Embedding Security and Privacy into the Development and Operation of Cloud Applications and Services

Tran Quang Thanh, Stefan Covaci, Thomas Magedanz
Technical University Berlin and Fraunhofer FOKUS
Berlin, Germany
q.tran@tu-berlin.de

Panagiotis Gouvas, Anastasios Zafeiropoulos
Ubitech Ltd.
Athens, Greece
pgouvas@ubitech.eu

***Abstract*— This paper introduces an approach allowing cloud application developers, service providers to consider security and privacy requirements across the application lifecycle. Specifically, a DevOps framework has been described that took into account several emerging technologies such as Network Functions Virtualization (NFV) and Microservice Pattern Design. As an illustration, a proof-of-concept application in the healthcare domain is presented to support such direction.**

Keywords— Cloud Computing; Microservice; NFV; Security;

I. INTRODUCTION

The recent advances in "Softwarization" technologies (e.g. virtualization, programming API, decoupling software and hardware) are considered as a game-changer for the whole economy and society. Various industries including Telecom are underway to move applications and services to the cloud as the given benefits are unarguable. However, the transition also poses several challenges. Cloud application nowadays is not developed for a single hardware platform or operating system anymore but runs on distributed, heterogeneous and systems. The use of cloud infrastructure for software application deployment and data storage makes security and privacy concerns unquestionable challenges. Security concern is also one of biggest challenges when adopting "Softwarization". As a result, newly design software solutions are required evolvable, adaptable and should guarantee nonfunctional properties including security and privacy [1]. In addition to that, the integration of cloud application developments with DevOps is getting a lot of interests. DevOps allows development, quality assurance and operation teams working together to get things done faster in an automated way [2].

In this paper, a DevOps software framework is proposed that allows cloud application developers and service providers to consider security and privacy across lifecycle: starting from the context model to the embedding of metadata in the source code, to the parsing and interpretation for automatic deployment, to active response during the operation phase. Specifically, several evolving technologies such as Microservice pattern design [3], Network Functions Virtualization (NFV) [4] and Policy-driven Orchestration are taken into account. Moreover, several interpretable software annotations are proposed to support developers to decorate their applications with specific information (e.g. configuration, performance metrics, lifecycle) to address different security and privacy requirements.

The remainder of the paper is organized as follows: Section 2 introduces the adopted technologies and concepts: NFV Management and Orchestration framework (NFV MANO) and microservice pattern design. The proposed framework is described in Section 3. The use of the framework to support security requirements is discussed in Section 4 through an evaluation use case in the health care domain. Section 5 and 6 present related works and conclude the paper.

II. BACKGROUND

A. NFV Management and Orchestration

NFV is set of de-facto standards and technologies that enable network managers running their infrastructure in an efficient manner by moving network functions out of dedicated hardware devices into software. To leverage NFV technology, the European Telecommunications Standards Institute (ETSI) has defined a standard reference framework [5] with four main functional blocks and several interfaces which connect the management and orchestration domain (MANO) and non-management domain (see Figure 1). These blocks include the MANO block (on the right), the legacy Operation Support System/ Business Support System (OSS/BSS), the Virtualization Network Functions (VNF) and the underline NFV infrastructure (virtual and physical compute, network and storage resources). The MANO block consists of three elements: the Virtualized Infrastructure Manager (VIM), the VNF Manager (VNFM) and the NFV Orchestrator (NFVO). The primary function of the VIM is to manage and allocate virtual resources (e.g. adding, removing). OpenStack is the popular VIM implementation. The VNFM controls the lifecycle of deployed applications/network functions (e.g. installation, configuration, update, scale and termination). The VNFM can be part of the NFVO (generic VNFM) or provided by NFV vendors (specific VNFM). The NFVO takes care of the end-to-end service including VNF provisioning, configuration, network setup and scaling.

