DOI 10.12694/scpe.v17i4.1200
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 17, Number 4, pp. 271–283. http://www.scpe.org

SLA-BASED SECURE CLOUD APPLICATION DEVELOPMENT

VALENTINA CASOLA*, ALESSANDRA DE BENEDICTIS†, MASSIMILIANO RAK‡, AND UMBERTO VILLANO§

Abstract. The perception of lack of control over resources deployed in the cloud may represent one of the critical factors for an organization to decide to cloudify or not its own services. The flat security features offered by commercial cloud providers to every customer, from simple practitioners to managers of huge amounts of sensitive data and services, is an additional problem. In recent years, the concept of Security Service Level Agreements (Security SLAs) is assuming a key role for the secure provisioning of cloud resources and services. This paper illustrates how to develop cloud applications that deliver services covered by Security SLAs by means of the services and tools provided by the SPECS framework, developed in the context of the SPECS (*Secure Provisioning of Cloud Services based on SLA Management*) European Project. The whole (SPECS) application's life cycle is dealt with, in order to give a comprehensive view of the different parties involved and of the processes needed to offer security guarantees on top of cloud services. The discussed development process is exemplified by means of a real-world case study consisting in a cloud application offering a secure web container service.

Key words: Secure cloud applications, Security Service Level Agreements, Automatic Enforcement of Security

AMS subject classifications. 68M14, 68Q85

1. Introduction. Nowadays, the adoption of the cloud computing paradigm is steadily spreading. The final step to convince the skeptics is the provision of solid security solutions for cloud applications and data. As a matter of fact, cloud resources are not permanently assigned to users and are not under the control of user software; they are just acquired *on-demand*. This is perceived as a security loss by some users, accustomed to have full control over all the resources involved in service delivery.

In the case of public clouds, the lack of full user control over resources is not the only security issue. Currently Cloud Service Providers (CSPs), who are the actual owners of the physical computing, storage and network resources hosted in their huge data centers, administer security according to common best-practice rules. Independently of the type of Cloud Service Customer (CSCs), they provide exactly the same security features. Most often, these features are simply *the best they can offer*. The very basic security guarantees offered are undoubtedly sufficient for a private computing practitioner, but surely not adequate for small enterprises or for publicly funded organizations managing, for example, healthcare and Personal Information (PI) data to be protected with specific security and privacy requirements.

The real problem is that security has a non-negligible cost, and so CSPs have no interest in offering such features to *every* CSC. To differentiate security features on a customer-by-customer basis is difficult, if not unfeasible. Currently there is actually a gap between CSCs, which look for "tailored" security features, possibly offered *on-demand* and *as-a-service*, exactly as other cloud resources, and CSPs, which offer security *as-a-whole*, integrated in the cloud services and transparently granted in the same way for all customers.

We deem that *Security Service Level Agreements* (SLAs) can play a key role for cloud security assessment, as they allow to declare clearly the security level granted by providers to customers, as well as the constraints posed to both parties (providers and customers). However, despite the strong interest recently shown in Security SLAs in the context of both academical research and industry and government-driven initiatives, their widespread adoption is not yet a reality. In 2011, ENISA published a report analyzing the use of security parameters in Cloud SLAs (mostly focused on the EC public sector) [1]. The report pointed out that, although security was considered by most respondents as a top concern, existing SLAs addressed only availability and other performance-related parameters, while security-related parameters were not taken into account. Since then the situation has not changed significantly, and Security SLAs are still far from being adopted by existing CSPs.

The framework developed in the context of the SPECS project [2] aims to promote the adoption of Security SLAs, by making it possible to develop applications offering cloud services controlled by such contracts. With

*DIETI, University of Naples Federico II, Napoli, Italy (casolav@unina.it)
†DIETI, University of Naples Federico II, Napoli, Italy (alessandra.debenedictis@unina.it)
‡DIII, Second University of Naples, Aversa, Italy (massiliano.rak@unina2.it)
§DING, University of Sannio, Benevento, Italy (villano@unisannio.it)

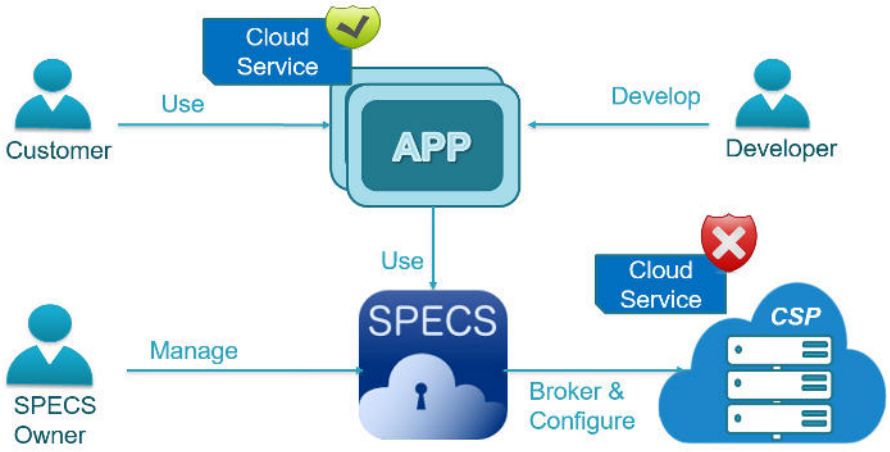

FIG. 2.1. *Overview of the SPECS solution*

SPECS, every cloud service is covered by a Security SLA that specifies the security grants offered, to be negotiated before cloud service delivery. Security features are automatically implemented by the SPECS framework according to the agreed SLA, and can be continuously monitored to verify that the SLA terms are actually respected.

The development of secure cloud applications by exploiting the SPECS framework was sketched in a previous paper [33]. In this paper, we provide a more comprehensive view of the SPECS applications' life cycle and discuss some of the tools that were developed to support it. Our exposition will go on as follows. In Sect. 2, we briefly introduce the SPECS framework and in Sect. 3 we describe the adopted Security SLA model. Sect. 4 illustrates the complete life cycle of a SPECS application, by discussing the methodology and tools adopted to enable the provisioning of secure cloud services based on SLAs. Sect. 5 discusses the introduced process with respect to a concrete example. Finally, Sect. 6 presents some related work and Sect. 7 reports our conclusions and plans for future work.

