This is the pre-peer reviewed version of the following article: Per-Service Security SLAs for Cloud Security Management: Model and Implementation by Valentina Casola; Alessandra De Benedictis; Jolanda Modic; Massimiliano Rak; Umberto Villano, which has been published in final form in International Journal of Grid and Utility Computing and is available at https://www.inderscienceonline.com/doi/abs/10.1504/IJGUC.2018.091719

DOI: 10.1504/IJGUC.2018.091719
Per-Service Security SLAs for Cloud Security Management: Model and Implementation

Valentina Casola, Alessandra De Benedictis
Università di Napoli Federico II, DIETI, Napoli, Italy E-mail: {casolav,alessandra.debenedictis}@unina.it

Jolanda Modic
XLAB, Ljubljana, Slovenia E-mail: jolanda.modic@xlab.si

Massimiliano Rak
Seconda Università di Napoli, DII, Aversa, Italy E-mail: massimiliano.rak@unina2.it

Umberto Villano
Università del Sannio, DING, Benevento, Italy E-mail: villano@unisannio.it

Abstract: In the cloud computing context, Service Level Agreements (SLAs) “tailored” to specific Cloud Service Customers (CSCs) seem to be still an utopia, and things are even worse as regards the security terms to be guaranteed. In fact, existing cloud SLAs focus only on few service terms, and Cloud Service Providers (CSPs) mainly provide uniform guarantees for all offered services and for all customers, regardless of any particular service characteristics or of customer specific needs.

In order to expand their business volume, CSPs are currently starting to explore alternative approaches, based on the adoption of a CSC-based per-service security SLA model. This paper presents a framework that enables the adoption of a per-service SLA model, 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, related to the provisioning of a secure web container service, is presented and discussed, to demonstrate the feasibility and effectiveness of the proposed solution.

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

1 Introduction

The Service Level Agreements (SLAs) are contracts between clients and providers, largely used in service-oriented architectures. They are intended to state the guaranteed quality and security levels that a provider is able to offer to its own clients. But, even if the SLA concept is not new, in the cloud computing context its adoption for contracts between Cloud Service Providers (CSPs) and Cloud Service Customers (CSCs) is still very limited.

Despite the intense standardization and research efforts [12, 1, 13, 8], currently cloud SLAs are essentially descriptions in natural language that focus only on few service terms (mostly on availability), ignoring completely all security-related aspects. Moreover, the leading CSPs on the market offer only SLAs that guarantee service terms for all offered services to all customers, regardless of possible peculiar service characteristics or customers’ specific needs. In other words, the CSPs offer per-Provider SLAs instead of the more desirable per-Service SLAs.

A standardized service level offered to all customers is hardly ever satisfactory, especially as far as security is concerned. Some commercial cloud providers (e.g., Google Cloud (https://cloud.google.com) and Amazon (https://aws.amazon.com)) have recently split their SLAs over the services they offer. However, (i) their contents hardly ever 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, with 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 security properties required). In practice, the same security policies are still applied by CSPs to all services and to all customers in a uniform way.

Many research activities are exploiting the possibility of an alternative approach, based on the adoption of a CSC-based per-service SLA model. This model entails the use of a “tailored” SLA for each service, in such a way that every CSC can stipulate a distinct SLA for each leased service. The model has generally been 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 to keep low the resulting overhead is not a priority for CSPs.

In this paper we present the state of the art on SLA definition and models, with particular focus on security related service terms, and a framework that enables the adoption of a novel CSC-based per-service SLA model. The main feature of the proposed model is that it is able to cope with quantitative aspects of security-related service terms, hence the possibility to measure, monitor and negotiate security. Furthermore, the deployment 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 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 (http://www.specs-project.eu) and MUSA (http://www.musa-project.eu) EU projects, whose objectives are 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, respectively. 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.

This paper is organized as follows. In Section 2 we present an analysis of the most relevant related work, with a wide discussion on the motivation behind adoption of security SLAs. In Section 3, we present our reference Security SLA model. Then, in Section 4 we deal with the steps needed to automatically implement a per-service SLAs. Finally, in Sections 5 and 6 we discuss the software architecture designed for the implementation of the proposed process and show a concrete case study, related to the provisioning of a secure web container service. The paper closes with some final remarks on our ongoing research activities.

