Noname manuscript No.
(will be inserted by the editor)

An autonomous cybersecurity framework for next-generation digital service chains

Matteo Repetto · Domenico Striccoli · Giuseppe Piro · Alessandro Carrega · Gennaro Boggia · Raffaele Bolla

Received: date / Accepted: date

Abstract Today, the digital economy is pushing new business models, based on the creation of value chains for data processing, through the interconnection of processes, products, services, software, and things across different domains and organizations. Despite the growing availability of communication infrastructures, computing paradigms, and software architectures that already effectively support the implementation of distributed multi-domain value chains, a comprehensive architecture is still missing that effectively fulfills all related security issues: mutual trustworthiness of entities in partially unknown topologies, identification and mitigation of advanced multi-vector threats, identity management and access control, management and propagation of sensitive data. In order to fill this gap, this work proposes a new methodological approach to design and implement heterogeneous security services for distributed systems that combine together digital resources and components from multiple domains. The framework is designed to support both existing and new security services, and focuses on three novel aspects: i) full automation of the processes that manage the whole system, i.e., threat detection, collection of information and reaction to attacks and system anomalies; ii) dynamic adap-

M. Repetto
IMATI - CNR, Genoa, Italy
E-mail: matteo.repetto@ge.imati.cnr.it

D. Striccoli, G. Piro and G. Boggia
Dept. of Electrical and Information Engineering (DEI), Politecnico di Bari, Bari (Italy) and
CNIT, Consorzio Nazionale Interuniversitario per le Telecomunicazioni
E-mail: {domenico.striccoli, giuseppe.piro, gennaro.boggia}@poliba.it

A. Carrega
S2N Lab, CNIT, Genoa, Italy
E-mail: alessandro.carrega@cnit.it

R. Bolla
DITEN, University of Genoa, Genoa, Italy and
CNIT, Consorzio Nazionale Interuniversitario per le Telecomunicazioni
E-mail: raffaele.bolla@unige.it

tation of operations and security tasks to newest attack patterns, and iii) real-time adjustment of the level of detail of inspection and monitoring processes. The overall architecture as well as the functions and relationships of its logical components are described in detail, presenting also a concrete use case as an example of application of the proposed framework.

Keywords Cybersecurity Framework · Digital Service Chains · Threat Identification · Identity Management · Access Control

1 Introduction

The most remunerative business in the digital economy will be the creation of value chains for processing data, through the interconnection of processes, products, services, software, and things from multiple vendors on a growing scale. Fully-automated software and environments will evolve and morph during run-time, without the explicit control of software engineers [1].

The uptake of cloud services and IoT has raised the interest in combining together digital resources and components from multiple domains and locations, to create Cyber-Physical Systems (CPSs). This evolution is already supported by pervasive and capillary communication infrastructures, computing models, and software architectures. Unfortunately, security paradigms have not evolved at the same pace. As a matter of fact, the prevalent model today is still the security perimeter, which is applied to individual domains with loose or no at all integration. This raises very important security questions, concerning the overall behavior of the system (attestation and availability), the location of personal and sensitive data (sovereignty), the protection of software and valuable information (integrity), and, most of all, the ability to perform quick remediation and mitigation actions in case of new and ever more sophisticated attacks [2, 3].

Even if cybersecurity appliances are constantly increasing their detection capabilities, they are usually deployed in vertical silos within each different administrative domain (e.g., cloud infrastructure, IoT device, enterprise, software repository). The lack of standard interfaces and common protocols hinders seamless composition of discrete cybersecurity appliances together [2, 4]. Indeed, today, cyber defense technologies, systems and applications often use proprietary software and commands to control system configurations. Most environments within a company or enterprise are comprised of hundreds of different types of cyber-defense devices.

Furthermore, the heterogeneity of ICT installations are progressively increasing the attack surface, fostering the raise of new attack models that join the more classical strategies like Distributed Denial of Service (DDoS) and botnets [5, 6]. Also identity management and access control strategies need attention: even if they have already been largely developed and integrated into distributed systems, they can neither guarantee the integrity and dependability of the whole chain over time, nor tracking the propagation of private data and sensitive information along the service chain [7–20]. Finally, the chain topology

and composition are usually unknown to the end user, who cannot easily check whether service owners, security mechanisms (e.g., encryption, integrity), and confidentiality policies are compliant with his/her own requirements and expectations. This scenario definitely helps attackers, which leverage the scarce visibility over the different subsystems and the lack of suitable integrated processes which are able to correlate events and measurements originated from multiple domains.

To overcome the issues described above, this paper proposes a new paradigm for managing cybersecurity in next-generation digital service chains. The proposed approach starts from the consideration that security functionalities must be embedded in every digital resource (e.g., cloud functions, networking services, databases, IoT), which give access to events and measurements for specific tenants. Based on this assumption, a novel methodology is described to dynamically discover security properties and features embedded in each digital component, connect them to a broad set of detection and risk assessment algorithms, and automatically trigger mitigation and response actions by user-defined policies, removing the need for legacy cybersecurity appliances, and providing better support for deep and effective analysis of the security context and more automation in the overall process.

The scope of this work does not cover the definition of new analytic toolkits or detection algorithms. Rather, it focuses on a methodology that is able to collect security-related events and measurements from dynamic and evolving ICT systems and infrastructures (including cloud and IoT services) in a programmatic way, and feed multiple state-of-the-art tools for detection of known attacks and investigation of new threats. Therefore, the novelty of this proposal lies in a new approach that supports two innovative key-aspects: *composability* and *programmability*. Composability is the capability to dynamically compose security processing chains at run-time which discover available agents and feed a rich set of detection and analytics engines with minimal or no manual intervention at all. Programmability can be conceived as the ultimate form of "flexibility," which creates tailored monitoring and inspection tasks in third-party infrastructures and services. These features will be explained by describing in detail the different macro-blocks of the framework (i.e., Local Agents, Security Manager, Identity Management and User Interface). Each macro-block is analyzed through the set of modules and components with their related functionalities. Specifically, Local Agents are in charge of collecting and analyzing data (service descriptors, events, data, and logs). The Security Manager shares the security context among multiple detection and identification algorithms, according to defined user policies. The Identity Management and Access Control block limits the access to the security context only to authorized roles, modules and algorithms. Finally, the user Interface is the main management tool used to build situational awareness and to perform reaction and investigation actions. A concrete application example in the automotive domain is proposed, describing the scenario, the workflow between the security architecture and the remote services, and the interactions among the main modules of the framework. Finally, the most relevant limitations and open challenges of

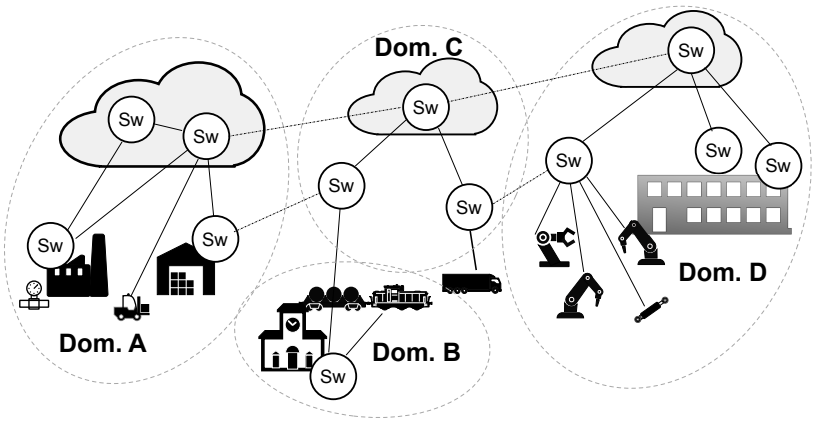

Fig. 1 An industrial supply chain creates, processes, shares, and distributes data among multiple actors and ICT infrastructures.

the proposed solution are addressed. They depend on the variegated security policies of external service providers and the standardization activities, that complicate the integration and harmonization of framework-related security procedures for the service chain.

The rest of the paper is organized as follows. Section 2 describes the reference scenario and the most important challenging requirements the proposed architecture aims to fulfill. Section 3 briefly summarizes the most relevant literature on security frameworks and access control. Section 4 presents and describes in detail the architecture with all its main components. Section 5 provides an application example of the proposed architecture. Section 6 describes the limitations and open challenges deriving from the adoption of the proposed framework. Finally, Section 7 summarizes the main findings of this work.

2 Reference scenario, related requirements and challenges

Today most business processes follow a fully-digital workflow, including design, implementation, creation, purchase, production, trading, delivery, and after-sales services, which extends across multiple domains, chains several processes, software and devices, and feeds them with relevant users data and context, as shown in Fig. 1.

Convergence among existing software paradigms, such as cloud computing, Software Defined Networking (SDN), and the Internet of Things (IoT) is expected to this purpose, leveraging automaticity and dynamic composition through service-oriented and everything-as-a-service models applied to CPSs. This represents a (r)evolution in the way of conceiving, designing, developing and operating systems, which pushes the adoption of service-centric

models, software and data sharing and multi-tenancy. The main challenges are described in the following sections.

2.1 The shift towards 'as-a-service' models

The ever-growing complexity and scale of information and communication technologies often represent a barrier for small businesses, which traditionally bring innovations and tailored solutions to the market. To overcome this hindrance, evolving business models are increasingly implementing the 'as-a-service' model as an effective and efficient alternative to full ownership of digital resources. The underpinning concept is represented by the possibility to virtualize and share devices, infrastructures, processes, and applications between multiple tenants. Such resources become accessible through software APIs, without the need for deep knowledge about their internal operation. APIs are effecively used to create even complex *service meshes*. Although this elementary definition has already generated a huge number of commercial offerings, the most common cases are Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), Software-as-a-Service (SaaS), Network-as-a-Service (NaaS), Data-as-a-Service (DaaS). In addition, Internet of Things-as-a-Service (IoTaaS), or just Things-as-a-Service (TaaS), is one of the latest iterations in the 'as-a-service' jungle, and there is not a shared understanding about this concept in technology or business jargon.

