



Deliverable D3.8

5G-PPP Security Enablers Documentation (v2.0)

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	31.08.2017	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	VTT: Olli Mämmelä, Kimmo Ahola	

This part will be removed before the final edition and publication

Document Version	Date	Change(s)	Author(s)
0.1	02.06.2017	Created template	Felix Klaedtke
0.2	16.7.2017	Release 2 info added	Olli Mämmelä, Kimmo Ahola
0.3	10.8.2017	Revision 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, standardization and vision for a secure, resilient and viable 5G network. The project covers research and innovation - from technical solutions (5G security architecture and testbed with 5G security enablers) to market validation and stakeholders engagement - spanning various application domains.

This manual is part of the project's deliverable D3.8. It describes how 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

The information in this document is provided 'as is', and no guarantee or warranty is given that the information is fit for any particular purpose.

The EC flag in this deliverable is owned by the European Commission and the 5G PPP logo is owned by the 5G PPP initiative. The use of the flag and the 5G PPP logo reflects that 5G-ENSURE receives funding from the European Commission, integrated in its 5G PPP initiative. Apart from this, the European Commission or the 5G PPP initiative have no responsibility for the content.

Copyright notice

© 2015-2017 5G-ENSURE Consortium

Contents

1	Introduction.....	5
2	Installation and Administration Guide	5
2.1	System Requirements.....	5
2.2	Enabler Configuration.....	6
2.3	Enabler Installation.....	6
2.3.1	Installation from source	6
2.3.2	Installation from Debian/Ubuntu packages	6
2.4	Troubleshooting	7
3	User and Programmer Guide.....	7
3.1	User Guide	7
3.1.1	Overview.....	7
3.2	Programmer Guide	10
4	Unit Tests.....	11
4.1	Unit Test 1.....	11
4.2	Unit Test 2.....	12
4.3	Unit Test 3.....	13
4.4	Unit Test 4.....	13
4.5	Unit Test 5.....	13
4.6	Unit Test 6.....	14
5	Abbreviations.....	14
6	References	15

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 second release of the enabler includes the creation of a micro-segment, adding nodes to a micro-segment, deleting a micro-segment, adding support for multi-domain micro-segments, and extended northbound API to enable integration with the Security Monitor for 5G Micro-Segments and Trust Metric enablers.

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 using Open vSwitch [2] virtual switches. The enabler uses OpenVirtEX [3] software for network virtualization and the Ryu SDN controller [4] 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, OpenVPN and Screen software and some other libraries. All required software is installed automatically when using “sudo ./install.sh all” command or when installing as a Debian/Ubuntu package.

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.

2.3 Enabler Installation

2.3.1 Installation from source

The enabler requires Mininet, Java, Python, OpenVirtEX and Ryu SDN controller, wpa_supplicant, FreeRadius, Hostapd, Screen, OpenVPN 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_@_hostIP_: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. In the single node version, the transport operator of the network creates micro-segments and is thus the owner of the micro-segments. In the two node version, the transport operator shares its network to virtual operators. In this case, the virtual operator creates the micro-segments and is thus the owner of the micro-segments. 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.

2.3.2 Installation from Debian/Ubuntu packages

The enabler is also bundled as a Debian package which simplifies the deployment. User can use the familiar apt / apt-get tools to install enabler and all the requirements.

```
sudo apt-get update  
sudo apt-get install microsegmentation
```

2.4 Troubleshooting

Mininet [1], Ryu [4], and OpenVirtX [3] 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 (OpenVirtX). The 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

In release 1 (R1), the enabler was functioning independently without a connection to the Trust Metric and Security Monitor for 5G Micro-Segments enablers. This means that the enabler was able to create micro-segments, delete micro-segments, add nodes to micro-segment, and delete micro-segments without control from the Security Monitor for 5G Micro-Segments enabler.

In release 2 (R2), the enabler is integrated with the Trust Metric and Security Monitor for 5G Micro-Segments enablers. Figure 1 shows the architectural diagram of the micro-segmentation approach inside a single administrative domain. In the figure, the micro-segmentation enabler is used for creating and deleting micro-segments dynamically. The monitoring framework gathers information and metrics from various sources, such as mobile network functions (MME, P-GW, S-GW, etc.), IoT gateway and eNodeB. This information is fed to the security inferencing (i.e., Security Monitor for 5G Micro-Segments) and Trust Metric enabler modules. Based on this information, the security inferencing module will generate actions for the micro-segmentation enabler to execute.

