

Towards Application-Aware Provisioning of Security Services with Kubernetes

Roberto Doriguzzi-Corin^α, Silvio Cretti^α, Tiziana Catena^β, Simone Magnani^α, Domenico Siracusa^α

^αCybersecurity, Fondazione Bruno Kessler, Italy, ^βDIET department, University of Roma Sapienza, Italy

Abstract—In network security, Network Function Virtualization can be exploited to implement flexible security services tailored to specific user needs. However, in practice this is hard to achieve due to the limitations of reference software platforms, such as Kubernetes, which are designed to orchestrate cloud-native services. In this work, we complement Kubernetes with a state-of-the-art algorithm for application-aware provisioning of security services. We demonstrate that the proposed solution improves basic provisioning mechanisms, such as the default Kubernetes scheduler, in terms of Quality of Service and security guarantees for the users.

I. INTRODUCTION

In the past decade, we have witnessed an increasing popularity of network softwarisation technologies, such as Network Function Virtualization (NFV), Software Defined Networking (SDN) and Service Function Chaining (SFC) [1], [2]. Much of this success is centered around the automation of network and workload management, achieved by replacing specialised appliances and middleboxes with Virtual Network Functions (VNFs), deployed on commodity hardware, that can be dynamically configured according to the specific needs and constraints of the target application.

In this context, a significant research effort has been devoted to the problem of network service provisioning in various use-case scenarios, including industrial control systems, eHealth, connected vehicles, Internet of Things, among others. A promising area of application for network softwarisation technologies is in network security, where chains of Virtual Security Network Functions (VSNFs), i.e., security-specific network functions such as firewalls or Intrusion Prevention Systems, can be created and configured to implement the desired security policies. In our previous studies [3] [4], we proposed PESS, a solution for the dynamic provisioning of security services in softwarised networks that implements a range of security constraints plus the concept of application awareness. The key idea is to reduce as much as possible the computation on the user traffic by deploying application-specific security services, such that the traffic of each application is processed only by the set of VSNFs that are necessary to fulfil its specific security requirements. The goal is to save resources by reducing the burden on the computing nodes and to minimise the latency overhead on the user traffic introduced by unnecessary VSNFs (including network and virtualisation layers traversed by the packets to reach such security functions). While meeting the security constraints, the application-aware provisioning approach of PESS reduces the utilisation of computing resources as well as the end-to-end

latency on the user traffic compared to common user-centric solutions (which we will refer to as *application-agnostic*), where security policies are implemented at user level, disregarding the specific requirements of user's applications.

In this work, we demonstrate the benefits of the application-aware approach for provisioning security services in real-world settings within a computing environment orchestrated with Kubernetes (K8S) [5]. To do so, we have first implemented an adapter to interface the K8S scheduler with the PESS application-aware heuristic and an implementation of the application-agnostic approach. Then, we have compared these two algorithms and the default scheduling algorithm of K8S in terms of security guarantees for the users and Quality of Service (QoS) experienced by the user's applications after the deployment of a set of security services with the three provisioning strategies. Since the K8S scheduling algorithm does not consider the network constraints (bandwidth and latency), it always deploys the VSNFs on the nodes with the highest available computing capacity. We demonstrate that this strategy might end up with deployments that do not respect the QoS needs of latency- and bandwidth-sensitive applications. On the other hand, the application-agnostic approach ignores the specific requirements of the user's applications, forcing their traffic to traverse unnecessary VSNFs. This leads to a higher consumption of computing resources that might prevent the provisioning of all the security measures. Instead, the application-aware heuristic algorithm of PESS solves the provisioning problem by ensuring that the resources are accurately utilised and that the user's security requirements are met.