2. The SPECS framework. The SPECS project aims at designing and implementing a framework for the management of the whole Service Level Agreement life cycle, intended to build applications (SPECS applications) whose security features are stated in and granted by a Security SLA [3, 4].

The SPECS framework provides techniques and tools for: a) enabling user-centric negotiation of security parameters to be included in a Security SLA; b) enforcing an agreed Security SLA by automatically putting in place all security features needed to meet user requirements; c) monitoring in real-time the fulfillment of Security SLAs and notifying both users and CSPs of possible violations; d) reacting and adapting in real-time to fluctuations in the provided level of security (e.g., by applying proper countermeasures in case of an SLA violation).

As represented in Fig.2.1, the SPECS operation scenario involves four main parties:

- A **Customer** of the cloud services, offered by SPECS and covered by Security SLAs;
- The **SPECS Owner**, a provider of cloud services covered by Security SLAs;
- An **(External) CSP**, an independent (typically public) cloud service provider, which is unaware of the SLAs and offers just basic cloud resources and infrastructural services;
- A **Developer**, a cloud service partner that supports the SPECS Owner in the development and delivery of security-enhanced cloud services.

The Customer negotiates his/her security requirements with the SPECS Owner, who acts as a broker by acquiring resources from External CSPs and by reconfiguring/enriching them in order to fulfill the Customer's requests. This is accomplished by the activation and configuration of suitable software mechanisms and tools, provided in an *as-a-service* mode by SPECS. These mechanisms are automatically enforced on top of acquired resources, according to what has been agreed in a Security SLA. In the above process, the security-enhanced

services are delivered to end-users by a SPECS application, developed and deployed by exploiting the SPECS framework services, depicted in Fig. 2.2.

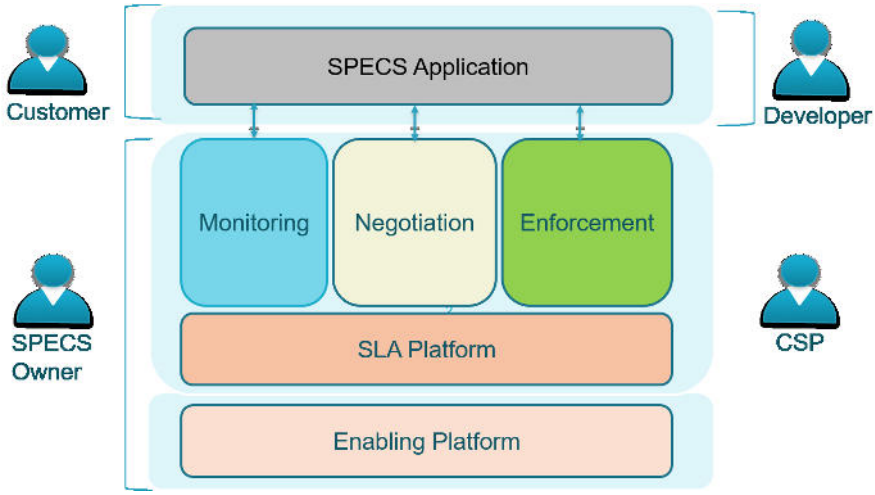

FIG. 2.2. *The SPECS framework*

A SPECS application orchestrates the SPECS *Core services* dedicated to SLA Negotiation, Enforcement and Monitoring, respectively, to provide the desired service (referred to as "Target Service") to the SPECS Customer (i.e., to the End-user). The Core services run on top of the *SPECS Platform*, which provides all the functionalities related to the management of Security SLA life cycle and needed to enable the communication among Core modules. In addition to this functionalities, referred to as "*SLA Platform* services", the SPECS Platform also provides support for developing, deploying, running and managing all SPECS services and related components [4]. These services are referred to as "*Enabling Platform* services".

3. The Security SLA model. As discussed in the previous section, the SPECS approach for *Security-as-a-Service* provisioning relies upon the idea that each cloud service is covered by a Security SLA, specifying related security-oriented terms and conditions, and that the cloud service delivery is controlled by the Security SLA life cycle. The Security SLA life cycle adopted in SPECS founds on and extends current standards on cloud SLAs (WS-Agreement [5], ISO19086 [6]) and consists of five phases: *Negotiation*, *Implementation*, *Monitoring*, *Remediation* and *Renegotiation*.

During the *Negotiation* phase, a cloud service customer and a cloud service provider carry out a (possibly) iterative process aimed at finding an agreement that defines their relationship as regards the delivery of a service. During the *Implementation*, the CSP provisions and operates the cloud service, but also sets up the processes needed for the management and monitoring of the cloud service, the report of possible failures and the claim of remedies. After the implementation of an SLA, the *Monitoring* phase takes place, where the service is continuously monitored to verify whether the SLA terms are respected. The Monitoring phase has also the responsibility of preventing, when possible, the violations, by rising alerts in presence of specific events. Alerts can be managed, during the *Remediation* phase, by reconfiguring the service while preserving the agreement, in order to avoid actual violations. If any SLA violation occurs, the cloud service customer may be entitled to a remedy (*Remediation* phase), which may take different forms, such as refunds on charges, free services or other forms of compensation. Finally, at any moment after implementation, either the cloud service customer or the cloud service provider may require a *Re-negotiation* of the SLA, aimed at changing any of its terms. The life cycle discussed above makes it possible to control cloud services according to SLA phases (and states). The interested readers are referred to [7] for a deeper analysis of the SLA life cycle and the description of a REST API for its management developed within the SPECS project, and to [8] for an illustration of some of the tools used in SPECS to monitor the SLA during the execution of a cloud service.

