Future Generation Computer Systems 115 (2021) 251–266

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

An architecture to manage security operations for digital service chains

Matteo Repetto [a,*], Alessandro Carrega [b], Riccardo Rapuzzi [b]

[a] IMATI, CNR, Via de Marini 6, 16149 Genova, Italy
[b] S2N Lab, CNIT, Via all'Opera Pia 13, 16145 Genoa, Italy

ARTICLE INFO

Article history:
Received 24 January 2020
Received in revised form 13 July 2020
Accepted 29 August 2020
Available online 3 September 2020

Keywords:
Cyber-security paradigms
Security architectures
Digital services
Service mesh

ABSTRACT

Evolving business models are progressively pushing for increasing digitalization of existing and novel processes. The ICT industry is already addressing this need by massive introduction of virtualization paradigms and tight integration with the physical environment, which allow the creation of multi-domain and complex business service chains. Emerging technologies undoubtedly bring more agility in service deployment and operation but also break traditional security models, which have not been conceived for dynamic and multi-tenancy environments.

In this paper, we briefly elaborate on existing gaps and research challenges towards advanced assurance and protection of trustworthy and reliable business chains spanning multiple administrative domains and heterogeneous infrastructures. We consolidate our analysis in a reference architecture, which includes all functional elements to effectively tackle the dynamic and agile nature of emerging ICT paradigms.

1. Introduction

Data will be the key driver for the digital economy. Novel digital products and services are expected to create, process, share, and consume data and content in a digital continuum, blurring the frontiers between application domains and breaking the current closed silos of information. The major trend in this respect is the interconnection of processes, products, services, and things from multiple vendors on a growing scale [1], to create, transform, share, and distribute data and content, as pictorially depicted for an industrial supply chain in Fig. 1.

The software industry has been progressively introducing new architectures and patterns that bring more agility in the creation and management of new services and products [2,3]. As a matter of fact, the provisioning of dedicated hardware and software infrastructures for every different service does not fit anymore the dynamic and agile nature of modern business models, where digital services and business chains are expected to emerge and dissolve much faster than traditional value-creating networks. Convergence among existing software paradigms, such as virtualization, cloud computing, multi-tenancy, software-defined networking, and the Internet of Things (IoT) is expected to this purpose, leveraging autonomicity and dynamic composition through service-oriented and everything-as-a-service models applied to cyber–physical systems.

However, the combination of such architectural patterns breaks legacy security models, which are still largely based on monolithic appliances and the creation of a sharp and effective security perimeter [4]. Security Information and Event Management (SIEM) systems are nowadays commonly used to collect, correlate and analyze logs, events, and measurements from end systems and cyber-security appliances, in order to build situational awareness over large and distributed systems. They provide advanced analytics which are able to identify on-going attacks and anomalies, but they implement rather rigid solutions because of the following considerations:

- detection, monitoring, and enforcement appliances (e.g., firewalls, antivirus, IDS/IPS) must be deployed at specific locations in the infrastructure (routers, gateways, hosts, end terminals);
- if the physical topology changes, security appliances must be re-deployed and re-configured;
- there is no visibility on third party's infrastructures (like public cloud services, communication networks, IoT devices);
- current design of cyber-security appliances does not fit emerging serverless architectures and Software-as-a-Service models.

In this paper, we propose a flexible and agile paradigm to effectively address cyber-security issues that arise from the on-going transformation of the digital economy towards cross-domain value-chains. Our approach is based on the availability of embedded security capabilities in all digital resources, which

* Corresponding author.
E-mail addresses: matteo.repetto@cnr.it (M. Repetto), alessandro.carrega@cnit.it (A. Carrega), riccardo.rapuzzi@cnit.it (R. Rapuzzi).

https://doi.org/10.1016/j.future.2020.08.044
0167-739X/

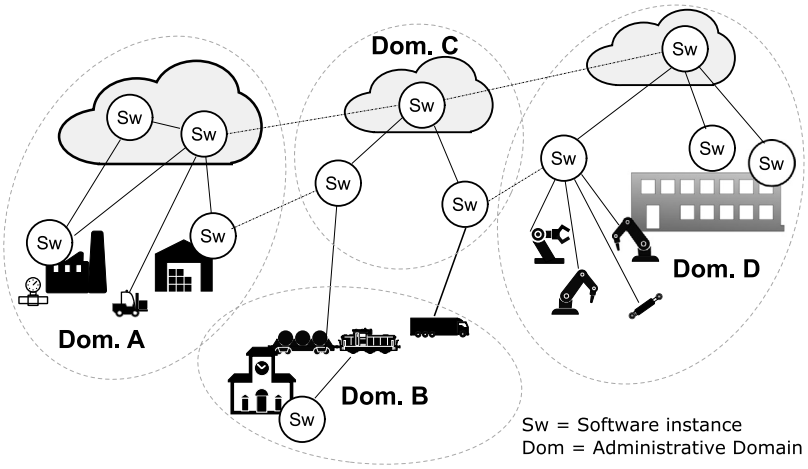


Fig. 1. An industrial supply chain creates, processes, shares, and distributes data among multiple actors and ICT infrastructures.

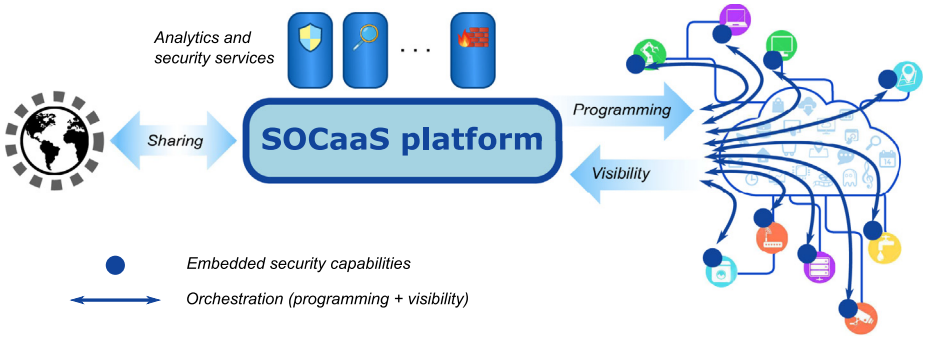


Fig. 2. Illustrative representation of the proposed concept. Security capabilities are embedded in digital resources (software, infrastructures, devices, IoT) and orchestrated by a centralized platform. Cyber-threat intelligence is also shared with other entities.

are then properly orchestrated by a centralized platform that runs analytics and detection services (see Fig. 2). In a nutshell, we advocate that public APIs already used to manage and interconnect digital services (like those exposed by OpenStack, AWS, Google-Cloud, FIWARE) are extended to include "programmable" security functions, e.g., for collecting logs, measurements, events. Then, orchestration is responsible to discover and configure these local security capabilities, and to retrieve security-related information that feeds the internal analytics. Our architecture represents an evolution of current SIEM frameworks, which includes all functional components to effectively tackles the heterogeneity, dynamicity, multi-tenancy, and unpredictability of modern service chains.

A relevant use case for our architecture is the externalization of security processes and the creation of Security Operation Centers as a Service (SOCaaS), since the proposed framework overcome the technical and administrative difficulty in having visibility over ICT resources owned and operated by different providers. It also largely automates most of repetitive and time-consuming activities concerning the deployment and configuration of security hooks throughout the system. The original concept behind this approach was already discussed in the context of virtualized services over Infrastructure-as-a-Service (IaaS) cloud models [5]. Here, we extend the architecture to cover with more general scenarios, including a full range of heterogeneous digital components (software, devices, infrastructures, IoT).

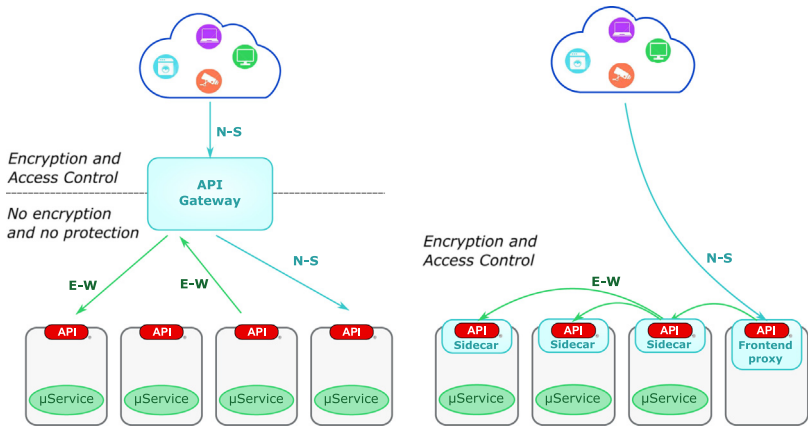
The rest of the paper is organized as follows. Section 2 briefly describes emerging software architectures based on service meshes, how they are changing business roles, and the corresponding impact on the logical boundaries between their operational scopes. Based on these evolving models, Section 3 elaborates on the limitations of existing cyber-security architectures and on current trends in the cyber-security industry; it also points out research challenges for building secure business value chains. Section 4 describes the proposed reference architecture for orchestration of security capabilities, and elaborates on its feasibility based on available technologies and tools. We compare our architecture with existing solutions in Section 5 and describe a relevant Use Case in Section 6. Discussion of related work is given in Section 7. Finally, Section 8 summarizes the main findings of our work and next steps.

2. Emerging computing and business models

Addressing the need for ever-more agility in service development and operation, new computing models are emerging that leverage virtualization and multi-tenancy. This evolution also brings new business models, which unfortunately complicate the management of security processes.