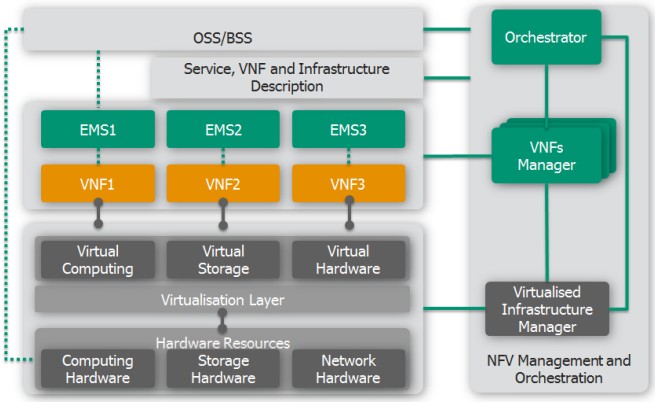

Fig. 1. ETSI NFV MANO framework

B. *Microservice Pattern Design*

The idea to split an application into a set of smaller and interconnected services (microservice) is currently getting many interests from application developers and service providers (e.g. Amazon, Netflix, eBay). Such modularity concept is revolutionizing the way software design, build and ship by its given advantages. For instance, individual services are much faster to implement, easier to understand and maintain, and each service can be developed, deployed and scaled independently. In addition to that, the development cost will be significantly reduced as several entities can be reused from existing software enabled components. Several tools and software frameworks are available to support developers implementing and deploying microservice (e.g. Spring Boot [6], Docker [7] and OSv [8]). Many software components have been developed and available as open source solutions including those recently implemented in the Future Public-Private Partnership Programme FIWARE project by European Commission [9]. In addition to that, several microservice reference architectures are proposed (e.g. by NGINX, FIWARE [10] [11]) to simplify the development and operation of cloud applications/services. The FIWARE reference architecture for the IoT application development is shown in Figure 2 that utilizes several FIWARE generic enablers/microservices (e.g. Context Broker GE, Identity Management GE).

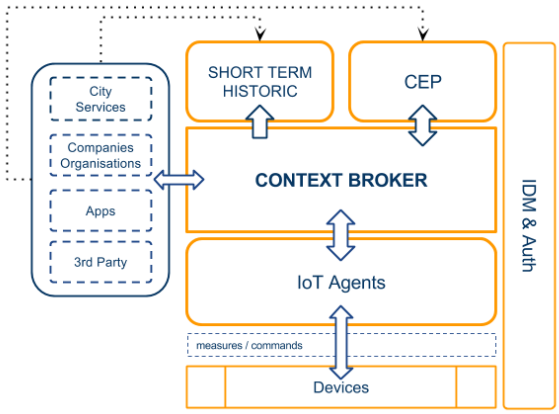

Fig. 2. FIWARE IoT Application Reference Architecture

III. ARCADIA FRAMEWORK

In this section, a novel software framework is described that enables software developer and service provider to develop and operate cloud applications in an efficient and flexible manner. The proposed framework is designed and ongoing developed in the context of European ARCADIA project in the Horizon 2020 Framework Programme. The high-level framework architecture is provided in Figure 3 with several interoperable software components/toolkits which are further discussed as follows:

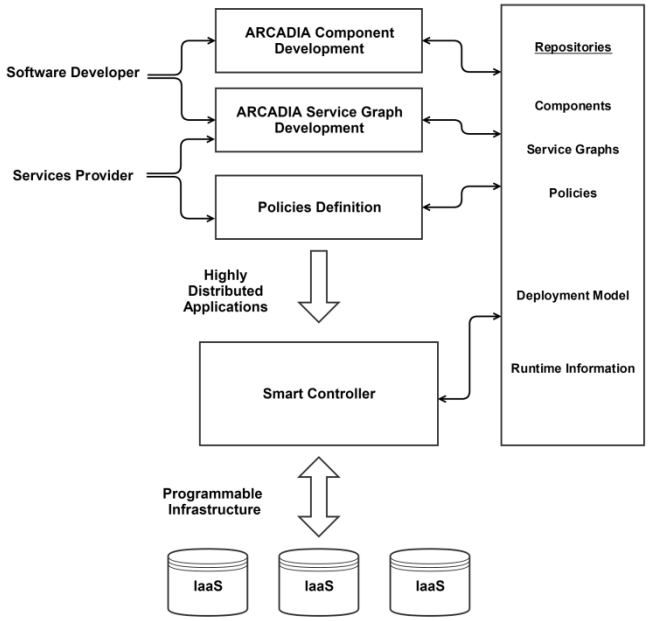

Fig. 3. ARCADIA framework architecture

In the upper level is a set of available components for designing, developing and deploying ARCADIA applications. Each ARCADIA application is represented in the form of a service graph that is based on a set of microservice along with their interconnection. A fundamental component used towards this direction is the Component/Service Graph Development/Editing/Deployment Toolkit. The toolkit supports various phases of the application development process (application development, deployment preparation, policies specification). The toolkit is also interconnected with the ARCADIA metamodel, the ARCADIA Annotation framework, the Component/Graphs Repository and the Policies Repository. A component metamodel has been specified that should be respected by any ARCADIA application. A microservice that is developed according to the ARCADIA framework must:

- expose its configuration parameters along with their metadata
- expose chainable interfaces which will be used by other microservices in order to create a service graph;
- expose required interfaces which will also be utilized during the creation of a service graph;
- expose quantitative metrics regarding the QoS of the microservice;

- encapsulate a lifecycle management programmability layer which will be used during the placement of one service graph in the infrastructural resources;
- be stateless in order to be horizontally scalable by design;
- be reactive to runtime modification of offered resources in order to be vertically scalable by design;
- be agnostic to physical storage, network, and general purpose resources.

As already mentioned, to providing access to the existing set of components and service graphs, a Components/Graphs Repository is made available to software developers and service providers. The development philosophy within ARCADIA supports the re-use of existing components and graphs for the design of applications and services. Furthermore, upon the validation of the proper development of an ARCADIA component/graph, this component/graph is made available for further use through the Components/Graphs Repository.

The Policies Repository is used for the collection of a set of policies on behalf of the services provider. Such policies can be high-level policies or policies directly associated with the potential usage of an ARCADIA Service Graph. A Policies Editor is used to this end for facilitating service provider to define a set of rules and actions taking into account the monitoring hooks/metrics available per Service Graph.

Within the Component/Service Graph Development/Editing/Deployment Toolkit, the software developer can develop native ARCADIA components and make them available –upon validation- to the Components/Graphs Repository, as well as create deployable service graphs based on native and/or reusable components or service graphs. The software developer is also able to use the Annotation framework for specifying annotations while the implementation of component/service graph interfaces has to be based on the existing context model. The software developer is also able to adapt legacy applications transforming them to ARCADIA components while the software developer and the services provider can make deployable service graphs out of reusable components or graphs as well as make multi-tenant deployable service graphs.

To support nonfunctional properties such as security and privacy, the proposed framework extends the current policy-driven MANO approach by integrating source code annotation technique. As embedding inside the source code, such interpretable software annotations enables smart controller having a deeper understanding about application semantic (e.g. specified metrics, configuration parameters) to adapt to new requirements or to support future operation decisions. As a result, an Annotation framework is used during applications development for the inclusion of a set of annotations at software level on behalf of the software developers. Such annotations are based on concepts represented in the ARCADIA Context Model and can be interpreted during deployment targeting at providing hints towards the optimal deployment of the application. For existing applications, the editing and annotation phases are skipped, and the development process only includes the generation of the metamodel.

The annotations are functionally grouped into five categories according to the nature of the business logic that is bound to the usage of the annotation. These categories include component management, configuration management, management of performance metrics, management of lifecycle and management of dependencies. Figure 4 gives an overview of current supported Java-based annotations (e.g. @ArcadiaComponent, @ArcadiaConfigurationParameter, @ArcadiaMetric and @LifecycleStart)