This evolution is also reflected in new business roles and relationships. As a matter of fact, Resource Providers (RPs) own valuable digital assets and make them available in non-exclusive yet segmented ways to Service Providers (SPs). The latter dynamically compose software, infrastructure and data into new value chains and business propositions for End Users EU. Examples of RPs include: Infrastructure Providers, which own physical resources and infrastructures (data centres, metropolitan and geographical networks, IoT installations, etc); Software Providers, which develop software functions and make them available in public or private repositories (e.g., github); Cloud Providers, which combine computing and storage infrastructures into virtualized services according to IaaS, PaaS, or FaaS models (i.e., they provide bare VMs, storage services, lambda functions); Network Operators, which implement large-scale communication services for public and private users (mobile networks, VPNs, NFV); Function Provider, which implement specific logical functions (e.g., authentication, databases, context brokers), and so on. Service Providers implement value-added services, for example, by selecting some software, deploying it in the cloud, connecting to IoT devices or data brokering services, reading data from data bases, connecting to external authentication services, securing networking with remote peers, etc. In some domains (i.e., cloud and NFV), they can largely automate most deployment and management processes through software orchestration tools, which help them to provision digital resources, to configure them, and to manage life-cycle events.

2.2 Multi-tenancy and virtualization issues

Even if service meshes bring more agility in service deployment and operation, the tighter integration among diverse business roles and the need to share infrastructures and data undoubtedly result in security and privacy concerns that have not been addressed in a satisfactory way yet [21]. As a matter of fact, multi-tenancy and virtualization create interdependencies between different tenants and between SPs and their RPs. If the target of the attack is a virtualized resource like, for example, a Virtual Machine (VM) or a Virtual Network Function (VNF), the impact on other tenants that share the same physical infrastructure can be limited by proper isolation (at the processor, memory, storage, and network level), provided that the overcommitment ratio is not very large. However, an attack against the physical infrastructure, for example a (D)DoS against the network of a cloud provider, will likely affect all tenants, even if this will not generate additional traffic within their virtual networks.

Even though many commercial tools are already available for cloud security, they are mainly meant for cloud providers because they only protect the infrastructure. Inspection of the tenants resources is limited due to privacy concerns and the usage of encryption. Cloud providers often provide security functions to their users (e.g., firewalling, Intrusion Detection System (IDS), Intrusion Prevention System (IPS), antivirus, etc.); however, the large heterogeneity of services and their interfaces hinders the implementation of uniform security policies for service chains that spread over multiple infrastructures and domains.

Affinity and anti-affinity policies are usually adopted by service providers, that take decisions whether different virtualized functional blocks of the service chain should be bounded to the same physical resource (affinity policy), or to different physical resources (anti-affinity policy) [22, 23]. Affinity is usually used for performance reasons, because it reduces the distance (hence the communication delay) between related logical functions; anti-affinity can be used for resilience and high-availability, assuming that different servers, networks, and infrastructures will unlikely fail simultaneously. From the generic perspective of security, affinity policies reduce the attack surface, since there is no communication link exposed to network attacks. However, a successful attack against a server or hypervisor will affect all service components clustered by the affinity policy. For what concerns the detection, attacks against a service instance will likely impact other service instances which fall under the same affinity group. So the knowledge of affinity policies could be used as an early indicator, to avoid the propagation of attacks among multiple services. Unfortunately, there is not yet a common way to easily and timely propagate this information from cloud providers or individual tenants to all other entities.

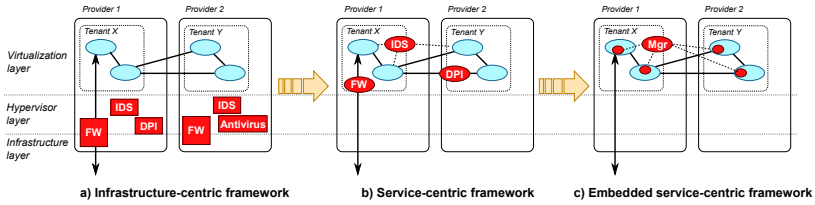

Fig. 2 The on-going evolution from infrastructure-centric to service-centric cybersecurity architectures.

2.3 From infrastructure- to service-centric models

From a purely architectural perspective, most cybersecurity appliances have been traditionally designed to protect the physical infrastructure, not the services implemented on top of it, as depicted in Fig. 2.a. The progressive dichotomization between the software and the underlying hardware brought by the adoption of virtualization and cloud paradigms has boosted a transition from infrastructure-centric to service-centric architectures (see Fig. 2.b). This model is largely used today, by deploying dumb probes in VMs and VNFs that collect events, logs and packets and send them for analysis to virtual instances of security appliances "plugged" into service graphs. Each tenant retains full control and responsibility of security management for its own graphs, without the need to rely on (and trust) external services. The application of this model is rather straightforward, and can be easily integrated with software orchestration techniques. However, additional resources are required to run the security appliances. In addition, the visibility is often limited to a few components, and does not allow to correlate events from the whole chain.

Chasing more efficiency, the next evolutionary step is a service-centric architecture that removes the need for legacy security appliances, embeds security capabilities into each software element, and orchestrates them by a common security manager that (logically) centralizes all security services, as depicted in Fig. 2.c.

A distributed cybersecurity framework removes the need for multiple and pervasive standalone and independent applications, with the ambitious goal of screening the whole system while correlating events in space and time. It aims at moving the detection of attacks and vulnerabilities from end terminals to common security centers (either hosted in the cloud or in specialized hardware). Differently from current practice in Security Operation Centres (SOCs), the ambition is to run most security services in a common centralized location, sharing the security context collected by smart local agents. The goal is to offload monitoring and inspection tasks to such agents dynamically, without the need for static detection and analysis appliances. The expected benefit is more dynamicity and adaptability of the whole framework to evolving threats, and fewer local resources to run it.

This new approach brings also new challenges that need to be effectively tackled. The first is a timely and efficient transmission of information over

the network, which should be protected against attacks and should not overwhelm the underlying communication channels. The second is how to perform lightweight operations on end terminals, which are heterogeneous and usually resource-constrained. The third challenge is how to share efficiently raw monitoring and inspection information among multiple detection algorithms, which should be able to analyze and correlate a large amount of data from different sources, and without neglecting identity management and access control techniques to manage sensitive data and provide authentication and authorization to the system components that exchange data and control commands.

2.4 Challenges and benefits for distributed cybersecurity frameworks

The availability of software-defined infrastructures allows unprecedented degree of agility in creating, changing, and destroying even complex service topologies, but the high dynamicity of these environments becomes a challenge for resource allocation. As a matter of fact, the locations and number of services can change dynamically, depending on the number of tenants, physical infrastructures and functions involved in the chain. Nevertheless, as the number of instances of a function and the number of functions involved in a chain grow, also the amount of resources increases accordingly. They can be identified as both hardware resources needed to run the functions (CPU, memory, storage) and network resources needed for data routing among instances and/or functions (bandwidth, link capacity, throughput).

The deployment of additional security functions has an impact on resource allocation. In general, a distributed cyber-security framework needs coordination in selecting, instantiating and placing security functions, and in delivering the set of collected data, measurements, and events among them. This should guarantee consistency with the detection needs, as well as respect the overall resource constraints and allocation policies for the service [24, 25].

One of the main problems of existing IDS/IPS appliances is the need to analyze network packet traces. This is feasible when the security appliance runs on the same host to protect, but becomes a problem when it runs remotely. The problem mainly arises from the usage of dumb local probes, which cannot extract different sets of features needed by the security appliance to detect specific attacks, because such features usually change in time. In this respect, the adoption of programmable technologies for network probing will largely overcome this issue. In addition, this approach will allow real-time adjustment of the level of detail of inspection and monitoring processes; the challenge in this case translates into finding the best trade-off between the level of granularity of data collected and exchanged and the overhead in resource allocation.

The presence of affinity policies could be used to improve the overall efficiency. When two or more service instances are clustered together, they would likely detect the same external context (e.g., network traffic, CPU and memory latency), so some detection tasks could only be run on a single instance.

2.5 Integration with management and orchestration tools

Beyond the collection of data and measurements for detection and analytics, fast and effective response and mitigation actions are very challenging issues for every distributed system. The efficiency of the reaction today largely relies on the ability of humans to identify the problem and implement countermeasures. However, new cyber-security frameworks are largely expected to leverage software orchestration tools for triggering faster and consistent response. For instance, in the Network Function Virtualization (NFV) world, integration with the management and orchestration entity [26] allows to replace a compromised VNF, to isolate a segment under attack and steer traffic across scrubbing centers or cloud-based services.

Whether the security framework should be integrated in the service orchestrator or left aside is still an open question. Probably it will mostly depend on commercial and business strategies. Indeed, the skills required to operate a SOC are very different from those required to manage an NFV or cloud service. Larger enterprise will probably benefit from integrated solutions, whereas small business will likely rely on externalization of security services. Decoupling security operation from service management will clearly lead to the need for authentication and access control mechanisms, to avoid introducing additional threats in the system.

3 State of the art on cyber-security frameworks

The need to tackle the challenging requirements addressed in Section 2 is testified by the numerous works present in literature, the most representative of which are briefly discussed in this section. Security aspects in distributed, multi-domain and multi-tenancy systems are tackled in [27–35].

The state-of-the-art in distributed cyber-security systems is surveyed in [27–31]. The survey [27] reviews the literature on distributed filtering and control strategies through dynamic models in industrial CPSs scenarios. Data collection strategies through data collectors, or agents, for distributed IDS are surveyed in [28]. The paper [29] analyzes security and privacy issues in distributed IoT architectures, addressing security and privacy-related features and challenges, in terms of data collection, aggregation, mining and analytics, at different layers. Some works on countermeasures against cyber attacks in distributed systems are surveyed in [30], with particular focus on scalability and computational effort issues. Theoretical models are adopted to make decisions on the countermeasures to be taken. Prediction methodologies for the evolution of the attacks in distributed systems are analyzed in [31]; they are based on threat correlation, sequences of actions, statistical models, and extraction of attack features.