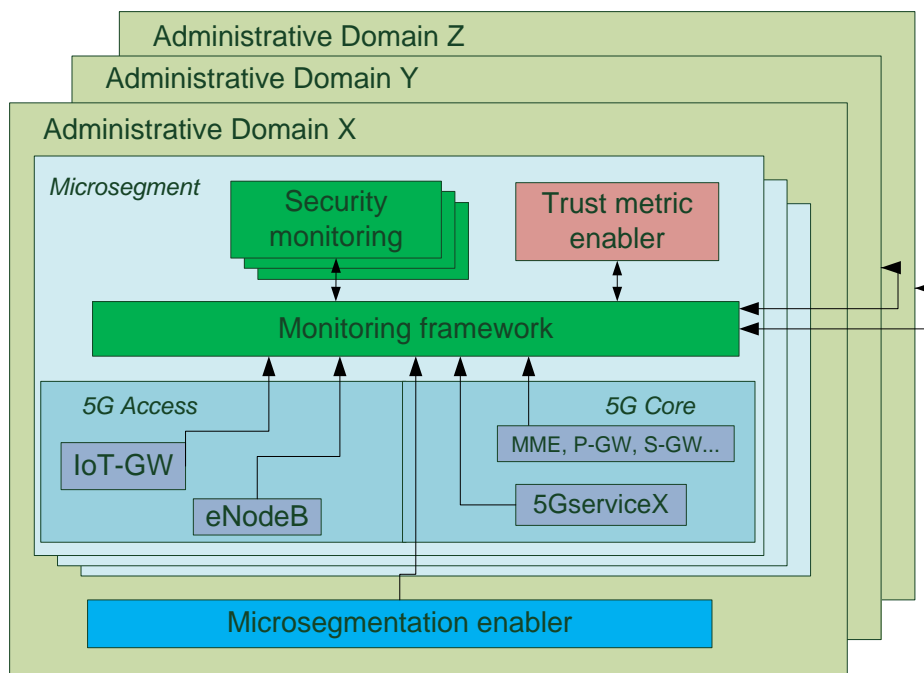


Figure 1 Architectural diagram of micro-segmentation approach inside a single administrative domain.

Figure 2 shows the architectural diagram of the micro-segmentation enabler. As can be seen from the figure, the enabler will be connected to a Micro-segment Apache Kafka feeder module that retrieves network topology and network traffic information via the northbound interface of the enabler. Micro-segment Apache Kafka feeder sends the information in JSON format to Event broker (Apache Kafka). Finally, the information is distributed to the Trust Metric and Security Monitoring enablers. Based on this information, the Security Monitoring enabler will generate control actions via Event broker and Micro-segment Apache Kafka feeder to the micro-segmentation enabler. The control actions are sent via WebSocket protocol. The trust metric enabler also needs information about micro-segments to calculate a trust metric value for a micro-segment.

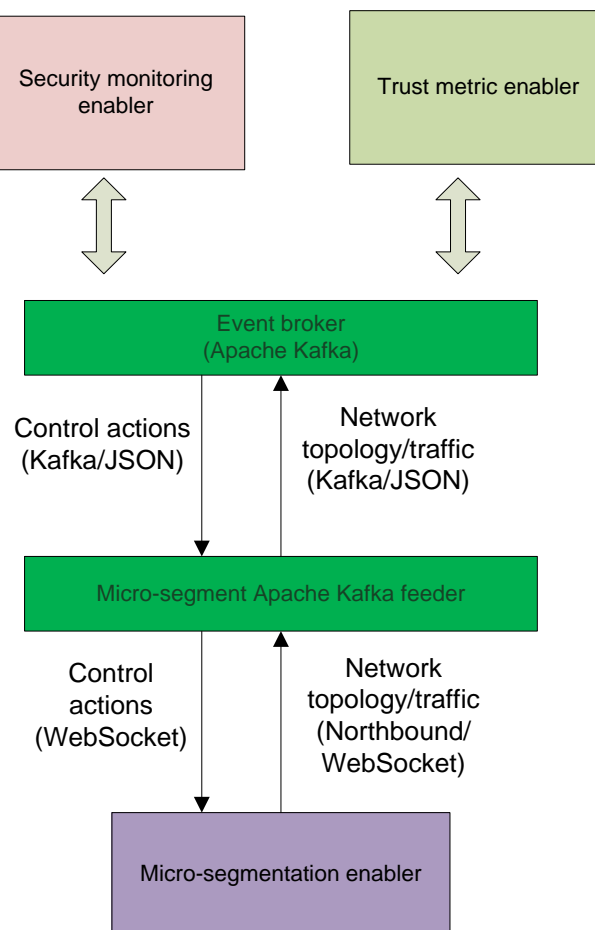


Figure 2 Architectural diagram of the micro-segmentation enabler.