II. SOFTWARE ARCHITECTURE

The proposed software architecture (Fig. 1) extends the *K8S and Istio* [6] control planes with a software layer called *K8S Adapter*, which enables external algorithms to manage computing and network resources of a K8S cluster and to drive the deployment of chains of VSNFs. By default, this task is executed by the *K8S Scheduler*, the reference K8S scheduling mechanism, which assigns workloads (such as VSNFs) to the nodes of a K8S cluster based on the computing resource availability, with no consideration of networking metrics such as link bandwidth and latency. In this work, PESS and the application-agnostic algorithms use the northbound API exposed by *K8S Adapter* to (i) build a model of the computing infrastructure by collecting topology and resource information (ii) provision security services according to the available network and computing resources by driving the *K8S*

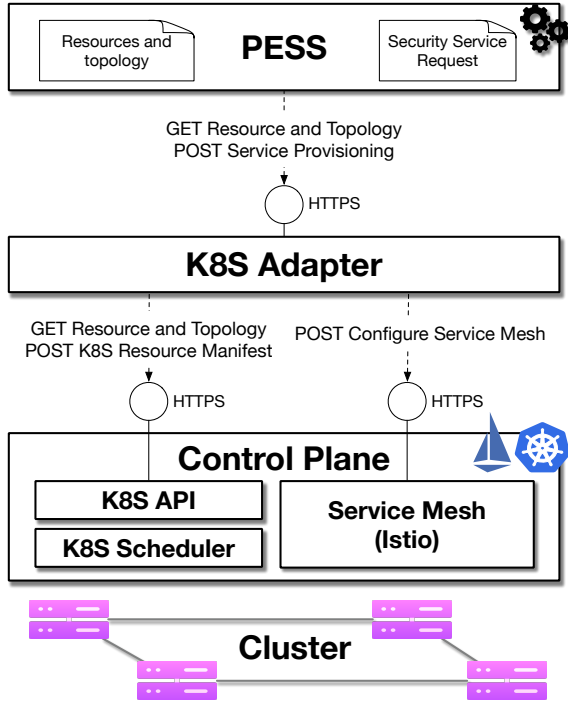


Fig. 1. Software architecture.

Scheduler and the Istio control plane in the placement of the VSNFs and in the configuration of the traffic routes among them.

III. DEMONSTRATION SETUP

The demonstration environment is a K8S cluster consisting of four workers (Fig. 2) and one master node (not shown in the figure). *Worker1* hosts the Istio Service Mesh (ISM) control plane and represents the ingress point for the users to access the web services hosted on *Worker4*. *Worker2* and *Worker3* are the nodes devoted to the deployment of VSNFs. The CPU capacity of the worker nodes is expressed in *millicores*, where one millicore (mc) is 1/1000 of a CPU core. In this regard, *Worker2* is equipped with three 2.8Ghz cores (hence 3000mc) and *Worker3* with two 2.8Ghz CPU cores (2000mc).

The workers are interconnected through virtual links, whose bandwidth (B) and latency (L) are configured with ISM and Traffic Control (tc). The values of bandwidth and latency are tuned in a way that differences between the three provisioning strategies can be shown in a controlled setup.

The demonstration focuses on deploying security services realised with chains of VSNFs, possibly with different CPU requirements. For the sake of simplicity, in this demo we implement all the VSNFs using *netfilter/iptables* [7] configured with blacklist tables of different sizes. We recall that *iptables* is a popular tool for network traffic filtering available in Linux-based systems. Moreover, *iptables* implements a packet matching algorithm, whose execution time grows linearly with the size of the matching table. We use this property to build

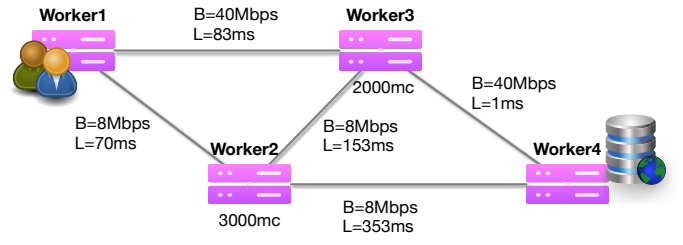


Fig. 2. Network topology of the cluster.