2.1. Software architectures for service meshes

The need for more operational agility and reduced time-to-market in the realization of digital services is pushing a progressive transition from monolithic architectures to service meshes. The first step in this evolution was the introduction of Service-Oriented Architectures (SOA) and Web Services (WS); they brought a new norm for modularity, by leveraging high-level



(a) Micro-services (b) Service mesh

Fig. 3. Micro-services vs. service mesh architectures. Legend: N–S = North–South service calls; E–W = East–West service calls.

interfaces for interconnecting semi-independent objects in private clouds. Following this evolution, micro-services removed the need for common ancillary services (like back-end databases), by encapsulating both logic and data. The idea of micro-services is that they are stand-alone service units completely decoupled and isolated from each other, designed according to the principle of "single responsibility." As such, the internal logic of each micro-service can be implemented with different hardware technologies and programming languages (C/C++, Node, Go, Python, Java, Scada, Kotlin, …); this only requires to expose an interface which is portable and independent from the coding language. Common choices are based on RESTful semantics based on HTTP (SOAP is another possibility, but again more rigid and heavier than the lightweight and free-format REST), gRPC (Google RPC — which provides a binary encoding which is more efficient compared to REST) and Apache Thrift (which provisions for streaming events).

Applications are built by combining functions provided by each micro-service. The loosely-coupled nature of micro-services allows to replace, duplicate, or remove part of them without affecting the operation of the overall architecture. Services can hence grow or shrink dynamically according to the evolving workload, they can update specific functions or protocols, they can be deployed over multiple and even heterogeneous infrastructures.

The ground assumption in this model is the ability to discover micro-services and related functions, which entails the presence of service registries. In addition, load balancers are commonly used to implement horizontal scaling. Though there are not rigid patterns, typical architectures use an API gateway that routes and secures requests that come from outside and acts as load balancer for internal service calls (see the left side of Fig. 3(a)). The former are usually denoted as North–South (NS) traffic, and the latter as East–West (EW) interactions. The presence of an API Gateway still represents a weak form of perimeter in the application; all E–W traffic may happen in clear text, whereas only communications that are situated north of the gateway are encrypted.

The latest step in this evolution goes beyond the heterogeneity in the implementation of external APIs. As a matter of fact, different codes, different languages, and different internal architectures usually lead to inconsistent semantics, methods, encryption schemes, authentication mechanisms, access policies, logging and so on. The API gateway definitely represents a potential bottleneck and single point of failure, so the natural evolution was to break this functionality down within each service. A common and shared component therefore is placed alongside the main business logic as a lightweight sidecar that acts like a communication proxy and provides the front-end communications (e.g., HTTPS, RPC, AQMP/HTTP) and security functions[1] (Fig. 3(b)). The architecture now becomes fully distributed and the boundary between N–S and E–S communications blurs, since every service can be leveraged as a public interface (even if dedicated proxies can still be deployed as public frontends). The result is a fully decentralized service mesh without any sort of security perimeter, which allows more freedom in the placement of each micro-service, since the sidecar proxy applies peer-to-peer encryption, authentication, and access control functions. A subsidiary framework is still needed to mediate security properties and to establish trust relationships, for instance by distributing certificates, collecting logs and performance measurements, assessing the global health.

2.2. Business roles and service chains

The growing size, complexity, and dynamicity of industrial and commercial value chains are increasingly demanding more and more flexibility in the creation, operation, and disposal of digital assets, in order to quickly follow new market and technological needs. Traditional business models based on the provisioning of dedicated hardware, software, networking, and data resources are no more sustainable today, because of the relevant investment in buying and configuring the infrastructure in front of the short lifespan of each service instance.

Since the largest revenue streams often come from the design and operation of tailored services, there is a growing diversification between *Service Providers* (SPs), which design and operate digital services, and *Resource Providers* (RPs), which own digital assets and make them available to multiple customers by leveraging virtualization and multi-tenancy. This is an effective mechanism to share infrastructures and resources, which can therefore be composed in new products and solutions faster than ever. Under this evolutionary process, we can make several examples of RPs:

- Infrastructure Providers own physical resources and infrastructures (e.g., data centers, metropolitan and geographical networks, IoT installations),
- Software Providers develop applications and make them available in public or private repositories (e.g., github),

[1] See, for instance, the Envoy framework: https://www.envoyproxy.io/.

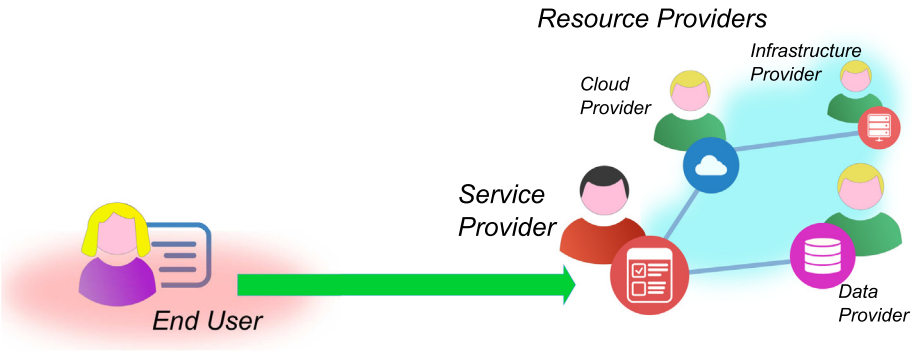


Fig. 4. Business roles in a modern service chain.

- Cloud Providers combine computing and storage infrastructures into virtualized services according to IaaS, PaaS, or FaaS models[2] (i.e., they provide bare VMs, storage services, lambda functions),
- Network Operators implement large-scale communication services for public and private users (mobile networks, Virtual Private Networks, Network Functions Virtualization),
- Function Providers implement specific logical functions (e.g., authentication, databases, context brokers).

Service Providers create and operate value-added digital chains, for example by selecting software, by deploying it in the cloud, by connecting to IoT devices or data brokering services, by reading data from data bases, by connecting to external authentication services, by securing communications with remote peers, etc. However, while the End User (EU) still expects to interface with a single SP, a number of additional hidden providers contribute to supply the necessary resources (see the illustrative example in Fig. 4). This raises important questions about who takes the overall liability towards the EU, especially when private and sensitive data are used within the service.

3. Security challenges for emerging ICT paradigms

Service meshes allow to discover available services and to combine them into processing chains. The ultimate benefit is the agility in deploying new services and shaping the processes to evolving business needs, but the composite nature of digital services will lead to complex dynamics, dispersion of data among the multitude of digital objects and infrastructures, unknown or hardly-traceable topologies. The dark side of this evolution is the risk for large unpredictability, due to non-deterministic, opaque and partially inscrutable service topologies. Indeed, 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 own requirements and expectations.

In general, open security issues include the overall behavior of the system, the location of personal and sensitive data, the sanity of the software, the availability of the whole service, and, most of all, the ability to perform quick remediation and mitigation actions in case something goes wrong. Novel security and privacy models are required, as agile development does not fit the sequential nature of traditional security engineering processes [6]. As a matter of fact, a compromised element in a service chain represents a privileged attack vector to affect other systems or the whole chain; it is no coincidence that small devices and smart things are ever-more preferred targets for initiating more complex attacks. In addition, the growing complexity of cyber-attacks, often based on multi-vector approaches, are urgently demanding more correlation in space and time of (apparently) independent events and logs, and more coordination among different security applications.

The major limitations of existing cyber-security paradigms can be attributed to the following aspects:

- rigidity of the architectures, which often rely on the isolation provided by the security perimeter and do not allow integration of multi-vendor solutions;
- flaws in security appliances, which increase the attack surface;
- limited visibility due to the difficulty to install security agents in large, distributed and heterogeneous systems;
- low degree of automation and the need to rely on the skills and expertise of humans;
- heterogeneity of control and management interfaces of cyber-security appliances, for which no standard is currently available [7];
- outdated trust models, which assume static topologies and white-box models.

3.1. Evolving cyber-security market

The market of cyber-security solutions largely reflects the different needs of different scenarios and customers. While discrete standalone appliances like firewalls, virtual private networks, intrusion prevention and/or detection systems (IPS/IDS), deep packet inspection (DPI), DOS mitigation, and antivirus perfectly fit the size of individuals and small businesses, they are impractical for larger and more complex systems. In this case, many vendors are chasing new business opportunities by delivering integrated solutions for the enterprise, pure and hybrid cloud, industrial devices and networks, and security analytics. Their solutions are often based on SIEM tools, which collect security logs and configurations from multiple sources (end systems, security appliances) and feed a powerful set of analytics and detection engines.

Table 1 gives a simple taxonomy of the main classes of solutions currently available in the cyber-security market, including an illustrative (yet not exhaustive) list of commercial products, which highlights a rather crowded market. By looking at their main characteristics, we can identify some major trends that address the limitations previously identified:

- *security services beyond the perimeter model*, which integrate the cloud and IoT in unified enterprise security solutions;
- *improved correlation techniques and recognition of unknown attacks*, by correlating events through (*i*) centralized machine learning and other artificial intelligence paradigms applied to wide data sets, which are aggregated across multiple inspection points, and (*ii*) big data capabilities;

[2] See below for a brief definition of these service models.

Table 1
Classification of main cyber-security appliances.

