A Novel Approach to Manage Cloud Security SLA Incidents

Ruben Trapero*[1], Jolanda Modic[§], Miha Stopar[§], Ahmed Taha*, Neeraj Suri*
*Department of Computer Science, Technische Universität Darmstadt, Darmstadt, Germany, 64289
E-mail: {rtrapero, ataha, suri}@deeds.informatik.tu-darmstadt.de
[§]XLAB d.o.o., Pot za Brdom 100, 1000 Ljubljana, Slovenia
E-mail: {jolanda.modic, miha.stopar}@xlab.si

Abstract. – Cloud computing is increasingly playing an important role in the service provisioning domain given the economic and technological benefits it offers. The popularity of cloud services is increasing but so are their customers' concerns about security assurance and transparency of the Cloud Service Providers (CSPs). This is especially relevant in the case of critical services that are progressively moving to the cloud. Examples include the integrated European air traffic control system or public administrations through the governmental clouds. Recent efforts aim to specify security in cloud by using security service level agreements (secSLAs). However, the paucity of approaches to actually control the fulfillment of secSLAs and to react in case of security breaches, often results in distrust in cloud services. In this paper, we present a solution to monitor and enforce the fulfillment of secSLAs. Our framework is able to (a) detect occurrences that lead to unfulfillment of commitments, and (b) also provide mitigation to the harmful events that may or do compromise the validity of secSLAs.

Highlights:

- ***A novel specification model to specify security information in SLAs.***
- ***Quantification of SLAs with development of metrics and measurements for attributes.***
- ***An SLA monitoring approach to detect events causing SLA violations.***
- ***An automated reaction schema for addressing anomalies causing SLA violations.***
- ***Specification of remediation actions to avoid or mitigate SLA violations.***

Keywords: security SLAs, cloud computing, cloud security, SLA monitoring, SLA remediation

1. Introduction

The economic and technological benefits of cloud computing are noticeable with an increasing number of CSPs offering diverse cloud-enabled services, and also the increasing number of actors, such as companies or governments that are moving their infrastructures or services to the cloud [1]. A recent survey [2] showed that 82% of potential customers consider security to be a major consideration for deciding to use a cloud service. Furthermore, critical services or services that handle sensitive information are moving to the cloud. This is the case of the European Organization for the Safety of Air Navigation (Eurocontrol) that is leveraging a cloud approach [3] to implement the integrated European air traffic control systems (SESAR) [4]. Also, an increasing number of governments are using the cloud to interact with their citizens [5]. As a result, security assurance is becoming even more relevant when referring to services moving to the cloud and customers willing to use them.

Many public *Cloud Service Providers* (CSPs) have adopted *security control frameworks* to represent the degree of security assurance and transparency provided to *Cloud Service Customers* (CSCs), e.g., ISO/IEC 27002 [6], the CSA's Cloud Control Matrix (CCM) [7], and the NIST's SP 800-53 [8]. By implementing their security control frameworks, CSPs can only assume the type of data that a CSC is going to generate and use, and are not aware of the requirements and controls that are necessary to actually protect CSC's data [9]. Thus, CSPs and CSCs need mechanisms and tools that enable them to understand and guarantee a demanded level of security. To this end, the cloud community (e.g., ENISA [10], ISO/IEC [11], NIST [8], and EC [13]) has moved towards the specification of security parameters in security service level agreements (termed as secSLA in the literature). SecSLAs are useful for defining common security related semantics among CSPs and CSCs.

However, the establishment of secSLAs only partially serves the needs of CSPs and CSCs especially if it is not linked to the management of the secSLA commitments. CSPs need tools and mechanisms

[1] Corresponding author

to react to eventualities that may threaten the fulfillment of the secSLA commitments (e.g., attacks, disasters, changes or regulation). The efficient detection and reaction to potentially harmful security related events becomes an essential activity to be carried out by CSPs in order to provide CSCs with a trustworthy service and also fulfill the agreed assurance levels. Thus, secSLAs become an essential element in this process. In the QoS context, SLAs typically include indicators that are subject to a progressive degradation that announce a potential violation of the SLA (for example, disk failures that might anticipate a degradation of the performance indicator of the cloud service). However, in the security context, the detection of such eventualities is something mostly unexplored. Typically, security indicators lack the aforementioned progressive degradation; for example, if an attacker breaks an encryption key, there is no apparent indicator that announces the potential security breach and the protected information is automatically exposed. In such situation, the minimization of the damages and its automated remediation is crucial. To this end, secSLAs are used in this paper for managing minimization of damages by providing with an automated remediation procedure supported by the continuous monitoring of secSLAs that enable detection of and reaction to potential and actual violations of the agreements.

The secSLA remediation approach presented in this paper was partially developed over the FP7-ICT project SPECS [37], [38], whose main objective is to provide a framework for automated management of secSLAs and enhancing CSPs' security offers by automatically deploying security mechanisms that implement security features negotiated by CSCs. All processes and the secSLA model presented in this paper have been designed and implemented by the SPECS team. The code with descriptions is available at the project's Bitbucket account [48].

1.1 Contributions

This paper addresses issues for trust and security assurance of cloud services by advocating an approach to manage secSLAs through the following contributions:

1) *SecSLA model definition.* We provide with a novel secSLA representation that (a) provides a formal specification of security requirements and constraints in cloud environment and (b) enables automated management of its life cycle. The model is based on current standards and working groups and relies on the definition of *security controls* and *Security Level Objectives* (SLOs). The novelty of our approach lies in the new layer of the existing SLA hierarchy, namely the measurement layer that is required to evaluate the fulfillment of the committed SLOs.
2) *SecSLA monitoring.* Through the use of secSLAs, we propose a methodology to detect events that either represents a potential or an actual violation of any of the security commitments included in the secSLA. To this end, we use the concept of monitoring rules generated from the secSLAs content.
3) *SecSLA remediation.* In addition, we propose an approach to automatically react to events that might or do entail a violation of security commitments specified in the secSLA. The designed methodology analyses detected events and selects remediation actions to be executed in order to avoid or recover from an invalidation of any clause of the secSLA.
4) *SecSLA management case study.* We validate the proposed methodology by means of a case study based on a real secSLA.

The organization of the rest of the paper is as follows. Section 2 describes the current related work in the area of secSLAs. The secSLA terminology and secSLA model is outlined in Section 3. Section 4 presents a high level approach to management of secSLAs in the cloud. In particular, we describe the approach and the software for automated management of secSLAs developed in SPECS. Further details about secSLA monitoring are presented in Section 5, the automated secSLA remediation is elaborated in Section 6, and the overall approach validated with a real use case in Section 7. Finally, our conclusions are presented in Section 8.

2. Related Work

Several approaches are emerging to represent security in cloud computing. The security policies specification (as defined in [6] and [14]) is a transversal aspect for all the participants in the service supply chain. Some industry efforts (Monahan and Yearworth [15] and Undheim [16]) have contributed to the security SLAs approaches. However, the format that providers are using to represent security policies or SLAs varies from one provider to another, namely because there is no common vocabulary to represent the different security aspects that are provided by them. The CSA provides a common format for the security provisions to be included in security policies and secSLAs, such as the STAR repository [17]. A machine-friendly approach that expresses the security provisions is considered by Casola [18], Luna [19], and Taha [20]. Savola [21] and CSA with the CCM [7] go one step beyond by organizing these security provisions into *categories* derived from a taxonomy. In [23] authors improve those models by including also dependencies between provisions that may also affect to the subsequent reasoning.

