A platform for security monitoring of multi-cloud applications

Pamela Carvallo[1 2], Ana R. Cavalli[1 2] and Wissam Mallouli[2]

[1] SAMOVAR, Télécom SudParis, CNRS, Université Paris-Saclay, Évry, France
[2] Montimage EURL, Paris, France
{pamela.carvallo, ana.cavalli, wissam.mallouli}@montimage.com

Abstract. This paper presents a security assurance platform to monitor and control the security in the context of multi-cloud applications. Indeed, this property is a crucial issue in multi cloud-based environments where many aspects need to be faced, including risk management, data privacy and isolation, security-by-design applications, and vulnerability scans. Moreover, it also becomes necessary to have an efficient system that interrelates and operates all security controls that are configured and executed independently on each component of the system.
In addition, as new attacks emerge every day, threat detection systems play a fundamental role in security monitoring schemes, identifying possible attacks. These systems handle an enormous volume of data, as they detect unknown malware by monitoring different activities from different points of observation, as well as adapting to new attack strategies and considering techniques to detect malicious behaviors and react accordingly.
In this paper, we describe a monitoring platform for securing multi-cloud applications, from a Service Level Agreement perspective. Moreover, we present a case study depicting the multi-cloud monitoring of a smart-city transport application for the citizens of Tampere, Finland. Considering the nature of the application under study, the service requires continuous execution and availability functionalities, as end-users may utilize the service at any time.

Keywords: Cloud Computing; Security monitoring; Service Level Agreement; Threat Detection; Reaction.

1 Introduction

Monitoring is a solution that is required to control the correct operation of the whole system running in a multi-cloud environment. According to the taxonomy proposed by [14] and [13], the term multi-cloud denotes situations where a consumer (human or service) uses multiple, independent clouds, unlike to Cloud Federations that are achieved when a set of cloud providers voluntarily interconnect their infrastructures to allow sharing of resources among them. According to the state of the art, few concrete multi-cloud solutions exist, topics addressed

in research projects like MUSA, OPTIMIS, mOSAIC, MODAClouds, PaaSAge and Cloud4SOA [6, 12]. It is out of the scope of this paper to offer a complete survey of such activities. We suggest the interested reader the following works: [14, 5, 20].

Malfunctioning or even minor problems in a Virtual Machine (VM) could introduce vulnerabilities and stability issues to other VMs, as well as threaten the integrity of the host machine. In this paper, the monitoring function is needed to be able to precisely understand what is happening at network, system and application levels, pursuing a twofold objective. First, a proper monitoring system that improves the security in the communications and services offered by the multi-cloud virtual environments. Second, from the administration and management's point of view, a system that helps to ensure the environment's health, guarantee that the system works as expected and respects its Security Service Level Agreements (SecSLAs) [8].

In this context, we present a platform for monitoring multi-cloud based applications where each application component can be deployed in a different Cloud Service Provider (CSP). The proposed platform architecture raises several challenges when fulfilling an end-to-end security monitoring of the application execution and communication at runtime. These efforts are due to the complexity of cloud platforms that may consist of multiple layers and service paradigms (SaaS, PaaS, IaaS) and therefore need a flexible monitoring management in a distributed scheme.

To the best of our knowledge, no security monitoring solution has been designed for such multi-cloud distributed systems. Consequently, the main contribution of this paper is the design and deployment of a security assurance platform that gives an answer to these challenges along with preliminary results of a smart-city case study that provides efficient and optimal transportation to the half-a-million citizens of Tampere, Finland. This paper extends our work in progress paper [7], by presenting results of the performed experiments.

The paper is organized as follows. In Section 2, we present an overview of the multi-cloud security assurance platform and describe each of its modules. Section 3 presents the workflow for an use-case in this platform. Section 4 presents the related work on monitoring tools and threat detection systems. Section 5 gives some elements for discussion of the exposed work and presents the conclusion and future work.

