Software Defined Networks and Network Function Virtualisation Testbed within FIRE+

Virtual Security Functions on a Federated and Orchestrated Virtualisation Testbed

Security as a Service provided by the SoftFIRE Security Manager

April 2018

Table of Contents

List of Figures

1 Introduction

Network Functions Virtualisation (NFV) [1] and Software-Defined Networking (SDN) [2] are beginning to have a key role in the telecommunications industry. The capabilities offered by NFV introduce new features to network services, with the possibility to virtualise and extend a network infrastructure by means of implementing network functions as software without the need for specialised hardware. On the other hand, SDN facilitates flexible and dynamic network management. It enables dynamic programmability of networking components. SDN also provides the possibility to add more flexibility to a cloud infrastructure. Together, these two virtualisation technologies, i.e. NFV and SDN, allow network operators to manage their network resources flexibly, whilst providing control and customisation of network services to end users.

In a context where a cloud platform is assembled as a federation of multiple component testbeds, it is necessary to make the platform's individual components *inter-operable*, even if they are based on different technologies, different management rules, and different networking capabilities. The SoftFIRE federated virtualisation platform [3] offers such interoperability capabilities by providing a heterogeneous environment in which a common access interface for all testbeds is exposed. On this basis, platform users can leverage NFV and SDN technologies in their virtualisation experiments.

One of the crucial aspects in the context of a federated virtualisation platform is security, in terms of:

1. Platform access,
2. Running set of network services initiated and used by platform users.

In project SoftFIRE [3][1], platform access is restricted to authorised users. Towards this, the project has set up a point-to-multipoint VPN in order to prevent access to the platform by an external subject.

In regard to security of running network services, several measures have been taken, consisting of a *monitoring tool* that can provide a dynamic and flexible method to configure security policies. Initially, this security monitoring tool was configured to work with already deployed physical resources in each testbed, which act as security probes that are able to perform IDS/IPS functions, and monitor the whole platform. However, because of the lack of scalability in this solution, i.e. reliance on physical security units, as well as dynamic adaptability of such probes, this solution was abandoned. Instead, the Project decided to provide *security-oriented resources* to its end-users, i.e. experimenters. This approach offers security-oriented services that can secure and protect network services, based on user requirements. As a result, SoftFIRE Middleware's [4] *Security Manager* [5] and the concept of *Security as a Service* (SECaaS) [6] as a virtualisation solution have emerged.

This white paper describes an overview of the security measures and policies put in place on the SoftFIRE platform, outlining how to ensure a sufficient level of security and privacy. The paper introduces the Security as a Service concept, and explains how the Security Manager realises it.

[1] SoftFIRE: Software Defined Networks and Network Function Virtualization Testbed within FIRE+.

2 SoftFIRE's Transition from SFA to TOSCA

SoftFIRE offers a federated heterogeneous experimentation infrastructure that allows to develop services and applications in the NFV and SDN domains and in the perspective of 5th Generation (5G) mobile network architectures. The platform is a loose federation of already existing component testbeds owned and operated by distinct organizations for purposes of research and development. Its main scope consists of providing the users, called *experimenters*, with an infrastructure where they can build and run their own experiments. It is a project proposed by the FIRE (Future Internet Research and Experimentation) initiative [7] with the aim of being open and flexible, in order to integrate different aspects of each individual testbed. To achieve this aim, SoftFIRE follows the standards defined by ETSI (European Telecommunications Standards Institute) to design the architectural aspects of the platform that are necessary for using NFV and SDN technologies. Open Baton [8] and OpenStack [9] represent the standard de-facto implementations of the ETSI NFV architecture [10] for building and federating the SoftFIRE infrastructure.

One of the major objectives of the SoftFIRE project is to make the interaction between external experimenters and the platform easy and secure. Since the project is part of ICT FIRE [7] research activities, compatibility/alignment with FIRE is necessary. During the initial phase of the project and before the beginning of its 1st Wave of Experiments [11], SoftFIRE has deployed FITeagle (Future Internet Testbed experimentation framework) [12], which is an open-source FIRE tool that allowed experimenters to access the resources offered by the platform, providing reservation, acquisition, monitoring, and release of the platform's arbitrary resources needed for building their experiments. Therefore, it was the main entry point for experimenters to deploy their experiments. The distributed and heterogeneous nature of the platform meant that FITeagle followed the standard Slice-based Federated Architecture (SFA) [13], and realised interoperability of resources with different technologies.

