Khettab, Yacine; Bagaa, Miloud; Cadette Dutra, Diego; Taleb, Tarik; Toumi, Nassima

Virtual Security as a Service for 5G Verticals

Published in:
IEEE Wireless Communications and Networking Conference, WCNC 2018

DOI:
10.1109/WCNC.2018.8377298

Published: 01/01/2018

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Khettab, Y., Bagaa, M., Cadette Dutra, D., Taleb, T., & Toumi, N. (2018). Virtual Security as a Service for 5G Verticals. In *IEEE Wireless Communications and Networking Conference, WCNC 2018* (IEEE Wireless Communications and Networking Conference). IEEE. https://doi.org/10.1109/WCNC.2018.8377298

Virtual Security as a Service for 5G Verticals

Yacine Khettab, Miloud Bagaa, Diego Leonel Cadette Dutra, Tarik Taleb and Nassima Toumi
Dep. of Communications and Networking, School of Electrical Engineering, Aalto University, Espoo, Finland
Emails:{firstname.lastname}@aalto.fi

***Abstract*—The future $5G$ systems ought to meet diverse requirements of new industry verticals, such as Massive Internet of Things (IoT), broadband access in dense networks and ultra-reliable communications. Network slicing is an important concept that is expected to support these $5G$ verticals and cope with the conflicting requirements of their respective services. Network slicing allows the deployment of multiple virtual networks, or slices, over the same physical infrastructure as well as supporting on-demand resource allocation to those slices. In this paper, we propose an architecture that will explore how both Network Function Virtualization (NFV) and Software Defined Networking (SDN) may be leveraged to secure a network slice on-demand, addressing the new security concerns imposed to the network management by the flexibility and elasticity support. Our proposed framework aims to ensure an optimal resource allocation that manages the slice security strategy in an efficient way. Moreover, experimental performance evaluations are presented to evaluate the security overhead in virtualized environments.**

I. Introduction

The next generation of wireless access systems ($5G$) aims to improve the Quality of Experience (QoE) for critical services, as well as to provide high availability, low latency, elasticity, and increased security [1]. SDN, NFV and Network Slicing are new concepts and technologies that have emerged as $5G$ enablers. These technologies can be leveraged to provide Security as a Service (SECaaS) by deploying Security Virtualized Network Functions (VNFs) within different slices and ensuring optimal resource provisioning to reduce Operational Expenditures (OPEX) while ensuring the provisioning of the Service Level Agreement (SLA). Furthermore, proper resource allocation is crucial as a malfunctiong security VNF can compromise the network; therefore, a predictive auto-scaling function, implementing application-specific policies, needs to be deployed along with the monitoring and flow control mechanisms.

Recent works have investigated solutions to enable auto-scaling mechanisms [2], [3], factors that impact the scaling delay [4]–[6], and mechanisms relying on SDN to enable security functions [7], [8]. Other solutions have focused on the performance evaluation of security VNFs [9]–[13]. To the best knowledge of the authors, no prior work has been taking into account ways to enforce security within $5G$ slices. This paper proposes and evaluates an architecture that leverages both SDN/NFV capabilities to enable SECaaS in multi-cloud environments.

The rest of this paper is organized as follows. Section II discusses previous works on the subject of this paper. Section III presents an overview of the proposed architecture for enabling SECaaS. Section IV provides the methodology for our experimental evaluation, and discusses the results. Finally, Section V concludes the paper.

II. Related Work and Background

A. Performance analysis of security VNFs

Brumen and Legvart [9], White et al. [10], and Cao et al. [12] have evaluated the performance and security level of open-source Intrusion Detection/Prevention System (IDS/IPS) softwares – mostly Snort and Suricata – under different parameters including: operating systems, hardware configuration, workload, types of attacks, and signature database. In [9], the authors perform a comparative analysis of Snort and Suricata on Windows and Linux, varying the attack types and using the number of dropped packets as the key performance metric. Their performance results have shown that the Windows deployment consumed fewer resources but had a higher drop rate than the Linux deployment. Moreover, they showed that Snort's resource consumption was less significant than Suricata's, albeit with a higher drop rate. They concluded that Windows-based solutions were not suitable for both open sources and that Suricata performed better than Snort. White et al. [10] compared the performance of Snort and Suricata under default and optimized configurations, e.g., multi-instance Snort. Their results lead to performance improvements in Suricata of up to 20x for computer nodes with more than 4 cores, while also showing that a single instance of Suricata outperforms Snort under all the evaluated configurations.