Type	Products	Main characteristics
Network visibility and segmentation	Darktrace, Cisco Stealthwatch, Plixer, Radware Cloud DDOS protection, Andrisoft Wanguard, Flowmon, f5 silverline, Trend Micro Network Defence	Inspect network traffic (sometimes encrypted traffic analytics are available). Use sensors or flow-monitoring protocols (NetFlow, sFlow, IPFIX, J-Flow, NetStream). Support for enterprise, industrial, cloud networks. AI and machine learning technologies. 3D visualization of network topology. Network segmentation (Cisco only). Closed solution (vendor lock-in). Single domain solution. No mechanisms for tracing user data. DDoS protection, anomaly detection, malware detection, zero-day exploit. Investigation and query-based filtering.
Endpoint protection	Cisco AMP for Endpoints, Kaspersky Endpoint Detection and Response, Symantec Endpoint Protection, Trend Micro Endpoint Security	Centralized analysis and detection based on Machine Learning and threat intelligence. Protection of applications, devices, web. GDPR-compliant. Vulnerability assessment and automatic patching. Mono-domain.
Cloud protection	Kaspersky Hybrid Cloud Security, Symantec Cloud Workload Protection Suite, Trend Micro Cloud-native Security, McAfee Cloud Security	Support private (VMware, KVM, Xen) and public clouds (AWS, Google, Azure) Use public cloud APIs, agentless deployment by providing services in relevant marketplaces. Integrate private and public clouds. Integrate analysis with enterprise's networks. Segmentation and access control. Mono-tenant.
Embedded, industrial, and IoT devices	Kaspersky Embedded Systems Security, Kaspersky Industrial Cybersecurity, Symantec Industrial Control Systems Protection, Fortinet	Efficient design for low-end hardware. Default Deny for Apps, Drivers and Libraries. Antivirus On-Demand. File Integrity Monitoring and Log Audit. Tackle specific security concerns for embedded systems: geographical dispersion, rare updates. Secure every industrial layer, including SCADA servers, HMI, engineering workstations, PLCs, network connections. Mono-domain.
SIEM/Security analytics	Cisco Cognitive Threat Analytics, McAfee SIEM, McAfee Security Analytics, Kaspersky Threat Management and Defence, Symantec Advanced Threat Protection, IBM QRadar, MicroFocus ArcSight, Solar Winds SIEM.	Get real-time visibility into all activities on systems, networks, databases, and applications. Apply artificial intelligence, machine learning to wide data sets. Aggregate intelligence across multiple control points. Correlate threat events. Provides in-depth threat visibility across IT environments in one place. Quick one-touch remediation actions. Big data capabilities. Some concerns from users about logging sensitive data or events. Advanced search and forensic analysis.

- *more efficient architectures* that reduce the overhead of running security appliances, based on the separation between monitoring tasks and detection intelligence, agentless deployment, efficient design of monitoring probes for low-end hardware, antivirus on-demand.
- *integration with other security processes*, i.e., vulnerability assessment and automatic patching, advanced search and forensic analysis;
- *improved policies*, which tackle specific security concerns for embedded systems (e.g., geographical dispersion, rare updates), including default deny behavior for apps, drivers and libraries;
- *improved awareness and reaction*, including (*i*) real-time visibility into all activity on systems, networks, databases, and applications, (*ii*) in-depth threat visibility across IT environments in one place, (*iii*) quick one-touch remediation actions;
- *privacy concerns* from users about logging sensitive data or events.

Despite the promising trends, we must ascertain the substantial heterogeneity in cyber-security tools and their interfaces, which hinders their composition in flexible and agile platforms and leads to non-interoperable vertical silos and a substantial market fragmentation. Indeed, the lack of open and clear specifications for control and management interfaces is already known, despite recent attempts to define a common framework [8].

Almost all products listed in Table 1 are basically black boxes, do not give any details about the internal monitoring hooks and detection algorithms, and are not interoperable with products from other vendors. The general feeling is that tailored solutions are delivered to each customer based on its specific environment, by composing together a number of standalone appliances. The difficulty to combine together heterogeneous appliances, interfaces and data formats is largely confirmed by open-source frameworks (for instance, OSSIM, Wazuh, Elastic Stack, Apache Metron), which are often limited to collect from a few sources (Snort, Suricata, Elastic beats among the most common tools) and protocols (e.g., NetFlow). Integration with additional sources usually requires specific extensions or plugins. In addition, deployment and configuration of all components implies manual intervention every time the composition and topology of the system changes.

3.2. Widening visibility over the chain

The progressive disaggregation between RPs and SPs is at the heart of the main cyber-security challenges for creating digital value chains. As a matter of fact, the separation of the ownership of resources and services makes the identification of responsibilities and security obligations more difficult than in traditional ICT installations. Indeed, the presence of multiple logical layers, different provisioning models, and multi-tenancy has a disrupting impact on the nature of risks, roles, and related responsibilities.

Though the overall objectives and scope of security do not change when cloud technologies are used, they must be split between different actors, and the demarcation line is not always clear to everybody. Yet this allocation is far from being simple to manage, since there are many interdependencies and correlations between the technical and administrative aspects that fall under the scope of different actors. Fig. 5 maps the responsibilities of SPs and RPs; the matrix decomposes ICT services in multiple logical layers, adopted from the Cloud Security Alliance [9], and shows how the responsibility is split for different business models.

The Infrastructure layer includes the hardware: servers, disk arrays, network equipment, sensors, actuators and any other device that can be involved in the realization of cyber–physical systems. The metastructure layer is the glue between the infrastructure and the applications. It includes protocols and mechanisms for management and configuration, which allow provisioning of virtual resources. The applistructure layer defines the main business logic and the underlying artifacts to create the software environment. Finally, the infostructure layer concerns data and information managed and stored in the service.

Multiple business models are taken into account that could be used by SPs. The legacy approach is to buy and operate the whole

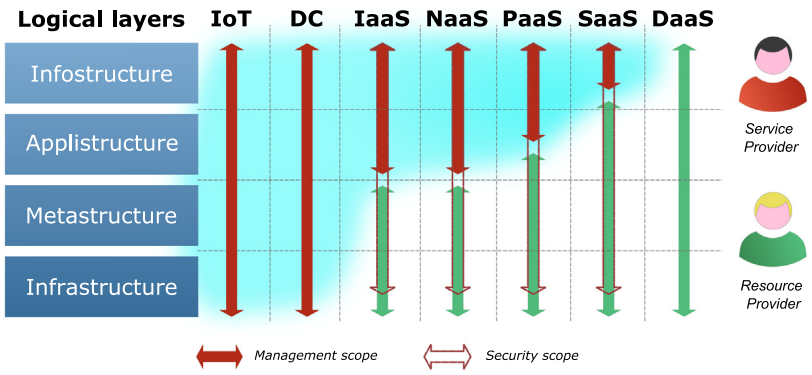


Fig. 5. The matrix of responsibilities (and visibility) for different virtualization models used in the realization of digital services.

infrastructure, including things (IoT) and ICT installations (Data Centers). Financial and operational sustainability suggests limited usage of this model, e.g., to implement critical services, while additional resources will be acquired on-demand by 'as-a-Service' paradigms[3]:

- Infrastructure-as-a-Service (IaaS): computing, networking, and storage resources are abstracted and pooled together to create virtual instances of typical ICT infrastructures. The RP is only responsible for the physical infrastructure, while the SP takes on the responsibility for everything he deploys and operates in the virtualized environment.
- Network-as-a-Service (NaaS): the scope is rather broad, ranging from local area networks (which are often included under the umbrella of IaaS) to wide area networks. From a security perspective, this model is largely equivalent to IaaS.
- Platform-as-a-Service (PaaS): an additional layer is added to the IaaS, including application development frameworks, middleware capabilities, and functions such as databases, messaging, and queuing. The RP manages all security aspects related to the platform, which includes the infrastructure and software facilities (like databases, domain name servers, operating systems, compilers, analytics services, etc.). The SP is responsible for everything he implements and runs on the platform (e.g., data processing, connection to his own things). However, there are some common security services (like logging, monitoring, auditing) that lie in between, since they may concern elements operated by both actors. In this case, the risk with respect to efficiency is unnecessary duplication of similar security frameworks and components.
- Software-as-a-Service (SaaS): SaaS services are full, multi-tenant applications, with all the architectural complexities of any large software platform. The RP is responsible for nearly all security aspects, including operation of the infrastructure and the hosted software. The responsibility for the SP is mostly limited to identity management and access control. Serverless models remove the day-to-day responsibility of updates for cloud users, but also require to work with providers with strong track records. This model should foster cloud provider to maintain extremely high security levels, since this is necessary to build their reputation and credibility.
- Data-as-a-Service (DaaS): data products can be provided to the user on demand, regardless of geographic or organizational separation between providers and consumers. The RP is responsible for providing reliable and safe data to authorized customers. The responsibility may be partially shared with other RPs, if the data is hosted on third party's infrastructure.

When virtualization is not used (e.g., IoT and legacy Data Centers), the SP is the only responsible for all layers, including tampering, eavesdropping, and any unauthorized physical access to devices. In this case, the SP has the best visibility on the evolving context, including information from the hardware, logs and events from applications, statistics from network appliances. However, the whole management burden and financial investment may be cumbersome for the smallest businesses.

The shaded cyan area in Fig. 5 highlights the visibility of the SP over the deployed services, which corresponds to his technical responsibility. It is clear that massive recourse to 'as-a-service' models hinders the possibility to consider events and conditions that may be symptomatic of forthcoming threats to valuable assets in the overall service. Thus, we argue that it is necessary to improve visibility over digital services beyond the cyan area. In abstract terms, the goal is to lengthen the red arrow down to the bottom layers for security purposes, while keeping the current separation of concern for management aspects, as pictorially depicted in Fig. 5.

Improving visibility helps create better awareness and detection opportunities, but does not remove the responsibility of each actor. Specific Service Level Agreements (SLAs) must be established between SPs and RPs, which establish the support that is offered in case of incidents. This could include additional logs for post-mortem analysis, timely notification of on-going or presumed attacks, escalation procedures, response time, etc.

4. A new architecture for orchestrating security capabilities

The operation of SOCs is largely based on SIEM tools that provide security engineers with the necessary situational awareness to investigate anomalies and to undertake actions. SOC and their technical platforms must be designed according to the system to be monitored; in particular, cyber-security appliances and monitoring agents must be carefully placed to collect all relevant information, events, and measurements that are useful for detection and assessment. Though the implementation of internal SOC is a common practice for large organizations, small businesses do not have the operational, technical, and financial capacity to autonomously operate these competence centers, so externalization of security services is a growing trend in the market.

Following the general trend towards Everything-as-a-Service models (XaaS), the concept of Security Operation Centers-as-a-Service (SOCaaS) has been used to denote a business model where security processes are delegated to an external entity. Specific features that characterize a SOCaaS approach include:

[3] 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 [6–8]. However, there is not a shared understanding about this concept in the technology or business jargon, so we do not include it in our analysis.

- outsourcing analysis and investigation to dedicated teams of security experts;
- managing detection and response, also in a (semi-)automated way;
- replacing bare SIEM with more powerful and flexible collection tools;
- ensuring compliance to regulations;
- carrying out behavioral analytics and log analysis with innovative tools, including AI;
- triggering real-time alerting;
- performing vulnerability scans and finding out bogus or compromised software.