Contemporary research advocates assessing the security provided by CSPs by comparing secSLAs with respect to CSC's security requirements (e.g., the QHP [20], QPT [23], and REM [24] techniques). However, there is still a conspicuous gap in what regards the real usefulness of the information included with a secSLA. *Furthermore,* the approaches to monitor security in CSPs are noticeably scarce. Most existing monitoring techniques focus on the monitoring of performance indicators, as shown in [25], [26], and [27]. In the cloud context, Amazon's CloudWatch [29] is one of the main approaches but the solution is limited to their proprietary domain. DeSVi [28] includes the detection of SLA violations, while Brower [30] presents continuous monitoring for detection of intrusions and malicious attacks for web service providers or cloud environments. Finally, NIST's SCAP specifications [31] and Cloud Security Alliance's Cloud Trust Protocol [32] provide interfaces for extracting monitoring data from clouds.

With respect to approaches to automatically react to SLA violations, the gap is even broader, especially when looking for the remediation of security aspects. Most of the literature refers to QoS indicators. That is the case of [33] that detects SLA violations of performance indicators based on resource measurements. However, this detection is not used to design remedies but to study the propagation of the effects of the violation into the system. Other approaches, such as [34], are focused on SLA violations but only in terms of determining penalties associated to them, not going into the remediation before applying the penalties. In [35] SLAs are used to adapt performance parameters of cloud services according to a model to monitor and evaluate them. However, this approach sets users aside and is focused just on the optimization of the provider resources. Finally, the SLA@SOI project [36] developed an infrastructure to monitor QoS related SLAs (mainly performance indicators). The model is based on modelling potential errors (and its possible dependencies) that might affect an enforced SLA, providing adjustment actions to solve them based on prediction models.

In the reminder of this paper we illustrate a new methodology that aims to solve the above discussed issues.

3. Cloud Security SLAs

This section presents the core secSLA terminology used in this paper. We also describe the SLA model used to formalize the CSC's requirements and CSPs commitments for the acquired cloud service.

SecSLAs constitute the "contracts" that specify the commitments (specification of services delivery and the liabilities for shortcomings) across the CSPs and CSCs for providing agreed upon levels of security for the chosen set of security attributes. The main part of these documents are the Service Level Objectives (SLOs) undertaken by CSPs in order to be fulfilled during the service operation. SecSLAs are also used to model CSP's security at the service level by defining a collection of security statements that describe the services that the CSP agrees to provide. As a paper contribution,

this led us to the definition and usage of metrics and their underlying measurements that are essential for enforcing, monitoring, and remediating secSLAs.

Typically, SLOs are the targets for service levels that the provider agrees to meet. If an SLO is not fulfilled, the customer may request a remedy (e.g., a new configuration, a new service or a financial compensation). In order to assess if the agreed SLA is being fulfilled, we need a way to quantitatively evaluate SLOs, which is critical in the case of secSLAs. Together with the problem of how to model secSLAs comes a challenge on how to design mechanisms to automatically enforce them. These are the main issues that this paper addresses.

Figure 1 illustrates the secSLA model developed and used in this paper. We use the concept of SLOs, where one SLO is an extension of exactly one quantitative metric. While each metric defines a various set of possible values associated to a certain security aspect, an SLO represents the commitments of the CSP with regards to a particular guaranteed value for one specific metric. As we detail in Section 5, SLOs extends the metric element with additional attributes necessary to verify the fulfillment of the secSLA, such as the *operator* used to check the assigned value (i.e., *"higher than"*, *"equals to"*) or the *level of importance* with respect to other SLOs. Metrics are categorized according to the CSA's CCM, in a hierarchical structure, into control groups called *security controls*. Similarly, security controls belong to a set of *control categories* that represent the highest level of abstraction in the hierarchy. Of course, security metrics could be categorized according to any security control framework, for example, according to NIST's SP 800-53 [8]. The choice of the control framework and mapping of metrics to controls is the responsibility of the CSP.

Our model combines the approaches advocated by practitioners of security controls frameworks, such as the CSA's CCM [7], the ISO/IEC 19086 [11], the NIST's RATAX working group [12], and the approach presented by Luna [47]. Whereas current standards and working groups consider a three levels model (control category, security control, SLO/metric), which are essential in SLA negotiation phase, our model adds the level of measurements. This level is the crucial one that enables monitoring and facilitates an automated process of remediating SLA violations.

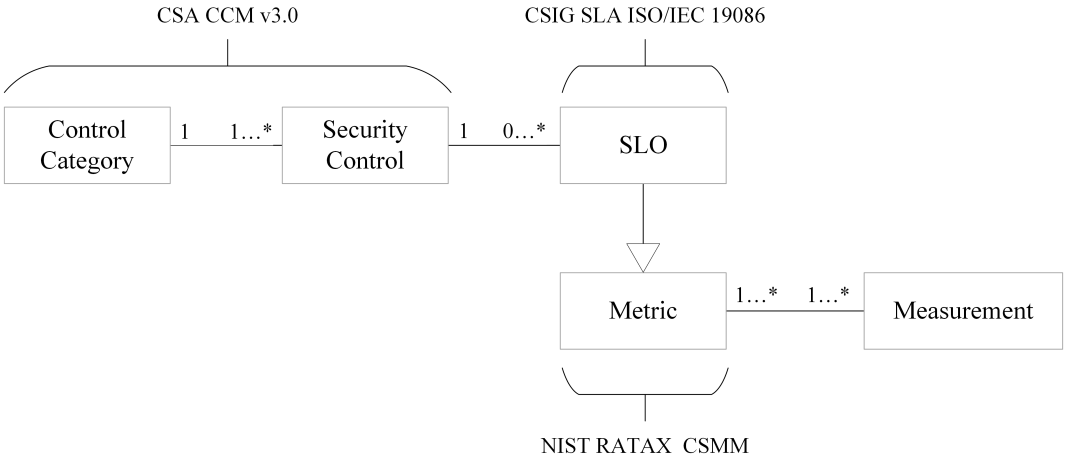

Figure 1. The cloud secSLA specification

As an example, let us suppose that a CSP implements the secSLA Control *Vulnerability/patch management* (i.e., TVM-02) from the CSA CCM v3 [22]. As observed in Figure 2, this control is actually contained within the group *Threat and vulnerability management* (i.e., TVM). After selecting TVM-02, the same CSP then refers to the SLO list provided in the C-SIG SLA report [13] (or any other relevant standard) and finds out that this control is associated to the SLO *Vulnerability scanning frequency time period*. This SLO is then mapped by the CSP to a specific security metric, namely *Vulnerability scanning frequency*. The specification of the security metric defines the possible values for its corresponding SLO. Figure 2 also shows the measurement layer that allows monitoring the current state of the corresponding metric (i.e., validity of the associated SLO).

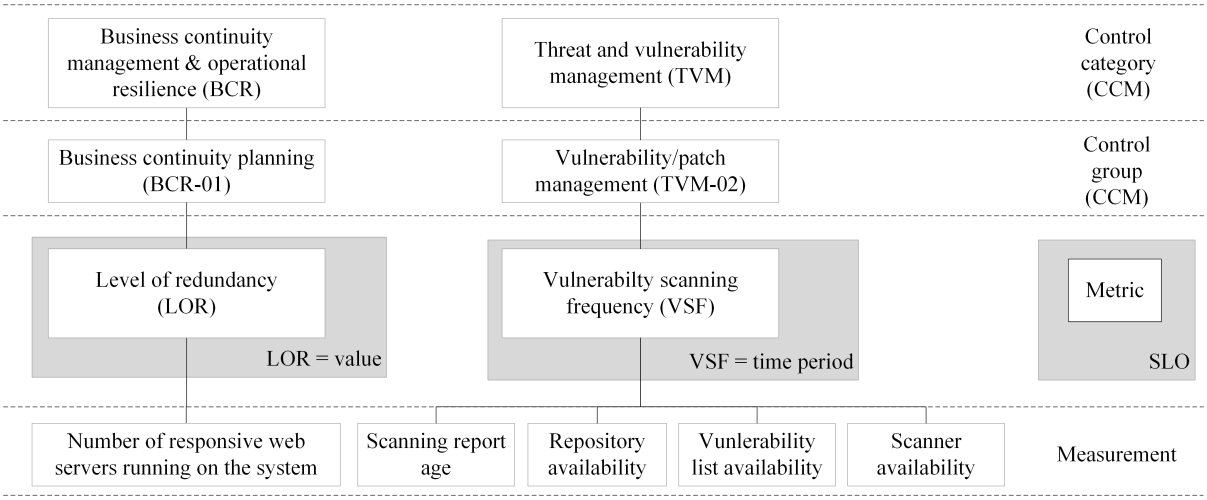

Figure 2. Example of the cloud secSLA hierarchy

Table 1 shows an example of the measurements required to assess the state of the metric *Vulnerability scanning frequency*. This metric represents the time elapsed between two sequential vulnerability scans. The current state of the metric is obtained with four measurements while the value of each measurement is obtained directly by monitoring the CSP's service. Further details about the relationship among metrics and measurements are provided in Section 5.