Cao et al. [12] presented a framework for performance characterization of different VNFs, using Clearwater, Snort, and Suricata as case studies. Their experiments showed that in contrast to Snort, Suricata's performance scales with the number of cores: Snort is a single-threaded architecture while Suricata is a multi-threads architecture. Bujlow et al. [11] conducted an extensive study comparing different proprietary and Open Source DPIs. They found that among the open source tools, *nDPI* [14] and *Libprotoident* exhibited the best performance and that is despite *Libprotoident*'s inability to identify *Google*, *Twitter*, and *Facebook*'s flows. They also did not evalute the DPI's resource utilization. Jati et al. [13] proposed a system for the detection of Distributed Denial of Service (DDoS) attacks using Ntopng. Their evaluation showed that the detection accuracy was not disturbed by the traffic load, while the dropped packets rate was less than 1%; performance-wise, the resource consumption was relatively stable regardless whether an attack is being launched or not.

978-1-5386-1734-2/18/$31.00 ©2018 IEEE

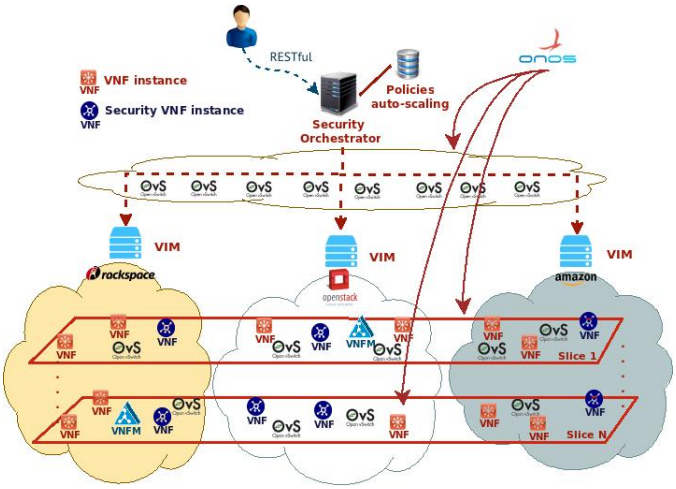


Fig. 1. Envisioned architecture for enabling Security as a Service.

B. Auto-scaling

The existing scaling solutions of could systems, e.g., Amazon Auto Scaling [15] in Amazon Elastic Compute Cloud, lack automation as they require users to set a scaling threshold for each scaling operation, adopting a reactive approach, and ultimately resulting in a non-optimal resource utilization by VNFs. Thus, cloud users, e.g., Netflix [16], have to develop their own predictive auto-scaling solutions.

In the recent literature, different forecasting techniques have been proposed to enable predictive approaches [2], [3], [17]. In [2], Shariffdeen et al. evaluated different forecasting models and proposed a new model combining neural networks, an exponential model, and a naive prediction scheme. Their experimental results showed that the proposed model performed better than the individual ones. Ren et al. [3] proposed a dynamic auto-scaling algorithm for $5G$ mobile networks taking into account the Virtual Machine (VM) setup time as well as the capacity of the legacy equipment.

Shen et al. [17] proposed an elastic resource scaling system for multi-tenant clouds by combining resource demand prediction, conflict prediction and migration to support concurrent scaling. In the same vein, Mao and Humphrey [5] as well as Ueda and Nakatani [6] evaluated different cloud infrastructures in terms of VM startup-time and scale-out time, as well as the factors that impact those metrics [4].