2 Motivation and background

Currently many cloud providers “offer” a Service Level Agreement associated with their services [23]. These SLAs state the policies governing the provisioning and the usage of the covered services and typically specify the remedies applied in case of violation (e.g., in terms of service credits). When looking at the SLAs published by providers as Amazon, Rackspace, or GoGRID, several common features can be found:

- cloud SLAs are specified in natural language, and therefore cannot be easily processed or understood;
- the same commitments are offered to all customers, with no possibility of customer-based negotiation;
- the control over the service is carried out by the provider with proprietary monitoring systems (e.g., Amazon CloudWatch);
- guarantees are expressed mostly on performance parameters, such as service uptime, network latency, packet loss, etc. Security guarantees are almost never included and, when present, they barely refer to the enforcement of non-technical safeguards (e.g., the GoGrid SLA includes a commitment on the availability of a 24x365 on-site physical security).

Even if security is currently out of cloud SLAs, providers do actually care about it, and offer plenty of security solutions that can be applied on top of their services. Amazon, for example, offers an Identity and Access Management solution, a Certificate Manager, and a Key Management Service on AWS, which can be configured by the customers, based on their needs. However, no guarantees are given on these services and there is no monitoring system made available by the provider to control that the configured security preferences are actually respected.

Since providers have the technical skills to offer security, even in an as-a-service fashion, and since customers often need explicit guarantees on the security level of acquired services, the introduction of security-oriented SLAs is both desirable and viable. In order to be effective, such SLAs should be tailored to the customers’ needs and should address particular service instances. They are therefore referred to as CSC-based per-service SLAs.

In order to enable the adoption of per-service SLAs, suitable solutions for their life-cycle’s management must be devised. According to WS-Agreement [1], which is currently the only standard recognized for SLA representation, the main SLA life-cycle phases are negotiation, enforcement and monitoring. Since the services we are referring to are customized based on (security) requirements, it is necessary to envision a negotiation mechanism to define the SLA to agree upon. Moreover, a process to automate the configuration and execution of the services covered by the customized SLA must be introduced (SLA enforcement). Finally, suitable monitoring systems must be provided to enable the provider and (hopefully) the customer to assess the level
of delivered security at any time, and to detect possible SLA violations.

Security SLA life-cycle management has been recently addressed by the EU FP7 project SPECS [14] (ended in 2016), which has introduced a novel Security SLA model as an extension of the WS-Agreement standard [4]. Additionally, the project has released an open-source framework and a platform to build and execute secure cloud applications covered by Security SLAs [10, 19, 5, 3]. Security SLAs are also adopted in the ongoing EU H2020 project MUSA [16], focused on the development of secure multi-cloud applications.

Besides these two recent projects and other previous similar European projects dealing with SLAs, several research groups are working on the issues related to negotiation, enforcement and monitoring of an SLA.

For what concerns negotiation, which is currently not supported by any public cloud provider, there exist some proposals that address the negotiation of dynamic and flexible SLAs in service-oriented environments [11]. Related to the cloud environment, in [9] a strategy for the autonomous negotiation among a CSP and its customers is presented. In this paper, however, the parameters considered for negotiation are limited to cost and availability. In [15], SLA negotiation of Quality of Service (QoS) parameters in the context of cloud federations is addressed, and a distributed solution based on inter-operating agents is proposed.

Regarding monitoring, as mentioned before, some providers (as Amazon, Eucalyptus, Nimbus or OpenNebula) do offer monitoring solutions to their customers. While in some cases these solutions are used (by the provider) to monitor the fulfillment of the published SLA (e.g., GoGrid monitoring systems), most of the times they are offered to the customers to let them monitor more in general the performance of acquired services from the system administrator’s point of view (e.g., Amazon CloudWatch can be used by customers to monitor the CPU and disk usage, data transfer speed etc.). As regards the security monitoring aspects, a framework to determine the trustworthiness of cloud service providers by employing the real time monitoring of their services is presented in [21]. Reference [20] proposes a cloud-monitoring scheme offering flexible and dynamically reconﬁgurable QoS monitoring services, adaptable to various cloud based service characteristics.

For cloud service providers, the monitoring scheme provides dashboard-like indicators to continuously manage the performance of their Software-as-a-Service platform and visualize the monitoring data. With particular reference to security monitoring and SLAs, [3] presents a monitoring architecture that is automatically conﬁgured and activated based on a signed Security SLA. Such monitoring architecture integrates different security-related monitoring tools (either developed ad hoc, or already available as open-source or commercial products) to collect measurements related to speciﬁc metrics associated with the set of security Service Level Objectives (SLOs) speciﬁed in the SLA.