Metric	Description	Possible values	Measurements	Value type
Scanning frequency	Sets the vulnerability scanning frequency (Example: Scanning frequency is 24h)	Integer > 0 (hours)	Scanning report age	*int (hours)*
			Repository availability	*boolean*
			Vulnerability list availability	*boolean*
			Scanner availability	*boolean*

Table 1. Example of a security metric and its measurements

Control categories and control levels are normally used to represent customers' requirements and are used to support the security assessment of CSPs (as in [20]), However, for the scope of this paper we focus on (a) the SLOs, (b) the metrics, and (c) the measurement levels, that together constitute the key elements to enforce, monitor, and remediate SLOs.

If any of the committed values for the SLOs is not fulfilled, then the secSLA is violated and a remediation process has to be initiated. In real-world cloud scenarios, the metrics are subject to possible dependencies. Thus, in order to plan and carry out remediation actions required for potential or actual violations, it is necessary to define these dependencies since one event may affect more than one SLO. Dependencies among metrics commonly appear in relevant standards and best practices, where metrics are directly depending in order to allow their composition to generate more complex ones. The Cloud Service Metrics model from NIST [12] supports these direct dependencies through the notion of *concrete* and *abstract metrics*. In this paper, we define dependencies among metrics/SLOs through their associated measurements.

4. Cloud SLA Management

As described in Section 3, a secSLA is an agreement between a CSC and a CSP about the level of security provided by the CSP for the cloud services acquired by the CSC. Figure 3 represents the phases of the secSLA life cycle.

During the *preparation phase*, the CSP defines security metrics to be included in the secSLA, their possible values, and their relevance according to the type of service offered. For example, in the cloud storage domain the most important metrics would be associated to data confidentiality and integrity, whereas a CSP offering web servers would give higher priority to metrics associated to availability and performance. Capabilities are also defined by CSPs according to their infrastructure. For example,

a CSP offering web servers can only define a metric defining the number of different web servers to be deployed on virtual machines if it supports at least two different web servers.

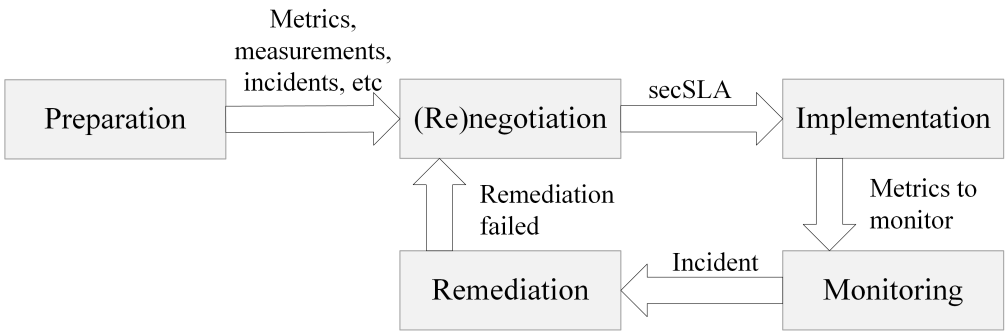

Figure 3. The secSLA life cycle

The agreement is formalized in the second phase of the secSLA life cycle, termed as the *negotiation phase*. During this process, the CSC chooses a set of offered security features (i.e., chooses security metrics) and determines their configurations (i.e., decides on metric values). For example, if a CSP providing web servers offers configurable software vulnerability assessment of virtual machines hosting the web servers, the CSC might be able to choose a metric related to the frequency and depth of vulnerability scans and assign preferred values to the chosen metrics (e.g., vulnerability scans should be conducted every 24 hours and vulnerability scans should include the entire operating system and all running applications). The set of chosen metrics and their values imply a set of SLOs to which the CSC assigns levels of importance. The resulting secSLA is then verified by the CSP and signed by both parties.

As we discuss in Section 5, the levels of importance along with the values of the chosen metric determine the frequency of actions performed by the CSP to enforce the secSLA, and the frequency of measurements taken on the system to evaluate the state of the signed secSLA. Metrics in SLOs with higher level of importance are evaluated more often as those with lower level of importance. In Section 6 we elaborate how importance weights also serve as a guideline for concurrent occurrence of violations of more than one SLO for the same secSLA. In such cases, the importance levels determine which part of the secSLA is remediated first.

When all services are negotiated, they are implemented by the CSP during the *implementation phase*. In this phase the CSP configures delivered services to automatically enforce and monitor negotiated security features. After configuring the services, the CSC's secSLA enters the *monitoring phase*. If any deviations from what is specified in the secSLA are detected by the CSP's monitoring tools, the secSLA undergoes *remediation phase*.

In order to automate the remediation process, the CSPs have to (a) define measurements for all the metrics they offer, (b) specify all detectable incidents and failures related to the defined measurements, and (c) project remediation actions for specified alerts and violations. Despite precise planning, not all incidents and failures can be automatically mitigated. In such a case, the CSC is notified about the occurrence, compensated (if applicable), and offered to renegotiate invalidated secSLA. Of course, in cases where automated remediation process fails to mitigate the risk of having a secSLA violation or fails to recover from one, not only the CSC but also CSP is notified about the event. In this way the CSP has the opportunity to manage the remediation process manually before offering the CSC a renegotiation (a new cycle of the *negotiation phase*) of the invalidated secSLA.

In Table 2 below we summarize the roles and responsibilities of CSPs and CSCs according to the described preparation step and the four secSLA phases.

	Actor	
Phase	**CSP**	**CSC**
Preparation	• Define security metrics to be offered and their possible values, levels of importance, measurements, and frequencies. • Define actions with which the defined metrics are automatically enforced and monitored. • Define security incidents/failures and remediation plans for the defined measurements.	/
secSLA negotiation	• Present supported security features in terms of negotiable security metrics.	• Select preferred security metrics and specify their values and levels of importance.
	Sign the secSLA.	
secSLA implementation	• Configure cloud resources and deploy security mechanisms able to enforce/monitor the signed secSLA.	/
secSLA monitoring	• Collect and filter monitoring data.	/
secSLA remediation	• Analyze and automatically remediate detected secSLA violation.	• Renegotiate the secSLA if remediation is unsuccessful.

Table 2. Roles of the CSC and the CSP

It is true that the main public CSPs (such as AWS [50] and Azure [51]) today offer no room for SLA negotiation. They provide their secSLAs on their web sites where the customer either accepts the terms and signs the secSLA or not accept the terms and decline their services. Nevertheless, the approach to the secSLA management presented in this paper will become an important solution when in the future (due to the rising number of cloud providers and more and more demanding and security aware cloud users) the secSLA negotiation phase becomes a reality in the public cloud domain. Until that point, the presented approach is applicable to a more private scenario, for example, a private cloud of a computing center, or to let developers specify their requirements to the infrastructure for the (off-premise) operators.