Machine Learning (ML) algorithms are proposed in [33] for intrusion detection. They are mainly based on neural networks or deep learning architectures to extract relevant information from large volumes of data [33] .

There are also some works analyzing security frameworks in specific scenarios [34,35]. In [34] a strategy is proposed to evaluate the trustworthiness of messages exchanged among distributed vehicles in a secure Vehicular Ad-hoc NETwork (VANET) environment. A risk assessment framework is proposed in [35] for industrial systems, where attacks are predicted based on specific propagation models to derive the probability of compromised nodes in the network.

In distributed cybersecurity frameworks identity management and access control capabilities are of great importance, to verify the authenticity of any physical and logical entity belonging to the whole architecture and verify the authorization to accessing to heterogeneous resources and services distributed and deployed across different organizations. The most representative works on this topic can be found in [7–11, 17–20].

The scientific literature often demonstrated that authentication and authorization services have been always considered a big challenge in decentralized environments. Most of the emerging solutions exploited a decoupled mechanism, which aims at separating authentication and authorization functionalities in a harmonized fashion [7]. Many interesting solutions have been recently formulated in the scientific literature for identity management in multi-domain environments, like OpenID Connect and OAuth 2.0 [8–10]. They introduce the possibility to authenticate users within a federated ecosystem by means of a trusted Identity Provider.

A concrete solution offering fine-grained authorization, namely Attribute-Based Access Control (ABAC), has been formulated by the National Institute of Standards and Technology (NIST) [12]. In this proposal, the access to resources is handled by considering attributes associated to the user identity, and access to resources and services is granted to the user after a proof of possession of attributes that satisfy the access control policy. Other approaches for protection of resources are Identity-Based Access Control (IBAC) and Role-Based Access Control (RBAC) mechanisms [11]. In IBAC, a user is authorized to access to a resource or service if its identity appears within a dedicate Access Control List. In RBAC, access rights are based on roles and privileges of the users. Starting from the ABAC logic, other approaches aim to solve the access control problem through cryptographic mechanisms [17–20].

4 The proposed architecture

The goal of this work is to present an architecture that aims at overcoming the limitations of the current literature on security frameworks and distributed platforms.

The proposed architecture is logically composed by four different macro-blocks. Each macro-block is characterized by a set of components that implement functionalities peculiar of the macro-block they belong to. The first macro-block is implemented locally in the *Local Agents* of digital services. It is composed by all that parts that add more security capabilities to the

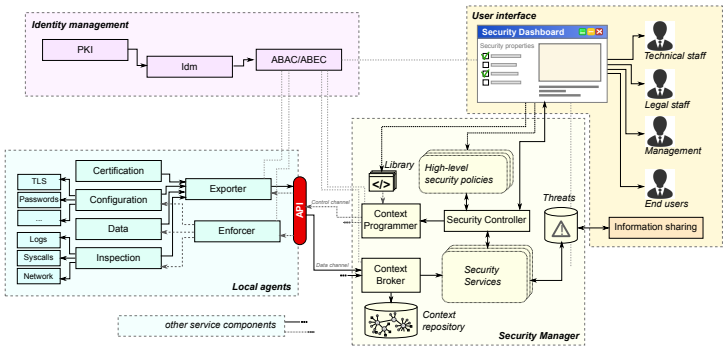

Fig. 3 Reference architecture for next-generation cyber-security frameworks for digital value chains.

local services for monitoring, inspection and enforcement purposes. The second macro-block is the *Security Manager*. It constitutes the centralized part of the framework and includes all the components that collect and process data from local services, implementing mitigation and reaction strategies. The third macro-block is the *Identity Management*. It permeates the large majority of the components of the framework, being present both in the local and centralized parts of it. The Identity Management is mainly responsible for the coordination of digital identities and access policies, and performs identity protection and access control functionalities. The fourth macro-block is the *User Interface*. It is also implemented locally, and regulates the human-machine interaction for a tailored presentation of analytics to different kinds of users, and the definition of control and management policies to react to security issues.

Overall, this reference architecture follows the typical structure of Security Information and Event Management (SIEM) systems. However, some relevant extensions are necessary, to effectively tackle the technical and procedural challenges brought by the dynamic composition of digital services. To this respect, interactions among the macro-blocks described above is performed through standardized security Application Programming Interfaces (APIs), that are exposed by any digital service, be it a (cloud) application, a virtualization infrastructure, a serverless function, an IoT device, etc. They are implemented at both the control and data planes. In the control plane, APIs deliver control and management data used to discover security capabilities and enable, disable and configure the security functions in the Local Agents. In the data plane, APIs allow the local security functions to report the collected set of events, data and measurements to the Security Manager for the application of advanced security services. All these components will be described in detail in the subsections that follow.

4.1 Local agents

Local agents are in charge of collecting service descriptors, events, data, and logs (collectively indicated as *security context*). The purpose is to expose some internal information of each service, so to allow the detection of multi-vector threats and to improve the trust in service operation. Multiple agents should be present to collectively cover at least the following scopes:

- *inspection*: collection of data, events, measurements from heterogeneous sources (application logs, system calls, network traffic) that can be used to detect attacks and identify new threats;
- *tracking data* belonging to users through metadata, with explicit identification of personal and sensitive information that may raise privacy issues;
- *configuration analysis*, to report incorrect, faulty, or weak settings as lack of encryption, weak or blank passwords, unnecessary network sockets in listen state, outdated or buggy software versions, etc.;
- *certification* of the origin and integrity of the software component, identity of the vendor/seller, etc.

An exporter function is responsible to authorize access by any remote party, according to the settings of the owner, as well as to configure the reporting behavior, e.g., by changing the frequency and/or verbosity of context information. An enforcer function applies enforcement policies: packet classification and filtering, removal of private and/or sensitive data, configuration changes. Enforcement will also cover data protection, by ensuring they are accessed, shared, and exported according to their owner policies in terms of data minimization, purpose limitation, integrity, and confidentiality.

Despite the large numbers of tools already available for monitoring and inspection, their usage in a multi-tenancy context is not straightforward. As a matter of fact, they should give visibility over local resources to external entities, so it is challenging to restrict the scope to a subset of resources in case of multi-tenancy. It is also important to ensure that only authorized entities have access to these components, to avoid making them an addition threat.

A very important requirement of the local agents is that they should be lightweight to not require additional resource allocation, with a small footprint on service execution. They have also to be efficient without increasing the attack surface. Security functions in local agents are controlled by a local management and control component, being responsible for managing the software of the functions, reporting information on their correct utilization, monitoring their internal structure, and generating report messages. It can also inspect traffic for security purposes, but anyway it provides descriptive information related to security functions.

The implementation of the security agents should be tailored to specific services, given the large heterogeneity of digital resources: applications, devices, functions, SaaS, or even more complex resources like a cloud infrastructure or a NFV framework. In this last case, two possible implementation scenarios can be realized. In the first scenario, NFV can be viewed as a digital service itself,

providing connectivity and networking functions on demand. Security agents can be used to monitor both VNFs and the virtualization infrastructure. Here a management and orchestration functional block (e.g., the NFV-MANO [26]) is needed to manage and orchestrate the VNFs, but only partially, since some security agents may be present in the infrastructure and therefore are not manageable by an orchestrator. In the second scenario instead, single VNFs can represent digital services that are orchestrated by NFV-MANO which, in this second scenario, can be used to automatically deploy and manage security agents within each VNF. These two examples give an idea of the different possibilities to implement local agents in a virtualized infrastructure, and how the NFV-MANO can be employed to manage and orchestrate the VNFs.

Remote collection of logs is already a well established practice, with many frameworks available for this purpose (Scribe, Flume, Heka, Logstash, Chukwa, fluentd, NSQ, and Kafka). From a research perspective, the real challenge is *programmability*, which is the capability of the framework to dynamically adapt operations to continuously evolving attack patterns, defining and updating monitoring, inspection, and enforcement tasks accordingly. It goes beyond plain configurability at run-time (e.g., to adjust the verbosity of logs, frequency of sampling, and other tuneable parameters), since programmability also includes the definition of new tasks, by injecting lightweight yet secure code on-the-fly, without the need for full or partial re-design of the whole system or some of its components. For example, it could enable tailored analysis of network packet bodies locally, without developing new full-fledged inspection modules. The target is more flexible operation than today, allowing lightweight processing for normal operation, while moving to deeper inspection (and larger overhead) at the early stage of any suspicious anomaly, or upon triggers from cyber-threat intelligence. Task offloading to local services helps balancing the trade-off between processing and network overhead in an effective way, tailoring the broad range of local capabilities to the specific nature of the digital service.

Luckily, the modern technologies selected for this task are not resource-hungry, so resource allocation is not a problem, like explained in a preliminary study on this topic [36]. At the same time, this kind of flexibility would allow more efficient allocation of resources, by dynamically adapting the processing load to the evolving context. Such approach is very useful whenever the detection is based on techniques (like ML, or Artificial Intelligence) which are largely based on the extraction and analysis of features that cannot be known in advance since attacks evolve and new threats emerge, thus effectively addressing the need to tackle the continuous evolution of attack patterns and to investigate or react to zero-day attacks. Indeed, in this second case, static configuration options might not be enough to detect or implement unexpected features in real-time. Summarizing, programmability is implemented in the control plane of each local agent, and develops on two main directions:

1. The operational parameters (log files, configurations, current status of the system, filtering events, etc.) are modified at run-time, according to predefined templates, patterns, and options.
2. Security programs can be on-boarded without re-designing, re-deploying, and even re-starting local agents. In this case, the same framework is also responsible for verifying authorization, integrity, and safety of any piece of code that is injected into remote objects.

Programming models should target lightweight tasks, to not overwhelm resource-constrained devices, and execution in safe sandboxes, to limit damages coming from compromised code. A promising technology to this purpose is the extended Berkeley Packet Filter (eBPF)[1], which currently provides inspection capabilities for both network packets and system calls.

