Deliverable D3.8
5G-PPP Security Enablers Documentation (v2.0)
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	31.08.2017	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	VTT: Jani Suomalainen, 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	22.6.2017	Manual for release 2	Jani Suomalainen
0.3	15.7.2017	Updated according to review feedback and input on Debian package installation	Jani Suomalainen, 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.

Trust metric enabler provides awareness of 5G networks trustworthiness for clients and stakeholders in other domains. The enabler is a tool for aggregating and tracking trust related information. Roadmap and open specification for the enabler has been published in 5G-ENSURE deliverables 3.5 [6]and 3.6 [7]. This software release 2 is a completely rewritten version, providing flexibility and support for run-time operations.

Disclaimer

Copyright notice

Contents

1 Introduction

5G networks may to provide different security levels. For instance, operators may have different practises, which affect to security levels, and network slices (provided e.g. the microsegmentation enabler) may have customised authentication and monitoring solutions. The users (e.g. organizations, service providers, end-users, or coopering operators) of 5G networks need awareness on how trustworthy the services are. The users need information on whether the network fulfill their requirements in a manner that it can be trusted. However, disclosing all security related information from the 5G network to the users is not be feasible due to the amount and complexity of information as well as due to sensitivity of the information for operators.

Trust Metric Enabler (TME) provides one solution to these problems. It enables 5G users to specify and track required level of security for the network. Users can specify arbitrary trust models that must be fulfilled before they can trust the network. Trust metric enabler tracks whether these requirements are fulfilled and informs users on changes in the required trust levels.

Release 1 of TME provided a prototype for resolving some trust metric examples using static configuration files. Release 2 is rewritten to provide more flexibility and support for dynamic run-time trust measurements. In Release 2, TME can be requested to follow any kind of measurements. TME receives event information from the monitoring components in 5G network. To enable distributed collection of event information from various sources, TME supports publish-and-subscribe based event information brokering platform (Kafka). One potential information source for TME is Security Monitor for 5G Micro-Segments enabler (that was integrated to TME in Release 2). Figure 1 illustrates - the role of TME between 5G service provider and 5G client as a mediator or aggregator of trust related information.

