Per-Service Security SLa: A New Model for Security Management in Clouds

5
Author(s)
Valentina Casola ; Alessandra De Benedictis ; Jolanda Modic ; Massimiliano Rak ; Umberto Villano
View All Authors

Abstract:
In the cloud computing context, Service Level Agreements (SLAs) are contracts between Cloud Service Providers (CSPs) and Cloud Service Customers (CSCs), stating the guaranteed quality level of the services offered by CSPs. Existing cloud SLAs focus only on few service terms, completely ignoring all security related aspects. They are often reported in a way that is hardly understandable for customers. Moreover, they offer guarantees uniform for all offered services and all customers, regardless of particular service characteristics or customers specific needs. This paper presents a framework that enables the adoption of a per-service SLA model, by supporting the automatic implementation of cloud Security SLAs tailored to the needs of each customer for specific service instances. In particular, the process and the software architecture for per-service SLA implementation are shown. A case study application demonstrates the feasibility and effectiveness of the proposed solution.

Published in: Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2016 IEEE 25th International Conference on

Date of Conference: 13-15 June 2016

INSPEC Accession Number: 16213627

Date Added to IEEE *Xplore*: 11 August 2016

DOI: 10.1109/WETICE.2016.27

ISBN Information:

Publisher: IEEE

SECTION I.
Introduction

In the cloud computing context, Service Level Agreements (SLAs) are contracts between Cloud Service Providers (CSPs) and Cloud Service Customers (CSCs), stating the guaranteed quality level of the services offered by CSPs. Despite the intense standardization and research efforts [1]–[2] [3]–[4][5], current cloud SLAs are essentially descriptions in natural language that focus only on few service terms (mostly on availability), completely ignoring all security related aspects. Moreover, CSPs only offer SLAs with which they guarantee these service terms uniformly for all offered services to all customers, regardless of particular service characteristics or customers specific needs.

The standardized service level offered to all customers by a given CSP, i.e., the use of a *uniform SLA*, is hardly ever satisfactory, especially as far as security is concerned, since it does not take into account different requirements of different CSCs nor the peculiar characteristics of different services provided by CSPs. Some commercial cloud providers (e.g., Google Cloud[1] and Amazon[2]) have recently split their SLAs over the services they offer; however, (i) their contents hardly adapt to services (i.e., SLAs for different services are mostly the same), (ii) security features are still not covered (i.e., SLAs are performance-focused, including objectives associated mostly to availability), and (iii) they are still not negotiable (i.e., the CSC does not have an option to acquire the service with the specific required security properties). Hence, in practice, the same security policy is applied by CSPs to all services and to all customers in a uniform way.

Contents

The alternative approach is to adopt a CSC-based *per-service* SLA model. This model entails the use of a "tai-lored" SLA for each service, where even CSC can stipulate a distinct SLA for each leased service. The model is commonly considered inapplicable in clouds, due to the inherent management complexity on the CSP side. It is clear that the CSP has to manage a widely different range of SLAs, and this is, if not unfeasible, simply not a priority for CSPs.

In this paper, we present a framework that enables the adoption of a novel CSC-based *per-service* SLA model. The deployment of the supporting software of the framework devoted to *per-service* SLA management requires no user intervention, i.e., it can be implemented automatically on the top of prevalent tools for the automated software management. This is perfectly in accordance with the automation, self-service, no-user intervention principles that are among the foundations of cloud computing. The results presented in this paper are partly related to the activities carried out in the context of the SPECS[3] and MUSA[4] EU projects, whose objectives are respectively to provide a platform-as-a-service to develop SLA-based secure cloud security services and to promote security-by-design in multicloud application contexts through the adoption of SLAs. However, although the paper focuses on the provisioning of CSC-based *per-service* security-related SLAs, the introduced approach can be adopted in different contexts, e.g., for the implementation of performance-oriented SLAs.

The paper is organized as follows. In Section II we present related work. Then, in Section III, we deal with the implementation of *per-service* SLAs and, in Section IV, we briefly present our reference Security SLA model. In Section V, we discuss the software architecture designed for the implementation of the proposed process, and in Section VI we show a concrete application, the provisioning of a secure web container service. The paper closes with our conclusions.

