

# Enabling Fog Computing based Dynamic Security Service Function Chaining for 5G IoT

Vashish N. Imrith\*, Pasika Ranaweera<sup>†</sup>, Sneha Damree<sup>‡</sup>, Madhusanka Liyanage<sup>§</sup>

\*<sup>†</sup>Department of Electrical and Electronics, University of Mauritius, Mauritius

<sup>†</sup><sup>§</sup>School of Computer Science, University College Dublin, Ireland

<sup>§</sup>Centre for Wireless Communications, University of Oulu, Finland

Email: \*vashish.imrith@ieee.org, <sup>†</sup>pasika.ranaweera@ucdconnect.ie, <sup>‡</sup>damreesneha9@gmail.com,

<sup>§</sup>madhusanka@ucd.ie, <sup>§</sup>madhusanka.liyanage@oulu.fi

**Abstract**—Fog computing is an edge computing strategy which fuels adaptation of Internet of Things (IoT) in many domains. The decentralized and dynamically deployable features of the fog nodes are useful to satisfy the service requisites of IoT nodes. However, security is a prime concern for both fog, and IoT deployments; where their inhibited limited resources are expanding the threat landscape for the malicious adversaries towards resource exhaustive attempts. Despite the available security servicing tools being effective, a singular service is not adequate to address all the intricacies of the contrived threat landscape. Hence, the requirement of multiple security services to operate in a cooperative domain is an obvious fact. However, integrating multiple security services in a resource constrained fog node is a challenge. Thus, in this research, we are leveraging Service Function Chaining (SFC) concept as a method to deploy multiple security services/ tools in a resource constrained edge node (i.e. Raspberry Pi), while evaluating its adaptability in a developed experimental virtual platform. The implemented SFC strategy on the fog node is compared with a resourceful virtual machine to understand the performance issues.

**Index Terms**—IoT, Fog Nodes, IDPS, Security Services, Scalability, Performance, Orchestration, 5G

## I. INTRODUCTION

Internet of Things (IoT) is a paradigm shift for mobile service providers and electronic device manufacturers that led to contemplate their business models and innovation context. The plethora of devices coming into existence due to IoT is the major reason for such a drastic transformation. On the mobile service providers perspective, upgrading the prevailing mobile network infrastructure to cater the myriads of devices is the most challenging aspect that require a holistic architectural alteration. Further, IoT is inviting many organizations/ institutions/ companies to develop their own proclaimed products; which create compatibility concerns for service providers. Managing compatibility while establishing inter-operability are prime challenges to overcome for successful integration of IoT applications and services for the network infrastructure.

On the IoT device manufacturers perspective, embedding complex features into the apparatus for satisfying intrinsic requirements is the main objective. Emerging applications of Augmented Reality (AR), massive Machine-Type-Communication (mMTC), and Unmanned Aerial Vehicles (UAVs) catered through Ultra-reliable Low-latency Commu-

nication (URLLC), and enhanced Mobile Broadband (eMBB) guarantees of the nascent 5G technology is contriving novel requirements that varies drastically to the prevailing ones [1]. Such demanded specifications are driving the IoT manufacturers to lodge more and more processing power while making the devices more miniature. Infact, this is an impossible goal with the existing hardware assets available for the economical constrictions raised due to the competitive market. Such restrictions are forcing the researches to investigate novel means for managing the demanded computing requirements.

The interest towards development of energy optimization and harvesting strategies for resource constrained IoT devices has become popular in the current era due to the technological limitations specified earlier. Such strategies are prominently investigated with the Fog computing paradigm, where fog nodes can be successfully deployed with IoT devices [2]. Thus, the fog concept is ideal for adapting resource constrained IoT strategies. Though, fog computing is not completely tamper-proof against security attacks [3]. Apart from well-known security attacks of Botnets, Denial of Service (DoS), malicious node injection, Advanced Persistent Threats (APT), or jamming; penetrative malicious threats are posing critical effect to the entire functioning of the fog system. Therefore, novel security protection mechanisms should account for all such attacks while tackling the compatibility issues imminent with IoT applications. In addition, adversaries can exploit the resource limitations of the IoT devices for launching resource exhaustion attacks. Thus, it is obvious that a singular security function is not capable of handling the diversified security threats instigated via different layers of the IoT protocol stack.