Security SLA implementation/enforcement has recently been tackled in [4], where the Authors have discussed both an optimization model and the automated process to deploy software components needed to enforce a Security SLA on (cloud) resources. In [2], the high-level architecture that enables the adoption of a per-service SLA model is presented, along with the implementation process. This paper will go further, detailing such process by a discussion of the underlying data model needed.

3 The Security SLA model

It is hard to reason concretely about the security associated to a cloud service, due to the informal representation of the security requirements and to the lack of measurable security metrics. Therefore, the concept of cloud security is often generic. As anticipated in the previous section, Security Service Level Agreements (Security SLAs), introduced and used by projects like SPECS, MUSA and SLA-Ready [22], enable the expression of the security associated to each cloud service in a formal and measurable way.

In the following, we adopt the SPECS Security SLA model, described in detail in [19, 4]. It is based on the WS-Agreement standard, which has been extended with provider-specific information and security-related concepts. In particular, in such model the security guarantees are specified by a set of enforced security controls. According to NIST [17], a security control is a safeguard or countermeasure prescribed for an information system or an organization designed to protect the confidentiality, integrity, and availability of its information and to meet a set of defined security requirements. In practice, the set of controls declared in a Security SLA expresses the security policy adopted by the service provider. It is a way to evaluate concretely the security requirements that such cloud service is able to grant. Moreover, the proposed model expresses the (security) Service Level Objectives (i.e., the security levels that the cloud services grants) in terms of the (security) metrics associated to such controls that can be used to monitor their correct implementation.

The main concepts of the SPECS Security SLA model are represented in the white boxes of the domain model depicted in Figure 1. The Security SLA model includes (i) a 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 defining the guarantees in terms of security offered for the delivered service are specified.

The declarative part includes:

- the description of the cloud resources (i.e., virtual machines - VMs) used to build the requested service, and of their providers (Resources Provider);
Figure 1  The Security SLA domain model

- the declaration of the **Security Capabilities** offered on top of the service object of the agreement, defined in terms of the set of *Security Controls*, belonging to a given *Control Framework* [17, 7] that must be implemented;

- the declaration of the **Security Metrics** that can be monitored by the service provider and the service customer, to verify the correct delivery of declared capabilities and controls.

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.

In addition to the security-related concepts that compose a Security SLA, Figure 1 shows the main concepts that enable its implementation, namely the **Security Mechanisms** and the **Security SLA Template** (grey boxes). The role of security mechanisms (SMs) will be discussed in detail in Section 4. For what regards the Security SLA Template concept, it is worth pointing out that we consider Security SLAs that are compliant with a given template, and customized based on the security requirements of a specific customer and on a specific service instance (CSC-based *per-service* SLA). They are possibly offered through a *supply chain* that involves the acquisition of resources/services from more than one provider.

In the following sections we will illustrate the entire Security SLA implementation process and discuss the high-level architecture that enables the implementation.

### 4 Implementing *per-service* SLAs

As mentioned before, the implementation of an SLA consists of three steps:

1. **Negotiation**, during which (1) the CSC selects the cloud service and specifies the business and security requirements and (2) an SLA is prepared accordingly. This step may end with (3) the CSP and CSC both signing the SLA, if this feature is supported;

2. **Enforcement**, during which cloud resources are automatically acquired on behalf of the CSC and configured with the requested services;

3. **Monitoring**, during which both CSP and CSC continuously monitor the status of the SLA.

The adoption of the CSC-based *per-service* SLA model requires different actions for the implementation of each individual SLA. For example, the implementation of an SLA related to a cloud storage service guaranteeing regular software vulnerability scans requires the deployment and configuration of specific software components, which are different from those to be used when the SLA related to a cloud storage service guaranteeing DoS protection must be implemented. Thus provisioning cloud services through “tailored” Security SLAs means to build a catalogue of software solutions able to implement different security policies and to offer them *as-a-service*. We refer to these software solutions as **security mechanisms**.