Full Text
Abstract
Authors
Figures
References
Citations
Keywords
Footnotes
Back to Top

SECTION II.
Related Work

The adoption of SLAs in cloud computing has been inspired by telecommunication and GRID systems. However, in the cloud context, the problem of stating formal guarantees on services is more complex, due to the lack of a reference technology and of well-assessed standards for implementation of services. As a consequence, cloud SLAs are a hot topic of research and there are several ongoing initiatives, both in industrial and academic contexts, aimed at supporting their use and their standardization. WS-Agreement (WSAG) [2] is currently the only standard supporting a formal representation of SLAs and a protocol for their automation. Although it was devised in a well-defined technological context (i.e., the GRID) and not completely fit in other contexts, most of recent cloud-oriented FP7 projects (Contrail[5], mOSAIC[6], Optimis[7], PaaSage[8]) are inclined to adapt it for use in clouds.

A review of available literature on SLA management in clouds has recently appeared in [6]. It shows that security is among the least studied SLA parameters. In service-oriented environments, several proposals addressing the negotiation of dynamic and flexible SLAs have appeared [7]. The problem of ensuring quality of service and SLA facilities on top of unreliable, intermittent cloud providers is discussed in [8]. A strategy for the autonomous negotiation among a CSP and its customers is presented in [9]. However, the negotiation parameters considered are essentially cost and availability. An interesting proposal addressing performance losses due interference effects is the use of a QoS-aware control framework to obtain quality-enabled clouds (Q-Clouds), proposed in [10]. A framework to determine the trustworthiness of cloud service providers by employing the real time monitoring of their services is presented in [11].

The main commercial IaaS providers (Amazon, Rackspace, GoGRID, etc.) usually propose an SLA contract that specifies simple grants on uptime percentage or network availability, along with additional services (for example, Amazon CloudWatch) that monitor the state of target resources (i.e., CPU utilization and bandwidth) [12]. Open Cloud Engine software like Eucalyptus, Nimbus, OpenNebula, also implement monitoring services for the private cloud provider, but do not provide solutions for SLA negotiation and enforcement. An interesting survey on the use of SLAs in clouds by current enterprises is [13]. Its Authors recognize that often the SLAs proposed by cloud providers are not "well defined", and that, since the monitoring tools are managed by the CSPs themselves, in case of any breach of SLA parameters the customers cannot invoke the agreed penalties from a legal point of view.

SECTION III.
Implementing Per-Service Slas

Contents

Full Text
Abstract
Authors
Figures
References
Citations
Keywords
Footnotes
Back to Top

As outlined in the introduction, our aim is to promote the adoption of a *per-service* SLA model, which entails the use of a "tailored" SLA for each service. In order to demonstrate the feasibility of the approach, we defined a process for the *implementation* of per-service Security SLAs that complies with the main cloud computing principles, including the *on-demand self-service* and the *resource pooling* capabilities. The key idea behind such process is the adoption of the Security-as-a-Service approach, enabled by the use of a configuration management solution. Configuration management solutions, like Chef[9] or Puppet[10], are commonly adopted to automate software installation and management over a pool of virtual machines (VMs). Through such tools, it is possible to build up cloud services delivered in a Software-as-a-Service (SaaS) fashion, automating the process of deployment of commercial-off-the-shelf (COTS) software over dynamically acquired infrastructure resources (as VMs or container hosting).

Security best practices prescribe the implementation of a set of security controls (such as those specified in the NIST *Security Control Framework* [14] or the Cloud Security Alliance's *Cloud Control Matrix* [15]) in order to cope with specific security risks. Usually, these controls are determined on the basis of a risk analysis process conducted by an expert. They represent the security policies set in the system and are implemented through suitable *security mechanisms*. Our approach consists in building a catalogue of security mechanisms able to implement given security policies, and in offering them as-a-service. Our security mechanisms are based on open source solutions and come with a set of metadata, declaring the implemented security controls and the configurable parameters. Through a dedicated Security SLA model, presented in the next section, we collect the CSC's security requirements in terms of security controls and identify how we can implement them through a set of dedicated security mechanisms. Figure 1 briefly summarizes the proposed approach: the core of the solution is the *SLA Automator*, a web application that on one side negotiates with the CSC the content of a Security SLA, and on the other controls a *Broker*, devoted to acquiring resources from CSPs, and a *Configuration Manager*, responsible for automating the enforcement of desired security controls through the activation of needed security mechanisms.