4.2 Security manager

The Security manager is the most valuable and innovative component in the proposed architecture. It is responsible for collecting and sharing the security context among multiple detection and identification algorithms, according to the overall objectives and behavior described by high-level user policies. As shown in Fig. 3, multiple logical components are required to implement the Security manager.

4.2.1 Context Broker

The first task for the Context Broker is to manage the heterogeneity of sources and protocols, which is reflected in different data and control interfaces. The Context Broker hides this heterogeneity and exposes a common data model to the other components, for discovering, configuring, and accessing the security context available from the execution environment (namely, the different digital resources).

The Context Broker has also capabilities of data abstraction, fusion, and querying. The flexibility in programming the execution environment is expected to potentially lead to a large heterogeneity in the kind and verbosity of data collected. For example, some virtual functions may report detailed packet statistics (i.e., those at the external boundary of the service), whereas other functions might only report application logs. In addition, the frequency and granularity of reporting may differ for each service. The definition of a security context model is therefore necessary for security services to know what could be retrieved (i.e., capabilities) and what is currently available, how often, with what granularity (i.e., configuration).

Data aggregation and fusion capabilities will help distill refined information from the large set of events and data collected by the local agents. A common

[1] The extended Berkeley Packet Filter is a low-level Linux socket interface that give access to raw network packets and system calls. It allows small assembly-like programs to be downloaded and run in a controlled virtual machine.

abstraction should be used to expose such capabilities in a consistent way, by organizing and aggregating data coming from local agents into features. A feature identifies what kind of data have to be extracted from the whole dataset that can be generated by local agents; it is kind of data "subsampling". Possible examples of data representing features are: sections of logs, specific fields of network packets, performance metrics, Operating System indicators, events from applications, protocols, traffic statistics, etc. The choice of the extracted feature is related to the threat under analysis, and is a critical issue for the correct identification of current and future threats, but it is helpful for two different reasons. First, resources are saved locally, according to the programmability requirement, because features are a usually small subset of all data that local agents can provide. Second, the feature is the same whatever the number and type of agents and the service implemented, so, whatever the agents/services added on-the-fly, the detection and analysis procedures are not modified.

The correct identification of the most appropriate features is very challenging, because it depends on the service topology, the agents mapped on it, the type of attack to be detected, and how to carry out the attack detection. The better the suitability of the feature extracted, the more effective the security service in its detection and analysis operations (security services will be described in detail in Section 4.2.3).

Correlation of data in the time and space dimensions will naturally lead to concurrent requests of the same kind of information for different time instants and functions. In this respect, searching, exploring and analyzing data in graph databases should be considered as implementation requirements. Indeed, unlike tabular databases, graph databases support fast traversal and improve look up performance and data fusion capabilities. Finally, the last implementation requirement is the ability to perform quick look-ups and queries, also including some forms of data fusion. That would allow clients to define the structure of the data required, and exactly the same structure of the data is returned from the server, therefore preventing excessively large amounts of data from being returned. This aspect could be very useful during investigation, when the ability to understand the evolving situation and to identify the attack requires to retrieve and correlate data beyond typical query patterns.

Another feature of the Context Broker is data storage. Given the very different semantics of the context data, the obvious choice is non-relation databases (NoSQL). This allows to define different records for different sources, but also poses the challenge to identify a limited set of formats, otherwise part of the data might not be usable by some security services. The validity and volume of data affect the size of the database and the need for scalability. Local installations are suitable when data are kept for days or months, but cloud storage services may be necessary for longer persistence or larger systems. On the other hand, remote cloud storage is not suitable for real-time or even batch analysis. Another design issue is the possibility to scale-out horizontally and/or inborn support for parallel processing and big data analytics, if the data volume becomes large.

4.2.2 Context Programmer

The first task of the Context Programmer is to manage the programmability of local agents, which, as detailed in Section 4.1, is the capability to shape the depth of inspection according to the current need, in both spatial and temporal dimensions, so to effectively balance granularity of information with overhead. This is a novelty with respect to the existing frameworks, which have an intrinsic rigidity in analysis and detection procedures often based, and dependent, on data sets generated by each agent.

Programmability also includes the capability to offload lightweight aggregation and processing tasks to each local environment, hence reducing bandwidth requirements and latency. This would change the reporting behavior by tuning parameters that are characteristic of each app (logs, events), network traffic, system calls (e.g., disk read/write, memory allocation/deallocation), remote procedure calls toward remote applications (e.g., remote databases), etc. The Context Programmer is the logical element that offers a homogeneous control interface for configuring and programming different data sources, by implementing the specific protocols (control channel). The Context Programmer has also a context discovery layer. Context discovery should manage an evolving topology by discovering new components that join or leave the service and that cannot be deployed and managed freely, since the related resources belong to SPs that are very often external to the framework. Since different actors are usually involved in the same service chain, access to the context is subject to identity management and access control. By selectively querying all components involved in the chain, this layer builds the logical topology of the overall service, including the security properties and capabilities of each node.

The Context Programmer can also enable pushing pre-defined programs from a programs library. The programs library is a collection of software that can be injected into the programmable hooks present in the execution environment. Different languages can be used by different hooks, e.g., ELF binaries, java bytecode, python scripts, or P4/eBPF programs. Such programs are written and compiled offline, and then inserted in the library by the Security Dashboard. They also include metadata for identification and description, so to be easily referred by the Security Controller.

From a security perspective, it is important to formally verify the programs safety and trustworthiness. This is implicitly guaranteed whenever the code is executed within an execution sandbox. But in case of general-purpose languages, the correctness and safety of the source code might be verified by static tools for source-code analysis.

4.2.3 Security Services

One of the main advantages of collecting heterogeneous security information in a centralized repository is the possibility to carry out analysis and correlation well beyond the typical limited scope of existing security functions (Denial of Service (DoS) detection, IPS, IDS, antivirus, etc.), and in a far more efficient

way, i.e. without replicating monitoring and inspection operations. This is the main task of the security services, that process data, exploiting possible correlations between apparently independent events which may come from the same multi-vector attack. Their main features are both detection and assessment, based on specific security policies that can allow or deny a service, depending on policy-dependent requisites. They are also conceived for log analysis; for example, depending on the monitored activities reported by logs coming from different digital services, they can detect traffic anomalies and signal them as suspicious activity. Security services are placed in the centralized part of the framework; they compare data coming from the Context Broker with predefined security and control policies, and take automatic actions accordingly. In turn, the Context Broker exposes to them a common security context, abstracted from data coming from heterogenous sources and protocols, and with different data and control interfaces.

Security services should run dynamically, eventually being combined together to carry out more complex analysis and assessment tasks. The security service components must be created ad-hoc, so that they are well-defined and with compatible APIs. The ambitious goal is to guarantee full interaction among them, through common and standardized API semantics. Accordingly, an entity responsible for managing and orchestrating the execution of security services is needed in the Security Manager. This management entity is also responsible for the right choice of the applications based on their interface compatibility, so that the exchange of data and control information is guaranteed.

Security services can also run in a virtualized environment, in containers like VMs, with a dynamic allocation of resources for scalability and optimization purposes, and without keeping a tight bound between the running software and the underlying hardware environment. So, there is virtually no limit to the number and types of security services that can be implemented: verification of trust properties, intrusion detection, DoS detection, remote attestation, etc. This is the same principle at the basis of the NFV architectural framework as described in [37], so it is not a novelty by itself. The real novelty resides in the application context of this architectural part of the framework, which is totally different from the NFV counterpart for two reasons. First, security services are not network functions and do not provide a network service. Second, the Context Broker abstracts control and information data at a high abstraction layer, that can be seen as transparent towards the underlying network layer at which packets are processed (please refer to Section 4.2.2 for details on this aspect).

Beyond the mere re-implementation of legacy appliances for performance and efficiency matters, the specific research challenge is how to implement a new set of security services aiming to detect anomalies and threats effectively and proactively. From this point of view, a possible but interesting approach can be the adoption of ML algorithms. As known, they have the capability to extract various patterns, which can be seen as sequences of subsampled data, that identify legitimate or malicious activities, based on the fact that

the behaviour of a traffic pattern in case of attack is different from that of a normal traffic pattern [32,38]. ML algorithms allow to learn the patterns that characterize a normal behaviour of the feature, so to recognize differences that can be identified as possible threats and attacks, and all this independently from the configuration of the local agents. This aspect is very useful in this scenario, since local agents are almost always implemented externally to the framework.

The features making part of the context can be used to train the ML algorithms that, in turn, will detect attacks and anomalies by discovering differences between the patterns learnt in case of normal traffic and patterns analyzed run-time. The strength of this approach is that ML algorithms are able to emulate the patterns behaviour without rigid and predefined rules, that instead are created in the training phase by the algorithms themselves. The main difficulty of this kind of approach is that new threats, or even variants of the existing ones, can affect features that are different from those chosen to detect traffic anomalies; so, as remarked in Section 4.2.2, it is of great importance to correctly choose the set of features to feed the ML algorithms and instruct the local agents. In this context, an analysis of the correlations among features can be of great help, since the relationships between different pattern behaviours helps improving the effectiveness of the detection process. In fact, if data extracted from different features are correlated, the behaviour of a feature influences the others, allowing ML algorithms to detect more effectively new threats as soon as they change the normal behaviour of a feature. Capturing correlations among features to feed ML algorithms is actually a challenge, given the wide variety of data coming from local agents that are manageable by ML algorithms and that can be used to build the context.

A broader classification of security services includes the features of *attack detection*, *threat identification*, *data tracking*, *trust and risk assessment*.