2 The MUSA security assurance platform SaaS

The MUSA Security Assurance Platform (MSAP) is part of the MUSA project framework, and is offered following the Software-as-a-Service (SaaS) model to the Cloud Service Client (CSC). The MSAP ensures the security of the whole application distributed across heterogeneous cloud providers. This platform integrates and offers the MUSA monitoring service, the MUSA enforcement support service and the MUSA notification service, all of them working together with embedded security libraries. The monitoring service aims at evaluating the se-

curity and functional measures gathered by the use of multiple mechanisms such as standard APIs offered by the cloud provider or the security libraries. Furthermore, the monitoring service is able to trigger security alerts based on the event rules defined by the *Application DevOps team* (Fig.1) following a SLA perspective. The notification service is in charge of sending the alerts to the CSC when relevant security incidents have been detected, so the *Application DevOps team* can react and adapt the application or the provisioned cloud resources if needed. At the same time, the enforcement support service collaborates with the MUSA security libraries to enforce the security protection of multi-cloud application components.

2.1 The MUSA framework

The MUSA Framework relies on the MUSA H2020 project [1]. The main goal of this framework is to support the security-intelligent life-cycle management of distributed applications over heterogeneous cloud resources, through a security framework that includes: a) security-by-design mechanisms to allow application self-protection at runtime, and b) a reactive security approach, monitoring application at runtime to mitigate security incidents, so multi-cloud application providers can be informed and react to them without losing end-user trust in the multi-cloud application.

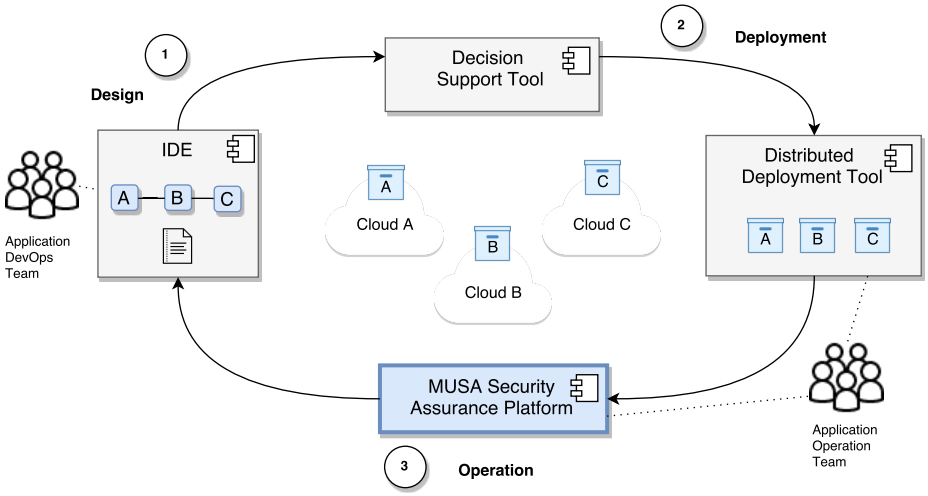


Fig. 1. MUSA Framework workflow

The MUSA framework workflow is depicted in Fig.1. in three phases, where each element of the global architecture of the system is presented. The workflow begins when a CSC's *Application DevOps team* uses the *IDE* module to specify and design the multi-cloud application based on modeling techniques, taking

into account (a) security requirements such as security embedded libraries for security at runtime, as well as (b) functional and business needs by delivering a composition of SLAs [8, 9] with respect to each cloud component. The second phase is provided by the *Decision support tool* regarding cloud resource modeling through a continuous discovery and categorization of cloud services from different CSPs. Moreover, this module assists the *Application DevOps team* by selecting the set of combinations of cloud resources that best matches with the multi-cloud application functional and security needs. The third phase is the monitoring and operational stage, provided by the MSAP.

2.2 The MSAP inputs

As mentioned previously, the MSAP fits the operation phase of the MUSA framework and considers two main inputs from the previous modules, in order to work properly:

- The Security SLA of the application to monitor: The MSAP recuperates the single application components SLAs or the multi-cloud composite application SLA. From single SLAs, the MSAP can monitor the security of single components, and from composite SLAs it can check the end-to-end security of the multi-cloud application taking the communication exchanges between remote components into account.
- The application deployment plan: From this plan, the MSAP recuperates the list of monitoring agents deployed with each application component as well as their IP addresses. This information is vital to link the monitoring agent with the application component to monitor the right security metrics that are specified in the application component security SLA.

The MUSA workflow is composed, as demonstrated in Fig.2, of four main steps that come after gathering and preprocessing data from different monitoring agents. More details about these steps are provided in the next subsections.