It is worth noticing that, usually, configuration managers are controlled by humans, in order to automate both the deployment of software instances on multiple resources and their synchronization. In our case, the control is given entirely to the *SLA Automator*, which configures and controls the underlying *Configuration Manager* based on the content of the SLA. The *SLA Automator* also controls a *Broker*, which is in charge of invoking the APIs exposed by CSPs to manage (acquire/release/reconfigure) the resources needed to build the service provided to customers.

Once the *SLA Automator* agreen on a Security SLA with the CSC, it builds an *implementation plan* that contains the list of the resources to acquire (e.g., type and number of VMs) and of the security mechanisms to deploy and activate on each of them. The implementation plan is then forwarded to the *Configuration Manager* for its actual implementation.

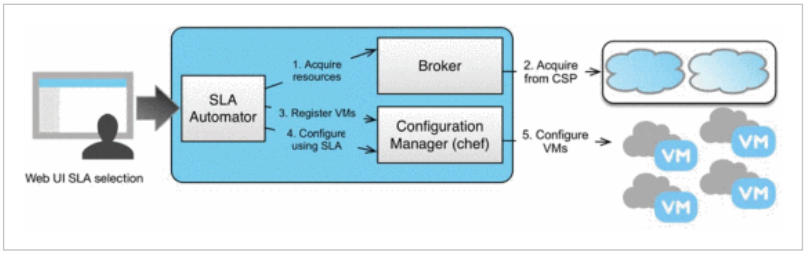

Figure 1.
SLA implementation: The proposed approach

Figure 2 summarizes the full implementation process. In the first step, the CSC defines the terms of the desired SLA. This step is carried out according to a template-based process, as prescribed by WS-Agreement. *SLA Templates* summarize all the features of supported services and represent potential SLA offers, which are translated into signed SLAs when an agreement is reached between the two parties. In our model, as will be clarified in the next section, the adopted *Security SLA Templates* include all the security policies that can be offered via the available security mechanisms. Once an agreement is reached, the SLA Implementation phase starts. It mainly consists in (i) identifying the security mechanisms that implement the terms agreed upon in the SLA, (ii) acquiring the resources as declared in the SLA, (iii) registering them into the configuration management system, and finally (iv) running the configuration procedure.

The full process described above enables the complete automation of the Security SLA implementation, even if the security policies are customized according to the customer requirements. The software architecture designed to implement such process will be described in

requirements. The software architecture designed to implement such process will be described in Section V, while the adopted Security SLA model is dealt with in the following section.

Contents

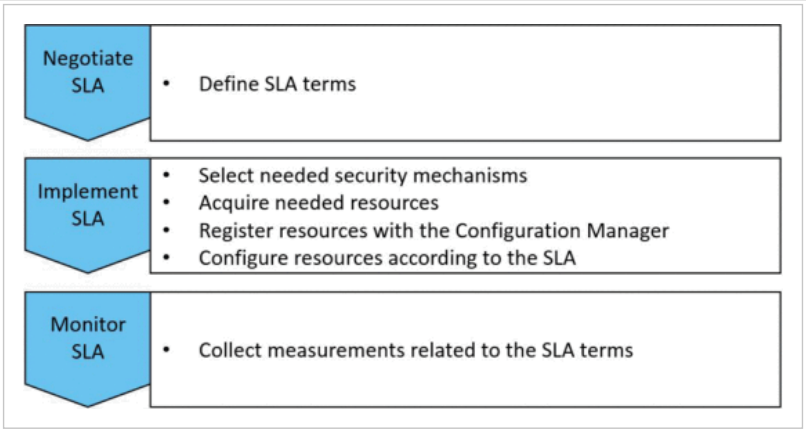

Figure 2.
Security SLA implementation process

Full Text
Abstract
Authors
Figures
References
Citations
Keywords
Footnotes
Back to Top

SECTION IV.
The Security SLA Model

In the context of the SPECS project, we have proposed a novel model of *Security SLA*, allowing for the representation of security-related concepts in a machine-readable format, based on WS-Agreement and amenable to the automatic management of the SLA life-cycle [16], [17].

