Situational Awareness for Virtualized Services: The ASTRID Approach

Situational Awareness in Virtual Networks: the ASTRID Approach

A. Carrega, M. Repetto
S^{3}ITI Lab – CNIT, Italy
Email: matteo.repetto@cnit.it

F. Risso
Politecnico di Torino, Italy
Email: fulvio.risso@polito.it

S. Covaci
Technical University of Berlin, Germany
Email: stefan.covaci@tu-berlin.de

A. Zafeiropoulos
Ubitech Ltd, Greece
Email: azafeiropoulos@ubitech.eu

T. Giannetsos
University of Surrey, UK
Email: a.giannetsos@surrey.ac.uk

O. Toscano
Ericsson Telecomunicazioni, Italy
Email: orazio.toscano@ericsson.com

***Abstract*—Cloud-based services often follow the same logical structure of private networks. The lack of physical boundaries and the dependence on third party's infrastructural security mechanisms often undermine the confidence in the overall security level of virtualized applications. Integrating software instances of common security middleboxes into cloud networks helps overcome most suspicions, but leads to inefficient solutions.**

In this paper, we describe the vision behind the ASTRID project. The novelty of our concept lies in decoupling detection algorithms from monitoring and inspection tasks, seeking better integration with virtualization frameworks. We briefly elaborate on the overall conceptual architecture and the foundation of its implementation components. Additionally, we give insights on the expected impacts and opportunities brought by this novel paradigm over the existing approaches.

I. Introduction

Many ICT processes are designed with modular architectures, assuming internal safe communication. Network virtualization is a key enabler to move such applications to the cloud, but it poses a number of additional security concerns, when compared to current legacy deployments [1].

While ICT virtualization technologies have rapidly and considerably evolved during the last decade, security has not advanced at the same pace. Basically, cloud users can easily create virtual layer-2 and layer-3 network topologies that behave like their physical counterpart, but the strength and reliability of isolation is not comparable. Cloud management software provides isolated sandboxes through hypervisors, namespaces, and overlay networks, but in practice the virtualization infrastructure (hypervisors, shared networks, management software) augments the attack surface and represents a privileged attack vector, totally transparent to tenants' software. Building on this consideration, common firewalling functions have been made available in a distributed form, so to effectively counter any injection of malicious traffic both in the hypervisor or shared networks.

Motivated by substantial limitations of security mechanisms in the virtualization infrastructure, especially in multi-cloud deployments, we introduce a novel security model for virtual services deployed in cloud networks, which combines efficiency of monitoring with effectiveness of detection. Our approach applies the same concept behind software-defined networking to security, by pursuing a separation between the data plane (inspection) and the control plane (detection), mediated by an orchestration logic. In our architecture, the programmable data plane is efficiently carved out in different forms of virtualization environments (i.e., VM, LXC or Docker container), while the control plane is left aside of the service graph and include several complementary detection algorithms; orchestration dynamically configures inspection tasks according to the evolving needs of the detection logic.

In this paper, we describe the concept, architecture and framework behind our approach, including requirements, technologies, and research directions that we are pragmatically developing in the ASTRID project[1]. Since we are targeting a very disruptive approach with respect to current practice, we believe it is worth sharing early ideas and getting feedback, even if concrete implementation and numerical results are not yet available at this stage.

The rest of the paper is organized as follows. Section II gives a concise overview of current trends for enforcing security in cloud networks. We describe the novel concept of ASTRID in Section III, together with a brief digression on software orchestration. We outline the proposed framework in Section IV, by outlining our main objectives and methodology. Finally, we give insights on expected impacts of our work in Section V.

II. Beyond the security perimeter model

Distributed firewalls integrate packet inspection and filtering in hypervisors, overcoming the legacy model of security perimeter, and moving towards micro-segmentation and capillary monitoring and enforcing. A distributed firewall removes the need for traffic steering (all network packets go through the hypervisor, which is part of the firewall) and enables very fine-grained control over security policies, beyond mere IP-based rule structures (through the notion of logical "containers" or "security groups", e.g., vCloud Directory and OpenStack Neutron). Despite their common usage in cloud networking, distributed firewalls have some important limitations. First, this

[1]ASTRID (AddreSsing ThReats for virtualIseD services)

approach is currently effective for enforcing filtering rules, but does not have the flexibility to provide deep inspection capability tailored to the specific needs for detecting threats and on-going attacks. Second, they cannot provide the same guarantees of private enterprise networks: external resources lie in third parties infrastructures where trust mechanisms are still missing (i.e., the behavior of physical hardware and networks cannot be controlled by cloud users). Third, their application in multi- and cross-cloud environments is not straightforward, since their configuration is based on internal communication mechanisms for each infrastructure. This issue will be even more severe in cyber-physical systems [2], with the integration of smart things in cloud applications, which are expected to be a consistent use case for 5G.