C. Security in SDN

Yoon et al. [7] explored the feasibility and efficiency of deploying, at the controller level, security functions as stateful firewalls, IDS/IPS, and anomaly detection applications. In their proposed framework, the deployed application examines each packet separately, then the controller receives instructions on how to issue the appropriate rule for the flow (e.g., permit, block, or relay to an IDS for in-depth inspection). Similarly, Shin et al. [8] demonstrated how each SDN feature can benefit the network security with example scenarios of NIDS, Intelligent Honeypot and monitoring applications deployed on the controller [18].

Both research work rely on the first packet of every flow unknown to the switches and therefore having its header relayed to the controller to be examined by the deployed applications. However, in many cases, the first packet's header has insufficient information to determine if a flow is suspicious, allowing malicious flows to pass undetected provided that their first packet seems legitimate. Furthermore, since only packet's headers are sent to the controller, payload-based attacks go through undetected.

III. Security as a Service Architecture

The $5G$ mobile system is expected to support the new requirements of new vertical industry services, such as massive Internet of things (mIoT), broadband access in dense areas, and ultra-reliable communications. The envisioned $5G$ systems needs then to re-architect the current uniform mobile architecture to allow multiple, logical, self-contained networks on a common physical infrastructure platform enabling a flexible stakeholder eco-system that allows technical and business innovations, integrating network and cloud resources into a programmable, software-oriented network environment [19].

Network slicing is one of the enabling technologies that will support $5G$ services, as it allows each vertical service to have a dedicated network slice that offers the required resources for that vertical service. Network slicing is mainly based on SDN, NFV and cloud computing. ETSI NFV [20] has defined a reference architecture for enabling NFV orchestration and VNF management in an efficient manner. The NFV [21] technology will enable the elasticity and flexibility for creating different slices across multiple domains. Meanwhile, the SDN technology will enable the programmability of different Open Virtual Switches (OVS) and SDN-enabled switches for ensuring the connectivity between different VNFs in the same network slice.

The creation of different VNFs in different slices would create more vulnerability in different VNFs comparing to the static network [22]. Therefore, ensuring the security within the same slice can be a challenging problem. Fig. 1 shows an overview of the proposed architecture that would enable SECaaS in an inter-domain platform. This architecture will deploy and manage different security VNFs including IDS/IPS and Deep Packet Inspection (DPI). The proposed architecture framework aims to ensure elasticity by dynamically deploying security VNF instances, monitoring their performance, and performing predictive auto-scaling based on pre-defined policies and metrics.

Fig. 1 shows our NFV architecture consisting of four main parts: i) Cloud networks, each of which is managed by a Virtual Infrastructure Manager (VIM; e.g., OpenStack), ii) VNF Managers (VNFMs) that are responsible for managing and monitoring a set of VNFs in the same slice during their run-time, iii) a NFV Orchestrator (NFVO) that is responsible for creating, managing and orchestrating all VNF instances in different cloud networks and iv) a distributed SDN controller

that controls and monitors the flows between the VNFs. The VIM functionality of each cloud provider must run different virtualization technologies (e.g., KVM, XEN or Containers) that allow the creation of multiple virtual resources on shared hardware resource (e.g., Compute, Storage, and Network) [23]. The VIM allows the instantiation of different VNF instances with different virtual resources using pre-stored VNF images. Different resources in a cloud network are defined through a set of flavors, whereby each flavor represents the amount of virtual resources (i.e., number of Virtual cores - CPU, memory, and storage) that would be dedicated to a specific VNF instance [24].

A. Enabling Security as a Service with SDN

In this section, we concentrate on the networking aspects (i.e., SDN) of network slicing which can be provisioned in other complementary and orthogonal ways. Indeed, in a full solution, one would have to take into account NFV resource management, workload mobility, VNF placement and VNF security [25]–[27].

We leverage ONOS [28] as SDN controller to enable SECaaS in our architecture. Indeed, we use the intent concept of ONOS and create per flow point-to-point intents to route traffic to specific firewall or IPS instances. We also write intents that aggregate the output traffic of a firewall instance and forward it to the correct node/VM in our secured network.