The main concepts of the SPECS Security SLA model are represented in white boxes in the *Security SLA domain model* of Figure 3. The figure shows the concepts introduced for SLA implementation and discussed in the previous section, namely the *security mechanisms* and the *Security SLA Template* (grey boxes).

The Security SLA model includes a (i) *declarative* part, where the functional and non-functional (i.e., security-related) characteristics of the service being provided are described, and (ii) a measurable part, where the concepts needed to define the guarantees in terms of security offered on the service are specified. The declarative part includes:

- the description of the cloud resources (i.e., VMs) used to build the service object of the agreement and of their providers (**Resources Providers);**
- the declaration of the **Security Capabilities** offered on top of the service object of the agreement, defined in terms of the *Security Controls*, belonging to a given *Control Framework*, which must be implemented [14], [15];
- the declaration of the **Security Metrics** that can be monitored by the service customer to verify the correct delivery of declared capabilities.

The part of the Security SLA model devoted to the definition of offered security guarantees (i.e., the measurable part) is represented by a set of **Security Service Level Objectives (SLOs).** SLOs are constraints on the admissible values of declared security metrics, and represent the security levels that the service customer requires and that the service provider accepts to offer. SLOs and the relative security metrics are *associated* with the declared security capabilities and are meant to offer a quantitative measure of the declared security controls.

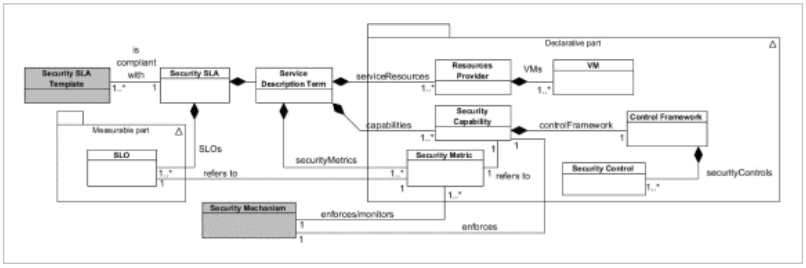

Figure 3.
The security SLA domain model

Contents

It is worth pointing out that the described model refers to an SLA built on the top of the SLA Template for a specific customer and a specific service instance (*CSC-based per-service* SLA), possibly offered through a supply chain that involves the acquisition of resources/services from more than one provider.

Full Text
Abstract
Authors
Figures
References
Citations
Keywords
Footnotes
Back to Top

SECTION V.
Design of the SLA Implementation Framework

In this section, we present the software architecture proposed to implement the approach illustrated in Figure 1, and provide some details on the design of the *SLA Automator* component. As shown in Figure 4, the *SLA Automator* consists of four software components, namely the *Application*, the *SLA Manager*, the *Services Manager* and the *Implementation component*.

