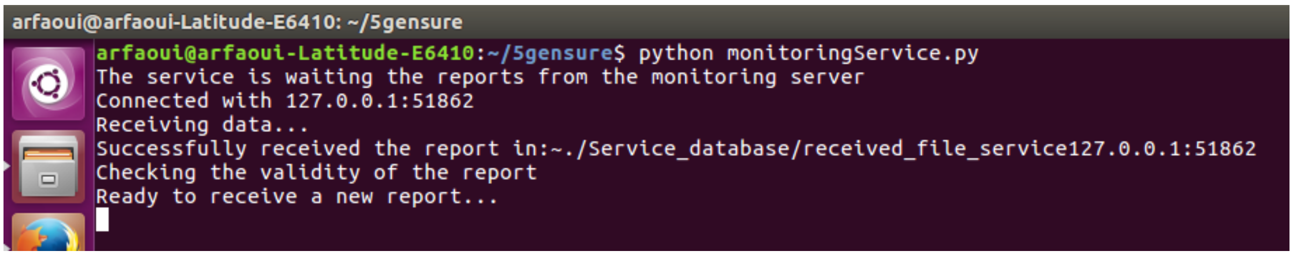
Figure 8: Start a report transmission

This command triggers the transmission of a report from the client engine to the server engine (Figure 9 and Figure 10).



```
arfaoui@arfaoui-Latitude-E6410: ~/5gensure
arfaoui@arfaoui-Latitude-E6410:~/5gensure$ python monitoringServer.py
The server is waiting the reports from monitoring clients...
Connected with 127.0.0.1:4444
Receiving data...
Successfully received the report in :./Server_database/received_file127.0.0.1:4444
Checking the validity of the report
Retreive the security policy coresponding to: ./Server_database/received_file127.0.0.1:4444
127.0.0.1
5555
Sending ./Server_database/received_file127.0.0.1:4444 to 127.0.0.1:5555...
Done sending
Delete the report
Waiting for another report...
```

Figure 9: Server engine when receiving a report



```
arfaoui@arfaoui-Latitude-E6410: ~/5gensure
arfaoui@arfaoui-Latitude-E6410:~/5gensure$ python monitoringService.py
The service is waiting the reports from the monitoring server
Connected with 127.0.0.1:51862
Receiving data...
Successfully received the report in:~./Service_database/received_file_service127.0.0.1:51862
Checking the validity of the report
Ready to receive a new report...
```

Figure 10: Service when receiving a report

3.2 Programmer Guide

This section is not applicable for this enabler.

4 Unit Tests

4.1 Unit Test 1

The objective of the first test is to separately check that the scripts “monitoringServer.py”, “monitoringClient.py” and “monitoringService.py” work properly. This means that the test will check that every script listens and sends on the designated IP address / port, and have the necessary access rights to the databases (i.e. “Server_database” and “service_database”) and to the security policy file. To perform this test, first, ping every script on the designated IP addresses / ports and monitor the exchanges by Wireshark. Then, check the access rights of the relevant databases and security policy file.

4.2 Unit Test 2

The goal of this test is to check that all the software components correctly communicate with each other, and hence, the XML report is correctly transmitted from a client engine to the corresponding service. To do so, after following the three steps described in section 3.1. Check manually that the file “report.xml” has been transmitted only to the authorized service. Using Wireshark to trace the various exchanges between the different scripts / machines can be helpful.

5 References

[1] D. C. Hansen, "Fast Data: Go Big. Go Fast.," *Data Integration - Inside Oracle's Data Integration community,* 2012.

[2] "Deliverable D3.2 5G-PPP security enablers open specifications (v1.0)," 2016.

[3] 5G-ENSURE, "Software Delivery information for R1," [Online]. Available: https://workspace.vtt.fi/sites/5g-ensure/SitePages/5G%20ENSURE%20-%20Software%20Delivery%20information%20for%20R1.aspx.

Deliverable D3.4
5G-PPP Security Enablers Documentation (v1.0)
Security Monitor for 5G Micro-Segments

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	30.09.2016	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	VTT: Jani Suomalainen	

Contents

1 Introduction

Security monitoring is needed to gain awareness of networks' security situation and to detect ongoing attacks. However, attacks will be difficult to detect from 5G networks which will be heterogeneous and will have massive amount of users and data flows. Micro-segmentation (or sub-slicing) can be used to increase the accuracy of monitoring. Monitoring can be focused to particular segment or slice (i.e. to particular applications and to restricted amount of users). Consequently, security monitor for micro-segments enables: 1) more accurate incident detection (by focusing on fewer data streams, we can study more parameters and correlations from homogeneous data flows), 2) customization of security monitoring based on 5G customers/end-users preferences, and 3) adaptation of 5G networks' (micro-segments') defenses based on monitored/inferred security awareness

The goal for the release 1 of the enabler is to provide a framework (software) enabling distributed security monitoring, security inferencing, and reactions to security incidents. The framework enables development of components that will detect selected ongoing attacks in micro-segments, in order to adapt 5G networks or segments' defenses and topology. Particularly, Release 1 of the enabler provides a Complex Event Processing (CEP) Framework, which is based on the state-of-the-art 'big data' technologies: Apache *Kafka* and Apache *Spark*. Release 1 of the enabler is a compilation of existing open source software. It does not introduce new software.[1]

The framework provides a *tool chain* that can be used when constructing different monitoring setups. It provides a mechanism to collect and share events from various sources and to distribute them to security inherence components. The framework increases ***scalability*** and ***flexibility*** of 5G security monitoring by:

- enabling new heterogeneous event sources (switches etc.) to be easily added
- reusable components to be used for processing of event streams (e.g. merging, aggregating)
- enabling different 'inherence components' - such as pattern detectors, machine learners, correlation analyzers… - to be integrated to the system when a need arises in different micro-segments (the solution provides efficiency as events are provided only to those components that are interested on them)
- deploying *'big data' technologies* for analytics.. S*tate-of-the-art* software components - that implement CEP, publish-and-subscribe and cluster computing paradigms - are utilized to handle large amounts of event streams in near real-time

The manual is organized as follows. In Section 2, the framework architecture and our software choises, and provide quick configuration and installation instructions are described. Section 3 describes how the enabler is used and illustrate some examples. Section 4 provides some basics tests. Abbreviations are listed in Section 5.

[1] The forthcoming (Release 2) features will include advanced inference capabilities – i.e. algorithms for detecting selected attacks against 5G / micro-segments – as well as response capabilities adapting Micro-Segment based on the inferred security situation awareness and calculated Trust Metrics.

2 Installation and Administration Guide

2.1 System Requirements

The main components, as well as technology selections for the enabler, are illustrated in Figure 1. The enabler architecture separates (network and infrastructure specific) security event sources from (application and resource specific) security processing and inferencing. The figure illustrates planned event publishers (blue in the figure), event consumers (i.e. inference components – green in the figure), and event distribution and processing framework (red in the figure). The figure also encompasses some event sources planned for future releases.

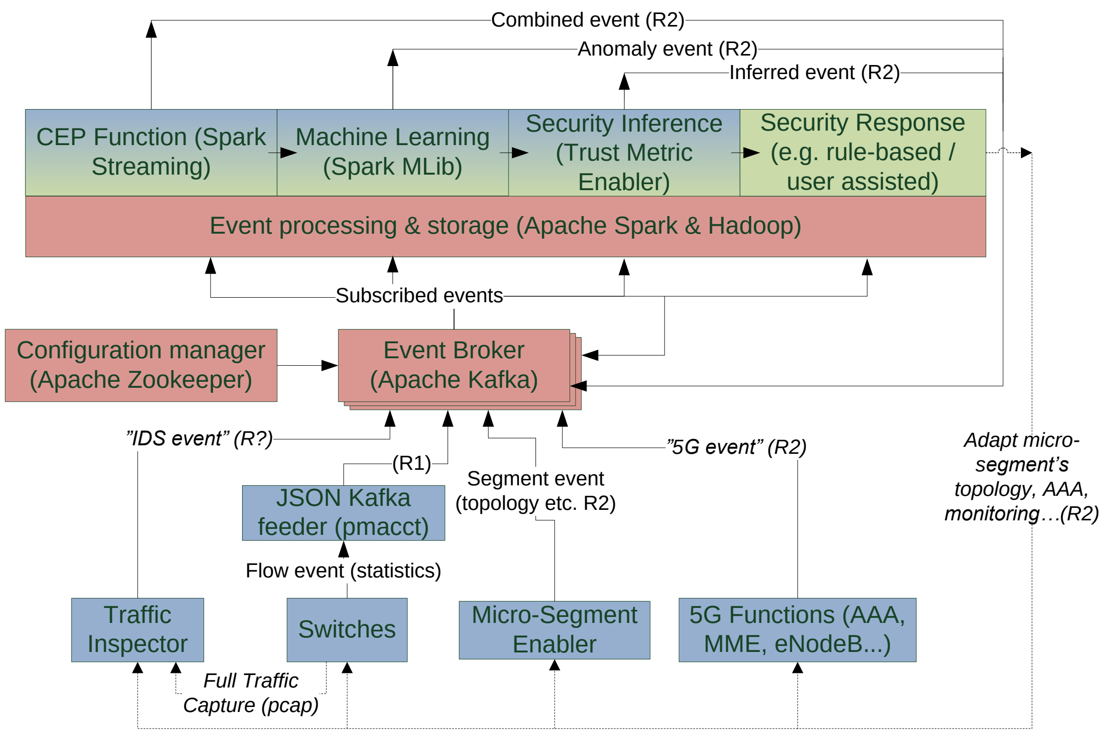

Combined event (R2)
Anomaly event (R2)
Inferred event (R2)
CEP Function (Spark Streaming)
Machine Learning (Spark MLib)
Security Inference (Trust Metric Enabler)
Security Response (e.g. rule-based / user assisted)
Event processing & storage (Apache Spark & Hadoop)
Subscribed events
Configuration manager (Apache Zookeeper)
Event Broker (Apache Kafka)
"IDS event" (R?)
(R1)
Segment event (topology etc. R2)
"5G event" (R2)
Adapt micro-segment's topology, AAA, monitoring…(R2)
JSON Kafka feeder (pmacct)
Flow event (statistics)
Traffic Inspector
Switches
Micro-Segment Enabler
5G Functions (AAA, MME, eNodeB...)
Full Traffic Capture (pcap)

Figure 1: Architecture and technology selections for security event distribution

Apache Kafka[2] [6] is used as an event broker to distribute information from event producers to event subscribers. New event publishers can be dynamically added to the framework. Similarly, new event subscribers can be added to the framework when need to measure security indicators or detect some specific attack symptoms emerge.

Kafka brokers may be clustered and chained. Event information from a broker in one micro-segment may be subscribed and delivered to other brokers in order to further distribute decision making (to enable further scalability) and e.g. to enable micro-segments to share information and detect cross-segment attacks.

[2] Apache Kafka is publish-and-subscribe system that was developed originally by LinkedIn. It is designed to be fast, scalable, and durable. Kafka brokers can be clustered to provide more resources elastically and transparently. A broker keeps messages on disk and replicates them within a cluster to prevent data losses. Each broker should be able to store terabytes of messages and handle megabytes of reads and writes per second from thousands of clients.

Kafka can utilize **ZooKeeper** [12] as a centralized service for maintaining configuration information. It is also possible to deploy Kafka as a stand-alone service that is connected directly by the event information producers and subscribers.

Apache Spark[3] [8] is used as a platform for event subscribers and processors, which provide (micro-segment specific) security inference such as anomaly detection and attack pattern detection. The subscribers may act also as publishers' and output inferred, anomaly, or combined events to other subscribers. Subscribers at the end of the chain (right top corner in Figure 1) are then expected to infer knowledge on micro-segment's security situation e.g. provide trust metrics or initiate some security responses. Event processing may be distributed. Different hosts in different domains may host interference components, each running on top of the CEP engine. Existing security inferencing components may be integrated to the system, by providing an adapter that transforms brokered events into the format understood by the component. Parsing of the published events is responsibility of event subscribers. However, event publishers should utilize open standards, such as JSON, when possible to ease transformation.

The framework is targeted for real-time or near-real-time processing of monitored data streams. However, in security inference, also history data is needed - e.g. short term history for correlation analysis, and longer history for machine learning. Smaller amounts of history data can be stored by the broker and for longer term history data the inference components must be integrated with a database for storing relevant events. Kafka and Spark integrate easily with Hadoop [11], which is the recommended database for the framework deployments.

Different *security event sources* may be integrated to the framework. In the release 1 prototype, the security monitor collects traffic statistics available from switches. In the future releases, more event information from 5G specific metrics, key performance indicators, and deeper packet analysis are planned. The information must be transformed for Kafka. In release 1, ***pmacct*** traffic collector [1] is used to feed Kafka with network statistic. Pmacct transforms proprietary (pcap derived) or acts as a collector for different standards Netflow[3], SFlow[4] or IPfix[2] and feeds Kafka with data in JSON format.

2.2 Enabler Configuration

Apache Kafka and Spark are distributed with example configurations that may be used in simple testing scenarios with one host. Configuration guidelines on more complex scenarios - using e.g. large amount of clustered servers - can be found from Apache web-site.

Pmacct must be configured to publish data in a Kafka compatible manner. Figure 2 shows the configuration file with directives needed for Kafka.

[3] Apache Spark is a general engine for cluster-based data processing originating from UC Berkeley. It provides specialized data processing libraries (including SQL and DataFrames, MLlib for machine learning, GraphX, and Spark Streaming) which may be combined to create own parallelized applications. Spark is designed to be fast and by supporting mainstream languages (Java, Scala, Python and R languages) it is easily accessible for developers. It can be run as a stand-alone mode or e.g. within Hadoop [11]- which is framework for distributed storage (using HDFS) and processing (using MapReduce) of large sets of (history) data.

```
! ..
plugins: kafka
!
aggregate: src_host, dst_host, src_port, dst_port, proto
interface: eth0
kafka_topic: pmacct.acct
kafka_broker_host: 127.0.0.1
kafka_broker_port: 9092
kafka_refresh_time: 5
kafka_history: 5m
kafka_history_roundoff: m
! ..
```

Figure 2. pmacct configuration file example with Kafka

2.3 Enabler Installation

In the following, we will provide quick guidelines for installation in Linux as well as references for further documentation, which are needed in more complex setups.

Apache Kafka is freely available for Apache's software repository [13]. Configuration guidelines can be found from the Apache web-site [14]. Installation is straightforward in typical scenarios as illustrated in Figure 3.

```
$ wget http://www.nic.funet.fi/pub/mirrors/apache.org/kafka/0.9.0.0/kafka_2.10-0.9.0.0.tgz
$ tar –zxvf kafka
```

Figure 3: Installing ZooKeeper and Kafka broker

Pmacct software [15] and configuration guidelines are available from the pmacc web-site (particularly see [16] Section IX for description how to run Kafka plugin in and [17] for configuration directives). Support for kafka requires json and kafka development libraries. When running a version creating network statistics from pcap data, the pcap development libraries must be installed. Deployment and installation instructions are given Figure 4 and an example configuration is given in Figure 2 The example works when running Kafka on the same machine and with default parameters. When running Kafka on a remote host or using with special requirements, the IP address or other parameters must be set accordingly.

```
sudo apt-get install libpcap-dev librdkafka-dev libjansson-dev
wget http://www.pmacct.net/pmacct-1.5.3.tar.gz
tar -xzf pmacct-1.5.3.tar.gz
cd pmacct-1.5.3
./configure --enable-kafka --enable-jansson
make
```

```
sudo make install
```

Figure 4. Installing pmacct with kafka and json support

Apache Spark software and documentation are available from Apache's web site. It is possible to use standalone Spark or one integrated with Hadoop. The installation, illustrated in Figure 5, is straightforward.

```
$ wget ftp://ftp.funet.fi/pub/mirrors/apache.org/spark/spark-1.6.1/spark-1.6.1-bin-hadoop2.6.tgz
$ tar –xzf  spark-1.6.1-bin-hadoop2.6.tgz
```

Figure 5: Installing Apache Spark

2.4 Troubleshooting

Kafka[6], Spark [8], and Pmacct [1] documentation and web-sites provide component specific troubleshooting information.

3 User and Programmer Guide

3.1 User Guide

The framework requires a working broker which distributes the monitoring information. Figure 6 illustrates how the Apache broker and Zookeeper providing configurations for the broker can be launched.

```
$ cd kafka_2.10-0.9.0.0
$ bin/zookeeper-server-start.sh config/zookeeper.properties
$ bin/kafka-server-start.sh config/server.properties
```

Figure 6: Starting ZooKeeper and Kafka broker

Examples of running event publishers and (Spark-based) event consumers are illustrated in Section 4.

3.2 Programmer Guide

The event information is processed in Spark-based components, which subscribe events through Kafka broker (see [9] and [10]). The components for micro-segment monitor are implemented using Python (it is also possible to use Scala when developing Spark applications).

The *simplified* examples below illustrate some essential functions that may be useful for monitoring:

1) attaching Kafka events to spark's stream abstraction (called DStream) using Spark's KafkaUtils:

```
kafkatopic = "micro-segment1.switch1.eventname"
eventstream = KafkaUtils.createStream(…kafkatopic…)
```

2) parsing / transforming events using map() and reduce():

```
transformedstream = eventstream.map(parsefunc…).reduce(transformfunc…)
```

3) learning a model of 'normal behavior' (using Spark MLlib's K-means clustering algorithm and some previously collected reference data) and then finding distance of new data point (from received event) to the closest previously clustered data points (a long distance indicating an anomaly):

```
model = StreamingKMeans(…).setRandomCenters(…)
clusters = KMeans.train(trainingData…)
distance = clusters.computeCost(transformedstream)
```

4) publishing anomaly event using Kafka producer API [Kafka_API]:

```
Producer<String, String> producer = new KafkaProducer<>(properties);
producer.send(new ProducerRecord<String,String>
("anomaly_detector.anomaly1","anomaly", "eventdetails");
producer.close();
```

4 Unit Tests

4.1 Unit Test 1

Kafka broker can be tested using the example producers and subscriber that are distributed with Kafka. First, start the broker and Zookeeper (Figure 3) and then execute producer and subscriber in own screens (Figure 7 and Figure 8).

```
$ bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --
topic test

$ bin/kafka-topics.sh --list --zookeeper localhost:2181 test

$ bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test

 …..
```

Figure 7: Publishing through Kafka command line tool

```
$ bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic test --from-beginning
```

Figure 8: Subscribing and outputting through Kafka command line tool

Text input in the publisher screen should become visible in the consumer screen.

4.2 Unit Test 2

Network event monitoring can be tested using pmacct and an example application coming with Spark package.

The libpcap-based pmaccd daemon (pmacctd) is executed using command presented in Figure 9.

```
$ sudo pmacctd -f tokafka.conf
```

Figure 9. Executing pmacctd for monitoring IP traffic statistics

```
{"port_src": 48922, "ip_dst": "109.105.109.212", "ip_src": "10.0.2.15", "port_dst": 443,
"ip_proto": "tcp", "stamp_updated": "2016-06-16 08:37:31", "stamp_inserted": "2016-06-16
08:35:00", "packets": 5, "bytes": 323}

{"port_src": 443, "ip_dst": "10.0.2.15", "ip_src": "109.105.109.208", "port_dst": 58726,
"ip_proto": "tcp", "stamp_updated": "2016-06-16 08:37:31", "stamp_inserted": "2016-06-16
08:35:00", "packets": 6, "bytes": 300}
```

Figure 10. Example of Kafka stream containing statistic from monitored network flows

The Spark package contains large amount of example applications. An application subscribing Kafka stream (running in default configuration in localhost port 9092) and outputting it can be executed as shown in Figure 11. The program should output data that has been published through Kafka topic called “pmacct.acct”.

```
$ cd ~/5g/spark/spark-1.6.1-bin-hadoop2.6/
$ bin/spark-submit --packages org.apache.spark:spark-streaming-kafka_2.10:1.6.1
examples/src/main/python/streaming/direct_kafka_wordcount.py localhost:9092
pmacct.acct
```

Figure 11: Running Spark example application

5 Abbreviations

5G-PPP	5G Infrastructure Public Private Partnership
CEP	Complex Event Processing
JSON	JavaScript Object Notation
PCAP	Packet CAPture – application interface to captured packets that are available from libpcap library implementations

6 References

[1] The pmacct project. http://www.pmacct.net.

[2] B. Claise et al. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. IETF, 2013. https://tools.ietf.org/html/rfc7011

[3] B. Claise. Cisco Systems NetFlow Services Export Version 9. Cisco Systems NetFlow Services Export Version 9. IETF, 2004. https://tools.ietf.org/html/rfc3954

[4] Peter Phaal. sFlow Version 5. 2004. http://www.sflow.org/sflow_version_5.txt

[5] The JSON specification. http://www.json.org/

[6] Apache Kafka project. http://kafka.apache.org/

[7] Kafka API http://kafka.apache.org/documentation.html#api

[8] Apache Spark project. http://spark.apache.org/

[9] Spark Streaming Programming Guide. http://spark.apache.org/docs/latest/streaming-programming-guide.html

[10] Spark Streaming + Kafka Integration Guide. http://spark.apache.org/docs/latest/streaming-kafka-integration.html

[11] The Apache Hadoop project. http://hadoop.apache.org

[12] Apache ZooKeeper. https://zookeeper.apache.org/

[13] Kafka download site. http://kafka.apache.org/downloads.html

[14] Kafka quickstart guide. http://kafka.apache.org/documentation.html#quickstart

[15] Pmact download site. http://www.pmacct.net/#downloads

[16] Pmact quickstart guide. Web-site. https://github.com/pmacct/pmacct/blob/master/QUICKSTART.