Attack detection - It is the capability to monitor the system behavior to recognize activity patterns that can be associated to known threats and attacks. Rule-based detection algorithms show their limits in the time to define new rules and to push updates to every installation. Similarly, the creation of legitimate profile usages is a complex and cumbersome task, which must be tailored to each different environment and use case. The challenge here lies in adding more intelligence to process the security context and to correlate even apparently uncorrelated heterogeneous events and data (network traffic, log files, user behavior) on different systems. This concept would add more flexibility to the detection process, freeing the algorithms from rigid and predefined rules and increasing their robustness in the detection of novel attacks, especially zero-day ones. Accordingly, the detection of such types of attacks is a peculiarity of this specific part of the framework. The effectiveness of such capability strongly depends on the choice and/or development of the specific algorithms to be run as security services. It is left to the implementation choices, and can be developed ah hoc for this purpose, or chosen among all the algorithms that handle the detection of zero-day attacks. What is important to remark here is that this architectural solution allows to overcome the

heterogeneity of the external infrastructures, each one with its own capability of detecting zero-day attacks. In this respect, the previously cited ML methods, including (but not limited to) K-Nearest Neighbors, Naive Bayes, Graph Kernel and Support Vector Machine can help in this direction [39–41].

Threat identification - It aims at identifying anomalies and suspicious activities that deviate from the average system behavior, and tries to define new patterns for unknown attacks; all this, in an automated way. Although very detailed classifications and taxonomies of both attack and defense methodologies have been already identified, attacks continuously transform to circumvent detection rules in security appliances. Again, ML methods promise significant advances in this field, especially when combined with the multilevel correlation analysis among the attributes of correct and malicious data [42,43]. A possible approach in the adoption of ML algorithms is the so-called supervised learning. Here, the ML algorithm is trained to possible malicious patterns that deviate from normal traffic, so to be able to recognize each of them in the detection phase. Given the impracticability of elaborating detection rules for unknown threats, an alternative and ambitious approach is the unsupervised learning, whose goal is to autonomously identify anomalies, i.e., non-conforming patterns compared to the well-defined notion of normal behavior. This would also satisfy the automaticity requirement of the framework. The most critical point in such approach lies in the selection of the most suitable data set that is used to train the ML algorithm. This data set must be composed by traffic that is not affected by anomalies of whatever kind. After the training phase, the ML algorithm should be able to identify the unknown anomaly during the detection phase.

Data tracking - It represents the capability to follow the position and transfer of private and sensitive data along the business chain, check compliance with user's privacy policies, and alert or remove data in case of violations. Data privacy solutions for the cloud entails the introduction of specific middleware to control and manage access to data. This works when data are shared among a pre-defined set of applications that run in a homogeneous environment, but it is more challenging to achieve in heterogeneous, dynamic, and composite systems. The recent introduction of the General Data Protection Regulation (GDPR) in Europe has boosted an increasing interest in data privacy and sovereignty. The typical approach is limited to the procedural level, while technical enforcement solutions are still missing. The proposal in this direction lies in the adoption of security APIs in each digital service, that will enable to query about the presence and usage of private and sensitive data; in addition, any access to data should trigger a notification and the verification of user policies. In this way, beyond enforcement of data access, records will be kept about the transfer of data to other services, enabling later verification of persistence and request for removal. Here, the main challenge is the identification of new ways to trade data. Blockchain technologies might provide interesting solutions, since the problem is not far from Digital Right Management (DRM), which is already present in recent research roadmaps [44].

Trust and risk assessment - It represents the capability to assess the reliability of the different actors and the services involved in the business chain, by evaluating the appropriateness of security properties (presence of vendor/software certification, presence of private/sensitive data, configuration settings, etc.) to the user's policies, and by evaluating the risk related to security breaches. When heterogeneous services are automatically selected from different domains to be chained together, their security properties should be formally verified to satisfy the high-level trust policies (trusted vendors/countries, minimal encryption requirements, trust chains, security mechanisms, etc.) of users, that should always be aware of the weakness of a service, and able to decide whether it is acceptable or not. Trustworthiness will involve the two dimensions of identity (service owner/provider) and integrity (software). Assuming the lack of a common authentication framework worldwide, the challenge here is to build reputation models based on recursive trust relationships, similarly to what already used in e-mail systems (i.e., PGP).

From an architectural perspective, each security service will only be required to implement the interfaces towards the Context Broker and the Security Controller. For existing tools, this could be achieved by developing plug-ins or adapters. The interface to the Context Broker will be used to retrieve relevant information, including both real-time and historical data. This interface will allow selective queries to return aggregated data, with respect to multiple services and time periods. The interface to the Security Controller is used to notify security events like threats and attacks, that may trigger some forms of reaction. The description of the event may include an estimation of the accuracy of the detection, so to trigger the collection of more detailed information; alternatively, this information could be retrieved by evaluating specific conditions on the current security context.

The combined analysis of the security context can greatly enhance the detection capabilities, especially in case of large multi-vector attacks. The challenge is clearly to merge knowledge without exposing sensitive information to external domains. In this respect, the notion of local processing and distributed security analysis as hereby proposed may provide an effective solution for multi-layer detection mechanisms. The combination of heterogeneous monitoring data will open the opportunity for novel detection capabilities. For example, analysis of application logs that indicate multiple login failures may help detecting attack patterns in the encrypted network traffic. From a practical perspective, however, the real range of security services will be limited by the possibility to find an acceptable trade-off between the complexity to implement local inspection and the communication overhead.

4.2.4 Security Controller

The Security Controller represents the most valuable part of the architecture, conceived to automate as much as possible the behavior of the whole framework. It positions between the high-level policies and the context, and orchestrates security functionalities, according to what already devised in on-going

initiatives [45]. So, the role of the Security Controller is to mediate between network applications and the underlying data plane.

It can work in three alternative ways:

- *fully automated*: the framework reacts to specific conditions based on pre-defined rules, without any intervention from humans. This is only possible for well-known threats. For example, a packet filter may be installed when the traffic streams grow beyond a given threshold. Another example is the request to isolate or remove a service upon indication of intrusion.
- *semi-automated*: in case of unknown or complex attacks, pre-defined policies might not be able to cover all possible situations or variants, so the system may only partially respond automatically and wait for further inputs from humans. This may be the case of anomalous (yet not overwhelming) flows of packets that are temporarily blocked while waiting for additional actions from the security provider.
- *supervised*: the system is able to react autonomously, but the likelihood or impacts of possible errors suggests confirmation from humans. In the same example as the previous point, the security provider is asked the permission to block the traffic, so to avoid to disrupt any critical activity.

Automatic reaction shortens response times and unburden humans from mechanical and repetitive tasks. However, full awareness and the need for post-mortem analysis recommend to keep track and report any action to the dashboard, at least to give visibility of the occurrence of attacks.

We can give a concrete example of how the Security Controller is expected to behave in case of DoS attack. Detection of volumetric DoS is typically based on analytics on the network traffic. Since deep inspection of the traffic leads to high computational loads and latency, an initialization policy only requires statistics about the aggregate network traffic that enters the service, which may be collected by standard measurements reported by the kernel. The same policy also initializes an algorithm for network analytics and sets the alert thresholds. Upon detection of an anomaly in the traffic profile, an event is triggered and the Security Controller invokes the corresponding DoS policy. The policy now requires finer-grained statistics, and the Security Controller selects a packet filtering tool (e.g., eBPF) for packet classification, installs and configures it. The policy also requires the detection algorithms to work with the broader context information available. As soon as the analysis comes to a new detection, it triggers a new alert, this time including the relevant context (i.e., identification of suspicious flows, origins, etc.). Before taking the decision about how to react, the mitigation policy may evaluate some conditions to check if the suspicious flow comes from an expected user of the service, if it has been previously put in a blocklist or in an allowlist, and if it is acceptable based on previously recorded time series. The actions to be implemented (e.g., dropping all packets, dropping selected packets, redirecting suspicious flows towards external DoS mitigation hardware/software, stop the service, move part or the whole service to a different infrastructure) is therefore notified to the Security Controller, which again translates them in a set of commands

for the external service orchestrator and/or configurations and programs to be installed in the execution environment. Notifications about the detected attack and the implemented actions are also sent to the Security Dashboard.

4.2.5 High-level security policies

Policies define the behavior of the system. Conceptually, policies do not implement inspection, detection or enforcement tasks, so they do not correspond to any existing security function (IDS/IPS, antivirus, Virtual Private Networks). Instead, they represent an additional upper layer for control of security services. Policies are therefore used to automate the response to expected events, avoiding whenever possible repetitive, manual, and error-prone operations done by humans.

The simplest way to define behavioral policies is the Event-Condition-Action (ECA) pattern, which covers a broad range of interesting cases. The definition of an ECA policy requires at least 3 elements:

- an *Event* that defines when the policy is evaluated; the event may be triggered by the data plane (i.e., detection algorithms), the management plane (i.e., manual indications from the dashboard, notifications from the service orchestrator), or the control plane (i.e., a timer);
- a *Condition* that selects one among the possible execution paths; the condition typically considers context information as data source, date/time, user, past events, etc.;
- a list of *Actions* that respond, mitigate, or prevent attacks. Actions might not be limited to simple commands, but can implement complex logics, also including some form of processing on the run-time context (e.g., to derive firewall configuration for the running instance). They can be described by imperative languages, in the forms of scripts or programs.

The range of possible operations performed by policies include enforcement actions, but also re-configuration and re-programming of the monitoring/inspection components in the execution environment. Enforcement and mitigation actions are mostly expected when the attack and/or threat and their sources are clearly identified and can be fought. Instead, re-configuration is necessary when there are only generic indications, and more detailed analysis could be useful to better focus the response. A typical example is a volumetric DoS attack. To keep the processing and communication load minimal, the monitoring process may only compute rough network usage statistics every few minutes. This is enough to detect anomalies in the volume of traffic, but does not give precise indication about the source and identification of malicious flows to stop. Re-configuring the local probes to compute per-flow statistics or more sophisticated analysis helps to implement traffic scrubbing[2].

[2] Scrubbing is a technical term used to indicate a cleansing operation that analyses network packets and removes malicious traffic (DDoS, known vulnerabilities and exploits). It is usually implemented in dedicated devices or infrastructures, able to sustain high volumetric floods at the network and application layers, low and slow attacks, RFC Compliance checks, known vulnerabilities and zero day anomalies.