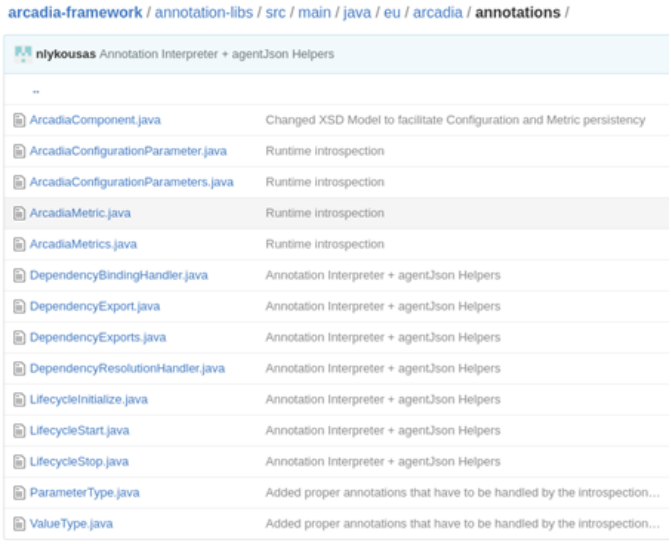

Fig. 4. ARCADIA annotations

In the lower level of the framework is the Smart Controller with its interfaces towards managed cloud resources. The Smart Controller is the application's on-boarding utility that extends and operates on top of several ETSI MANO functional block (e.g. NFVO, VNF Manager) which undertakes the tasks of i) translating deployment instances and annotations to optimal infrastructural configuration, ii) initializing the optimal configuration to the registered programmable resources, iii) supporting monitoring and analysis processes and iv) reacting pro-actively and re-actively to the configuration plan based on the infrastructural state, the application's state and the applied policies. The application's software components –as denoted in the corresponding service graph- are instantiated on demand by the Smart Controller. The defined monitoring hooks initiate a set of monitoring functionalities for specific performance metrics. The status of these metrics trigger re-configurations in the deployed application based on optimization objectives (as denoted in the selected policies) along with a set of constraints that are considered during the application deployment and runtime. Resources reservation and release is realized on demand over the programmable infrastructure.

IV. PROOF-OF-CONCEPT APPLICATION

In this section, an eHealth application is selected to illustrate the effectiveness of our proposed framework towards secure developing and operating a cloud application.

Specifically, a Remote Patient Monitoring (RHM) scenario is implemented in which patients' vital health parameters (e.g. heart rate) are securely collected, stored on the cloud and given access on real-time through a set of APIs. Figure 4 presents a high-level architecture of the use case containing several software components and interconnecting interfaces.

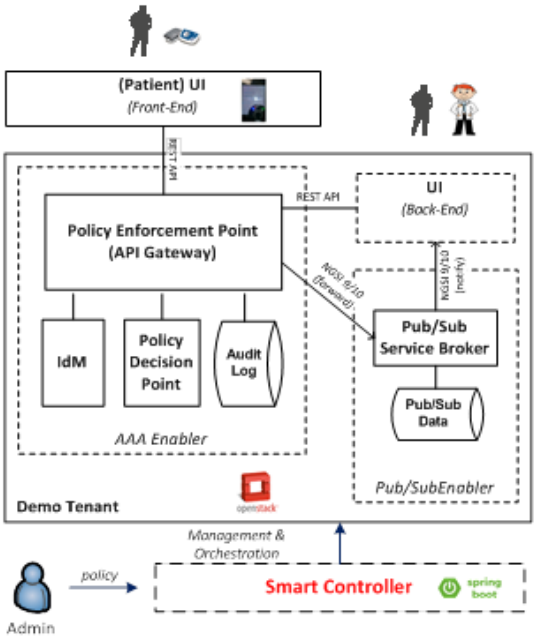

Fig. 5. RPM application architecture

The Front-end is an Android application connects to the Zephyr "BioHarness-3" device through Bluetooth. User heart rates are collected in real-time and securely sent to the Back-end service (users are required to authenticate themselves and to use VPN service to secure data collection and transmission). The development of the Back-end follows the aforementioned FIWARE reference architecture for IoT application. Such "microservice-based" approach enables us to reuse available open source solutions when developing our application. In the current version, three microservices (enablers) are developed including the AAA Enabler (extends the FI-STAR Integrated Access Control Enabler [12]), the Pub/Sub Enabler (extends the FIWARE Orion Context Broker [9]) and the UI Enabler (User Interface web application). The ARCADIA framework allows decomposing current enablers into smaller potential microservices (Figure 5) but it will be our future works. The Back-end application is deployed on the TU-Berlin OpenStack Cloud infrastructure. The current implementation of the Smart Controller is also based on our open source solution, OpenBaton [13]. OpenBaton is developed using Spring Boot Java framework to provide core functional blocks in the ETSI MANO reference architecture: the NFV Orchestrator and the generic VNF Management.

The proposed framework allows us to consider security and privacy requirements across the application lifecycle. Specifically, two significant security benefits are taken into account: Virtualized Security Function/Service and Centralized Security Management and Orchestration. The former enables us to select and integrate multivendor security solutions when designing and operating our application. The latter allows our application to be configured and protected effectively according to consistent policies as well as to support a set of automated mechanisms.