Given the reduced set of security features integrated in virtualization platforms and the increasing needs for cross-cloud deployments, users are generally left most of the burden for protecting their applications against external threats. Since, on first approximation, virtualization environments could be viewed as special instances of physical networks, software-based versions of security middleboxes may be integrated in service graph design [3], [4]. We argue that this approach comes with important limitations in the current cyber-security landscape:

- *performance*: the ever growing number and complexity of protocols and applications complicate the detection of threats and anomalies, evolving inspection from memory-less simple string matching to stateful rules (such as regular expressions), hence more processing power is required, which is likely to overwhelm software-based implementation of load balancers, firewalls, and intrusion prevention systems, especially in case of large volumetric attacks;
- *context-awareness*: the nature and composition of multi-vector attacks requires pervasive monitoring and global view, and the deployment of Security Information Event and Management (SIEM) software for effective detection, which may be too cumbersome and ineffective for small applications and services;
- *attack surface*: virtual security appliances are more exposed to attacks than their physical counterpart, since they run in the same virtualization environment to protect;
- *propagation of vulnerabilities*: the growing trend to re-use software, often distributed as pre-package images, for multiple applications brings the risk of propagating software, architectural, and configuration vulnerabilities to many applications running in different infrastructures, which can become very dangerous botnets.

III. A NOVEL SECURITY CONCEPT FOR CLOUD NETWORKING

To address the limitations and flaws of current practice, ASTRID pursues a novel approach, based on the disaggregation of cyber-security appliances into business logic (i.e., detection algorithms) and data plane (i.e., monitoring and inspection tasks), mediated by the orchestration logic through proper security policies and configurations, as shown in Fig. 1. Instead of overloading the execution environment with complex and sophisticated threat detection capabilities, ASTRID will only perform lightweight monitoring and inspection tasks in service graphs and their execution environments, which feed detection algorithms placed outside the graph design, as part of a powerful and overarching awareness logic.

In what follows, Fig. 2 depicts the reference architecture behind the main concept outlined above.

A. The Data Plane

The Data Plane is represented by multiple *programmable security hooks*, which are present in the virtualization environment. The hooks include logging and event reporting capability developed by programmers into their software, as well as monitoring frameworks built in the kernel and system libraries that inspect network traffic and system calls. Simpler hooks may limit to data reporting, but many of them should also include processing capabilities, to reduce the amount of network traffic generated. Security hooks are 'programmable' because they can be configured at run-time, hence shaping the system behavior according to the evolving context. This means that packet filters, types and frequency of event reporting, and verbosity of logging are selectively and locally adjusted to retrieve the exact amount of knowledge, without overwhelming the whole system with unnecessary information. The purpose is to get more details for critical or vulnerable components when anomalies are detected that may indicate an attack, or when a warning is issued by cyber-security teams about new threats and vulnerabilities just discovered. This approach allows lightweight operation with low overhead when the risk is low, even with parallel discovery and mitigation, while switching to deeper inspection and larger event correlation in case of anomalies and suspicious activities, hence, being able to properly scale with the system complexity, even for the largest services (e.g., carriers large scale virtual networks, and worldwide mass applications as social nets).