From a research perspective, the ambition is the definition of high-level policies in terms of objectives and intents, that could be defined even by non-technical users. The adoption of advanced reasoning models, even based on some forms of artificial intelligence, is clearly a very promising yet challenging target to automate the system behavior. This would open the opportunity for dynamically adapting the response to new threat vectors. In this respect, the historical analysis and correlation of the events and conditions with the effects of the corresponding actions from existing policies or humans would provide useful hints to assess the effectiveness of the latter, so to identify and improve the best control strategies.

4.3 Identity management and access control

The security context retrieved by the Context Broker contains a lot of information about service usage patterns, users, exchanged data, and so on. Access to this data should therefore be limited to authorized roles and algorithms. In addition, configuration of the remote agents must remain a prerogative of the security controller and trusted policies, so it is important to track the issuer of such commands. The Context Broker is therefore expected to enforce access policies settled by the Identity management module (Idm). In line with the reference scientific literature on this topic, identity management and access control can be flexibly managed through the ABAC logic.

4.3.1 Public key Infrastructure

The overall security architecture grounds its roots on a Public key Infrastructure (PKI), embracing a Local Certification Authority and a list of authentic users. From the cryptography perspective, Local Certification Authority and users are in possession of a public-private key pair. The private key is kept secret for all the entities. The public key, instead, can be shared within the whole architecture by means of a X.509v3 certificate, signed by the Local Certification Authority. In order to offer a good level of flexibility, the proposed architecture also envisages the possibility to integrate Local Agents and users belonging to heterogeneous domains/platforms (multi-domain approach). In that case, the Identity management block depicted in Fig. 3 can integrate more PKIs, each one managed by its own Local Certification Authority.

4.3.2 Idm component

The proposed architecture envisaged in this paper implements the decoupling between authentication and authorization functionalities. In this case, a key role is provided by the Idm component. Specifically, the Idm component contains a database that maps the identity of both Users, Local Agents, and any other component belonging to the Security Manager to a specific list of attributes. From one side, it is able to authenticate Users, Local Agents, and

any other component belonging to the Security Manager within the system. From the other side, it is able to provide them the right set of attributes that, according to the ABAC logic, will be used to protect resources or grant the access to protected resources during the authorization phase.

4.3.3 ABAC/ABE component

Authorization procedures and policy enforcement are managed through the Distributed Multi-Authority Ciphertext-Policy Attribute-Based Encryption (DMA-CP-ABE) algorithm, as suggested in [20]. After a successful authentication process managed by the Idm component, the ABAC/Attribute-Based Encryption (ABE) component delivers attributes to users, Local Agents, and any other component belonging to the Security Manager through a trusted file structure (like, for instance, an extended version of a JSON web token [20]). These attributes are encoded as a list of cryptographic material. The Security Manager drives the generation of the policies that control the access to resources at both Exporter and Enforcer components of the architecture. Protection against pollution attacks is implemented to avoid that attackers can bind access rights from different platforms to satisfy a complex policy. Policies for time-limited authorization and revocation of access rights are also implemented to increase the security level. Once authenticated, users can use attributes in their possession for accessing to resources and services available within the architecture. Depending on the access policy, they must demonstrate to be in possession of the right set of attributes by performing specific cryptographic operations [20].

4.4 User interface

The Security Dashboard is the main management tool used to build situational awareness, to perform reaction and investigation operations, and to share cyber-threat intelligence.

4.4.1 Situational awareness

Upon analysis, detection, and assessment, users must be made aware of the current situation. Bare technical information (e.g., available algorithms for encryption or integrity, the software version) will be totally useless for most users. The real value added here is to deliver tailored informative contents at different levels of the companies structure, to bring awareness to humans and ensure the better understanding of the current situation. For example, loss or uncertainty in the position of private data triggers a warning about potential violation of a specific regulation to the legal staff. Any loss of integrity, data, or availability can be reported to the management staff, in terms of potential impact on the overall company business (block of the production, loss of customers, bad reputation). Risk assessment at the management layer also

requires to automatically feed existing tools, reducing the reliance on labor intensive and potentially error-prone analysis by experts.

4.4.2 Reaction and investigation

The user interface can be used to select specific analysis and detection algorithms, to visualize anomalies and security events and to pinpoint them in the service topology, to set run time security policies, and to perform manual reaction. With respect to the last two options, it has to be pointed out that security policies are the best way to respond to well-known threats, for which there are already established practice and consolidated methodologies for mitigation or protection. However, the identification of new threats and the elaboration of novel countermeasures require direct step-by-step control over the ongoing system behavior. The dashboard interacts with the orchestration system to give security provider full control over the graph in case of need.

4.4.3 Cyber-Threat Intelligence

Effective reaction and mitigation of attacks largely depend on their timely detection and deep understanding of causes and implications. The accuracy of detection and analysis algorithms is of paramount importance, but the greatest benefit comes from collaboration at the national and international levels, so that appropriate countermeasures and remediations could be undertaken in advance. Again, automation is the main challenge, to overcome the intrinsic slowness of current manual processes. From a technical perspective, the main aspect is the automatic generation of incident reports in standard formats (e.g., STIX), their collection in common repositories, and the generation of cyber-threat intelligence with attack patterns and threat description [46, 47].

From the description of the framework carried out in this section, it is clear that its main goal relies in the quick identification of the compromised parts of the service chain, taking the related remediation and mitigation actions. This process is automatic, i.e., there is no need to declare the weak points of the chain by the service providers. The inter-working among external local agents, Context Broker and security services allows the quick identification of services, or parts of them, that are malicious or under attack. Unfortunately, adversarial or dishonest participants in the chain are very difficult to detect. An effective identification of such actors strongly depends on the trust mechanisms that can be implemented in the platform. This is actually an open issue, that will be further discussed in Section 6.

5 Running example

In this section, an example of application of the proposed architecture in the automotive domain is described in detail. It is pictorially shown in Figure 4. It includes the identification of digital services involved in the creation of an

application for assisted driving, the agents used to expose security capabilities and the main logical elements of the proposed framework.

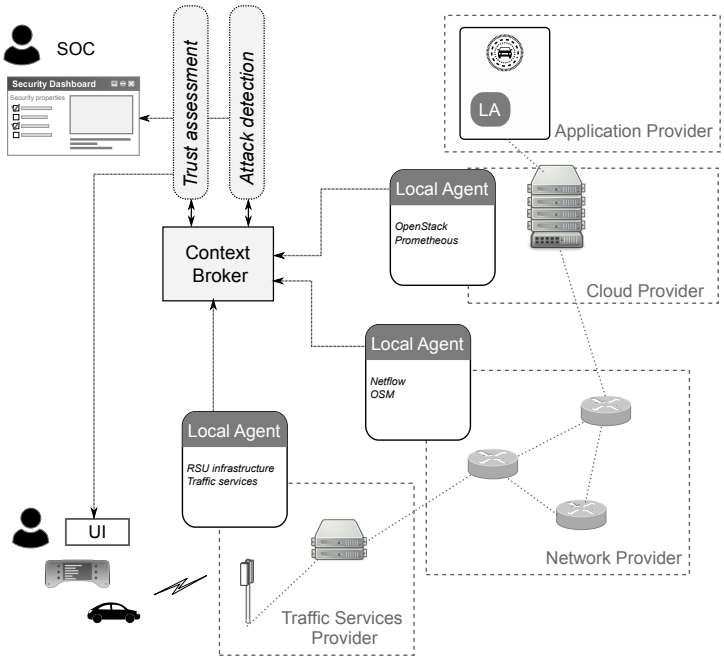

Fig. 4 An illustrative example for the automotive scenario.

Let us suppose that a Traffic Services Provider (TSP) wants to offer infomobility and smart driving services to drivers. According to emerging business models, it likely deploys and manages a number of RoadSide Units (RSUs), but relies on third parties for additional infrastructures and services. For instance, it rents a network slice from a Network Provider (NP) to interconnect all its RSUs on the geographic scale. It also uses an Autonomous Driving Application operated by an Application Provider (AP), which is deployed in the virtualization infrastructure of a Cloud Provider (CP). In this scenario, the TSP plays the role of Service Provider, which combines together services and infrastructures operated by external Resource Providers (NP, AP, CP). The service composition mechanism is not relevant here: this can be done manually by the TSP, by typical bilateral agreements with other providers, or it could be based on some orchestration mechanism that allows to select among a list of alternative providers for the different resources. In the second case, reconfiguration is easier in case of failure or attack to one link in the chain. The End Users of the system are the drivers, through their connected vehicles, which only see the TSP interface but are not aware of the different infrastructures and domains involved in the creation of traffic services.

In general, there are two kinds of cyber-security services that can be implemented in this scenario. On the one hand, the TSP is mostly interested in integrity and availability of the whole chain, which are necessary for its business and, most of all, for the safety of drivers. On the other hand, drivers might have privacy concerns about the propagation and usage of private data (vehicle identification, owner and driver identity, location and routes).

Each service in this business chain runs a Local Agent (LA). Each LA exposes both trust information about the domain (provider identity, security mechanisms in place, certifications, attestations, etc.) and the internal monitoring and inspection capabilities. For example, the LA of the software application collects its logs and events, the LA of the CP collects all the context information obtained through OpenStack or other software orchestrators, the LA for the NP collects information coming from the different subsystems (routers, network devices, NFV orchestrators, etc.), and the LA of the TSP collects information on the RSU infrastructure and the different traffic services. Each LA also includes the links to the LAs of external connected services. The description of identity management and access control is omitted for the sake of brevity, although these represent mandatory procedures in the system.

Let us now suppose that the main concern for the TSP is availability. The TSP asks the SOC this kind of service and provides the root of its chain, namely the identity of the LA operating in the RSU infrastructure. The SOC inputs the root of the service chain to a Discovery Service, which iteratively queries the others LAs of the connected services, hence building the logical topology. Now, based on the security service requested by the TSP, the SOC selects:

- what data to collect from each domain, based on the capability of each LA and what required by the detection logic. In this example, the SOC should consider suitable indicators that help to detect degradation in the overall service quality. This could include, for example: delay and jitter provided by the NP; CPU/memory usage for the Autonomous Driving application, provided by the CP; part of the logs generated by the same application, provided by the AP. It is supposed that this kind of information must be restricted to legitimate users of these digital services, and should not be publicly exposed.
- the *Attack Detection* algorithm that processes the data collected by the agents.