For passive security, e.g., IDS, the ONOS controller can deploy multi-points to single-point intents in order to forward traffic to its original destination and mirror it to a specific IDS instance. Besides, when an IDS detects a malicious flow, it generates an alert and sends it to the security orchestrator, which will then take into account the number of received alerts, as well as the level of severity in order to instruct the controller to either stop the malicious flow temporarily or permanently, or constrain its bandwidth to avoid overloading the network while maintaining a certain level of service.

Algorithm 1 Attack-Response Algorithm

Require:
L: Level of the received alert.
T: Type of the received alert.
F: Flow that triggered the alert.

1: $NumAlerts$[L][T] $\leftarrow$ $NumAlerts$[L][T] + 1;
2: **if** $NumAlerts$[L][T]$>=$ $trigThreshold$[L][T] **then**
3: $triggResponse$[F] $\leftarrow$ $triggResponse$[F] + 1 ;
4: **end if**

Furthermore, SDN's capabilities are mandatory to enable the auto-scaling support, i.e., need to guarantee that a complete traffic analysis may be supported even during an attack that would overload the current slice security configuration. Scaling-out an IDS instance requires splitting the incoming traffic between the new instance and the existing ones. Additionally, network flows need to be managed in a way that ensures security isolation between slices.

B. Auto-Scaling Mechanism

As shown in Fig. 1, our envisioned security orchestrator offers a RESTful API that allows the admin user to specify different management rules and policies for the instantiation and auto-scaling of the VNF instances. Based on these policies, the security orchestrator enforces the rules for a specific slice by communicating them to VNFM of the slice, allowing it to enforce the security rules by communicating to different security VNFs. The VNFM dynamically launches security VNF instances in different slices with pre-installed software in the cloud and monitors their performance metrics in order to trigger scaling actions according to the predefined policies. The scaling policies are set according to the VNF's performance requirements and behavior depending on traffic's load. Moreover, the security orchestrator communicates with the SDN controller, e.g. ONOS, to provide connectivity for the different security VNFs and VNF instances together in the same vertical.

In the proposed architecture, an auto-scaling algorithm is executed at the VNF Manager of each slice in order to scale-in or scale-out each security VNF instance according to the predefined policies, and the performance and features of that VNF. Furthermore, the auto-scaling solution should take into account the VM startup time that can vary according to the cloud platform [5], [6], and can also be impacted by the OS image and VM type, as well as the number of requested VMs and data-center load [4].

Lastly, a multi-slice architecture means that concurrency for resources needs to be managed at the orchestrator level by setting minimal and maximal resource limits for each slice, as well as levels of priority matching their service requirements.

Algorithm 2 Scale-Out Algorithm

Require:
VID: ID of the monitored VNF.
VT: The type of the VNF.
FL: Flavor of the VNF.
SID: ID of the slice the VNF is assigned to.
CP: Type of the cloud platform on which the VNF is deployed.

1: **if** $prediction(t0 + startupTime$[VT][FL][CP]$)>= maxThreshold$[VT][FL] **then**
2: **if** $allocatedInstances$[SID]$<maxAllocate$[SID] **then**
3: $requestResource$(FL);
4: newVID=$scaleOut$(FL,VT);
5: $loadBalance$(VID,newVID);
6: $allocatedInstances$[SID] + + ;
7: **end if**
8: **end if**

To set the appropriate threshold for the aforementioned policies, we should determine the maximum traffic load that each security VNF can process given a certain amount of resources without dropping packets or inducing latency; in that way, the scaling can be performed in a proactive manner, thus ensuring continuity of service. In the next section, we will present the methodology for evaluating the performance of each security VNF.

IV. Methodology

A. Evaluated Virtual Network Functions

1) Network Intrusion Detection/Prevention System: IDS/IPS is a network appliance which captures and analyzes network traffic, to detect and prevent attacks against the system. It monitors and logs the traffic for signs of malicious activity generating an alert upon discovery of a suspicious event. In this work, we will be using two Open Sources of IDS:

Snort is a cross-platform signature-based Network IDS (NIDS) that can be also configured to run as an IPS [29]. The analysis of packets is performed using a large set of signature-based rules.