Deploying multiple security services however, is raising issues with the limited resources available on the fog envisioned IoT nodes and at the edge level [4]. This research directive is an attempt to solve this trade-off of the application of security measures while optimizing the resource limitations. In order to deploy multiple security services, we employ the Service Function Chaining (SFC) principle, that facilitates a standardized deployment strategy across cross-platforms [5]. In spite of its wide adaptations for automating services in the industrial internet, SFC has not been considered for launching security services in the IoT or fog context. Thus, this research

is pioneering the automated security approaches, where SFC provides an edge towards achieving feasible deployment.

The remainder of the paper is organized into five Sections. Background technologies and concepts are specified in Section II while related researches are explicated in Section III. The proposed security orchestration based service architecture is proposed in Section IV. Section V describes the implementation of the experimental environment where experiments were carried out to validate the adaptability of resource constrained edge nodes against resourceful virtual machines. Finally, Section VI concludes the paper.

## II. BACKGROUND

### A. Fog Computing Edge Nodes

Fog computing leverages the computing capabilities within a local network to perform computation tasks that are generally launched at cloud platforms, hence reducing the latency with improved access capacity. This concept enables the processing of data within an edge node, also known as a fog node or an IoT gateway. Fog nodes provision cloud computing services to fog devices/users, allowing mobility support in a geographical dispersed context [6]. Virtualization technologies are employed for service deployment at the fog devices. Fog enabled IoT nodes are typically envisaged to be constrained with resources that feature low-budget processors, limited memory, low-range communication transceivers, and scarce firmware support [7].

### B. Service Function Chaining

SFC is a concept proposed to enable dynamic and flexible services deployment in softwarized 5G networks. The requirement for SFC arises when there are more than one service or Service Function (SF) to be handled for the same input, and the processing should be conducted sequentially to preserve the maximized processing capability while the outcomes of a certain service is essential to the subsequent service. In the implementation context, SFC is the incorporation of a set of running Virtual Network Functions (VNFs) that are linked together in order to work in a sequential manner [5]. In case of security with Deep Packet Inspection (DPI), each packet can be thoroughly inspected; and allows services like email (SMTP or POP3) to be investigated with the maximum available security level. Thus with SFC, a range of security services can be applicable in the context of fog enabled IoT.

### C. Docker

Docker is one of the leading container orchestration tools in the current market; that is an open source platform for developing and running distributed or multiple service instances rapidly or independently, while using less CPU overhead and enhanced network performance [8]. It enables the creation of: containers with unique container IDs, that are originating from a Dockerfile. Docker is considered as a lightweight virtualization technique while operating systems are decoupled by containers, enabling the minimal use of resources. In the containers, software can be wrapped into a complete file

system that shares the OS, while the containers use less RAM than the machine they are running. Stacks of Docker containers containing network security tools can be built at edge devices to strengthen the network security and privacy.

### D. Intrusion Detection and Prevention System (IDPS)

Network IDPSs protect the networks by preventing attacks by first detecting, alerting, and blocking or isolating the infected packets from accessing the network with the firewall resembled functionality. The detection is based on either signature based or anomaly based. The two open source IDPSs that we have chosen in this paper are Snort [9] and Suricata [10]; that can be easily installed in an Alpine image. These IDPSs have their own predefined set of rules that can be obtained from their online repositories [11].

## III. RELATED WORK

Boudi et al. in [12] emphasized on the use of lightweight virtualization technologies to implement security services in resource-limited edge nodes, to reduce latency and network traffic overhead. Security functions were executed in Docker containers. Testing and comparison of Suricata running on both bare metal (SoBM) and in a Docker container (SoDC) proved that Docker is a lightweight alternative for virtual machines, where even though only one Docker container was running to evaluate performance, it had total control over the network interface. Hence, the implication of the characterization of low overheads with containerization. Though, this research did not extend to formalizing multiple security services at the edge nodes.

