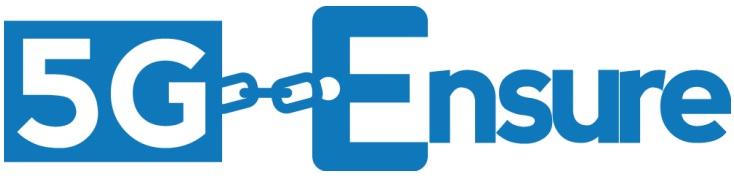

Deliverable D3.4
5G-PPP Security Enablers Documentation (v1.0)
Enabler Micro-Segmentation

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	

Document Version	Date	Change(s)	Author(s)
0.1	28.06.2016	Created template	Felix Klaedtke
0.2	19.8.2016	First draft	Olli Mämmelä, Kimmo Ahola
0.3	27.8.2016	Version for review	Kimmo Ahola
1.0	13.9.2016	Revision of the manual based on review comments	Olli Mämmelä, Kimmo Ahola

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 manual is part of the project's deliverable D3.4. It describes how the Micro-Segmentation Enabler is developed within the work package 3 of the 5G-ENSURE project is installed and administrated. Furthermore, this manual contains a user guide of the security enabler.

Disclaimer

Copyright notice

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.

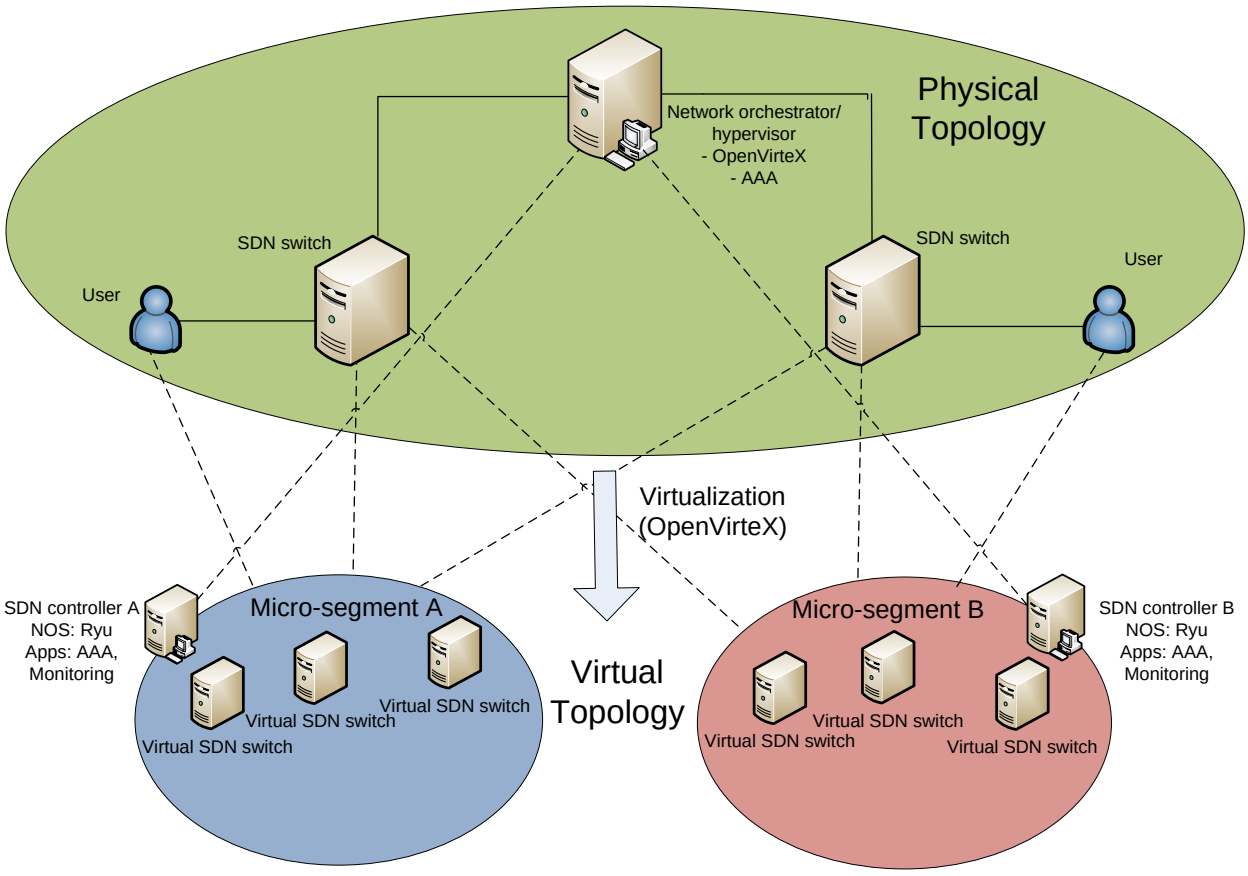

Figure 1: Micro-segmentation in an SDN network.

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.