The implementation of SOCaaS clearly requires tight integration between the infrastructures of SOC operators and their customers, to provide the former with visibility over the evolving execution context, and also demands for more agility in deploying and configuring SIEM platforms. Indeed, the flexibility and dynamicity in composing and modifying value chains complicates the deployment and set up of cyber-security systems, because new resources could be added to and existing resources could be removed from the overall process. The deployment of capillary and pervasive security agents throughout the ICT infrastructure (network devices, servers, users' terminals, smart things) is among the most critical issues in the realization of SIEM platforms, not to mention that this operation is usually impossible in third party's infrastructures. However, the transition towards micro-services and service meshes described in Section 2.1 is already boosting the adoption of common management APIs and sidecars, which may be extended to include security capabilities as well.

A second important aspect for the implementation of the SOCaaS paradigm is more agility and modularity in the set of available analytics and assessment algorithms. A SOCaaS platform should not be limited to a rigid and pre-defined set of security services, but should be easily reconfigured to run new algorithms, so that security specialists can implement tailored security processes for the specific context of each customer. Technical speaking, the challenge is not the deployment of a new analytics engine, rather the automatic re-configuration of the whole system to enable the required agents and to format data in the proper way.

Though the current approach is largely based on the collection of logs from discrete cyber-security appliances placed in specific devices in the infrastructure (servers, routers, gateways), the integration of different computing paradigms like the cloud, IoT, and serverless computing will make this solution largely unpractical. The long-term vision and emerging trends in the cyber-security industry are therefore suggesting the need to evolve from the composition of discrete detection and enforcement appliances to integrated, pervasive and capillary systems, which decouple distributed context monitoring from (logically) centralized detection and analytics logic. Key enabler for this evolution will be the architectural separation between analysis and data sources, mediated by proper abstraction; this paradigm will result in an open, modular, pluggable, extendable, and scalable security framework.

Beyond the conceptual view suggested by open challenges and evolving paradigms, it is worth elaborating more in details on a reference architecture that highlights the main functional components and suggests research directions under a common perspective. Fig. 6 shows the main architectural elements that are necessary to achieve the aforementioned objectives. The model is composed by four main macro-blocks:

- *digital services*, which are extended with embedded security capabilities for monitoring, inspection and enforcement;
- *SOCaaS platform*, which extends current SIEM technologies with additional enforcement capabilities, if available, and the possibility to configure all components to work together;
- *Security Dashboard*, which is the Human–Machine interface for visual analytics and definition of control/management policies;
- *AAA framework* for managing digital identities and access policies in a coordinated way.

4.1. Digital services

The realization of secure business processes through chaining of elementary functions will only be possible if security capabilities are integrated within each digital service, and such capabilities are dynamically combined to create complex security appliances. Once common interfaces and semantics are available to access security functions, their implementations can be tailored to the specific environment: programming language, virtualization mechanism, software architecture, etc. In this respect, similarly to what already envisioned for the implementation of public APIs, the development of portable sidecars could be an effective way to improve the quality and effectiveness of security capabilities.

A security sidecar should include the following set of *Security Functions*:

- *monitoring* of log files and system events (including, for instance, information and measurements available from pseudo-filesystem like *proc* and *sys* in Linux);
- *inspection* of network traffic at the different OSI layers, from raw packets to application messages (like HTTP, FTP, SOAP, DNS);
- *tracing* of execution and control paths, which may be limited to system calls and I/O calls, or may extend to memory and registry dumps;
- *enforcing* of security rules at the network and application layer (for instance, by dropping packets based on IP header fields, by blocking specific HTTP request messages, by preventing access to files or peripherals).

Additionally, data fusion and aggregation functions are required to perform ancillary operations like removing redundancy, compressing data, timestamping, etc. In general, detection services could be included as well by running existing cyber-security appliances, but we do not include this option in the list because it does not fit well many relevant scenarios (e.g., serverless computing).

A very challenging feature for security sidecars is programmability, i.e., the capability to take on the execution of lightweight data filtering, aggregation, processing, and enforcement tasks. This goes beyond mere re-configuration of individual functions and brings the challenge to safely run code injected by an external (though trusted) entity. However, offloading tasks to local services helps balance the trade-off between processing and network overhead in an effective way.

The range of local capabilities is rather broad and should be tailored to the specific nature of the digital service (IoT, cloud, lambda function, storage, …). Yet it should be dynamically tuned to the evolving context, while pursuing the better trade-off between granularity and overhead. A *Local Control and Management framework* is therefore necessary to govern life-cycle events of each security function (e.g., start, stop, reload, update) and to set its operational parameters (e.g., name of files, filter rules, event names). The control and management framework also exposes security properties of the digital service, including vendor, name,

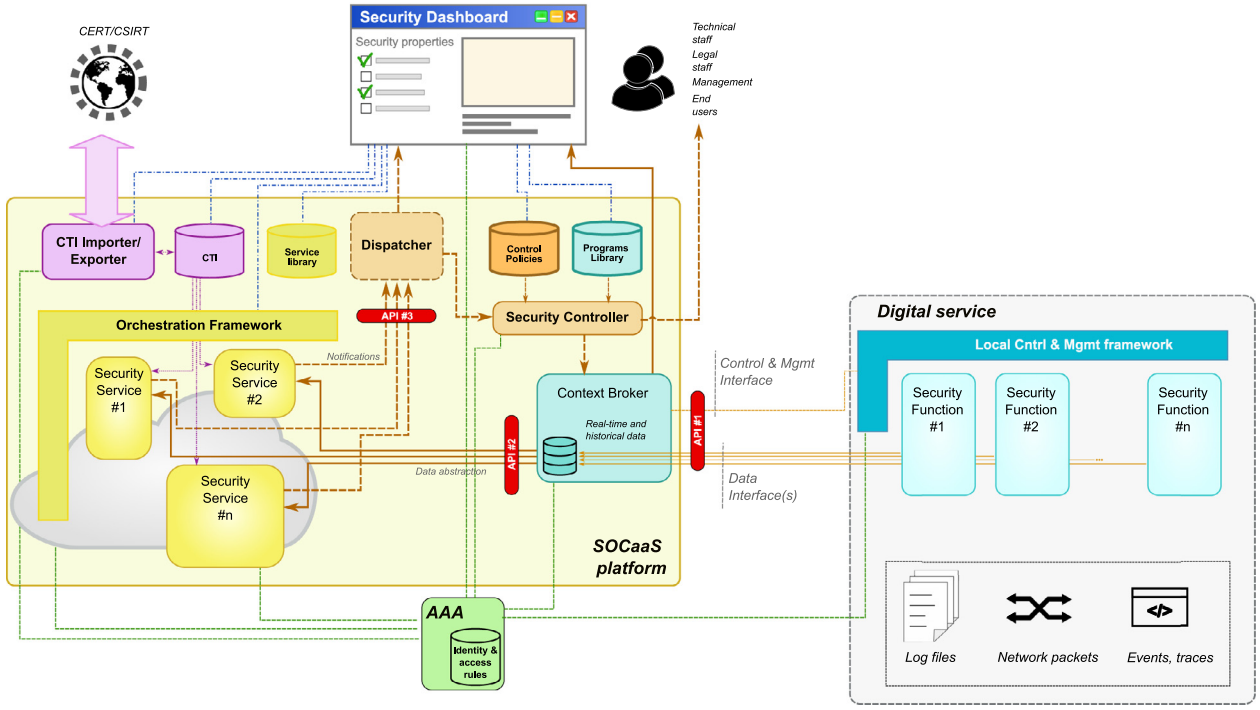


Fig. 6. Reference architecture for next-generation SOCaaS platforms.

description, version, release date, digital certificates, and all other internals that can be safely shared with a SOC.

The external API encompasses two aspects: control and data (API #1). The control interface heads to the control and management framework and is used to discover security capabilities and to enable, disable and configure the security functions. This is a major innovation which is necessary to orchestrate security functions but it is still missing today [7]. The data interface can be directly implemented by each security function and is used to report the collected set of events, data and measurements.

4.2. SOCaaS platform

When provided "as-a-Service", a SOC must have the flexibility to implement any security process that is required by different and heterogeneous business value chains. Accordingly, alongside legacy functions for collecting and storing security-related data, the design of an effective platform for implementing SOCaaS should encompass the ability to deploy and run new algorithms at run-time, by connecting to and orchestrating multiple agents in different technical and administrative domains.

4.2.1. Context broker

The *Context Broker* is the architectural entity which collects information, data, events, and measurements from local security agents embedded in each digital service. The Context Broker is not a plain database, since it should address the heterogeneity of sources and protocols; in this respect, it provides a common data model and abstraction for discovering, configuring, and accessing the security context available from the different services.

The Context Broker terminates both the control and data channels towards remote services. It has a context discovery layer that queries all components involved in the chain and builds the logical topology of the overall chain, including the security properties and capabilities of each service. The main purpose for a Context Broker is therefore to describe what security capabilities are available from each service (i.e., what can be retrieved) and the current configuration (i.e., the set of data that are being collected). It also allows to change the reporting process, by enabling/disabling security features, by changing the configuration of remote security functions, and by offloading simple tasks. With respect to data management, the Context Broker delivers events and measurements in real-time to the detection and analysis algorithms (API #2) and stores historical data for off-line analysis, forensics, and legal obligations. It hides the heterogeneity and asynchrony of the sources, organizes historical data, and provides simple querying and fusion capabilities in data access.

The presence of a common Context Broker for all detection algorithms improves the efficiency of the whole system, since the same information is collected only once and shared among all active instances. The Context Broker should always mediate between conflicting requests coming from different algorithms (enable/disable monitoring, frequency and granularity of reporting) and find the best configuration that fulfills all of them.

4.2.2. Security services

Current SIEM platforms are often designed to collect refined events from local cyber-security appliances (DOS detection, IDS/IPS, firewall, antivirus, etc.). However, the likelihood of detection of distributed, advanced, and persistent threats increases when the largest base of raw events are analyzed and correlated in their time and space dimension. A progressive migration of cyber-security appliances towards the SOC platform is therefore necessary to improve the detection processes, though partially supported by task offloading for better efficiency. More efficiency is also expected because similar monitoring and inspection operations are not replicated for each appliance.