Imrith et al. in [13] investigated the viability of launching several security instances comprising IDPS tools such as Snort and Suricata in a fog node, managed by a Security Orchestrator to enhance privacy and security services. To overcome the resource scarcity in the fog nodes, Docker as a lightweight virtualization technology was employed. Experiments on performance and resource utilization of multiple instances of deployed IDPSs were performed, with different ranges of traffic data rates; depicting the difference in characteristics between these two IDPS tools, while proving the feasibility of multiple service support at the resource constrained fog nodes. This paper can be considered as an extended research to [13], where SFC is utilized as an orchestration strategy to optimize the service flow.

In [14] the authors analysed the container-based mapping approach which leveraged VNF dependent links and incorporated resource limited architecture into a physical network. The paper proposed the use of container-based SFC while mapping into Network Function Virtualization (NFV) networks. A scheme was proposed in order to leverage resource utilization and reduce the cost whereby the system performs SFC mapping by taking interconnected bandwidth requirements and functional dependencies between VNFs in to consideration.

Zhou et al. in [5] worked on the cost minimization of the SFC process as a strategy to effectively orchestrate the edge offloaded services online. This approach leads to optimal

resource provisioning and traffic routing in real-time scenarios with maximal cost-efficiency. The holistic cost incurred by an edge based SFC deployment is formed as a NP hard problem that constitute SF running cost, SF switching cost, cloud outsourcing cost, and traffic running cost. Regularization techniques are adopted to decompose the formed long-term problem while a randomized dependent scheme is followed to simulate the cost efficiency via the formalized two algorithms. The simulation employs security SFs in addition to proxy and NAT services conditioned according to real-world workload traces. The proposed cost-optimization models will be vital for future containerized deployments of SFC at the edge platforms. TABLE I justifies the contribution of this paper in regard to state-of-the-art.

TABLE I  
COMPARISON OF NOVEL SFC BASED RESEARCH DIRECTIVES TO THE OBJECTIVES OF THIS PAPER

Objectives	[12]	[13]	[14]	[5]	Ours
SFC Concept			✓	✓	✓
Resource Constrained IoT	✓	✓			✓
Strategic Orchestration Goals		✓		✓	✓
Employing Security SFs	✓	✓		✓	✓
Edge Computing	✓	✓		✓	✓
Fog Computing	✓	✓			✓
Edge Compatible Architecture		✓			✓
Adapting Virtualization	✓	✓	✓		✓
Containerization/ Docker		✓	✓		✓

#### IV. PROPOSED ARCHITECTURE

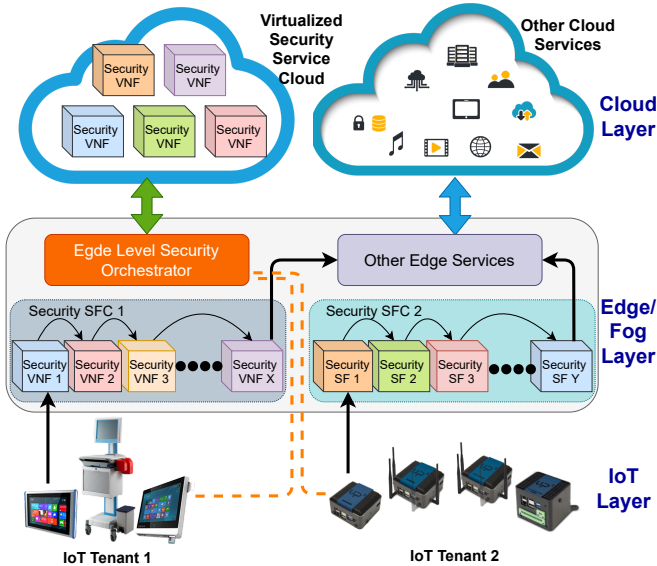


Fig. 1. Proposed Architecture

The proposed architecture depicted in Fig. 1 focuses on implementing dynamic orchestration strategies for resource constraint edge nodes. Based on OpenFog reference architecture mentioned in [15], there exist a separate layer for managing the node or simply, in performing orchestration

under the edge layer. This prescribed layer involves in hardware virtualization, security, and node management either in network, computing, storage, or in accelerated resources. The fog nodes are operating within the edge layer. Our proposed architecture would adhere to the OpenFog architecture that would present a path in the implementation or orchestration with high security assurances.