In order to automate the SLA implementation process, we propose the use of configuration management solutions as Chef [6] or Puppet [18]. These are widely adopted for the automation of installation and management of software over a pool of virtual machines (VMs). By using these configuration management tools, we can automate both the (1) dynamic acquisition of cloud resources and the (2) deployment and management of security mechanisms in a Software-as-a-Service (SaaS) fashion.

Security best practices prescribe the implementation of security controls (for example, those specified in the
NIST Security Control Framework [17] or the Cloud Security Alliance’s Cloud Control Matrix [7]) in order to cope with specific security risks. These controls are usually determined on the basis of a risk analysis process conducted by an expert, and represent the security policies to be set up in the system to detect/prevent possible incidents. As discussed in the previous section, in our model the security controls are used in an SLA to specify the declared security features of a cloud service. Specific controls or sets of controls (security capabilities) are implemented through suitable security mechanisms. Moreover, in our model, the catalogue of security mechanisms also includes solutions that can be deployed to monitor the security metrics specified in the SLA.

In the remainder of this section, we provide further details on each step of the SLA implementation process and thus demonstrate the feasibility of our approach. The architecture that supports this process is described in Section 5, while a concrete use case is presented in Section 6.

The SLA negotiation step is a template-based process. This means that the CSP maintains an SLA Template, based on the WS-Agreement, which specifies all provided services and all their supported features (all security policies that can be offered via the available security mechanisms). During the SLA negotiation process, the CSC selects the preferred cloud service and expresses security requirements in terms of security capabilities (i.e., in terms of security controls), and security metrics. The set of requirements is translated into an SLA Offer (i.e., a customized SLA Template, which contains only attributes and parameters selected by the CSC), and once both parties agree on the terms, the SLA Offer becomes a signed SLA.

After that the SLA agreement has been signed, the SLA enforcement phase starts. It is made up of the following steps:

2A. Identification of security mechanisms: First, the set of security mechanisms that enforce and monitor the terms agreed upon in the signed SLA has to be determined. The selection depends on the cloud service, the security capabilities and the security controls included in the signed SLA. An example of security mechanism may be a software solution that orchestrates pools of virtual machines devoted to web hosting and ensures a specific level of redundancy throughout their life to thwart possible denial of service attacks. This mechanism may be activated, for example, when the CSC selects, during negotiation, the security controls related to business continuity and contingency plan.

2B. Deployment specification: Next, the number of resources to acquire has to be defined. According to the considered model, secure cloud services are built by deploying and configuring ad hoc software solutions on top of infrastructure cloud services (virtual machines) acquired from existing cloud providers, while other types of services (e.g., SaaS services) are not considered. The number of VMs to be acquired depends on the combination of security mechanisms chosen in step 2A. The way security mechanisms work together is defined by the system administrator, and the configuration information for each mechanism is specified by its developer in a suitable metadata file, which will be discussed in the next section.

2C. Configuration specification: Each security mechanism selected in step 2A needs to be configured so as to enforce and monitor the signed SLA. For example, if the CSC required daily software vulnerability scans, the corresponding security mechanism has to be configured so that it activates scans every 24 hours. To this end, a configuration specification is prepared, which collects configuration details for each security mechanism to be deployed.

2D. Build an implementation plan: Once the security mechanisms to be deployed are identified, and the deployment and configuration specifications are set accordingly, an implementation plan is built that collects all this information.

2E. Acquisition of cloud resources: Cloud resources are automatically acquired according to the implementation plan.

2F. Registration of acquired cloud resources: In order to manage and monitor the acquired resources, they are registered in the configuration management system.

2G. Running the deployment and configuration procedure: The final step of the enforcement process involves deploying security mechanisms on the acquired resources and configuring them according to the implementation plan.

The proposed approach is briefly summarized in Figure 2. The presented functionalities are performed by three software components, namely the SLA Automator, the Broker, and the Configuration Manager. The SLA Automator is a web application that (1) enables CSCs to select cloud services and to negotiate the content of associated Security SLAs and (2) activates the Broker, which is responsible for the acquisition of external cloud resources. The Configuration Manager, also activated by the SLA Automator, orchestrates the deployment and configuration of security mechanisms, according to the implementation plans.

In practice, the configuration management tools are usually controlled by humans, in order to perform both the deployment of software instances on multiple resources and their synchronization. The novel and innovative aspect of our approach is that we give control over resources and deployed software entirely to the
SLA Automator. The Broker and the Configuration Manager components, which are orchestrated by the SLA Automator, rely only on the content of the signed Security SLA (in particular, on the content of the implementation plan built for the SLA). The SLA Automator provides them with an implementation plan, and they invoke the APIs exposed by CSPs to manage (acquire/release/configure) the resources needed to implement the service requested by CSCs.

