Towards Novel Security Architectures for Network Functions Virtualization

M. Repetto, A. Carrega
S2N Lab, CNIT
Genoa, Italy
Email: {matteo.repetto,alessandro.carrega}@cnit.it

G. Lamanna
Infocom Srl
Genoa, Italy
Email: guerino.lamanna@infocomgenova.it

***Abstract*—The definition of elastic network services that can be orchestrated at run-time brings unprecedented agility and dynamicity in network operation, but also complicates security management. As a matter of fact, cyber-security appliances are still largely stuck to traditional paradigms, based on relatively static topologies and the security perimeter model. The uptake of service-oriented architectures and microservices is now suggesting to compose security services by orchestrating monitoring, inspection, and enforcement capabilities, which are natively implemented in each elementary component (virtual functions, software-defined network equipment).**

In this paper, we describe and evaluate a novel framework for monitoring, inspection and enforcement that provides a broad and heterogeneous security context for centralized analytics, correlation and detection. Our work represents the preliminary step towards the creation of true Security-as-a-Service (SecaaS) paradigms in virtualized environments, through programmatic composition of common capabilities available in each virtual function.

I. Introduction

Network Functions Virtualization (NFV) enables the creation of elastic network services, which can be modified at run-time according to the evolving context (e.g., by scaling one or more functions based on the workload to serve). Starting from their description in terms of templates and life-cycle management rules, network services are deployed and orchestrated over virtualization infrastructures, which resources can be provisioned and released by software APIs [1]. The possibility to add, remove, replace, and scale Virtual Network Functions (VNFs) means that the exact service topology is unknown at design time, so it is difficult to define the set of security appliances that are needed to protect the service and their position in the overall topology.

Novel trends are recently emerging to manage security aspects for service meshes, leveraging the decomposition of monolithic security appliances into modular components [2]. The general posture is towards more centralized analytics, so to capture even the weakest correlations in the time and space dimensions, which is necessary to effectively tackle the new classes of Advanced Persistent Threats (APTs). At the same time, local inspection and enforcement in each system component remains necessary to ensure the broadest visibility over the on-going context.

In this paper, we describe and evaluate a programmable and pervasive monitoring framework for NFV. The framework collects events, logs, and measurements from any VNF that is present in the service graph, and moves them to a central location for analysis and correlation. It builds upon existing technologies, and makes them more programmable at run-time. This is just the starting point for a more complete architecture, able to bring more automation and elasticity to cyber-security paradigms for NFV [3].

The rest of the paper is organized as follows. In Section II we briefly review the overall concept and architecture introduced in [3]. The contribution of this paper, namely the framework for programmable collection of the security context, is described in Section III. We presents the results of our preliminary experimental evaluation in Section IV. Finally, we give our conclusion in Section VI.

II. Concept and architecture

Relying on infrastructure-level components for building security services is a common approach today. This works well when the service is deployed in a single infrastructure, but becomes difficult (even impossible) when multiple domains are involved, with different security models, protocols, and interfaces. Indeed, this approach is conceived to protect the infrastructure, not the network services running on top of it. The alternative solution is the deployment of virtual instances of security appliances as part of the service graph, which can be orchestrated by the same mechanisms as the network service, though this approach faces performance issues and increases the cost of running the overall service.

Following the basic principles behind the Interface to Network Security Functions (I2NSF) proposal in IETF, which advocates a common control plane for configuring different security appliances [4], we already devised a new architecture to manage security of NFV services. Our approach is based on centralized processing fed by a distributed, pervasive, and capillary security context fabric, which an overall orchestration logic which takes the responsibility to translate high-level behavioral policies into low-level configuration rules [3].

Our security architecture includes two macro-elements: the Security Orchestrator (SO) and Virtual Network Functions

The final publication will be available at IEEE Xplore.