As outlined in the introduction, the presented approach to the automated management of secSLAs has been developed over the European FP7-ICT research project SPECS [37], [38]. In SPECS, we have implemented a set of components that oversee negotiation, implementation, monitoring, and remediation phases. As seen in Figure 4, the CSC accesses the SPECS web application through which she/he acquires a cloud service and negotiates its security level. The negotiation phase is orchestrated by the Negotiation module. The signed secSLA is sent to the Planning component of the Enforcement module, which prepares a so-called implementation plan which specifies resources to be set up, security mechanisms to be deployed in order to enforce and monitor the negotiated security features, and their configuration. The implementation plan is later executed by the Implementation component. During the SLA monitoring phase, the Monitoring module collects and filters the monitoring data. Any detected deviations from what is expected (according to signed secSLAs) are notified to the Diagnosis component which analyses notified events. According to the analysis done, the Remediation Decision System component identifies the optimal remediation plan, which is executed by the Implementation component. The code for the components that orchestrate the described SLA life cycle is available at the project's Bitbucket account [48].

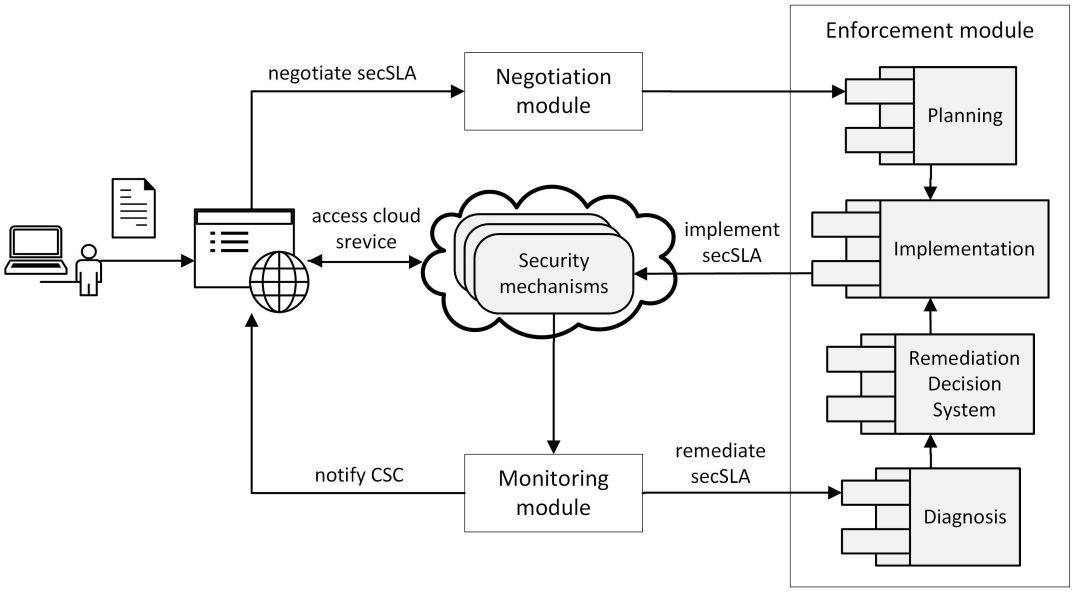

Figure 4. SPECS architecture

The following Sections 5 and 6 respectively detail the automated monitoring and remediation phases.

5. Monitoring SLAs

With every signed secSLA, the CSP does not only commit to providing a certain set of security services but also to sustain the guaranteed level of security of chosen services. For example, if a CSP agrees to offer vulnerability scans of provided resources, they should be executed and they should be executed as often as specified in the secSLA. For the purpose of continuously assessing the state of each signed secSLA and guaranteeing its fulfillment, as already introduced in the description of the secSLA model, each metric has a defined set of measurements.

Every metric is associated with at least one measurement, but can also be mapped to more than one measurement. Similarly, every measurement corresponds to at least one metric but can also be mapped to more than one metric. For example, the *vulnerability list update frequency* (VLUF) metric, which assures that the data related to published vulnerabilities is retrieved from a public repository and the corresponding report with the list of disclosed vulnerabilities is generated periodically, is evaluated with measurements *vulnerability list age* (VLA) and *repository availability* (RAV). At the same time, the measurement RAV is used to evaluate also, for example, the metric related to the frequency of vulnerability scans (VSF), which is enforced by periodically retrieving vulnerability data from the public repository, generating vulnerability list, and executing vulnerability scans. This metric is evaluated with more than one measurement, for example, also with the age of the produced scanning report (SRA) and availabilities of the vulnerability list (VLAV) and vulnerability scanner (SAV), as depicted in Figure 5.

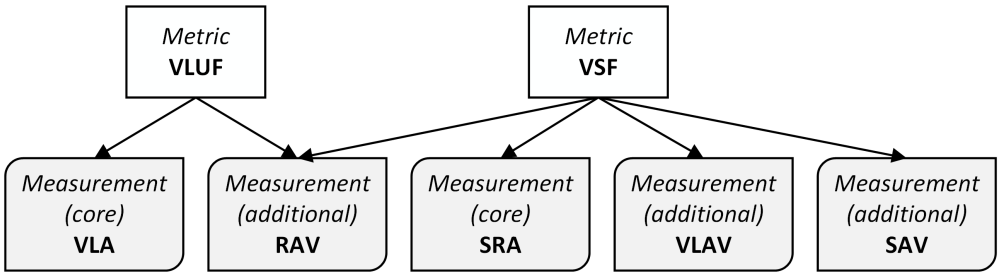

Figure 5. Relationship among metrics and measurements

For each security metric exactly one of the associated measurements should be the *core* one, and the rest of the measurements are *additional*. With the core measurement we are able to detect violations,

but the additional ones enable detection of possible violations (i.e., alerts) even before they occur. For example, metric VLUF has one core (VLA) and one additional measurement (RAV), and metric VSF has one core (SRA) and three additional measurements (RAV, VLAV, and SAV).

The set of agreed upon SLOs (i.e., metrics, their values, associated operators, and assigned levels of importance) is transformed into a set of *enforcement and monitoring actions* (i.e., actions executed by the CSP in order to enforce and evaluate the state of all associated metrics), a set of *monitoring rules* (i.e., measurements and associated operators and thresholds), and accompanied enforcement and monitoring *frequencies*; see Figure 6.

For the example above, an SLO stating *VLUF = 24h* with a high importance weight would transform into a couple of enforcement actions related to retrieving vulnerability data, generating vulnerability list, and checking its age every 24 hours. The set of monitoring actions would be related to evaluating, for example, every 4 hours, if the monitoring rules are respected. The associated monitoring rules would be defined as $VLA \leq 24h$ and *RAV = yes*.

Similarly, an SLO stating *VSF = 48h* with a low importance weight would transform into the following actions and rules. Enforcement actions, executed every 48 hours, would entail retrieving vulnerability data, generating vulnerability list, executing vulnerability scans, generating scanning report, and checking its age. Monitoring rules would be specified as $SRA \leq 48h$, *RAV = yes*, *VLAV = yes*, and *SAV = yes*. And the set of monitoring actions would be related to verifying, for example, every 24 hours, validity of monitoring rules.