The negotiation, enforcement and monitoring of security-related terms are enabled by the adoption of a

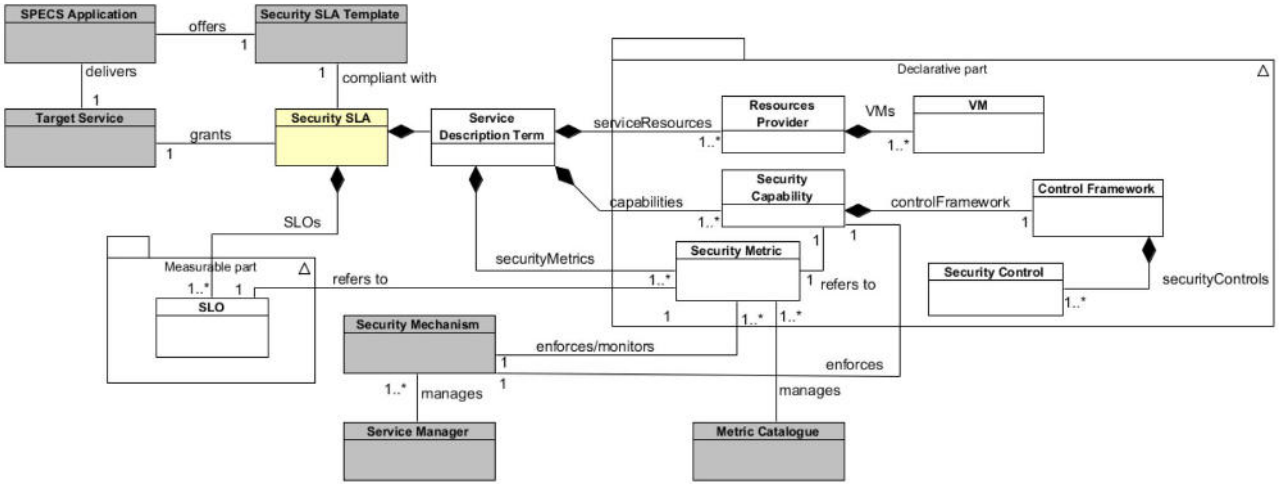

FIG. 3.1. *The SPECS Security SLA model*

novel Security SLA model, introduced in [32], which extends the WS-Agreement standard to include security concepts by taking into account both the End-user requirements and the technical offers from the providers' point of view.

The SPECS Security SLA model is depicted in Fig. 3.1, which shows more in general the *SPECS domain model*. In accordance with the WSAG standard, a Security SLA is compliant with a template (*Security SLA Template*). The template summarizes the available (and negotiable) offers and is used as a guideline during the negotiation of a *Target Service*. The whole process of the Target Service acquisition is managed by a *SPECS Application*, which is configured based on the template.

As depicted in Figure 3.1, a Security SLA (and the related template) consists of a *declarative* part and a *measurable* part. The former includes all the concepts that describe the service being delivered, both in functional and in non-functional terms. In particular, it reports the information regarding:

- the cloud resources used to build the *Target Service*. Note that, in SPECS, only Infrastructure-as-a-Service (IaaS) cloud resources are used (i.e., Virtual Machines, VMs), and the Target Service is built by properly deploying and configuring software components on the acquired VMs. For this reason, the SLA must contain the reference to the considered `Resource Provider(s)`, and the related offered `VMs`;
- the `Security Capabilities` [9] offered/required on top of the service covered by the agreement, each defined in terms of related security controls belonging to a `Security Control Framework` (NIST's Control Framework [9] and Cloud Security Alliance's Cloud Control Matrix [10] are currently supported);
- the `Security Metrics` that can be used to enforce (i.e., configure) and monitor different aspects of the declared security capabilities. Security metrics are specified in the *SPECS Security Metric Catalogue* and used to define security-related guarantees.

The measurable part of a Security SLA includes the specification of the guarantees expressed on the Target Service, represented by a set of Service Level Objectives (`SLOs`) built on top of the security metrics declared above. During negotiation, through the SPECS application, the End-user selects a subset of the available security capabilities, chooses the metrics of interest and defines SLOs on top of them.

The enforcement of security capabilities and the monitoring of related security metrics (as specified in the SLOs) is performed by software tools called *Security Mechanisms*: they are selected, deployed and configured during the *Implementation* phase. The *Service Manager* maintains all the information associated with available security mechanisms that are needed to automate their deployment and execution together with the Target Services.

4. Life cycle of a SPECS application. As discussed, a SPECS application enables an End-user to acquire an up-and-running secure cloud service after a negotiation process based on a pre-defined Security SLA template. The cloud service is delivered with specific security guarantees, which can be verified by the End-user

through monitoring functionalities, also made available by the SPECS platform.

In [33] we briefly illustrated the process of developing a SPECS application. In this paper, we aim at providing a more comprehensive view of the SPECS applications' life cycle that, at current state, is more mature and is supported by several tools. Like for any application, the SPECS application life cycle consists of three phases, i.e., (i) development, (ii) deployment, and (iii) execution. The actors involved in the first two phases are the SPECS Owner, who acquires the SPECS framework and uses it to offer secure services to his/her End-users, and the developer, at the service of the SPECS Owner, who is responsible for the implementation of the software tools needed to build secure services. The execution phase involves the interaction between the application and the End-users during the negotiation, enforcement and monitoring phases.

In the following, we discuss in detail the SPECS application's life cycle, with the aim of providing the reader with a deep understanding of the steps and tools needed to deliver secure services based on Security SLAs.

4.1. Development of a SPECS application. In order to support the development process, the SPECS framework provides a *default SPECS application* in the form of servlets for Apache Tomcat, which includes the basic functionalities to orchestrate the SPECS core services and enable the negotiation, enforcement and monitoring of an SLA, independently of the service to offer. To provide a specific Target Service, the developer must customize the default application by configuring a set of additional services that implement both the functionalities (e.g., a web container service, a database service) and the security features that the SPECS Owner is willing to offer. In order to automatize the deployment of such mechanisms, SPECS uses a cloud automation technology, represented by the *Chef* deployment solution [12], which automates the process of building, deploying, and managing software over ICT infrastructures. With Chef, it is possible to automate the deployment and configuration of a given software component on a resource such as a VM by specifying the operations to perform inside a *recipe*. Recipes are collected in *cookbooks* and stored in a Chef Server, which is responsible for launching their execution on specific nodes (hosting a Chef Client) in order to configure them. The SPECS Enforcement module includes a Chef Server, which is responsible for the set-up, at run time and based on an SLA, of all the software components needed to deliver a negotiated cloud service along with required security and monitoring mechanisms. Hence, customizing the SPECS default application implies supplying to the Enforcement module the needed cookbooks for each mechanism to support, and providing it with all the information needed to automatically configure the mechanism during the SLA implementation phase (i.e., the *mechanism descriptor*, described later).