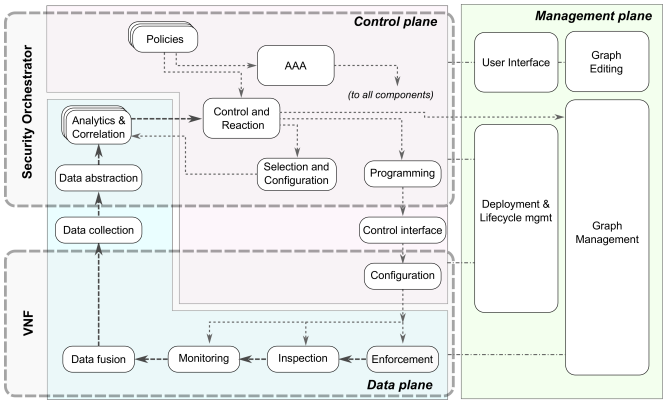

Fig. 1: The conceptual architecture to implement security services for NFV.

(VNFs), shown in Fig. 1. The SO is the smart component that analyzes security-related events and reacts according to security policies. It is not part of the ETSI MANO framework [5], though it interacts with MANO to discover the topology of the network service and to suggest remediation at the graph level (e.g., remove or replace compromised VNFs). VNFs include additional functions to gather the security context, in addition to their main business logic; such functions should be embedded in software images used to instantiate the service graph.

The architecture is made of three logical layers, which are partially implemented in both the aforementioned components. The *data plane* is composed by all the functions to create, collect, store, and analyze the security context. The security context includes all information, data, events, logs, and measurements that can be used to detect attacks and investigate new threats. In principle, it could encompass logs from the operating system and daemons, logs from network functions, lifecycle and management events, current configuration and status (e.g., IP addresses, open sockets, software version, running programs), memory dumps and execution traces (for instance, system calls or I/O commands), network statistics, packet inspection, and so on (*Monitoring* and *Inspection* blocks). The data plane is also expected to implement *Enforcement* capabilities: packet filtering and access control are the most straightforward examples. Since the security context is processed remotely, *Data fusion* can be present in each VNF to reduce bandwidth usage. *Data collection* bears the security context from VNFs to the SO over secure channels, which can be implemented by in-band or out-of-band communication with respect to the service graph. *Data abstraction* gives access to the set of heterogeneous data in a uniform way; it captures the evolving service topology and the capabilities embedded in each VNF. Finally, a group of *Analytics and Correlation* algorithms process the security context and implements typical security functions that today are provided by different appliances: antivirus, IPS, IDS, firewalling, etc.

The *control plane* supervises the operation of the data plane in the short time horizon (*Control and Reaction*). It acts according to high-level security policies; it also reacts to notifications about topology or configuration changes from the service orchestration (e.g., ETSI MANO). The main purpose of the control plane is to reconfigure the data plane, by dynamically creating security services. It chooses the set of analysis and correlation algorithms to run (*Selection and Configuration*) and it tunes the deep of inspection to the actual needs (*Programming*). For example, it can activate/deactivate the collection of logs and measurements, it can change the frequency for gathering such data, it can install filtering rules. One specific ambition is the capability of pushing dynamic *programs* to the control plane, for example to define parsing rules or processing pipelines for packets belonging to new protocols. An important feature in the control plane is Identity Management (IdM) and Access Control (AC), which are essential for safe and reliable operation of any distributed and multi-layer framework (*AAA*).

The *management plane* is responsible for all mid/long-term scale operations. It includes the definition of high-level security policies, the configuration of user identities and roles, and the representation to users (*User Interface*); the deployment of security components within each VNF, and the set up of communication channels for connection to the SO (*Graph Editing*); the notification of any topological or configuration change on the service graph (*Deployment and Lifecycle Management*); the modification of the service graph so to recover from or mitigate attacks (*Graph Management*); etc. These functions fall within the scope of NFV management, so they are already present in tools that implement the ETSI MANO architecture (e.g., OpenBaton, Open Source MANO). The only strict requirement is the presence of a remote interface (for instance, a REST API) which notifies life-cycle events and allows the invocation of management actions, as previously described.

III. Pervasive monitoring and inspection

The implementation of the security architecture described in Section II started from the data plane, as schematically depicted in Fig. 2. We created a set of hooks for lightweight monitoring and inspection tasks within the VNF, as well as a *Context Broker* (CB) that collects the data, stores and abstracts it in the SO. The solid blocks shown in Fig. 2 are already integrated and working; the opaque blocks with dashed contours are still under development/integration. As the picture shows, also the security control plane in VNFs was implemented.