The broad set of security challenges for multi-domain value chains suggests the need for a wide range of *Security Services*, including but not limited to:

- *Attack detection* — Rule- and signature-based algorithms show their limits in addressing the growing complexity and continuous mutation of attacks, while the creation of legitimate usage profiles is a complex and cumbersome task. More intelligence is needed to process the security

context and to correlate even apparently uncorrelated heterogeneous events and data (network traffic, log files, user behavior) from different systems. In this respect, the current challenge is to understand the applicability and effectiveness of existing Machine Learning methods, including but not limited to, K-Nearest Neighbors, Naive Bayes, Graph Kernel and Support Vector Machine [10–12].

- *Threat identification* – The investigation of new threats and detection of zero-day attacks requires the ability to inspect events, data, and measurements beyond what can be envisioned at design time. Programmability helps in this direction, by providing the freedom to define new filtering and monitoring processes at run-time, tailored to the specific environment. Given the impracticability of elaborating detection rules for unknown threats, the real challenge is unsupervised learning, which could autonomously identify anomalies, i.e., non-conforming patterns compared to the well-defined notion of "normal" behavior.
- *Data tracking* – The availability of security API in each digital service 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's 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 requests for removal.
- *Trust and risk assessment* – When heterogeneous services are automatically selected from different domains and chained together, their security properties (including, but not limited to: vendor identity, encryption/integrity/digital signing mechanisms, digital certificates, supported monitoring/inspection/enforcement hooks, installed software versions and patches) should be formally verified to satisfy the high-level trust policies (trusted vendors/countries, minimal encryption requirements, trust chains, security mechanisms, etc.) of users. In other words, users should be aware of any weakness, and should be able to decide whether it is acceptable or not.

From a more technical perspective, an *Orchestration Framework* is necessary to run and scale security services according to the current workload, so to not waste resources or delay the analysis. It is responsible for loading the services from an internal repository, deploying them, and performing life-cycle management actions. There are no specific requirements on the underlying infrastructure (a single server, a data center, a IaaS cloud, big data framework) apart that a management interface must be available to automate as much as possible provisioning and instantiation.

4.2.3. Security controller

The *Security Controller* is the smart entity that automates as much as possible all security workflows. According to on-going initiatives [8], the main role of the Security Controller is mediation between control policies and security services. In practice, it translates high-level behavioral guidelines into concrete actions, rules and configuration settings applied both to the Context Broker, Digital services, and the Orchestration framework. This means that the Security Controller launches Security Services when required, sets up local security capabilities, collects the relevant context in the proper data format, feeds analytics engines, and dispatches notifications to humans through the *Dispatcher* function (API #3).

The implementation of the Security Controller may vary from bare event-driven engines to more complex rule management systems, up to the application of Artificial Intelligence to learn and improve strategies from humans' behavior. Accordingly, the specification of policies also changes from Event-Condition-Action (ECA) patterns to more abstract definition of goals, constraints, and even intents, which could be defined by non-technical users [8,13]. The adoption of advanced reasoning models would open the opportunity for dynamically adapting the response to new threat vectors. In this respect, the analysis of historical data about the actions undertaken in response to specific events and conditions with their impact on the system could be used to assess the effectiveness of response strategies, and in case to improve them.

The degree of autonomy in taking decisions and applying the corresponding actions may also range from full automation to supervision, based on the fact some additional input or just explicit consent is required to human operators. Automatic reaction shortens response times and unburdens humans from mechanical and repetitive tasks. However, full awareness and the need for post-mortem analysis recommend to keep track and to report any action to the dashboard, at least to give visibility of the occurrence of attacks.

4.2.4. Information sharing

A SOC should always work in close cooperation with Computer Emergency Response Teams (CERTs) and Computer Security Incident Response Teams (CSIRTs), because they represent the main sources of information about new threats, vulnerabilities, and attack patterns; SOCs should also promptly report any discovered threat or vulnerability, so to contribute to limit the impact of zero-day attacks. Unfortunately, despite of massive digitalization in all economical and business sectors, sharing of information about cyber-threats and security incidents is still largely based on paperwork and emails. This delays the knowledge of new threats and attacks, as well as the elaboration of remediation actions and countermeasures for every different context.

The creation of cyber-threat intelligence in an automated way needs to address two main aspects. The first is the creation of common models and abstractions for the description of threats, vulnerabilities, and infrastructures. The second, and more challenging, is the derivation of the necessary information from an on-going attack or offline analysis [14,15]. While the former is already addressed by several standardization initiatives (e.g., STIX), the latter is still an open challenge with unclear directions.

4.2.5. Internal repositories

The different architectural components of the SOCaaS platform rely on a set of internal repositories that store the following information:

- *Programs library*: though the long-term ambition could be the generation of new inspection, monitoring, and data fusion tasks according to high-level instructions or policies (i.e., a sort of "compiler"), short/medium-term implementations will likely make use of user-defined programs. They should include proper description, constraints, and any metadata that is necessary for their correct deployment and instantiation.
- *Service library* contains software packages that are needed to run the corresponding security services. The package may only consist of the binary executable or deployment scripts, depending on the underlying infrastructure and orchestration tools.
- *Control policies* define different security workflows in abstract terms. 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.

- *Cyber-Threat Intelligence* (CTI) describes possible threats for one or a group of organizations or infrastructures. It is a bi-directional repository, which should merge existing and new findings for both local usage and sharing with national and international CERTs/CSIRTs.

4.3. Security dashboard

The *Security Dashboard* is the main management tool used to build situational awareness, to perform reaction and investigation, and to share cyber-threat intelligence. The design of user-friendly human interfaces is among the key priorities for establishing the success of commercial solutions. Beyond visual appearance, it is worth widening the base of users and tailoring the informative content to their specific role and background. As a matter of fact, bare technical information (e.g., available algorithms for encryption or integrity, the software version) will be totally useless for most users. For example, loss or uncertainty in the position of private data should trigger a warning about potential violation of a specific regulation to the legal staff. Any loss of integrity, data, or availability should 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.

For the purposes of reaction and investigation, the Security Dashboard 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. The identification of new threats and the elaboration of novel countermeasures require direct step-by-step control over the ongoing system behavior. The dashboard should therefore allow direct interaction with the Context Broker, to set the most suitable monitoring and inspection configuration in each remote service.

4.4. Identity management and access control

Following the generalized trends towards improved modularity and agility, the implementation of SOCaaS platforms is likely to adopt micro-services and service mesh architectural patterns. North–South (N–S) interactions take place with external components (like digital services and management dashboards), whereas East–West (E–W) transactions involve the internal components already identified in Section 4.2. Without dwelling on practical considerations about secure deployment of the platform itself, it is clear that each micro-service should include a specific sidecar for authentication and access control.

An Authentication, Authorization, and Accounting framework (*AAA*) is therefore necessary to mediate between multiple domains and protocols, especially in N–S communications. However, E–W transactions might contain 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 data plane must remain a prerogative of the security controller and trusted policies, so it is important to track the issuer of such commands.

Given the distributed nature of the system and impermanence of relationships, identity management will be based on a Public Key Infrastructure (PKI) and the availability of digital certificates for each entity. The presence of multiple and potentially untrustworthy Certification Authorities would demand for some reputation models based on recursive trust relationships, similarly to what already used in e-mail systems (i.e., PGP).

4.5. Available technologies and research directions

From a technical perspective, the key question is about the feasibility of the proposed architecture, while assuming the willingness of RPs to converge towards this kind of approach. Indeed, even if the technology to implement some components is still missing and more research is needed, a lot of tools are already available. Table 2 shows a list of existing tools and technological gaps. The analysis is far from being a thorough inventory; indeed, it is just conceived to give some concrete examples that demonstrate the feasibility of the proposed solution.

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). These tools implement both local Security Functions and Context Broker functions. The main technological gap is currently the substantial lack of programmability and the ability of remote control. Examples of programmable agents include the extended Berkeley Packet Filter (eBPF), a Linux framework for fast and lightweight packet filtering in kernel space, and Logstash, where custom pipelines can be created to monitor, inspect, and aggregate data. However, in general, other languages can also be used: ELF binaries, java bytecode, python scripts, P4 programs.

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

Common technologies for message buses and databases can be used to implement the Context Broker. 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 detection algorithms. 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. However, the abstraction of monitoring capabilities and current configurations is largely missing in existing frameworks, though their definition does not bring any critical challenge.

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 lookups 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.

Business rules management systems can be effectively used to implement the Security Controller, even if true intent frameworks that distill rules and actions from non-technical objectives

Table 2
Inventory of existing technologies and tools.

Element	Technologies and tools	Gaps
Security functions	Scribe, Flume, Heka, Elastic Beats, eBPF, Logstash, Prometheus, …	Only Logstash and eBPF programs can be developed. Other tools are configurable but not programmable.
Local Cntr & Management	Custom security agents	Most security functions are configured by local files but do not provide external APIs.
Context broker	Elastictree, Mongodb, Kafka, Logstash, Prometheus, …	Context models and full abstractions of the underlying security hooks are largely missing in all frameworks.
Security services	See Table 1	Most security functions are designed as standalone security appliances with no interfaces for orchestration and management.
Orchestration	Juju, Kubernetes, OpenStack Heat	–
Security controller	Drools, Apache Nifi	Mostly based on ECA patterns or rules. True intent frameworks are still missing.
Dispatcher	Kafka, RabbitMQ, Qpid, JBOSS messaging, etc.	–
Information sharing	–	Standard reporting formats. Automatic derivation of novel findings for CTI is an open research topic.
Security dashboard	Kibana, Prometheus	Informative content tailored to user roles is largely missing.
AAA	OAUTH 2.0, OpenID, Shibbolesh	Many protocols available for ABAC/ABE. Some federated identity management solutions available. Many frameworks are based on shared secrets (e.g., Kerberos, OpenStack Keystone) and not suitable for distributed systems.

are still an open research issue. Similarly, many protocols and tools are already available for AAA; here the main challenge is derivation of trust relationships when federation is not technically possible or administratively acceptable. Finally, graphical user interfaces are commonly available both for open source and commercial systems, and their extension to multiple users and tailored informative content should be rather straightforward.