[17] Pmact configuration directives. Web-site. http://wiki.pmacct.net/OfficialConfigKeys.

Deliverable D3.4
5G-PPP Security Enablers Documentation (v1.0)
Satellite Network Monitoring

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	30.09.2016	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	TASE: Gorka Lendrino	

Contents

1 Introduction

The main goal of this security enabler is to provide pseudo real-time monitoring and threat detection in 5G integrated satellite and terrestrial systems.

“Satellite Network Monitoring” enabler features are in various states of research at the writing time of this document and there is no planned software delivery for all the features. Nevertheless, in the scope of R1 some client-side features are currently being developed as a preliminary open specification and as a proof of concept.

In this SW release, several indicators (including security metrics) will be collected from the listed 5G integrated satellite and terrestrial systems and will be periodically delivered to the monitoring system using. The “Generic Collector Interface” enabler will be analysed (R2) in order to be used as a building block. Later, an active security analysis will be used to detect specific threats and vulnerabilities in the satellite network and response to the threats identified.

The manual is organized as follows. In Section 2, we describe enabler is installed, configured, and administrated, and in Section 3, we describe how to use the enabler. In Section 4, we provide the tests needed to cover all the features of the enabler. In Section 5, we thank people for contributions. Finally, abbreviations and references are listed in Section 6 and Section 7.

2 Installation and Administration Guide

The “Satellite Network Monitoring” feature is the unique part of this security enabler. This section covers the system requirements, and describes how the enabler is installed and configured.

2.1 System Requirements

The “Satellite Network Monitoring” enabler consists of:

- A client-side feature (e.g. eNodeB, satellite terminals, UE....), able to collect indicators from the network component and send them to the server-side feature. This feature shall be deployed on each network component.
- A server-side feature (i.e. security monitoring server), able to configure the client-side features and provide pseudo real-time monitoring of consumed indicators. This feature shall be deployed on a central server.

The basic/minimal system requirements for this enabler are that the server and the network components should be running Linux, preferably Ubuntu Server 16.04.1 LTS or newer on an x64 architecture (you can get help from the Ubuntu server guide [5]), with several network interfaces.

The client host(s) (i.e. 5g-mon-sat-cli01.5g-ensure.eu) in the test-bed is a vSmall virtual host with the following requirement:

- 1 CPU (x86_64)
- 2 GB RAM
- 20 GB Hard disk
- Ethernet network interfaces:
    - Control interface
    - Terrestrial data interface
    - Satellite data interface

It will act as an eNodeB.

The server host (i.e. 5g-mon-sat-srv01.5g-ensure.eu) in the test-bed is a vSmall virtual host with the following requirement:

- 2 CPUs (x86_64)
- 4 GB RAM
- 40 GB Hard disk
- Ethernet network interfaces:
    - Control interface
    - Terrestrial data interface
    - Satellite data interface

It will act as a security monitoring server.

The client-server API follows the RESTful Web Services (JAX-RS 2.0), therefore a Java Application Server is needed in both sides (any should work, but only Apache Tomcat has been tested):

- Client side: to configure the client-side.
- Server side: to configure the server-side.

Also, the server-side feature uses:

- An Apache Active MQ instance
- A PostgreSQL instance

Remote service access:

- Host access to Ubuntu repositories (main, universe).
- Host access to NTP server of stratum ≤ 4.
- SSH access to host (public access or restricted to specific public source IP address).
- HTTPS (TCP/443) access to host (public access or restricted to specific public source IP address).

2.2 Enabler Installation

The installation package contains two archived web applications (client-side and server-side).

Download the enabler package from the software repository of the 5G-ENSURE project (e.g. https://<5g-ensure.git>/enablers/SatelliteNetworkMonitoring/archive/SatelliteNetworkMonitoring.vXX_YY_ZZ.tar.gz) later extract the package on the temporal folder **/tmp** or clone the GitHub project (e.g. https://<5g-ensure.git>/enablers/SatelliteNetworkMonitoring.git) on the temporal folder **/tmp**.

As a result, all the installation files of this enabler will be located in **/tmp/SatelliteNetworkMonitoring.vXX_YY_ZZ**.

2.2.1 Server-side

First of all install the dependencies:

```
$ sudo apt install -y openjdk-8-jre tomcat8 postgresql postgresql-contrib

$ wget http://archive.apache.org/dist/activemq/5.14.0/apache-activemq-5.14.0-bin.tar.gz

$ tar xzvf apache-activemq-5.14.0-bin.tar.gz
```

Then, use the following command:

```
$ /tmp/SatelliteNetworkMonitoring/server/install-server.sh
```

Among other things, this script allows to:

- Set time zone to UTC.
- Configure the server IP address.
- Map hostnames to IP addresses.
- Configure JAVA_HOME and PATH environment variables.
- Create the 5G_MON_SAT environment variable.

2.2.2 Client-side

First of all install the dependencies:

```
$ sudo apt install –y openjdk-8-jre tomcat8
```

Then, use the following command:

```
$ /tmp/SatelliteNetworkMonitoring/client/install-client.sh
```

Among other things, this script allows to:

- Set time zone to UTC.
- Configure the client IP address.
- Map hostnames to IP addresses.
- Configure JAVA_HOME and PATH environment variables.
- Create the 5G_MON_SAT environment variable.

2.3 Enabler Configuration

2.3.1 Server-side

For security of communications it is highly recommended to enable SSL/TSL using the following command:

```
$ /tmp/SatelliteNetworkMonitoring/server/security-server.sh
```

Among other things, this script allows to:

- Generate RSA keys and self-signed certificate

Finally, create the SatNM database using the following command:

```
$ /tmp/SatelliteNetworkMonitoring/server/database.sh
/tmp/SatelliteNetworkMonitoring/server/database.sql
```

2.3.2 Client-side

For security of communications it is highly recommended to enable SSL/TSL using the following command:

```
$ /tmp/FGS-InSatelliteSystems/client/security-client.sh
```

This script configures SSL/TSL accordance with the server (see 2.3.1).

3 User and Programmer Guide

This section describes how to use the enabler and how to “program” the enabler.

3.1 User Guide

This section is not applicable as the enabler is a programmer tool; therefore there are no different user and programmer guides.

3.2 Programmer Guide

The enabler provides a REST interface, a resource-oriented API accessed via HTTP that uses JSON-based representations for information interchange. This Programmers Guide mainly consists of an overview of these RESTful API calls.

Description	Configure the indicators to be collected in the network element (i.e. eNodeB
HTTP Method	POST
URI	https://5g-mon-sat-cli01.5g-ensure.eu:8080/mon-sat-cli/api/v01.00.00/sna/resource/indicators
URI Parameters	N/A
Request Header	N/A
Request Body Content-type	application/json
Request Body Content	indicators:[indicatorType]
HTTP status code	201 when success 400 when error
Response Header	N/A
Accept Content-type	application/json
Response Body Content	N/A

Description	Configure the network element
HTTP Method	POST
URI	https://5g-mon-sat-cli01.5g-ensure.eu:8080/mon-sat-cli/api/v01.00.00/sna/resource/topology
URI Parameters	N/A
Request Header	N/A
Request Body Content-type	application/json
Request Body Content	topologies:[{interfaceName,interfaceStatus}]
HTTP status code	201 when success 400 when error
Response Header	N/A
Accept Content-type	application/json
Response Body Content	N/A

4 Unit Tests

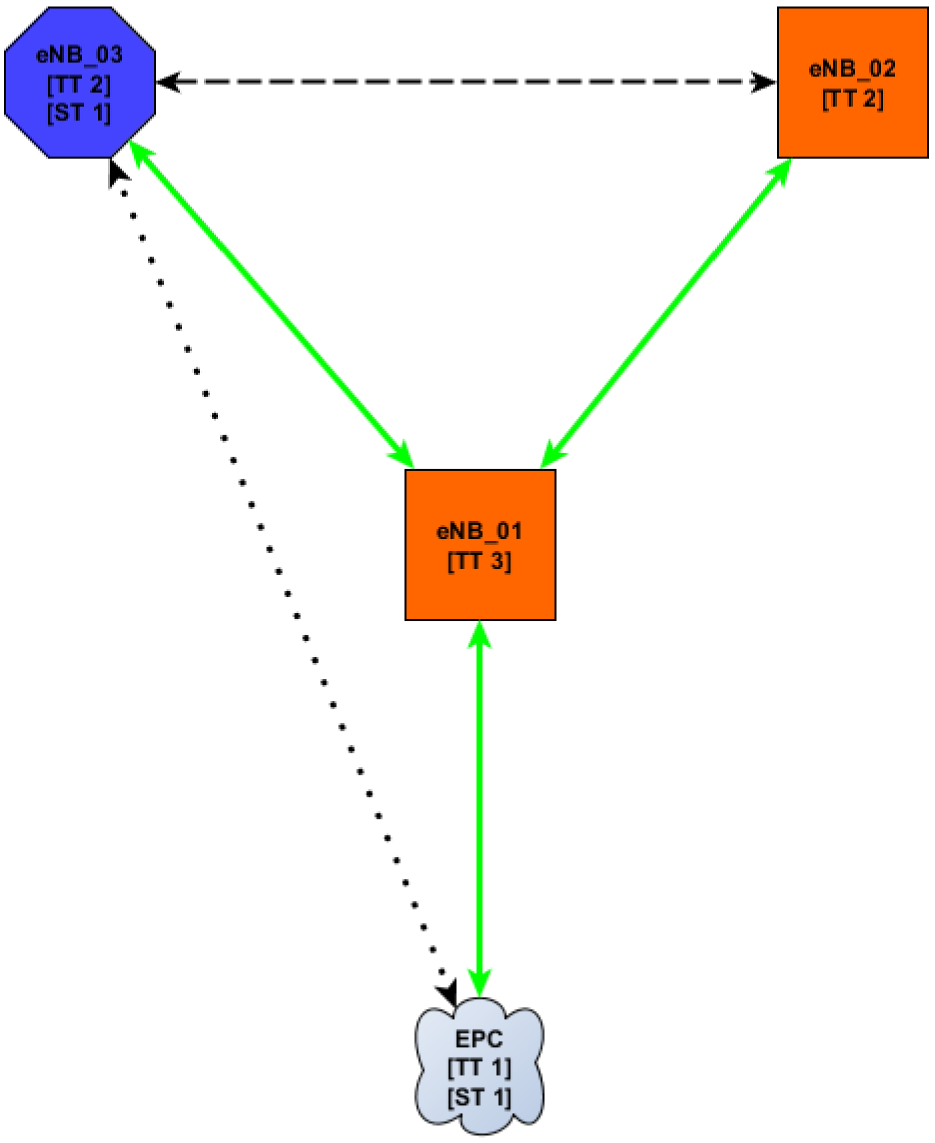
This section describes the tests needed to cover all the features of the enabler. These tests verify that the enabler is ready to be evaluated in WP4.

This features is based on RESTful HTTP API, therefore any HTTP client can be used to perform the tests. For testing purposes curl command line tool has been used.

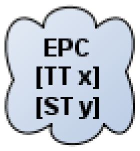
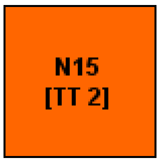
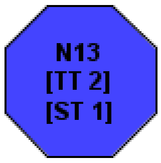
4.1 Information about Tests

Important: the following tests must be performed in order on the server host, and using $5G_MON_SAT directory as current directory.

The scenario used is described below



Legend:

Objects		
		Grey cloud: EPC [TT x] states the number of Terrestrial Terminals [ST x] states the number of Satellite Terminals
		Orange square: eNodeB with terrestrial links [TT x] states the number of Terrestrial Terminals
		Blue octagon: eNodeB with terrestrial and satellite links [TT x] states the number of Terrestrial Terminals [ST x] states the number of Satellite Terminals
Link statuses		
	⟷	Green lines: link power-on
	⟷	Black lines: potential link (currently power off)
	⟷	Red lines: link not enabled
Link types		
	⟷	Solid line: fixed link
	← – – →	Dashed line: Dynamic beam
	← · · · →	Dotted line: Satellite link

4.2 Unit Test 1

This test aims to initialize the indicators to be collected.

The content of the test/UT01/input/UT01_indicators.json file is:

```
{"indicators":[{"indicatorType":< indicatorType 1>},{"indicatorType":< indicatorType 2>},…}
```

Access to the server host and request indicators from the client:

```
$ curl -X POST -H "Content-Type: application/json" -H "Accept: application/json" --data
"@test/UT01/input/UT01_indicators.json" https://5g-mon-sat-cli01.5g-ensure.eu:8080/mon-sat-
cli/api/v01.00.00/sna/resource/indicators
```

Expected result is HTTP status code 201.

If success, server host will start collecting all the configured indicators (see Apache Tomcat log located in /var/log/tomcat8). These indicators are sent from the client to the server using JMS messages (Apache ActiveMQ).

4.3 Unit Test 2

This test aims to initialize the topology.

The content of the test/UT02/input/UT02_topology.json file is:

```
{"topology":[{"interfaceName":"EPC_FixedLink_eNodeB01","interfaceStatus":"on"},{"interfaceName":"E
PC_SatelliteLink_eNodeB03","interfaceStatus":"off"},{"interfaceName":"eNodeB01_FixedLink_EPC","int
erfaceStatus":"on"},{"interfaceName":"eNodeB01_FixedLink_eNodeB02","interfaceStatus":"on"},{"inter
faceName":"eNodeB01_FixedLink_eNodeB03","interfaceStatus":"on"},{"interfaceName":"eNodeB02_Fix
edLink_eNodeB01","interfaceStatus":"on"},{"interfaceName":"eNodeB02_DynamicBeam_eNodeB03","i
nterfaceStatus":"off"},{"interfaceName":"eNodeB03_FixedLink_eNodeB01","interfaceStatus":"on"},{"int
erfaceName":"eNodeB03_DynamicBeam_eNodeB02","interfaceStatus":"off"},{"interfaceName":"eNode
B03_SatelliteLink_EPC","interfaceStatus":"off"}}
```

Access to the server host and forward the topology to the clients:

```
$ curl -X POST -H "Content-Type: application/json" -H "Accept: application/json" --data
"@test/UT02/input/UT02_topology.json" https://5g-mon-sat-cli01.5g-ensure.eu:8080/mon-sat-
cli/api/v01.00.00/sna/resource/topology
```

Expected result is HTTP status code 201.

5 Acknowledgements

The following partners contributed to this deliverable: Thales Alenia Space.

Thanks Jean-Philippe Wary (WPL), 5G-ENSURE peers and 5G-ENSURE Steering Committee for their collaboration and support.

6 Abbreviations

This section comprises a summary of terms and definitions used during the later sections:

5G-PPP	**5G infrastructure Public Private Partnership**
API	**Application Programming Interface**
eNodeB	**evolved Node B**
EPC	**Evolved Packet Core**
HTTP(S)	**HyperText Transfer Protocol (Secure)**
IP	**Internet Protocol**
JMS	**Java Message Service**
JSON	**JavaScript Object Notification**
LTS	**Long Term Support**
LVM	**Logical Volume Manager**
NTP	**Network Time Protocol**
REST	**REpresentational State Transfer**
RSA	**Rivest, Shamir Adleman public-key cryptosystem**
SSH	**Secure SHell**
SQL	**Structured Query Language**
SSL	**Secure Sockets Layer**
ST	**Satellite Terminal**
TCP	**Transmission Control Protocol**
TSL	**Transport Layer Security**
TT	**Terrestrial Terminal**
UE	**User Equipment**
URI	**Uniform Resource Identifier**
UT	**Unit Test**
UTC	**Coordinated Universal Time**

7 References

[1] 5G-Ensure Consortium, Deliverable 2.1 Use Cases. Available online at

http://www.5gensure.eu/sites/default/files/Deliverables/5G-ENSURE_D2.1-UseCases.pdf.

[2] 5G-Ensure Consortium, Deliverable 3.1 5G-PPP security enablers technical roadmap. Available online at http://www.5gensure.eu/sites/default/files/Deliverables/5G-ENSURE_D2.1-UseCases.pdf.

[3] Standard: ETSI - TR 101 984 "SatelliteEarth Stations and Systems (SES); Broadband Satellite Multimedia (BSM); Services and Architectures".

[4] 3GPP TR 22.891, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Feasibility Study on New Services and Markets Technology Enablers; Stage 1 (Release 14)", v1.0.0, September 2015.
Section 5.3
Section 5.20
Section 5.22
Section 5.72

[5] Ubuntu server guide, [Online]. Available: https://help.ubuntu.com/lts/serverguide/index.html

Deliverable D3.4
5G-PPP Security Enablers Documentation (v1.0)
Enabler PulSAR: Proactive Security Analysis and Remediation

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	30.09.2016	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	TS : Theo Combe	

Contents

1 Introduction

The PulSAR enabler (ex-CyberCAPTOR) provides comprehensive risk analysis on a network through attack graph generation in 5G networks. It relies on topological information of the network (firewall rules, flow matrix, routing tables, virtual machines placement, etc) and vulnerability scan information from physical and virtual machines (Nessus [1], OpenVAS [2], Docker-scanner [3]) to enumerate all attack paths from any machine to any other machine. These attack paths are then scored according to their likelihood and difficulty (using metrics such as their length and the CVSS difficulty of the exploited vulnerabilities) and presented to the user through a REST API. At the time of writing this document, a remediation capability is being developed, to suggest possible actions to take to thwart the most critical attack graphs.

The typical use-case of this enabler is a network in which physical machines and placement / routing are mastered and virtual machine images are known (the vulnerability scan can be performed on cold images, for instance in a local image repository).

The enabler is composed of different programs that call each other:

- A Tomcat webserver that exposes the REST API.
- A java program that computes attack paths, scoring and remediation, and acts as the main program of the enabler. It runs as a Tomcat plugin.
- The MulVAL attack graph engine, using the XSB prolog engine. Its input and output are `Datalog` files.
- A python script that generates an input file for PulSAR from the CSV topology files and vulnerability scans. This script is embedded in the container to load the example topology, but should be called standalone in order to post a new topology to the API.
- A SQLITE database containing the vulnerabilities and scoring (CVE, CVSS, CWE) extracted from the NIST database [4].

2 Installation and Administration Guide

2.1 System Requirements

The PulSAR enabler is packaged in a Docker container. It has been tested for small network instances on an Ubuntu 16.04 machine with Docker 1.10.3 (using the `devicemapper` storage backend). The container embeds all dependencies, so it is expected to run out-of-the-box.

The amount of RAM and hard disk needed for PulSAR can be high, according to the network topology. 8GB of RAM and 5GB of hard disk dedicated to the application should be enough for small to medium systems (which is what was used for testing purposes at development time). For medium to big information systems, 32GB of RAM and 30GB of hard disk dedicated to the application may be needed.

2.2 Enabler Configuration

PulSAR's configuration is located in the `config.properties` file in the container, at `/root/.remediation/config.properties`. It can be overridden at container launch adding the switch: `–v /path/to/new/file:/root/.remediation/config.properties`

Here is the `config.properties` embedded in the container:

```
#Mandatory parameters

xsb-path=/opt/XSB/bin

output-path=/root/.remediation/configuration-files/tmp

mulval-path=/opt/mulval

mulval-rules-path=/root/.remediation/configuration-files/rules-with-topology.P

cost-parameters-path=/root/.remediation/cost-parameters

database-path=/opt/cybercaptor/vulnerability-remediation-database.db

python-path=/usr/bin/python

mulval-input-script-folder=/root/.remediation/cyber-data-extract/

host-interfaces-path=/root/.remediation/configuration-files/inputs/hosts-
interfaces.csv

vlans-path=/root/.remediation/configuration-files/inputs/vlans.csv

routing-path=/root/.remediation/configuration-files/inputs/routing.csv

flow-matrix-path=/root/.remediation/configuration-files/inputs/flow-matrix.csv

placement-path=/root/.remediation/configuration-files/inputs/hosts-vms.csv

controllers-path=/root/.remediation/configuration-files/inputs/controllers-
hosts.csv

#vulnerability-scan-path=/root/.remediation/configuration-
files/inputs/scan.nessus

generic-scan-path=/root/.remediation/configuration-files/inputs/scan-
generic.json

mulval-input=/root/.remediation/configuration-files/tmp/mulval-input-generated.P

topology-path=/root/.remediation/configuration-files/tmp/topology-generated.xml

remediations-history-path=/root/.remediation/configuration-
files/tmp/remediations-history.bin

alerts-temporary-path=/root/.remediation/configuration-files/inputs/tmp/alerts-
temp.binalerts-temporary-path=/root/.remediation/cybercaptor-
server/configuration-files/inputs/tmp/alerts-temp.bin
```

This file contains 2 types of parameters:

- Parameters that reference resources the PulSAR server must access inside the container (mulval-path, output-path, etc): these parameters should not be modified.
- Parameters that reference input files that will make the network topology: these filenames start by `/root/.remediation/configuration-files/inputs:` these files may be overridden at container launch by another topology (see the user manual for more information), but the recommended way is to upload the topology through the API once the server is running.

