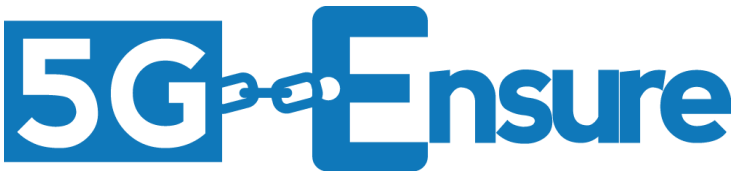


Deliverable D3.8
5G-PPP Security Enablers Documentation (v2.0)
Enabler Flow Control

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	TCS: Filippo Rebecchi, Mathieu Bouet	

Document Version	Date	Change(s)	Author(s)
0.1	02.06.2017	Created template	Felix Klaedtke
0.2	07.08.2017	Initial version of the manual	Filippo Rebecchi
0.3	14.08.2017	Reviewed	Aleksi Dahl
0.4	21.08.2017	Document Revised according to review	Filippo Rebecchi

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 Flow Control Enabler is installed and administrated within the work package 3 of the 5G-ENSURE project. Furthermore, this manual contains a user guide of the security enabler.

Disclaimer

Copyright notice

Contents

1. Introduction

The *Flow Control* enabler described in this section proposes a means to detect and mitigate in-network threats targeting critical VNFs, both in the control and data planes, logically deployed in a larger core network. To this end, a software switch embedding the capability to protect the virtual network interfaces is added to the network's data plane as a gateway to the targeted critical VNF. In particular, the *Software switch* can be deployed and instructed at runtime, by exploiting SDN concepts – namely, through a dedicated Flow Controller. The Software switch is capable of automatically detecting network-based security threats and to act appropriately to minimize their impact (e.g., applying rate limiting policies, black holing or discarding certain flows coming from suspicious hosts). As a result, in order to protect deployed VNFs (typically offering a network function or service application) from network-based attacks, the effects of which can be harmful for the entire network - a large scale DDoS attack targeting a critical function in the core of the network (e.g., HSS database) - three steps are typically required 1) traffic monitoring; 2) threat detection; and 3) threat mitigation. We will detail these three steps in the following:

1. In order to perform traffic monitoring, the forwarding device (e.g., the Software switch in this enabler) must be able to retrieve statistics on the packets flowing through, such as host/port binding, and to report them to a controller to take the ultimate decision on the dangerousness of that traffic. Data collection is particularly tricky here, because we would like the Software switch to track efficiently any flow feature with fine-grained information without overloading too much the forwarding performance and the control channel towards the controller.
2. Anomaly detection is performed on the monitored data in order to discriminate heavy hitters and suspicious activities from the legitimate traffic, and also to identify attacker and target tuples. Since this task is complex and requires correlating several flows in real-time, detection is typically done at the controller, by frequently requesting Software switch for flows statistics e.g., through OpenFlow messages. For completeness, it should also be noted that some recent approaches propose offloading a part of the detection process directly to the switches in order to have a quicker reaction process.
3. Finally, threat mitigation should set up the most appropriate reactions on the data-plane to counter the detected attack (e.g., by black-holing certain flow or rate limiting some hosts), by issuing and installing flow rules.

2. Installation and Administration Guide

2.1 System Requirements

This enabler runs on any host (preferably Linux) with Docker installed. The Flow Control enabler is provided as a Docker container in order to ease the installation and deployment steps.