2.3 Monitoring agents

Network monitoring agent Monitors a set of combined functionalities presented in the following list: (a) Data capture, filtering and storage (b) Events extraction and statistics collection, and (c) Traffic analysis and reporting providing, network, application, flow and user-level visibility.

This agent facilitates network performance monitoring and operation troubleshooting through its real-time and historical data gathering. With its advanced rules engine, the monitoring agent can correlate network events to detect performance, operational, and security incidents.

System monitoring agent Monitors operating system resources which may be the cause of server performance degradation, and spots performance bottlenecks early on. The agent relies on Linux top command, which is frequently used by

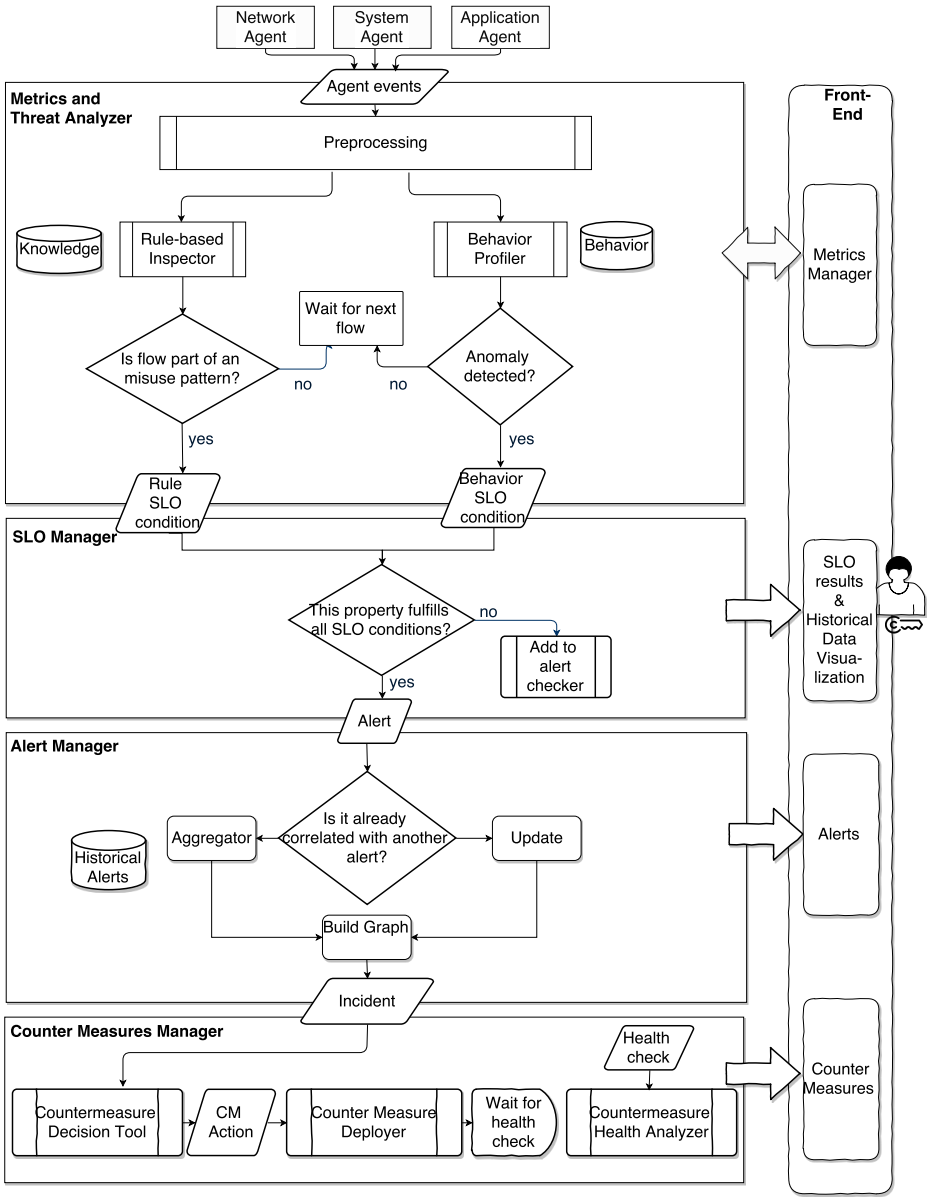


Fig. 2. A MSAP as a service instance general workflow

many system administrators to monitor Linux performance, being available in many Linux/Unix-like operating systems. The top command is used to display all the running and active real-time processes in an ordered list updating it

regularly. It displays CPU usage, Memory usage, Swap Memory, Cache Size, Buffer Size, Process PID, User, among others.