The presented SLA implementation enables the complete automation of the Security SLA management, even if security policies are tailored to the needs of each customer. Further details about the software architecture implementing this approach are provided in Section 5.

5 Design of the SLA Implementation framework

In this section, we present the software architecture proposed to implement the approach illustrated in Figure 2, and provide some details on the design of the SLA Automator component. Moreover, we discuss the set of information that must be available in order to enable the implementation process.

As shown in Figure 3, the SLA Automator consists of four software components, namely the Application, the SLA Manager, the Services Manager and the Implementation component.

The Application component orchestrates all the other components by invoking their exposed APIs. It offers a web interface to customers, through which they can (1) negotiate the (security) features of the desired target service, (2) follow the whole implementation process and (3) monitor the state of their SLAs.

The SLA Manager provides the basic functionalities to create, store and retrieve SLAs, and keeps track of the current state of processed SLAs. A detailed description of the admissible SLA states and of the SLA API exposed by the SLA Manager can be found in [10], which presents the extension of the WS-Agreement standard by the introduction of additional features. As an example, the extended model devises that alerts are raised in case of deviations from the "normal" behaviour, to prevent a future violation of an SLA.

The SLA Manager manages a set of SLA Templates. As mentioned before, templates represent the security offers that are available to customers, and are used during negotiation, according to the WS-Agreement specification. In practice, a template is built by the system administrator (the owner of the framework) by accessing a database of information that includes:

- the set of cloud (IaaS) providers known to the Broker component, along with the respective connection and configuration parameters;
- the set of security capabilities related to the target service, for which a security mechanism has been developed;
- the security metrics related to the above security capabilities, which it is possible to monitor by means of available monitoring services (i.e., security mechanisms).

Once the template is ready, it is supplied to the SLA Automator, and used to build the SLA offers during
the interactive negotiation with the CSC. Available security mechanisms are listed in a catalogue, which can be enriched by any developer willing to offer some security features as-a-service. The security metrics to be included in the template are also taken from a catalogue. In particular, they belong to a Security Metric Catalogue, first introduced in SPECS and now also used in the MUSA project. The catalogue collects metrics from different sources (e.g., from metric standards, or from specific project results) and represents them in a homogeneous way (the current model is partly based on ISO 19086-2). Metrics are statically associated with specific controls in a database. During negotiation, the CSC is asked to select the metrics of interest among those related with selected controls.

The Service Manager is responsible for managing the information on available security capabilities and on the supported security mechanisms that can be activated 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.

During the development phase, it is possible to create new security mechanisms, and to add them to the pool of available security mechanisms. Moreover, security mechanisms may either enable the enforcement of some security features, or allow the monitoring of specific security metrics. A mechanism is represented in terms of:

- the covered capabilities,
- the enforced and monitored metrics,
- the set of software components implementing the mechanism,
- the deployment information and constraints.

The last two pieces of information are included in the mechanism metadata, and are used to build the implementation plan in the steps 2B, 2C and 2D discussed in the previous section. The deployment constraints, when present, refer to conditions that must be satisfied to configure correctly a mechanism. Let us consider, for example, a security mechanism related to the contingency planning security control, which manages a pool of (replicated) VMs devoted to web hosting and provides proxy functionalities to detect attacks and redistribute network traffic. In this case, a deployment constraint may establish that the proxy component is deployed on a separate machine. Such constraint should be taken into account when deciding how many machines must be acquired, and how to allocate the components on the machines. In [4], a linear programming model has been proposed to provide an optimal solution to the allocation problem.

Once the implementation plan is ready, the Implementation component is responsible for implementing it, by managing the acquisition and configuration of the resources needed to put in operation the service covered by the SLA. In order to acquire the needed resources, the Implementation component invokes the Broker, which interacts with public or private providers to buy resources on behalf of the customer. For each available provider, the Broker stores a JSON description reporting information about the provider’s name, the geographical zones of offered machines, the type of virtual machines, the firewall rules, etc.

The resource configuration is automated by a 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).