The main design goals were the collection of the broadest range of data and events and the *programmability* of the inspection/monitoring processes. In this respect, we are enhancing the Elastic Stack framework[1], a collection of open-source projects for data acquisition, processing, and storage. It was originally composed of ElasticSearch, Logstash, and Kibana (for which it was formerly known as ELK) and is now evolving to include additional options. Though these tools

[1]The Elastic Stack. URL: https://www.elastic.co/elk-stack.

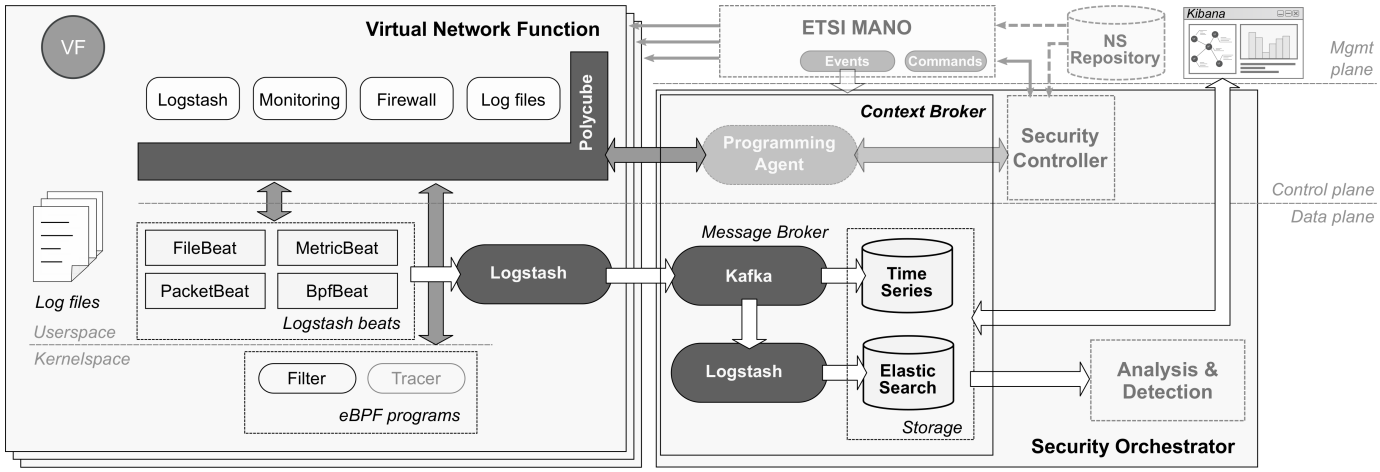

Fig. 2: Implementation of the context fabric for collecting and processing the security context.

were originally conceived to work statically, i.e. with minimal or no possibility to change the configuration at run-time, we are implementing an additional dimension of programmability, so to easily support the definition of new inspection and monitoring tasks at run-time.

Logstash is a modular data processing pipeline that gathers data from heterogeneous sources, aggregates or pre-process it, and then sends it to some remote consumer like ElasticSearch. In more detail, data is caught by a family of lightweight, single-purpose data shippers called *beats*. Several beats are already available to parse log files from popular applications (FileBeat), to read metrics from the kernel (MetricBeat), and even to perform measurements on network traffic (PacketBeat).

We developed a new beat, named BpfBeat, which reads measurements, statistics, and events generated by eBPF programs. The enhanced Berkeley Packet Filter (eBPF) is an efficient, safe, and programmable event processing framework implemented in the Linux kernel, suitable for both packet inspection and analysis of system calls. It can be used for monitoring, inspection, and enforcement tasks.

eBPF programs can be easily inject at run-time, but we need a control plane to support this feature. We adopted Polycube[2], a framework that provides fast and lightweight network functions such as bridges, routers, firewalls, and others. Polycube is made of control functions (*cubes*) implemented in high-level languages, and eBPF programs as data plane. Currently, we are using a simple eBPF programs that collects limited statistics for detection of Denial of Service, but there is virtually no limit to the classification, measurement, inspection, and enforcement filters that can be implemented and dynamically orchestrated by the Security Controller (which is the component that realizes the Control and Reaction tasks). An additional cube is currently under implementation to control other beats, so to change their configuration at run-time, as previously mentioned. A further cube will be implemented to load processing tasks in Logstash, through the Polycube interface. Indeed, Polycube offers a REST API for remote control, which in our framework implements the control channel envisioned by the overall architecture.