Application monitoring agent Monitors information about the internal state of the target system, i.e., multi-cloud application component to the MSAP during its operation. It notifies the MSAP about measurements of execution details and other internal conditions of the application component. The application monitoring agent is a Java library composed by two parts. The first is an aspect to be weaved into the application code via pointcuts in order to send application-internal tracing information to the MSAP for analysis. It is composed of a set of functions that can be weaved in strategic application points to capture relevant internal data. The second part connects the aspect with the notification tool via a connector library, providing a simple interface for sending log data to the MSAP in a secure way. In other words, the application monitoring agent is responsible for extracting the information from the system, and the connector is in charge of transferring it.

2.4 Preprocessing module

This module has a particular challenge, which is extracting the right information from the collected data events provided by different monitoring agents and from different CSPs, in order to build the correct usage profiles. Additionally, in real cloud environments, periodic reports may be subject to loss or high latency, due to the applications elasticity or VM-related features (e.g., restarting a VM, rolling back). Therefore, this unit is meant to be dynamic, where features are analyzed regarding time-based contextual information. This has the advantage of decreasing the usage of resources for the analysis of large amounts of data, therefore increasing the performance of the framework and reasoning detection. Also following this direction, it is relevant at the moment of keeping a non-redundant knowledge and behavior dataset.

2.5 Metrics and Threat analyzer

The detection module consists of two sub-modules: a Rule-based inspector and a Behavior profiler, as shown in Fig.2.

The first relies on an engine that receives information events from the pre-possessing module, regarding user's access to non-authorized data, which are checked against these permission rules. Additionally to this policy control, some of the attributes obtained from the agents are inspected for specific pattern-matching detection.

The second module also receives the preprocessed data and comprehends two functions: the online learning and anomalies detection.

Most of the literature related to anomaly detection establishes a separated two-stages process, where systems are trained with *normal* data for second-stage comparison with new incoming information. This idea lacks dynamism, as cloud

behavior may vary in mid-long term, and is highly dependent upon the nature of the training data. Therefore, the proposed self-learning module is capable of feeding and updating itself dynamically from new incoming data flows. This system will discriminate if it is appropriate to feed itself or not, lowering the possibility of training the engine with *malicious activity* as *normal*.

The model uses a semi-unsupervised learning, given the fact that new input data has not been labeled yet and it needs to be classified on the basis of their statistical properties only. The supervised component comprehends a smaller labeled dataset created in a lab environment, which learns from known attacks.

2.6 Service Level Objectives (SLO) Manager

The SLO Manager is able to check measured metrics to assert which objectives are useful in defining an anomalous behavior or a disrespected rule. The latter is already paved since it consists of rules that are continuously checked, but the challenge is designing criteria for stating SLO's for abnormal activity.

2.7 Alert Manager and Countermeasures Manager

The agents implement a prevention and mitigation methodology through a set of defenses, practices, and configurations prior to any attack, with the aim of reducing the impact of such attack. These issues may be addressed by network security, data protection, virtualization or isolation of resources.

The Incident handler responds to a policy-based alert and countermeasures mechanisms, given the severity of the incident diagnosed. This corresponds to the *Alert Manager* and *Countermeasure Manager* components from Fig.2. The last module is intended to advise the CSPs and may consist in notifying the administrator to roll back the composite application, replicating a database, upgrading passwords complexity, disabling a specific user, among others.

The latter presents a crucial challenge because sometimes CSPs are unaware precisely of the countermeasures to consider because there are no established relationships between cloud components and their dependencies. This can be solved by clarifying these relationships.