Suricata has been developed by the Open Information Security Foundation (OISF) as an alternative to Snort [30]. Similarly to Snort, it can act as an IPS and perform packet inspection in the same way. Snort's ruleset can be also imported. An important additional feature compared to snort is the support for multi-threading, which allows optimal multi-CPU usage.

2) Deep Packet Inspection: DPI engines inspect network packets up to the Layer 7. They are used to prevent sophisticated attacks such as viruses and worms. Classified packets can be redirected, marked/tagged, blocked, rate limited, or reported to a monitoring system within the network.

Ntopng [31] is a cross-platform Open Source DPI based on *libpcap* and the *DPI* [14] libraries that can analyze and sort network traffic at the application level according to different criteria, and produce detailed statistics and reports of the different application flows. It can also detect suspicious activities and allows blocking malicious flows.

B. Experimental Evaluation

We evaluated the previously mentioned VNFs using the hardware described in Table I, whereby our virtualized environment was setup over a VMware ESXi Hypervisor on a dual Intel $E3-1231$ computer node. Using ESXi, we were able to deploy three configuration flavors for our benchmarking as detailed in Table II.

TABLE I
TESTBED HARDWARE CONFIGURATION.

Component	Configuration
CPU	2 x Intel Xeon CPU E3-1231 (4 Cores) v3 $3.40GHz$
RAM	$16GB$
Links	1000Mbps
Hypervisor	VMware ESXi6.0.0

As for the software versions we used, they are **Snort**, version 2.9.6.0; **Suricata**, 3.2RC1; and **Ntopng**, 2.4.170215. Snort, Suricata, and Ntopng were evaluated on the GNU/Linux Ubuntu 14.04 Operating System. Furthermore, we vary the traffic load and measure software's performance for each rate. This was carried out leveraging *hping3* to send different types of traffic from multiple hosts and at different rates. Our goal was to overload the evaluated VNF for each flavor to assess their performance limitations.

TABLE II
DEPLOYMENT FLAVORS.

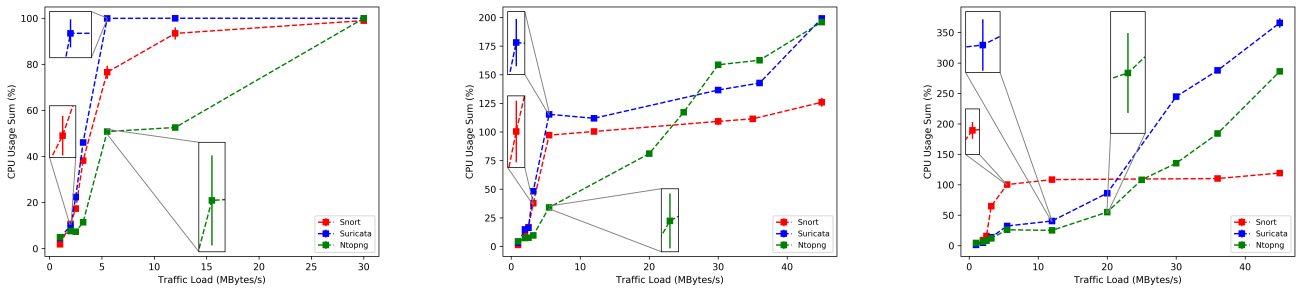
Deployment Flavor	Mini	Small	Medium
CPU	1	2	4
RAM (GB)	1	2	4

Based on our bibliographic revision, we chose to evaluate the computational environment under test using the following metrics: *CPU Usage*, *Packet Processing Speed*, and *Packet Loss*. It is worth noting that the two latter metrics are crucial for our evaluation as a slow packet processing speed indicates that the component is creating a bottleneck in the network in case of an in-line VNF: in case it is an IDS, the latency could lead to an important delay in attacks' detection and response. On the other hand, if an IDS starts dropping packets without analyzing them, the rate of false negatives would highly increase. We were able to measure the aforementioned metrics using the built-in logging features of Snort, Suricata, and Ntopng. Memory usage was not considered as a metric in our evaluation because its value was stable during all the experiments.