Fig. 7 shows a preliminary software architecture for the SO-CaaS platform, where available technologies are composed together with novel components. Dark gray boxes indicate available software tools, while light gray boxes are representative of new concepts that are necessary to implement most functionalities related to abstraction of capabilities and orchestration. They are being designed and prototyped in on-going research projects.

4.6. Security considerations

One common concern raised by any cyber-security paradigm is its inherent security. Pursuing more automation in the management of security processes, our architecture removes the need for manual intervention, hence reducing the reaction delay and the risk for human errors. Nevertheless, any architectural or implementation vulnerability will represent a potential attack vector. In addition, automation is eventually driven by control and management policies defined by humans, and this should not give the false confidence that the system reacts in the correct way to every possible scenario and condition.

Some critical aspects of the overall framework can be pointed out already at this stage, and possible remediation actions and countermeasures can be identified. Though it is impossible to address every vulnerability at the architectural level, many issues can be solved during the implementation, deployment, or even operation phases.

4.6.1. Single point of vulnerability

The collection of security information in the centralized repository represents a potential vulnerability of the system, since all data can be accessed in case of successful penetration. Centralized collection is anyway considered a key feature of the framework, since the ephemeral nature of digital components does not guarantee the persistence of local logs for the whole lifespan of the service. This typically happens for elastic applications, where horizontal scaling can be used to provision and de-provision Virtual Machines according to the actual workload. In addition, most digital resources should be considered less robust than the centralized platform, which makes them preferred targets for attacks (as in case of IoT devices).

The persistence of the whole security context over time is required for forensics and offline investigation, when some components might not be available anymore or might have been got compromised. It is unquestionable that central data collection and security control represent a potential threat for the system, but it should also be pointed out that the platform is conceived to be operated in SOC by security specialists, so the risk for successful penetration is estimated to be much lower than for protected services.

4.6.2. Untrusted resources

The proposed architecture relies on security capabilities implemented by each digital resource for monitoring, inspection, and enforcement processes. This clearly poses the question whether such resources are safe or not, because eavesdropping, snooping, denial of service and other kinds of attacks can be easily performed internally. We propose to leverage APIs to discover what security capabilities are available, but there is no technical mean to guarantee their correct and reliable implementation; in addition, only a reduced set of capabilities might be available.

However, the possibility to identify the resource and its owner, version, implementation, release, and security capabilities paves the way for implementing risk assessment procedures to estimate the level of trustworthiness and reliability of each discrete service and the overall chain. Clearly, the availability of APIs does not guarantee their correct implementation and the safety of internal processes; in this respect, the adoption of common cybersecurity certification schemes is expected to provide assessments about the degree of adherence of products, services and processes against specific requirements.

4.6.3. Internet links

The distributed nature of digital services and externalization of security processes will require that all communication channels are implemented over the (unsafe) Internet. This represents a major threat, because of the number of attacks that can be successfully performed in such scenario: eavesdropping, spoofing, replication, delay, alteration, denial of service.

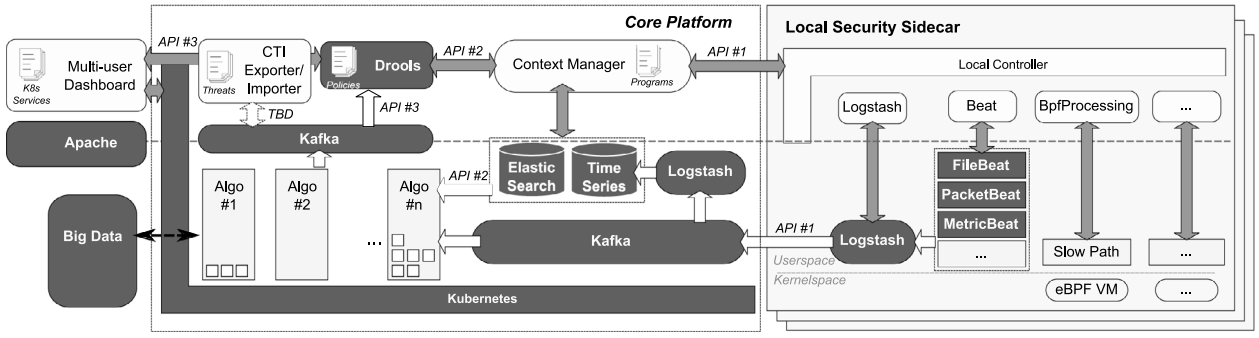


Fig. 7. Preliminary software architecture for the implementation of programmable SOCaaS platforms. Dark gray boxes are software tools already available from (mainly open-source) projects, whereas white and light gray boxes are novel components that are currently under design/prototyping.

This threat can be easily mitigated in the implementation, by creating encrypted links between all components. The usage of TLS/SSL, IPSec, or other encryption mechanisms at the application layer will ensure confidentiality, integrity and the authenticity of the source, hence addressing most of the Internet threats. However, they do not provide an effective solution against denial of service. As a matter of fact, heartbeating can only provide indication that the remote peer is no more reachable, or previous messages were not successfully delivered. In such scenarios, there is no option but to have a fallback local procedure that safeguard the service integrity. The Local Control and Management Framework is the architectural component in charge of establishing and maintaining connectivity with the centralized Context Broker. Hence, it can trigger a local fallback procedure. Possible actions may include buffering security-related data or, as last resort in case of extended disconnection, wiping any local data and software off the digital resource, to avoid potential disclosure.

4.6.4. Integrity of local security agents

Beyond trustworthiness in the Resource Provider, the reliability of the overall framework heavily relies on the integrity of local security agents. In case they do not work properly or send wrong information, the detection processes will be cheated. Beside missed detection, this could even be used to affect service operation (e.g., to stop it, to move it to a rogue infrastructure, to steer data, etc.). Since any system component is a potential target for unknown attacks, the integrity of security agents cannot be taken for granted after boot. In particular, even if the bytecode is not changed, unexpected behavior can be driven by misleading external triggers (like in case of SQL injection and similar attacks).

The availability of a Trusted Platform Module (TPM) allows to verify the integrity of the software during the boot procedure, but additional attestation procedures are needed at run-time to ensure the software does not deviate from the correct behavior. Static tools for source-code analysis and run-time verification are therefore expected to be used for guaranteeing the reliable execution of monitoring and enforcement processes. However, it is a known problem of code analysis that the number of program paths grows exponentially with the number of conditional branch statements. In real-world programs, this is known as path/state explosion, and it poses a significant challenge for analyzing big programs, where the number of paths is typically very large. For this reason, we advocate the adoption of lightweight monitoring and enforcement programs instead of complex cyber-security appliances. This would significantly reduce the path explosion issue. In addition, eBPF programs are explicitly suggested as enabling technology because they are executed in a controlled environment, are limited in size, and are guaranteed to be loop-free. Definitely, once the source of the program is known and trusted, there are no security issues in running its code.

4.6.5. Identity and key distribution

The reliability and trustworthiness of the communication between local agents and the Core platform is ensured by encrypted communications. However, secret material should be installed in each Local Security Sidecar to correctly perform authentication and key exchange. Since the two components are deployed in different administrative domains, the usage of digital certificates is probably the best option. However, secrets might be stolen during deployment or operation, hence opening the door for a number of attacks (spoofing, eavesdropping, modification, etc.).

Storing credentials and secret material both in the volatile or persistent storage represents a possible attack vector, since there are multiple vulnerable points where this information could be snooped (e.g., image repository, storage systems, VM migration). A possible solution is to rely on the TPM for storing and managing private keys, so that they cannot be read by any external process. This issue should be further considered in the definition of the Identity Management and Access Control component of the framework.

4.6.6. Unexpected system behavior

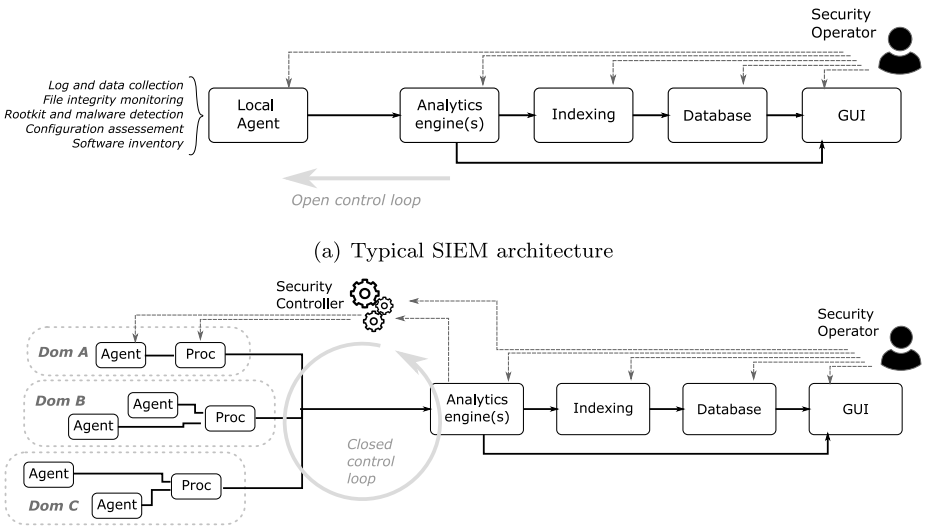
Automation is one major objective for the proposed framework. The simultaneous satisfaction of multiple inter-dependent policies provides coverage for a far larger set of scenarios, but this brings the risk for inappropriate, wrong, or harmful response for some scenarios.

Careful design of the Security Controller component is vital to guarantee reliable operation of the SOCaaS platform. However, since automatic behavior might not be always trusted, it is common practice to define multiple operational modes (fully automated, supervised, manual). For critical services, the supervised mode is the recommended choice, giving humans the possibility to review and approve the decision taken by the machine. For non-critical services, full automation will provide faster response and littler likelihood for propagation of attacks.