Among these parameters, the vulnerability scan files (`vulnerability-scan-path, openvas-scan-path and generic-scan-path`) are not mandatory, but at least one of them should be provided to generate the attack graphs.

2.3 Enabler Installation

The container is provided as a `.bz2` archive to be extracted and imported by Docker.

Archive extraction :

Type the command:

```
bzip2 –d pulsar.tar.bz2
```

This will create the file `pulsar.tar`

Container import :

Type the command:

```
docker load –i pulsar.tar
```

This will load the container image. You can check that it was successfully loaded by typing

```
docker images
```

The image is called `pulsar-xxx`.

Container launch :

If you want to run the server in foreground, launch the following command:

```
docker run –ti –p 8080:8080 pulsar-xxx
```

It will redirect the port 8080 of the local machine to the port 8080 of the container, on which listens the PulSAR API server. The local machine port can be changed. If an orchestrator is used, the configured exposed port must be 8080.

If you want to run the server in background, launch the following command:

```
docker run –d –p 8080:8080 pulsar-xxx
```

2.4 Troubleshooting

PulSAR errors appear either in error messages in API call responses, or in exceptions thrown and logged by Tomcat. The main logs of the application are stored in the files:

- `/var/log/tomcat7/catalina.out` (the tomcat log file)
- `/root/.remediation/configuration-files/tmp/xsb_log.txt` (the MulVAL log file)
- `/root/.remediation/configuration-files/tmp/input-generation.log` (the cyber-data-extract log file)

These files are in the container. They can be accessed from the host with the docker exec command:

- `docker exec pulsar-xxx tail -n 50 -f /var/log/tomcat7/catalina.out`
- `docker exec pulsar-xxx tail -f /root/.remediation/configuration-files/tmp/xsb_log.txt`

- `docker exec pulsar-xxx tail -f /root/.remediation/configuration-files/tmp/input-generation.log`

They can also be replaced by host files with the –v switch, to get logs directly on the host.

3 User and Programmer Guide

3.1 User Guide

3.1.1 Topological inputs file specifications

Inputs can be passed to the enabler by 3 different ways:

- Override the topological and scan files referenced in the `config.properties` file at container launch. This requires a container restart.
- Call the `cyber-data-extract` preprocessor to generate the XML topology file, then upload it to the PulSAR API.
- **Manually generate the XML topology file**, according to the specification in section 3.1.6, and upload it to the API. **This allows using any vulnerability scanner.**

`cyber-data-extract` can take several types of inputs to generate the XML topology file. We describe in this section the format of the inputs that are currently taken by the script. `cyber-data-extract` may be extended to take into account new types of inputs.

Note that all CSV files use a semi-colon ';' as separator, as it is done by default with Microsoft Excel.

3.1.2 Topological files

In the rest of this documentation, we use the test topology embedded in the container. This topology contains 3 physical hosts, 4 virtual machines and 2 VNFs (running as simple VMs). There are 2 VMs per host and the orchestrator (running on 'machine4') controls the whole network. Machine1, machine2 and machine3 expose a vulnerable service enabling remote code execution on the network, while host2 and host3 have vulnerability in the hypervisor allowing a malicious VM to execute code on the host.

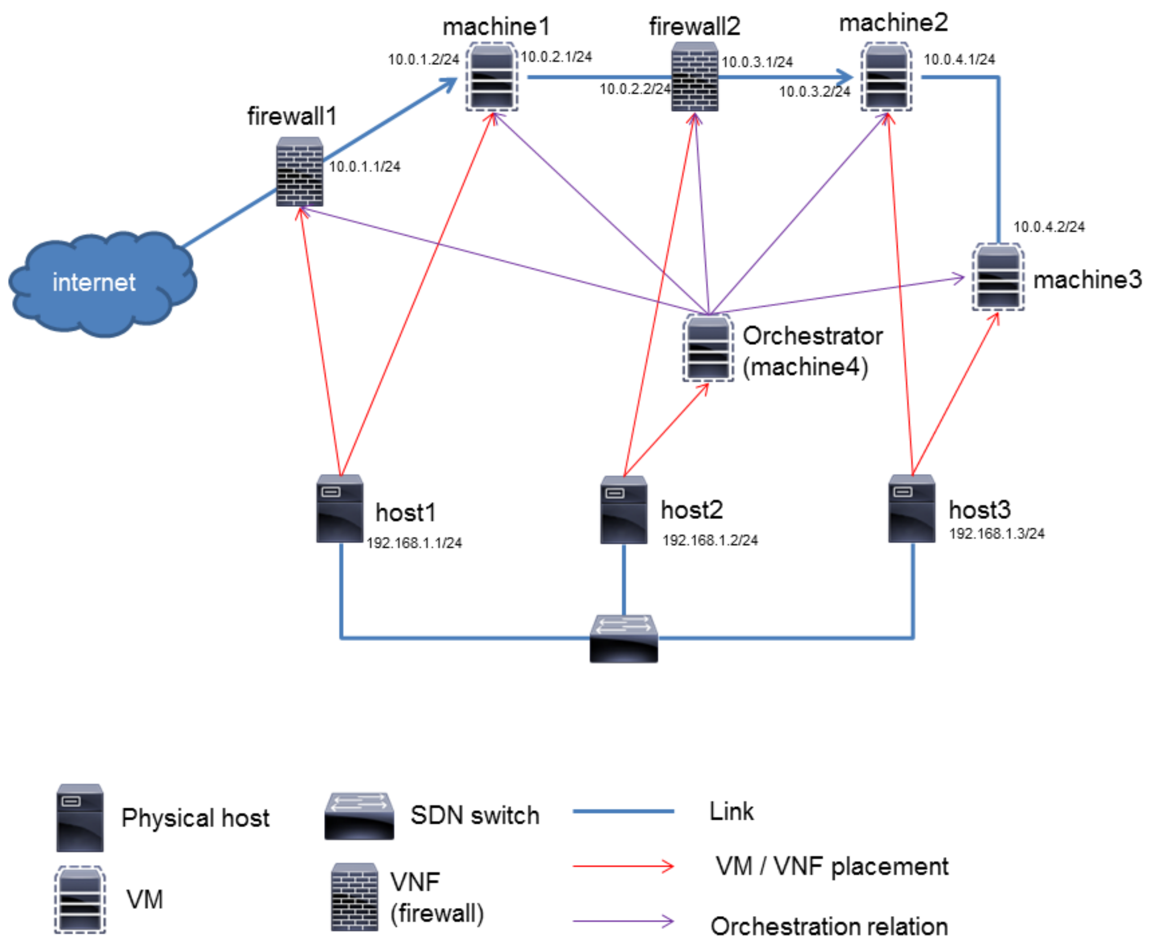


Figure 1 : The example topology

3.1.2.1 *Host-interfaces file*

This CSV file describes the hosts of the topology, with their network interface.

3.1.2.1.1 Columns explanations

Hostname	Interface Name	IP address	Connected to WAN	Metric
The name of the host (without spaces)	The name of the interface	The IP address of the interface	Whether or not this network interface is connected to WAN (Internet)	A Metric describing the importance of the services running on this IP address.

3.1.2.1.2 Example

```
Hostname;ifName;ifAddr;connectedToInternet;metric
firewall1;eth0;42.42.42.42;True;1
firewall1;eth1;10.0.1.1;False;1
machine1;eth0;10.0.1.2;False;1
machine1;eth1;10.0.2.1;False;1
firewall2;eth0;10.0.2.2;False;1
firewall2;eth1;10.0.3.1;False;1
machine2;eth0;10.0.3.2;False;1
```

```
machine2;eth1;10.0.4.1;False;1
machine3;eth0;10.0.4.2;False;1
host1;eth0;192.168.1.1;False;1
host2;eth0;192.168.1.2;False;1
host3;eth0;192.168.1.3;False;1
machine4;eth0;10.0.5.1;False;1
```

3.1.2.2 *Vlans file*

This CSV file describes the subnets/VLANS of the network topology.

3.1.2.2.1 Columns explanations

VLAN Name	VLAN Address	VLAN Netmask	VLAN default Gateway
The name of the VLAN	The IP address of the network	The subnet mask CIDR	The IP address of the VLAN default gateway

3.1.2.2.2 Example

```
name;address;netmask;gateway
vlan0;192.168.1.0;24;192.168.1.254
vlan1;10.0.1.0;24;10.0.1.254
vlan2;10.0.2.0;24;10.0.2.254
vlan3;10.0.3.0;24;10.0.3.254
vlan4;10.0.4.0;24;10.0.4.254
vlan5;10.0.5.0;24;10.0.5.254
```

3.1.2.3 *Flow Matrix File*

This CSV file describes the authorized accesses in the network topology. Note that all accesses that are not specified are supposed unauthorized.

3.1.2.3.1 Columns explanations

Source	Destination	Source port	Destination port	Protocol
The source network (IP/mask) or internet	The destination network (IP/mask) or internet	The source port or any	the destination port or any	the protocol or any.

Each line describes an authorized access.

3.1.2.3.2 Example

```
source;destination;source_port;destination_port;protocol
internet;10.0.1.2;any;443;TCP
10.0.1.2;internet;80;any;TCP
10.0.2.0/24;10.0.3.0/24;any;5432;TCP
10.0.3.0/24;10.0.2.0/24;5432;any;TCP
```

3.1.2.4 *Routing file*

This file describes the routes of the hosts that have routes, others than the default gateways of the interfaces' VLAN.

3.1.2.4.1 *Columns explanations*

Host	Destination	Mask	Gateway	Interface
The name of the host for which this route is specified	The destination network of this route	the network mask of this route	The gateway IP address for this route	The outgoing interface of the route.

3.1.2.4.2 *Example*

```
host;destination;mask;gateway;interface
router;10.15.10.1;255.255.255.0;10.15.10.1;eth1
router;192.168.1.1;255.255.255.0;192.168.1.1;eth0
router;0.0.0.0;0.0.0.0;1.1.1.1;eth2
```

3.1.2.5 *Hosts-vms file*

This file describes the placement of virtual machines on physical hosts.

3.1.2.5.1 *Columns explanations*

Vm	Host	Software	User
The name of the VM	The name of the physical host	The hypervisor software (to be referenced by the vulnerability scan)	The user running the hypervisor software on the host.

3.1.2.5.2 *Example*

```
vm;host;software;user
firewall1;host1;kvm;root
firewall2;host2;kvm;root
machine1;host1;kvm;root
machine2;host3;kvm;root
machine3;host3;kvm;root
machine4;host2;kvm;root
```

3.1.2.6 *Controllers-hosts file*

This file describes the control relationships between controllers / orchestrators and hosts (physical or virtual). This relationship expresses the fact that a controller / orchestrator can execute code on the corresponding machine (for instance through VM image modification and restart).

3.1.2.6.1 *Columns explanations*

Host	controller_global_name
The	The global name of the

name of the machine	controller / orchestrator service, referenced in the services section of its host.

3.1.2.6.2 Example

```
host;controller_global_name
machine1;orchestrateur_global
machine2;orchestrateur_global
machine3;orchestrateur_global
firewall1;orchestrateur_global
firewall2;orchestrateur_global
host1;orchestrateur_global
host2;orchestrateur_global
host3;orchestrateur_global
```

3.1.3 Vulnerability scanner files

Currently, 3 vulnerability scanners can be used: Nessus, OpenVAS and a custom JSON format. For our test topology we used the custom JSON scan format.

3.1.3.1 Nessus scanner files

The outputs of the vulnerability scanner Nessus are stored in a .nessus file, which is an XML file.

The only outputs that are used in this file are:

```
<Report>
<ReportHost name="host ip address or hostname">
<ReportItem port="service port" svc_name="service name" protocol="service proto
col">
<cve>CVE-2015-1234</cve>
<cve>CVE-2015-2345</cve>
</ReportItem>
<ReportItem>
...
</ReportItem>
```

3.1.3.2 OpenVAS files

The outputs of the vulnerability scanner OpenVAS are stored in an XML file.

3.1.3.3 Generic scan file

Other scanner files can be added, provided they are converted to the generic scan file format, stored as a JSON file. This file contains the hosts, services and vulnerabilities information, as follows:

```
{
    "date" : "2016-08-09 11:02:00",
    "hosts" : [
        {
            "name" : "machine1",
            "firstIP" : "10.0.1.2",
            "services" : [
                {
                    "serviceName" : "apache2",
                    "serviceVersion" : "2.2.4-rc10",
                    "serviceProto" : "TCP",
                    "servicePort" : 443,
                    "vulnerabilities" : [
                        {
                            "name" : "CVE-2013-2249"
                        }
                    ]
                }
            ]
        }
    ]
}
```

The “firstIP” field corresponds to the IP of the interface with the default route.

3.1.4 PulSAR Topology XML Input File description

The XML topological file defined here is the main input which is used globally for PulSAR. It unifies all topological data used PulSAR to compute the attack graphs and do the risk analysis.

The main goal of the XML topology file is to describe the network topology, all hosts and their network configuration. Each host can have several network interfaces which can be in different VLAN. The routing and filtering information attached to each host allow computing the network topology (packet route in the network, filtered packets, position of firewalls…). This file can be generated automatically thanks to the cyber-data-extract script.

3.1.5 Description of all fields (xml tags)

3.1.5.1 Machine

Each Machine tag is related to a specific host. This machine tag is used by the Remediation. By way the algorithm is developed, this information is necessary to compute the solutions proposed by Remediation tool.

3.1.5.1.1 Name

- Type : String
- Usage : Contains the name of a host

3.1.5.1.2 Security Requirement (Optional)

- Type : String : NEGLIGEABLE/MINOR/MEDIUM/SEVERE/CATASTROPHIC
- Usage : A value describing a security requirement related to this host.

3.1.5.1.3 Physical host

These XML tags contain attributes related to the physical machine on which the current machine is running. It the current machine is a physical host, this section must be omitted.

3.1.5.1.3.1 Hostname

- Type : String
- Usage : Contains the name of the physical host

3.1.5.1.3.2 Hypervisor

- Type : String
- Usage : Contains the name of the hypervisor program on the host

3.1.5.1.3.3 User

- Type : String
- Usage : Contains the user running the hypervisor program on the host

3.1.5.1.4 Controllers

These XML tags contain the name of the controllers that control the current machine. These names must be global_name properties of services running on controllers.

3.1.5.1.4.1 Controller

- Type : String
- Usage : The name of a controller

3.1.5.1.5 Interfaces - Interface

These XML tags contain all the attributes related to an interface of a machine. Each tag is related to a specific network interface.

3.1.5.1.5.1 Name

- Type : String
- Usage : Contains the name of this interface

3.1.5.1.5.2 VLAN - Name (Optional)

- Type : String

- Usage : Contains the name of the VLAN attached to this interface

3.1.5.1.5.3 VLAN – Label (Optional)

- Type : String
- Usage : Contains the label of the VLAN attached to this interface

3.1.5.1.5.4 IPaddress

- Type : IP address (string)
- Usage : Contains the IP address of this interface

3.1.5.1.6 Services - Service

The description of the network services or applications running on this machine.

3.1.5.1.6.1 Name

- Type : String
- Usage : The name of the service

3.1.5.1.6.2 IPaddress (Optional)

- Type : IP address (string)
- Usage : The IP address on which the service is listenning (if applicable).

3.1.5.1.6.3 Protocol (Optional)

- Type : TCP/UDP/ICMP/ANY (string)
- Usage : The protocol on which the service is listenning (if applicable).

3.1.5.1.6.4 Port (Optional)

- Type : Integer
- Usage : The port on which the service is listenning (if applicable).

3.1.5.1.6.5 Global name (Optional)

- Type : String
- Usage : The global name to identify the service on the network. Is used for instance to identify a controller service in the Controllers section of slave machines.

3.1.5.1.6.6 Vulnerabilities - Vulnerability (Optional)

The vulnerabilities of this service, if applicable.

3.1.5.1.6.6.1 Type

- Type : remoteExploit/localExploit
- Usage : The type of vulnerability (cf CVSS).

3.1.5.1.6.6.2 CVE

- Type : String (CVE-YEAR-1234)
- Usage : The CVE identifier of the vulnerability.

3.1.5.1.6.6.3 Goal

- Type : String
- Usage : The goal of the vulnerability

3.1.5.1.6.6.4 CVSS

- Type : Double
- Usage : The CVSS score of the vulnerability.

3.1.5.1.7 Routes - Route

These XML tags contain the routing table attached to each host. Each tag contains a route of the routing table. Each host needs at least a route containing its default gateway (0.0.0.0/0.0.0.0).

3.1.5.1.7.1 Destination

- Type : IP address (string)
- Usage : Contains the destination network address of the route

3.1.5.1.7.2 Mask

- Type : IP address (string)
- Usage : Contains the network mask of the destination network

3.1.5.1.7.3 Gateway

- Type : String
- Usage : Contains the IP address of the gateway to take for this route (next hop)

3.1.5.1.7.4 Interface

- Type : IP address (string)
- Usage : Contains the interface of the host to use for this route

3.1.5.1.8 Input-Firewall

This XML tag contains the input firewall table attached to each host.

3.1.5.1.8.1 Default-policy

- Type : ALLOW/DENY
- Usage: Contains the default policy of the input firewall table, selected if no firewall line match.

3.1.5.1.8.2 Firewall rule (Optional)

This XML tag contains one line of the input firewall table.

3.1.5.1.8.2.1 Protocol

- Type : String : TCP/UDP/ANY
- Usage : Contains the network flow protocol to match for this firewall line.

3.1.5.1.8.2.2 Source IP

- Type : IP address (string)
- Usage : Contains the source network address to match for this firewall line

3.1.5.1.8.2.3 Source Mask

- Type : IP address (string)
- Usage : Contains the source network mask to match for this firewall line

3.1.5.1.8.2.4 Source Port

- Type : integer or ANY
- Usage : Contains the source port to match for this firewall line

3.1.5.1.8.2.5 Destination IP

- Type : IP address (string)
- Usage : Contains the destination network address to match for this firewall line

3.1.5.1.8.2.6 Destination Mask

- Type : IP address (string)
- Usage : Contains the destination network mask to match for this firewall line

3.1.5.1.8.2.7 Destination Port

- Type : integer or ANY
- Usage : Contains the destination port to match for this firewall line

3.1.5.1.8.2.8 Action

- Type : ACCEPT / DROP
- Usage : Contains the action to do if a packet match this firewall line

3.1.5.1.9 Output-Firewall

This XML tag contains the output firewall table attached to each host.

3.1.5.1.9.1 Default-policy

- Type : ALLOW/DENY
- Usage : Contains the default policy of the output firewall table, selected if no firewall line match.

3.1.5.1.9.2 Firewall rule (Optional)

This XML tag contains one line of the output firewall table.

3.1.5.1.9.2.1 Protocol

- Type : String : TCP/UDP/ANY
- Usage : Contains the network flow protocol to match for this firewall line.