The tests with the enabler have been done in the Mininet environment [1]. The physical topology of switches created in Mininet is presented in Figure 3. Topology consists of 8 OpenFlow enabled switches (Open vSwitch [2]), two servers and two clients. The SDN controller is controlling the physical network (not shown in the figure).

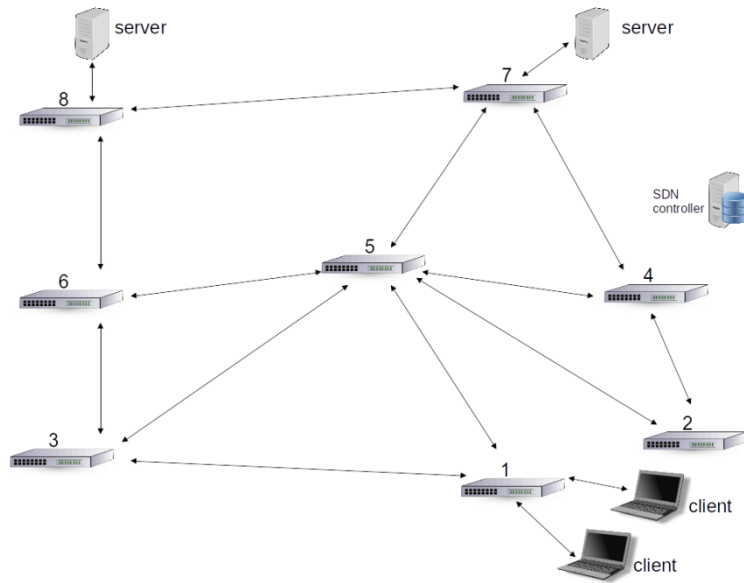


Figure 3 Physical topology in Mininet

Next picture Figure 4 shows the physical topology and two micro-segments (blue and green) on top of physical topology. The OpenVirtex is now virtualising the switches and both virtual networks (blue and green) sees only their part of the network topology. In this picture, the micro-segments are isolated and communication between them is not supported. Only IEEE 802.1X Extensible Authentication Protocol over LAN (EAPoL) authenticated users can use the micro-segments.