FIG. 4.1. *SPECS application development process*

The whole development process is depicted in Fig. 4.1. It consists in the following steps:

1. **Cloud Service Definition**: the developer identifies the functionalities that should be offered by the application (e.g., web containers, databases) and implements (or retrieves, if already available) the software mechanisms that provide them *as-a-service*. Moreover, for these mechanisms, the developer prepares related cookbooks.
2. **Security Mechanisms Definition**: the developer identifies the security capabilities that should be offered by the application over the cloud services defined at the previous step, and implements (or retrieves, if already available) the related security mechanisms. Afterward, the developer prepares the mechanisms' cookbooks. Moreover, for each mechanism, the developer has to prepare a *mechanism descriptor* that specifies:
   - the granted security capabilities (and related security controls);
   - the enforceable/monitorable security metrics (and related measurements, representing the actual parameters gathered to check the identified metrics);

- the monitoring events associated with reported measurements, used by the Enforcement module (Diagnosis component) to detect violations or alerts related to an SLA;
- the mechanism's metadata, which includes information on the software components implementing the mechanism and on respective deployment constraints (e.g., incompatibility or dependency of software components implementing the mechanism, used during the mechanism's deployment).

This information is used during the whole SLA life-cycle, since it enables to negotiate capabilities, select available metrics, configure and monitor related mechanisms and detect and manage related alerts or violations.

3. **Security SLA Template Preparation**: once all the mechanisms needed to build the target cloud service have been defined and set-up, the developer prepares an SLA template, compliant with the model discussed in Sect. 3, which summarizes all available features.

It is worth noticing that the SPECS application development mainly focuses on the development of ad-hoc Chef cookbooks for the security mechanisms to be offered. When cookbooks are already available (there are many archives of already-developed cookbooks), the only additional work consists in the preparation of the metadata and SLA templates used to automate the SLA implementation.

4.2. Deployment of a SPECS application. The deployment of a SPECS application is performed through the SPECS Platform Interface, publicly available at [34]. The functionalities available to the SPECS Owner by means of the Platform Interface are reported in the Use Case diagram of Fig. 4.2.

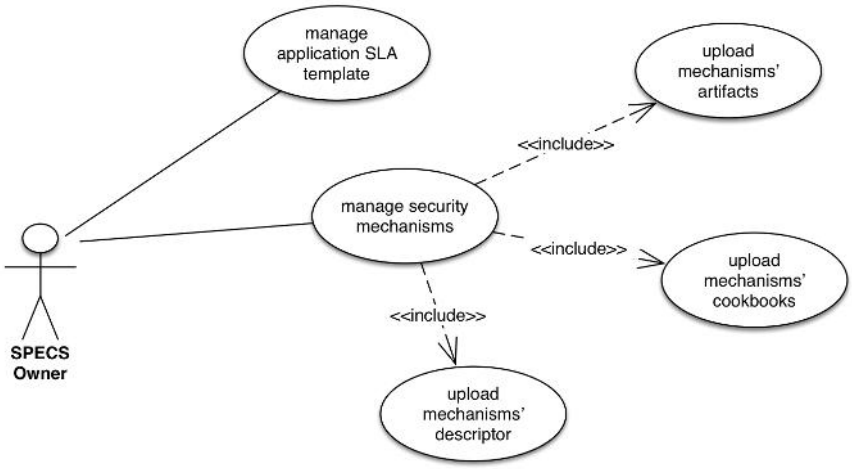

FIG. 4.2. *SPECS Owner use case diagram*

By selecting the "SPECS Services management" tab (see Fig. 4.3), the application allows the SPECS Owner to manage all the available (secure) services, along with related capabilities and security mechanisms. As discussed before, each Security Service is identified by an SLA template, prepared by the developer during the application development phase. At deployment time, the SPECS Owner must provide the template to the Negotiation module via the interface offered by the dashboard. It will be used during negotiation for the generation of the SLA Offers and during enforcement for the configuration of the Monitoring module based on included SLOs. Moreover, at deployment time, the SPECS Owner has to provide the Enforcement module with the cookbooks previously prepared for all supported mechanisms and with related artifacts. Finally, the SPECS Owner has to make available the mechanisms' descriptors to the SLA Platform, in order to enable their automatic deployment and configuration based on an SLA.

4.3. Execution of a SPECS application. A running SPECS application comes in form of a wizard that enables the End-user to negotiate, implement and monitor an SLA (cf. Fig. 4.4).

First, the negotiation wizard allows the End-user to select the security capabilities to activate. Related to these capabilities, the subsequent steps require the selection of the security metrics of interest and the definition