3.1.5.1.9.2.2 Source IP

- Type : IP address (string)
- Usage : Contains the source network address to match for this firewall line

3.1.5.1.9.2.3 Source Mask

- Type : IP address (string)
- Usage : Contains the source network mask to match for this firewall line

3.1.5.1.9.2.4 Source Port

- Type : integer or ANY
- Usage : Contains the source port to match for this firewall line

3.1.5.1.9.2.5 Destination IP

- Type : IP address (string)
- Usage : Contains the destination network address to match for this firewall line

3.1.5.1.9.2.6 Destination Mask

- Type : IP address (string)
- Usage : Contains the destination network mask to match for this firewall line

3.1.5.1.9.2.7 Destination Port

- Type : integer or ANY
- Usage : Contains the destination port to match for this firewall line

3.1.5.1.9.2.8 Action

- Type : ACCEPT / DROP
- Usage : Contains the action to do if a packet match this firewall line

3.1.5.2 Flow-matrix - Flow-matrix-line

This contain all the lines of the flow matrix in this network (all authorized accesses)

3.1.5.2.1 Source - Resource

- Type : String
- Usage : The name of the authorized source resource

3.1.5.2.2 Source - Type

- Type : VLAN/IP
- Usage : The type of the authorized source resource

3.1.5.2.3 Destination - Resource

- Type : String
- Usage : The name of the authorized destination resource

3.1.5.2.4 Destination - Type

- Type : VLAN/IP
- Usage : The type of the authorized destination resource

3.1.5.2.5 Source Port

- Type : Integer
- Usage : The authorized source port

3.1.5.2.6 Destination Port

- Type : Integer
- Usage : The authorized destination port

3.1.5.2.7 Protocol

- Type : TCP/UDP/ICMP/ANY
- Usage : The authorized protocol

3.1.6 APPENDIX: Example topology file

This topology file represents topology from Figure 1.

```
<topology>
  <machine>
    <name>firewall1</name>
    <cpe>cpe:/</cpe>
    <physical_host>
      <hostname>host1</hostname>
      <hypervisor>kvm</hypervisor>
      <user>root</user>
    </physical_host>
    <controllers>
      <controller>orchestrateur_global</controller>
    </controllers>
    <interfaces>
      <interface>
        <name>eth1</name>
        <vlan>
          <name>vlan1</name>
          <label>vlan1</label>
        </vlan>
        <ipaddress>10.0.1.1</ipaddress>
        <directly-connected>
          <ipaddress>10.0.1.2</ipaddress>
        </directly-connected>
      </interface>
      <interface>
        <name>eth0</name>
        <vlan>
          <name />
          <label>6548</label>
        </vlan>
        <ipaddress>42.42.42.42</ipaddress>
        <directly-connected>
          <internet />
        </directly-connected>
      </interface>
    </interfaces>
    <services />
    <routes />
    <input-firewall>
      <default-policy>ACCEPT</default-policy>
    </input-firewall>
    <output-firewall>
      <default-policy>ACCEPT</default-policy>
    </output-firewall>
  </machine>
  <machine>
    <name>machine1</name>
    <cpe>cpe:/</cpe>
    <physical_host>
      <hostname>host1</hostname>
      <hypervisor>kvm</hypervisor>
      <user>root</user>
    </physical_host>
    <controllers>
      <controller>orchestrateur_global</controller>
    </controllers>
    <interfaces>
      <interface>
        <name>eth1</name>
        <vlan>
          <name>vlan2</name>
          <label>vlan2</label>
        </vlan>
        <ipaddress>10.0.2.1</ipaddress>
        <directly-connected>
          <ipaddress>10.0.2.2</ipaddress>
        </directly-connected>
      </interface>
      <interface>
        <name>eth0</name>
        <vlan>
          <name>vlan1</name>
          <label>vlan1</label>
        </vlan>
        <ipaddress>10.0.1.2</ipaddress>
        <directly-connected>
          <ipaddress>10.0.1.1</ipaddress>
        </directly-connected>
      </interface>
    </interfaces>
    <services>
      <service>
        <name>apache2</name>
        <global_name />
        <ipaddress>10.0.1.2</ipaddress>
        <protocol>TCP</protocol>
        <port>443</port>
        <CPE>cpe:/</CPE>
        <vulnerabilities>
          <vulnerability>
            <type>remoteExploit</type>
            <goal>privEscalation</goal>
            <cve>CVE-2013-2249</cve>
          </vulnerability>
        </vulnerabilities>
      </service>
    </services>
    <routes />
    <input-firewall>
      <default-policy>ACCEPT</default-policy>
    </input-firewall>
    <output-firewall>
      <default-policy>ACCEPT</default-policy>
    </output-firewall>
  </machine>
  <machine>
    <name>firewall2</name>
    <cpe>cpe:/</cpe>
    <physical_host>
      <hostname>host2</hostname>
      <hypervisor>kvm</hypervisor>
      <user>root</user>
    </physical_host>
    <controllers>
      <controller>orchestrateur_global</controller>
    </controllers>
    <interfaces>
      <interface>
        <name>eth1</name>
        <vlan>
          <name>vlan3</name>
          <label>vlan3</label>
        </vlan>
        <ipaddress>10.0.3.1</ipaddress>
        <directly-connected>
          <ipaddress>10.0.3.2</ipaddress>
        </directly-connected>
      </interface>
      <interface>
        <name>eth0</name>
        <vlan>
          <name>vlan2</name>
          <label>vlan2</label>
        </vlan>
        <ipaddress>10.0.2.2</ipaddress>
        <directly-connected>
          <ipaddress>10.0.2.1</ipaddress>
        </directly-connected>
      </interface>
    </interfaces>
    <services />
    <routes />
    <input-firewall>
      <default-policy>ACCEPT</default-policy>
    </input-firewall>
    <output-firewall>
      <default-policy>ACCEPT</default-policy>
    </output-firewall>
  </machine>
  <machine>
    <name>machine2</name>
    <cpe>cpe:/</cpe>
    <physical_host>
      <hostname>host3</hostname>
      <hypervisor>kvm</hypervisor>
      <user>root</user>
    </physical_host>
    <controllers>
      <controller>orchestrateur_global</controller>
    </controllers>
    <interfaces>
      <interface>
        <name>eth1</name>
        <vlan>
          <name>vlan4</name>
          <label>vlan4</label>
        </vlan>
        <ipaddress>10.0.4.1</ipaddress>
        <directly-connected>
          <ipaddress>10.0.4.2</ipaddress>
        </directly-connected>
      </interface>
      <interface>
        <name>eth0</name>
        <vlan>
          <name>vlan3</name>
          <label>vlan3</label>
        </vlan>
        <ipaddress>10.0.3.2</ipaddress>
        <directly-connected>
          <ipaddress>10.0.3.1</ipaddress>
        </directly-connected>
      </interface>
    </interfaces>
    <services>
      <service>
        <name>postgreSQL</name>
        <global_name />
        <ipaddress>10.0.3.2</ipaddress>
        <protocol>TCP</protocol>
        <port>5432</port>
        <CPE>cpe:/</CPE>
        <vulnerabilities>
          <vulnerability>
            <type>remoteExploit</type>
            <goal>privEscalation</goal>
            <cve>CVE-2013-1903</cve>
          </vulnerability>
        </vulnerabilities>
      </service>
    </services>
    <routes />
    <input-firewall>
      <default-policy>ACCEPT</default-policy>
    </input-firewall>
    <output-firewall>
      <default-policy>ACCEPT</default-policy>
    </output-firewall>
  </machine>
  <machine>
    <name>machine3</name>
    <cpe>cpe:/</cpe>
    <physical_host>
      <hostname>host3</hostname>
      <hypervisor>kvm</hypervisor>
      <user>root</user>
    </physical_host>
    <controllers>
      <controller>orchestrateur_global</controller>
    </controllers>
    <interfaces>
      <interface>
        <name>eth0</name>
        <vlan>
          <name>vlan4</name>
          <label>vlan4</label>
        </vlan>
        <ipaddress>10.0.4.2</ipaddress>
        <directly-connected>
          <ipaddress>10.0.4.1</ipaddress>
        </directly-connected>
      </interface>
    </interfaces>
    <services>
      <service>
        <name>apache2</name>
        <global_name />
        <ipaddress>10.0.4.2</ipaddress>
        <protocol>TCP</protocol>
        <port>80</port>
        <CPE>cpe:/</CPE>
        <vulnerabilities>
          <vulnerability>
            <type>remoteExploit</type>
            <goal>privEscalation</goal>
            <cve>CVE-2011-3192</cve>
          </vulnerability>
        </vulnerabilities>
      </service>
    </services>
    <routes>
      <route>
        <destination>0.0.0.0</destination>
        <mask>0.0.0.0</mask>
        <gateway>10.0.4.254</gateway>
        <interface>eth0</interface>
      </route>
    </routes>
    <input-firewall>
      <default-policy>ACCEPT</default-policy>
    </input-firewall>
    <output-firewall>
      <default-policy>ACCEPT</default-policy>
    </output-firewall>
  </machine>
  <machine>
    <name>host1</name>
    <cpe>cpe:/</cpe>
    <controllers>
      <controller>orchestrateur_global</controller>
    </controllers>
    <interfaces>
      <interface>
        <name>eth0</name>
        <vlan>
          <name>vlan0</name>
          <label>vlan0</label>
        </vlan>
        <ipaddress>192.168.1.1</ipaddress>
        <directly-connected>
          <ipaddress>192.168.1.2</ipaddress>
          <ipaddress>192.168.1.3</ipaddress>
        </directly-connected>
      </interface>
    </interfaces>
    <services>
      <service>
        <name>kvm</name>
        <global_name>host1</global_name>
        <ipaddress>192.168.1.1</ipaddress>
        <protocol>ANY</protocol>
        <CPE>cpe:/</CPE>
      </service>
    </services>
    <routes>
      <route>
        <destination>0.0.0.0</destination>
        <mask>0.0.0.0</mask>
        <gateway>192.168.1.254</gateway>
        <interface>eth0</interface>
      </route>
    </routes>
    <input-firewall>
      <default-policy>ACCEPT</default-policy>
    </input-firewall>
    <output-firewall>
      <default-policy>ACCEPT</default-policy>
    </output-firewall>
  </machine>
  <machine>
    <name>host2</name>
    <cpe>cpe:/</cpe>
    <controllers>
      <controller>orchestrateur_global</controller>
    </controllers>
```

```
<interfaces>
  <interface>
    <name>eth0</name>
    <vlan>
      <name>vlan0</name>
      <label>vlan0</label>
    </vlan>
    <ipaddress>192.168.1.2</ipaddress>
    <directly-connected>
      <ipaddress>192.168.1.1</ipaddress>
      <ipaddress>192.168.1.3</ipaddress>
    </directly-connected>
  </interface>
</interfaces>
<services>
  <service>
    <name>kvm</name>
    <global_name>host2</global_name>
    <ipaddress>192.168.1.2</ipaddress>
    <protocol>ANY</protocol>
    <CPE>cpe:/</CPE>
    <vulnerabilities>
      <vulnerability>
        <type>localExploit</type>
        <goal>privEscalation</goal>
        <cve>CVE-2011-4622</cve>
      </vulnerability>
    </vulnerabilities>
  </service>
</services>
<routes>
  <route>
    <destination>0.0.0.0</destination>
    <mask>0.0.0.0</mask>
    <gateway>192.168.1.254</gateway>
    <interface>eth0</interface>
  </route>
</routes>
<input-firewall>
  <default-policy>ACCEPT</default-policy>
</input-firewall>
<output-firewall>
  <default-policy>ACCEPT</default-policy>
</output-firewall>
```

```
  </machine>
  <machine>
    <name>host3</name>
    <cpe>cpe:/</cpe>
    <controllers>
      <controller>orchestrateur_global</controller>
    </controllers>
    <interfaces>
      <interface>
        <name>eth0</name>
        <vlan>
          <name>vlan0</name>
          <label>vlan0</label>
        </vlan>
        <ipaddress>192.168.1.3</ipaddress>
        <directly-connected>
          <ipaddress>192.168.1.1</ipaddress>
          <ipaddress>192.168.1.2</ipaddress>
        </directly-connected>
      </interface>
    </interfaces>
    <services>
      <service>
        <name>kvm</name>
        <global_name>host3</global_name>
        <ipaddress>192.168.1.3</ipaddress>
        <protocol>ANY</protocol>
        <CPE>cpe:/</CPE>
        <vulnerabilities>
          <vulnerability>
            <type>localExploit</type>
            <goal>privEscalation</goal>
            <cve>CVE-2011-4622</cve>
          </vulnerability>
        </vulnerabilities>
      </service>
    </services>
    <routes>
      <route>
        <destination>0.0.0.0</destination>
        <mask>0.0.0.0</mask>
        <gateway>192.168.1.254</gateway>
        <interface>eth0</interface>
      </route>
    </routes>
    <input-firewall>
      <default-policy>ACCEPT</default-policy>
    </input-firewall>
    <output-firewall>
      <default-policy>ACCEPT</default-policy>
    </output-firewall>
  </machine>
  <machine>
    <name>machine4</name>
    <cpe>cpe:/</cpe>
    <physical_host>
      <hostname>host2</hostname>
      <hypervisor>kvm</hypervisor>
      <user>root</user>
    </physical_host>
    <interfaces>
      <interface>
        <name>eth0</name>
        <vlan>
          <name>vlan5</name>
          <label>vlan5</label>
        </vlan>
        <ipaddress>10.0.5.1</ipaddress>
        <directly-connected />
      </interface>
    </interfaces>
    <services>
      <service>
        <name>orchestratord</name>
        <global_name>orchestrateur_global</global_name>
        <ipaddress>10.0.5.1</ipaddress>
        <protocol>ANY</protocol>
        <CPE>cpe:/</CPE>
      </service>
    </services>
    <routes>
      <route>
        <destination>0.0.0.0</destination>
        <mask>0.0.0.0</mask>
        <gateway>10.0.5.254</gateway>
        <interface>eth0</interface>
      </route>
    </routes>
    <input-firewall>
      <default-policy>ACCEPT</default-policy>
    </input-firewall>
    <output-firewall>
      <default-policy>ACCEPT</default-policy>
    </output-firewall>
  </machine>
  <machine>
    <name>internet_host</name>
    <cpe>cpe:/</cpe>
    <interfaces />
    <services />
    <routes />
    <input-firewall>
      <default-policy>ACCEPT</default-policy>
    </input-firewall>
    <output-firewall>
      <default-policy>ACCEPT</default-policy>
    </output-firewall>
  </machine>
  <flow-matrix>
    <flow-matrix-line>
      <source type="INTERNET" />
      <destination type="IP" resource="10.0.1.2" />
      <source_port>any</source_port>
      <destination_port>443</destination_port>
      <protocol>TCP</protocol>
    </flow-matrix-line>
    <flow-matrix-line>
      <source type="IP" resource="10.0.1.2" />
      <destination type="INTERNET" />
      <source_port>80</source_port>
      <destination_port>any</destination_port>
      <protocol>TCP</protocol>
    </flow-matrix-line>
    <flow-matrix-line>
      <source type="VLAN" resource="vlan2" />
      <destination type="VLAN" resource="vlan3" />
      <source_port>any</source_port>
      <destination_port>5432</destination_port>
      <protocol>TCP</protocol>
    </flow-matrix-line>
    <flow-matrix-line>
      <source type="VLAN" resource="vlan3" />
      <destination type="VLAN" resource="vlan2" />
      <source_port>5432</source_port>
      <destination_port>any</destination_port>
      <protocol>TCP</protocol>
    </flow-matrix-line>
  </flow-matrix>
</topology>
```

3.2 Programmer guide

<u>**API usage :**</u>

In the following, we assume the container is running and port 8080 of the host redirects to port 8080 of the container, and that all commands are issued on the host.

<u>**Initialization calls:**</u>

Before using the API to manipulate the attack graph, the attack paths, and the remediations, the PulSAR server must be initialized with the topology and vulnerability scans. The attack graph is then generated calling MulVAL, and all attack paths are computed on initialization. There are 2 ways to initialize the server:

- Initialization through input files inside the container. The corresponding API call is: `curl -c /tmp/curl.cookie http://localhost:8080/cybercaptor-server/rest/json/initialize`
  This will fetch the `.csv` files inside the container containing the topology and the `.xml / .json` file containing the vulnerability scans referenced in the `config.properties` file. By default these file contain a test topology, and can be overridden with the –v switch on the `docker run` command. The detail of these files (located at `/root/.remediation/cybercaptor-server/configuration-files/inputs`) is given in the “input detail” section.
- Initialization through posting an XML file containing all the topology to the API. This file is uploaded with the API call:
  ```
  curl -c /tmp/curl.cookie -X POST  -H "Content-Type: multipart/form-data"  -F "file=@./topology.xml" http://localhost:8080/cybercaptor-server/rest/json/initialize
  ```
  The `topology.xml` must be generated by the cyber-data-extract program, as a preprocessor. This program takes as input the `.csv` files containing the topology and the `.xml / .json` files containing the vulnerability scans.

<u>**Attack graph, attack paths and remediation calls :**</u>

Then, the calls to get the attack paths, attack graph or remediations can be used:

<u>Get the number of attack paths:</u>

```
curl -b /tmp/curl.cookie http://localhost:8080/cybercaptor-server/rest/json/attack_path/number
```

Note the `-b /tmp/curl.cookie` option of curl, to load the previously saved session cookie.

<u>Get the attack path 0:</u>

```
curl -b /tmp/curl.cookie http://localhost:8080/cybercaptor-
server/rest/json/attack_path/0
```

Get the attack graph:

```
curl -b /tmp/curl.cookie http://localhost:8080/cybercaptor-
server/rest/json/attack_graph
```

Get the remediations for attack path 0:

```
curl -b /tmp/curl.cookie http://localhost:8080/cybercaptor-
server/rest/json/attack_path/0/remediations
```

Get the XML network topology (useful for backups, same format as the XML input file):

```
curl -b /tmp/curl.cookie http://localhost:8080/cybercaptor-
server/rest/json/topology
```

These API calls can be tested with the input files embedded in the container, located at:

```
/data/build/cybercaptor-server/configuration-files/inputs
```

4 Unit Tests

4.1 Information about Tests

These tests describe basic procedures to check if PulSAR is correctly installed and running. They check that all components of PulSAR (Tomcat, MulVAL, cyber-data-extract, and the java core) are installed by making some calls to the API.

4.2 Unit Test 1

This test aims at verifying that the Tomcat server and the PulSAR module are correctly installed and running. This first call to test that the server is available is:

```
curl http://localhost:8080/cybercaptor-server/rest/version/detailed
```

which should return something like :

```
{"version":"4.4"}
```

Otherwise, log files described in section 2.4 should contain relevant information.

4.3 Unit Test 2

A basic test to make sure the enabler is properly installed is calling the ‘initialize’ API call. This test aims at verifying that the PulSAR server can call cyber-data-extract to generate the topology, then MulVAL and generate attack graphs, attack paths and remediations. The test relies on the example data provided in the container.

If the port 8080 of the host was redirected to the API server, the command:

```
curl http://localhost:8080/cybercaptor-server/rest/json/initialize
```

on the host should return `{"status":"Loaded"}`.

Otherwise, relevant information should be in the log files.

5 Acknowledgements

We would like to thank the FiWare-CyberCAPTOR development team for their work.

6 Abbreviations

5G-PPP	5G Infrastructure Public Private Partnership

7 References

[1] "NESSUS," [Online]. Available: http://www.tenable.com/products/nessus-vulnerability-scanner. [Accessed 22 09 2016].

[2] "OpenVAS," 2016. [Online]. Available: http://www.openvas.org/. [Accessed 22 09 2016].

[3] "CoreOS - CLAIR," 2016. [Online]. Available: https://github.com/coreos/clair. [Accessed 22 09 2016].

[4] "NIST NVD vulnerability feed," [Online]. Available: https://nvd.nist.gov/download.cfm. [Accessed 22 09 2016].

Deliverable D3.4
5G-PPP Security Enablers Documentation (v1.0)
Enabler Access Control Mechanisms

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	30.09.2016	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	NEC: Felix Klaedtke, Mihai Moraru	

European Commission

Contents

1 Introduction

In 5G, a much stronger adoption of software-defined networking (SDN) is expected than in current telecommunication networks. It is also expected that various network applications will run at the network's control plane on top of an SDN controller. These applications will manage the network's data plane and offer a wide range of network services. Examples of such applications are routing applications, load balancers, and monitoring and analysis tools for network traffic. The diversity of network applications and their large-scale deployment actually applies to SDN in general. The network applications, however, might not be trusted by the network operator. Reasons for this are: (1) they might be from different network tenants or service providers, (2) they might be developed by third parties, or (3) they might contain bugs—as any complex software—and the control plane is therefore vulnerable to various kinds of attacks. Note that even if the network applications run in separate virtualized networks or network slices, they still indirectly access the same physical network resources.

Widely used APIs, as expected in 5G networks and the resulting deployment of various third-party applications pose new security threats in SDN. Indeed, "malicious" network applications can leverage such an API and infiltrate the network. These threats become even more evident when the network owner "leases" network slices, which are administrated by the leasing tenants, which in turn might install their own, possibly third-party, network applications on top of the network owner's controller. Since the leasing tenants might have competing objectives, access to the shared network resources must be appropriately handled and secured. Cf. the use case 5.2 of Deliverable 2.1 [1].

The security enabler of this section applies the principle of least privilege [11] to the network applications in an SDN network, that is, the enabler enforces that each network application must only be able to access the information and resources that are necessary for performing its tasks. To this end, a *reference monitor* is added to the network's control plane as a component of the state-of-the-art SDN controller [3]. The reference monitor permits and denies access to network resources according to a given access control policy [4, 5]. In particular, the reference monitor targets the controller's southbound interface (SBI), i.e., it restricts the OpenFlow [7] messages a network application can send to the components at the network's data plane.

Note that the enabler is "SDN generic," i.e., in principle any SDN controller can be extended with a reference monitor. We choose to build the reference monitor for the state-of-the-art controller ONOS [3, 9]. This manual describes how the reference monitor for ONOS is installed, administrated, and used. We refer to the ONOS webpage [9] for details on the ONOS controller.

The manual is organized as follows. In Section 2, we describe how the SDN controller ONOS with the reference monitor is installed and administrated, and in Section 3, we describe how the reference monitor is used, in particular, the specification of access control policies. In Section 4, we provide some basics tests for the reference monitor. Abbreviations are listed in Section 5.

2 Installation and Administration Guide

The reference monitor is distributed as an extension to the ONOS controller. They are bundled together in one Debian package. The package, `onos-1.3.0.deploy.deb`, is available from the 5G-ENSURE software repository. Detailed installation and configuration instructions follow.

2.1 System Requirements

The reference monitor and ONOS are Java applications that should run on top of any Java VM, regardless of the underlying operating system. Nevertheless, the package format is specific to Debian GNU/Linux and its derivatives, in particular Ubuntu.

The enabler has been tested on the Ubuntu Server 16.04 GNU/Linux distribution with Oracle JDK 8 (version 1.8.0_101). The daemon control scripts are compatible with the old `init` and with `systemd`. We recommend using the same Ubuntu Server LTS release (with eventual security updates) and Oracle JDK 8 (with eventual minor updates).

The hardware requirements are dictated by the ONOS controller. The reference monitor footprint is negligible. The ONOS project recommendation is at least 2GiB of RAM and 2 or more (virtual) cores. The enabler has been tested with 2GiB of RAM and 4 cores, although performance testing can require up to 8GiB of RAM.

A base Ubuntu Server installation with ONOS and a few utilities (OpenSSH, python, editors) fits in less than 2GiB of disk space. More is required for correct OS operation, logs, and additional software.

It is assumed that the deployment user has sudo access.

In summary, for installing and administrating the reference monitor for ONOS, the following system requirements need to be fulfilled.