The main reason to use FITeagle in SoftFIRE initially was to support and control the experiment life-cycle. However, the Project had experienced that SFA would not fully support the usage and exploitation of virtualization capabilities when exposing all the necessary features and functionalities of the resources provided by the platform. Such exposure was actually possible with TOSCA (Topology and Orchestration Specification for Cloud Applications) [14], which is a cloud-based reference model for modelling network services. Using TOSCA, FITeagle could be replaced by a new single point of platform access offered to the experimenters: the *Experiment Manager* [15]. The motivations and approach used for this move to TOSCA has been explained in [16]. This has not only made it possible to define a variety of resources and enable configuration options for each resource in an extensible open-source code-base, but also made it easier and quicker for experimenters to prepare and deploy their experiments. This approach has also had an impact on the security solutions provided by the SoftFIRE platform. Specifically, there was the need to introduce security mechanisms for offering security features to experimenters as well as to secure the Experiment Manager itself. In the next sections, the concept of Security as a Service (SECaaS) is presented and discussed.

3 SoftFIRE's Security Approach

The first security aspect that SoftFIRE took into consideration was to make the interaction between an experimenter and the platform as secure as possible. Experimenters must connect to the platform's Virtual Private Network (VPN) server, running OpenVPN [17]. This means that experimenters must authenticate with and have authorisation to access the platform.

Access credentials were provided to only those experimenters whose proposals were accepted by SoftFIRE. After the approval of an experiment proposal, experimenters can register at the Project's web access portal and receive a VPN certificate, which grants them the right to access the SoftFIRE VPN and manage their experiment lifecycle.

The second security feature that SoftFIRE has put in place is for the NFV and SDN capabilities of the cloud environment, i.e. the virtualisation resources used by experimenters, whilst taking into account the necessary support for flexibility, scalability, and a level of central control. This was possible by adopting the concept of *Security as a Service* (SECaaS) [6].

3.1 Security as a Service

Security as a Service (SECaaS) is a business model for security management, in which a service provider integrates several of their security services into a single infrastructure with a cost-effective solution compared to what most individuals or corporations can provide on their own. It is based on the *Software as a Service* model and considers security services and needs, whilst not requiring any on-premise hardware. Normally, security services often include authentication services, anti-virus, anti-malware/spyware, intrusion detection systems (IDS), intrusion prevention systems (IPS), and so on.

3.1.1. Why SECaaS?

The choice to leverage the concept of SECaaS is due to two main reasons: (*i*) to evolve network security services towards virtualisation, and (*ii*) to provide on-demand security solutions to end users.

- ***Virtualisation in Security Services:*** **Save time and money!**

  The traditional approach to deploy physical network security functions is time-consuming and error-prone, due to the need for additional specialised hardware and increased maintenance efforts and costs. It entails inefficient service provisioning, and represents a serious problem for large federated cloud networks due to potential changes in security policies and network topology. In addition, without SDN, network functions are configured statically. This leads to lack of flexibility and adaptability for the network, and a consequent change in network function characteristics and behaviour.

- ***On-demand Security:*** **Benefits of NFV and SDN**

  In a federated cloud network, NFV and SDN offer a flexible and scalable approach to build on-demand solutions that are readily available to platform users, which also reduce labour and cost. NFV allows to deploy, manage, and provide different virtual network functions

(VNF) in the form of security functions, and in a simplified manner. SDN allows to create dynamic networks that are able to adapt to on-demand configuration changes, with a centralised control of the network's topology. With SDN, it is also possible to collect traffic statistics in real-time. Together, NFV and SDN enables network services that can consist of a variety of security resources, effectively minimising the number of security appliances that would normally be needed in a physical infrastructure.

In SoftFIRE, SECaaS concept has been realized using three main entities:

- **Security resources**
  These are specialised VNFs with a specific purpose to deliver security-related services, e.g. a firewall or an intrusion detection system.
- **Security chain**
  This represents a special case of Service Function Chaining (SFC) [18] that links two or more VNFs using SDN, and improves the performance of NFV solutions, making them dynamic. Specifically, it is a chain of security resources.
- **Security Manager**
  This is the Middleware component [4] which is in charge of managing the life-cycle of security resources. It enables creation, management, and termination of security resources. Such operations are exposed to experimenters via the Experiment Manager [15] interface.