Virtualized Security Function/Service:

In our current application, two virtualized security functions are taken into account to provide different layers of protections. To secure access at the network layer, the OpenStack Firewall as a Service (FWaaS) extension is used. The FWaaS enables us to apply different filters (access control policy) on network traffic entering and leaving the demo tenant network. As part of the application, the "AAA Enabler" provides protection towards application APIs. This microservice has been implemented based on several open source solutions (e.g. Spring Cloud Netflix [14] for the API Gateway module, WSO2 Identity Server [15] for the Identity and Access Management). To provide token-based authorization for API accesses, two security standards are taken into account: OAuth version 2 [16] and Attribute-based Access Control (ABAC). The former is the evolving standard solution to secure API access. OAuth2 allows users (resource owner) to grant third-party applications (client) accessing user data (resource server) without sharing credential (e.g. password). The latter is developed by OASIS to standardise the authorization decisions in enterprise applications. It defines XACML [17] (a XML-based policy language), a request / response scheme and access control architecture. The microservice design architecture is flexible enough to integrate more security and privacy microservices (e.g. VPN, privacy enhancing, threat detection and mitigation) to support different application requirements.

Centralized Security Management and Orchestration:

A Java-based Policy-driven Orchestration and Management Software component (the Smart Controller) has been developed by extending our OpenBaton toolkit. The controller supports two types of policies: Decision-Making and Access Control. The former will be enforced by the Smart Controller during either application placement (e.g. IaaS selected policies based on security and privacy requirements) or run-time (e.g. scale policies based on monitoring data and predicted workload). Several monitoring metrics are taken into consideration such as "Status" (DOWN|UP|UNKNOWN), "Request Arrival Rate" (/s), "Average Response Time" (ms), CPU and Memory Usage (%), Error Request Rate (/s). The latter will be automatically deployed in required microservices. In our application, such access control policies are deployed on the "AAA Enabler" and OpenStack FWaaS [18]. The Drools [19], a business process rules language, is selected to specify our decision-making policies and XACML, an industry standard for authorization, is used for access control policies. Both Drools and XACML are supported in many programming environments (e.g. Java, .NET, Python, PHP). Adopting centralised policy allows us to separate

business rules from application logic and to change policies (rules) to adapt to new requirements without deploying a new code or restarting the application. Figure 6 gives an example of a placement policy using Drools to support the requirement to deploy our demo eHealth application on a private cloud (more secure than the public model) in Germany (to comply with data protection laws).

```
rule "IaaS Policy 1"

        when
                item : IaaS(model == IaaS.Model.PRIVATE,
                            location == IaaS.Country.GERMANY)
        then
                insert(item);

end
```

Fig. 6. Example of an ARCADIA policy (using Drools)

To support policy decision and deployment in an automatic way, some security-related annotations are specified belonging either to the configuration (C) or metric (M) categories. Table 1 gives a brief overview of all security-related annotations in our application. The microservice which uses metric-type annotations must implement a method to return user-defined measurements. The configuration annotations are specified to support microservice discovery that is required in any cloud-based microservice application.

TABLE I. SECURITY-SUPPORTED ANNOTATIONS

Name	Type	Description
health	M	Status of running microservice (UP|DOWN|UNKNOWN)
cpuUsage	M	CPU Usage (%)
memoryUsage	M	Memory Usage (%)
diskSpace	M	Storage Usage (%)
requestArrivalRate	M	Number of API Requests (/s)
errorRequestRate	M	Number of Error Requests (/minute)
unauthoriredRequestRate	M	Number of Unauthorized Requests (/minute)
isPEP	C	Whether a microservice plays a role as Policy Enforcement Point (0|1)
isPDP	C	Whether a microservice plays a role as Policy Decision Point (0|1)
isPAP	C	Whether a microservice plays a role as Policy Administration Point (0|1)
isOAuthEndPoint	C	Whether a microservice plays a role as OAuth API End Point (0|1)
isSCIMEndPoint	C	Whether a microservice plays a role as SCIM (a standard for identity provisioning) API End Point (0|1)
requiredPEPProxy	C	Whether a microservice needs an external PEP proxy for authorization (0|1).
serviceAPI	C	Information about exported REST APIs