The data channel is implemented by Kafka[3], a message bus system for building real-time data pipelines and streaming apps. Though Logstash could directly feed the ElasticSearch database, this additional component can directly steer measurements towards detection applications, hence reducing the latency for real-time processing. Currently, both instances of Logstash, namely in VNFs and the Context Broker, are simply used to add timestamps to data, so to evaluate the latency introduced by the different stages.

ElasticSearch is a NoSQL search and analytics engine, a perfect choice for storing unstructured data and performing queries on graph-based topologies. We already created a data model for the security context, which is able to describe the service topology and relate data to its origin in the service graph. A REST interface was also defined to add/remove graph elements (i.e., VNFs and communication links) in the database. This API will be invoked on reception of topology change notifications from the ETSI MANO-compliant orchestrator. The REST interface also includes common queries (e.g., get specific data for all VNFs, get time-series for a single or multiple VNFs) that can be used by analysis algorithms, so to remain unaware of the underlying database technology.

For the mere purpose of demonstration and validation, we also included a graphical interface to query and visualize data from ElasticSearch with charts and graphs. Kibana represents an optimal tool for this purpose; the long term goal is its seamless integration in a more powerful tool for security management, also integrating the interface to the ETSI MANO framework.

IV. Experimental evaluation

We set up an experimental testbed for preliminary validation and identification of performance gaps. Our aim was to analyze

[2]Polycube.network. URL: https://github.com/polycube-network/polycube.

[3]Apache Kafka. URL: https://kafka.apache.org/.

VNF	OS	vCore	vRAM	Beat
Apache	Debian 4.9	1	1 GB	FileBeat
MySQL	Debian 4.9	1	1 GB	MetricBeat
mini_httpd	Ubuntu 18.04.2 LTS	2	2 GB	BpfBeat
CB	Debian 4.9	4	4 GB	–

TABLE I: Configuration of the testbed.

the trade-off between overhead and granularity of reporting, so to identify performance constraints that deserve further investigation in the future. In this respect, the primary objective was to validate the different *beats* and the collection framework described in Sec. III. For this reason, we did not create a realistic network service, but chose a group of heterogeneous applications that need different monitoring hooks.

We deployed three virtual machines (VMs), hosting an Apache web server, a MySQL server, and a mini_httpd server. We deployed the Context Broker in a separate VM too. We installed FileBeat, MetricBeat, and BpfBeat in different VMs, according to Table I. The same Table also reports the configuration of the VMs; all VMs ran on a server with 4 8-cores Intel Xeon E5-4610, 128 GB of RAM, hyper-threading enabled.

The evaluation was conducted by varying the following parameters:

- the *rate of requests* to the servers, which in some cases increase the volume of logs generated (Apache function, which records every access), from 1 to 1000 requests/s;
- the *period of collection*, which affects the latency to access the context and in some cases the volume of traffic generated over the network (MetricBeat and BpfBeat, because they report the current status at each request), from 1 to 20 seconds.

Requests to Apache and MySQL servers were generated to increase their workload; requests to the mini_httpd server were DoS attacks to check the correct reporting of measurements from the eBPF filter.

Fig. 3 shows the cumulative CPU usage by all components of our framework. The overhead is rather limited (below 10% of the available CPU) in all conditions but for the largest number of requests for Apache. This is due to the large volume of logs to be processed and transferred. In particular, Logstash is adding a timestamp to each log records in our setup, and this implies more processing in case of larger workload. A similar consideration holds for the overall delay in collecting the data at the CB. Fig. 4 shows the latency to gather data from the file by the beat (this step is only present for Apache), to move data from the beat to the local Logstash instance, and to transfer data to the CB. The latency is generally shorter than a few seconds, but it increases a lot under heavy workload (i.e., 1000 requests/s) for Apache. This suggests that some kind of optimization might be possible for Apache/FileBeat, maybe aggregating some records, targeting a maximum latency below a few seconds. However, the impact and possible conflicts of resource usage (CPU, RAM, disk) from all processes should be also taken into account. In any case, the effective need for such improvements will be evaluated according to specific real-time constraints of the detection algorithms, which will be considered as next step of our work.

Overall, the results shown in Fig. 3 and 4 demonstrate the potential overhead in collecting large amount of information, and the need for dynamic adaptation procedures that are able to sample the data or perform other data fusion operations.