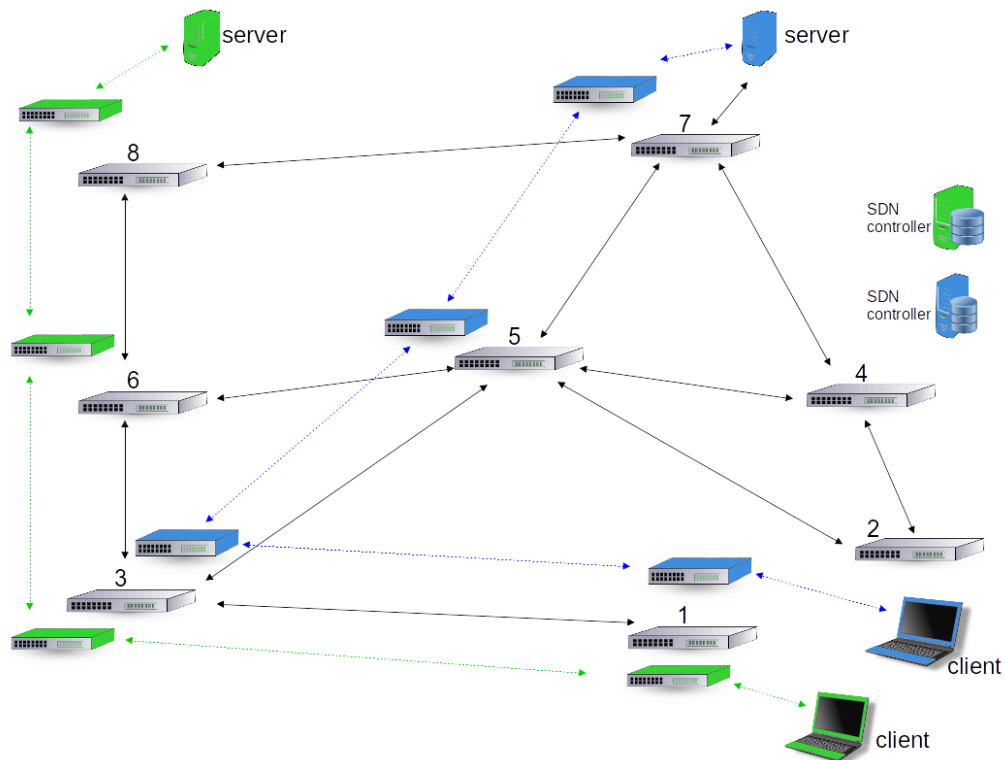


Figure 4 Micro-segmentation in Mininet

Figure 5 depicts the scenario in which there are two micro-segments Green (G) and Blue (B) with four SDN switches inside them. Green client is connected to micro-segment G and client blue is connected to micro-segment B. Green SDN controller is used for managing network traffic in micro-segment G and Blue SDN controller for managing network traffic in micro-segment B with the use of OpenFlow protocol. The two micro-segments are in different IP subnets. Communication between micro-segments are connected using VPN tunnel and therefore enabling a multi-domain scenario.