To include such annotations into application microservices, an IDE platform has to be used which relies on a state-of-the-art web-based IDE called "Eclipse Che" [20], allowing developers to develop microservices using the Bring-Your-Own-Device (BYOD) principle without the need to setup anything locally. Eclipse Che is a general-purpose development environment that can be used to develop anything. A specific plug-in has been developed that interacts with the Smart Controller to assist developers during component development. Through the plug-in, developers can view all supported annotations, including some Javadoc-based documentation and trigger component validation. Specifically, when a developer starts writing an ARCADIA annotation, the plug-in offers auto-complete suggestions where developers can just choose which annotation they want to use. Also, they can read the specifications of each annotation, like the required fields, naming conventions and also examples.

Figure 7 presents several user interfaces (UI) of current RPM application. They are an Android UI at the Frontend to collect user heart rate, a Backend UI to visualize such information together with profile data and a Smart Control UI that display the current deployment of three microservices (AAA, UI and PubSub) on top of TU-Berlin OpenStack cloud infrastructure.

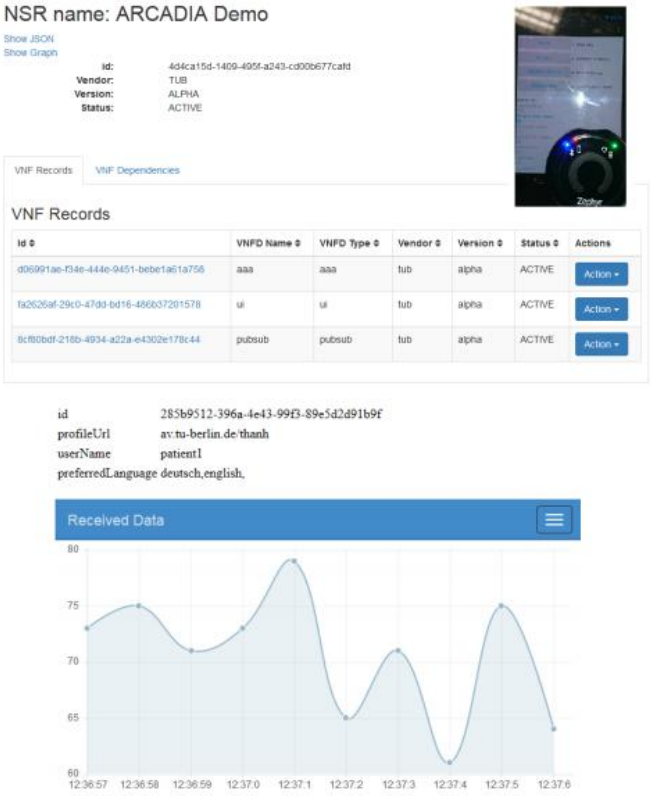

Fig. 7. RPM Application User Interfaces

V. RELATED WORKS

The development and deployment of a variety network security functions on virtual infrastructure are getting interests recently as the benefits of virtualization technology are impossible to ignore. Many common security solutions are going to be virtualized (e.g. OpenWRT, Squid Proxy, Bro IDS, OpenVPN, CloudRouter). Several tools and software frameworks are available to support developers implementing and deploying microservice (e.g. Spring Cloud Netflix [14],

Docker [7] and OSv [8]). Popular cloud platform such as OpenStack, AWS have integrated different security services to increase the privacy and control of the accesses (e.g. FWaaS, Secure Storage, Multi-factor authentication, DDoS mitigation) [18][21]. Adopting NFV MANO and SDN to build security solutions is constantly growing by the given advantages (e.g. fast, scalable and capable of supporting both application context and isolation) [22][23] [24][25][26]. Many European research projects were and are ongoing to follow such directions to provide different types of security services, toolkits and evaluate them in various use case domains (e.g. FI-PPP FIWARE, MCN, T-NOVA, 5G-PPP ENSURE and SONATA) [9] [27] [28] [29] [30].

Our work shares common ideas with existing works. Moreover, different technologies are taken into account to help developers, service providers to deliver application/service in a fast, scalable and secure manner. To be best of our knowledge, this is the first time the use of source code annotations to support centralized security policy management and orchestration is applied to the NFV infrastructure.