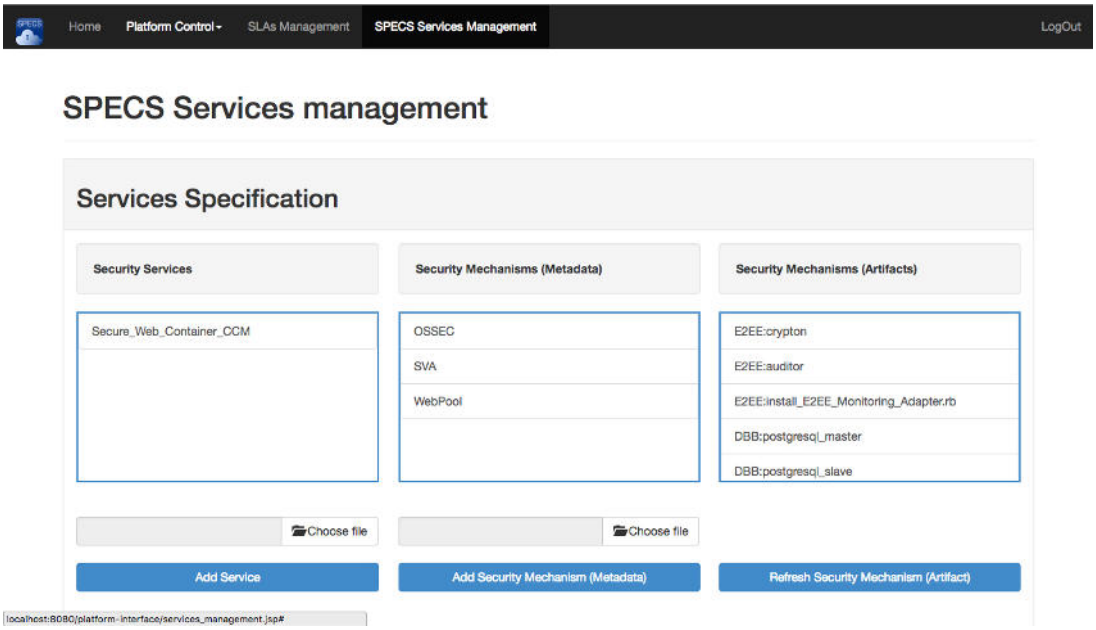

FIG. 4.3. *SPECS platform interface*

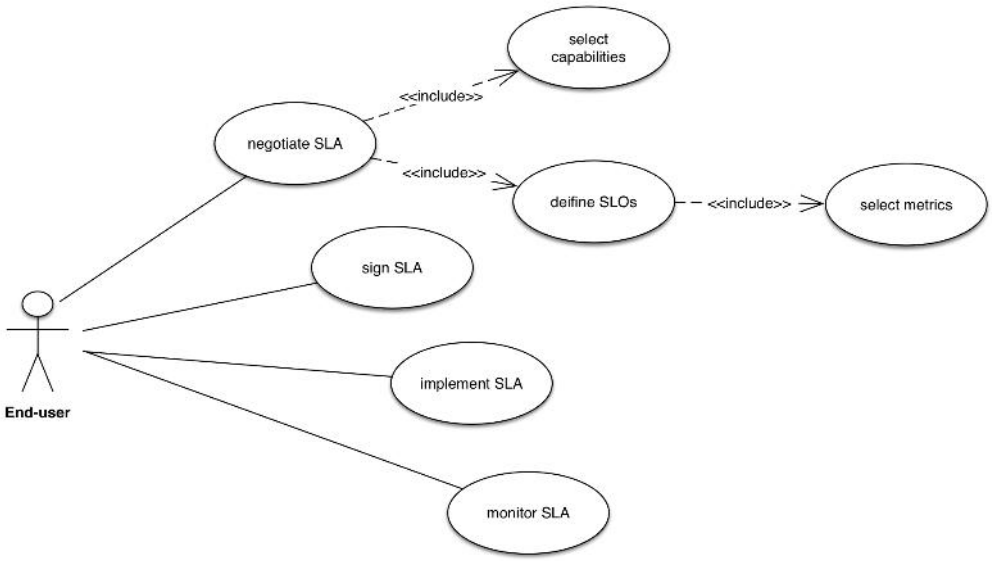

FIG. 4.4. *End-user use case diagram*

of the SLOs. Currently the negotiation focuses only on the SPECS-supported Security SLOs. However, it is possible to extend it to other non-functional SLOs. At the end of this process, the End-user can formally accept the SLA (i.e., sign it) and proceed with its implementation.

During implementation, the SPECS application orchestrates the Enforcement module's services to acquire the needed resources from external providers and to configure them with (i) the security mechanisms that implement the security capabilities included in the SLA and with (ii) the monitoring systems able to monitor the metrics reported there.

After the implementation, the SPECS application provides the End-user with a monitoring dashboard,

through which he can verify the values of the metrics and check the correct fulfillment of the SLA.

In the following section, we will illustrate the above discussed process with respect to a concrete application offering a secure web container.

5. A secure web container service. As an example of cloud service that may be enhanced through SPECS, let us consider a web container solution. An example of such a solution is Amazon AWS Elastic Beanstalk [35], which allows to quickly deploy and manage applications in the AWS cloud infrastructure. This solution, which is very complex indeed, provides support for different programming languages and web containers, and comes with dedicated security management tools developed by Amazon.

When considering smaller providers, it is reasonable to suppose that they would offer more simple platforms for web applications management, with very limited security features. A web developer targeting such providers but with specific security requirements should get on all the responsibility of managing them by developing and integrating ad-hoc software tools inside his/her applications.

It should be noticed that, at the state of the art, existing appliances offer predefined services (for example, a pre-configured web server), but checking and comparing the security features offered by different CSPs is not an easy task. The web developer has to (i) manually find 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 at runtime the respect of the security features.

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) automatically configures 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) alerts and violations that may occur to the SLA associated to the web container.

Below we will present the development of the secure web container as a SPECS application, following the steps dealt with in the previous section.

5.1. Cloud Service Definition. The main goal of this case study is to deliver web containers that an end-user can acquire by negotiating his/her desired security features. To this aim, we developed a mechanism named *WebContainerPool* (WEBPOOL) that not only provides the web container as a cloud service but also offers some basic security-hardening features on top of it. In particular, the mechanism allows to acquire a pool of virtual machines and to configure them with several replicas of the web container with different software solutions (e.g., Apache Tomcat, NGINX, Jetty), in order to ensure resiliency to failures through sw diversity and redundancy. Moreover, the mechanism enables to configure such replicas each with a different software solution, and to randomize the handling of incoming requests among the available replicas.

In practice, the *WebContainerPool* mechanism has been developed as a security mechanism, but it is mandatory for the set-up of the web container service delivered by the application. The information related to the mechanism has been included in the WEBPOOL mechanism descriptor, which specifies:

- the capability provided (*Web Resilience*),
- the metrics associated ((i) `LevelofRedundancy` and (ii) `LevelofDiversity`),
- the monitoring events that can be detected by the Monitoring module and handled by the Enforcement module for remediation activities (e.g., a web container replica is down),
- the actions to perform to prevent and manage violations (e.g., acquire and configure a new machine), and
- the mechanisms' metadata including the software components that implement it (i.e., Apache and Ngnix web containers and a load balancer based on HAProxy) and their deployment contraints (e.g., the load balancer must be hosted by a separate machine).