##### A. Edge Level Security Orchestrator

An Edge Level Security Orchestrator (SO) is capable of configuring containers automatically in accordance to the specifications of the security services that are available as Docker images in the local edge level repositories. The docker containers, or VNFs are monitored by the SO for logging resource allocation and load balancing. In the context of this paper, the security service based SFC processes are handled by the SO, that recall the relevant VNFs from the cloud layer to form the SFC strategy/ sequence. The offloaded traffic/ parameters from the IoT tenants in the IoT layer are forwarded towards the SFC sequence, where the outcomes are conveyed to the core-level orchestrator for decision making. In the security context, offloaded content becomes the ingress traffic while the outcome is to detect the intrusions from the IDPS acts. In addition to the orchestration of docker containers, following strategies are expected to be performed by the SO.

1) *Multiple Tenant Support*: To provision multi-tenant access capability to the edge layer, an efficient and temper-proof access control mechanism is crucial. This functionality should also be administered by the SO, where a light-weight authentication mechanism would be ideal considering the IoT context [16]. Formalizing an access control mechanism in a virtualized / Dockerized environment is a challenge that yields effective identification and isolation among the containers. The higher scalability featured by IoT demands servicing multiple tenants from a single service instance or a SFC flow/ sequence. Thus, apart from being secure, proper distinguishing among the access channels should be achieved in case of a same SFC flow being accessed by several tenants.

2) *Performance Optimizing and Dynamic Resource Allocation*: Resource consumption of containers should be monitored real-time for detecting possible service disruptions via resource depletion, and the services should be migrated to a new fog node in such a circumstance. Optimizing the performance in real-time is quite challenging and depends on several factors; specifically for SFC based deployments [5]. Even the order or the sequence of the services in the SFC flow is vital for optimized performance. Further, SO is tasked with limiting the resources of each service instance if the priority is for serving maximum number of tenants simultaneously. Thus, balancing the performance for maximum output while altering the resource utilization is an optimization research directive worth investigating.

3) *Classification of Ingress Traffic*: Nowadays with the variety of devices connected to the Internet, we have a range of packets that are received by servers. Starting from video packets to messaging ones, each one have its own specification

and size. Each of them are treated differently when it comes to security services as seen in Fig. 2. Thus the proposed SO should classify the ingress traffic; differentiated and send to their respective security services. The packets will be treated differently, each type of packet individually instead of all type of packets together. In this way, the containers acquiring these packets will run constantly and smoothly, and the IDPS will need only a certain limit of rules, related to the packet to be coming. We believe that lesser rules implements, faster the alerts will be detected and lesser packet drops will occur.

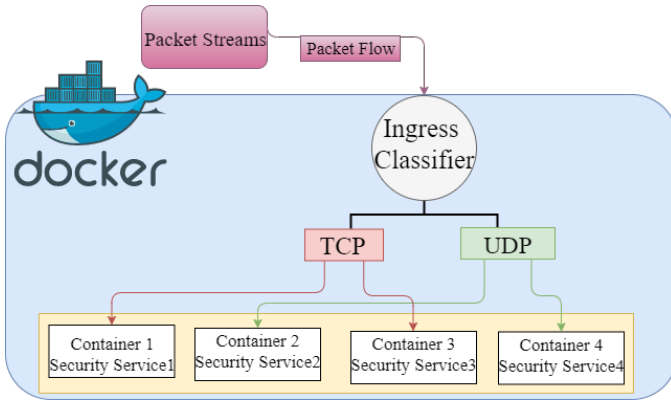


Fig. 2. Ingress classifier

### B. Cloud and Edge Layer Interventions

The functionalities of the cloud layer has been eased by our architecture through the edge layer intervention, where only storage and centralized controlling are performed at the cloud. In Fig. 1, the SFC plays an important role for security services that improves the reliability and efficiency of the service delivery with optimized resource consumption. The SO has the ability to launch as many service function chains as required depending on the resource consumption monitored real-time. This flexibility in launching dynamic service functions is the key to managing overloading ingress traffic towards the edge layer. In the long run, SFC aids to dwindle the load on the cloud layer entities.