- Ubuntu Server 16.04 LTS 64-bit
- Java 8 JDK (Oracle Java recommended)
- 2GiB or more memory
- 2 or more processors
- 8GiB disk storage
- network connectivity for remote administration and to the data plane
- sudo access

In addition, please make sure that the SDN switches are using OpenFlow version 1.0.

2.2 Enabler Configuration

Reference monitor. The reference monitor is configured by specifying the desired whitelist in the file `/etc/onos/refmon/policy.txt`. Section 3.1.2 gives details on the content of this file.

The command `print-refmon-logs` shows the reference monitor logs.

Interacting with ONOS. Existing ONOS commands still apply. In particular, the command-line interface (CLI) of ONOS can be used. The CLI can for example be used to add and remove flows, intents, etc. See https://wiki.onosproject.org/display/ONOS/The+ONOS+CLI.

Configuring ONOS. The ONOS home resides in the directory `/opt/onos`. The file `/opt/onos/options` can be used to control some aspects of ONOS:

```
  export ONOS_APPS="drivers,openflow,fwd,proxyarp,mobility,ref.mon"
```

controls which application are started automatically.

Manually removing `/opt/onos/apps/$app/active` disables an application `$app` from starting automatically.

ONOS itself is built on top of Apache Karaf (an OSGI implementation). Karaf commands can still be used to manage bundles as indicated in the Karaf documentation. See http://karaf.apache.org/. Moreover, the files in `/opt/onos/karaf/etc/*` can be used to adjust Karaf parameters if the user is familiar with them.

2.3 Enabler Installation

First, install Oracle JDK 8. The following commands will add a new repository that contains the non-free Oracle Java, and then install the package. The user needs to accept the license terms.

```
$ sudo apt-get install software-properties-common -y
$ sudo add-apt-repository ppa:webupd8team/java -y
$ sudo apt-get update
$ sudo apt-get install -y oracle-java8-installer oracle-java8-set-
default
```

Second, install the enabler package.

```
$ sudo dpkg -i /tmp/onos-1.3.0.deploy.deb
```

It is assumed here that the Debian package comprising ONOS and the reference monitor, has been downloaded from 5G-Ensure software repository and is located in the `/tmp` directory.

Finally, once installed, ONOS can be started by executing

```
$ sudo /etc/init.d/onos start
```

Alternatively, the arguments `status` and `stop` can be used to check whether ONOS is running and to stop it, respectively.

Once the daemon is running, one can connect to it with the command:

```
$ /opt/onos/bin/onos
```

2.4 Troubleshooting

2.4.1 Installation

If errors are encountered during the installation of the package, make sure that JDK 8 has been installed. Unlike the APT front-end, `dpkg` does not automatically install dependencies.

2.4.2 Running

Depending on the hardware configuration, ONOS can take a bit more time to start. Even when the Karaf container reports to be up and running, it might still be initializing bundles in the background.

When starting the command-line interface of ONOS, the following

```
$ /opt/onos/bin/onos
Logging in as karaf
Failed to get the session.
```

means that Karaf is not running, or that it is still initializing. One can check that Karaf is indeed running with `$ sudo /etc/init.d/onos status` or with `$ ps ax | grep onos`.

```
onos> apps -s -a
Command not found: apps
```

means that the Karaf initialization is still in an early stage and has not loaded the “apps” command, which is specific to ONOS.

```
onos> apps -s -a
Service org.onosproject.app.ApplicationService not found
```

means that Karaf is still initializing. It has loaded the command, but the corresponding service has not been activated yet.

```
onos> print-refmon-logs
Service org.onosproject.refmon.ReferenceMonitorService not found
```

means that the reference monitor is not running yet (either because Karaf is still busy initializing, or perhaps the reference monitor has not been activated).

When exiting ONOS, the message

```
[WARN] Failed to create directory: /home/onos/.karaf
```

is harmless. It indicates that the command history could not be saved. Creating a home for the ONOS system user would allow the commands to be saved if this is more convenient for the administrator.

If the outputs of `onos> apps -s -a` and `onos> bundle:list` disagree about the status of the reference monitor, or if the reference monitor does not behave as expected for some unknown reason, the easiest way is to remove the package and install it again.

```
$ sudo dpkg --purge onos
$ sudo rm -rf /opt/onos
$ sudo dpkg -i /path/to/onos-1.3.0.deploy.deb
```

2.4.3 Connecting to Mininet

Experiments are often performed on simulated networks with Mininet [6]. ONOS, as any SDN controller, can easily be used to control such a network. See http://mininet.org/ for more details on Mininet, including tutorials.

If Mininet shows:

```
Unable to contact the remote controller at x.x.x.x:6633
```

make sure that the IP address is the correct one for reaching the controller and that there are no interfering systems like a firewall. This also happens if Mininet is started before the controller. This is normally not an issue. When the controller is available, Mininet will connect to it.

Other Mininet errors might be due to previous improper shutdowns (e.g., when Mininet crashes for some reason). After an improper shutdown, one should execute:

```
$ sudo mn –c
```

to clean up Mininet.

3 User and Programmer Guide

3.1 User Guide

3.1.1 Overview

The main component of the enabler is the reference monitor, which is a component of an SDN controller. The reference monitor's inputs are an access control policy and access requests. For each access request, it outputs a permit or deny. See Figure 1 for illustration.

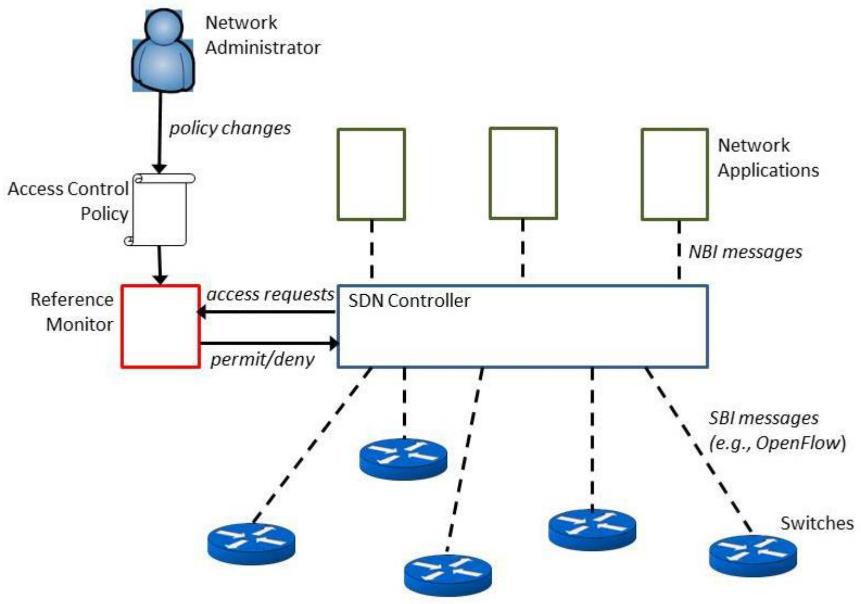


Figure 1 SDN network components and their interactions.

In the first release of the reference monitor, we opted for access control mechanisms (access control policies and reference monitor) that are simple, focused, and close to the SBI of the controller, which interfaces directly with the switches using the OpenFlow protocol. The rationale behind this design decision is as follows. First, it enables one to build a tamperproof and verifiable reference monitor. This is based on the scheme's simplicity and its focus on the controller's SBI. Furthermore, since the controller only communicates via OpenFlow messages with the switches, we obtain complete mediation by permitting or denying OpenFlow messages by the reference monitor before they are sent. These are essential principles for a reference monitor [2].

The access requests are initiated by network events. For example, a network application requests to install new flow rules in a switch, or a switch receives a network packet that does not match any of its flow rules and is thus forwarded to the controller. In the first case, the controller requests to send OpenFlow messages of the type OFPT_FLOW_MOD to a switch, which, e.g., originate from a NBI message sent from a network application to the controller. In the second case, the controller receives an OpenFlow message of the type OFPT_PACKET_IN from a switch and requests to forward the message to a network application.

The decisions of the reference monitor are according to the given access control policy. The network administrator can make changes to this policy. The network applications can also make policy changes. However, their changes are limited and only concern the objects that they own. For example, a network application A can allow another network application B to read the counters of a flow rule that A owns (i.e., created).

3.1.2 Access Control Policies

When the reference monitor is enabled, only switch rules installed by the application `org.onosproject.core` are allowed. All other applications are blocked by default. Exceptions are specified in the policy file (whitelist). The file (`policy.txt`) is located in the directory `/etc/onos/refmon/`. The policy file contains one policy entry per line, each containing four space-separated fields as detailed in Table 1.

Table 1 Policy file structure.

Option	Description	Example
`switch_dpid`	Data path ID of the target switch	`00:00:00:00:00:00:00:01`
`application_id`	ID of the application where the rule originates from	`org.onosproject.fwd`
`mac_src`	Source MAC address	`00:00:00:00:00:00:00:01`
`mac_dst`	Destination MAC address	`00:00:00:00:00:00:00:02`

When the reference monitor matches a rule against the whitelist, the rule is allowed to pass the SBI and be installed in the switches. Otherwise, the rule is in violation of the policy and is discarded.

Note: the controller is not aware that the reference monitor may block some rules. It might continue retrying, thus triggering again policy violations.

Note that in release 1, policy changes are read during initialization. Hence, ensure that the controller is stopped, apply the desired changes, and then start the controller for the new policy to take effect.

3.1.3 Inspecting the Log

The command

```
$ print-refmon-logs
```

shows the reference monitor logs, including rules that have been checked and policy violations.

Note that the command can be combined like other Karaf commands to provide more flexibility:

```
$ print-refmon-logs | wc -l
$ print-refmon-logs | grep fwd
```

3.2 Programmer Guide

This enabler is not programmable.

4 Unit Tests

4.1 Unit Test 1

This test checks that the enabler runs. Executing `/opt/onos/bin/onos` from a terminal will connect to the CLI of ONOS:

```
$ /opt/onos/bin/onos
Logging in as karaf
Welcome to Open Network Operating System (ONOS)!
     ____  _  ______  ____
    / __ \/ |/ / __ \/ __/
   / /_/ /    / /_/ /\ \
   \____/_/|_/\____/___/


Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown ONOS.
```

```
onos>
```

With the command `apps` all running applications can be listed, namely:

```
onos> apps -s -a
```

The expected output is:

```
*  13 org.onosproject.mobility         1.3.0.SNAPSHOT Host mobility
application
*  16 org.onosproject.fwd              1.3.0.SNAPSHOT Reactive
forwarding application using flow subsystem
*  20 org.onosproject.ref.mon          1.3.0.SNAPSHOT ONOS access
control application
*  25 org.onosproject.proxyarp         1.3.0.SNAPSHOT Proxy ARP/NDP
application
*  30 org.onosproject.openflow         1.3.0.SNAPSHOT OpenFlow
protocol southbound providers
*  31 org.onosproject.drivers          1.3.0.SNAPSHOT Builtin device
drivers
```

The IDs in the first column and their order may vary at runtime. However, all those bundles must be listed. The `org.onosproject.ref.mon` line indicates that the reference monitor is running.

4.2 Unit Test 2

This test checks that the reference monitor functions according to the whitelist. It assumes an existing Mininet installation. Note that Mininet can run on a remote machine.

1. Stop ONOS.
2. Edit the policy file to allow traffic between hosts 1 and 2:

```
switch_dpid application_id mac_src mac_dst
00:00:00:00:00:00:00:01 org.onosproject.fwd 00:00:00:00:00:01
00:00:00:00:00:02
00:00:00:00:00:00:00:01 org.onosproject.fwd 00:00:00:00:00:02
00:00:00:00:00:01
```

3. Start ONOS and ensure that the reference monitor is running.
4. Start mininet with a simple linear topology, pointing it to the ONOS IP address:

```
$ sudo mn --mac --topo single,3 --switch ovs,protocols=OpenFlow10 --
controller=remote,ip=x.x.x.x,port=6633
```

5. Verify that the controller sees the switches:

```
onos> devices
id=of:0000000000000001, available=true, role=MASTER, type=SWITCH,
mfr=Nicira, Inc., hw=Open vSwitch, sw=2.5.0, serial=None,
protocol=OF_10, channelId=192.168.56.104:42664
```

6. Try to ping the two hosts:

```
mininet> h1 ping h2
```

The hosts should be able to ping each other and there should be no message from the reference monitor.

7. Now try to ping other hosts that are not in the whitelist:

```
mininet> h1 ping h3
```

The hosts should not be reachable because the reactive forwarding application (`org.onosproject.fwd`) is blocked from installing the rule. That means, ONOS should log and print out a policy violation.

5 Abbreviations

5G-PPP	5G Infrastructure Public Private Partnership
CLI	Command Line Interface
JDK	Java Development Kit
NBI	Northbound Interface
ONF	Open Networking Foundation
ONOS	Open Network Operating System
SBI	Southbound Interface
SDN	Software Defined Networking

6 References

[1] 5G-ENSURE Consortium. Deliverable D2.1: Use Cases. Available online: http://www.5gensure.eu/sites/default/files/Deliverables/5G-ENSURE_D2.1-UseCases.pdf. 2016.

[2] J. Anderson. Computer Security Technology Planning Study. Technical Report ESD-TR-73-51. US Air Force Electronic System Division, 1973.

[3] P. Berde, M. Geralo, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Conner, P. Radoslavov, W. Snow, and G. M. Parulkar. ONOS: Towards an open, distributed SDN OS. In *Proceedings of the 3rd SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*. ACM Press, 2014.

[4] F. Klaedtke, G. O. Karame, R. Bifulco, and H. Cui. Access control for SDN controllers. In *Proceedings of the 3rd SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*. ACM Press, 2014.

[5] F. Klaedtke, G. O. Karame, R. Bifulco, and H. Cui. Towards an access control scheme for accessing flows in SDN. In *Proceedings of the 1st IEEE Conference on Network Softwarization (NetSoft)*. IEEE Computer Society, 2015.

[6] B. Lantz, B. Heller, and N. McKeown. A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM Workshop on Hot Topics in Networks (HotNets).* ACM Press, 2010.

[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling innovation in campus networks. SIGCOMM Computer Communication Review, 38(2):69–74, 2008.

[8] Mininet. An instant virtual network on your laptop. Available: http://mininet.org/.

[9] ONOS. A new carrier-grade SDN network operating system designed for high availability, performance, scale-out. Available online: http://onosproject.org/.

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

[11] J. H. Saltzer. Protection and the control of information sharing in multics. Comm. ACM, 17(7):388—402, 1974.

Deliverable D3.4
5G-PPP Security Enablers Documentation (v1.0)
Enabler Policy Compliance Checker

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	30.09.2016	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	NEC: Felix Klaedtke	

Contents

1 Introduction

Networks comprise multiple components, e.g., endhosts and switches, and a controller in case of an SDN network. Policies specify how these components must and must not behave. There is a wide spectrum of policies, targeting various aspects of a network like correctness, performance, quality of service, reliability, and security. Detecting noncompliant behavior of components with respect to a given policy is an important task to ensure the correct and safe operation of a network. In particular, in a network in which physical and virtual components are managed by different tenants (e.g., a network with multiple network or service providers), the detection of noncompliant behavior of a component is a major concern for the network operator. It helps the operator to protect the network, e.g., against misbehaving components and misconfigurations.

The scope of this enabler is to check the interactions between network components against a given policy, which specifies how components must and must not interact with each other. It is however not in the scope of the enabler to perform corrective actions in case the interactions between the network components are not policy compliant. The interactions can be checked online, i.e., while the components are running, or offline during an audit. These checks are performed by the *compliance checker*, which is described in this manual. The compliance checker supports a rich formally defined specification language for expressing a large variety of policies.

- One use case of this enabler is to increase the resilience of the network. For this, the network administrator deploys the compliance checker into the network. Furthermore, he instruments some of the network components such that they send messages describing their performed actions to the compliance checker. A policy specifies how these components should behave, e.g., how they must and must not react to certain network events. Whenever the policy is violated, the network administrator is notified by the compliance checker and can take countermeasures against this noncompliant behavior. Policy violations can for example be caused by a wrongly configured component or a malicious component. Cf. the use case 5.2 of the 5G-ENSURE project deliverable D2.1 [1].
- Another use case is where the network administrator deploys a new component in the network. Similar to the first use case, the performed actions of this new component are checked against a policy and noncompliant behavior is detected and reported to the network administrator. For instance, a new component might behave not as intended because of a software bug or a misconfiguration, which are then detected by the compliance checker. Note that buggy software and misconfigurations might make the network vulnerable to attacks. Cf. the use case 5.4 of the 5G-ENSURE project deliverable D2.1 [1].

The manual is organized as follows. Section 2 describes how the compliance checker is installed and administrated. Section 3 describes how the compliance checker is used. Section 4 provides some basics tests for the compliance checker. Abbreviations are listed in Section 5.

2 Installation and Administration Guide

The compliance checker is a single standalone executable, which is run and configured over the command line. The name of the executable is `runverif`, which is available from the software repository of the 5G-ENSURE project. In the following, we describe how `runverif` is installed and configured.

2.1 System Requirements

The provided executable `runverif` was compiled with Debian GNU/Linux 8.1 (Jessie), with the Linux kernel version 3.16.0.-4-586, running on an i686 architecture (little endian) with CPU op-mode 32-bit. It should therefore be executable on most computers with an i686 compatible processor that are running the Linux operating system, as provided by any recent Linux distribution (Debian, Ubuntu, RedHat, etc.). The user that is executing `runverif` needs the permissions of doing so. Furthermore, the user needs permissions to create and write to temporary files (log files in the `/tmp` directory), and to create, write to, and read from UDP sockets.

Furthermore, it is remarked here that actions of network components are timestamped. The compliance checker assumes that these timestamps are accurate. In case the Network Time Protocol (NTP) [7, 8] is used for clock synchronization between the network components, it is favorable when network components are in a low NTP stratum, e.g., the stratum 3 or smaller.

2.2 Enabler Configuration

Table 1 `runverif`'s command-line options.

Option	Description	Default
`-spec FILENAME`	File containing the formula to be monitored	
`-msgs FILENAME`	File specifying how to interpret the messages	
`-comp FILENAME`	File listing the monitored components	
`[-prefix STRING]`	Shortcut/alternative for the options `–spec`, `–msgs`, and `–comp` (the provided string is automatically extended with the suffixes `.spec`, `.msgs`, and `.comp`; the resulting filenames are used when the corresponding option is not provided)	
`[-input FILENAME]`	File with the logged messages (offline use)	
`[-inport=NUMBER]`	UDP port on which the messages are received (online use)	`50010`
`[-outport=NUMBER]`	UDP port to which the verdicts are sent (currently disabled)	`50011`
`[-cores=NUMBER]`	Number of used CPU cores	`1`
`[-log FILENAME]`	Log file for output	`/tmp/runverif.log`
`[-loglevel=NUMBER]`	Logging level (≥0: errors and verdicts, ≥1: warnings, ≥2: info, ≥3: everything)	`1`
`[-quiet]`	Do not print log messages; only log into a file	
`[-violations]`	Only report violations of the specification	
`[-version]`	Print `runverif`'s version and exit	

The `runverif`'s command-line options are listed in Table 1, including their default values. The options in brackets are optional. Note that the command-line option `–help` lists these options on the command line. The configuration files (given through the options `–spec`, `–msgs`, and `–comp`, or the `–prefix` option) are described in the enabler's user guide, see Section 3.

2.3 Enabler Installation

We suggest to place the executable `runverif` into the `/usr/bin` directory of the Linux operating system, with the permission for any user to execute it. Note that the network components (e.g., switches, SDN controller, and network components) need to be instrumented in such a way that they send their actions to the compliance checker. Alternatively, in the offline use of the compliance checker, the components can also log their actions. The instrumentation is described in the user guide, see Section 3.

2.4 Troubleshooting

If an error occurs, `runverif` prints or logs the error. The default log file is `/tmp/runverif.log`. See the command-line options in Section 2.2 to change this. In case of a fatal error, `runverif` also terminates. For example, if the specification configuration file (command-line option `–spec`) cannot be opened, `runverif` reports the error

```
ERROR: 2016/07/28 02:47:33 Failed to open file foo.spec.
```

and terminates. Analogously, `runverif` reports syntax errors and points to the location (line and column), where it failed to parse the specification. Another error is when `runverif` could not create the UDP socket for receiving messages. In this case, it reports

```
ERROR: 2016/07/28 02:48:24 Failed to create UDP socket (<nil>:30).
```

An example of a nonfatal error is when `runverif` receives a message that cannot be parsed. It reports

```
ERROR: 2016/07/28 02:58:14 Failed to parse the message "Hello, world!".
```

but it does not terminate. Instead, `runverif` ignores this message and continues with processing the next received message.

Warnings are also printed or logged, provided that the logging level is equal to or greater than 1 (command-line option `–loglevel`). For example, if `runverif` receives a message that it cannot interpret according to the provided configuration file for interpreting messages (command-line option `–msgs`), `runverif` reports

```
WARNING: 2016/07/28 02:58:24 Ignoring the message "...".
```

Either `runverif` accidentally received a message that could not be interpreted or the configuration file for interpreting messages (command-line option `–msgs`) does, e.g., not cover all messages correctly. In general, warnings often hint towards a problem.