3 Case study: Service availability in smart-city application

We studied the MSAP in the context of a multi-cloud platform for a smart city application, as depicted in Fig.3. The TSM application (which stands for Tampere Smart Mobility) with the exception of the mobile application, has an architecture which is distributed in nature. Thus, each of the TSM components are decoupled and developed independently. This application utilizes resources, services and information from the FMI (Finnish Meteorological Institute), Google directions and the Tampere Intelligent Transport System and Services (ITS) platform. The general schema is composed of:

1. TSM Engine (TSMe): It is an orchestrator which receives requests from different TSM components, analyses the requests according to several mappings, determines the appropriate TSM component required for processing the request and forwards the request to it.
2. Component Journey Planner (MJP): It provides multi-modal and optimal journey options based on the specified departure and destination points. Journey options are provided for buses, cars, cycling and walking.
3. Component Consumption Estimation Calculator (CEC): It calculates the energy needed to complete every journey option specified in the TSM application via a mobile application. It provides this information in a user-friendly way such as: the amount chocolate bars that would be burnt if the user chooses to walk to his/her destination.
4. Component IDM: It handles user authentication and authorization, as the security pillar of the entire TSM application. It authorizes the requests made over the resources and services that each TSM component exposes.

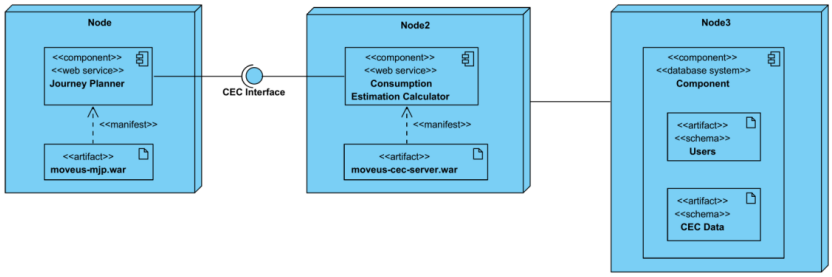


Fig. 3. Topology for multi-cloud case study

Given the real-time requirement for this platform, we decided to study the service availability metric, defined as a non-functional requirement, specified in terms of the percentage of time a system or a service is accessible [16]. To monitor service availability, we deployed dockers for each of the mentioned components and implemented the topology in OpenStack. We collected events from the system, network and application agents, and checked that the application processes are operative. Additionally, this metric uses the active monitoring module of the MSAP, in favor of checking the service availability from an end-user perspective. This observation is helpful given the fact that the agent events may show the service is running when it may not be visible from outside the cloud.

Continuing the MSAP flow, we parse the SLA and extract the SLO metric for service availability, as described in Fig.3. This SLO is an individual example which is instantiated for each component.

To assess the functionality of the MSAP, our testbed consisted in actively shutting down a component of the system and checking visualization results, alert notifications and countermeasures activations, through the dashboard front-end. Fig.5a. and Fig.5b. correspond to the metrics presented in dashboard of

```
<musa:SLO SLO_ID="1">
    <musa:Metric>Service availability</musa:Metric>
    <musa:SLOexpression>
        <musa:oneOpExpression operator="ge (>=)" operand="99.9"/>
    </musa:SLOexpression>
    <musa:importance_weight>HIGH</musa:importance_weight>
</musa:SLO>
```

Fig. 4. SLO XML for service availability metric in TSM application

the MSAP, and contain all the monitored historical data regarding the service availability metric. Detailed in Fig.5a., the first white part of the time slot in the graph accurately presents the unavailability of service for several seconds (also depicted as the orange portion of the pie chart). Additionally, Fig.5b. illustrates the numerous notifications for all the period of evaluation.

The Countermeasure Manager module of the MSAP is in charge of using a High Availability (HA) framework[3] as a way of reacting to a possible alert or violation of this metric. This HA framework, one of the security enforcement mechanisms of the MUSA framework, is based on an open-source software built around the Corosync/Pacemaker stack, patched and configured to work together to bring clustering mechanisms to multi-cloud-based services. This framework encapsulates each of the TSM application components (e.g., TSMe, CEC, IDM and MJP) and handles the task of their deployment in a redundant way managing thus different availability failures and proposing a fault tolerant system that guarantees the availability of different TSM components.