3.1.2. Security resources

A *security resource* is essentially a security function which can be instantiated as a VNF instance. A network service can use multiple security resources, providing an experiment with one or more specific security capabilities. SoftFIRE provides two network security functions:

- **Firewall**
  Firewall is a network security function that allows to perform network access control via security policy definitions, including filtering and forwarding of network traffic. SoftFIRE provides two kinds of these resources: (*i*) Uncomplicated Firewall (UFW) [19], which a firewall configuration tool that allows to define simple security rules to filter and route network packets, and (*ii*) pfSense [20], which is a free and open-source firewall that has more complex firewall capabilities.

- **Network IDS/IPS**
  Network Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) are network security functions that allow mechanisms for analysing and identifying network traffic. SoftFIRE provides these functions with Suricata [21], which is a free and open-source network threat detection engine that is able to perform intrusion detection and prevention, as well as network security monitoring.

4 Technical Aspects

In the previous section, the SECaaS concept is introduced to justify the need for security resources for virtual network services. Security Manager [5] is the implementation in SoftFIRE to realise this concept. It is primarily based on the NFV architecture that the SoftFIRE Middleware makes available through its NFV Manager [22], not only to experimenters but also to the other SoftFIRE Middleware managers [4]. It is considered as a specialised NFV manager, handling security resources.

In this section, the implementation choices and details of the Security Manager are explained. Furthermore, two use cases are presented to explain how a security resource can be integrated in a complex experiment including VNFs that require some sort of traffic chaining in order to exploit security services.

4.1 Security Manager

Security Manager [5] is one of the SoftFIRE Middleware [4]'s sub-manager components. It can be considered as a specialised form of the NFV Manager [22] in that it deploys security VNFs, namely:

- A firewall service via the firewall resource (UFW on-board) [19],
- A combined IDS and IPS service via the Suricata resource [21],
- A combination of gateway, firewall, and routing service via the pfSense resource [20].

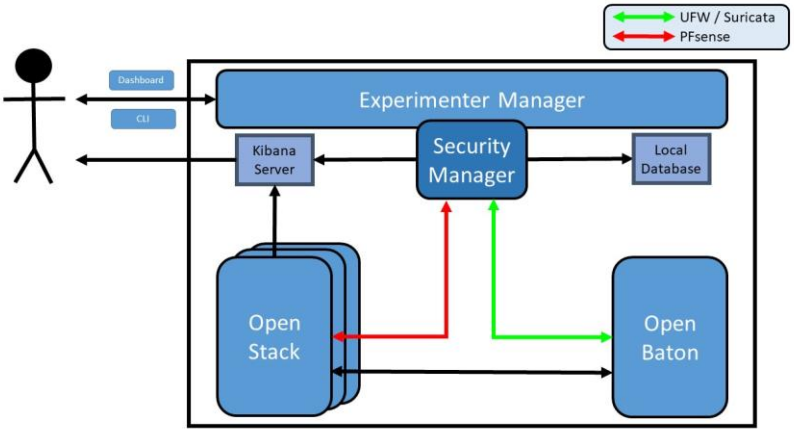

Figure 1. Security Manager architecture

Figure 1 shows the architecture of the Security Manager. Similar to the NFV Manager, to allocate and deploy each security VNF, Security Manager contacts Open Baton. It interacts with Open Baton [8] in order to deploy security VNFs (and eventually with OpenStack), and directly with OpenStack [9][23] in case of pfSense.

When a user requests a network service that includes a security resource, this triggers the Experimenter Manager to forward the request to the Security Manager. Specifically, this request contains: (*i*) the type of the security resource, (*ii*) the ID of the experimenter, and (*iii*) requested resource configurations. The Security Manager then stores some information regarding the

experiment in a local database, and interacts with Open Baton or directly with Open Stack, based on the kind of security resource requested by the user.

Security Manager can also provide a monitoring dashboard to store and analyse the logs generated as a result of the deployed security VNF activities. This monitoring tool is based on the Elasticsearch-Logstash-Kibana stack, known as ELK Stack [24].