Note that with a logging level equal to or greater than 2 (command-line option `–loglevel`), `runverif` prints or logs additional information, which might be helpful to better understand what is happening. For example, when starting `runverif` with `–loglevel=2`, `runverif` reports

```
INFO: 2016/07/28 03:08:58 Automatic garbage collector is turned on.
INFO: 2016/07/28 03:08:58 1 out of 4 CPU core(s) is used.
```

```
    3 components (ovs-switchd_s1, ovs-switchd_s2, ovs-switch_s3) are
    monitored.
    ...
  INFO: 2016/07/28 03:08:58 UDP socket (port 50010) is open.
```

Another source of problem can be that `runverif` does not receive messages sent from the network components, although `runverif` successfully created the UDP socket. First, note that UDP does not guarantee that sent messages are indeed received, i.e., messages can be lost. Furthermore, note that messages can be received in any order. `runverif` soundly operates in the presence of message loss and the reordering of messages. However, if no messages from a network component are received by `runverif` over the UDP socket, one should check whether the network component actually sends the messages to the correct address and port.

3 User and Programmer Guide

3.1 User Guide

3.1.1 Overview

The enabler is “SDN generic,” i.e., the compliance checker is a network component at the control plane of an SDN network. It runs separately to the other control plane components (e.g., controller and network applications). The interaction with the other components is “one way.” In particular, in its online use, it only receives over a socket messages from the other network components, both at the data plane and control plane. The compliance checker does not directly impact the network traffic and the network configuration. Nevertheless, it might indirectly impact the network configuration, since it outputs verdicts on which a network administrator, e.g., can take corrective actions.

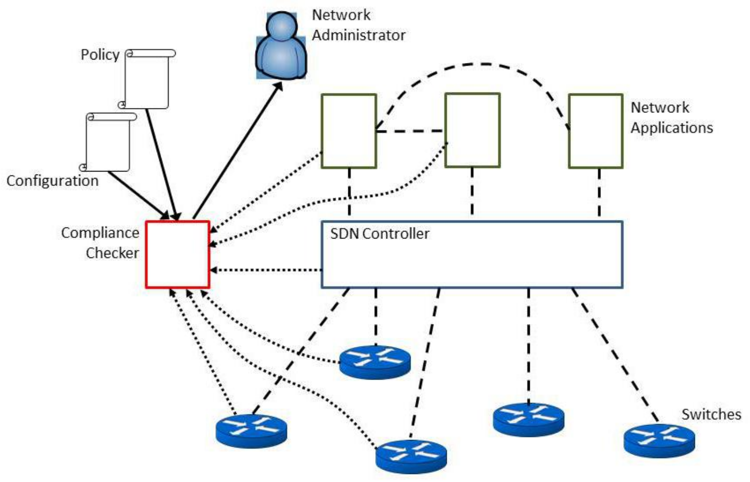


Figure 1 Illustration of the use of the compliance checker.

Figure 1 illustrates the online use of the enabler in an SDN network. Here, the network controller interacts via a southbound interface (SBI) with the switches, e.g., via the OpenFlow protocol [6, 10]. An OpenFlow message can, e.g., be initiated by a network application, which interacts with the controller via some northbound interface (NBI). It might also interact with other network applications. The main component of the enabler is the compliance checker. It receives messages from the components that describe their actions, e.g., the sending or the receiving of a message. The actions are checked against a given policy. In case of a policy violation, the network administrator is notfied, i.e., the compliance checker outputs a verdict.

Note that the enabler can also be used offline. In this case, the components log their actions. The log is then analyzed later by the compliance checker and policy violations are output.

3.1.2 Instrumentation

The monitored system components must be instrumented in that they send messages to the compliance checker, informing it about their performed actions. Alternatively, these messages can also be logged and checker later offline. The instrumentation is policy specific. Obviously, the compliance checker must only be informed about the performed actions that are policy relevant. The instrumentation is also component specific. As a proof of concept we have instrumented the SDN controller ONOS [5, 9] and the software switch OVS [11, 12] for the OpenFlow protocol. That is, these components inform the compliance checker whenever they send or receive certain OpenFlow messages. Similar instrumentations, e.g., for ONOS for its NBIs and for network applications that interact with ONOS are also possible.

In the following, we describe how a network component informs the compliance checker about its performed actions. In particular, we describe the message format, and the online and offline use of the compliance checker.

3.1.2.1 Message Format

The messages that are sent to the compliance checker and describe the monitored components' performed actions must be timestamped. The timestamp of a message describes when the action was carried out. The enabler assumes the timestamp in Unix time (with integral part and a possibly fractional part). Furthermore, the message must specify the sender and a sequence number. The sequence number is the number of messages that the sender has sent so far to the compliance checker. With the sequence numbers the compliance checker can determine whether it has received all messages up to a given time. Finally, the message must describe the performed action. Overall, the fields of a message are as follows.

timestamp	sender identifier	sequence number	description of performed action

As an example consider the following message.

```
1438868431.071476@[ovs-switchd_s1 (S1)] (9): OFPT_ECHO_REPLY (OF1.3)
  (xid=0x0): 0 bytes of payload
```

The number before the @ letter is the timestamp of the message in Unix time (August 6, 2015 13:40:31 GMT with additional precision beyond seconds). The sender information is given in the brackets. Here, the sender is the OpenVSwitch (OVS) daemon named s1. The sequence number, given in parentheses after the sender information, is 9, i.e., this is the ninth message of the OVS daemon has sent to the compliance checker. Finally, after the double colon, the performed action is described. Here, the daemon handles the OpenFlow message OFPT_ECHO_REPLY.

We remark that the compliance checker assumes that (1) messages are not tampered with, and (2) components correctly report their actions and do not send bogus messages. The first assumption can be easily discharged by adding information to each message such as a cryptographic hash value, which is checked by the compliance checker when receiving the message. To discharge the second assumption additional mechanisms need to be deployed. For example, a message contains a timestamp when the action is performed. To ensure the correctness of the timestamp a trustworthy clock must be used that signs the timestamp. The compliance checker can then check the validity of a timestamp by checking its signature. For ensuring that a reported action was performed or a non-reported action did not take place, one could deploy additional monitors or "traps" that check this. Such monitors or "traps" are, however, action and component dependent and not in the scope of the current version of the enabler.

3.1.2.2 Online and Offline Use

The compliance checker can be used online and offline. When used online, the compliance checker creates and UDP socket (command-line option `-inport`). It processes a message (received in form of a UDP packet) as soon as it receives the message on the specified port. When used offline, the compliance checker reads the messages from a log file (command-line option `-input`). The messages are processed in the order as they are given in the file.

In an online use, the monitored components must be instrumented in such a way that they send their messages to the UDP port of the compliance checker. In an offline use, the monitored components must log their messages. This can be either done in a central log file or in separate log files, which must then be collected and merged later.

We note that although messages are timestamped, the messages do not need to be ordered with respect to their timestamps. For instance, in the offline case, the log file does not have to be ordered ascendingly according to the messages' timestamps. However, we also note that the ordering might have an impact of the compliance checker's performance.

3.1.3 Configuration Files

In the following, we provide details about the configuration files of the compliance checker. Comments in the configuration files are marked with the # symbol, i.e., the letters after a # symbol on the line are ignored.

3.1.3.1 Components

This configuration file (command-line option `-comp`) lists the names of the components from which the compliance checker accepts to receive messages, which should be interpreted and processed. The name of each component is listed on a single line of the configuration file. Note that if the compliance checker receives a message from a component that is not listed in the configuration file then the message is silently ignored by the compliance checker.

3.1.3.2 Messages

For interpreting the received messages, a configuration file (command-line option `-msgs`) must be provided to the compliance checker. This configuration file consists of rules with regular-expression matching to identify the performed action and to extract relevant data values. The first matching rule determines the interpretation of the message. One can think of these rules as an if-then-else program.

The configuration file comprises components and match rules. It is of the form

`[`*component match* 1`]` **:**
    match rule 1.1

```
    match rule 1.2
    ...
    match rule 1.n1
    ;;
 ...
 [component match m]:
    match rule m.1
    match rule m.2
    ...
    match rule m.n1
    ;;
```

Each *component match* is a regular expression. The sender identifier field of each received message is matched against these regular expressions, starting from the top. For the first successful match, the received message is matched against the match rules in the respective component, again starting from the top.

An example of a match rule is the following rule.

```
[component matches "S(?P<switch>[0-9]+)",
 event matches "OFPT_FLOW_MOD",
 event matches "\(xid=0x(?P<xid>[0-9abcdef]+)\)"]
==>
{flow_mod(<switch>, <xid>)}
```

A message satisfies the antecedents (given in the brackets before the arrow symbol) of this rule if the message successfully matches the three given regular expressions. First, the sender information of the message must contain a string of the form `S`*number*. More precisely, the part in parenthesis of the sender information of the message must contain such a string. Furthermore, in case of a successful match, the string *number* is bound to the register `<switch>`. Second, the string `OFPT_FLOW_MOD` must be contained in the message's description of the performed action. Third, the description of the performed action must also contain a string of the form `(xid=0x`*hexnumber*), where *hexnumber* is bound to the register `<xid>`. Overall, the interpretation is that the performed action is the receiving of an OpenFlow message by the switch `S`*number* of the type OFPT_FLOW_MOD with the xid *hexnumber*. This is, the succedent of the rule (the set after the arrow symbol) provides the interpretation of the predicate symbol `flow_mod` at the message's timestamp. Namely, it is the singleton relation containing the pair (*number*, *hexnumber*). Note that the specified policy, which we describe next, can refer to predicates like `flow_mod` and bound values like `<xid>` and `<switch>`.

The succedent of a match rule can also be `ignore` instead of a set that determines the interpretation of the predicates of the message's timestamp. In this case, the message is ignored by the compliance checker. For example, the component

```
[ ]:
  [ ] ==> ignore
  ;;
```

can be useful at the end of a configuration file for interpreting the messages. Such a component has the effect that messages that did not match against any of the previously specified rules are ignored.

3.1.3.3 Specification

This configuration file (command-line option `–spec`) contains the policy to be monitored. Policies are expressed as formulas of a temporal logic. More precisely, a variant of the real-time logic MTL is used with a point-based semantics and a dense time domain. Furthermore, a freeze quantifier accounts for data values. We refer to Alur and Henzinger [2], Baier and Katoen [3], and Basin et al. [4] for theoretical background, in particular, for the underlying temporal model and the semantics of the specification language.

The core grammar of the policy specification language is given by the following grammar.

```
spec   ::=   TRUE
         |   p(x1, …, xn)
         |   (NOT spec)
         |   (spec OR spec)
         |   (spec SINCE[a,b] spec)
         |   (spec UNTIL[a,b] spec)
         |   (FREEZE x1[r1], …, xn[rn]. spec)
```

Here, `p` is a predicate symbol, *x1*, …, *xn* range over variables, *a* and *b* are nonnegative integers, and *r1*, …, *rn* range over data registers.

- `TRUE` specifies the trivial policy that any behavior satisfies.
- `p(`*x1*`, …, `*xn*`)` specifies the policy that a behavior satisfies at time *t* whenever in (*x1*, …, *xn*) are in the relation that interprets the predicate symbol `p` at time *t*.
- `(NOT` *spec*`)` specifies the policy that a behavior satisfies whenever it does not satisfy *spec*.
- `(`*spec1* `OR` *spec2*`)` specifies the policy that a behavior satisfies whenever it satisfies *spec1* or *spec2*.
- `(`*spec1* `SINCE[`*a*`,`*b*`]` *spec2*`)` specifies the policy that a behavior satisfies at time *t* whenever the behavior satisfies at some time *s≤t*, with *a≤t-s≤b*, the policy *spec2*, and since *s*, the behavior satisfies the policy *spec2*.
- `(`*spec1* `UNTIL[`*a*`,`*b*`]` *spec2*`)` specifies the policy that a behavior satisfies at time *t* whenever the behavior satisfies at some time *s≥t*, with *a≤s-t≤b*, the policy *spec2*, and until *s*, the behavior satisfies the policy *spec2*.
- `(FREEZE` *x1*`[`*r1*`], …,` *xn*`[`*rn*`].` *spec*`)` specifies the policy that a behavior satisfies at time *t* whenever the behavior satisfies the policy *spec*, where *x1* to *xn* are assigned to the register values *x1* to *xn* at time *t*.

Various additional syntactic sugar is defined. First, there are predefined predicate symbols for comparison, which are written infix: `=`, `/=`, `<=`, `>=` for comparing integers and `==` and `=/=` for comparing strings. Second, the standard Boolean connectives `AND` (conjunction) and `IMPLIES` (implication) are defined. For example, `(`*spec1* `AND` *spec2*`)` abbreviates `(NOT ((NOT` *spec1*`) OR (NOT` *spec2*`)))`. As expected, `(`*spec1* `AND` *spec2*`)` specifies the policy that a behavior satisfies whenever the behavior satisfies the policy *spec1* and the policy *spec2*. `(`*spec1* `IMPLIES` *spec2*`)` is syntactic sugar for `((NOT` *spec1*`) OR` *spec2*`)`; its meaning is as expected. Third, the unary temporal connectives `EVENTUALLY`, `ALWAYS`, `ONCE`, and `HISTORICALLY` are derived from the binary temporal connectives `UNTIL` and `SINCE`. For example, `(ONCE[`*a*`,`*b*`]` *spec*`)` abbreviates `(TRUE SINCE[`*a*`,`*b*`]` *spec*`)`. A behavior satisfies the policy

`(ONCE[`*a*`,`*b*`]` *spec*`)` at time *t* whenever the behavior satisfies *spec* at some time *s*, with *a≤t-s≤b*. Fourth, register names can be omitted if they are identical to the corresponding variable names. For example, `(FREEZE` *x*`.` *spec*`)` abbreviates `(FREEZE` *x*`[`*x*`] .` *spec*`)`.

Furthermore, note that the connectives are assigned to different binding strengths, which allow one to omit parenthesis. We use the standard conventions here. For example, `NOT` binds stronger than `OR`. By this convention, `NOT` *spec1* `OR` *spec2* abbreviates `((NOT` *spec1*`) OR` *spec2*`)`. We also allow open intervals and half-open intervals as metric temporal constrains like `(`*a*`,`*b*`)` and `[`*a*`,`*b*`)`, for integers *a* and *b* with 0≤*a*<*b*. In these two cases *b* could also be infinity (denoted by the symbol `*`) to specify an unbounded interval. The interval `[0,*)`, which does not impose any metric temporal constraint, can be dropped. A time unit (e.g., `ms`, `s`, and `m`) can be attached to the numbers. If no time unit is given the numbers are interpreted as seconds.

For example, consider the following the formula.

```
  FREEZE id, msg. send(id, msg) IMPLIES EVENTUALY[0,1ms] receive(id, msg)
```

It specifies the policy that whenever a message `msg` with identifier `id` is sent it is eventually received. Furthermore, it is required that the message is received within one millisecond.

We point out that there is an implicit temporal connective `ALWAYS` in front of any specification, i.e., the specified policy must hold at every point in time. It is required that formulas are closed, i.e., they do not contain free variables. It is also required that any variable is bound at most once by a freeze quantifier.

3.2 Programmer Guide

This enabler is not programmable.

4 Unit Tests

4.1 Information about Tests

The following simple tests check whether `runverif` is installed properly. For the provided input, we describe how `runverif` should behave and `runverif`'s expected output. In the following, we assume that `runverif` is installed in the `/usr/bin` directory. Furthermore, we assume that the configuration files are in the user's directory `~/runverif`. Note that the configuration files used in these test are provided with the enabler's software from the software repository of the 5G-ENSURE project.

4.2 Unit Test 1

This simple test checks that `runverif` can be executed by the user. Executing `/usr/bin/runverif –version` from a terminal should print out `runverif (version 0)`. The version number might differ, in particular, it might be greater than 0.

4.3 Unit Test 2

In this test, `runverif` checks a simple policy on a small log file. The log file is `unittest_nontemporal.log` and contains two messages. The policy is given in the configuration file `unittest_nontemporal.spec`. The formula of this file expresses that a message with an identifier 0 must not be sent. The other configuration files for this test are `unittest_nontemporal.comp` and `unittest_nontemporal.msgs`.

Executing `runverif` with the options `–prefix unittest_nontemporal` and `–input unittest_nontemporal.log` should result in the following output.

```
VERDICT: @1.000000000: true
VERDICT: @2.000000000: false
```

This should be also logged. See `/tmp/runverif.log`.

4.4 Unit Test 3

We use the same configuration files as in the previous test (Section 4.3). However, instead of letting `runverif` to read the messages from a log file, we let `runverif` to receive the messages from the default UDP socket (50010). To this end, we start `runverif` in one terminal:

```
/usr/bin/runverif –prefix ~/runverif/unittest_nontemporal
```

From another terminal we send the following message to the UDP port 50010 by using the standard Unix command `nc`:

```
echo -n "1@[C] (1): send(1, helloworld)" | nc –u –q0 localhost 50010
```

As in the previous test, no policy violation should be reported by `runverif`, i.e., runverif should output

```
VERDICT: @1.000000000: true
```

When sending the second message

```
echo -n "2@[C] (2): send(0, helloworld)" | nc –u –q0 localhost 50010
```

`runverif` should report a policy violation. This is, it should print out

```
VERDICT: @2.000000000: false
```

These verdicts should also be logged in `/tmp/runverif.log`.

When sending the UDP message

```
echo -n "3@[C] (3): receive(0, helloworld)" | nc –u –q0 localhost 50010
```

`runverif` should print out and log an error that it failed to match the message.

4.5 Unit Test 4

This test is similar to the previous two tests. However, the specification is slightly more complex, and more and different kinds of messages are processed by `runverif`. In particular, two components are monitored, C and D. C is sending messages and D is receiving messages. We check whether the messages sent by C are received, within one second, by D. The configuration files are `unittest_temporal.comp`, `unittest_temporal.msgs`, and `unittest_temporal.spec`. The actions carried out by the two components (i.e., the messages that are sent by C and D to `runverif`) are listed in the file `unittest_temporal.log`.

The expected output of this test is

```
VERDICT: @1.800000000: true
VERDICT: @1.000000000: true
VERDICT: @3.000000000: true
VERDICT: @1.100000000: false
VERDICT: @2.000000000: false
```

5 Abbreviations

5G PPP	5G Infrastructure Public Private Partnership
MTL	Metric Temporal Logic
NBI	Northbound Interface
NTP	Network Time Protocol
ONF	Open Networking Foundation
ONOS	Open Network Operating System
OVS	OpenVSwitch
SBI	Southbound Interface
SDN	Software Defined Networking

6 References

[1] 5G-ENSURE Consortium. Deliverable D2.1: Use Cases. Available online: http://www.5gensure.eu/sites/default/files/Deliverables/5G-ENSURE_D2.1-UseCases.pdf. 2016.

[2] R. Alur and T. A. Henzinger. Logics and models of real time: A survey. In Proceedings of the 1991 REX Workshop on Real Time: Theory in Practice, volume 600 of Lect. Notes Comput. Sci., pages 74–106. Springer, 1992.

[3] C. Baier and J.-P. Katoen. Principles of model checking. MIT Press, 2008.

[4] D. Basin, F. Klaedtke, and E. Zalinescu. Monitoring metric first-order temporal properties. J. ACM, 62(2): 15, 2015.

[5] P. Berde, M. Geralo, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Conner, P. Radoslavov, W. Snow, and G. M. Parulkar. ONOS: Towards an open, distributed SDN OS. In *Proceedings of the 3rd SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*. ACM Press, 2014.

[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling innovation in campus networks. SIGCOMM Computer Communication Review, 38(2):69–74, 2008.

[7] D. L. Mills. Improved algorithms for synchronizing computer network clocks. IEEE/ACM Trans. Netw., 3(3):245–254, 1995.

[8] Network Time Protocol (NTP). http://www.ntp.org/.

[9] ONOS. A new carrier-grade SDN network operating system designed for high availability, performance, scale-out. Available online: http://onosproject.org/.

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

[11] OpenVSwitch (OVS). A production quality, multilayer virtual switch. Available online: http://openvswitch.org/.

[12] B. Pfaff, J. Petit, T. Koponen, K. Amidon, M. Casado, and S. Shenker. Extending networking into the virtualization layer. In *Proceedings of the 8th ACM Workshop on Hot Topics in Networks (HotNets)*. ACM Press, 2009.

Deliverable D3.4
5G-PPP Security Enablers Documentation (v1.0)
Enabler Bootstrapping Trust

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	30.09.2016	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	SICS: Nicolae Paladi	

Contents

1 Introduction

The physical infrastructure supporting 5G networks will be multiplexed among multiple tenants with own environments, similar to a cloud infrastructure model. In this model, SDN allows tenants to configure complex topologies with rich network functionality, managed by a network controller. The availability of a global view of the data plane enables advanced controller capabilities – from pre-calculating optimized traffic routing, to managing software applications that replace hardware middleboxes. 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 – may distort the controller's view of the data plane and influence the network-wide routing policies. Virtual switches are security sensitive elements in SDN deployments. They run on commodity operating systems (OS) and are often assigned the same trust level and privileges as physical switches – 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 switches. For example, their configuration can be modified to disobey the protocol, breach network isolation and reroute traffic to a malicious destination or hijack other network edge elements through lateral attacks. Such risks are accentuated by the extensive control a cloud provider has over the compute, storage and networking infrastructure – this requires long-term, absolute trust from the tenants in the integrity of the cloud provider's security perimeter.