The Chef cookbook associated to the *WebContainerPool* mechanism can be used also independently of the SPECS framework and is available at [36]. It is worth pointing out that, if the aim is to apply the same process to a different cloud service (e.g., a Secure CMS), it is first necessary to develop a Cloud service cookbook dedicated to offer the CMS, and later on to select the security mechanisms that can be offered for it, possibly developing custom ones.

5.2. Security Mechanisms Definition. The proposed service (web container), as outlined above, relies on (a pool of) virtual machines, hosting synchronized web servers. The service offers some integrated security features (redundancy and diversity), but a lot of additional security capabilities can be provided. In SPECS, three main security mechanisms are already available to enhance the web container service:

- **TLS**: it is a preconfigured TLS server, configured according to security best practices.
- **SVA (Software Vulnerability Assessment)**: it regularly performs vulnerability assessment over the virtual machines, through software version checking and penetration tests.
- **DoSprotection**: it consists of a solution for denial of service attacks detection and mitigation based on the OSSEC tool.

The main role of the developer is to select the mechanisms to be provided together with the WEBPOOL mechanism from the catalogue of available security mechanisms (mantained by the SPECS Service Manager). If the developer is interested in offering additional security mechanisms, and/or in enforcing security metrics and capabilities not yet supported in SPECS, he has to implement the mechanism by releasing related artifacts, to prepare a mechanism's descriptor in the proper format and to develop a cookbook for it. Let us assume that the *SVA* and *DoSprotection* mechanisms are to be offered. The main information related to these mechanisms is provided in Table 5.1 and in Table 5.2, which respectively report the security controls and the metrics associated with them, these can be offered and guaranteed in the SLA. Moreover, Table 5.1 reports also the information that the WEBPOOL mechanism is required for the set-up of the service, while the others are optional. Table 5.2 reports the measurable information associated with the three mechanisms that will be offered in the SLA during the negotiation phase. For the sake of brevity, we do not report here all the information included in the mechanisms' descriptors. The interested reader is referred to the SPECS Bitbucket repository [15] for all the details.

TABLE 5.1
Definition of the capabilities to offer with the web container

ID	Name	Description	Req.	Controls
WEBPOOL	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 diversity	yes	CONTINGENCY PLAN - CP-2
				HETEROGENEITY - SC-29
				DISTRIBUTED PROCESSING AND STORAGE - SC-36
				DENIAL OF SERVICE PROTECTION - SC-5
OSSEC	DOS Detection and Mitigation	Capability of detecting and reacting to security attacks aimed at disrupting a system's availability	no	DENIAL OF SERVICE PROTECTION - SC-5
				DENIAL OF SERVICE PROTECTION - SC-5(3)
				CONTINUOUS MONITORING - CA-7
				INFORMATION SYSTEM MONITORING - SI-4
SVA	Software Vulnerability Assessment	Capability of detecting and mitigating vulnerabilities	no	CONTINUOUS MONITORING - CA-7
				CONTINUOUS MONITORING — TREND ANALYSES - CA-7(3)
				PENETRATION TESTING - CA-8
				VULNERABILITY SCANNING - RA-5
				VULNERABILITY SCANNING — UPDATE BY FREQUENCY - RA-5(1)

5.3. Security SLA Template Preparation. This is the main developer task, as it summarizes all the possible offers to the End-user. Once the template is available, the SPECS application execution is fully automated. WS-Agreement templates are written according to the SLA model proposed in Section 3, following the WS-Agreement schema and the SPECS security extensions. The XML schema corresponding to our Security SLA model is available at [13], while the complete SLA template for the SPECS Web Container application is available at [37].

5.4. Application deployment. To complete the deployment of the case study application, the security mechanisms and the template have to be deployed.

TABLE 5.2
Definition of the security metrics associated with the mechanisms

Name	**Unit**	**Value Type**	**Metric Type**	**Description**
Level of Redundancy	n/a	integer	Quantitative/ Ratio	Number of replicas of a software component that are set-up and kept active during system operation
Level of Diversity	n/a	integer	Quantitative/ Ratio	Number of different versions of a software component that are set-up and kept active at the same time during system operation
Scanning Frequency - Basic scan	hours	integer	Quantitative/ Ratio	Frequency of the basic software vulnerability scanning.
Age of scanning report - Basic scan	hours	integer	Quantitative/ Ratio	Age of the scanning report (basic scan)
Vulnerability list availability	n/a	yes/no	Qualitative/ Nominal	Availability of the vulnerability list
Scanners availability	n/a	yes/no	Qualitative/ Nominal	Availability of the installed scanners
List Update Frequency	hours	integer	Quantitative/ Ratio	Frequency of updates of the list of known/disclosed vulnerabilities from OVAL/NVD databases
Age of vulnerability list	hours	integer	Quantitative/ Ratio	Age of the vulnerability list
Vulnerability repository availability	n/a	yes/no	Qualitative/ Nominal	Availability of the vulnerability repository
Scanning Frequency - Extended Scan	hours	integer	Quantitative/ Ratio	Frequency of the extended software vulnerability scanning. Scanning is performed with two scanners and both scanning reports are joint
Age of scanning report - Extended Scan	hours	integer	Quantitative/ Ratio	Age of the scanning report (extended scan)
Up Report Frequency	hours	integer	Quantitative/ Ratio	Frequency of checks for updates and upgrades of vulnerable installed libraries.
Age of the update/upgrade report	hours	integer	Quantitative/ Ratio	Age of the update/upgrade report
Availability of the scanning report	n/a	yes/no	Qualitative/ Nominal	Availability of the scanning report
Availability of the update/upgrade report	n/a	yes/no	Qualitative/ Nominal	Availability of the update/upgrade report
Penetration Testing Activated	n/a	yes/no	Qualitative/ Nominal	Activation of the penetration testing activity
DDoS Attack Detection Scan Frequency	hours	integer	Quantitative/ Ratio	Frequency of the dDoS attack report generation
Age of dDoS report	hours	integer	Quant/Ratio	Age of the dDoS attack scanning report

As said, this operation is accomplished by the SPECS Owner through the Platform Interface and entails that:

- All cookbooks of the chosen security mechanisms are added to the Chef repository associated to SPECS implementation component.
- All cookbook metadata are made available on the SLA Platform, which offers a simple REST API to upload such data and check them.
- The application template is uploaded to the Negotiation module.