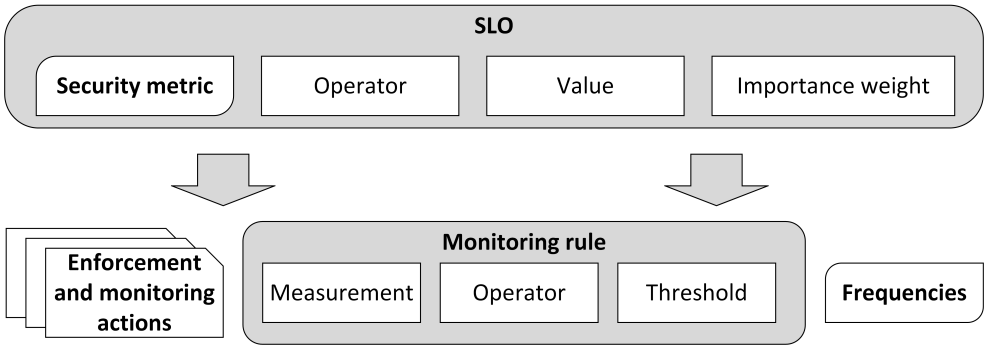

Figure 6. Transformation of SLOs into a set of enforcement actions and monitoring rules

Due to dependencies among security metrics that are indicated through their mapping to measurements, it may happen that two or more SLOs in the same secSLA imply some identical enforcement and monitoring actions. In such a case, those actions are only executed once but the frequency is determined by taking the most restrictive values out of all implied by SLOs. Similarly, whenever two or more SLOs with different importance weights imply the same measurement, its operator and threshold are determined by taking the most restrictive values out of all implied by SLOs.

Monitoring frequencies depend not only on importance weights, but also on the nature of the metric (e.g., assuring frequency of some action, assuring availability of some service) and the type of the measurement (core vs. additional). Core measurements and the ones monitoring availability of services should, in general, be taken more often than the rest.

All these details about how to plan enforcement and monitoring action, how often to execute them, and how to optimize them in order to allow parallel management of secSLAs, depend on the entire assortment of services offered by the CSP and its underlying infrastructure. Therefore, they should be defined by the CSP.

During the SLA monitoring phase (Figure 7), all measurements are continuously taken by security mechanisms deployed on cloud resources and evaluated against the defined thresholds (implied by all signed secSLAs and maintained by the Monitoring module). Any deviation from the expected

threshold should be further analyzed. Thus all measurement results indicating suspicious behaviour immediately trigger remediation phase (the detected suspicious events are notified to the Diagnosis component of the Enforcement module which analyses them) and all measurement results that comply with assigned thresholds are stored for possible future risk assessment and auditing (in the archiver component of the Monitoring module).

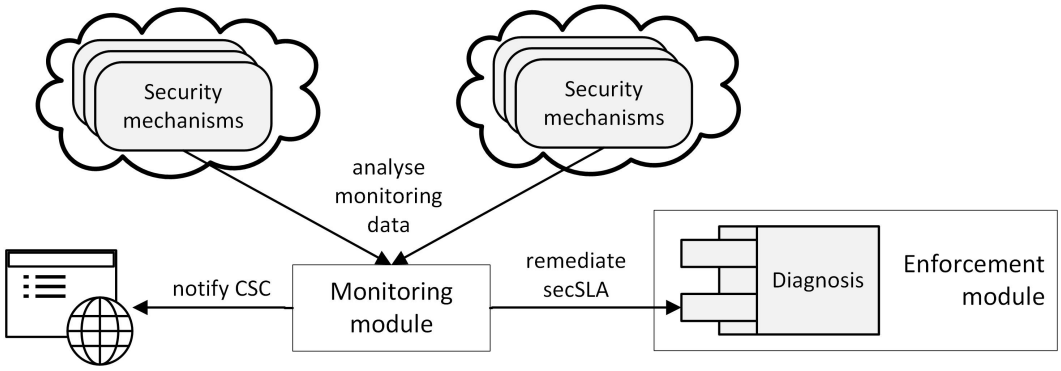

Figure 7. SecSLA monitoring

6. Automated Remediation of SLA Violations

The remediation phase starts when the CSP's monitoring processes detect anomalous behaviour, i.e., whenever the monitoring detects that some measurement value deviated from the defined threshold.

The first step of the remediation process is an analysis of the event with respect to the associated secSLA. As illustrated in Figure 8, the set of metrics and SLOs that the measurement is linked to is identified. Based on the type of the measurement defined for each metric, the type of the event is determined for each affected SLO. If the measurement is the core measurement for a metric, this implies an SLO violation, and if the measurement is an additional measurement for a metric, the event implies an SLO alert. If the event represents an alert to all affected SLOs in the secSLA, then the entire secSLA is labelled as alerted, but if at least one of the affected SLOs is violated by the measurement's deviation, the entire secSLA is violated. According to the event type and considering importance weights of the alerted/violated SLOs, the risk or severity level can be determined for an alerted or a violated secSLA, respectively.

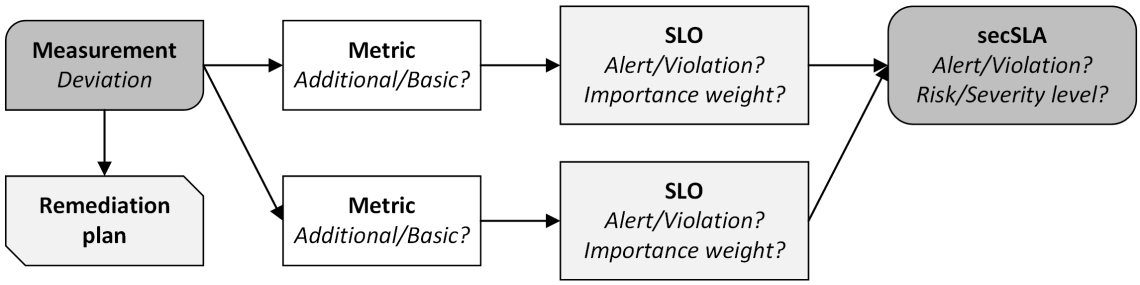

Figure 8. Diagnosis process

It is possible that the monitoring tools detect a deviation of several measurements for one secSLA at "the same time". In this case, risk/severity levels determine the sequence in which all detected events should be handled in order to avoid performing different reconfigurations on the system for one secSLA simultaneously and possibly causing more damage.

After the diagnosis process, a *remediation plan* is identified for the measurement associated to the notified incident/failure. A remediation plan is built specifically for each measurement and comprises a set of *remediation actions* and a clear *sequence* in which they should be executed. A remediation action is composed of some monitoring activity (i.e., take some measurement) or an enforcement

activity followed by a monitoring action (i.e., change some configuration and check if the reconfiguration was successful).

Remediation plans are built by CSPs possessing detailed information about their infrastructure, the services they offer, and the metrics they offer for negotiation. Since CSPs may over time change the set of offered services, the plans many need revisions to match the changes. Also given the continually changing attacker behaviors, the remediation plans require frequent revisions to match newly discovered vulnerabilities and due to increased risk of exposure to attacks which are becoming more complex and more targeted.

As depicted in Figure 9, a sequence of remediation actions is executed until either the event is resolved or until there are no more remediation actions available. Regardless of the event type and the result of the remediation process, the CSC is always notified about the occurrence and about the outcome. In case of an alert, the secSLA reenters the monitoring phase, and in case of a violation, the CSC is entitled to a compensation as specified in the secSLA. If the violation is resolved, the CSP compensates the CSC and the secSLA reenters the monitoring phase, but if planned remediation activities failed to recover from the violation, the CSC is compensated and also offered a renegotiation of the invalidated secSLA.