B. Orchestration

Elasticity and agility brought by the different cloud models have pushed the transition from legacy script-based automation tools (e.g., Ansible, Puppet, Chef) to smarter orchestration platforms, which aims at filling the gap between software development, deployment and operation. Emerging devops and orchestration tools are increasingly empowering software developers to automate the development process by common and standardized templates that make use of third-party software. They leverage novel development paradigms, building on modular architectures based on "chains" or "graphs" of software components and descriptive metadata (*service models*) , which are suitable for policy-based cloud deployment and life-cycle management [5]. This effort has resulted in several models and architectures for both cloud applications (e.g., TOSCA [6]) and network function virtualization (e.g., ETSI MANO [7], and IEEE SFC [8]), which have been adopted

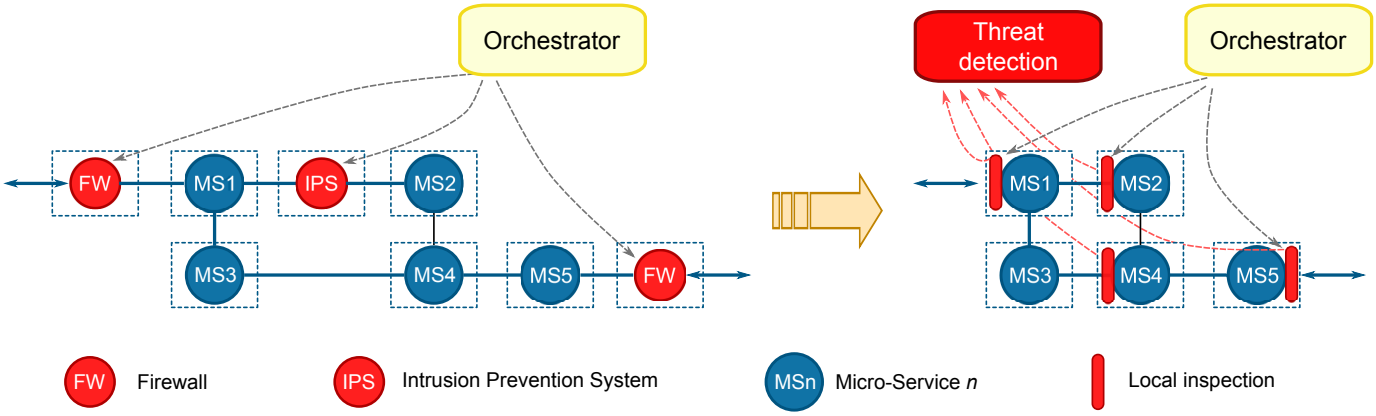

Fig. 1: The ASTRID concept entails a transition from the usage of virtual instances of security appliances in service graphs (left side) to external detection logic fed by programmable security hooks mediated by the orchestration logic (right side).

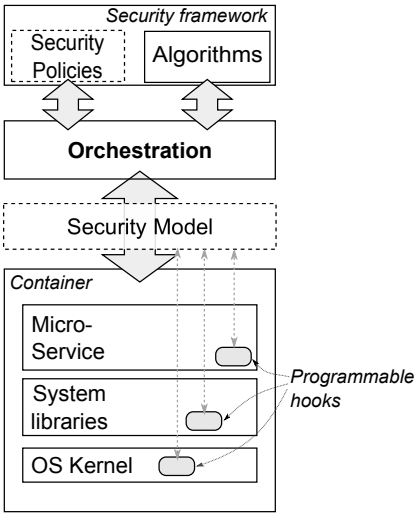

Fig. 2: Reference architecture for the ASTRID concept: decoupling centralized detection logic from local inspection tasks.

and extended in many orchestration tools and platforms (e.g., Juju[2], Cloudiator[3], and OpenBaton[4], just to mention a few).

Orchestration is the smart process that deploys and manages services according to specific requirements and life-cycle policies. Specific tasks for orchestration usually include: provisioning of virtual resources, selection of software instances that match the graph model, (re-)configuration of the execution environment and the hosted virtual functions, monitoring and collection of measurements. In this case, orchestration must be extended to understand security policies and to translate them in configurations of the security hooks, including monitoring/inspection tasks.

The *Security Model* provides a common abstraction for the underlying programmable hooks. It uses specific semantics to describe security-related capabilities (e.g., logging, event reporting, filtering, deep packet inspection, system call interception), and provides a common interface for their configuration.

Policies describe in an abstract form various life-cycle management actions; for instance, types of events that should be collected, anomalies that should be reported, actions that should be undertaken upon detection of potential attacks, etc. Policies may be encoded in high-level descriptive languages (e.g., XML, JSON) for requesting specific orchestration services (e.g., setting a packet filter for a given traffic flow, replacing a buggy or misbehaving function, trigger packet or software inspection). They are agnostic of the underlying data planes, so that they can be used with different (even heterogeneous) programming technologies.

