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	

European Commission

5G PPP

This part will be removed before the final edition and publication

Document Version	Date	Change(s)	Author(s)
0.1	28.06.2016	Created template	Felix Klaedtke
0.2	8.09.2016	First draft	Pekka Ruuska
0.3	13.09.2016	Updated after VTT's review	Pekka Ruuska
0.4	21.9.2016	Updated after B-COM's review	Pekka Ruuska

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 one of the security enablers that are developed within the work package 3 of the 5G-ENSURE project is installed and administrated. Furthermore, this manual contains a user guide of the respective security enabler.

Disclaimer

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

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.

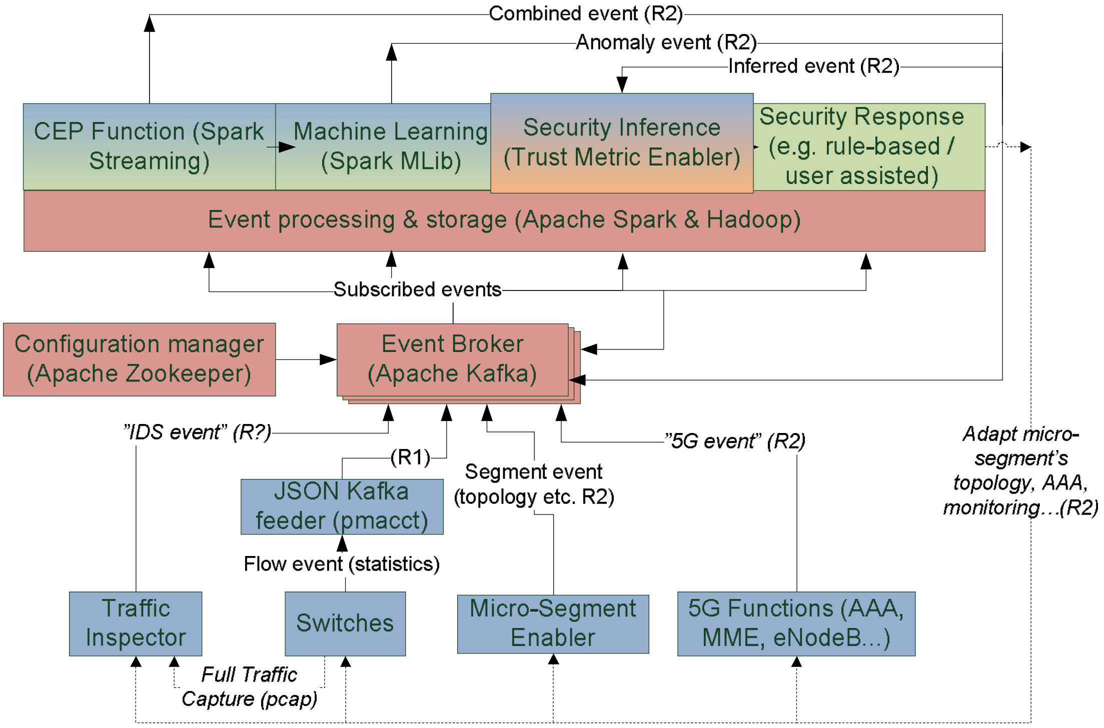
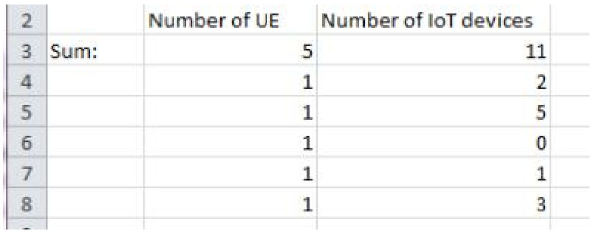

Combined event (R2)
Anomaly event (R2)
Inferred event (R2)
CEP Function (Spark Streaming)
Machine Learning (Spark MLib)
Security Inference (Trust Metric Enabler)
Security Response (e.g. rule-based / user assisted)
Event processing & storage (Apache Spark & Hadoop)
Subscribed events
Configuration manager (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.

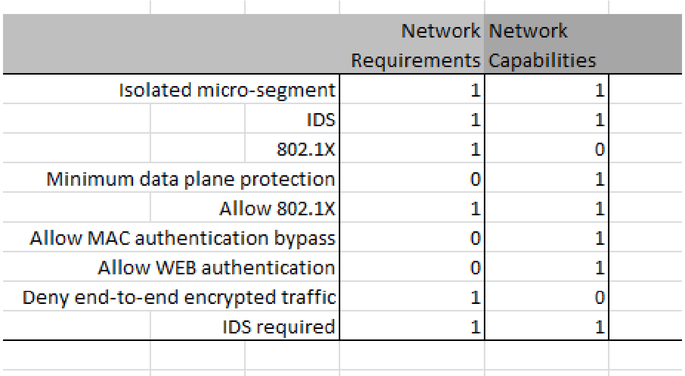
In Release 1, input data for the Trust Metric Enabler's prototype version is presented as .csv-files, in other words Excel-files, which are named **`number_of_devices.csv`** and **`trust_requirements.csv.`** The file **`number_of_devices.csv`** must present a table with two columns and at least one row,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