The aim of this enabler is to attest the integrity virtual switch components and prevent virtual switch impersonation attacks by enrolling into the topology *only* attested switches, as well as storing the authentication keys in a trusted execution environment. This is expressed in two use cases described in [2], namely *attest integrity of a virtual switch in the SDN deployment* and *enroll attested virtual switch into SDN deployment*.

The final version of this enabler is due to be delivered in Release 2 -- including a comprehensive implementation of *integrity attestation of virtual network components* -- according to the description in Deliverable 3.1 [1] and Deliverable 3.2 [2]. The current manual addresses the functionality implemented in Release 1, namely, prototypes of *enrollment of integrity attested virtual switches* and *provisioning of switch-specific credentials to trusted execution environments (TEEs)*.

2 Installation and Administration Guide

Release 1 of the Bootstrapping Trust enabler consists of a set of libraries and configuration scripts that enable basic integrity verification and enrollment functionality. Given the reliance on software emulation (to emulate the functionality of TEEs, it is essential to note that the current release of the enabler *does not* provide any security guarantees.

The current enabler release contains the following elements:

1. A binary of the Open vSwitch containing modification in the module <lib/stream-ssl.c>
2. Implementation of the TEE for provisioning of switch-specific credentials (further referred to as "OVS TEE").
3. Implementation of the TEE remote attestation functionality, as an extension of the Ryu controller.
4. Installation and configuration scripts (further referred to as "Controllerapp").

Figure 1 illustrates the interaction of the "Bootstrapping Trust" enabler with other relevant components. Further, the two use cases introduced in Deliverable 3.2 and prototyped in this release can be observed in the same illustration: the integrity of the virtual switch is attested by the OVS TEE, using the measurements made by the Linux *Integrity Measurement Architecture* (IMA) subsystem [5] and the expected values (currently configured in the code of the OVS TEE). The integrity of the OVS TEE is in turn attested by the attestation application residing on the controller.

The IMA is an open source trusted component included in the Linux kernel, which aims to detect file modifications (either malicious or accidental), appraise a file's measurement against an expected value, and enforce local file integrity.

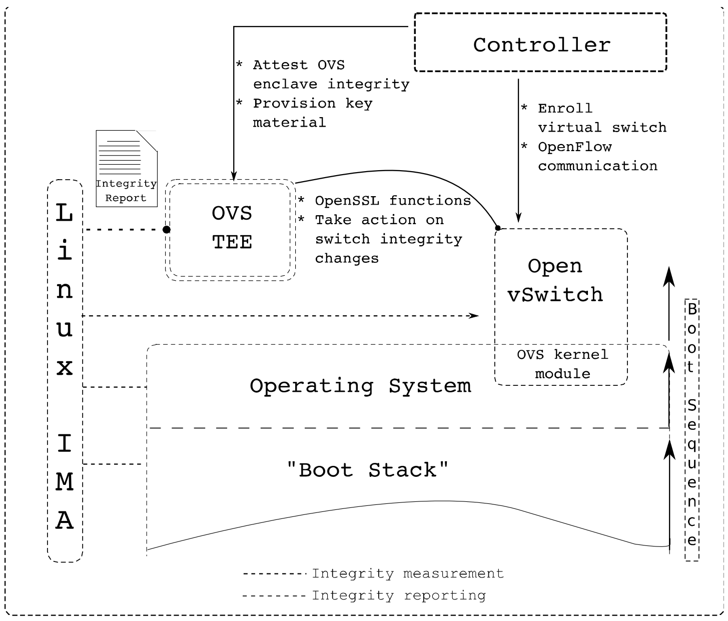


Figure 1 Interaction of the Bootstrapping Trust enabler with the other components

2.1 System Requirements

Following the system architecture presented in Figure 1, we next describe the system requirements for the current enabler. Note that the placement of the controller, relative to the rest of the aforementioned elements of the enabler (i.e. on the same host or on a different host), is implementation-specific and not relevant for the purpose of the installation. The requirements for the current enabler depend in turn on the requirements of the enabler components:

1. Ryu SDN controller, extended with an implementation of the TEE remote attestation functionality. It manages the SDN network and interacts with the Open vSwitch TEE in order to provision switch-specific credentials.
2. Modified Open vSwitch binary, running on a Linux platform with the IMA enabled. Enabling IMA is necessary in order to measure specific components of the Open vSwitch on the platform and later attest their integrity, by comparing the measurements with the expected values.
3. Emulation of Intel Software Guard Extensions (SGX) [3], an extension of the x86 Instruction Set Architecture. SGX introduces support for TEEs with remote attestation capabilities, which – once instantiated – cannot be modified by the underlying OS. The emulation is necessary to address the lack of hardware or firmware support for SGX on the platform, and is implemented using Open SGX [4]. Our current intent is to replace, in Release 2 of the enabler, the OpenSGX emulator with the recently released SGX SDK [9].

The above components require the presence of a Linux-based OS – such as Ubuntu 14.10 or above – for installation and operation. In addition to the above components, the underlying platform must provide support for the Linux Integrity Measurement Architecture [5]. According to its manual, this kernel integrity subsystem detects if files have been accidentally or maliciously altered, both remotely and locally, appraises a file's measurement against expected values stored as an extended attribute, and enforces local file integrity [10].

In addition to the standard library dependencies for the Ryu SDN controller and Open vSwitch, of the OpenSGX emulator requires the presence of the `<qemu>` and `<libelf-dev>` libraries (and underlying dependencies).

2.2 Enabler Installation and Setup

The enabler configuration consists of three main steps: OS-level configuration, emulation environment configuration, SDN configuration. We describe the installation and configuration procedures in the below sequence. We assume the host operating system is Ubuntu 14.10 and above.

2.2.1 OS-level Configuration

Integrity Measurement Architecture must be enabled and configured in order to obtain the initial measurement of the Open vSwitch binary. In Ubuntu 14.10 and above IMA is enabled in the Kernel by default. To verify that IMA is enabled, execute the following commands:

```
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 measurements considered relevant for this enabler, on a Ubuntu 15.04, Linux Kernel 3.19.0-51-generic are:

- `/lib/modules/3.19.0-51-generic/kernel/net/openvswitch/openvswitch.ko`
- `/etc/default/openvswitch-switch`
- `/lib/systemd/system/openvswitch-switch.service`
- `/lib/systemd/system/openvswitch-nonetwork.service`
- `/usr/share/openvswitch/scripts/ifupdown.sh`
- `/etc/default/openvswitch-switch`
- `/usr/share/openvswitch/scripts/ovs-ctl`
- `/usr/share/openvswitch/scripts/ovs-lib`
- `/usr/share/openvswitch/vswitch.ovsschema`
- `/var/lib/openvswitch/conf.db`
- `/usr/sbin/ovs-vswitchd`

- `/etc/openvswitch/system-id.conf`

The measurements collected by IMA at boot time are found in
`/sys/kernel/security/ima/ascii_runtime_measurements`
For release 1 of the enabler, only one measurement – namely that of the *Open vSwitch kernel module*[1] is attested by the TEE code. This feature is not configurable. Release 2 of the enabler will include a more comprehensive attestation.

2.2.2 Emulation environment configuration

The OpenSGX emulation environment is installed and configured according to the manual in [4]. Assuming the prerequisites are installed, the following commands will build the OpenSGX environment:

```
cd qemu
./configure-arch
make -j $(nproc)
cd ..
make -C libsgx
```

Next, from the enabler repository check out the file 'stream_ssl_enclave.c', place it into the directory '`opensgx/user/demo/`', and compile the code from the root `opensgx` directory:

```
make -C user
```

2.2.3 SDN configuration

In this section we describe the installation of the Open vSwitch binary modified to interact with the TEE emulated in Open SGX, as well as the configuration of the Ryu controller running a custom application that attests the integrity of the TEE and provisions key switch-specific key material.

2.2.3.1 Open vSwitch installation and configuration

To install and configure the modified open vSwitch binary, download the respective package from the enabler repository and run:

```
dpkg -i openvswitch-tswitch
```

All other configuration commands are identical to the Open vSwitch installation manual [5].

Next, launch the modified Open vSwitch instance on the OVS host, with the desired configuration. For the current enabler, the configuration options described in the Open vSwitch manual [7] are sufficient. To launch and configure the modified OVS instance, run the configure-ovs.sh script (included in the enabler distribution):

```
sudo sh configure-ovs.sh
```

Once the OVS instance has been configured the one must launch the custom TEE application. As mentioned above, due to the insufficient software support for SGX in Linux at the moment, the TEE application relies

[1] `/lib/modules/3.19.0-51-generic/kernel/net/openvswitch/openvswitch.ko`

on an emulator. The emulator code, along with additional source code part of the enabler implementation (files "`steam-ssl.c`" and "`controllerapp.c`"), is provided in the enabler distribution.

To launch the OVS TEE, execute the following:

```
cd ~./opensgx
./opensgx -c user/demo/stream-ssl.c (this step creates the file "stream-ssl.sgx")
./opensgx -s user/demo/stream-ssl.sgx --key sign.key (this step creates the file "stream-ssl.conf")
sudo ./opensgx -i user/demo/stream-ssl.sgx user/demo/stream-ssl.conf
```

The commands above *compile* the OVS TEE application, *sign it* using the emulator-generated key[2], and *launch it* in the emulated environment.

2.2.3.2 Controller installation and configuration

To generate the PKI material for the controller and the virtual switch, run the 'gen-pki.sh' script on the controller host:

```
sudo sh gen-pki.sh
```

The enabler relies on the Ryu SDN controller framework [6]. To launch the custom controller application required by the enabler, run:

```
sudo ryu-manager --ctl-privkey ctl-privkey.pem --ctl-cert ctl-cert.pem --ca-certs cacert.pem --verbose ryu/ryu/app/attest_and_switch.py
```

Launching the 'attest_and_switch.py' controller application will also launch the "controllerapp.c" program. Following the SGX attestation protocol, this program attests the integrity of the OVS TEE application (which in turn verifies the integrity of the OVS kernel module).

2.3 Troubleshooting

Both of the SGX-related applications executing in the emulated environment (`stream-ssl.c`, `controllerapp.c`) may occasionally crash due to unexpected errors or bugs that have not been found yet. In that case, they should just be restarted, on their respective hosts:

```
sudo ./opensgx -i user/demo/stream-ssl.sgx user/demo/stream-ssl.conf
sudo ./opensgx -i user/demo/controllerapp.sgx user/demo/controllerapp.conf
```

3 User and Programmer Guide

The "Bootstrapping Trust" enabler is a meant to strengthen the security of the SDN infrastructure itself, rather than of any particular instance of the software defined network implemented in that infrastructure. It adds additional, optional integrity verification mechanisms to the infrastructure. Such mechanisms allow *detecting* changes in the integrity of the virtual switches (compared to a certain "known good value") and *preventing* integrating such switches into the *data plane* view of the controller. Obtaining the "known good value" necessary for the integrity evaluation of the switch is implementation-specific and is out of the scope of the current enabler. Note that the enabler does not provide any guarantees regarding the security

[2] For simplicity, the key is supplied along with the code. A new key can be generated by running: './opensgx –k'

properties of the attested switch implementations – this can, and should be done through the security testing of the product. Rather, the "Bootstrapping Trust" enabler merely allows verifying that the *integrity* of the switches has not been violated.

3.1 User Guide

3.1.1 Overview

In a typical usage scenario, the installed, configured and launched "Bootstrapping Trust" enabler does not require any recurrent user interaction. The current release implements only rudimentary functionality and serves as a basic proof of concept. Thus, the code of the enabler itself does not provide any configuration options (configurability is planned for release 2). However, other components on which the enabler relies – in particular, Linux IMA, offer by default extensive integrity measurement configuration capabilities. We refer the reader to [5] for a detailed description of the configuration options of Linux IMA.

The regular workflow of the enabler is presented in Figure 2. The enumerated steps shown in Figure 2, that are not in **bold** have a very limited implementation in the current release, and will be addressed in more depth in Release 2. The workflow of the enabler is as follows: at boot time (in steps **1c**), the Linux Integrity Measurement Architecture (IMA) is used to measure the components of Open vSwitch and report it to the IMA runtime measurements file[3] [4] (the specific components to be measured depend on the configuration of the IMA policy)**.** In step (**2**), the OVS TEE attests the measurement collected in step (**1c**).

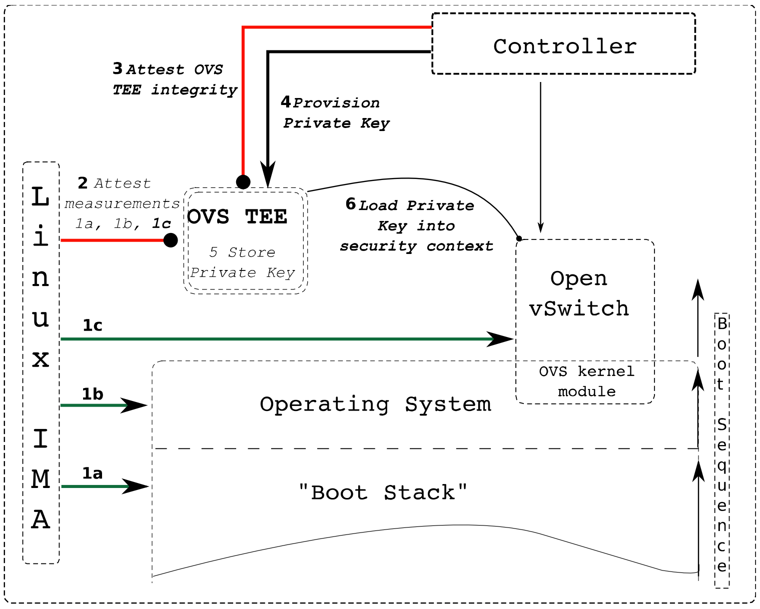


Figure 2: Workflow of enabler interaction with components on the host.

In case the local integrity attestation of the Open vSwitch component reveals a mismatch against the expected value, the OVS TEE reports the error – along with the mismatching value – to the controller. No key material is transferred to the OVS TEE; no other behavior of the controller is defined for this scenario.

Otherwise, the OVS TEE requests the *Open vSwitch private key* from the Controller; in turn, the Controller attests the integrity of the OVS TEE (**3**).

[3] We do not assume the presence of a Trusted Platform Module at this point.
[4] `/sys/kernel/security/ima/ascii_runtime_measurements`

Upon a successful attestation, the controller provisions the private key to the OVS TEE (**4**), enrolls the virtual switch in its topology (**6**) and communicates using the OpenFlow protocol; communication is protected with TLS, using the key material provisioned to the OVS TEE. Since the key will be stored within the security perimeter of the TEE, this will prevent the impersonation of the virtual switch[5] **even in the case when the adversary compromises the security of the platform where the virtual switch is executing**[6].

In case the remote integrity attestation of the OVS TEE fails, the behavior of the controller is undefined.

3.1.2 Enabler Output

As no interaction is expected between the "Trusted Boot" enabler and the network operator, the current implementation of the enabler only outputs log messages. Such messages are produced at the following points: Open vSwitch attestation by the OVS TEE, integrity attestation of the OVS TEE by the controller; provisioning of key material to the OVS TEE; enrollment of the virtual switch instance into the topology.

3.1.3 Open vSwitch enrollment

Once the attestation and key provisioning has been done (as described in 2.2.3.1, 2.2.3.2), the virtual switch will communicate with the Ryu network controller over a protected channel, without the risk of being impersonated, even if the host operating system is compromised[7].

3.2 Programmer Guide

The enabler is not programmable.

4 Unit Tests

4.1 Information about Tests

The enabler contains several unit tests in order to test the functionality of using key material stored in the TEE. These are relevant only for *release 1* and will be superseded by a more comprehensive test harness in *release 2*. The tests have been added to the collection of tests made part of the source distribution of Open vSwitch. To run the individual tests relevant to the "Bootstrapping Trust" enabler tests, check out the source of `openvswitch-tswitch`, and following the build steps described in [7], run:

```
make check TESTSUITEFLAGS=<TestNr>
```

where <TestNr> corresponds to the test numbers specified below.

[5] This risk is mentioned in the Open vSwitch TLS configuration manual [8].

[6] As mentioned in [2] **the emulator does not provide any security guarantees** and is meant to be a proof of concept. However, such security guarantees can be provided once the enabler is implemented using the SGX SDK recently released by the CPU vendor.

[7] At this point, the TEE application does not do *runtime* monitoring of the virtual switch kernel module, so runtime attacks are not included in this scenario.

4.2 Unit Test 2252

This test aims to check that the integrity attestation of the `openvswitch.ko` kernel module in the OVS TEE, followed by the integrity attestation of the OVS TEE itself by the "Controllerapp" and the following provisioning of the private key is successful.

Test scenario:

- OVS TEE attests OVS kernel module.
- Controller app attests the integrity of the OVS TEE.
- Controller app provisions the private key to the OVS TEE.

Expected result: Success

Sample Test Output:

```
set /bin/bash './tests/testsuite' -C tests
AUTOTEST_PATH=utilities:vswitchd:ovsdb:vtep:tests::ovn/controller-
vtep:ovn/northd:ovn/utilities:ovn/controller 2252; \
"$@" || (test X'' = Xyes && "$@" --recheck)
## ------------------------------ ##
## openvswitch 2.5.90 test suite. ##
## ------------------------------ ##
2252: Test enclave attest and provision               ok

## ------------- ##
## Test results. ##
## ------------- ##

1 test was successful.
```

4.3 Unit Test 2253

This test aims to check that once provisioned, the OVS can interact with the OVS TEE to load the key into the openssl security context and establish connectivity with the controller over TLS.

Test scenario:

- Use the key obtained as a result of the integrity attestation
- Configure connectivity between Open vSwitch and controller over TLS
- Test configured connectivity between virtual switch and controller

Expected result: Success

Sample Test Output:

```
set /bin/bash './tests/testsuite' -C tests
AUTOTEST_PATH=utilities:vswitchd:ovsdb:vtep:tests::ovn/controller-
vtep:ovn/northd:ovn/utilities:ovn/controller 2253; \
"$@" || (test X'' = Xyes && "$@" --recheck)
## ------------------------------ ##
## openvswitch 2.5.90 test suite. ##
## ------------------------------ ##
2253: Test OVS-Controller communication over TLS with provisioned private key ok

## ------------- ##
## Test results. ##
## ------------- ##

1 test was successful.
```

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
TLS	Transport Layer Security
OS	Operating System

6 References

[1] 5G ENSURE Deliverable D3.1, “5G-PPP security enablers technical roadmap (early vision)”

[2] 5G ENSURE Deliverable D3.2, “5G-PPP security enablers open specifications (v1.0))”

[3] Jain, Prerit, et al. "OpenSGX: An Open Platform for SGX Research."*Proceedings of the Network and Distributed System Security Symposium, San Diego, CA*. 2016.

[4] Open SGX project website: https://github.com/sslab-gatech/opensgx/blob/master/README.md

[5] Linux IMA Manual: https://sourceforge.net/p/linux-ima/wiki/Home/

[6] Ryu SDN controller framework: https://osrg.github.io/ryu/

[7] Open vSwitch installation manual: https://github.com/openvswitch/ovs/blob/master/INSTALL.md

[8] Open vSwitch TLS configuration manual:

https://github.com/openvswitch/ovs/blob/master/INSTALL.SSL.md

[9] Intel SGX SDK release notes, Intel Corp.

[10] Integrity Measurement Architecture Project Website: https://sourceforge.net/projects/linux-ima/

Deliverable D3.4
5G-PPP Security Enablers Documentation (v1.0)
Enabler Micro-Segmentation

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	30.09.2016	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	VTT: Olli Mämmelä, Kimmo Ahola	

Contents

1 Introduction

The upcoming 5G networks are currently being designed and the general vision is that 5G will be software-defined and virtual. There will be a new mobile ecosystem that consists of heterogeneous services, applications, networks, users and devices.

The security of 5G will be critical, as multiple different players will be included in the 5G ecosystem, such as Massive Machine-Type Communications (mMTC), Machine to Machine (M2M) or Industrial Internet based companies. In the case of a cyber-attack, the consequences could be dramatic.

Consequently, the role of isolation, virtualization and network management is going to be important. Applications or services requiring high level of security need to be clearly isolated and secure from the rest of the network. Network slicing has been introduced to provide network isolation. This work presents the concept of micro-segmentation into 5G network security.

Potential customers for the micro-segmentation approach in 5G networks include hospitals, factories, Industrial Internet and IoT based companies. An IoT company that is using the 5G network needs to gather data from different sensors reliably and securely. The delay may not need to be an important requirement but security is of high concern. Hospitals would also benefit from a dedicated micro-segment from the 5G network that is highly secure since e.g. gathering patient information from various sensors needs to be extremely private.

This manual describes how the micro-segmentation security enabler is installed and administrated. The first release of the enabler includes the creation of a micro-segment, adding nodes to a micro-segment and deleting a micro-segment.

2 Installation and Administration Guide

The micro-segmentation enabler consists of several different software and configuration files that are used to run the enabler.

2.1 System Requirements

The micro-segmentation enabler runs in Mininet [1] that creates a virtual networking environment. The enabler uses OpenVirteX [2] software for network virtualization and the Ryu SDN controller [3] for management of micro-segments. OpenVirteX has been modified to fit the purpose of the enabler. For IEEE 802.1X based authentication, wpa_supplicant, FreeRadius and Hostapd software are used. The enabler also requires Java, Python and Screen software and some other libraries. All required software are installed automatically when using "*sudo ./install.sh all*" command.

The enabler has been tested in an Ubuntu Linux OS (either release 14.04.x LTS or release 16.04 LTS, server version recommended), which is the preferred OS for the enabler. The OS needs to have two network interfaces, the first one needs to be connected to the Internet (e.g. through NAT) and another one can be internal network. For example, using VirtualBox environment, check that Adapter 2 is enabled in "Settings -> Network -> Adapter 2 -> Enable Adapter 2". The type of network (e.g. "Attached to") can be "Internal Network".

2.2 Enabler Configuration

The enabler comes with shell scripts which are used for configuration of either one node version or two node version. These scripts are used for creating the underlying network topology, creating transport operator network, creating micro-segments and configuring the authentication process of the micro-segments. More detailed information on the scripts can be found from Section 3.2.

2.3 Enabler Installation

The enabler requires Mininet, Java, Python, OpenVirteX and Ryu SDN controller, wpa_supplicant, FreeRadius, Hostapd, Screen and some additional programs and libraries. All of the required software should be installed in the Ubuntu OS.

The enabler comes with a tar file that includes a modified version of OpenVirteX and scripts that are used for installing the required additional software, starting and running the enabler. The package can be downloaded from host machine with the following command (when using VirtualBox, Adapter 1 is attached to "NAT" and package can be found from home directory of user _username_ in host machine):