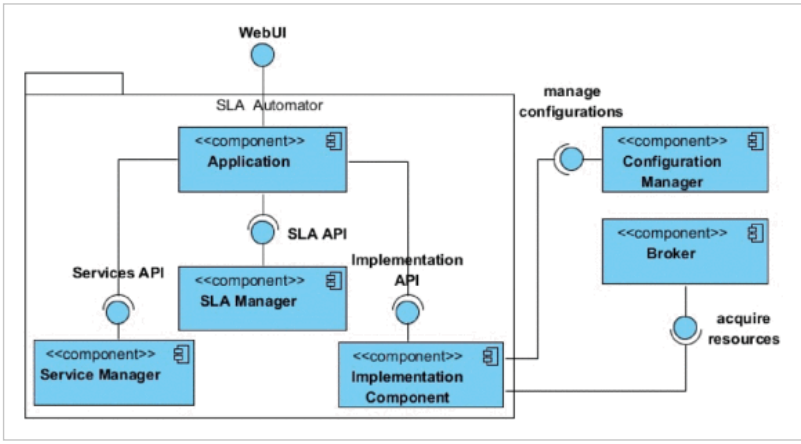

Figure 4.
Per-service SLA implementation architecture

The *SLA Manager* provides the basic functionalities to create, store and retrieve SLAs and keeps track of the current state of processed SLAs. The *SLA Manager* also manages a set of SLA Templates, representing the security offers that are available to customers and which are used during negotiation, according to the WS-Agreement specification. The functionalities provided by the *SLA Manager* are exposed through the *SLA API*, a publicly available REST API [18], which can be easily integrated into existing applications to enable the construction of software solutions for orchestrating services based on SLAs.

The *Service Manager* is responsible for managing all the information on available security capabilities and on the supported security mechanisms that can be activated in order to offer such capabilities in an as-a-service fashion. The *Service Manager* exposes the *Services API*, which can be invoked to add or remove security capabilities and to update the set of supported security mechanisms and security metrics.

The *Implementation component* is able to implement an SLA by managing the acquisition and configuration of the resources (i.e., the VMs) needed to put in operation the service object of the SLA, along with all security features included as guarantees. This includes the activation and configuration of the security mechanisms that offer the security capabilities selected by the CSC. Resource acquisition is carried out by the *Broker* component, which interacts with public or private providers to buy resources on behalf of the customer. Resource configuration is automated by the *Configuration Manager*, which is responsible for automating the installation, execution and configuration on acquired virtual machines, of the software components that implement the requested security features (i.e., the needed security mechanisms). The *Configuration Manager* has been implemented by exploiting Chef, one of the most popular cloud automation tools.

The component identified as *Application* in Figure 4 orchestrates all the components previously discussed by invoking their exposed APIs. The *Application* offers a web interface to the customers, through which they can select the desired services and their features and can follow the whole implementation process. Moreover, the Application interface also allows the customers to monitor the state of their SLAs.

It is important to point out that the target service, along with all requested security mechanisms

Contents

Full Text
Abstract
Authors
Figures
References
Citations
Keywords
Footnotes
Back to Top

It is important to point out that the target services, along with the requested security mechanisms, are installed and activated on the resources acquired and controlled by the customer. Nothing can be said in general on services that run on resources that are not directly under the customer control, since no guarantees can be expressed on them unless they are provided by the provider itself. Moreover, it is worth mentioning that, since acquired services are configured on the customer's resources, it is possible to monitor that the guarantees stated in the SLA are actually respected. Measurements related to security metrics associated with the selected security capabilities are continuously gathered by monitoring systems that are activated by the *Implementation component* together with security mechanisms. SLA monitoring is out of the scope of this paper. The interested reader is referred to [19] for further details.

SECTION VI.
A Case Study: A Secure Web Container Service

In order to show the feasibility of our approach, in this section we illustrate a case study related to the acquisition of a *secure web container* service through the *Secure Web Container Application* available online as a SPECS demonstrator[11]. The target customers of this service are represented by web developers that are aware of the main security threats for web servers and are able to formulate specific security requirements (possibly derived from best practices and guidelines), but need support in identifying and implementing the required security countermeasures.

First of all, the developed application enables the customer to negotiate the desired features of the service. As SPECS supports a template-based negotiation, the customer is presented with a set of possible choices, including the provider to be used for the acquisition of needed resources (which, in this case, are represented by VMs configured with a web server instance) and the available enforceable security capabilities. In our example, three capabilities are available:

- *Web Resiliency*. Capability of surviving to security incidents involving a web server, by implementing proper strategies aimed at preserving business continuity' achieved through redundancy and/or diversity.
- *Vulnerability Detection*. Capability of detecting the vulnerabilities a machine (and the installed software) is subject to.
- *DoS Detection and Mitigation*. Capability of detecting and reacting to security attacks aimed at distrupting the system availability.