VI. Conclusion

In this paper, a novel DevOps software framework is proposed that allow cloud application developers, service providers to consider security and privacy across application's lifecycle by leveraging Network Functions Virtualization (NFV) and Microservice Pattern Design technologies. In addition to that, the use of policies and source code annotations are taken into account. As an illustration, the Remote Patient Monitoring application has been developed to support such direction. Our future work is to improve the current software toolkits and proof-of-concept use case towards multi-tenant, multi-domain and SDN-based service function chaining support.

Acknowledgment

The research leading to these results has received funding from the European Community's Horizon 2020 Framework Programme under grant agreement no. 645372

References

[1] "Toward a Strategic Agenda for Software Technologies in Europe," Information Society Technologies Advisory Group (ISTAG), July 2012, Available Online: http://cordis.europa.eu/fp7/ict/docs/istag-soft-techwgreport2012.pdf

[2] From Dev to Ops: An introduction, https://www.appdynamics.de/media/uploaded-files/White_Paper_-_An_Intro_to_DevOps.pdf

[3] Microservices Architecture, http://microservices.io/patterns/microservices.html

[4] Network Functions Virtualization, https://portal.etsi.org/nfv/nfv_white_paper.pdf

[5] Network Function Virtualization Management and Orchestration. http://www.etsi.org/deliver/etsi_gs/NFVMAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf

[6] Spring Boot framework, http://projects.spring.io/spring-boot/

[7] World's leading software containerization platform, https://www.docker.com

[8] Operating system designed for the cloud, http://osv.io

[9] FIWARE: Open APIs for Open Minds, http://catalogue.fiware.org

[10] Introducing the Microservices Reference Architecture from NGINX, https://www.nginx.com/blog/introducing-the-nginx-microservices-reference-architecture/

[11] FIWARE IoT Stack, http://fiware-iot-stack.readthedocs.org/en/latest/index.html

[12] Tran Quang Thanh, Stefan Covaci, Benjamin Ertl, Paolo Zampognano, "An Integrated Access Control Service Enabler for Cloud Applications", in Book Future Network Systems and Security, Springer International Publishing, Switzerland, p. 101-112, June 2015

[13] OpenBaton. http://openbaton.github.io

[14] Spring Cloud Netflix, https://cloud.spring.io/spring-cloud-netflix/

[15] WSO2 Identity Server, http://wso2.com/products/identity-server/

[16] OAuth 2.0 Authorization Framework, http://tools.ietf.org/html/rfc6749

[17] OASIS eXtensible Access Control Markup Language, https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

[18] OpenStack Firewall as a Service, https://wiki.openstack.org/wiki/Neutron/FWaaS

[19] Business Rules Management System, http://www.drools.org

[20] Eclipse Next-Generation IDE, http://www.eclipse.org/che/

[21] AWS Cloud Security, https://aws.amazon.com/security

[22] K. Giotis, Y. Kryftis and V. Maglaris, Policy-based orchestration of NFV services in Software-Defined Networks, Network Softwarization (NetSoft), 2015 1st IEEE Conference on, London, 2015, pp. 1-5

[23] Fung, C.J.; McCormick, B., "VGuard: A distributed denial of service attack mitigation method using network function virtualization," in Network and Service Management (CNSM), 2015 11th International Conference on, vol., no., pp.64-70, 9-13 Nov. 2015

[24] Bridge Virtualization and Security with OpenStack and Contrail, https://www.openstack.org/summit/openstack-summit-atlanta-2014/session-videos/presentation/bridge-virtualization-and-security-with-openstack-and-contrail

[25] Seungwon Shin and Guofei Gu, CloudWatcher: Network security monitoring using OpenFlow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?), 20th IEEE International Conference on Network Protocols (ICNP), Austin, TX, 2012, pp. 1-6.

[26] M. A. Lopez and O. C. M. B. Duarte, Providing elasticity to intrusion detection systems in virtualized Software Defined Networks, IEEE International Conference on Communications (ICC), London, 2015, pp. 7120-7125.

[27] Mobile Cloud Networking, http://www.mobile-cloud-networking.eu/site/

[28] T-NOVA European FP7 Project, http://www.t-nova.eu/objectives/

[29] 5G Enablers for network and system security, http://www.5gensure.eu

[30] Agile Service Development and Orchestration in 5G Virtualized Networks, http://www.sonata-nfv.eu