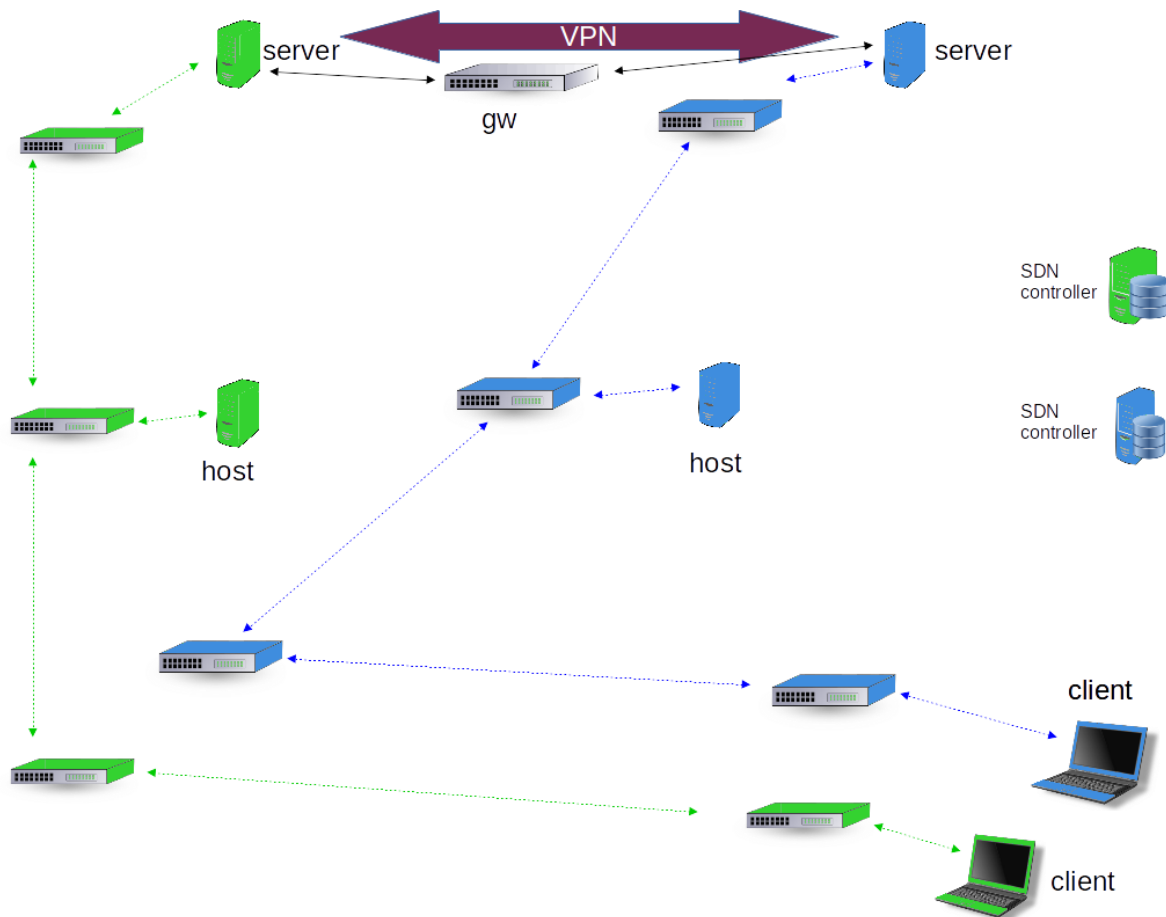


Figure 5 Communication between micro-segments

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[_node1.sh|_api_test|_multidomain]`: starts all the needed programs / scripts for the enabler running in different scenarios.
- `start_mininet[_multidomain].sh`: starts the Mininet software with the specific topology.
- `topo_mininet[_multidomain].py`: Network topology in Mininet in different scenarios.
- `Create_transport_operator[_node1|_node2].sh`: This script creates switches, ports and connects links within the underlying transport network from which virtualized network is created.

- `Create_mno01_operator[_node2|_multidomain].sh`: This script creates the first virtualized network mobile network operator MNO1 in different scenarios.
- `Create_mno02_operator[_node2|_multidomain]].sh`: This script creates the second virtualized network for MNO2 in different scenarios.
- `Start_transport_operator_ryu[_node2].sh`: starts the Ryu SDN controller on the transport operator network.
- `Start_mno01_operator_ryu[_node2].sh`: starts the Ryu SDN controller on the MNO1 network.
- `Start_mno02_operator_ryu[_node2].sh`: starts the Ryu SDN controller on the MNO2 network.
- `Start_hostapd.sh`: starts the hostapd software.
- `Hostapd.conf`: configures the authentication server.
- `Wpa_supplicant-mno01.conf`: configures the authentication procedure for MNO1 network.
- `Wpa_supplicant-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.
- `multidomain.sh`: create / delete multidomain communication between micro-segments.

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 wpa_supplicant-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 wpa_supplicant-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 CTRL-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/OpenVirtX/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/OpenVirtX/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 CTRL-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 CTRL-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 CTRL-C and then “exit” command) and therefore stopping the screen software when ending this unit test.

4.5 Unit Test 5

The purpose of this test is to check that the APIs that were implemented between the Micro-segmentation, Trust Metric, and Security Monitor for 5G Micro-Segments enablers works correctly. This means that it is possible for the Micro-segmentation enabler to see, e.g., all the switches and rules of a micro-segment and block certain nodes in a micro-segment. The prerequisite of this test is that test is started with

`./start_screen_api_test.sh` and Unit Test 1 commands has been completed and ping command is running all the time.

Go to screen window 7 and press Enter/Return (which will start the below command):

```
python -m microsegmentation/ensure_websocket_tester
```

Afterwards, the script should print information regarding the micro-segment to the screen. Also the ping command in screen window 1 shows that connection is not working about 10 seconds.

4.6 Unit Test 6

The purpose of this test is to check that two micro-segments can be securely connected together. The prerequisite of this test is that test is started with command “`start_screen_vpn.sh`” and Unit Test 1 has been completed.

Go to Screen window 7 and run the following script (press only Enter/Return):

```
sudo ./multidomain.sh up
```

Next, go to screen window 8 and start the *tcpdump* with the following parameters:

```
sudo ip netns exec gw tcpdump -i gw-eth0 -n- vvv -e -l
```

Navigate to screen window 1 to ping a node in another micro-segment:

```
mno1_h ping 192.168.20.100
```

Check that the ping works and notice from the *tcpdump* window that the ping is going through a VPN tunnel (all traffic between microsegments are going through OpenVPN tunnel with UDP and port 1194).

The multidomain test can be stopped with command:

```
sudo ./multidomain.sh down
```

5 Abbreviations

5G-PPP	5G Infrastructure Public Private Partnership

--	--

6 References

- [1] "Mininet: an instant virtual network on your laptop," [Online]. Available: <http://mininet.org/>.
- [2] "Open vSwitch," [Online]. Available: <http://openvswitch.org>.
- [3] "OpenVirtex," [Online]. Available: <http://ovx.onlab.us/>.
- [4] "Ryu SDN Framework," [Online]. Available: <https://osrg.github.io/ryu/>.