C. Algorithms

Finally, Algorithms analyze and correlate information provided by the orchestration at graph level to detect threats, anomalies, vulnerabilities, attacks. The goal is to cluster typical functions (currently available) as separate appliances: Intrusion Prevention/Detection Systems (IPS/IDS), Network Access Control (NAC), Antivirus, Application Level Gateways (ALG), and more. One of the main advantages is the availability of data from different subsystems (disk, network, memory, I/O), instead of relying on a single source of information (network traffic) as is the common practice nowadays.

IV. The ASTRID framework

To implement the novel concept outlined in Section III, ASTRID has already devised an overarching framework comprising the following macro-blocks (see Fig. 3):

- *service engineering*, concerning the development and modeling of software components and service graphs;
- *service management*, dealing with secure deployment and life-cycle management of service graphs;
- *situational awareness*, responsible for detecting threats and certifying data for security audits and court investigations.

[2]https://jujucharms.com.
[3]http://cloudiator.org.
[4]https://openbaton.github.io.

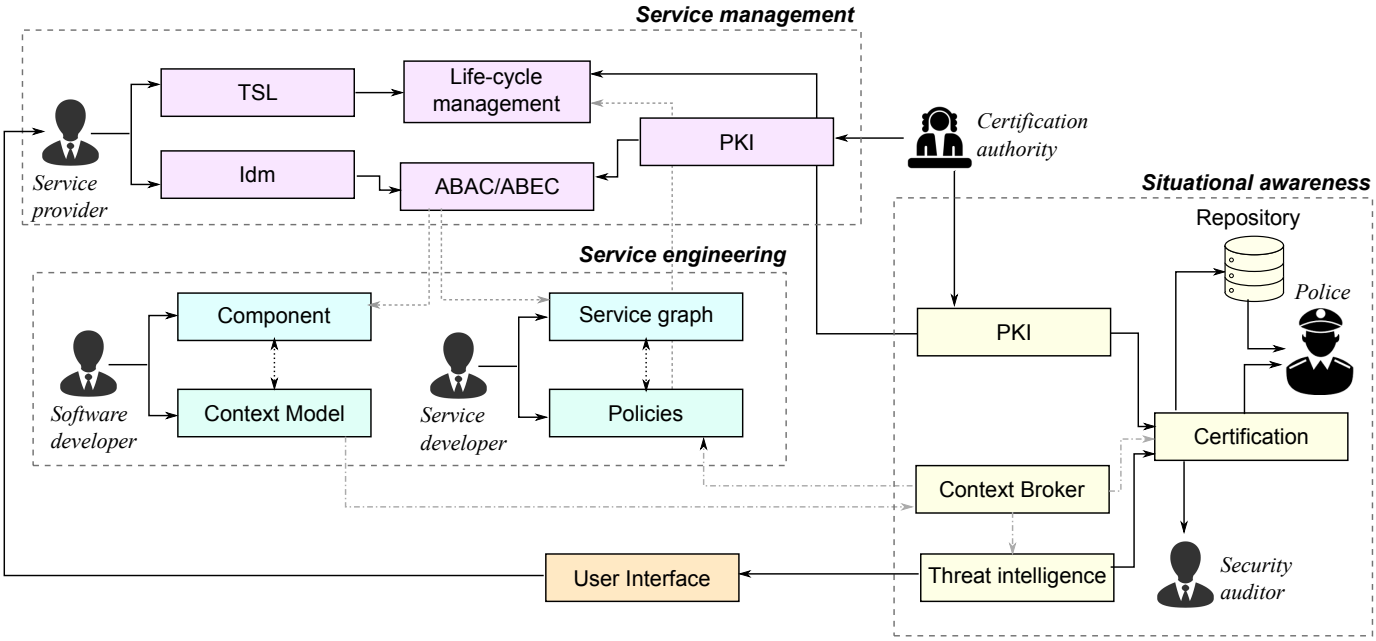

Fig. 3: The ASTRID framework.