The Security Manager handles the following life-cycle phases:

- *Validate Resource:* The resource definition file is checked and all the properties that it contains are validated. Security Manager then proceeds to the next phase only if there are no syntax errors or inconsistent values. Otherwise, it raises a *ResourceValidationError* and rejects the experiment.
- *Provision Resource:* During this phase, the selected security resource is allocated and deployed on OpenStack. Security Manager needs to upload the archive containing the security resource definition in TOSCA syntax to Open Baton. Then, Open Baton mediates the interaction between the Security Manager and OpenStack. To provide the experimenter with access to the security VNF, the experimenter public key is uploaded to the virtual machine hosting the security VNF.
- *Terminate Resource:* This phase is triggered when the experimenter decides to terminate the network service containing one or more security resources. Using the resource ID previously saved, Security Manager requests Open Baton to release all OpenStack virtual resources associated with the security resource.

It must be noted that the pfSense resource must be managed in a different way compared to how Firewall and Suricata resources are managed. This is because this technology is built on a FreeBSD system [25] that has issues with interfacing with Open Baton. In order to handle all its lifecycle phases, Security Manager interacts directly with OpenStack. This means that some actions must be manually managed, which are transparent to the experimenter.

When the requested security resource is allocated and deployed as a VNF, the Manager takes some additional steps to provide a usable resource to the experimenter. Specifically, a security resource needs to be chained to those other locations/entities that it is supposed to protect by declaring routing rules. For instance, a firewall would be useless if not connected to the network endpoint that it needs to monitor. This requires enabling an important capability, which is *port forwarding*. Without this capability, a virtual machine that runs a security VNF would be unable to forward traffic, e.g. a firewall would not be able to send the traffic to its intended destination after having inspected it.

The port forwarding mechanism is by default disabled by OpenStack in order to prevent IP address spoofing. Such measure prevents a virtual machine from spoofing the identity of another and eventually modifying or dropping its packets. To enable port forwarding and configure it, Security Manager forwards a request to OpenStack through an API call. This request is performed for each virtual machine in each VNF belonging to the network service that needs to be secured. Essentially, Security Manager manages port forwarding by port security policies [26]. By editing these policies, a properly configured firewall resource can intercept and analyse the traffic generated by a Virtual Deployment Unit (VDU) belonging to the same tenant and network.

With traffic forwarding enabled, experimenters can manipulate traffic routing in order to include security functions on traffic flows and to let packets reach security VNFs. This can be achieved in various ways, yet the ideal way is to use an SDN controller provided by the SoftFIRE SDN Manager. This method is preferable as it exploits a feature integrated with the platform and enforces the concept of *platform programmability*. With SDN, it is possible to divert, mirror, or drop traffic flows programmatically by sending instructions to the SDN controller that is in charge of interacting with virtual switches installed in OpenStack. Many SDN controller implementations are available as open-source or proprietary solutions; SoftFIRE provides the following two: (*i*) OpenDaylight (ODL) [27], and (*ii*) OpenSDNCore [28].

5 Security as a Service Use Cases

In this section, two use cases are described in order to show how the SECaaS concept is applied to a real context. Each use case is composed of a security resource and some other resources.

In order to use the security functionalities provided by a security VNF, it is necessary to chain VNFs, which is possible with SoftFIRE's SDN Manager. With an SDN controller, it is possible to manage Open Virtual Switches (OVS) [29]. After an experimenter requests an SDN resource provided by the platform, traffic can be directed to the security VNF without modifying the routing tables of VMs.

Experiments can choose between two SDN controller implementations available on the platform: OpenSDNCore [28] and OpenDaylight [27][2]. In the below example, OpenDaylight (ODL) and its OpenFlow API plugin are used, as ODL is considered to be a *baseline project* which many other controllers are built on.

5.1 Flow Match Criteria and Actions in SDN

Traffic flows can be manipulated by sending HTTP requests to the SDN controller. The content of the request is a JSON formatted file. In order to edit a flow, it is necessary to identify that particular traffic by specifying a rule inside the request. Using the SDN terminology, a rule is defined as a "match" and can be declared with a classic ACL match criteria based on protocol types, port numbers, MAC addresses, and/or IP addresses, etc.