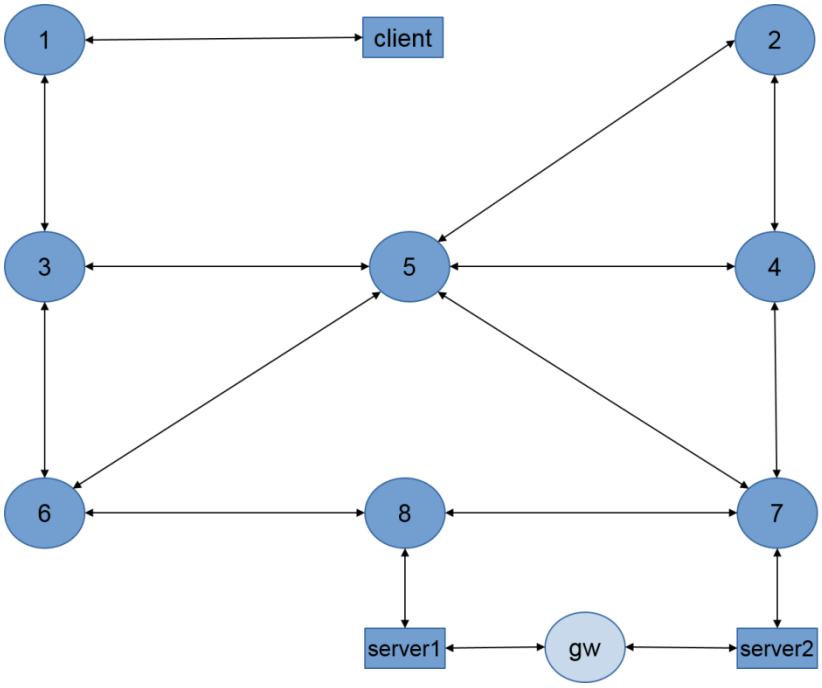

Figure 2 : Physical topology in Mininet

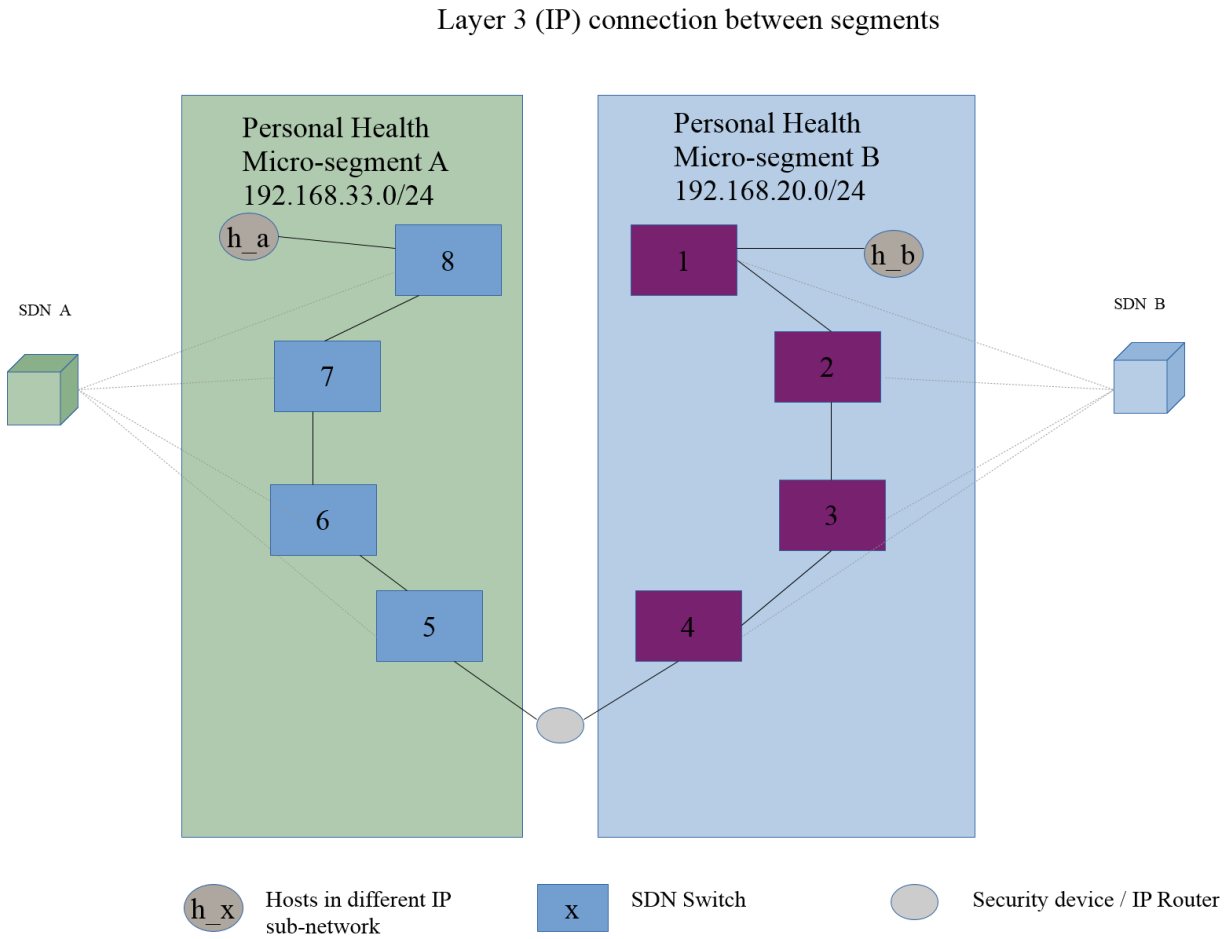

Figure 3 : Micro-segmentation in Mininet.

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/.