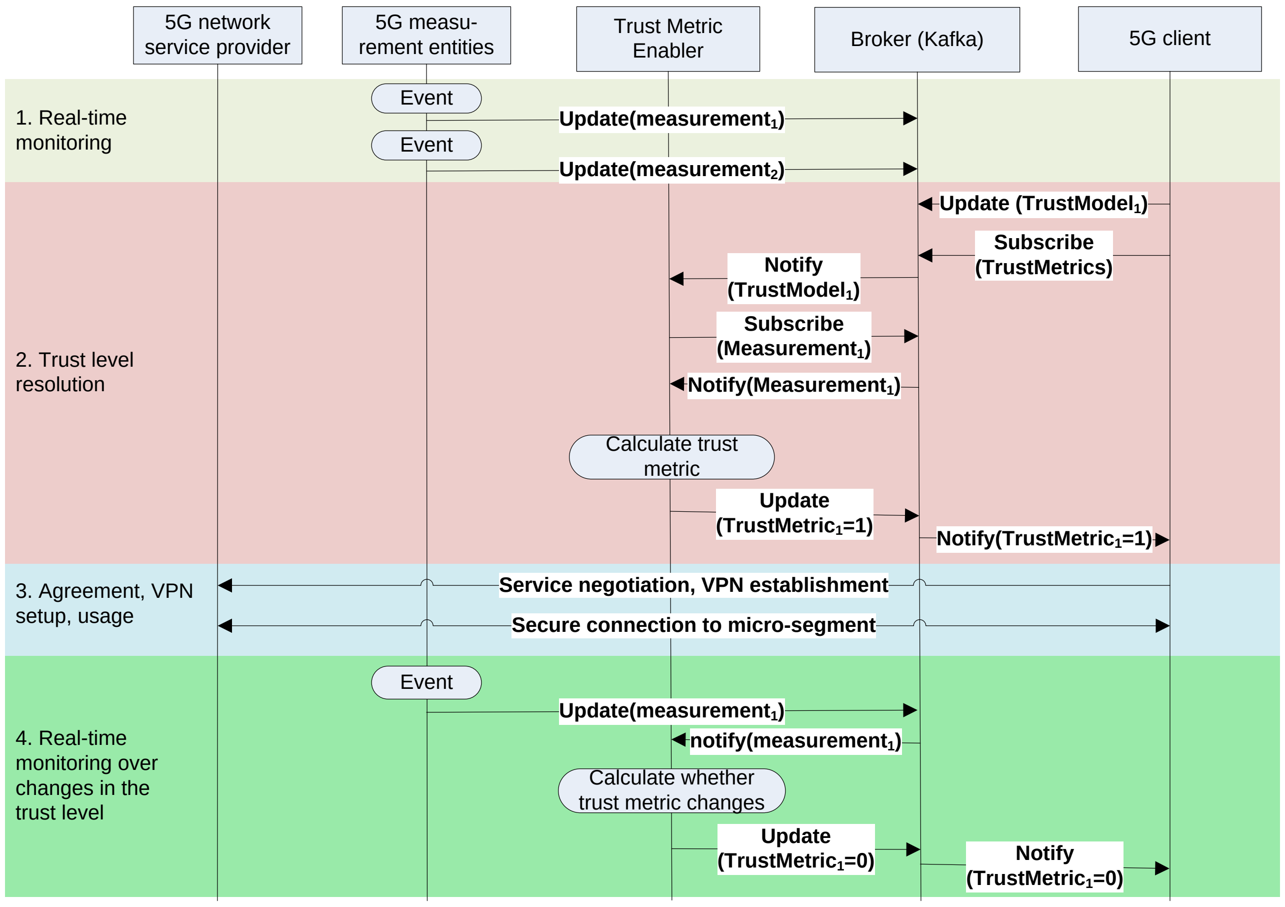

5G network service provider
5G measu-rement entities
Trust Metric Enabler
Broker (Kafka)
5G client
1. Real-time monitoring
Event
Update(measurement$_1$)
Event
Update(measurement$_2$)
Update (TrustModel$_1$)
Subscribe (TrustMetrics)
Notify (TrustModel$_1$)
Subscribe (Measurement$_1$)
2. Trust level resolution
Notify(Measurement$_1$)
Calculate trust metric
Update (TrustMetric$_1$=1)
Notify(TrustMetric$_1$=1)
3. Agreement, VPN setup, usage
Service negotiation, VPN establishment
Secure connection to micro-segment
Event
Update(measurement$_1$)
notify(measurement$_1$)
4. Real-time monitoring over changes in the trust level
Calculate whether trust metric changes
Update (TrustMetric$_1$=0)
Notify (TrustMetric$_1$=0)

Figure 1. Messaging diagram illustrating the role of Trust Metric Enabler

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

2 Installation and Administration Guide

2.1 System Requirements

The enabler is executed in Linux host. Apache Kafka, Java and Python must be available (see Kafka installation instruction later).

2.2 Enabler Configuration

2.2.1 Kafka

Apache Kafka is distributed with example configurations that may be used in simple testing scenarios with one host. Default configuration is sufficient for tests. Configuration guidelines on more complex scenarios - using e.g. large amount of clustered servers - can be found from Apache web-site [1].

2.2.2 Trust Metric Enabler

Trust metric enabler does not have any configuration options. Trust policy specification is described in Section 3.

2.3 Enabler Installation

2.3.1 Manual installation

Apache Kafka is freely available for Apache's software repository [4]. Configuration guidelines can be found from the Apache web-site [5]. Installation is straightforward in typical scenarios as illustrated in Figure 2.

```
$ wget http://www.nic.funet.fi/pub/mirrors/apache.org/kafka/0.8.2.2/kafka_2.11-0.8.2.2.tgz

$ tar –zxvf kafka
```

Figure 2: Installing ZooKeeper and Kafka broker

The enabler requires also some python libraries:

```
sudo apt-get install python-pip

pip install kafka-python
```

Figure 3: Installing python libraries

2.3.2 Installation with Debian packages

When enabler deployed as Debian packages, there is also easier way to install enabler. The user can use the familiar apt / apt-get tools to install enabler and all the requirements.

```
sudo apt-get update

sudo apt-get install trustmetric
```

Figure 4: Installation with Debian packages

2.4 Troubleshooting

Kafka[1] documentation and web-sites provide Kafka specific troubleshooting information.