4.6.7. Disclosure of internal information

Including security properties and capabilities in external APIs exposes useful information to potential attackers. For instance, the description of internal software narrows the set of possible vulnerabilities and threats for each digital resource.

Access to these APIs should be restricted to certified SOCs. Similar to what already envisioned for Resource Providers, SOCs should also be subject to certification processes that attest their role and credibility. The Identity Management and Access Control framework should take this aspect into account, by applying access rules on the interface exposed by Local Security Sidecars that only select trusted security operators.



(a) Typical SIEM architecture

(b) Proposed architecture

Fig. 8. Comparison between a typical SIEM architecture and the proposed approach.

5. Comparison with existing architectures

The proposed architecture directly compares to SIEM systems, since it basically represents an evolution of their operation. A typical SIEM architecture looks like that of Fig. 8(a).[4] Usually, a local agent is deployed in each monitored system; it collects information from multiple subsystems (log files, configuration files, system activity, network statistics) and performs a certain set of active operations (e.g., checksumming of binary files). Once collected, data is sent to a centralized set of analytics engines, which filter relevant information, perform correlations and look for Indicators of Compromise (IoC). The output is therefore indexed and stored in a database, for additional off-line investigation, historical analysis, and forensics. All this information can be retrieved by a graphical interface for visual analytics and representation to humans.

Notably, though there are usually many configurable options at each stage, the whole pipeline shown in Fig. 8(a) is rather rigid, because agents have fixed capabilities and there is a predefined set of engines for analytics. If integration with a different monitoring subsystem is necessary, either a new version of the local agent must be deployed or a specific plugin must be installed. Unfortunately, this operation is rather unlikely to be available in third parties' infrastructures. If a new analytics service is required, it must be installed and tailored to the system, in terms of data format and interfaces. Therefore, existing architectures basically realize an "open control loop" solution, because there is no way to dynamically change the behavior of the system according to the evolving context.

The proposed architecture solves many of the above issues in light of emerging computing paradigms and business models. Indeed, the local processing capabilities (implemented by Logstash in the software architecture) can be used to make data normalization, aggregation, fusion, and transformation locally; moreover, the combination with Kafka allows to create multiple logical channels, each one intended to a different security service in the centralized platform. Our approach assumes local agents are available for each digital resource; however, alongside with traditional monitoring agents, our architecture also leverages eBPF as an effective mean to safely augment the code at run-time. Once proper authentication and control access mechanisms are in place, eBPF programs can be loaded at run-time, without affecting the stability of the system, in a much more effective and secure way than deploying new agents or plugins. In the centralized platform, again Logstash and Kafka can be used to "pipeline" multiple algorithms and perform the necessary adaptation between different interfaces.

Our architecture provides an abstraction entity (the Context Broker) which keeps the logical models of monitoring programs, security services, transformation and adaptation plugins. This abstraction includes the interfaces and data formats. With all this information, the smart logic of the Security Controller can be programmed to create multiple processing pipelines, which start from data generation, perform the necessary data transformation, feed one or more analytics services and finally store data in the common repository. This basically implements a "closed control loop" approach, which allows to dynamically bind heterogeneous local agents and the centralized set of analytics services on-demand, while the system is running and without long downtime.

6. Use case example

Though the range of the potential applications of our solution is very broad, we just limit to a very representative use case. Let us suppose a reasonable layout for a Smart City application, where data are collected from the environment by IoT devices and made available through a data broker. A database is also available as a service from an external provider which holds static information (e.g., timetables, tourist information, guides). A VM hosts some applications that combine data from IoT and the static database into a variety of services for citizens, which are available through a public web portal. This scenario includes a combination of physical and virtual resources, as shown in Fig. 9, likely coming from different administrative domains.

Let us focus on the problem of malware and covert channels. In this prospective scenario, malware can be mainly inserted through weak IoT devices and rogue or careless users. Once the malware is installed, it may exfiltrate sensitive and private data, or just become part of a Distributed Denial of Service attack.

[4] See for example, tools like Wazuh and Splunk.

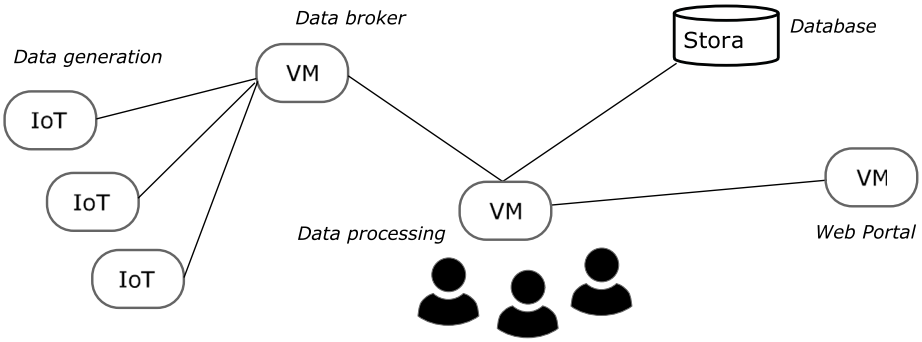


Fig. 9. A possible service chain for Smart City applications.

Covert channels are largely used in case low bandwidth is required, to remain unnoticed to detection tools. Given the large number of potential covert channels, there is not currently a single software which is able to detect all of them.

So, how the proposed architecture can be used to identify malware and hidden communications? Let us suppose the initial configuration of the platform is collecting system logs, file integrity measurements, network statistics, system events. This is not enough to detect malware, since there are developers who are continuously implementing new applications. However, once any anomaly or suspicious event is detected, a deeper analysis can be started, by looking for a known list of cover channels. To detect a cover channel, eBPF programs can be used to trace the system and look for specific "signatures". For example, file access, changing of file permission and ownership, uncommon values in packet headers, unexpected delays in packets, number and duration of tasks and processes, and so on. The number of techniques is so broad that full continuous monitoring will be highly inefficient and impractical. Instead, with the proposed architecture, we can investigate a large number of cover channels sequentially, by re-configuring the system: (a) a new BPF program is downloaded on suspicious resources to perform specific tracing of system execution or to measure statistics; (b) a Logstash plugin is installed locally to perform any data adaptation and to put data on a dedicate Kafka topic; (c) an analytics engine is started on the centralized platform that subscribes to the Kafka topic, processes incoming data, and looks for a specific covert channel. If the algorithm does not find anything in a pre-defined time, the corresponding agents, analytics engine, plugins, and configurations are removed and a new set up can be created for a different covert channel.

We point out that, as soon as new techniques are elaborated to discover additional covert channels, they can be simply loaded in the system, including monitoring programs, processing algorithms, and transformation plugins. They will be then dynamically instantiated by the Smart Controller according to the overall control policies.

7. Related work

The emergence of enterprise cloud environments and the related new services, applications and even heterogeneous devices they can support and interact with, poses a number of new issues in the design and operation of SOC architectures. Tafazzoli and Garakani present an interesting use case of SOC deployment in OpenStack [16]. Since the cloud service layers and multi-tenant architectures are not well-suited to use traditional monitoring and incident response methodologies, they argue that installing SOC directly on the cloud platforms can help detect ongoing incidents and attacks and reduce security risks as information theft, denial of service, information destruction, etc. Under similar considerations, Alruwaili et al. [17] propose a framework based on the as-a-Service paradigm, namely the Security Operation Center as a Service (SOCaaS), with the aim of collecting security events and logs from the cloud environment and use this information to provide security event management. The proposed framework leverages a multi-layer security model that is based on the correlation of events, along with advanced event analysis and according to cloud customer SLAs, policies and regulatory requirements.

Miloslavskaya considers the management of Information Security (IS) implemented for the field of the Internet of Things (IoT), arguing that it presents unique risks that require devoted assessments [18]. In this respect, the author provides a survey on the mission and main functions of existing SOCs and identifies the key indicators of IS incidents in IoT infrastructures to highlight current limitations.

Another aspect that has been often tackled in the design of SOCs regards structuring the architecture into several hierarchies that have been proposed with the goal of improving robustness, especially by removing single points of failure. Niu et al. [19] design a security operation center modeled on natural world immunology. In more details, their SOC detects intrusions by analyzing log information collected in network packets and dynamically generates so-called immunity cell agents every time an access via network occurs. Every time an agent detects an illegal intruder as non-self, it cooperates with other immunity agents and removes all files and processes identified as intrusion.

Li et al. [20] propose a hierarchical mobile agent security operation center (HMSOC) to overcome the traditional SOC vulnerability in a fixed location, which can be mainly identified as single point of failure vulnerability. The proposed SOC uses mobile agent groups to provide mobile agent characteristics, such as autonomy, mobility, intelligence and adaptability, etc. By designing a four-level distributed security management with a hierarchical architecture for distributed event correlation, they aim at enhancing intelligence, coordinated detection, and load reduction by the mobile agent technique, while improving the false positive ID ratio and removing single points of failure.

More autonomicity in analysis and detection, again boosted by the increased programmability of the underlying execution environment, is also required to detect polymorphic attacks and to investigate new threats. Although very detailed classifications and taxonomies of both attack and defense methodologies have been already identified [21,22], attacks continuously transform to circumvent detection rules in security appliances. Machine Learning (ML) methods promise significant advances in this field,

especially when combined with the multilevel correlation analysis among the attributes of the correct and malicious data [23, 24].

The concepts of orchestrating security agents and introducing more programmability in the design of cyber-security tools was originally proposed by the ASTRID project [5]. This project specifically addresses cloud and Network Functions Virtualization services, deployed and managed by software orchestration tools. The ASTRID objective is the separation of concerns between management and security processes, which allows externalization of the latter. Following the same principle, we extended the scope to more general scenarios, where digital resources are provided from heterogeneous computing models and administrative domains, and combined according to service-oriented and service mesh architectures.

