Enabler Manual
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
Authors	NEC: Felix Klaedtke, Mihai Moraru, Alessandro Sforzin, Hien Truong

Document Version	**Date**	**Change(s)**	**Author(s)**
0.1	09.06.2017	Copy in D3.4	Felix Klaedtke
0.2	24.06.2017	Update and small changes	Felix Klaedtke, Alessandro Sforzin, Hien Truong
0.3	15.8.2017	Reviewed	Aleksi Dahl
0.4	15.8.2017	Revised	Felix Klaedtke

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 D3.8. It describes how one of the security enablers that are developed within the work package WP3 of the 5G-ENSURE project is installed and administrated. Furthermore, this manual contains a user guide of the respective security enabler.

Disclaimer

Copyright notice

Contents

1 Introduction 5
2 Installation and Administration Guide 5
2.1 System Requirements 6
2.2 Enabler Configuration 6
2.3 Enabler Installation 7
2.4 Troubleshooting 7
2.4.1 Installation 7
2.4.2 Running 7
2.4.3 Connecting to Mininet 8
3 User and Programmer Guide 8
3.1 User Guide 8
3.1.1 Overview 8
3.1.2 Access Control Policies 9
3.1.3 Inspecting the Log 10
3.2 Programmer Guide 10
4 Unit Tests 10
4.1 Unit Test 1 10
4.2 Unit Test 2 11
5 Abbreviations 12
6 References 12

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 [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 [9, 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, 10] 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 [9, 3]. This manual describes how the reference monitor for ONOS is installed, administrated, and used. We refer to the ONOS webpage (http://onosproject.org/) 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 is located in the directory `/tmp`.

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 `$ apps -s -a` and `$ 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, 8]. 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.

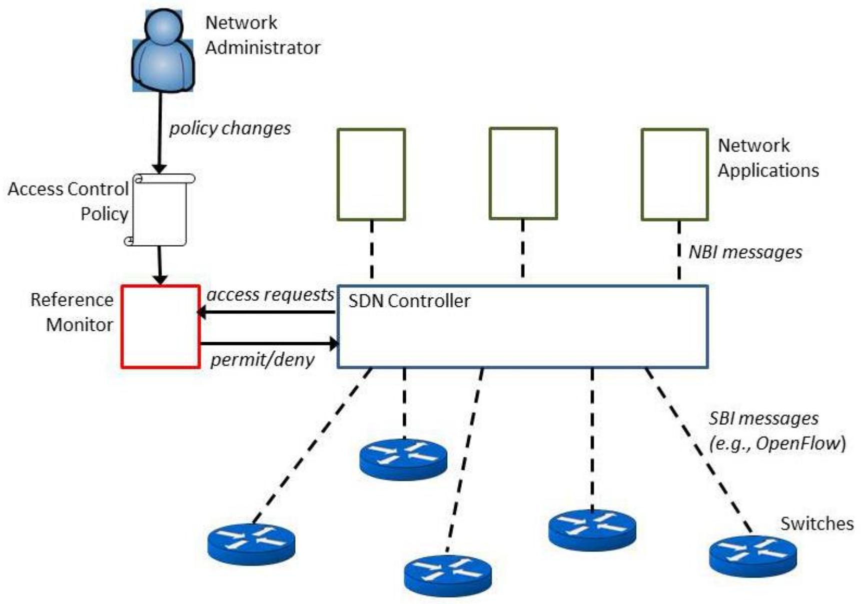

Figure 1 SDN network components and their interactions.

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 supports one to build a tamperproof and verifiable reference monitor. This is based on the scheme's simplicity and since it is focused. 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 entries 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.