Automated Security Management for Virtual Services

M. Repetto, A. Carrega
S2N Lab, CNIT, Genoa, Italy
Email: {first.last}@cnit.it

J. Yusupov, F. Valenza, F. Risso
DAUIN, Politecnico di Torino, Turin, Italy
Email: {first.last}@polito.it

G. Lamanna
Infocom Srl, Genoa, Italy
Email: {first.last}@infocomgenova.it

***Abstract*—The virtualization of applications and network functions facilitates the dynamic creation of compound services, automating both the provisioning of computing/networking/storage resources and their life-cycle management. Virtualization of security appliances is a common approach to protect such services, but can neither offer broad visibility across the whole deployed service nor implement coordinated and fine-grained enforcement actions.**

This paper proposes a novel security framework based on the integration of lightweight and programmable monitoring and enforcement hooks in each virtual function, which are collectively controlled by a common logic for prevention, detection, reaction, and mitigation of security threats. Our framework keeps direct control over the functionalities of the security hooks, and leverages standard orchestration tools for management actions on the service graph. It can be automatically instantiated by common orchestration operations, hence seamlessly integrating with the deployment process of service graphs.

I. Introduction

The introduction of virtualization paradigms and software-defined infrastructures enables fully-digital workflows in the orchestration of applications and services, from dynamic resource provisioning to automatic software deployment and configuration. The large correspondence between the Infrastructure-as-a-Service model and physical infrastructures has nurtured the belief that virtual services could have been effectively protected by software instances of legacy security appliances. However, the absence of a strong security perimeter, multi-tenancy, and the different threats landscape bring this attitude into question [1]. Furthermore, the lack of interoperability and shared management interfaces also hinders the creation of common control and management frameworks, which would be necessary to bring more automation towards a true Security-as-a-Service paradigm [2] and avoid anomalies in security configuration [3].

Based on these considerations, we have already proposed a novel approach, based on the separation between pervasive and capillary monitoring and enforcement tasks and the centralized logic for prevention, detection, mitigation, and reaction [4]. This concept is now being implemented in a framework that complement existing orchestration tools. This paper shows how the framework is deployed as part of the network service graph, and how it behaves at run-time. Specifically, we demonstrate (*i*) how firewalling rules are automatically inferred by the service topology and security policies [5]; (*ii*) how heterogeneous data is collected, including logs from the operating system and applications, as well as custom network statistics. The current implementation leverages Kubernetes as service orchestrator. We describe the overall system architecture in Section II and current features in Section III.

II. Architecture

Fig. 1 shows the logical architecture of our framework, which is based on four pillars. First, the integration of lightweight monitoring and enforcement hooks in each virtual function, which can be dynamically programmed. Second, a Context Broker that hides the heterogeneity of the security hooks. Third, a Security Controller that reacts to management events and security alerts, by invoking specific security services. Fourth, an Automatic Configuration Element (ACE) and a set of specific configuration modules.

Monitoring and enforcement hooks are automatically deployed in each virtual function and consist in Logstash *beats* running in userspace (monitoring) plus kernel eBPF programs (monitoring and enforcement). They gather information from system and application logs and include both standard components (i.e., FileBeat, PacketBeat, MetricBeat) and a new one (BpfBeat) that collects measurements from eBPF programs (network statistics, system calls). The Polycube framework is used to run control applications (*cubes*) that configure the data plane. The interface exposed by the control plane is the Polycube API, while the data channel is implemented by the existing Logstash-Kafka pipeline.

The Security Controller receives notifications from the orchestrator, for example when a deployment starts/finishes, and from security administrators when a new policy is required. It invokes the ACE and carries out the required actions. Actions entail both re-configuration of the security hooks (e.g., increase verbosity, monitor additional files, measure statistics of network flows) and management operations on the service graph through the orchestrator (e.g., remove/re-deploy/terminate a VNF).

The Context Broker provides an abstraction of the security hooks in each virtual function. The abstraction includes the graph topology, current configurations (including IP addresses, received from the orchestrator), security data and events. The internal architecture of the Context Broker is based on the Elastic Stack framework (Elastic Search + Logstash); information is saved in a database for offline processing of historical data. A Kafka message broker is included to stream data to