As mentioned before, the Configuration Manager has been implemented by exploiting Chef, one of the most popular cloud automation tools. Chef is a framework for the automation of cloud systems and infrastructures, which uses the so-called recipes to specify the infrastructure and the related configuration tasks. Recipes use building blocks called resources, representing pieces of an infrastructure, and included in a cookbook, which is the fundamental unit for the configuration and deployment of content on a target machine, referred to as a node. Chef cookbooks, as well as other configuration data, are stored in the Chef server, which is the main component of the Chef architecture, and may be either installed on local machines or invoked as a remote SaaS service. The Chef server communicates with Chef clients, which are installed on the network nodes.

It is important to point out that the target service, along with all requested security mechanisms, are installed and activated on the resources acquired and controlled by the customer. In general nothing can be said 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 along with all the other security mechanisms.

6 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 SPECS demonstrator application (available on-line on the SPECS web site). The target customers of this service are web developers aware of the main
security threats and able to formulate specific security requirements (possibly derived from best practices and guidelines), but needing support for identifying and implementing the required security countermeasures.

At the state of the art, the web developer, even if acquires VMs from a public CSP, is the only person in charge of setting up the security configuration. Appliances offering predefined services (for example, a pre-configured web server) do exist, but checking and comparing the security features offered by different CSPs is not an easy task. The web developer has to (i) find manually the security features provided by each CSP, (ii) evaluate and compare existing offers, (iii) apply a suitable configuration, if not natively supported, and (iv) implement a monitoring solution to verify the respect of the security features at runtime.

The SPECS ecosystem provides a turnkey solution to the above issues, as it (i) offers a single interface to choose among multiple offerings on multiple providers, (ii) enables the web developer to specify explicitly the needed security capabilities on the target web container, (iii) configures automatically the VMs in order to enforce the security controls requested, (iv) offers a set of security metrics to monitor the respect of the security features requested, (v) enables continuous monitoring of the security metrics negotiated, and (vi) can automatically remediate to (some of) the SLA alerts and violations that may occur.

After that the Secure Web Container service has been chosen (Figure 4), the SPECS demonstrator application guides the customer across the main phases of the SLA life cycle, namely Negotiation, Enforcement and Monitoring. First of all, a wizard 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 (Figures 5 and 6, respectively). The selection of the provider may be done on the basis of several criteria, including the analysis of the cost of offered services, or the evaluation of the quality of the providers from multiple points of view (including reliability or security). Such evaluation relies on the information collected by the Cloud Service Alliance (CSA) in the Security, Trust & Assurance Registry (STAR) program.

As for the security capabilities to be added to the web server, three capabilities are available:

- **Vulnerability Scanning.** 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 disrupting the system availability.

- **Web Resilience.** 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.

Each capability is associated with a set of security controls and with one or several security metrics, available to the customer for monitoring purposes. The customer can choose the metrics of interest and define suitable thresholds to specify SLOs. For example (Figure 7), the Web Resiliency capability has two metrics associated, namely Level of Redundancy (LoR), which identifies the number of aligned web server replicas kept active to counteract possible attacks against availability, and Level of Diversity (LoD), which represents the number of different web servers (e.g., Apache and Nginx) that will be actually running on the replicas. When specifying SLOs on top of such metrics, a customer may require, for example, that $LoR \geq 3$ and $LoD = 2$. This implies that at any time at least three replicas of the web server are kept alive, and at least two of them run different web server 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. 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.

- **DoSprotection.** It consists in a solution for denial of service attacks detection and mitigation based on the OSSEC tool.

Once the Security SLA negotiation is completed (taking into account both the security capabilities requested by customers and the security mechanisms available from provider), the SLA Automator, as described in Section 4, automatically identifies and acquires from the selected CSPs the amount of resources (i.e., the virtual machines) needed. Then it distributes the software components of the different mechanisms to each VM. Among the mechanisms started, there are the monitoring agents used to identify and (possibly) to prevent possible SLA violations. The customer can continuously monitor the web container through a simple interface that reports the actual measured service level, for each of the agreed SLOs.
Per-Service Security SLA for Cloud Security Management

Figure 4  The Secure Web Container Demonstrator Application - Service Selection

Figure 5  The Secure Web Container Demonstrator Application - CSP Selection

Figure 6  The Secure Web Container Demonstrator Application - Security Capabilities Selection
7 Conclusions

In this paper, we have investigated the adoption of CSC-based per-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.

References