Each capability is associated with a set of security controls and with one or more security metrics, available to the customer for monitoring purposes. The customer can choose the metrics of interest and define thresholds on them to specify SLOs. For example, the *Web Resiliency* capability has two metrics associated, namely Level of Redundancy (LoR), which identifies the number of aligned web server replicas kept active during service operation to counteract possible attacks against availability, and Level of Diversity (LoD), which represents the number of different web server instances (e.g., Apache and Nginx) that are actually installed on such replicas. When specifying SLOs on top of such metrics, a customer may require that $LoR \geq 3$ and $LoD = 2$, implying that, at any time, at least 3 replicas of the web server are kept alive, and at least two of them run different web container instances.

At the end of the SLA negotiation phase, the SLA offer, including all selected capabilities and metrics, is signed and pushed to the *Implementation component* for the SLA implementation phase. As illustrated in the previous section, the *Implementation component* coordinates the acquisition and configuration of needed resources by installing and activating proper security mechanisms. In particular, the three capabilities discussed above are implemented through the following security mechanisms:

- *Web Pool*. It offers (a pool of) virtual machines, hosting synchronized web servers and a load balancer[12]. The service provides redundancy and diversity capabilities.
- *SVA (Software Vulnerability Assessment)*. It regularly performs vulnerability assessment over the virtual machines, through software version checking and penetration tests [13].
- *DoSprotection*. It consists in a solution for denial of service attacks detection and mitigation [14]based on the OSSEC too1[15].

For example, in the case of the *Web Resiliency* capability, in order to enforce the two SLOs

discussed above via the *Web Pool* mechanism, the *Implementation component* will acquire 4 VMs: one will host the load balancer component, while the others will host 3 replicas of the target web server. Two of these replicas will be configured with an Apache web server, while the third one will be configured with an Nginx web server. When implementation is complete, the SLA monitoring phase starts, and the customer can monitor the behaviour of the deployed service by checking the measured values of selected security metrics.

Contents

Full Text

Abstract

Authors

Figures

References

Citations

Keywords

Footnotes

Back to Top

SECTION VII.
Conclusions

In this paper, we investigated the adoption of CSC-based *par-service* Service Level Agreements in clouds. In a context where existing CSP proposals essentially provide simple grants on performance aspects to all prospective customers, we focused on security guarantees offered to customers, according to their particular needs and related to specific service instances.

Based on a novel *per-service* Security SLA model, we identified the process needed to enable the automatic management of the whole life-cycle of cloud SLAs, and presented the software architecture designed for the implementation of such process. In order to demonstrate the feasibility of the whole solution, we also discussed a case study related to a demonstrator application developed in the context of an EU project we are involved in. Our approach is perfectly in accordance with the automated, self-service, no-user intervention principles that are among the foundations of cloud computing.

ACKNOWLEDGMENT

This research is partially supported by the grant FP7-ICT-2013-11-610795 (SPECS) and H2020-ICT-07-2014-644429 (MUSA).

Keywords

IEEE Keywords
Security, Cloud computing, Context, Monitoring, Contracts

INSPEC: Controlled Indexing
software architecture, cloud computing, contracts, security of data

INSPEC: Non-Controlled Indexing
software architecture, per-service security SLA model, security management, cloud computing, service level agreements, cloud service providers, CSPs, cloud service customers, CSCs, cloud security SLAs

Author Keywords
Security Service Level Agreement, Cloud Security, Per-service SLA

Authors

Valentina Casola
DIETI, Univ. di Napoli Federico II, Naples, Italy

Alessandra De Benedictis
DIETI, Univ. di Napoli Federico II, Naples, Italy

Jolanda Modic
XLAB, Ljubljana, Slovenia

Massimiliano Rak
DII, Seconda Univ. di Napoli, Aversa, Italy

Umberto Villano
DING, Univ. del Sannio, Benevento, Italy

Related Articles

Contents

Full Text
Abstract
Authors
Figures
References
Citations
Keywords
Footnotes
Back to Top