### C. Development Strategy of the Proposed Edge Layer

In our proposed architecture we are focusing on the IDPS systems that has the ability of monitoring network intrusions as the main security services. For multiple devices to be able to work on a platform, we are proposing the creation of an Internet Protocol (IP) based network. We decided to employ OpenVSwitch (OVS) in our architecture since the former Docker bridge network has limited interoperability and scalability issues. A virtual switch as OVS [17] would assign an IP address being on the same bridge for each containers to be created. These security services will have a unique IP address and will be their only way to be distinguished as there may be several containers running same security services. SO will have the job in monitoring and updating the list of security services and the IP assigned to them.

To prevent unauthorized intrusion towards the system, Alpine Wall firewall management framework can be leveraged within the Linux environment [18].

## V. IMPLEMENTATION AND EVALUATION

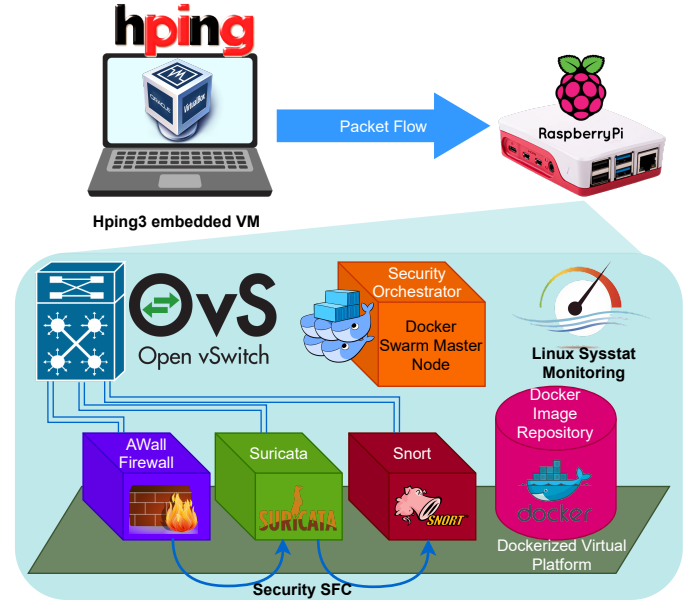
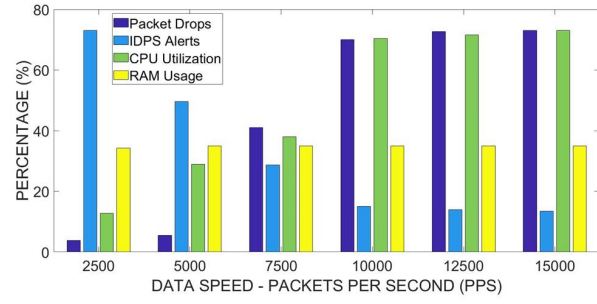


Fig. 3. Experimental Platform

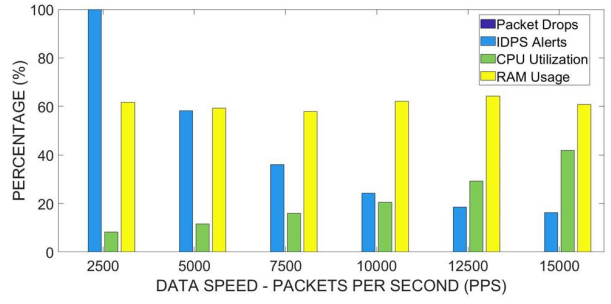
A virtual platform was developed employing a Raspberry Pi (RPI) as the resource constrained fog node; and was set up to accommodate packets coming from a packet emulator deployed in a PC. Hping3 penetration testing tool was used as the packet emulator for this experiment. Docker containerization was used with Alpine images as their operating system. The security orchestrator function was leveraged under the Master-node of the docker swarm mode. The containers were set up with distinct IP addresses where emulated packets can be extracted independently. The specifications of the employed RPi (ver. 4) and the PC are mentioned in TABLE II. The Hping3 was running in a Virtual Machine (VM) created in the PC employing VirtualBox tool. The firewall and the IDPS SFs were launched as Docker containers. The sequence of the security services follows the order AlpineWall firewall, Suricata, and Snort. The intention of employing a VM is to make the comparison of the performance convenient with the RPi as a fog node, and to validate its adaptability. The developed virtual platform is depicted in Fig. 3. The respective VNFs acting as the SFs are deployed in containers on a bridge network. The flow of the packets can be modified and changed by the SO. We deployed OVS to route the traffic between containers. And the performance of the containers were measured and monitored employing Sysstat tool.