Kafka broker may output excessive amount of debug information and also error messages when used with python clients. These messages can be ignored if the code otherwise works.

Startup order may cause problems. Kafka must be running before TME. TME must be running before trust client can request services.

3 User and Programmer Guide

3.1 User Guide

TME use Apache Kafka to broker trust policies, trust metrics and monitoring data. Figure 5 illustrates starting of Apache Kafka. Before launching the actual Kafka, Zookeeper must be started to provide configuration information.

```
$ cd /usr/local/src/kafka_2.11-0.8.8.2
$ ./bin/zookeeper-server-start.sh config/zookeeper.properties
$ ./bin/kafka-server-start.sh config/server.properties
```

Figure 5: Starting ZooKeeper and Kafka broker

After the broker has been started Trust Metric Enabler can be started.

```
$ python –m trust_metric/tme_kafka/tme_kafka
```

Figure 6: Starting trust metric enabler

3.1.1 Trust models

Trust models specify requirements that the 5G network must fulfil to provide the required trust level. There are three types of requirements:

- Service - Stating that a particular flag is set. For instance, this can be used to check that particular service is running.
- Max_level - Setting an upper boundary for particular aggregated events. “The network trusted if there is at most x amount of event y occurrences.”
- Min_level - Setting a lower boundary for particular aggregated events. “The network trusted if there is at least x amount of event y occurrences.”

All requirements stated in the model must be fulfilled in order to the metric be True.

Trust models are presented using JSON files. An example is given in Figure 7. The example contains identifier of the model, requirements that two particular services must be running (1), as well as requirements that the amount of devices that have been authenticated using MD5 based AKA is at most 50, and that the anomaly level is 1.

```
{
  'policyid': 1232,
  'services':
```

```
  {
      'msidx.securitymonitoringenabler': 1,
      'msidx.policycompliancechecker': 1
  },
 'maxlevels':
  {
      'msidx.authcounter.MD5': 50,
      'msidx.anomality_level': 1
  }
}
```

Figure 7: TME trust model example

Trust models are send to TME through Kafka. The client must know the broker that TME uses (its IP address and port). Trust models are send to Kafka topic: 'msidx.trustrequirements', where msidx can be replaced with microsegment identifier. (In the default configuration 'msidx' can be used for testing.)

Trust metrics are received from the broker by subscribing Kafka topic: 'msidx.trustmetrics'. The metric is presented as a simple JSON file, with model's identifier (as specified by trust model creator) and metric (True or False):

```
{
  'policyid':<id>,
   'metric': <True/False>
}
```

Figure 8: TME result example

3.2 Programmer Guide

Release 2 of the enabler consist of the following python files:

- **tme_kafka.py** - Listens kafka streams for incoming trust models (a.k.a. policies) from clients and measurements (events) from 5G network. Creates handlers for each model and also forwards incoming Kafka notifications to the handlers. This file contains the main() function that is used to launch the enabler.

- **tme_policy_handle.py** - Implements TrustPolicyHandler class for handling trust models. For each trust model, a new handler object is generated. The handler processes model and subscribes information described for the model. When subscribed information arrives, the handler checks whether the status of the model changes i.e. if an unfilled model becomes fulfilled or filled model becomes unfilled. Sends Kafka notifications to clients on status changes (fulfilled -> true; unfilled->false).

- **sp.py** - example trust metric client (a service provider wanting trust data from a microsegment). A python script for sending trust models and getting trust measurements via Kafka broker. Generates also web page 'trust indicator' illustrating trust with 'traffic lights' and displaying the related policy.

- **monitor.py** - Kafka feeder providing some microsegment specific 'pseudo-data'. This feeder can be used to test trust metric enabler in cases where the microsegmentation monitor enabler is not integrated. It also serves as an example on how the monitored measurements / key performance indicators are inputted to the trust metric enabler through Kafka.