While fully automatic selection of algorithms and data is rather challenging and likely unfeasible in the short/medium term, assisted configuration of the relevant components is already rather easy today. In this respect, the SOC prepares a sort of descriptive template for the service (similar to what already happens for software orchestration in Docker Swarm and Kubernetes), and hands it over to the Security Controller. The Security Controller parses the template and sets up the processing environment:

1. it loads the Attack Detection module and configures it according to the template description;

2. it configures all LAs through the Context Broker; if this feature is supported, it can push its own programs to collect and aggregate specific statistics (e.g., the number of network packets of a specific protocol, the detection of packets with given bit patterns in their payloads, etc.);
3. it loads the set of remediation/mitigation policies from the template, which define what action to undertake when specific events occur; for instance, it could notify the operator for further investigation, or ask a LA to drop malicious packets.

At some time, a botnet is used to perform a DDoS attack against the Autonomous Driving application. This increases the resource utilization of that application reported by the CP, and would probably result in some anomalies in the logs reported by the AP (e.g., incomplete requests, wrong data, access failures). By combining this information together, the Attack Detection engine will identify the DDoS attack before the overall service becomes unresponsive. Mitigation actions might include asking the NP/CP to discard malicious packets (if they can be easily identified) or migrating to a different AP/CP.

For what concerns data sovereignty, the EU may provide some constraints to the SOC in terms of propagation policies. A policy may include the list of providers, countries, geographical areas, and services allowed to use the user data. For example, the user may deny the consent to store his/her data on a storage service that he/she does not trust.

From the user perspective, the TSP presents a list of available services in its infrastructure. Let the user activate the remote Autonomous Driving application, which feeds the on-board systems with information about the behavior of the surrounding vehicles. Before using the system, the user is expected to load his/her preferences and security policies, by using the link provided by the TSP within the Autonomous Driving service. Such policies may also partially reflect expectations from the car manufacturer, which may have business or technical concerns to share information with unknown parties.

In this case, the Security Controller loads the Data Tracking module. This module scans the topology of the service previously discovered and checks the compliance to user policies. Since this service has severe safety implications, the trust policy must consider external services in addition to the TSP. The current topology reports the external network connection, a cloud service, and a third party application. The Data Tracking checks the reliability of the service, for instance, by verifying its identity and digital certificates, the trustworthiness of the external service provider, its location, identity management and access control policies. The trustworthiness of the providers can be verified by checking a user-supplied list of trusted operators, or building their reputation based on external trust chains.

Once the trust policy has verified the compliance with the user preferences, the on-board system requires the Autonomous Driving service to the TSP. Should any topology change occur while the service is used, the event is reported to the Data Tracking module, which checks again the required

policies. The Data Tracking also collects logs provided by the LA about personal information exchanged between the involved services. Then, it records the whole history to make it available to the user either in real-time or offline. This is easy to implement for data-oriented services, as in the case of the Industrial Data Spaces (IDS), where a Connector is present that manages all data exchanges [48].

6 Limitations and Challenges

As described throughout this paper, new computing paradigms and service-oriented architectures require new cybersecurity paradigms beyond the legacy security perimeter model. The general structure and functionality of an innovative framework that could fit the new scenarios has been outlined in the previous sections. It reflects the general trends towards software-defined and multi-tenancy solutions. However, the implementation of the proposed framework needs to address several limitations and open issues that are briefly reviewed in the following.

From a technical perspective the centralized approach is always challenging when a huge amount of information must be collected by many distributed agents. As a matter of fact, existing SIEM architectures mostly gather refined events from local cyber-security appliances, rather then raw data from monitoring agents. However, they already process huge amount of data, implementing a sort of logical funnel to provide selected events that can be understood by humans. All other collected information is lost, together with the possibility to find additional correlations. Even if bearing all monitoring and tracing information to a remote location is unfeasible in most use cases, the centralized approach is anyway necessary to cope with the dynamic and ephemeral nature of cloud-based services, whose lifetime is far shorter than typical enterprise processes. Hence, it is necessary to create persistent storage of events and data, for multiple purposes: offline analysis, forensics investigation, evidence in court. In this respect, more programmability and flexibility in defining monitoring tasks would help mitigating the communication overhead by switching between coarse-grained and fine-grained data, according to the detection needs.

While *programmability* looks like the main keyword in the outlined evolution (and follows the general trend towards software-defined infrastructures and services), it is not anyway straightforward to implement. Technically speaking, loading and running different agents is not a problem today, especially when some form of software orchestration is used (e.g., Docker and Kubernetes). Unfortunately, the real challenge lies in administrative and policy matters. It is rather questionable if and how a Resource Provider could allow an external entity to run its own monitoring agents. First, there is the issue of the integrity and trustworthiness of any additional software, which must not introduce new vulnerabilities and instability into the running system. Second, the scope should be carefully limited to resources used by the

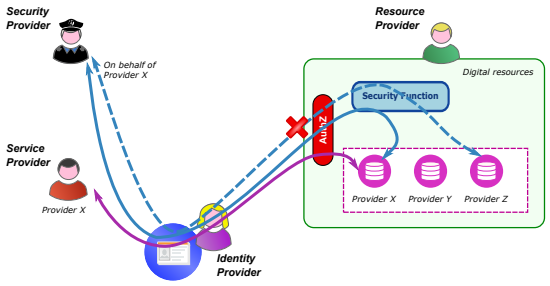

Fig. 5 Externalization of security processes requires to give third party security providers access to monitoring and tracing data of specific tenants.

customer, and should not provide visibility over resources owned and operated by the provider and other tenants. The growing trend towards externalization of security processes further complicates this issue, because proper identity management should be in place to restrict the visibility of a security operator that acts on behalf of a specific customer (like shown in the schematic example of Fig. 5). And, of course, this should be done without introducing manual processes.

In the authors' opinion, full programmability will only be possible with container technologies, where the resources of a common kernel are shared among several users. However, this approach is much challenging for traditional VMs, because the design of hypervisors does not envision the possibility to share its resources and scope. While kernel-level monitoring and tracing is rather simple to implement thanks to namespaces, similar features for monitoring the execution of VMs are more difficult, and require specific extensions to each different hypervisor.

A single agent instance fits well the typical scenarios, where monitoring, inspection and tracing features are implemented by the Resource Provider. If such processes are directly managed by every tenant, either multiple instances with restricted visibility are deployed or existing agents are extended to support multiple users. This represents an important limitation, because of the difficulty to use existing software.

The ability to change monitoring and inspection processes can be effectively used to balance the granularity of collected data with the overhead and to address new and evolving threats. However, this has also an impact on the detection process, that must be able to use different data. One relevant example is ML, which is today a growing trend in cyber-security. While supervised learning is the most common approach to identify anomalies, its application with variable workload patterns is challenging and constitutes the main threat to the validity of such approach. As a matter of fact, the typical difficulty to have a "clean" scenario for learning the normal behavior is now amplified by the fact that the scenario continuously evolves, by including different infrastructures, different topologies, and different measurements. It is therefore challenging to train a system without a solid and persistent baseline.

In this respect, unsupervised learning should be better explored in terms of adaptability to evolving conditions.

The interest towards service-oriented architectures is largely motivated by the possibility to use alternative implementations or providers for a given function. Even though this is really useful to avoid vendor lock-in and to provide resiliency, it introduces trust issues from the security perspective. Indeed, the composition is more useful when it can happen automatically, but this brings the threat of dishonest or adversarial participants in the service chain. This is already taken into account by the illustrative example in Sec. 5, which elaborates on two different aspects. First, the trustworthiness of a Resouce Provider should be verified in advance. To this end, standardized certifications mechanisms would be of great help in determining the provider reliability and reducing the probability of unfair behaviors by participants in the chain. Second, inspection and tracing mechanisms available from each Resource Provider should be used to monitor its infrastructure/software (with all necessary limitations on visibility already discussed above), in order to detect suspicious or fraudulent behaviors. It is anyway unquestionable that this would only provide a weak protection mechanism, largely depending on visibility hooks provided by the provider itself.

From the market perspective, the possibility of rolling out the proposed approach, even partially, to an existing chain is actually a challenge. The main limitation here resides in the lack of standardized interfaces for cybersecurity frameworks, which is mainly a commercial limitation. It complicates the interoperability between the proposed framework and other external entities and infrastructures. This issue has already been pointed out by the I2NSF framework from IETF, which however only discussed the overall concept and use cases, but did not go beyond a very general architecture [4, 45]. The Open2C specification from OASIS has instead delivered the language syntax and transfer protocols [49, 50], but only the profile for stateless packet filtering is currently available [51]. The process is very complex due to the many different relevant cyber-security appliances, and the definition of usage profiles for more complex appliances is still a work in progress. Alternatively, the definition of "security extensions" to existing service-oriented models may represent a quicker approach but with limited scope to specific domains.

7 Conclusions

In this work, a new methodology is described for managing cyber-security of digital service chains. The main contribution of this work is the definition of a new cyber-security paradigm, that goes beyond the legacy security perimeter models and looks at emerging norms for ICT services. This contribution points out how existing SIEM architectures must be re-thought to cope with multi-tenancy, heterogeneous administrative domains, and dynamic topologies. To better show the operating mode of the architecture and the sequence of actions in response to specific events, a real application example has been described,

that shows how this architecture can be considered as a very promising tool for security reinforcement in current and next-generation service chains.

Future work in this direction will consider a concrete implementation of the basic framework for collection of information and management of security services, which represents the starting point to address the main research challenges on detection and automatic operations in the discussed framework.

Acknowledgements This work was framed in the context of the projects ASTRID and GUARD, which receive funding from the European Union's Horizon 2020 research and innovation programme under grant agreement 833456 and 786922, respectively.