### A. Aspects of the Experimental Setup

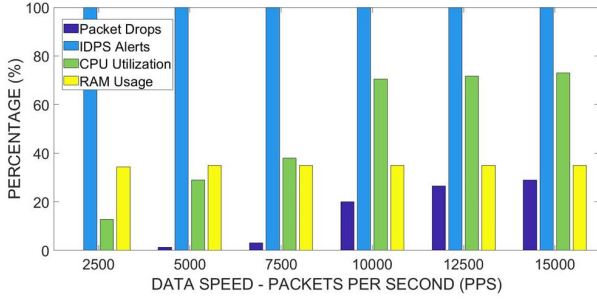
1) *Network Traffic Emulation:* Alerts were generated from a traffic emulator at different rates. TCP/IP packets with TCP,



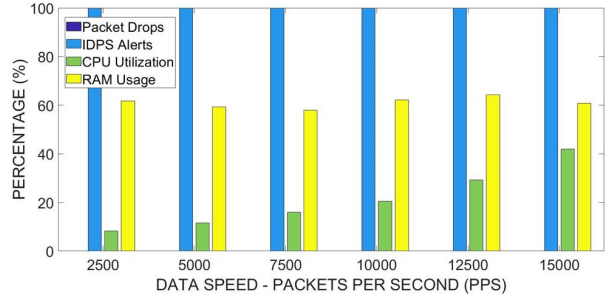
(a) Performance of Suricata Instance in the Raspberry Pi



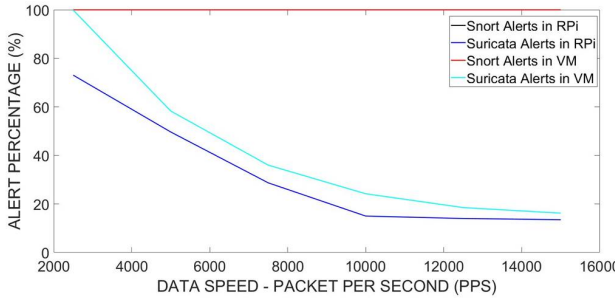
(b) Performance of Suricata Instance in the VM



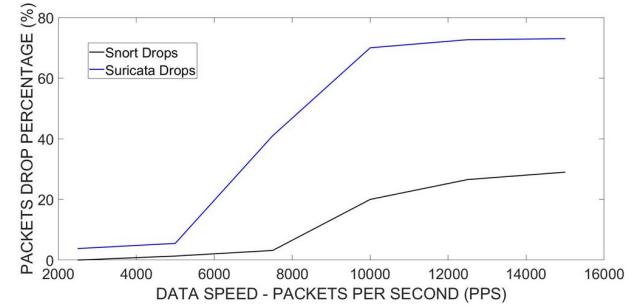
(c) Performance of Snort Instance in the Raspberry Pi



(d) Performance of Snort Instance in the VM



(e) Alert Percentage Comparison in Suricata and Snort for RPi and VM



(f) Comparison of Drops in Suricata and Snort for Different Data Speeds

Fig. 4. Combined Experimental Results

TABLE II  
SPECIFICATIONS OF THE EXPERIMENTAL PLATFORM

	PC	Raspberry pi
CPU	Intel(R) Core(TM) i7 1.80GHz 8 Core	Quad Core Cortex A72 1.5GHz
RAM	8GB DDR4	4GB LPDDR4
OS	Ubuntu 20.04	Raspbian Debian Buster