V. Related work

Some basic concepts about remote detection, context sharing, and agent-based cyber-security frameworks were already explored in the past [6]–[8]. In many cases, legacy security appliances are used and their output is correlated; the duplication at the local and global level introduces redundancy and also communication overhead.

More recently, some works on countermeasures against cyber attacks in distributed systems are surveyed in [9]. They present a high degree of scalability, in terms of computational efforts, because of the architectures that distribute the computational tasks to the peripheral part of the network (i.e., the nodes), instead of concentrating them in a single central point. The weak point is that the works do not consider dynamic changes in the network parameters, so the countermeasures cannot dynamically change. Furthermore, the amount of information on the security state of the system, which strongly grows with the size of the system, makes the proposed solutions not suitable in real distributed environments.

VI. Conclusion

In this paper we have described the overall architecture for a novel cyber-security framework that fulfill latest requirements on visibility, efficiency, and effectiveness for emerging computing paradigms like NFV. We have also implemented and evaluated the monitoring and inspection part of the data plane, which extends existing framework with more programmability.

Our work demonstrates that security data and events can be collected in a pervasive and capillary way with limited overhead, which can be dynamically adjusted to the actual needs of detection algorithms. Future steps will be the development of a Security Controller and some algorithms for analytics. For what concerns the Security Controller, the ambition is to automate control of the framework, by dynamically running algorithms, changing the frequency and verbosity of collected data, and triggering notification based on high-level policies. For what concerns analytics, the objective is to improve the detection capability through better correlation in time and space dimensions, leveraging machine learning and other techniques of artificial intelligence.

Acknowledgment

This work was supported in part by the European Commission, under Grant Agreement no. 786922 and 833456.

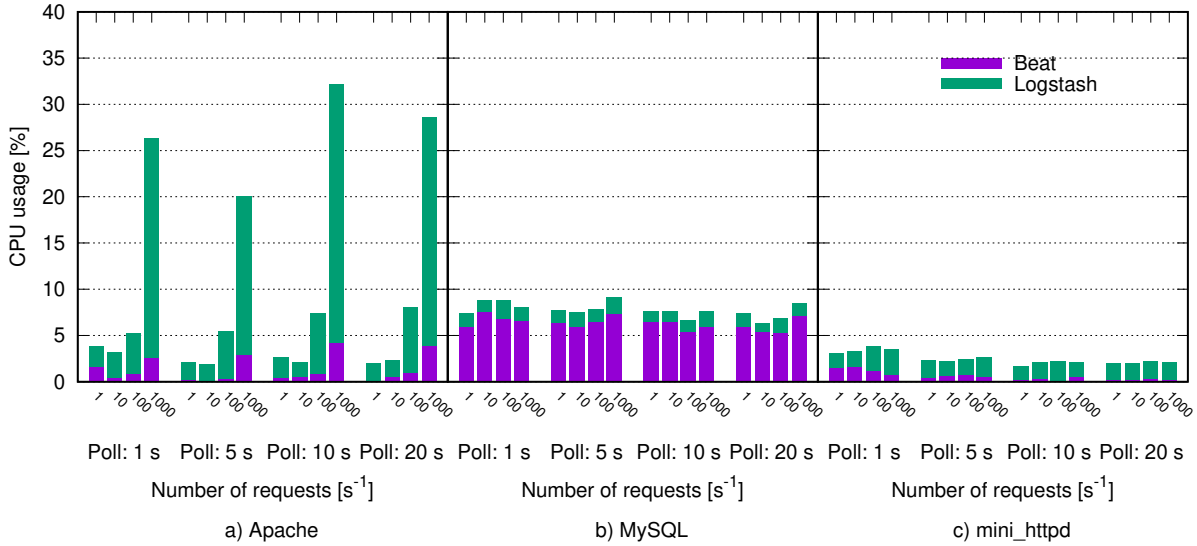

Fig. 3: Cumulative CPU usage by *beats* and Logstash in each VNF.