The above example is available online as demonstrator application at [14].

6. Related Work. The large adoption of cloud computing solutions in a wide variety of domains opens several security issues: customers must often face the loss of control over their data and have to trust that their applications are securely executed on providers' resources. As pointed out in [16], many *ad hoc* security solutions have been proposed to cope with these issues, but they are often not portable and even not useful in different contexts.

Recent approaches are trying to address security problems from the start, i.e., at application design time, since it is hard (and ineffective sometimes) to add security features to an existing application *a posteriori*. Mohammadi *et al.* are working on the development of applications that can provide *trustworthiness* (the assurance that the system will perform as expected [17]) by design [18, 19]. The idea is to design the software in a way so that there will be mechanisms to ensure, evaluate and monitor trustworthiness, relying on reusable development process building blocks, consisting of method descriptions (guidelines, patterns and check-lists) ensuring that the right mechanisms are put in place to ensure trustworthiness.

However, as discussed previously in this paper, effective solutions exist that enable to profitably enhance the level of security of a cloud application by adopting a Security-as-a-service approach. In particular, the FP7-ICT Programme project SPECS addressed cloud security and proposed an open source development framework and a running platform dedicated to offer Security-as-a-Service using an SLA-based approach, by enabling negotiation, continuous monitoring and enforcement of security [3, 4].

As said, SPECS strongly relies upon Security SLAs. The definition of Service Level Agreement is an active topic for standardization bodies, because they are at the interface between cloud user needs and the services and features that cloud service providers (CSPs) are able to offer. The European Commission has set up a dedicated Working Group (CSIG-SLA) to cope with the definition and usage of SLAs, whose first result was a guideline for standardization bodies; a more advanced state of SLA standardization is offered by ISO 19086 [6], which proposes a standard for SLAs in clouds. However, standards for the definition of the security terms in an SLA are still lacking, even if there is currently a lot of ongoing work by dedicated groups (as the CSIG itself and the CSCC SLA group [20]) and research projects (see CUMULUS [21], A4Cloud [22], and SPECS [3]) on the topic.

Despite the strong impact that the introduction of Security SLAs may have on providers' profit, the main commercial IaaS providers (Amazon, Rackspace, GoGRID, etc.) currently still do not offer negotiable Security SLAs (see [28] for a survey of the SLAs offered by commercial cloud providers).

Regarding the configuration of security requirements specified through SLA documents, a few proposal exist. Karjoth et al. [29] introduce the concept of Service-Oriented Assurance (SOAS). SOAS adds security providing assurances (an assurance is a statement about the properties of a component or service) as part of the SLA negotiation process. Smith et al. [30] present a WS-Agreement approach for a fine-grained security configuration mechanism to allow an optimization of application performance based on specific security requirements. Brandic et al. [31] present advanced QoS methods for meta-negotiations and SLA-mappings in Grid workflows.

7. Conclusions and Future Work. In this paper, we illustrated a solution to develop cloud applications offering *secure* services covered by Security SLAs. The proposed solution relies upon a framework of services and tools released in the context of the SPECS EU Project and founds on the adoption of a novel Security SLA model, based on WS-Agreement standard and compliant with current standards and guidelines provided by the NIST and by CSA.

The paper provided a detailed discussion of the methodology followed in SPECS to build secure cloud applications and of the tools introduced and leveraged to support their life cycle. The discussion was also supported by the application of the proposed methodology to a real-world case study, for which a prototype implementation is available.

Our plans for future research include the development of new security mechanisms to enhance existing cloud services and the support for a wider set of cloud service types (at current state, only storage and web container services are considered). Moreover, we plan to include more sophisticated functionalities in the SPECS default application, such as the reasoning on the security level associated with different SLA offers, in order to enable customers to make a selection among different possible offers.

The SPECS applications may be deployed and offered by Cloud Service Providers, in order to define and agree on SLAs with their customers, but even by third-party providers that can act as brokers of services to enrich security capabilities of larger providers. This last delivery model may open new business opportunities, especially in those contexts (i.e., public sectors) where security represents the key factor to decide to cloudify a service.

Acknowledgments. This work has been partially supported by the FP7-ICT-2013-10-610795 (SPECS).

REFERENCES

[1] M. Dekker and G. Hogben, *Survey and analysis of security parameters in cloud SLAs across the european public sector,* Technical report. European Network and Information Security Agency, 2011.

[2] SPECS Consortium, *SPECS Project Web Site.* [Online]. Available: http://www.specs-project.eu

[3] M. Rak, N. Suri, J. Luna, D. Petcu, V. Casola, and U. Villano, *Security as a service using an SLA-based approach via SPECS,* in Proc. of the 2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom 2013), 2013, pp. 1–6.

[4] V. Casola, A. De Benedictis, M. Rak, and U. Villano, *Preliminary design of a platform-as-a-service to provide security in cloud,* in Proc. of the 4th International Conference on Cloud Computing and Services Science (CLOSER 2014), 2014, pp. 752–757.

[5] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu, *Web services agreement specification (WS-Agreement),* The Global Grid Forum (GGF), 2004.

[6] International Organization for Standardization, *ISO/IEC NP 19086-1. Information Technology – Cloud Computing – Service Level Agreement (SLA) Framework and Technology – Part 1: Overview and Concepts*, 2014.

[7] A. De Benedictis, M. Rak, M. Turtur, and U. Villano, *REST-based SLA Management for Cloud Applications,* in Proc. of the 2015 IEEE 24th International Conference on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2015), 2015, pp. 93–98.