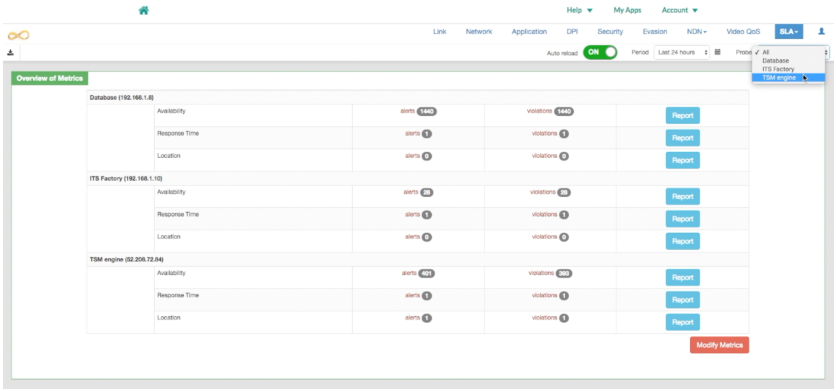
4 Related work

From the monitoring perspective, current solutions to assess security can still be used in virtualized network environments [2, 4]. Nevertheless, they need to be adapted and correctly controlled since they were meant mostly for physical and not virtual systems, and they do not allow fine-grained analysis tailored to the needs of CSCs and virtualized networks. The lack of visibility, controls on internal virtual networks and the heterogeneity of devices used, make many performance assessment applications ineffective. On one hand, the impact of virtualization on these technologies needs to be assessed. On the other hand, these technologies need to cope with ever-changing contexts and trade-offs between the monitoring costs and benefits involved. Here, virtualization of application components facilitates changes, making it necessary for monitoring applications to keep up with this dynamic behavior.

Solutions such as Ceilometer [2], a monitoring solution for OpenStack, provide efficient collection of metering data regarding CPU and network costs. However, it is focused on creating a unique contact point for billing systems to acquire

[3] https://dspace.cc.tut.fi/dpub/handle/123456789/24492?locale-attribute=en

(a) Visualization



(b) Notifications

Fig. 5. MSAP Dashboard

all of the measurements they need, and it is not oriented to perform any action to improve the metrics that it monitors. Furthermore, security issues are not part of the monitored features. StackTach [4] is another example oriented to monitor performance for billing purposes by auditing the OpenStack's Nova component. Similarly, but not specifically oriented to billing, Collectd [10] gathers system performance statistics and provides mechanisms to store the collected values. A recent project from OPNFV named Doctor [3], focuses on the creation of a fault management and maintenance framework for high availability of network services on top of virtualized infrastructures.

In terms of security, OpenStack provides a security guide [15] with best practices determined by cloud operators when deploying their solutions. Some tools go deeper to guarantee certain security aspects in OpenStack, for instance: Bandit [18] provides a framework for performing security analysis of Python source code; Consul [11] is a monitoring tool oriented to service discovery that also performs health checking to prevent routing requests to unhealthy hosts.

Also, threat detection systems in cloud-based environments usually enhance security mechanisms by monitoring system's health. They correspond to a hardware device or software application that monitors activity (e.g., from network, VM host, user) for malicious policy violations. Zbakh et al. evaluated in [19] several Intrusions Detection Systems (IDS) architectures through proposed multi-criteria decision technique, according to the above-introduced requirement together with few others such as: Performance, availability, scalability, secure and encrypted communication channels, transparency with respect to end-users, information security policies as input to the architecture, accuracy including the number of false positives (FP) and false negatives (FN) and detection methods used, among others.

According to such literature, IDS architectures may vary if they are distributed, centralized, agent-based or collaborative [19]. Patel et al. [17] provided an extended systematic-based study of intrusion detection systems, presenting a classification with regards to response time, alarm management, detection method, data collection type, among others. In general, these systems are designed with the following modules: data capturing (Section 2.3) and preparation (Section 2.4), which function as an input for the data analysis and detection (Section 2.5). The latter functionality corresponds to the algorithms implemented to detect suspicious activities and known attack patterns.

5 Conclusion and Future work

The MUSA Security Assurance Platform is proposed as a service that needs to be deployed in the suitable CSP (or CSPs since we can divide the platform into multiple components or micro-services). It offers a set security controls and requirements according to the application needs. Moreover, the MSAP is able to enforce the security of multi-cloud applications by executing the necessary countermeasures to security requirements or to mitigate undesired issues. Its real-time data collection and analysis, together with its virtualized (cloud-based) nature, makes the MSAP a powerful tool to provide multi-cloud applications with end-to-end assurance capabilities.