```
scp _username_@10.0.2.2:microsegmentation_v0_1.tbz2 .
```

The tar file can be extracted in the home folder of the user in the virtualised host:

```
tar xjvf microsegmentation_v0_1.tbz2
```

After the package is unpackaged, the OpenVirteX directory can be found in the home directory. Then all you need to do is following:

```
cd OpenVirteX/scripts/ensure

sudo ./install.sh all
```

There are two alternative ways to use this enabler: single node version and two node version. The two node version implements transport operator vs. virtual operator division and virtual operator can create micro-segments. The single node version is used as transport operator makes micro-segment creations. When the scripts ask the question "Which node configuration (node1 or node2):", please answer exactly node1 or node2 accordingly when using two node configuration and only node1 if using the single node configuration.

The *install.sh* script will install all required packages from Ubuntu repositories or git repositories, the user needs only to accept installation of packages.

2.4 Troubleshooting

Mininet [1], Ryu [3], and OpenVirteX [2] websites provide troubleshooting information of those programs. The biggest troubles are caused when creating virtual networks (slices / micro-segments) before the SDN switches and connections between them are registered to the virtualization software (OpenVirteX). So user should check the Log screen window and notice SDN switch registrations before creating the virtual network. And when using the two node configuration, also the flow of user interaction is very important.

3 User and Programmer Guide

3.1 User Guide

3.1.1 Overview

Figure 1 depicts how the micro-segmentation concept could be realized in an SDN network. The physical topology consists of two micro-segment subscribers, i.e. organizations or companies, connected to SDN switches and a network orchestrator/hypervisor. The micro-segments are virtual networks that consist of a SDN controller and several virtual SDN switches. The physical users are connected to one micro-segment while the physical SDN switches may be connected to both micro-segments. The SDN controller is responsible for controlling the traffic inside the micro-segment and the network orchestrator/hypervisor used for creating the two micro-segments A and B by the use of OpenVirteX virtualization software. The SDN controllers may hold applications for AAA and Security monitoring. Each micro-segment thus can have its own AAA entity for authenticating and authorizing users and accounting.

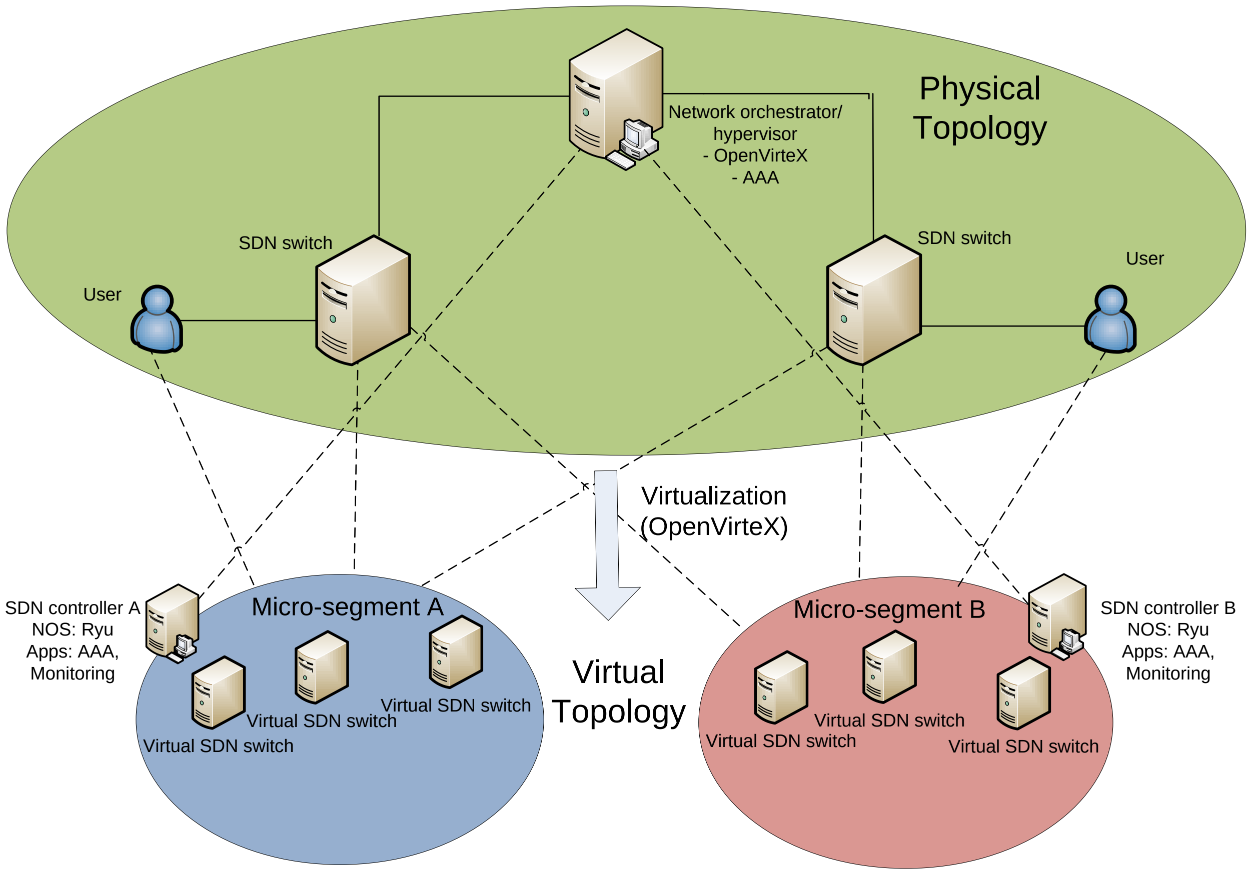


Figure 1: Micro-segmentation in an SDN network.

First tests with the enabler have been done in the Mininet environment [6]. The physical topology of switches created in Mininet is presented in Figure 2. Figure 3 depicts the scenario in which there are two micro-segments A and B with four SDN switches inside them. Host h_a is connected to micro-segment A and host h_b is connected to micro-segment B. SDN A controller is used for managing network traffic in micro-segment A and SDN B controller for managing network traffic in micro-segment B with the use of OpenFlow protocol. The two micro-segments are in different IP subnets, as shown in the figure. Between the micro-segments, there is an IP router used to connect the micro-segments to enable a multi-domain scenario. Currently, the connection between h_a and h_b is working, and we have implemented an IEEE 802.1X Extensible Authentication Protocol over LAN (EAPoL) based authentication method into the scenario for authenticating users and authorizing them to use the micro-segment.

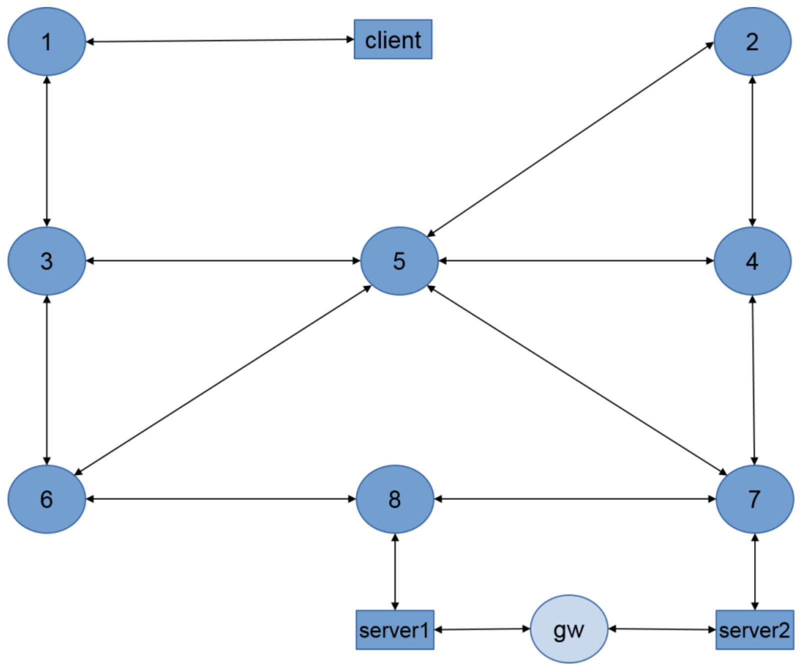


Figure 2 : Physical topology in Mininet

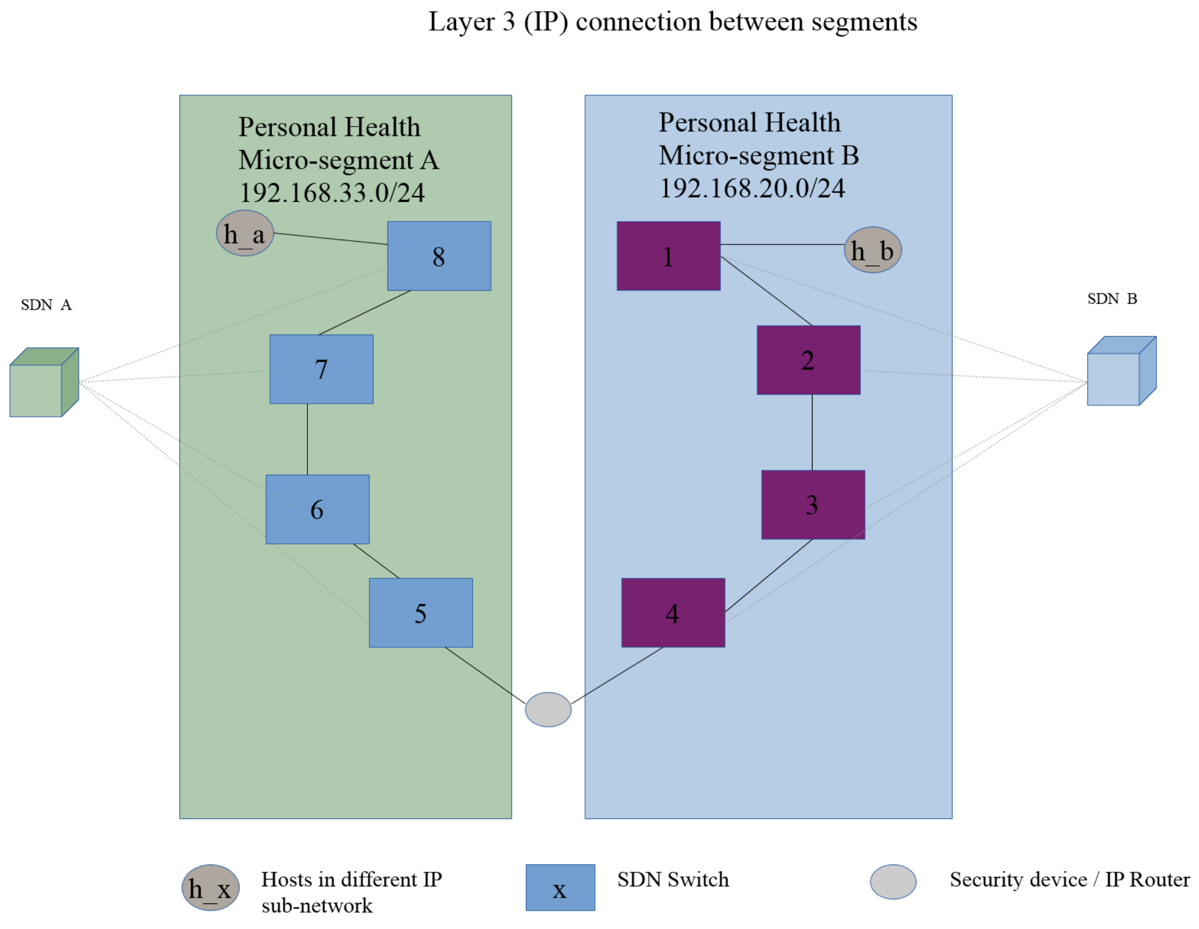

Layer 3 (IP) connection between segments
Personal Health Micro-segment A 192.168.33.0/24
Personal Health Micro-segment B 192.168.20.0/24
h_a
8
7
6
5
1
h_b
2
3
4
SDN A
SDN B
h_x
Hosts in different IP sub-network
x
SDN Switch
Security device / IP Router

Figure 3 : Micro-segmentation in Mininet.

3.2 Programmer Guide

The underlying network topology, the topology of the micro-segments and everything else can be modified within the following shell scripts:

- Start_screen.sh: starts all the needed scripts for the enabler in node.
- Start_screen_node1.sh: starts all the needed programs / scripts for the enabler running in node1.
- Topo_own_3h_8s.py: For modifying the underlying network topology.
- Create_transport_operator.sh: This script creates switches, ports and connects links within the underlying transport network from which virtualized networks are created.
- Create_transport_operator_node1.sh: This script creates switches, ports and connects links within the underlying transport network from which virtualized network is created.
- Create_transport_operator_node2.sh: This script creates switches, ports and connects links within the virtualized network from which micro-segment networks are created.
- Create_mno01_operator.sh: This script creates the first virtualized network mobile network operator (MN0)1.
- Create_mno01_operator_node2.sh: This script creates the first virtualized micro-segment known as mobile network operator (MN0)1.
- Create_mno02_operator.sh: This script creates the second virtualized network for MNO2.
- Create_mno02_operator_node2.sh: This script creates the second virtualized micro-segment known as MNO2.
- Start_transport_operator_ryu.sh : starts the Ryu SDN controller on the transport operator network
- Start_transport_operator_ryu_node2.sh : starts the Ryu SDN controller on the virtual operator network
- Start_mno01_operator_ryu.sh: starts the Ryu SDN controller on the MNO1 network.
- Start_mno01_operator_ryu_node2.sh: starts the Ryu SDN controller on the first micro-segment (MNO1 network).
- Start_mno02_operator_ryu.sh: starts the Ryu SDN controller on the MNO2 network.
- Start_mno02_operator_ryu_node2.sh: starts the Ryu SDN controller on the second micro-segment (MNO2 network).
- Start_hostapd.sh: starts the hostapd software.
- Hostapd.conf: configures the authentication server.
- Wpasupplicant-mno01.conf: configures the authentication procedure for MNO1 network.
- Wpasupplicant-mno02.conf: configures the authentication procedure for MNO2 network.
- Ensure_test.sh: a script used for the IEEE 802.1X authentication procedures.
- install.sh: a script, which installs all the needed additional software to Ubuntu OS.

4 Unit Tests

4.1 Unit Test 1

This simple test checks that the enabler is started in only node1, micro-segments are created, nodes are added to the micro-segments and nodes are correctly authenticated to the micro-segments.

The enabler is started with the start_screen.sh command in directory $HOME/OpenVirteX/scripts/ensure. Then change to screen window 1 (CTRL+A 1) and give your password for sudo command. Next same for screen window 2 (CTRL+A 2). After waiting for a while (5-10s), navigate to screen window 5 (OVX_creation) (by pressing CTRL+A 5) and then press ENTER/RETURN. Once these commands have been executed, navigate to window 1 (CTRL+A 1, mininet) and execute the following commands:

```
remote ./ensure_test.sh br-ensure port; wpa_supplicant -i tap-ensure -Dwired -c wpasupplicant-mno01.conf
```

Afterwards, "tap-ensure: CTRL-EVENT-CONNECTED" should be shown on the screen and you can press "CTRL-C". This authenticates a node to the first micro-segment. Connection can be tested by the following command:

```
remote ping 192.168.33.1
```

For authenticating a node to the second micro-segment and testing it, the following commands can be executed in screen window 1:

```
remote ./ensure_test.sh br-ensure port2; wpa_supplicant -i tap-ensure2 -Dwired -c wpasupplicant-mno02.conf
```

Press "CTRL-C".

```
remote ping 192.168.20.2.
```

In both cases, if the ping works, the nodes have been correctly authenticated and the enabler is functioning properly. Remember to exit all screen windows (some windows needs first pressing once or twice CTLR-C and then "exit" command) and therefore stopping the screen software when ending this unit test.

4.2 Unit Test 2

This test presumes that two node version is installed to two different VMs. This test also checks that the micro-segments are created, nodes are added to the micro-segments and nodes are correctly authenticated to the micro-segments.

The node 1 enabler is started with the start_screen_node1.sh command in directory $HOME/OpenVirteX/scripts/ensure. Then navigate to screen window 1 (CTRL+A 1) and give your password for sudo command. After waiting for a while (1-2s), navigate to screen window 2 (OVX_creation) (by pressing CTRL+A 2) and then press ENTER/RETURN. Once these commands have been executed, change to Node 2 installation.

In node 2, go to the directory $HOME/OpenVirteX/scripts/ensure and execute start_screen_node2.sh command. Then navigate to window 1 (CTRL+A 1) and give your password for sudo command. Then wait for a while (2-4s) and navigate to screen window 4 (CTRL+A 4) and press ENTER/RETURN. Check that you don't see any error messages in the output of scripts.

Then change again to Node 1 installation. Navigate to screen window 1 and execute the following commands:

```
remote ./ensure_test.sh br-ensure port; wpa_supplicant -i tap-ensure -Dwired -c wpasupplicant-mno01.conf
```

Afterwards, "tap-ensure: CTRL-EVENT-CONNECTED" should be shown on the screen and you can press “CTRL-C”. This authenticates a node to the first micro-segment. Connection can be tested by the following command:

```
remote ping 192.168.33.1
```

For authenticating a node to the second micro-segment and testing it, the following commands can be executed in screen window 1:

```
remote ./ensure_test.sh br-ensure port2; wpa_supplicant -i tap-ensure2 -Dwired -c wpasupplicant-mno02.conf
```

Press “CTRL-C”.

```
remote ping 192.168.20.2.
```

In both cases, if the ping works, the nodes have been correctly authenticated and the enabler is functioning properly. Remember to exit all screen windows (some windows needs first pressing once or twice CTLR-C and then “exit” command) and therefore stopping the screen software when ending this unit test.

4.3 Unit Test 3

This test checks if a micro-segment can be deleted from the topology and the test should be done after Unit Test 1 or Unit Test 2 is working correctly and screen software is still running. This is done by executing the script remove_mno01_operator.sh in screen window 5 (OVX_creation), which removes the first micro-segment when using single node version. If this test is done in two node version, then script should be executed in Node 2. The second micro-segment can be deleted by the script remove_mno02_operator.sh.

After executing the command, the following should text should come to the screen:

```
./remove_mno1_operator.sh

Removing MVO 01 network

Network (tenant_id_2) has been removed.
```

Also, running the command $HOME/OpenVirteX/utils/ovxctl.py –n getVirtualTopology 2 should print no information about the topology when mno1 operator is deleted. Remember to exit all screen windows (some windows needs first pressing once or twice CTLR-C and then “exit” command) and therefore stopping the screen software when ending this unit test.

4.4 Unit Test 4

This test checks if a node can be removed from a micro-segment and the test should be done after Unit Test 1 or Unit Test 2 is working and screen software is still running. This is done by executing either the script remove_host_mno01_operator.sh or remove_host_mno02_operator.sh in screen window 5 (OVX_creation) when using the single node version. If this test is done in two node version, the script should be executed in Node 2.

After executing the command, the following should text should come to the screen:

```
./remove_host_mno01_operator.sh

Removing server from virtual switch in MVO 01 network
```

```
Host (host_id 1) has been disconnected from the virtual network (tenant_id 2).
```

Also, running the command $HOME/OpenVirteX/utils/ovxctl.py –n getVirtualHosts 2 should print no information about the node when removing the node from mno1 operator network. Remember to exit all screen windows (some windows needs first pressing once or twice CTLR-C and then “exit” command) and therefore stopping the screen software when ending this unit test.

5 Abbreviations

5G-PPP	5G Infrastructure Public Private Partnership

6 References

[1] "Mininet," [Online]. Available: http://mininet.org/.

[2] "OpenVirteX," [Online]. Available: http://ovx.onlab.us/.

[3] "Ryu SDN Framework," [Online]. Available: https://osrg.github.io/ryu/.