When a security VNF needs to protect another VNF, it needs to capture all the IPv4 packets directed to a specific IP address. In the following example as illustrated in Figure 2, a "match" rule for IP address 10.0.10.2 is created.

```
"match": {
    "ipv4-destination": "10.0.10.2/24",
    "ethernet-match": {
        "ethernet-type": {
            "type": 2048
        }
    }
}
```

Figure 2. Sample SDN flow rule to match a specific destination IP address.

[2] It must be noted that the platform is extensible, which means experimenters can integrate their own SDN controllers, considering the API that SDN Proxy implementations in SoftFIRE are followed.

It is possible to define more complex match rules when handling traffic flows more selectively. Experimenters can achieve complex routing configurations by combining more *match fields*. For example, it is possible to select all HTTP traffic flows generated by a virtual machine, regardless of its MAC address, by matching the source VM IP address and the TCP destination port 80. This is shown in Figure 3.

```
"match": {
    "ipv4-source": "10.1.2.3/24"
    "tcp-source-port":"25364",
    "tcp-destination-port":"80"
    "ip-match":{
        "ip-protocol":"6"
}
```

Figure 3. Sample SDN rule to match for all HTTP traffic from a specific source IP address.

A match criterion is followed by an "instructions" declaration, which includes the set of actions that the SDN controller applies to the matched traffic. Each packet flow has one or more actions associated it, which get executed once a packet matches the rule. In this case, the matched packets are forwarded to the security VNF; i.e. the "instructions" must contain an action that contains the security VNF's IP address as the destination. In the following example (see Figure 4), traffic is redirected to IP 200.71.9.52/32, where the security VNF runs:

```
"instructions": {
    "instruction": [
        {
            "order": 0,
            "apply-actions": {
                "action": [
                    {
                        "order": 0,
                        "set-field": {
                            "ipv4-destination": "200.71.9.52/32"
                        }
                    }
                ]
            }
        }
    ]
}
```

Figure 4. SDN instruction to forward traffic to a specific destination IP address.

These static SDN rules can be dynamically created or changed by a component that implements the traffic logic. In this use case, the component that makes these decisions is the security VNF itself.

By combining the features of the SDN controller and Security Manager, it is possible to dynamically modify packet flows and enforce security policies. For example, by choosing Suricata as a security resource, it is possible to perform Deep Packet Inspection (DPI) on traffic flows that go through a local network. When a malicious sequence of packets is identified by the security resource, a "<match, instructions>" rule pair is generated and forwarded to the SDN controller in order to drop such malicious packets.

5.2 Use Cases for SDN-based SECaaS

In this section, two use cases are presented, in which security policies are enforced on a set of user resources, by exploiting SDN capabilities with OpenFlow [30] and Security Manager.

5.2.1. Use Case #1: Port based traffic filtering

In this use case, the considered network service is composed of a firewall resource and an NFV resource with two Ubuntu virtual machines, depicted as Source_VM and Destination_VM in Figure 5. The objective is to deploy these virtual machines in a virtual LAN where FTP traffic is blocked; i.e. packets on port 21 are dropped by the firewall resource's enforced SDN rule.

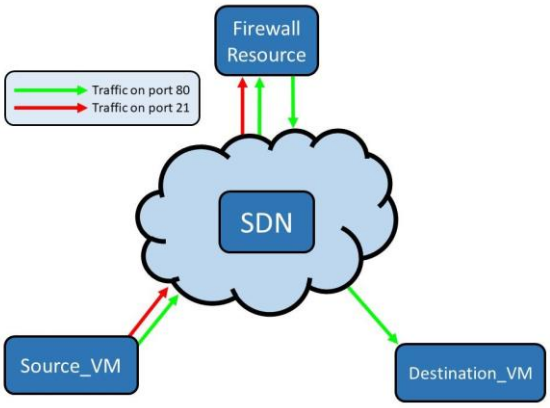

Figure 5. Use Case #1: Firewall filtering traffic flows based on application ports.

To implement this scenario, a traffic rule can indeed be statically set at the beginning. However, such an approach would not be ideal, as the firewall VNF can make decisions dynamically and set routing rules by contacting the SDN controller, which is a better fit for dynamic environments where firewall operations do not need to be statically kept but modified when needed. In other words, such security policies (converted into SDN flow rules) can now be applied on-demand, and based on the traffic sent to/received from virtual machines. Furthermore, as opposed to static rules, the capability to impose dynamic security policies makes it possible to apply security policies to a subset of all running virtual machines, and modify this subset, depending on changing traffic characteristics.