A. Service engineering

Software development will be based on existing orchestration platforms that enable graphical design, development, and deployment of distributed applications over multiple clouds. The design will be based on descriptive metamodels, compliant with TOSCA and ETSI MANO, that already include a rich context model, describing deployment constraints and requirements. ASTRID will enrich the existing models, by accounting for enhanced security-related capabilities:

- *awareness*, which describes the types of logs, events, and behavioral/traffic monitoring that can be generated or consumed by the service, and provides the necessary hooks for configuring and controlling the generation of such data at the virtual function level. Data sources can be classified in *static* and *dynamic* ones. Static sources consist of structured resources that are provided manually, while dynamic sources consist of information that flows from a systemic endpoint;
- *trust and certification*, which are used for generating and conserving data (events, logs, measurements) with legal validity for forensics investigation;
- *privacy and encryption*, to set up confidential channels for communication;
- *inspection*, which provides technical means for legal interception of network traffic (and possibly other events) exchanged between the service components.

One of the most challenging tasks for ASTRID will be the definition of performance-optimized data plane components, which capture and distill events and knowledge with minimal performance impact.

Programmability is the basic requirement to implement an effective data plane for collecting security-related events and information, well beyond the basic monitoring capability envisioned by today's flow collectors like NetFlow, sFlow, IPFIX, and, more recently, OpenFlow and NetConf. The objective is to include stateless and/or stateful inspection criteria on flows and/or packets, kernel-level system calls, disk I/O, and more, hence, offering a broader and more precise coverage of what happens in the system under control. ASTRID will leverage the IOVisor[5] technology, which offers a wider range of options, including in-kernel eXpress Data Paths (XDP), enhanced Berkeley Packet Filters (eBPF), and inspection of system calls issued within the system.

Current IOVisor technology has been validated mostly with monitoring applications, hence, with limited (or no) capabilities to perform more effective actions (e.g., data modification/manipulation) on the incoming data. Furthermore, only simple data plane programs are allowed, i.e., without support for complex programs created according to the split data/control plane paradigm as originally proposed with SDN/OpenFlow. ASTRID will extend this technology, (i) to support more powerful programs, which can operate according to the split data/control plane paradigm; (ii) to support more powerful actions on the data in transit, which enable to implement some proactive security actions (e.g., drop network traffic, modify packet information, craft ad-hoc packets for specific purposes) that go beyond simple monitoring.

B. Service management

Service management entails various life-cycle operations on service graphs, in addition to policies for automated tasks. Specific tasks carried out by the orchestration process include: provisioning of virtual resources, selection of software

[5]https://www.iovisor.org.

instances that match the graph model, (re-)configuration of the execution environment and the micro-services, monitoring and collection of measurements. True and effective integration of security in service orchestration requires a new approach that addresses at least the following needs:

- checking the trustworthiness of software and service graphs at deployment (*design*) time;
- adapting the service graph to the evolving security context during *run-time*; e.g., replace compromised or vulnerable components with equivalent (but better) ones, inject new functions or disable existing ones;
- triggering software security analysis at *run-time*, either periodically or after suspicious events;
- translating policies and high-level instructions into proper code and configurations at the container/micro-service level; e.g., setting/changing firewalling rules, deep packet inspection, forwarding and routing policies;
- ensuring, though formal models and methods, the correct implementation of security policies.

Formal verification of service graphs before deployment is already included in orchestration platforms under consideration for the project; the objective is to include security verification as well, to ensure that orchestration actions do not break any security constraints. For instance, deployment and placement of service graphs must take into account selection of software images from trusted developers, selection of trusted/secure infrastructures, usage of encryption/integrity algorithms for network traffic.

Semi-autonomous operation of the orchestration process is usually driven by policies. These are sets of operating rules, usually in the form 'on event, if condition, then action', that describe life-cycle management operations, so to shape the system behavior to the evolving context without altering the system implementation. ASTRID will develop security policies, including but not limited to:

- *re-configuration* of individual components and programming of their virtualization environments, to change the reporting behavior, including parameters that are characteristics of each app (logs, events), network traffic, system calls (e.g., disk read/write, memory allocation/deallocation), RPC toward remote applications (e.g., remote DB); programming also include the capability to offload lightweight aggregation and processing tasks to each virtual environment, hence reducing bandwidth requirements and latency;
- *composition* of trusted business chains, by including suitable mechanisms to interact with external services and middleware, in order to guarantee privacy and certification of data origin; in this case, management is mediated by protocols such as Attribute-Based Access Control primitives and Attribute-Based Encryption Control primitives, while authentication relies on an Identity Management component and a Public Key Infrastructure (rooted at a trusted and public Certification Authority);
- *re-action* to threats and attacks (by filtering and/or dropping packets, isolating compromised resources, etc.) when triggered by detection algorithms;

The ASTRID framework also envisions authentication and encrypted channels for interacting with the service components. Thus, Secure deployment also entails the selection of trusted services, hence, a TSL (Trusted Service List) component is present.