2.2 Enabler Installation

It is necessary to retrieve the container image from the 5G-Ensure Artifactory repository at fivegensure-docker-virtual/flow_control/1.0.1/ .

```
export TAG=1.0.1
export IMAGE_NAME=flow_control
```

```
export DOCKER_REGISTRY= fivegensure-docker-virtual.artifact.b-com.com
docker pull $DOCKER_REGISTRY/$IMAGE_NAME:$TAG
```

The container images are now available to be deployed.

```
docker run -it -d --privileged -h flow_control --name flow_control fivegensure-docker-virtual.artifact.b-com.com/flow_control:1.0.1
```

2.3 Enabler Configuration

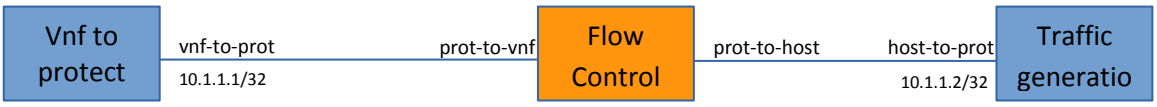
Once the container is up and running, you can attach to it:

```
docker attach flow_control
```

Modify the script located in `app/execution/start.sh` as follow:

```
screen -dmS ofdatapath ofdatapath -v -d 000000000001 -i intfName1,intfName2,... punix:/var/run/sock --no-slicing > ofd.log 2> ofd.log
```

where intfName1, intfName2 and so on are the names of the datapath interfaces connected to the enabler (i.e., the interfaces from/to where we want to provide the protection of the enabler). The names and the number of the interfaces may change depending on the deployment setup. However, a classic deployment could be the following:



In this case, intfName1=prot-to-vnf and intfName2=prot-to-host

2.4 Troubleshooting

The script start.sh must be modified as explained in Section 2.3.

In case of malfunctioning perform the following checks:

1. `screen -r` should prompt three existing screens.
2. Attach to the ofdatapath screen by doing `screen -r ofdatapath` and check that the ofdatapath is running correctly and attached to the correct dataplane interfaces (as configured in start.sh).
3. By Ctrl-a n go to the ofcontrol screen and check that it has discovered both the datapath and the controller, and the communication between them is ok (No error messages displayed).
4. By Ctrl-a n go to the controller screen and check that it has configured the software switch and it is receiving periodic reports from it.

3. User and Programmer Guide

3.1 User Guide

Before starting the enabler, application parameters can be configured by modifying the file located in app/controller/etc/ryu/ryu.conf.

MONITORING_SLEEP_TIME (default 5): in second, is the interval between each request for counter performed by the controller. Longer intervals generally improve performance at the cost of a reduced reactivity.

ENABLE_NETWORK_WIDE_SYN_DETECTION (default FALSE): Boolean, MUST be set at FALSE for this application.

LOCAL_SYN_DDOS_ACTIVE_THRESHOLD (default 500): integer, is the lower number of new SYN per MONITORING_SLEEP_TIME before mitigation actions are taken.

LOCAL_SYN_DDOS_INACTIVE_THRESHOLD (default 300): integer, is the upper number of new SYN per MONITORING_SLEEP_TIME before mitigation actions are removed (when activated).

FIXED_TIMEOUT_TIME (default 30): in seconds, is the TTL of the mitigation rules installed at the switch when a threat is detected.

Precision (default 4): integer, parameter to tune the sensitivity of the detection algorithm. A larger value reduces the false positives at the cost of detecting only heavier attacks.

To launch the enabler, execute the script:

```
bash /app/execution/start.sh
```

At this point there should be three detached screens running. By doing:

```
screen -r
```

Three screens should appear with the following names

- controller: this is the SDN controller running the flow control application.
- ofcontrol: this is the daemon controlling the communications between the SDN controller and the software switch.
- ofdatapath: this is the process actually doing the switching that is programmed and controlled by the SDN controller.

3.2 Programmer Guide

N/A

4. Unit Tests

4.1 Information about Tests

The test should assess the good functioning of the Enabler. Since it will be deployed as a gateway for traffic going to/from critical VNF it should guarantee that bidirectional traffic can flow. In addition to this, the Flow Control enabler should be capable of counting traffic features at the controller. In order to do this, it is necessary to establish a connection between the data path – in charge of performing data-plane switching – and the controller – in charge of monitoring traffic and mitigating threats.

4.2 Communication between the data path and the controller

This test ensures the correct instantiation of the different processes composing the enabler. It guarantees that the software switch has been correctly configured and that the controller receives periodic updates.