In this example, as depicted in Figure 5, a source virtual machine (called Source_VM) generates two traffic flows: (*i*) *Flow 1:* legitimate traffic on port 80, and (*ii*) *Flow 2:* illegitimate traffic on port 21. To block traffic on port 21 and allow traffic on port 80, the firewall resource applies an SDN rule such that packets on port 80 get dropped, whereas those on port 21 are directed to the destination VM.

5.2.2. Use Case #2: Traffic mirroring

In this use case, a Suricata resource is used to detect malicious traffic flows and trigger SDN traffic flow changes to take actions against them. The Suricata resource sets more complex SDN actions to be applied to traffic flows. Besides the Suricata resource, three virtual machines are involved: (*i*) a source VM that produces legitimate traffic, (*ii*) a second source VM that produces malicious traffic, and (*iii*) a destination VM.

To guarantee better network performance, during the detection phase, incoming traffic is mirrored exploiting the mirroring feature provided by the SDN controller. In this way, the original traffic is still able to reach the intended destination without any delay, as shown in Figure

6 (a). Then, Suricata is able to inspect the mirrored traffic and react when a malicious flow is detected, as shown in Figure 6 (b), and then drop packets from the malicious VM.

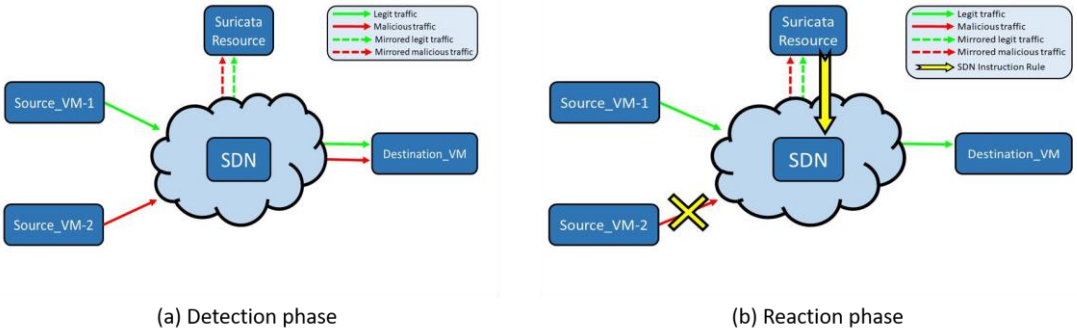

Figure 6. Use Case #2: Traffic mirroring

In this use case, two SDN "match" rules are needed that mirror the traffic flows from the two source VMs (Source_VM-1 and Source_VM-2) towards the Suricata VNF. Initially, the destination VM receives both the legitimate and the malicious traffic (as in Figure 6(a)), because the security VNF is not yet aware of the malicious behaviour. When the IDS functionality of the security VNF detects a malicious signature contained in Source_VM-2's packets, it generates an SDN instruction to drop all the packets from this VM and sends it to the SDN controller (as shown in Figure 6(b)).

6 Future Work: Distributed Security Probes

In this section, a new method for Security Manager is proposed. The proposal aims to provide an extension of the current security monitoring and assurance capabilities within the SoftFIRE platform, by further extending the monitoring capability to cover not only a single tenant at a time, but also multiple tenants.

Currently, Security Manager monitors only internal traffic of a single tenant belonging to a single testbed, yet it could be interesting to monitor the network traffic of the whole SoftFIRE platform. This would allow dynamic deployment and re-configuration of security policies through security probes, each placed in a component testbed, and able to inspect and detect suspicious network traffic flows. This is illustrated in Figure 7, where multiple component testbed are linked via VPN, as in SoftFIRE. Each component testbed may be running different network services.

In the figure, a chain of security probes are shown, which monitor traffic of the whole platform, and force some traffic flows to go through a sequence of running network services. With an SDN controller, it is possible to define and modify traffic flow rules. The solution allows to inspect traffic flows over the whole platform, and enables a platform-wide reaction to malicious traffic. It involves generation of alerts, and also instructing an SDN controller to block malicious network traffic. Such alerts can be analysed in a dashboard/console environment to understand the features of the malicious activity, or could be linked to automation engines that act upon detection of such attacks. In either case, the SDN controller can then centrally define and enforce security policies to be applied on the platform.