C. IDS/IPS

Fig.2 presents the results of our evaluation. We plot the mean and 95% Confidence Interval (C.I.) of five executions. The Y-axis shows the CPU utilization for each of the applications, while the X-axis shows how much bandwidth we input in each test.

Fig. 2(a) shows the performance of our reference IDS/IPS's, when running in our Mini VM. We conduct our experiment until all the CPU utilization was at 100%. However, it shall be noted that this is not an indication that any of this application dropped packets as we will discuss this later in this section. Still in Fig. 2(a), our results show that Suricata quickly starts to consume 99.955% of CPU at $5.5\ MB/s$, with a 95% C.I. of 0.394%. Snort increases its CPU usage as fast as Suricata until it reaches 76.562% at $5.5\ MB/s$ whereas its 95% C.I was 2.858%, reducing its increase rate as it only consumes 98.985% of CPU at $5.5\ MB/s$ with 95% C.I. of 0.565%. Meanwhile, Ntopng exhibits a linear CPU utilization profile, reaching 99.955% of CPU usage at $30\ MB/s$, with the exception of results for $5.5\ MB/s$ and $12\ MB/s$, where its CPU utilization was 50.791% and 52.567%, respectively.



(a) CPU Usage of Snort, Suricata, and Ntopng on the Mini VM.

(b) CPU Usage of Snort, Suricata, and Ntopng on the Small VM.

(c) CPU Usage of Snort, Suricata, and Ntopng on the Medium VM.

Fig. 2. Scalability of Snort, Ntopng and Suricata.

Fig. 2(b) presents the results for our small VM flavor. Here, it can be easily observed that Snort is unable to use the extra core, while Suricata still saturates one core at 5.5 MB/s. However, after that point it reduces its increase rate arriving at 200% (2x 100%) at 45 MB/s. Ntopng also reaches 195.966% of CPU usage at the same input traffic albeit with a more linear increase rate than the others. Finally, Fig. 2(c) shows that the initial CPU usage of Suricata only occurs for the one CPU in the system, as in our experiments with our medium VM it showed a CPU profile similar to Ntopng.

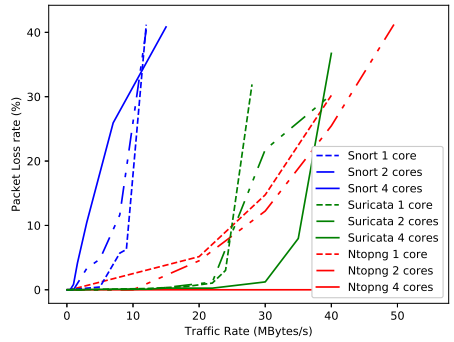
1) Scalability: The results in Fig. 2 illustrate that scaling-up resources for Snort does not improve its performance since it does not support multi-threading and therefore can only use one CPU at a time. A solution would be then to scale-out by creating additional Snort instances and performing flow-based load-balancing between them. However, splitting the traffic between several instances increases the risk of false negatives [12]. Indeed, for threshold-based rules, it would take more time to detect an attack. Moreover, as previously mentioned, a multi-instance Snort [10] has a worse performance than a single-instance Suricata given the same amount of resources. In contrast with Snort, Suricata scales well when increasing its CPUs, and that is due to the fact that all the allocated CPUs are used by creating multiple threads.

2) Efficiency and Responsiveness: Fig. 3(a) shows that Snort slowly starts dropping packets while its CPU usage is still comparatively low. On the other hand, Suricata does not drop packets until its CPU becomes overloaded. The packet loss rate then increases exponentially. When comparing computing speed, we can also notice Suricata's is very close to the incoming packet's speed, thus reducing latency.

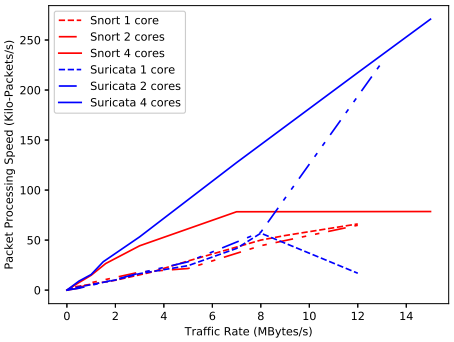
Another observation is that for four CPUs for example, Snort starts dropping packets when its CPU usage is at 64,87% while Suricata starts dropping packets only when its CPU usage is at 244,84%. This illustrates how important it is to deploy fine-grained application-specific scaling policies.