UDP and ICMP packets were generated as seen in Fig. 3. The intention behind this testing scenario was to check the amount of resources being consumed by each VNF when the traffic flows through the firewall, Suricata, and Snort respectively; where each IDSP execute a heavy rule set. This variation of packets will give a range of workload to the IDPS and the firewall. The Packets Per Second (PPS) value was incremented to capture the behaviour of the IDPS in terms of the drop packets. The varied range would also allow the CPU to vary the workload, whereby this workload will have a direct impact on the packet drops and alerts.

2) *IDPS Rules*: A signature based detection mechanism was followed with both the IDPSs for emulating this research directive as this is a proof-of-concept. As rules play a vital role in the IDPS, users have the ability to choose from a number of rules that are available in both IDPSs. Suricata has its rules automatically updated while Snort need to be updated from its repositories. We have used the open source rules for Snort. We kept the number of rules as default for both Snort and Suricata. Snort and Suricata embed 12900 and 26000 signature-based rules respectively. We have used Suricata 5.0.0 and Snort 2.9.15 with the registered rule-set.

### B. Testing Scenarios

Experiments were conducted to validate the adaptability of SFC considering a scenario where the packet flow was directed through network based IDPS services and a firewall. The tests showed that SFC was able to accept very higher rate of packets per seconds in both VM and the fog node as seen in Fig. 4. The higher packet drops in the fog node compared to the VM

is due to VMs' superior processing power. The fog node has started to drop the packets at around 7500 pps, which is around 50-60 Mbps for a packet size of average 358 bytes. It should be noted that packets exceeding 800 bytes in size exhibit lesser packet drops on the fog nodes as seen in [13]. Thus, small sized packets were employed to run the experiments on the drop rate.

#### 1) Performance of IDPS tools with the ingress data rate:

The drop rate of only 0.13 % was observed due to higher processing capability of the VM with Suricata performing. In contrast, higher packet drop was observed with RPi, due to the difference in processing capability. The packet drops for Suricata raises up to 73 % whereas for Snort, the maximum packet drop was 29 %. According to the results, optimum data rates for Snort and Suricata are approximately 7000 PPS and 10 000 PPS respectively for the RPi and the VM. We have tested Suricata and Snort by using a packet emulator that had a very specific number of packets and were being sent for a certain period of time. Although we had packet drops, we were able to match 99 percent the amount of packets per seconds(pps) sent and received.

2) Comparing the performance and resource utilization with multiple range of data rates: In this testing scenario, the effect of running multiple containers in parallel with the same IDPS is experimented to determine the performance and resource metrics for Suricata and Snort.

a) Suricata in RPi: There is a gradual accumulation of CPU performance as seen in Fig. 4(a) for the PI, that was noticed during the tests. The increment of CPU utilization was due to the higher arriving data rate at the instances, and the instances had to find a matching rule from the rule engine out of over 26000 rules. When the latter happens, there is a high consumption of the CPU. Comparing this result to the VM's ones in Fig. 4(b), the VM was having a reasonable CPU usage with zero drops and nearly 35 % of alerts detected for Suricata and 100 % packet detection for Snort.

b) Snort: For Snort in RPi and VM in Fig. 4(c) and Fig. 4(d) respectively, we have seen a very high improvement compared to Suricata since we had very less packet drops as seen in Fig.4(f). Although being a single core IDPS and having 12000 rules, Snort was constantly responding with 99.9 % of alerts detection. This makes sense as Snort has less than half the amount of signature rules than Suricata, therefore finding the ideal signature takes lesser time.

It can therefore be concluded that Snort was more reliable compared to Suricata. However, the RPi has also proved to be efficient for lightweight virtualization for same amount of traffic used in VM, which can be cost-efficient.