References

1. L. Fawcett, S. Scott-Hayward, M. Broadbent, A. Wright, and N. Race, "Tennison: A Distributed SDN Framework for Scalable Network Security," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 12, pp. 2805–2818, Dec 2018.
2. S. Scott-Hayward, S. Natarajan, and S. Sezer, "A Survey of Security in Software Defined Networks," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 623–654, Firstquarter 2016.
3. N. Schnepf, R. Badonnel, A. Lahmadi, and S. Merz, "Automated verification of security chains in software-defined networks with synaptic," in *2017 IEEE Conference on Network Softwarization (NetSoft)*, July 2017, pp. 1–9.
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. 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.
6. G. Pék, L. Buttyán, and B. Bencsáth, "A survey of security issues in hardware virtualization," *ACM Computing Surveys*, vol. 45, no. 3, pp. 40:2–40:34, June 2013.
7. B. Lang, J. Wang, and Y. Liu, "Achieving flexible and self-contained data protection in cloud computing," *IEEE Access*, vol. 5, pp. 1510–1523, 2017.
8. L. Lynch, "Inside the identity management game," *IEEE Internet Computing*, vol. 15, no. 5, pp. 78–82, Sep. 2011.
9. M. Shehab and S. Marouf, "Recommendation Models for Open Authorization," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 4, pp. 583–596, July 2012.
10. A. Vapen, N. Carlsson, A. Mahanti, and N. Shahmehri, "A Look at the Third-Party Identity Management Landscape," *IEEE Internet Computing*, vol. 20, no. 2, pp. 18–25, Mar 2016.
11. I. Indu, P. Rubesh Anand, and V. Bhaskar, "Identity and access management in cloud environment: Mechanisms and challenges," *Engineering Science and Technology, an International Journal*, vol. 21, no. 4, pp. 574 – 588, 2018.
12. V. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone, "Guide to Attribute Based Access Control (ABAC) Definition and Considerations," NIST, NIST Special Publication 800-162, Jan. 2014.
13. Y. Zhu, D. Huang, C. J. Hu, and X. Wang, "From RBAC to ABAC: Constructing Flexible Data Access Control for Cloud Storage Services," *IEEE Trans. on Services Computing*, vol. 8, no. 4, pp. 601–616, Jul. 2015.
14. D. Ramesh and R. Priya, "Multi-authority scheme based cp-abe with attribute revocation for cloud data storage," in *2016 International Conference on Microelectronics, Computing and Communications (MicroCom)*, Jan. 2016, pp. 1–4.
15. K. Yang, X. Jia, K. Ren, and B. Zhang, "DAC-MACS: Effective data access control for multi-authority cloud storage systems," in *Proceedings IEEE INFOCOM*, Apr. 2013, pp. 2895–2903.

16. K. Yang, Z. Liu, X. Jia, and X. S. Shen, "Time-domain attribute-based access control for cloud-based video content sharing: A cryptographic approach," *IEEE Transactions on Multimedia*, vol. 18, no. 5, pp. 940–950, May 2016.
17. K. Xue, W. Chen, W. Li, J. Hong, and P. Hong, "Combining data owner-side and cloud-side access control for encrypted cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, pp. 2062–2074, Aug. 2018.
18. R. LI, C. Shen, H. He, X. Gu, Z. Xu, and C. Xu, "A lightweight secure data sharing scheme for mobile cloud computing," *IEEE Transactions on Cloud Computing*, vol. 6, no. 2, pp. 344–357, Apr. 2018.
19. J. Wei, W. Liu, and X. Hu, "Secure and efficient attribute-based access control for multiauthority cloud storage," *IEEE Systems Journal*, vol. 12, no. 2, pp. 1731–1742, Jun. 2018.
20. S. Sciancalepore, G. Piro, D. Caldarola, G. Boggia, and G. Bianchi, "On the design of a decentralized and multi-authority access control scheme in federated and cloud-assisted Cyber-Physical Systems," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 5190 – 5204, Dec. 2018, doi: 10.1109/JIOT.2018.2864300.
21. M. Repetto, A. Carrega, and R. Rapuzzi, "An architecture to manage security operations for digital service chains," *Future Generation Computer Systems*, vol. 115, pp. 251–266, February 2021.
22. "Network functions virtualisation (nfv); terminology for main concepts in nfv," ETSI GS NFV 003, Aug. 2018, v1.4.1. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV/001_099/003/01.04.01_60/gs_nfv003v010401p.pdf
23. A. A. Khan, M. Khan, and W. Ahmed, "Improved scheduling of virtual machines on cloud with multi-tenancy and resource heterogeneity," in *2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)*, Sep. 2016, pp. 815–819.
24. 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, Nov. 2017, doi: 10.1109/JSAC.2017.2760178.
25. N. Bouten, R. Mijumbi, J. Serrat, J. Famaey, S. Latrè, and F. De Turck, "Semantically Enhanced Mapping Algorithm for Affinity-Constrained Service Function Chain Requests," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 317–331, Jun. 2017, doi: 10.1109/TNSM.2017.2681025.
26. "Network functions virtualisation; management and orchestration," ETSI GS NFV-MAN 001, Dec 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
27. D. Ding, Q. Han, Z. Wang, and X. Ge, "A Survey on Model-Based Distributed Control and Filtering for Industrial Cyber-Physical Systems," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 2483–2499, May 2019.
28. H. Lin, Z. Yan, Y. Chen, and L. Zhang, "A Survey on Network Security-Related Data Collection Technologies," *IEEE Access*, vol. 6, pp. 18 345–18 365, Apr. 2018.
29. J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, Oct. 2017.
30. 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.
31. M. Abdlhamed, K. Kifayat, Q. Shi, and W. Hurst, *Intrusion Prediction Systems*. Springer, 2017.
32. R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," *IEEE Access*, vol. 7, pp. 41 525–41 550, 2019.
33. R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, "Robust Intelligent Malware Detection Using Deep Learning," *IEEE Access*, vol. 7, pp. 46 717–46 738, 2019.
34. F. Ahmad, V. N. L. Franqueira, and A. Adnane, "TEAM: A Trust Evaluation and Management Framework in Context-Enabled Vehicular Ad-Hoc Networks," *IEEE Access*, vol. 6, pp. 28 643–28 660, Jun. 2018.

35. K. Huang, C. Zhou, Y. Tian, S. Yang, and Y. Qin, "Assessing the Physical Impact of Cyberattacks on Industrial Cyber-Physical Systems," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 10, pp. 8153–8162, Oct. 2018.
36. M. Repetto, A. Carrega, and G. Lamanna, "An architecture to manage security services for cloud applications," in *4th IEEE International Conference on Computing, Communication & Security (ICCCS-2019)*, Oct. 2019, pp. 1–8.
37. "Network functions virtualisation; architectural framework," ETSI GS NFV 002, Oct 2013, v1.1.1. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf
38. Q. Li, Z. Tan, A. Jamdagni, P. Nanda, X. He, and W. Han, "An Intrusion Detection System Based on Polynomial Feature Correlation Analysis," in *2017 IEEE Trustcom/BigDataSE/ICESS*, Aug. 2017, pp. 978–983.
39. S. Dua and X. Du, *Data Mining and Machine Learning in Cybersecurity*. Boston, MA, USA: CRC Press, 2011.
40. M. Panda, A. Abraham, and M. R. Patra, "Discriminative multinomial naïve bayes for network intrusion detection," in *2010 Sixth International Conference on Information Assurance and Security (IAS)*, Atlanta, GA, USA, Aug. 23rd-25th, 2010, pp. 5–10.
41. C. Wagner, G. Wagener, R. State, and T. Engel, "Malware analysis with graph kernels and support vector machines," in *4th International Conference on Malicious and Unwanted Software (MALWARE)*, Montreal, QC, Canada, Oct. 13th-14th, 2009, pp. 63–68.
42. H. H. Pajouh, G. H. Dastghaibyfard, and S. Hashemi, "Two-tier network anomaly detection model: a machine learning approach," *Journal of Intelligent Information Systems*, vol. 48, no. 1, pp. 61–74, February 2017.
43. M. Kruczkowski, E. Niewiadomska-Szynkiewicz, and A. Kozakiewicz, "Cross-layer analysis of malware datasets for malicious campaigns identification," in *International Conference on Military Communications and Information Systems (ICMCIS)*, Cracow, Poland, May 18th-19th, 2015.
44. P. Boucher, S. Nascimento, and M. Kritikos, *How blockchain technology could change our lives – In-depth Analysis*. European Parliament Scientific Foresight Unit (STOA), February 2017, iSBN 978-92-846-0549-1.
45. D. Lopez, E. Lopez, L. Dunbar, J. Strassner, and R. Kumar, "Framework for interface to network security functions," IETF RFC 8329, February 2018. [Online]. Available: https://tools.ietf.org/pdf/rfc8329
46. G. Settanni, F. Skopik, Y. Shovgenya, R. Fiedler, M. Carolan, D. Conroy, K. Boettinger, M. Gall, G. Brost, C. Ponchel, M. Haustein, H. Kaufmann, K. Theuerkauf, and P. Olli, "A collaborative cyber incident management system for European interconnected critical infrastructures," *Elsevier Journal of Information Security and Applications (JISA)*, vol. 34, no. 2, pp. 166–182, June 2017.
47. F. Skopik, G. Settanni, and R. Fiedler, "A problem shared is a problem halved: A survey on the dimensions of collective cyber defense through security information sharing," *Elsevier Computers & Security Journal*, 2016.
48. International Data Spaces association, "IDS reference architecture model industrial data space," 2018, version 2.0. [Online]. Available: https://www.fraunhofer.de/content/dam/zv/de/Forschungsfelder/industrial-data-space/IDS_Referenz_Architecture.pdf
49. "Open command and control (OpenC2)," November 2019, language Specification Version 1.0, Committee Specification 02.
50. "Specification for transfer of OpenC2 messages via https," July 2019, version 1.0, Committee Specification 01.
51. "Open command and control (OpenC2) profile for stateless packet filtering," July 2019, version 1.0, Committee Specification 01.