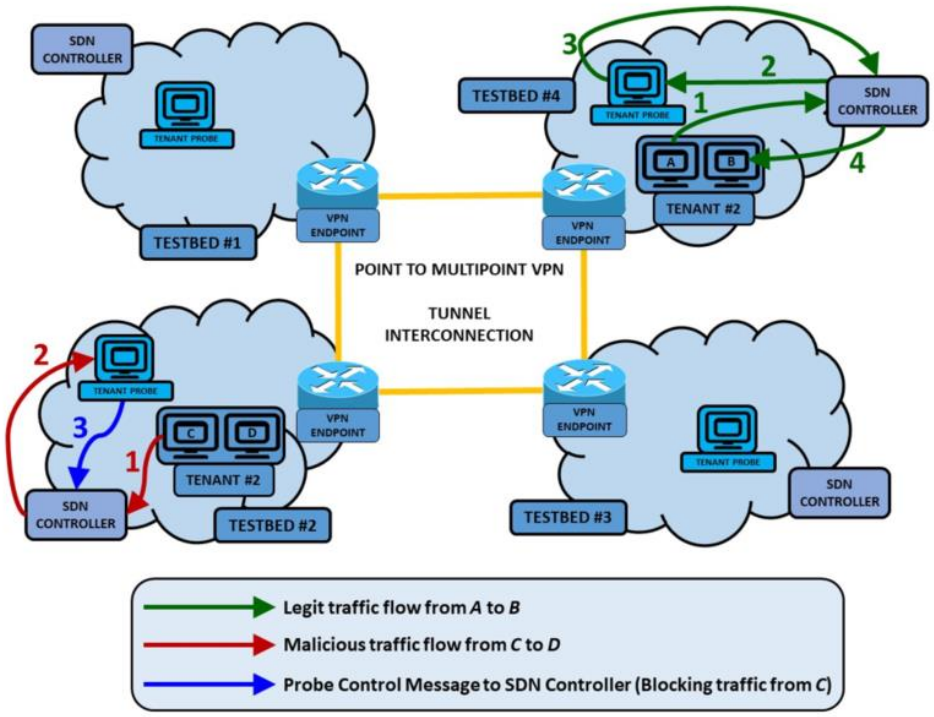

Figure 7. Schema for distributed security probes.

It must be noted that some network services may be running on local networks that are not reachable from outside (no floating IP address assigned). To be able to forward traffic to a destination that has no floating IP address, some specific SDN traffic rules must be set up.

7 Conclusions

In this white paper, SoftFIRE presents its security solution adopted to integrate a security concept, Security as a Service (SECaaS), on the NFV platform provided by the Project. The concept is presented in detail, motivating the importance of providing security-oriented resource(s) to experimenters, and describing the main benefits of the approach.

First, a number of approaches are mentioned, explaining their principal motivations, before focusing on the approach taken by the Project. SoftFIRE's Middleware component *Security Manager* is described as the solution developed in order to implement SECaaS. Then, the paper explains how security related capabilities are provided to experimenters by the Security Manager. Users can integrate security features directly in their experiment, i.e. virtualised security. This has the advantage of full exploitation of the virtual environment provided by NFV technologies. By deploying the SECaaS concept, the Project has demonstrated its capability to provide virtual security functionalities directly to end-users, i.e. experimenters. The Project believes that its SECaaS deployment is a contribution to the NFV-SDN communities, towards realising programmable and inter-operable 5G platforms.

Although the provided solution covers security VNFs that can monitor the testbed environment and take action, the Project notes that for environments in which platform access is not controlled, there is still a need for measures to protect such environments from malicious outsider activities. In other words, for platforms in which an undefined number of platform users run a number of heterogeneous services, multiple security technologies should be in place.

The paper notes that, in SoftFIRE, this was not needed, as users were provided with VPN access credentials, which enabled filtering and regulating the access to the platform. Furthermore, the platform is equipped with a monitoring system that tracks all user activities, thanks to SoftFIRE's Middleware logs that provide a way to identify unusual and potentially dangerous activity by the platform's users.

Finally, the paper mentions the concept of *security chains,* to discuss how to further improve network security in a federated NFV-SDN environment consisting of multiple testbeds. This motivates a future scenario in which more security capabilities can be added to the SoftFIRE platform. In this scenario, placing monitoring probes over the platform, which is represented by a security network service, is presented as a distributed security monitoring solution.