3.2.1 Sending Trust Models

The following Python example shows how trust models are send through Kafka. See sp.py for a complete example.

```
from kafka import KafkaProducer
from kafka import KafkaConsumer
import json

broker = 'localhost:9092'

model = { 'policyid': 1,
  'services': {
     'msidx.securitymonitoringenabler': 1
  },
  'maxlevels': {
    'msidx.authcounter.MD5': 0
   }
}

p=KafkaProducer(broker,value_serializer=lambda m: json.dumps(m).encode('ascii'), retries=5)
producer.send('msidx.trustrequirements', model)
producer.flush()
```

Figure 9: Example of sending trust models

3.2.2 Receiving Trust Metrics

The following Python example shows how trust metrics are send through Kafka. See sp.py for a complete example. The example is in a loop receiving incoming messages. When a particular metric is received, the value is printed.

```
consumer = KafkaConsumer(‘msidx.trustmetrics’, group_id=’tmeclient’, broker,
value_deserializer=lambda m: json.loads(m.decode('ascii')))
for message in consumer:
   if message.value['policyid']==1:
      print "privacy related trust metric:", message.value['metric']
```

Figure 10: Example of receiving messages from Kafka

3.2.3 Sending 5G Measurements

The following Python example shows how trust measurements/key performance indicators are send through Kafka. See monitor.py for a complete example.

```
from kafka import KafkaProducer
import json

broker = 'localhost:9092'
```

```
producer = KafkaProducer(broker, value_serializer=lambda m: json.dumps(m).encode('ascii'),retries=5)
on = 1
off = 0
jsonmsg = {'id': msidx, 'msidx.securitymonitoringenabler': on}
producer.send('msidx.securitymonitoringenabler', jsonmsg)
producer.flush()
```

Figure 11: Example of sending measuremens / KPI through Kafka

4 Unit Tests

4.1 Unit Test 1: Trust measuring using pseudo monitoring data

This unit test shows how trust metric enabler can be populated with trust policies and security monitoring information. Based on this information the enabler provides trust metric to the client, which will then visualize the results.

The test has the following phases:

1) Start kafka

```
$ cd /usr/local/src/kafka_2.11-0.8.2.2
$ ./bin/zookeeper-server-start.sh config/zookeeper.properties
$ ./bin/kafka-server-start.sh config/server.properties
```

2) Start Trust Metric Enabler (release 2)

```
$ python –m trust_metric/tme_kafka/tme_kafka
```

Figure 12: Start Trust Metric Enabler

3) Start client (‘service provider’) - test client which is delivered with the package

```
$ python –m trust_metric/tme_kafka/sp
```

Figure 13: Start service provider

4) Feed example measurements to TME through Kafka – using the test program included in the package. (The measurements are similar to the ones that could be proveded by the micro-segment monitor.)

```
$ python –m trust_metric/tme_kafka/monitor
```