8. Conclusion

The combination of digital resources from heterogeneous technological and business domains into business value chains points out all the limitations of existing cyber-security paradigms, which are still largely based on the combination of discrete cyber-security appliances. In this paper, we have explained the main motivations behind the trend to externalize security processes, and why existing SIEM tools lack the ability to follow the dynamics of elastic service-oriented architectures.

The implementation of SOCaaS requires more automation in connecting cyber-security platforms with ICT infrastructures, taking into consideration the presence of heterogeneous computing models, multiple administrative domains, and multi-tenancy. In this respect, the main contribution of this paper is a reference architecture which includes all functional elements that are necessary to address such requirements.

To demonstrate the feasibility of the proposed approach, our work has also identified a set of architectural components, and mapped them to existing technologies and tools. We have also outlined a preliminary software architecture, which selects complementary tools and identifies missing components. The purpose is to foster complementary research activities in a common framework, which serves as reference for comparison and prototyping. In this respect, the reference architecture has been already adopted by on-going research projects, which are implementing and integrating the whole framework in relevant use cases, for the purpose of validation and demonstration.

CRediT authorship contribution statement

Matteo Repetto: Conceptualization, Methodology, Writing - original draft, Supervision, Funding acquisition. **Alessandro Carrega:** Visualization, Resources. **Riccardo Rapuzzi:** Writing - review & editing, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has received funding from the European Commission (Grant Numbers: 786922 and 833456). The original concept and architecture was conceived by the ASTRID project, while extensions to digital services and service mesh architectures are being carried out by the GUARD project.

The authors would like to thank the following people that actively contributed to the definition of the overall vision, objectives, and challenges for next-generation cyber-security frameworks: Joanna Kolodziej (NASK), Markus Wurzenberger (AIT), Bartłomiej Kołodziejczyk (MC2), Giovanni Coppa (Wobcom), Stefano De Panfilis (FIWARE Foundation).

References

[1] NESSI, Strategic research and innovation agenda, 2017, [cited November 19th, 2018]. URL http://www.nessi-europe.com/files/NESSI_SRIA_2017_issue_1.pdf.

[2] NESSI, Cyber physical systems – opportunities and challenges for software, services, cloud and data, 2015, White Paper, [cited November 15th, 2018]. URL http://www.nessi-europe.eu/Files/Private/NESSI_CPS_White_Paper_issue_1.pdf.

[3] 5G-PPP, 5g innovations for new business opportunities, 2017, Whitepaper, [cited January 20th, 2019]. URL https://5g-ppp.eu/wp-content/uploads/2017/03/5GPPP-brochure-final-web-MWC.pdf.

[4] R. Rapuzzi, M. Repetto, Building situational awareness for network threats in fog/edge computing: Emerging paradigms beyond the security perimeter model, Future Gener. Comput. Syst. 85 (2018) 235–249, http://dx.doi.org/10.1016/j.future.2018.04.007.

[5] S. Covaci, R. Rapuzzi, M. Repetto, F. Risso, A new paradigm to address threats for virtualized services, in: IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, Japan, 2018, pp. 689–694, http://dx.doi.org/10.1109/COMPSAC.2018.10320.

[6] ECSO, European cybersecurity strategic research and innovation agenda (SRIA) for a contractual public-private partnership (cPPP), 2017, [cited December 9th, 2018]. URL https://www.ecs-org.eu/documents/publications/59e615c9dd8f1.pdf.

[7] S. Hares, D. Lopez, M. Zarny, C. Jacquenet, R. Kumar, J. Jeong, Interface to network security functions (I2NSF): Problem statement and use cases, 2017, IETF RFC 8192, URL https://www.rfc-editor.org/rfc/pdfrfc/rfc8192.txt.pdf.

[8] D. Lopez, E. Lopez, L. Dunbar, J. Strassner, R. Kumar, Framework for interface to network security functions, 2018, IETF RFC 8329, https://tools.ietf.org/pdf/rfc8329.

[9] C. S. Alliance, Security guidance for critical areas of focus in cloud computing, 2017, v4.0, https://cloudsecurityalliance.org/download/security-guidance-v4/.

[10] S. Dua, X. Du, Data Mining and Machine Learning in Cybersecurity, CRC Press, Boston, MA, USA, 2011.

[11] M. Panda, A. Abraham, 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, 2010, pp. 5–10, http://dx.doi.org/10.1109/ISIAS.2010.5604193.

[12] C. Wagner, G. Wagener, R. State, T. Engel, Malware analysis with graph kernels and support vector machines, in: 4th International Conference on Malicious and Unwanted Software (MALWARE), Montreal, QC, Canada, 2009, pp. 63–68, http://dx.doi.org/10.1109/MALWARE.2009.5403018.

[13] D. Montero, M. Yannuzzi, A.L. Shaw, L. Jacquin, A. Pastor, R. Serral-Gracià, A. Lioy, F. Risso, C. Basile, R. Sassu, M. Nemirovsky, F. Ciaccia, M. Georgiades, S. Charalambides, J. Kuusijärvi, F. Bosco, Virtualized security at the network edge: a user-centric approach, 53 (4) (2015) 176–186.

[14] 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, P. Olli, A collaborative cyber incident management system for European interconnected critical infrastructures, Elsevier J. Inf. Secur. Appl. (JISA) 34 (2) (2017) 166–182.

[15] F. Skopik, G. Settanni, R. Fiedler, A problem shared is a problem halved: A survey on the dimensions of collective cyber defense through security information sharing, Elsevier Comput. Secur. J. (2016).

[16] T. Tafazzoli, H.G. Garakani, Security operation center implementation on openstack, in: 8th International Symposium on Telecommunications (IST), Tehran, Iran, 2016, pp. 776–770.

[17] F. Alruwaili, T. Gulliver, Socaas: security operations center as a service for cloud computing environments, Int. J. Cloud Comput. Serv. Sci. 3 (2) (2014).

[18] N. Miloslavskaya, Security operations centers for information security incident management, in: IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), Vienna, Austria, 2016, pp. 131–138.

[19] Y. Niu, Q. Zhang, Q. Zheng, H. Peng, Security operation center based on immune system, in: IEEE International Conference on Computational Intelligence and Security Workshops (CISW), Heilongjiang, China, 2007, pp. 97–103.

[20] J. Li, C. Hsieh, Implementation of a distributed hierarchical security operation center using mobile agent group, in: IEEE International Symposium on Computer, Communication, Control and Automation (3CA), Tainan, Taiwan, 2010, pp. 79–82.
[21] N. Hoque, M.H. Bhuyan, R.C. Baishya, D.K. Bhattacharyya, J.K. Kalita, Network attacks: Taxonomy, tools and systems, J. Netw. Comput. Appl. 40 (2014) 307—324, http://dx.doi.org/10.1016/j.jnca.2013.08.001.
[22] I. Corona, G. Giacinto, F. Roli, Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues, Inform. Sci. 239 (2013) 201–225, http://dx.doi.org/10.1016/j.ins.2013.03.022.
[23] H.H. Pajouh, G.H. Dastghaibyfard, S. Hashemi, Two-tier network anomaly detection model: a machine learning approach, J. Intell. Inf. Syst. 48 (1) (2017) 61–74, http://dx.doi.org/10.1007/s10844-015-0388-x.
[24] M. Kruczkowski, E. Niewiadomska-Szynkiewicz, A. Kozakiewicz, Cross-layer analysis of malware datasets for malicious campaigns identification, in: International Conference on Military Communications and Information Systems (ICMCIS), Cracow, Poland, 2015, http://dx.doi.org/10.1109/ICMCIS.2015.7158682.

Matteo Repetto, Ph.D., received the Ph.D. degree in Electronics and Computer Science in 2004 from the University of Genoa. From 2004 to 2009 he was a postdoc at University of Genoa. From 2010 to 2019 he was a Research Associate at CNIT. In 2019 he joined the Institute for Applied Mathematics and Information Technologies (IMATI), CNR. He has been teaching many courses in telecommunication networks and network security. He has been involved in several research national and international projects on quality of service, mobility in data networks, energy efficiency, cloud computing, and network function virtualization. He is currently the scientific and technical coordinator of the ASTRID (www.astrid-project.eu) and GUARD (https://guard-project.eu/) projects, which investigate new security paradigms for cloud services. He has co-authored over 60 scientific publications in international journals and conference proceedings. His current research interests include pervasive communications and mobility management, energy-efficient networking, software-defined networking and network function virtualization, cloud/fog/edge computing and network security.

Email: matteo.repetto@cnr.it

Alessandro Carrega, Ph.D., is currently research associate at CNIT. He took part in the activities of many national and European projects and he is an active reviewer for many different international journals and conferences (IEEE and ACM). He has co-authored several papers in international conference proceedings. In 2010, he won the best paper award at the 3rd Int. Workshop on Green Communications (GreenCom 2010) co-located with the IEEE GLOBECOM Conference. His main research topics are on networking (energyaware, performance optimization, and virtualization), software routers, NFV and SDN (in particular, OpenFlow), network security. Finally, he is involved in collaboration with many industries, such as Telecom Italia, Broadcom, Nokia, Ericsson, Huawei, etc., and industrial fora, like GeSI.

Email: alessandro.carrega@cnit.it

Riccardo Rapuzzi, Ph.D., received his "laurea" degree cum laude in Computer Science in 2004 and he obtained the Ph.D. degree in Electronic, Computer Engineering and Telecommunications in 2009 from University of Genoa. Since 2009, he has been holder of research grants at the Department of Communication, Computer and System Sciences (DIST) and, starting from 2012, at the newly constituted Department of Naval, Electrical, Electronics and Telecommunication Engineering (DITEN) of the University of Genoa. He is now with the National Inter-University Consortium for Telecommunications (CNIT). Since 2006, he has been involved in both Italian and European research projects. He is co-author of several research papers, which have been published in proceedings of international conferences or international journals. His main research interests include the Linux software router platforms, the Internet traffic classification and characterization, with particular focus on peer-to-peer applications, the IP mobility issues in wireless networks and pervasive environments, the energy efficiency techniques for green networking, and security in distributed environments.

Email: riccardo.rapuzzi@cnit.it