## VI. CONCLUSION

In this paper, our focus was to implement a security strategy for deploying multiple security services operated in a sequential manner, at a resource constrained edge node. We have evaluated multi-instance chaining called as SFC and it proved realistic in both VM and RPi cases. The deployment of SFC proved feasible at the RPi, to a pragmatic PPS

range. Further, proposed ingress classification and resource optimization orchestration strategies would bring a lot of smoothness in acquiring packets, reduction of packet drops and better number of alerts. SFC was successfully validated, showing a greater reliable security strategy to combat malware in an IoT environment with signature based rules from the IDPS and the firewall blocking mechanism.

## ACKNOWLEDGMENT

This work is partly supported by 6Genesis Flagship (grant 318927) project.

## REFERENCES

- [1] P. Ranaweera, A. D. Jurcut, and M. Liyanage, "Realizing multi-access edge computing feasibility: Security perspective," in *2019 IEEE Conference on Standards for Communications and Networking (CSCN)*. IEEE, 2019, pp. 1–7.
- [2] C.-H. Hong and B. Varghese, "Resource Management in Fog/Edge Computing: a Survey on Architectures, Infrastructure, and Algorithms," *ACM Computing Surveys (CSUR)*, vol. 52, no. 5, pp. 1–37, 2019.
- [3] D. Puthal, S. P. Mohanty, S. A. Bhavake, G. Morgan, and R. Ranjan, "Fog Computing Security Challenges and Future Directions [Energy and Security]," *IEEE Consumer Electronics Magazine*, vol. 8, no. 3, pp. 92–96, 2019.
- [4] P. Ranaweera, V. N. Imrith, M. Liyanag, and A. D. Jurcut, "Security as a service platform leveraging multi-access edge computing infrastructure provisions," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.
- [5] Z. Zhou, Q. Wu, and X. Chen, "Online Orchestration of Cross-edge Service Function Chaining for Cost-efficient Edge Computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 8, pp. 1866–1880, 2019.
- [6] M. Aazam, S. Zeadally, and K. A. Harras, "Fog computing architecture, evaluation, and future research directions," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 46–52, 2018.
- [7] Q. Zhu, B. Si, F. Yang, and Y. Ma, "Task offloading decision in fog computing system," *China Communications*, vol. 14, no. 11, pp. 59–68, 2017.
- [8] M. Alam, J. Rufino, J. Ferreira, S. H. Ahmed, N. Shah, and Y. Chen, "Orchestration of microservices for iot using docker and edge computing," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 118–123, 2018.
- [9] Cisco. Snort Network IDPS. <https://www.snort.org/>. Accessed: 2021-07-28.
- [10] OISF. Suricata IDS. <https://suricata.io/>. Accessed: 2021-07-28.
- [11] M. Ghaznavi, N. Shahriar, S. Kamali, R. Ahmed, and R. Boutaba, "Distributed service function chaining," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2479–2489, 2017.
- [12] A. Boudi, I. Farris, M. Bagaa, and T. Taleb, "Assessing lightweight virtualization for security-as-a-service at the network edge," *IEICE Transactions on Communications*, vol. 102, no. 5, pp. 970–977, 2019.
- [13] V. N. Imrith, P. Ranaweera, R. A. Jugurnauth, and M. Liyanage, "Dynamic orchestration of security services at Fog nodes for 5G IoT," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.
- [14] N. Siasi, M. Jasim, J. Crichigno, and N. Ghani, "Container-based Service Function Chain Mapping," in *2019 SoutheastCon*. IEEE, 2019, pp. 1–6.
- [15] OpenFog, "OpenFog Reference Architecture for Fog Computing," *OPFRA001*, vol. 20817, p. 162, 2017.
- [16] A. Shahidinejad, M. Ghobaei-Arani, A. Souri, M. Shojafar, and S. Kumari, "Light-edge: A lightweight authentication protocol for iot devices in an edge-cloud environment," *IEEE Consumer Electronics Magazine*, 2021.
- [17] J. Stringer, "Openvswitch without Open vSwitch: The API and its users," 2017.
- [18] Alpine-Linux. Alpine Wall Firewall. [https://wiki.alpinelinux.org/wiki/How-To\\_Alpine\\_Wall](https://wiki.alpinelinux.org/wiki/How-To_Alpine_Wall). Accessed: 2021-07-28.