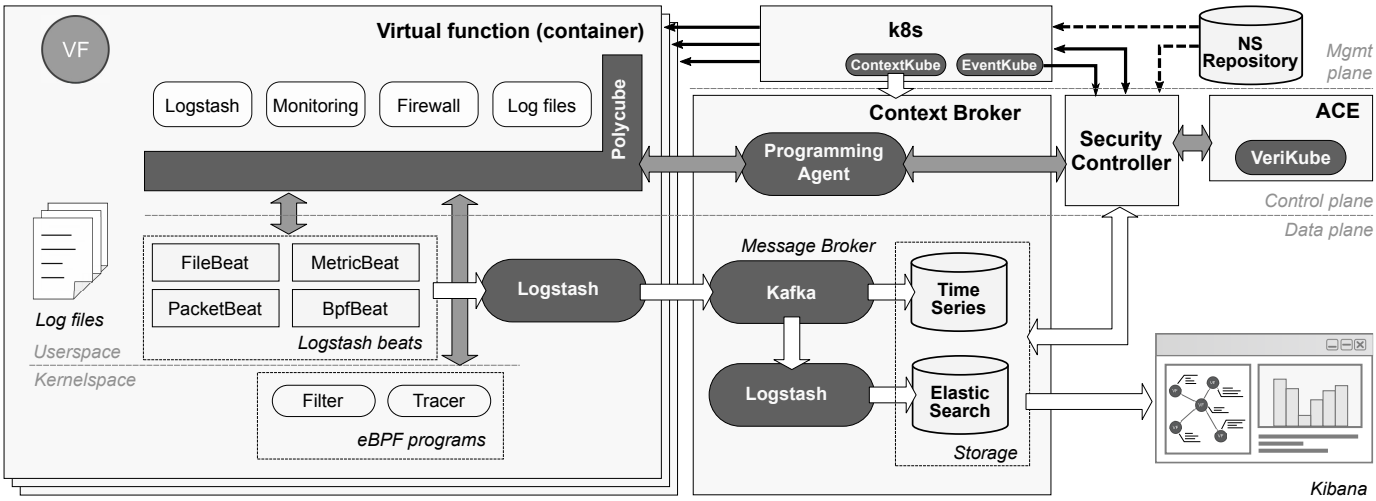

Fig. 1: Framework for security management of virtualized applications.

detection algorithms that process it in real time. Through the Context Broker, the Security Controller controls the behavior of the security hooks by changing the type, frequency, and verbosity of data and events collected.

The "smart" logic of the framework is implemented by ACE and its modules. ACE takes as input the service topology, the current network configurations, and the security policies, and returns as output the configuration of the security hooks. In the current implementation, the scope is limited to automatic firewall configuration, through a specific module named VeriKube.

We also developed two additional modules to interface Kubernetes to our system. EventKube delivers infrastructure-level events (e.g., "service has been deployed") to the security controller, whereas ContextKube gives access to management-level configurations (e.g., IP addresses assigned for management) which may not be visible outside Kubernetes.

III. Security services

There are two security services already implemented, namely automatic firewall configuration and collection of heterogeneous security context.

A typical workflow starts by deploying the service; for instance, a web-based application made of the Apache and MySQL servers. Docker images already include Polycube, Logstash, and the *beats*. After deployment and initialization, a notification is sent by EventKube to the Security Controller. At this stage, only the current configuration is monitored by the Context Broker (through ContextKube). The Security Controller retrieves the network service description from the repository and the current configuration from the Context Broker. It invokes, by ACE, the VeriKube module, which determines the firewall rules to enable communication between the servers and external clients, according to security policies (i.e. the communication requirements) which are part of the service description (e.g., Apache to MySQL, external client to Apache). Firewall rules are then returned to the Security Controller, which enables the firewalling service and pushes its configuration through the Context Broker. The correct behavior of the eBPF-based firewall can be verified by some connection attempts.

Once the firewall is operating, the set of collected logs can be progressively increased, starting from basic system logs to server logs and network statistics, so to adjust the depth of inspection to the current needs, in order to reduce the overhead. In this case, the Security Controller re-programs the local *beats* through the Context Broker, which in turn invokes the Polycube API to notify the local control "cubes." By altering the network traffic (i.e., HTTP requests, a SYN-flooding attack), it is possible to compare the verbosity and frequency of data displayed by the Kibana interface with the overhead on the network and CPU, as shown by standard performance monitoring tools (i.e., *Wireshark* and *top*).

Acknowledgment

This work was supported in part by the European Commission, under Grant Agreement no. 786922 and 833456. The authors thank Elis Lulja who contributed to the first implementation of the prototype.

References

[1] R. Rapuzzi and M. Repetto, "Building situational awareness for network threats in fog/edge computing: Emerging paradigms beyond the security perimeter model," *Future Generation Computer Systems*, vol. 85, pp. 235–249, August 2018.

[2] S. Hares, D. Lopez, M. Zarny, C. Jacquenet, R. Kumar, and J. Jeong, "Interface to network security functions (I2NSF): Problem statement and use cases," IETF RFC 8192, July 2017.

[3] F. Valenza, S. Spinoso, C. Basile, R. Sisto, and A. Lioy, "A formal model of network policy analysis," in *2015 IEEE 1st Int. RTSI Forum (RTSI)*, Sep. 2015, pp. 516–522.

[4] S. Covaci, R. Rapuzzi, M. Repetto, and F. Risso, "A new paradigm to address threats for virtualized services," in *IEEE 42nd COMPSAC*, Tokyo, Japan, Jul. 23rd-27th, 2018, pp. 689–694.

[5] C. Basile, F. Valenza, A. Lioy, D. R. Lopez, and A. Pastor Perales, "Adding support for automatic enforcement of security policies in nfv networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 2, pp. 707–720, April 2019.