D. Ntopng

Similarly to Suricata, Ntopng scales well when adding CPUs, and the percentage of packet drops remains very low until the CPU usage reaches its maximum for all of the tested flavors.



(a) Packet Loss Rate.



(b) Packet Processing Speed.

Fig. 3. Efficiency and responsiveness of Snort, Ntopng and Suricata.

V. Conclusion

In this paper, we proposed an application-aware framework that enables Security as a Service (SECaaS) within network slices using SDN and NFV technologies. We evaluated the performance of different security VNFs. Based on the obtained results, we concluded that any auto-scaling solution would need to take into account the specific performance require-

ments and behavior of each VNF, in addition to the instance startup time and traffic load prediction in order to trigger the scaling operations. This consideration is even more critical as a security VNF's malfunction (e.g., latency and dropped packets) can compromise the security of the whole system. Moreover, we showed how SDN can be leveraged to deploy security applications, ensuring inter-slice isolation as well as intra-slice traffic control.

ACKNOWLEDGMENT

This work was partially funded by the Academy of Finland Project CSN under Grant Agreement No. 311654 and also partially supported by the ANASTACIA project, that has received funding from the European Union's Horizon 2020 Research and Innovation Program under Grant Agreement No. 731558 and from the Swiss State Secretariat for Education, Research and Innovation.

REFERENCES

[1] "https://www.gsmaintelligence.com/research/?file=9e927fd6896724e7b26f33f61db5b9d5&download," Tech. Rep.

[2] R. S. Shariffdeen, D. T. S. P. Munasinghe, H. S. Bhathiya, U. K. J. U. Bandara, and H. M. N. D. Bandara, "Adaptive workload prediction for proactive auto scaling in paas systems," in *2016 2nd International Conference on Cloud Computing Technologies and Applications (CloudTech)*, Marrakesh, Morocco, May 2016, pp. 22–29.

[3] Y. Ren, T. Phung-Duc, J. C. Chen, and Z. W. Yu, "Dynamic auto scaling algorithm (dasa) for 5g mobile networks," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Washington, DC USA, Dec 2016, pp. 1–6.

[4] Y. Govindaraju and H. Duran-Limon, "A qos and energy aware load balancing and resource allocation framework for iaas cloud providers," in *2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC)*, Shanghai, China, Dec 2016, pp. 410–415.

[5] M. Mao and M. Humphrey, "A performance study on the vm startup time in the cloud," in *2012 IEEE Fifth International Conference on Cloud Computing*, Honolulu, HI, USA, June, pp. 423–430.

[6] Y. Ueda and T. Nakatani, "Performance variations of two open-source cloud platforms," in *Workload Characterization (IISWC), 2010 IEEE International Symposium on*, Paris, France, Dec 2010, pp. 1–10.

[7] C. Yoon, T. Park, S. Lee, H. Kang, S. Shin, and Z. Zhang, "Enabling security functions with sdn: A feasibility study," *Computer Networks*, vol. 85, pp. 19 – 35, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128615001619

[8] S. Shin, L. Xu, S. Hong, and G. Gu, "Enhancing network security through software defined networking (sdn)," in *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, Hawaii, USA, Aug 2016, pp. 1–9.

[9] B. Brumen and J. Legvart, "Performance analysis of two open source intrusion detection systems," in *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, Croatia, May 2016, pp. 1387–1392.

[10] J. S. White, T. Fitzsimmons, and J. N. Matthews, "Quantitative analysis of intrusion detection systems: Snort and suricata," in *Cyber Sensing 2013*, vol. 8757, May 2013, p. 875704.

[11] T. Bujlow, V. Carela-Espaol, and P. Barlet-Ros, "Independent comparison of popular dpi tools for traffic classification," *Computer Networks*, vol. 76, pp. 75 – 89, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128614003909

[12] L. Cao, P. Sharma, S. Fahmy, and V. Saxena, "Nfv-vital: A framework for characterizing the performance of virtual network functions," in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, San Francisco, CA, USA, Nov 2015, pp. 93–99.