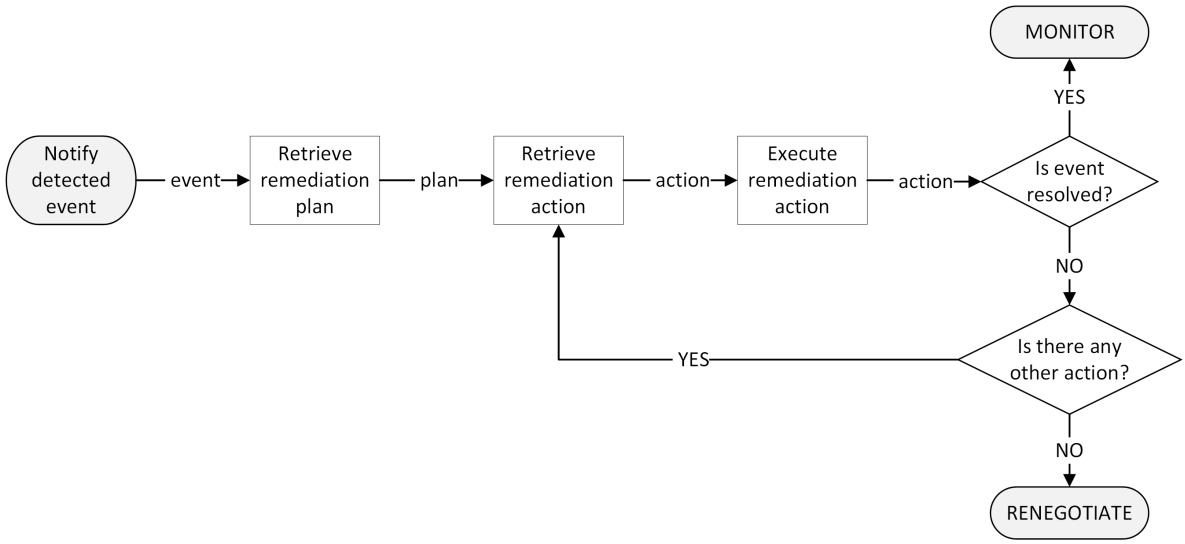

Figure 9. Remediation flow

Note that the assumption in the presented remediation methodology is that CSP's monitoring processes only observe parameters associated to the measurements defined for all offered security metrics, i.e., parameters associated to SLOs in undertaken secSLAs. Thus each detected incident/failure is always associated to at least one measurement/metric and associated remediation procedure to handle the event.

As mentioned above, all alerts and violations related to the same secSLA should be handled one by one in order to avoid different reconfigurations on the system simultaneously. But in order to allow for a better service and ensure higher level of security, various events related to different secSLAs can be handled in parallel, as long as reconfigurations do not affect the same system (e.g., the same VM). Determining compatibilities among remediation actions in order to identify those that can be executed in parallel is up to the CSPs since it depends on the CSP's infrastructure, services they provide, and security metrics they offer.

In order to evaluate penalties associated to a violated secSLA, the CSP has to consider all violated SLOs. This means considering all SLOs affected by the deviation of the measurement initially reported in the notification of the event, and also all SLOs affected by the deviation of measurements taken during the remediation process. For example, consider a secSLA with which a CSP guarantees a set of VMs with web servers and assures software vulnerability assessment of provided VMs by

means of periodic vulnerability scans. At some point the vulnerability scan fails because one of the scanners installed on one of the VMs is unresponsive. The process of remediation reveals that the cause for it is a crash of the entire VM hosting it. The CSP should consider a violation to the metric/SLO related to the frequency of vulnerability scans as well as the metric/SLO associated to assuring web server replication (assuring availability of web servers and VM replicas). Note that the actual calculation of penalties is related to the business side of the secSLA and is out of scope for this paper.

In some cases, a remediation plan or even a specific remediation action may cost the CSP more than paying penalties for the associated secSLA violation. It is up to the CSP to plan appropriate remediation processes for cost tradeoffs across remediation/mitigation/recovery actions for business continuity, reputation, and profitability.

Remediation plans depend on the CSP and on the security mechanisms that it implements. Namely, every CSP decides (i) on cloud services to provision, (ii) on security features to offer for the defined cloud services, and (iii) on security mechanisms able to enforce and monitor them. Hence the remediation plans depend on security metrics considered by the CSP and on functionalities supported by security mechanisms implemented by CSPs.
For example, let us consider two CSPs that offer web servers and assure resilience to security incidents through the following features:

- *Level of Redundancy* (LOR), i.e. the number of server replicas;
- *Level of Diversity* (LOD), i.e. the number of different types of web servers;
- *Maximum acceptable CVSS score of vulnerabilities* (MCS), i.e. the maximum acceptable CVSS score [40] for the unfixed software vulnerabilities detected on the system.

Furthermore, let us assume that one of the CSPs (CSP1) only offers 2 different types of web servers and that the other CSP (CSP2) supports 3 different types of web servers. Let us now take a CSC that signs a secSLA with objectives *LOR=2*, *LOD=2*, and *MCS=5*, which means that the CSC requires two different web servers for which only vulnerabilities with CVSS<5 are acceptable. If at some point a serious vulnerability is discovered and published (for example, with CVSS=9) for one of the offered web servers (which would represent a secSLA violation for the SLO associated to the MCS metric), CSP2 could replace the vulnerable web server and simply recover from the secSLA violation, whereas remediation plans for the CSP1 would involve applying patches to the vulnerable web server.

A practical example that demonstrates the introduced approach to the automated secSLA remediation is provided in the next section.

7. Validation: A Practical Example

As discussed in Section 4, in SPECS we have developed a framework that orchestrates the presented approach to secSLA monitoring and remediation. Moreover, we have developed a set of security mechanisms that are able to enforce and monitor different sets of security properties in the cloud. For example, we have developed the End-to-End Encryption (E2EE) mechanism [49] that offers to CSCs client-side encryption and monitors write-serializability (i.e., consistency among updates) and read-freshness (i.e., assurance that the requested data is always fresh as of the last update), and the Software Vulnerability Assessment (SVA) mechanism that offers to CSCs periodic vulnerability scans, penetration testing, and automated patch management. The code for the mechanisms that are deployed to cloud resources through the SPECS framework in an automated way according to SLOs specified in signed secSLAs is available on the project's Bitbucket account [48].

Every CSP can offer to its cloud customers different security features (associated to data confidentiality, data integrity, denial of service protection, etc.) that are enforced and monitored with different security mechanisms. As a proof of concept we consider in this section one set of features implemented by one security mechanism, namely the SVA mechanism. In particular, we consider a CSC signing a secSLA with a CSP offering the SVA mechanism, and we discuss automated

management of a set of security incidents/system failures according to the procedures presented in this paper (according to Table 2).

Preparation phase – definition of metrics and measurements for the SVA mechanism

The SVA mechanism is offered to CSCs through a set of various security metrics related to frequency of vulnerability scans, depth and breadth of scanning coverage, penetration testing, detection of misconfigurations and presence of unauthorized components or libraries, etc. As a proof of concept, we consider the security metrics reported in Table 3.

ID	Name	Description	Possible values
VSF	Vulnerability scanning frequency	Set the vulnerability scanning frequency.	integer > 0 (hours); default 24
MCS	Maximum acceptable CVSS score of vulnerabilities	Set the maximum acceptable CVSS score for the unfixed software vulnerabilities detected on the system.	0 < integer < 10 (CVSS score); default 4

Table 3. SVA security metrics

In order to perform vulnerability scans and enforce metric *Vulnerability scanning frequency* (VSF), a list of published software vulnerabilities (including misconfigurations) has to be provided to the scanner. And to assure meaningful scans, the list should be repeatedly updated. Since many repositories exist that provide data related to security vulnerabilities (e.g., [39], [40]), the SVA mechanism can not only retrieve this data at any point, but it can also retrieve it from many different sources, if needed (e.g., when one repository is unavailable, the mechanism can switch to another source).

When the vulnerability scan is conducted, the SVA mechanism generates a scanning report outlining all detected security vulnerabilities, their characteristics, and their impact in terms of CVSS scores [40]. On the basis of the produced scanning report, the CSP can determine the maximum CVSS score of all discovered vulnerabilities. The CSC can configure desirable security level of the resources acquired with the CSP by setting the upper thresholds for the severity of unfixed software vulnerabilities detected on the system through metric *Maximum acceptable CVSS score of vulnerabilities* (MCS). With this metric the CSP guarantees automated remediation of all detected vulnerabilities with higher security risk (e.g., assures automated patching, assures automated updates and upgrades of detected vulnerable libraries).