Bibliography

[1] "Network Function Virtualisation: State-of-the-Art and Research Challenges", Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, Raouf Boutaba, *IEEE Communications Surveys & Tutorials*, vol 18, no 1, 236-262, September 2015,

[2] "Software-Defined Networking: A Comprehensive Survey", Diego Kreutz, Fernando M. V. Ramos, Paulo Esteves Veríssimo, Christian Esteve Rothenberg, Siamak Azodolmolky, Steve Uhlig, *Proceedings of the IEEE*, vol 103, no 1, pp 14-76, January 2015,

[3] EU SoftFIRE project, https://www.softfire.eu/

[4] SoftFIRE Middleware, http://docs.softfire.eu/softfire-middleware/

[5] SoftFIRE Security Manager, http://docs.softfire.eu/security-manager/

[6] "Security position paper – network function virtualization", Cloud Security Alliance, https://cloudsecurityalliance.org/download/security-position-paper-network-function-virtualization/

[7] FIRE initiative, https://www.ict-fire.eu/

[8] Open Baton, https://openbaton.github.io/

[9] OpenStack open source cloud computing software, https://www.openstack.org/

[10] ETSI NFV Architectural Framework, http://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf

[11] "First wave of experiments on the SoftFIRE testbed", EU SoftFIRE white paper, March 2018, https://www.softfire.eu/wp-content/uploads/White-Paper-5-First-Wave-of-Experiments-on-the-SoftFIRE-Testbed.pdf

[12] FITeagle, http://fiteagle.github.io/

[13] SFA, https://wiki.confine-project.eu/sfa:start

[14] TOSCA, http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html

[15] SoftFIRE Experiment Manager, http://docs.softfire.eu/experiment-manager/

[16] SoftFIRE White Paper: "SoftFIRE Approach to Experiment Management: Why and How" June 2017 available at https://www.softfire.eu/publications/white-papers/

[17] OpenVPN, https://openvpn.net/

[18] Service function chaining architecture, IETF, https://tools.ietf.org/html/rfc7665

[19] UncomplicatedFirewall, https://wiki.ubuntu.com/UncomplicatedFirewall

[20] pfSense, https://www.pfsense.org/

[21] Suricata, https://suricata-ids.org/

[22] SoftFIRE NFV Manager, http://docs.softfire.eu/nfv-manager/

[23] OpenStack Newton, https://www.openstack.org/software/newton/

[24] Elastic Stack (ELK), https://www.elastic.co/elk-stack

[25] The FreeBSD project, https://www.freebsd.org/

[26] Openstack Port Security, https://docs.openstack.org/dragonflow/latest/specs/mac_spoofing.html

[27] OpenDaylight, https://www.opendaylight.org/

[28] OpenSDNcore, http://www.opensdncore.org/

[29] Open Virtual Switch, http://openvswitch.org/

[30] OpenFlow, https://www.opennetworking.org/sdn-resources/openflow

List of Acronyms and Abbreviations

Acronym	Meaning
5G	Fifth Generation Mobile Network
ACL	Access Control List
API	Application Programming Interface
DPI	Deep Packet Inspection
ELK	Elasticsearch-Logstash-Kibana
ETSI	European Telecommunications Standards Institute
FIRE	Future Internet Research and Experimentation
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
ICT	Information and Communications Technology
ID	Identifier
IDS	Intrusion Detection System
IP	Internet Protocol
IPS	Intrusion Prevention System
IPv4	Internet Protocol version 4
JSON	JavaScript Object Notation
LAN	Local Area Network
MAC	Medium Access Control
NFV	Network Function Virtualisation
NS	Network Service
ODL	OpenDaylight
OVS	Open Virtual Switch
REST	REpresentational State Transfer
SDN	Software Defined Network
SECaaS	Security-as-a-Service
SFA	Slice-based Federated Architecture
TCP	Transport Control Protocol
TOSCA	Topology and Orchestration Specification for Cloud Applications
UFW	UncomplicatedFirewall
VDU	Virtual Deployment
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNF	Virtual Network Function
VPN	Virtual Private Network

Disclaimer

SoftFIRE has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement no. 687860.