C. Situational awareness

Situational awareness represents a totally new functional block in frameworks for developing and deploying cloud applications. It includes all the components to collect and process security-related data, and to provide knowledge and evidence about cyber-security threats, vulnerabilities, and attacks.

The *context broker* collects contextual information, likely by a Pub/Sub or similar paradigm, and feeds other engines that process and store such information. Many sources of information may be integrated into the platform, coming from service graph components and from the data plane; in principle, they have diverse format, frequency, and extraction type (i.e. pull or push). The lack of semantic alignment should be overcome by the creation of a common data representation meta-model.

Threat intelligence is a collection of detection algorithms that analyzes events, data, and logs, arguably by combining innovative detection methodologies (rules-based, machine learning) with big data techniques; the purpose is to locate vulnerabilities in the graph and its components, to identify possible threats, and to timely detect on-going attacks.

We have to highlight that, since orchestration manages all service graphs of the same service provider, threat intelligence combines and correlates contextual information from different graphs, which further improves threat detection and brings the possibility to fix vulnerabilities in advance before other components get compromised. Obviously, a larger base of data and events would increase the processing burden, but this should not be cumbersome, since the control plane is outside the service graph and could run in dedicated infrastructures with big data techniques.

ASTRID targets protection from both software vulnerabilities and network threats, hence, it involves a mix of source and run-time code analysis, formal verification, network analytics, and packet filtering techniques. It will assemble a diverse array of vulnerability analysis techniques to facilitate the transition of the application development industry to new security paradigms. This adaptive approach allows to fully embrace all the advantages of micro-service deployment and to keep up with an ever-evolving threat landscape.

ASTRID will develop two algorithms, as practical examples of usage of the framework. One algorithm will build on fine grained monitoring and inspection capabilities for volume anomaly detection that processes raw flow information (e.g., [9]). The other algorithm will deal with static and dynamic analysis of code to protect virtual services during their life-cycle. Even though there are many static analyser tools [10]–[12], there is a clear lack of code assessment mechanisms

that targets specifically cloud applications and virtual network functions. For dynamic analysis, the challenge is to develop hybrid vulnerability analysis tools that leverage packet fuzzing, packet sniffing and selective concolic execution (some of the most prominent vulnerability assessment techniques) in a complementary manner [13]–[15], to find deeper (possible) bugs during the execution of the services and their components. By combining their strengths and mitigating their weaknesses, we manage to avoid the path explosion inherent in concolic analysis and the incompleteness of fuzzing.

ASTRID will also tackle the critical issue of the legal validity of the extracted data to prosecute attackers, in case the graphs may be occasionally compromised. Common challenges in this area include: *i)* storing trusted evidence, *ii)* respecting user privacy when acquiring and managing evidence, *iii)* preserving the chain of custody of the evidence. We highlight that in the proposed framework the problem is not the same as the definition of *Cloud forensics* [16], [17], since investigation in our case is carried out by the service owner and not by the cloud provider. In the ASTRID framework, the *certification process* is responsible for origin, timestamping, digital signing, integrity of relevant information that is used for security audits and legal interception; the solution should be able to capture enough information to trace security attacks in a reliable manner and to interpret the data post-factum. In addtion, the *secure repository* conserves data with legal validity (security audit trails) for forensics investigation that is initiated after the threat or attack has been identified.

Finally, the *user interface* provides proper visual representation of the current situation to the service provider enabling decision making for remediation actions and countermeasures.