The following table lists all measurements defined for the considered security metrics that are needed to support mechanism's functionalities and assurances.

Measurement			Security metric	
ID	**Name**	**Type**	VSF	MCS
SRA	Scanning report age	core	●	
MDCS	Maximum CVSS score of detected vulnerabilities			●
RAV	Repository availability	additional	●	●
VLAV	Vulnerability list availability		●	●
SAV	Scanner availability		●	●

Table 4. SVA measurements

Preparation phase – definition of enforcement and monitoring actions, their frequencies, and levels of importance

As discussed in Section 5, each metric has an associated list of enforcement actions. For example, in order to enforce metric VSF and check validity of the SLO associated to it, the CSP should (a) retrieve data related to published vulnerabilities, (b) generate vulnerability list, (c) perform vulnerability scan, (d) generate scanning report, and (e) check if the age of the report is under the

threshold set by the CSC (to ensure that the report has been generated). Similarly, in order to enforce and verify validity of the SLO associated to metric MCS, the CSP (a) retrieves data related to disclosed vulnerabilities, (b) generates vulnerability list, (c) performs vulnerability scan, (d) generate scanning report, (e) determines maximum CVSS of all detected vulnerabilities, and (f) checks if the risk level of the vulnerability with the highest CVSS score is below the threshold set by the CSC. Table 5 below presents all main actions required for automated enforcement, and also all supplemental actions required for automated monitoring and remediation of the considered security metrics.

For the considered example, where we offer three importance levels, high (H), medium (M), and low (L), we assume enforcement and monitoring frequencies as shown in Table 6.

Enforcement/monitoring action			**Security metric**	
			VSF	MCS
Main Actions				
ID	**Type**	**Description**	**Enforcement step**	
EA1	enforcement	Retrieve data related to published vulnerabilities.	1	1
EA2	enforcement	Generate vulnerability list.	2	2
EA3	enforcement	Perform vulnerability scan.	3	3
EA4	enforcement	Generate scanning report.	4	4
EA5	monitoring	Check if *SRA* ≤ *VSF value*.	5	/
EA6	enforcement	Determine maximum CVSS score of detected vulnerabilities.	/	5
EA7	monitoring	Check if *MDCS* ≤ *MCS value*.	/	6
Supplemental actions				
EA8	monitoring	Check if *RAV* = *yes*.		
EA9	monitoring	Check if *VLAV* = *yes*.		
EA10	monitoring	Check if *SAV* = *yes*.		
EA11	enforcement	Reconfigure repository.		
EA12	enforcement	Restart scanner.		
EA13	enforcement	Check for available patches for detected vulnerabilities.		
EA14	enforcement	If possible, apply patches[2].		
EA15	enforcement	Determine maximum CVSS score of unfixed detected vulnerabilities.		

Table 5. SVA enforcement actions

	Security metric					
	VSF			MCS		
	Importance level					
Action type	H	M	L	H	M	L
enforcement	*VSF value*	*VSF value*	*VSF value*	12h	24h	48h
monitoring	*VSF value*/6	*VSF value*/4	*VSF value*/2	2h	6h	24h

Table 6. SVA frequencies

Preparation phase – definition of security incidents/system failures and associated remediation actions

Based on the defined metrics and measurements, the CSP is able to detect events presented in Table 7.

[2] This includes checking for possible updates and upgrades of vulnerable libraries, packages, etc. If an update or upgrade is performed, this includes removing old version of vulnerable software.

Event			Security metric	
ID	**Description**	**Type**	VSF	MCS
E-SRA	*SRA > VSF value*	violation	●	
E-MCS	*MDCS > MCS value*			●
E-RAV	*RAV = no*	alert	●	●
E-VLAV	*VLAV = no*		●	●
E-SAV	*SAV = no*		●	●

Table 7. Detectable SVA events

For each detectable event, the CSP defines a remediation plan in order to eliminate its occurrence. As described in Section 6, a remediation plan comprises a set of remediation actions, and each remediation action consists of one or more enforcement/monitoring actions. Table 8 specifies all SVA remediation actions and the sequence of enforcement/monitoring actions each of them imply.

Remediation action		Enforcement/monitoring actions			
ID	**Description**	**Step 1**	**Step 2**	**Step 3**	**Step 4**
RA1	Check if configured repository for vulnerability data is available.	EA8			
RA2	Download vulnerability data from the configured repository, generate vulnerability list, and check if it is available.	EA1	EA2	EA9	
RA3	Reconfigure vulnerability data repository and check if it is available.	EA11	EA8		
RA4	Check if the vulnerability list is available.	EA9			
RA5	Check if the installed scanner is responsive.	EA10			
RA6	Perform vulnerability scan, generate scanning report, and check if its age is below the threshold.	EA3	EA4	EA5	
RA7	Restart installed scanner and check if it is responsive.	EA12	EA10		
RA8	Check for available patches for detected vulnerabilities and apply them if possible, determine the maximum CVSS score of unpatched detected vulnerabilities, and check if it is below the threshold.	EA13	EA14	EA15	EA7

Table 8. SVA remediation actions

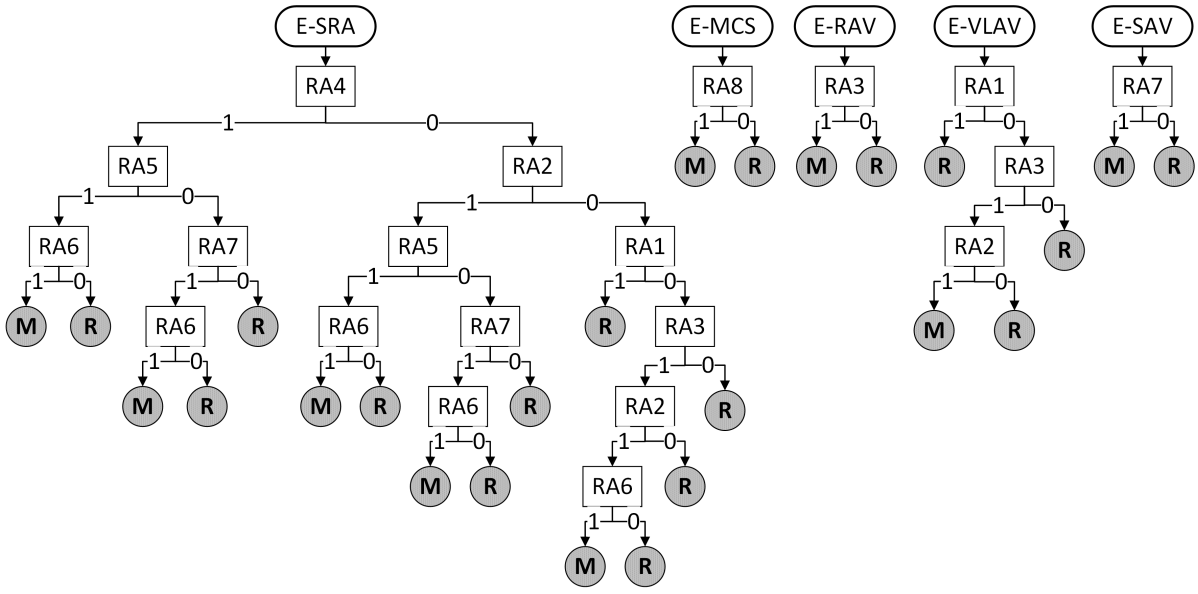

Figure 10. SVA remediation plans

Finally, in Figure 10 above we present remediation plans associated to all SVA events listed in Table 7. Since all remediation actions conclude with a monitoring activity that either results in a *true* or a *false* (in diagrams a *true* corresponds to a 1 and a *false* corresponds to a 0), each remediation plan can be constructed as a decision tree.