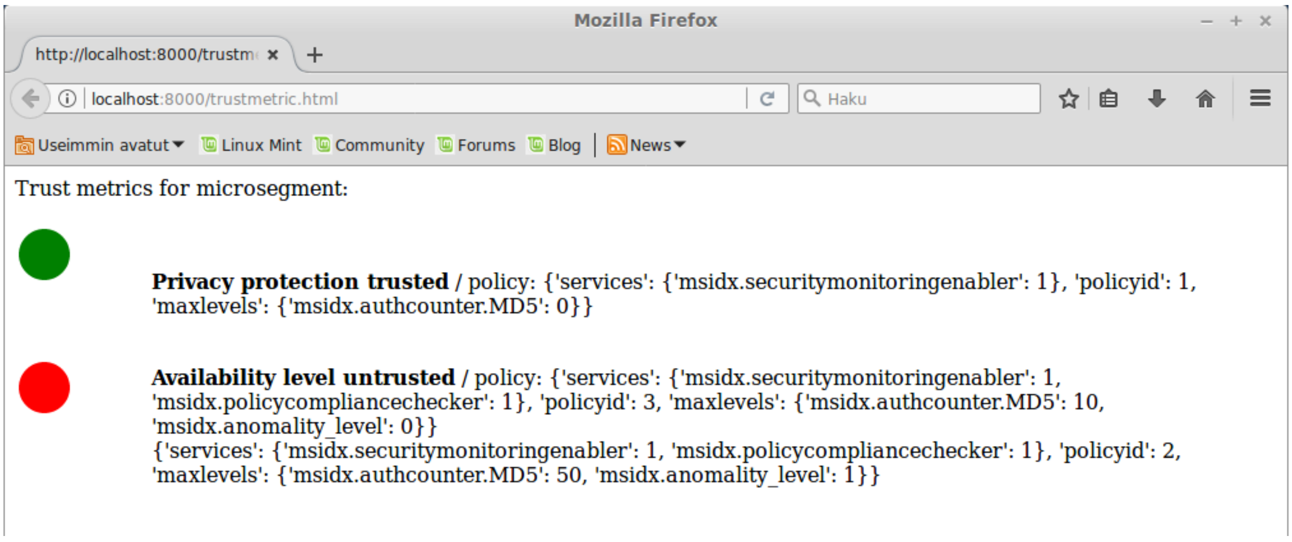
Figure 14: Start monitor

The output of sp.py is shown below[1].

[1] The client first sends the policies and receives false reply to each of them (as at initial state none of the policies where matched. Then (when monitor.py is running) TME starts to receive notifications, which change TME’s interpretation of the trust situation. The availability related metrics remain false (as the amount of MD5 authenticated devices is large) but the privacy requirements are now fulfilled and the client outputs that privacy related trust metric is ‘True’.

```
send {'services': {'msidx.securitymonitoringenabler': 1, 'msidx.policycompliancechecker': 1}, 'policyid': 2, 'maxlevels': {'msidx.authcounter.MD5': 50, 'msidx.anomality_level': 1}}

send {'services': {'msidx.securitymonitoringenabler': 1}, 'policyid': 1, 'maxlevels': {'msidx.authcounter.MD5': 0}}

send {'services': {'msidx.securitymonitoringenabler': 1, 'msidx.policycompliancechecker': 1}, 'policyid': 3, 'maxlevels': {'msidx.authcounter.MD5': 10, 'msidx.anomality_level': 0}}

availability related trust metric: False

privacy related trust metric: False

high availability related trust metric: False

privacy related trust metric: True

privacy related trust metric: True
```

The client also generates a web page indicating how well the trust policies it set where met. The html page is available in the working directory from where it can be opened with a web browser. Alternatively, the web page can be also shared using a web server. For instance, Python's web server can be used by starting it in the same directory (cd /usr/local/lib/python2.7/dist-packages/trust_metric/tme_kafka/; sudo python -m SimpleHTTPServer).



Trust metrics for microsegment:

Privacy protection trusted / policy: {'services': {'msidx.securitymonitoringenabler': 1}, 'policyid': 1, 'maxlevels': {'msidx.authcounter.MD5': 0}}

Availability level untrusted / policy: {'services': {'msidx.securitymonitoringenabler': 1, 'msidx.policycompliancechecker': 1}, 'policyid': 3, 'maxlevels': {'msidx.authcounter.MD5': 10, 'msidx.anomality_level': 0}}
{'services': {'msidx.securitymonitoringenabler': 1, 'msidx.policycompliancechecker': 1}, 'policyid': 2, 'maxlevels': {'msidx.authcounter.MD5': 50, 'msidx.anomality_level': 1}}

Figure 15: Example web view of trust metrics acquired from Trust Metric Enabler

5 Abbreviations

5G-PPP	5G Infrastructure Public Private Partnership
CEP	Complex Event Processing
JSON	JavaScript Object Notation
TME	Trust Metric Enabler

6 References

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

[2] Kafka API http://kafka.apache.org/documentation.html#api

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

[4] Kafka download site. http://kafka.apache.org/downloads.html

[5] Kafka quickstart guide. http://kafka.apache.org/documentation.html#quickstart

[6] 5G-ENSURE. Deliverable D3.5 - 5G PPP security enablers technical roadmap (Update). 2017. https://5gensure.eu/sites/default/files/5G-ENSURE_D3.5%205G-PPP%20security%20enablers%20technical%20roadmap%20%28Update%29.pdf

[7] 5G-ENSURE. Deliverable D3.6 - 5G PPP Security Enablers Open Specifications (v2.0). 2017. https://5gensure.eu/sites/default/files/5G-ENSURE_D3.6%205G-PPP%20security%20enablers%20open%20specifications%20%28v2.0%29.pdf