V. Conclusions

ASTRID pursues a novel approach for managing security of service graphs, beyond the mere deployment of virtual instances of legacy security appliances. Decoupling the data plane from the detection logic is expected to bring important impacts in security of virtual services and systems:

- reduced attack surface and increased efficiency for virtual applications, by pulling security software out of service graphs while relying on lightweight and effective data plane technologies;
- increased agility and elasticity in service graph design and deployment, since the graph topology is no more overwhelmed by legacy security appliances;
- increased immunity of the detection logic to attacks and better detection capability, since algorithms are clustered together and share information;
- wider and uniform situational awareness, by correlating events and information from multiple service graphs.

For the latter, we expect that higher efficiency and wider awareness will prove to be critical factors for increasing the security of national and EU critical infrastructure, which will heavily rely on virtualization and cloud technologies after the massive deployment of 5G networks.

ASTRID will demonstrate the developed technology and its applications within two envisaged scenarios, involving secure VoIP communication and remote collection of medical data.

Acknowledgment

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

References

[1] G. Pék, L. Buttyán, and B. Bencsáth, "A survey of security issues in hardware virtualization," *ACM Computing Surveys*, vol. 45, no. 3, pp. 40:1–40:34, June 2013.

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

[3] T. Quang Thanh, S. Covaci, T. Magedanz, P. Gouvas, and A. Zafeiropoulos, "Embedding security and privacy into the development and operation of cloud applications and services," in *17th International Telecommunications Network Strategy and Planning Symposium*, Montreal, QC – Canada, Sep. 26th–28th, 2016, pp. 31–36.

[4] 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, and F. Bosco, "Virtualized security at the network edge: a user-centric approach," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 176–186, April 2015.

[5] J. Wettinger, U. Breitenbücher, and F. Leymann, "Standards-based DevOps automation and integration using TOSCA," in *IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC)*, London, UK, Dec. 8–11, 2014, pp. 59–68.

[6] "Topology and orchestration specification for cloud applications," OASIS Standard, November 2013, version 1.0. [Online]. Available: http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.pdf

[7] "Network functions virtualisation," ETSI ISG NFV, October 2014. [Online]. Available: http://portal.etsi.org/NFV/NFV_White_Paper3.pdf

[8] J. Halpern and C. Pignataro, "Service function chaining (SFC) architecture," RFC 7665, October 2015. [Online]. Available: https://tools.ietf.org/rfc/rfc7665.txt

[9] H. Kasai, W. Kellerer, and M. Kleinsteuber, "Network volume anomaly detection and identification in large-scale networks based on online time-structured traffic tensor tracking," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 636–650, September 2016.

[10] G. Chatzieleftheriou, A. Chatzopoulos, and P. Katsaros, *Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications*. Springer, 2014, vol. 8803, pp. 486–488.

[11] L. Xin and C. Wandong, "A program vulnerabilities detection frame by static code analysis and model checking," in *IEEE 3rd International Conference on Communication Software and Networks (ICCSN)*, Xi'an, China, May, 27th–29th, 2011, pp. 130–134.

[12] A. Armando, G. Bocci, G. Chiarelli, G. Costa, R. De Maglie, G. Mammoliti, and M. Alessio, *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2015, vol. 9035, ch. SAM: The Static Analysis Module of the MAVERIC Mobile App Security Verification Platform, pp. 225–230.

[13] I. Haller, A. Slowinska, M. Neugschwandtner, and H. Bos, "Dowsing for overflows: A guided fuzzer to find buffer boundary violations," in *USENIX Security Symposium*, USA, 2013, pp. 49–64.

[14] V. Chipounov, V. Kuznetsov, and G. Candea, "S2e: A platform for in-vivo multi-path analysis of software systems," in *Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems (ASPLOS XVI)*, Newport Beach, California – USA, Mar., 5th–11th, 2011, pp. 265–278.

[15] S. K. Cha, T. Avgerinos, A. Rebert, and D. Brumley, "Unleashing mayhem on binary code," in *IEEE Symposium on Security and Privacy*, San Francisco, California – USA, May, 20th-25th, 2012, pp. 380–394.

[16] K. Ruan, J. Carthy, T. Kechadi, and M. Crosbie, "Cloud forensics: An overview," in *7th IFIP Int. Conf. on Digital Forensics*, USA, 2011.

[17] D. Barrett and G. Kipper, *Virtualization and Forensics – A Digital Forensic Investigators Guide to Virtual Environments*. Elsevier, 2010.