four VSNFs with different CPU requirements expressed as CPU cycles necessary to process a bit of traffic:

- **VSNF1:** 10000 rules, 115 CPU cycles/bit
- **VSNF2:** 5000 rules, 60 CPU cycles/bit
- **VSNF3:** 2500 rules, 45 CPU cycles/bit
- **VSNF4:** 2500 rules, 45 CPU cycles/bit

The values above include the overhead introduced by the traffic forwarding mechanism (around 25 CPU cycles/bit on average), implemented using the nginx reverse proxy [8].

IV. DEMONSTRATION

We envision a computing infrastructure where the security services are managed using the software platform described in Section II, and deployed in the form of chains of VSNFs according to the security and QoS requirements of the users. In addition, the access to the infrastructure is regulated through per-user Service Level Agreements (SLAs) that include security policies and QoS constraints. In our demonstration, we consider a user connected to *Worker1*, who accesses the web services offered on *Worker4*, including: web page browsing, file sharing and video streaming. The user's SLA contract include security-level requirements (e.g., traffic monitoring and filtering, parental control, malware detection, etc.) and QoS guarantees of 10Mbit/sec minimum bandwidth and 200ms maximum latency (Round Trip Time (RTT)).

To ensure that the security parameters specified in the SLA are respected, the infrastructure manager evaluates three different service provisioning mechanisms:

- 1) The default K8S scheduler, a consolidated mechanism that is natively integrated within the K8S control plane.
- 2) A straightforward approach, which consist of provisioning one single chain composed of all the VSNFs necessary to satisfy the security SLA. We refer to it as application-agnostic approach or AA for brevity.
- 3) PESS, which deploys application-specific chains of VSNFs. We refer to it as application-aware approach.

The goal of this work is to show that, within the constraints of the aforementioned demonstration scenario, only the application-aware approach ensure that both security and QoS parameters specified in the SLA are respected.

A. Security Policies

The infrastructure manager identifies VSNF1, VSNF2, VSNF3 and VSNF4 as the set of security functions that is sufficient to fulfil the security-level requirements specified

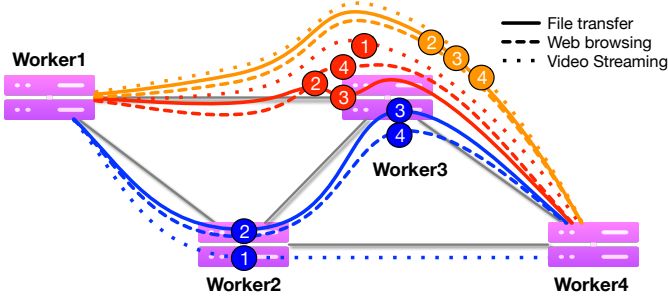


Fig. 3. Placement output with AA (orange), K8S (blue) and PESS (red).

in the SLA. With the basic application-agnostic approach, the solution is deploying a single chain VSNF1-VSNF2-VSNF3-VSNF4 to inspect and filter the whole user's network traffic. Alternatively, application-specific chains of VSNFs can be deployed using either the K8S scheduler or PESS. For the latter approach, the infrastructure manager identifies the following chains as sufficient to respect the security SLA:

- *File Transfer*: chain VSNF2-VSNF3
- *Web browsing*: chain VSNF2-VSNF4
- *Video Streaming*: chain VSNF1

B. Results

We demonstrate that inefficient service provisioning decisions can lead to the violation of the SLA contract, either preventing the deployment of all the security functions or causing a performance degradation of latency-sensitive applications (web browsing) and of bandwidth-sensitive applications (file transfer and video streaming). As shown in Fig. 3, both AA and PESS place everything on the node with the highest CPU capacity available on the only path with enough bandwidth to satisfy the QoS requirements. Note that, with the AA approach, the video streaming application cannot be secured, since there is not enough CPU to execute VSNF1 on *Worker3*. As already mentioned, placing VSNF1 on *Worker2* is not a viable option, as there is not enough bandwidth available on the paths traversing such a node. On the other hand, K8S, which disregards the QoS constraints, places the two VSNFs with the highest CPU requirements on *Worker2*, while the other two on *Worker3*.