Once the enabler is started attach to it by `docker attach flow_control` and then go to the controller process (`screen -r controller`).

The logs should resemble to the following:

```
Starting DDoS detection ...
Sleeping for 5 seconds..
Configuring switch 1...
Setting up Table 0 ...
Setting up Table 1 ...
Setting up Table 2 ...
Setting up Table 3 ...
Setting up Table 4 ...
Setting up Table 5 ...
Setting up Table 6 ...
Setting up Table 7 ...
Done
Setting up Table 8 ...
Loading Normal mode Table 6 (MATCH) on datapath 1.
Done.
Loading Normal mode Table 7 (SYN counter) for datapath 1.
Sleeping for 5 seconds..
1502118465 07.08. 15:07:45 New flow count for switch 1 is 0
1502118466 07.08. 15:07:46 Entropy values for switch 1 : 0.768870 0.672680
0.563450 0.000000
==========================================
Sleeping for 5 seconds..
1502118470 07.08. 15:07:50 New flow count for switch 1 is 5
1502118470 07.08. 15:07:50 Entropy values for switch 1 : 0.777520 0.725140
0.882390 0.765700
==========================================
Sleeping for 5 seconds..
1502118475 07.08. 15:07:55 New flow count for switch 1 is 8
1502118475 07.08. 15:07:55 Entropy values for switch 1 : 0.759660 0.693230
0.572170 0.708650
```

```
==========================================

Sleeping for 5 seconds..

1502118480 07.08. 15:08:00 New flow count for switch 1 is 163

1502118480 07.08. 15:08:00 Entropy values for switch 1 : 0.819930 0.708050
0.721130 0.615140

==========================================
```

4.3 Communication between external host and the protected VNF

The test ensures that the critical VNF is accessible from the outside. From an external host it MUST be possible to reach the VNF.

For instance, from the external Host it should be possible to reach a VNF located at 10.1.1.1:

```
ping 10.1.1.1
```

```
PING 10.1.1.1 (10.1.1.1): 56 data bytes

64 bytes from 10.1.1.1: icmp_seq=0 ttl=64 time=2.825 ms

64 bytes from 10.1.1.1: icmp_seq=1 ttl=64 time=2.164 ms

64 bytes from 10.1.1.1: icmp_seq=2 ttl=64 time=1.252 ms

64 bytes from 10.1.1.1: icmp_seq=3 ttl=64 time=0.968 ms

64 bytes from 10.1.1.1: icmp_seq=4 ttl=64 time=0.792 ms

64 bytes from 10.1.1.1: icmp_seq=5 ttl=64 time=0.954 ms

64 bytes from 10.1.1.1: icmp_seq=6 ttl=64 time=0.766 ms

64 bytes from 10.1.1.1: icmp_seq=7 ttl=64 time=0.716 ms

64 bytes from 10.1.1.1: icmp_seq=8 ttl=64 time=0.810 ms

64 bytes from 10.1.1.1: icmp_seq=9 ttl=64 time=0.662 ms

64 bytes from 10.1.1.1: icmp_seq=10 ttl=64 time=0.621 ms

64 bytes from 10.1.1.1: icmp_seq=11 ttl=64 time=5.222 ms

64 bytes from 10.1.1.1: icmp_seq=12 ttl=64 time=5.243 ms
```

Depending on the setup, note that it may be needed to add routing instruction at the emitting host in order to route the VNF through the enabler. For a Linux machine this can be done using:

```
ip route add 10.1.1.1/32 dev host-to-prot
```

5. Abbreviations

5G-PPP	5G Infrastructure Public Private Partnership
NFV	Network Functions Virtualization
VNF	Virtual Network Function
SDN	Software-Defined Networking

DDoS	Distributed Denial of Service
HSS	Home Subscriber Server
EPC	Evolved Packet Core

6. References

[1] 5G-Ensure D3.5 5G-PPP security enablers technical roadmap (Update)