[8] V. Casola, A. De Benedictis, and M. Rak, *Security monitoring in the cloud: an SLA-based approach,* in Proc. of the 2015 10th International Conference on Availability, Reliability and Security (ARES 2015), 2015, pp.749–755.

[9] NIST, *NIST Special Publication 800-53 Revision 4: Security and Privacy Controls for Federal Information Systems and Organizations*, 2013.

[10] Cloud Security Alliance, *Cloud Control Matrix v3.0*, https://cloudsecurityalliance.org/download/cloud-controls-matrix-v3/

[11] NIST, *NIST Special Publication 500-307 Draft: Cloud Computing Service Metrics Description*, 2015.

[12] Chef, *Chef Tool Web Site.* [Online]. Available: http://www.chef.io/chef/

[13] SPECS Consortium, *SPECS Security SLA Model.* [Online]. Available: http://www.specs-project.eu/schema

[14] SPECS Consortium, *SPECS Web Container Demo Application.* [Online]. Available: http://apps.specs-project.eu/specs-app-webcontainer-demo_CCM/

[15] SPECS Consortium, *SPECS Bitbucket Repository.* [Online]. Available: https://bitbucket.org/specs-team/

[16] C. A. Ardagna, R. Asal, E. Damiani, and Q. H. Vu, *From security to assurance in the cloud: a survey,* in ACM Computer Survey, vol. 48, no. 1, pp. 2:1–2:50, Jul. 2015.

[17] A. Aviienis, J.-C. Laprie, B. Randell, and C. Landwehr, *Basic concepts and taxonomy of dependable and secure computing,* in IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 1, pp. 11–33, 2004.

[18] F. Di Cerbo, P. Bisson, A. Hartman, S. Keller, P. Meland, M. Moffie, N. Mohammadi, S. Paulus, and S. Short, *Towards trustworthiness assurance in the cloud,* in Cyber Security and Privacy, ser. Communications in Computer and Information Science, M. Felici, Ed. Springer Berlin Heidelberg, 2013, vol. 182, pp. 3–15.

[19] N. Mohammadi, T. Bandyszak, S. Paulus, P. Meland, T. Weyer, and K. Pohl, *Extending software development methodologies to support trustworthiness-by-design,* in Proc. of the CAiSE 2015 Forum co-located with 27th International Conference on Advanced Information Systems Engineering (CAiSE 2015), 2015, pp. 213–220.

[20] CSCC, *The CSCC practical guide to cloud service level agreements,* Technical report*, 2012.* [Online]. Available: http://www.cloudstandardscustomercouncil.org/webSLA-download.htm

[21] A. Pannetrat, G. Hogben, S. Katopodis, G. Spanoudakis, and C. Cazorla, *D2.1: Security-aware SLA specification language and cloud security dependency model.* Technical report, Certification Infrastructure for Multi-Layer Cloud Services (CUMULUS), 2013.

[22] S. Pearson, *Toward accountability in the cloud,* in Internet Computing, IEEE, vol. 15, no. 4, pp. 64–69, July 2011.

[23] Contrail Consortium, *Contrail Project Web Site.* [Online]. Available: http://www.contrail-project.eu

[24] mOSAIC Consortium, *mOSAIC Project Web Site.* [Online]. Available: http://www.mosaic-cloud.eu

[25] Optimis Consortium, *Optimis Project Web Site.* [Online]. Available: http://www.optimis-project.eu

[26] PaaSage Consortium, *PaaSage Project Web Site.* [Online]. Available: http://www.paasage.eu

[27] R. Kübert, G. Katsaros, and T. Wang, *A RESTful implementation of the WS-Agreement specification,* in Proceedings of the Second International Workshop on RESTful Design (WS-REST '11) ACM, 2011, pp. 67–72.

[28] L. Wu and R. Buyya, *Service Level Agreement (SLA) in Utility Computing Systems,* in Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions, IGI Global, USA, 2011, pp. 1–25.

[29] G. Karjoth, B. Pfitzmann, M. Schunter, and M. Waidner, *Service-oriented assurance, comprehensive security by explicit assurances,* in Quality of Protection, ser. Advances in Information Security, D. Gollmann, F. Massacci, and A. Yautsiukhin, Eds., vol. 23. Springer US, 2006, pp. 13–24.

[30] M. Smith, M. Schmidt, N. Fallenbeck, C. Schridde, and B. Freisleben, *Optimising Security Configurations with Service Level Agreements,* in Proc. of the 7th International Conference on Optimization: Techniques and Applications (ICOTA 2007).IEEE Press, 2007, pp. 367–381.

[31] I. Brandic, D. Music, S. Dustdar, S. Venugopal, and R. Buyya, *Advanced QoS methods for Grid workflows based on meta-negotiations and SLA-mappings,* in Proc. of the 2008 Third Workshop on Workflows in Support of LargeScale Science, 2008, pp. 1–10.

[32] V. Casola, A. De Benedictis, M. Rak, J. Modic, and M. Erascu, *Automatically enforcing Security SLAs in the Cloud,* in IEEE Transactions on Services Computing, 2016, PrePrints: doi:10.1109/TSC.2016.2540630.

[33] V. Casola, A. De Benedictis, M. Rak and U. Villano, *SLA-Based Secure Cloud Application Development: The SPECS Framework,* in Proc. of the 2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2015, pp. 337–344.
[34] SPECS Consortium, *The SPECS Platform Interface.* [Online]. Available: http://apps.specs-project.eu:8080/platform-interface/.
[35] Amazon, *Amazon AWS Elastic Beanstalk Home Page.* [Online]. Available: https://aws.amazon.com/it/documentation/elastic-beanstalk/.
[36] SPECS Consortium, *The WEBPOOL mechanism cookbook.* [Online]. Available: https://bitbucket.org/specs-team/specs-mechanism-enforcement-webpool.
[37] SPECS Consortium, *The SPECS Secure Web Container Application Template.* [Online]. Available: https://bitbucket.org/specs-team/specs-utility-xml-sla-framework.

Edited by: Dana Petcu
Received: May 10, 2016
Accepted: July 17, 2016