The measures reported in Table I show that the decisions taken by K8S scheduler lead to a violation of the QoS constraints specified in the SLA contract, with a high latency experienced by the browsing application, and to a low throughput measured on file transfer and video streaming. On the contrary, both AA and PESS mechanism allow the fulfilment of the QoS constraints, but only PESS respects the security SLA by successfully deploying all the required security functions. Indeed, the inefficient AA strategy has an impact on the CPU utilisation, which goes from a baseline of around 11% at rest, to 70% of CPU load, with an increase of 59% that prevents VSNF1 from being deployed. Since PESS assigns a different chain of VSNFs to each application, we

TABLE I
COMPARISON OF RTT, THROUGHPUT AND CPU USAGE.

Approach	Application	RTT	TR	CPU
AA	File Transfer		11Mbps	59% (w3)
	Web Browsing	127ms		
	Video Streaming		11Mbps	
K8S	File Transfer		3.4Mbps	21% (w2) 19% (w3)
	Web Browsing	2004ms		
	Video Streaming		3.5Mbps	
PESS	File Transfer		11Mbps	50% (w3)
	Web Browsing	115ms		
	Video Streaming		11Mbps	

observe a lower increase of CPU load (50%), despite also VSNF1 is executed on *Worker3*.

V. CONCLUSIONS

In this work, we demonstrate the benefits of the application-aware approach for provisioning security services in software networks. To this aim, we have implemented a software platform based on Kubernetes and PESS, a heuristic algorithm to provision application-specific security services defined as chains of VSNFs. In contrast, similar demonstrations in the state of the art limit their scope to the implementation of the SFC concept in generic application scenarios [9] [10].

We have validated our platform by comparing PESS with the default K8S scheduler, and with a straightforward approach that focuses on the global security and QoS needs of the users. We have shown that only PESS is able to always fulfil the security requirements of the users and, at the same time, to respect QoS guarantees by optimising resource usage.

In our experiments, we could observe a non-negligible latency overhead introduced by the traffic forwarder implemented with the nginx reverse proxy. As a future work, we plan to replace nginx with a more efficient framework called Network Service Mesh [11].

REFERENCES

- [1] R. Mijumbi et al., "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications surveys & tutorials*, vol. 18, no. 1, pp. 236–262, 2016.
- [2] S. Khorsandroo et al., "Hybrid SDN Evolution: A Comprehensive Survey of the State-of-the-Art," *Computer Networks*, vol. 192, 2021.
- [3] R. Doriguzzi-Corin et al., "Dynamic and Application-Aware Provisioning of Chained Virtual Security Network Functions," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 294–307, 2020.
- [4] R. Doriguzzi-Corin et al., "Application-centric provisioning of virtual security network functions," in *IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2017.
- [5] "Kubernetes," 2022, <https://kubernetes.io/>.
- [6] "The istio service mesh," 2022, <https://istio.io>.
- [7] "The netfilter.org project," 2022, <https://www.netfilter.org>.
- [8] "Nginx reverse proxy," 2022, <https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/>.
- [9] J. Santos et al., "Live Demonstration of Service Function Chaining allocation in Fog Computing," in *Proc. of IEEE NetSoft*, 2020.
- [10] H.R. Kouchaksaraei et al., "Service Function Chaining Across Openstack and Kubernetes Domains," in *Proc. of the ACM DEBS*, 2019.
- [11] "Network service mesh," 2022, <https://networkservicemesh.io/>.