In detail, this platform presents several advantages, as includes techniques to perform the monitoring of applications that are deployed over heterogeneous cloud resources. It is also based in the concept of monitoring security metrics from SLAs to detect potential deviations and trigger countermeasures to protect applications against attacks and anomalies. This service is available following this link http://assurance-platform.musa-project.eu/ and a demonstration of the tool for the presented use-case is available on You-Tube following this link: https://www.youtube.com/watch?v=zc6p-0H9yFo.

As future work, we consider experimenting with an automatic deployment of reactive countermeasures. Additionally, we plan on extending the set of security metrics available for monitoring, by enhancing our monitoring agents and by developing new techniques for detection in the Metrics and Threat Analyzer

module. This last section will focus further on the detection of unknown threats with anomaly-behavior detection techniques.

6 Acknowledgment

The work presented in this paper has been developed in the context of the MUSA EU Horizon 2020 project [1] under grant agreement No 644429.

References

1. Musa project. http://www.musa-project.eu/, (Retrieved January 2017)
2. Openstack ceilometer. http://docs.openstack.org/developer/ceilometer/, (Retrieved January 2017)
3. Opnfv doctor. http://wiki.opnfv.org/doctor, (Retrieved January 2017)
4. Stacktach. http://stacktach.readthedocs.org/en/latest/index.html, (Retrieved January 2017)
5. Lifecycle management of service-based applications on multi-clouds: a research roadmap (2013)
6. Multi-Cloud: expectations and current approaches (2013)
7. Carvallo, P., Cavalli, A.R., Mallouli, W., Rios, E.: Multi-cloud Applications Security Monitoring, pp. 748–758. Springer International Publishing, Cham (2017)
8. Casola, V., Benedictis, A.D., Modic, J., Rak, M., Villano, U.: Per-service security sla: A new model for security management in clouds. In: 2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE). pp. 83–88 (June 2016)
9. Casola, V., Benedictis, A.D., Rak, M., Rios, E.: Security-by-design in clouds: A security-sla driven methodology to build secure cloud applications. Procedia Computer Science 97, 53 – 62 (2016), http://www.sciencedirect.com/science/article/pii/S1877050916320968, 2nd International Conference on Cloud Forward: From Distributed to Complete Computing
10. Collectd: http://collectd.org/, (Retrieved January 2017)
11. Consul: https://www.consul.io/, (Retrieved January 2017)
12. Ferry, N., Rossini, A., Chauvel, F., Morin, B.: Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems. 2013 IEEE Sixth International Conference on Cloud Computing (2013)
13. Global Inter-cloud Technology Forum: Use Cases and Functional Requirements for Inter-Cloud Computing. Tech. rep. (2010)
14. Grozev, N., Buyya, R.: Inter-Cloud architectures and application brokering: taxonomy and survey. Software - Practice and Experience 44(3), 369–390 (2012)
15. Guide, O.S.: http://docs.openstack.org/sec/, (Retrieved January 2017)
16. Nabi, M., Toeroe, M., Khendek, F.: Availability in the cloud: State of the art. Journal of Network and Computer Applications 60, 54 – 67 (2016), http://www.sciencedirect.com/science/article/pii/S1084804515002878
17. Patel, A., Taghavi, M., Bakhtiyari, K., Celestino Júnior, J.: An intrusion detection and prevention system in cloud computing: A systematic review. Journal of Network and Computer Applications 36(1), 25–41 (Jan 2013)

18. Project, B.: http://wiki.openstack.org/wiki/Security/Projects/Bandit, (Retrieved January 2017)
19. Zbakh, M., Elmahdi, K., Cherkaoui, R., Enniari, S.: A multi-criteria analysis of intrusion detection architectures in cloud environments. In: 2015 International Conference on Cloud Technologies and Applications (CloudTech). pp. 1–9. IEEE (2015)
20. Zeginis, C., Kritikos, K., Garefalakis, P., Konsolaki, K.: Towards cross-layer monitoring of multi-cloud service-based applications. Lecture Notes in Computer Science pp. 188–195 (2013)