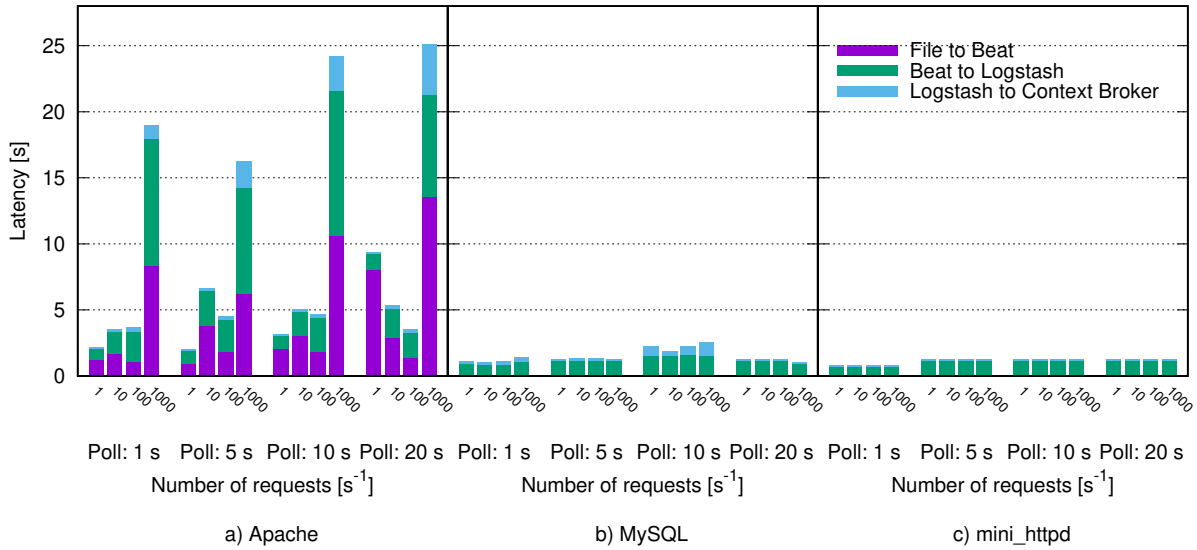

Fig. 4: Latency breakdown from data source to the CB. The "File to Beat" delay is only present when data are collected from a file (i.e., Apache VNF).

REFERENCES

[1] P. Bellavista, L. Foschini, R. Venanzi, and G. Carella, "Extensible orchestration of elastic IP multimedia subsystem as a service using Open Baton," in *5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, San Francisco, CA – USA, Apr., 6th-8th, 2017, pp. 88–95.

[2] 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.

[3] A. Carrega, M. Repetto, F. Risso, S. Covaci, A. Zafeiropoulos, A. Gi-annetsos, and O. Toscano, "Situational awareness in virtual networks: the astrid approach," in *IEEE 7th International Conference on Cloud Networking (CloudNet)*, Tokyo, Japan, Oct., 22nd–24th, 2018.

[4] 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. [Online]. Available: https://www.rfc-editor.org/rfc/pdfrfc/rfc8192.txt.pdf

[5] ETSI, "Network functions virtualisation (nfv); management and orchestration," ETSI GS NFV-MAN 001, December 2014, v1.1.1. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf

[6] F. J. Jon Oberheide, Evan Cooke, "Cloudav: N-version antivirus in the network cloud," in *Proceedings of the 17th conference on Security symposium (SS'08)*, San Jose, CA – USA, Jul. 28th – Aug. 1st, 2008, pp. 91–106.

[7] A. V. Dastjerdi, K. A. Bakar, and S. G. Hassan Tabatabaei, "Distributed intrusion detection in clouds using mobile agents," in *Third International Conference on Advanced Engineering Computing and Applications in Sciences*, Sliema, Malta, Oct. 11th–16th, 2009, pp. 175–180.

[8] S. T. Zargar, H. Takabi, and J. B. Joshi, "DCDIDP: A distributed, collaborative, and data-driven intrusion detection and prevention framework for cloud computing environments," in *7th International Conference on*

Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), Pittsburgh, PA – USA, Oct. 15th–18th 2011, pp. 332–341.

[9] P. Nespoli, D. Papamartzivanos, F. G. Marmol, and G. Kambourakis, "Optimal Countermeasures Selection Against Cyber Attacks: A Comprehensive Survey on Reaction Frameworks," *IEEE Communications Surveys Tutorials*, vol. 20, no. 2, pp. 1361–1396, Secondquarter 2018.