[13] G. Jati, B. Hartadi, A. G. Putra, F. Nurul, M. R. Iqbal, and S. Yazid, "Design ddos attack detector using ntopng," in *2016 International Workshop on Big Data and Information Security (IWBIS)*, Jakarta, Indonesia, Oct 2016, pp. 139–144.

[14] L. Deri, M. Martinelli, T. Bujlow, and A. Cardigliano, "ndpi: Open-source high-speed deep packet inspection," in *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*, Nicosia, Cyprus, Aug 2014, pp. 617–622.

[15] Amazon, "Amazon ec2 - virtual server hosting," 2016. [Online]. Available: https://aws.amazon.com/ec2/

[16] Netflix, "Scryer: Netflixs predictive auto scaling engine," 2013. [Online]. Available: http://techblog.netflix.com/2013/11/scryer-netflixs-predictive-auto-scaling.html

[17] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "Cloudscale: Elastic resource scaling for multi-tenant cloud systems," in *Proceedings of the 2Nd ACM Symposium on Cloud Computing*, ser. SOCC '11. New York, NY, USA: ACM, 2011, pp. 5:1–5:14. [Online]. Available: http://doi.acm.org/10.1145/2038916.2038921

[18] I. Oliver and S. Holtmanns, "Providing for privacy in a network infrastructure protection context," in *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, Paris, France, March 2017, pp. 79–86.

[19] T. Taleb, A. Ksentini, and R. Jantti, ""anything as a service" for 5g mobile systems," *IEEE Network*, vol. 30, no. 6, pp. 84–91, November 2016.

[20] Network functions virtualisation (nfv); management and orchestration. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf

[21] ETSI GS NFV 002, "Network functions virtualization (nfv); architectural framework v1.1.1," ETSI, Tech. Rep., October 2013. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf

[22] S. Lal, T. Taleb, and A. Dutta, "NFV: security threats and best practices," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 211–217, 2017. [Online]. Available: https://doi.org/10.1109/MCOM.2017.1600899

[23] S. Lal, A. Kalliola, I. Oliver, K. Ahola, and T. Taleb, "Securing VNF communication in NFVI," in *IEEE Conference on Standards for Communications and Networking, CSCN 2017, Helsinki, Finland, September 18-20, 2017*, 2017, pp. 187–192. [Online]. Available: https://doi.org/10.1109/CSCN.2017.8088620

[24] F. Z. Yousaf and T. Taleb, "Fine-grained resource-aware virtual network function management for 5g carrier cloud," *IEEE Network*, vol. 30, no. 2, pp. 110–115, 2016. [Online]. Available: https://doi.org/10.1109/MNET.2016.7437032

[25] M. Christodorescu, R. Sailer, D. L. Schales, D. Sgandurra, and D. Zamboni, "Cloud security is not (just) virtualization security: A short paper," in *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, ser. CCSW '09. New York, NY, USA: ACM, 2009, pp. 97–102. [Online]. Available: http://doi.acm.org/10.1145/1655008.1655022

[26] I. Oliver, S. Ravidas, L. Hippeläinen, and S. Lal, "Incorporating trust in nfvi: Addressing the challenges," in *Proceedings of 20th Innovations in Clouds, Internet and Networks Conference ICIN'2017*, Paris, France, 2017, pp. 87–91.

[27] S. Lal, S. Ravidas, I. Oliver, and T. Taleb, "Assuring virtual network function image integrity and host sealing in telco cloue," in *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–6.

[28] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "Onos: Towards an open, distributed sdn os," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '14. New York, NY, USA: ACM, 2014, pp. 1–6. [Online]. Available: http://doi.acm.org/10.1145/2620728.2620744

[29] Snort. [Online]. Available: https://www.snort.org/

[30] Suricata. [Online]. Available: https://suricata-ids.org/

[31] Ntop. [Online]. Available: http://www.ntop.org/