Each remediation of an event can either end successfully (i.e., the end node, denoted with **M**, represents the monitoring phase that the secSLA can reenter) or with a demand for further CSP's and/or CSC's assistance (i.e., the end node, denoted with **R**, represents the renegotiation phase that the secSLA enters in worst case scenario[3]).

Negotiation phase

Consider a CSC who wants to acquire a pool of web servers and secure them with the SVA mechanism. Let us assume that the CSC signs a secSLA with two SLOs, namely SLO1 with medium importance defined as $VSF = 24h$ and SLO2 with low importance defined as $MCS = 4$. According to Table 4, the CSP has to monitor all defined SVA measurements, and the following monitoring rules have to be periodically evaluated: $SRA \leq 24h$, $MDCS \leq 4$, $RAV = yes$, $VLAV = yes$, and $SAV = yes$.

Implementation and monitoring phase

In order to enforce SLO1, the CSP has to execute enforcement actions EA1-EA4 every 24 hours, and in order to enforce SLO2, the CSP has to execute enforcement actions EA1-EA4 and EA6 every 48 hours (see Table 5 and Table 6). Thus the final setting is to perform enforcement actions EA1-EA4 once every 24 hours and action EA6 once every 48 hours.

In order to check validity of SLO1, the CSP has to execute monitoring actions EA5 and EA8-EA10 every 6 hours, and in order to verify validity of SLO2, the CSP has to execute monitoring actions EA7 and EA8-EA10 every 24 hours. Hence, the CSP executes monitoring actions EA5 and EA8-EA10 once every 6 hours, and monitoring action EA7 once every 24 hours.

All actions and associated frequencies related to enforcing and monitoring signed secSLA are summarized in Table 9. The actual schedule is up to the CSP who manages scheduling of actions for all undertaken secSLAs in an optimized and efficient way (i.e., scheduling the same actions for different secSLAs in the same time slots). But in general, for example, the availability of the vulnerability list should be verified (monitoring action EA9) prior to vulnerability scans (enforcement actions EA1-EA4) and age of the scanning report should be verified (monitoring action EA5) after each scan.

Action	Description	Frequency	Related SLOs	
			SLO1	SLO2
EA1	Retrieve data related to published vulnerabilities.	24h	●	●
EA2	Generate vulnerability list.	24h	●	●
EA3	Perform vulnerability scan.	24h	●	●
EA4	Generate scanning report.	24h	●	●
EA5	Check if $SRA \leq 24h$.	6h	●	
EA6	Determine maximum CVSS score of detected vulnerabilities.	48h		●
EA7	Check if $MDCS \leq 4$.	24h		●
EA8	Check if $RAV = yes$.	6h	●	●
EA9	Check if $VLAV = yes$.	6h	●	●
EA10	Check if $SAV = yes$.	6h	●	●

Table 9. Frequencies of actions for the secSLA with SLO1 and SLO2

[3] Of course, the CSP could try to resolve the incident/failure manually or even provide a new VM, install different and/or newer OS, and migrate entire service requested by the CSC, before offering renegotiation of the invalidated secSLA.

Remediation phase – Example 1

Let us now consider that during one of the periodic vulnerability scans the scanner crashes due to a missing vulnerability list. We assume that the list has been maliciously or accidentally deleted or corrupted by an attacker or the CSP during the vulnerability scan which means that the absence of the list was not detected between the verification of its availability (EA9) and the planned vulnerability scan (EA3).

After the planned vulnerability scan CSP's monitoring tools detect that the scanning report has not been updated ($SRA > 24h$) which corresponds to the event E-SRA (see Table 7). The occurrence is immediately analyzed as depicted in Figure 8. The involved measurement (SRA) is the core measurement for metric VSF which indicates that the occurred event represents a violation of SLO1. Since VSF is the only metric affected by the event, the SLA is labelled as violated. According to CSC's assigned importance weight to SLO1, the CSP can determine the severity level of the detected violation.

According to defined remediation plans (see Figure 10), remediation of event E-SRA starts with verifying availability of the vulnerability list (remediation action RA4). Since availability list is missing, the next step (i.e., the next remediation action RA2) is to download all data related to published vulnerabilities, generate new vulnerability list, and verify again its availability. This time we assume the list has been successfully generated and is now available. Next remediation action (RA5) indicates checking if the vulnerability scanner installed on the system (i.e., acquired VM hosting one of the web servers) is responsive. We consider the scanner to be reactive, thus the CSP can (according to the next remediation action RA6) execute another vulnerability scan, generate scanning report, and verify that its age is below the set threshold (i.e., $SRA < 24h$). The last remediation action is successful and remediation process ends with a resolution of the detected secSLA violation. The secSLA reenters the monitoring phase, the CSP calculates penalties to be paid to the CSC, and notifies CSC.

Remediation phase – Example 2

Next, let us consider the situation where the CSP's monitoring tools at some point detect unavailability of the vulnerability list ($VLAV = no$) which corresponds to the event E-VLAV (see Table 7). The CSP's remediation tools react according to remediation process defined in Figure 8. Measurement VLAV is an additional measurement for both metrics, VSF and MCS, thus the event represents an alert for both SLOs in the secSLA, and the secSLA enters the alerted state. According to CSC's assigned importance weights to both SLOs, the CSP can determine the risk level of the detected incident/failure.

Following remediation plans presented in Figure 10, remediation of event E-VLAV starts with verifying availability of the configured repository (remediation action RA1). We assume that the repository crashed after the monitoring tools last checked its availability. Since the repository is currently unavailable, the next step (i.e., the next remediation action RA3) is to reconfigure repository (i.e., use a different repository) and verify again its availability. Since we presume that the replacing repository is available, the CSP can (according to the next remediation action RA2) download all vulnerability data, generate new vulnerability list, and check its availability. We consider the last action to be successful, thus the remediation process ends with a resolution of the detected alert, the CSC is notified about the event, and the secSLA reenters the monitoring phase.

Remediation phase – Example 3

The last example we consider is the occurrence of the event E-MCS which indicates that at some point vulnerability scanners detected a software vulnerability of a too high CVSS score (for example, scanning reports shows a presence of a vulnerability with a CVSS score 6, thus $MDCS > 4$). The involved measurement MDCS is the core measurement for the metric MCS, so the event represents a

violation of SLO2 and thus a violation of the entire secSLA. The CSP can determine the severity level of the violation according to CSC's assigned importance weights to SLO2.

According to remediation plans (see Figure 10), the only planned remediation action (i.e., action RA8) for the event E-MCS is to check for availability of patches for the detected high risk vulnerability, to apply it if possible, check if the vulnerability is remediated, and check if the maximum CVSS score of all persisting vulnerabilities on the system is below the assigned threshold. In our example we assume that the detected high risk vulnerability is new and that patches for it are not yet available. This results in an unresolved SecSLA violation and the CSC is offered to renegotiate the agreement.

Technical notes

Note that all presented enforcement, monitoring, and remediation actions can be implemented by one of the existing open source configuration, orchestration, and management tools (like Chef [42], Puppet [43], Salt [44] or Ansible [45]) due to its capability of managing large number of devices. For example, it is highly probable that once a new software vulnerability is found and disclosed, a large number of virtual machines in a cluster (if not all) will be affected and will require remediation actions (like patching the vulnerable software as in the Heartbleed bug case [46]). In SPECS project, where the remediation process was developed, the adopted orchestration tool is Chef.

8. Conclusions

Developing techniques and tools for automated and dynamic management of secSLAs is a challenging task. One needs to define security metrics such that the resulting security assurances and the corresponding secSLAs are not only enforceable by CSPs, but also that they can be monitored and remediated in case of detected failures and incidents.

In this paper, we have described a secSLA specification model and a methodology to manage security commitments included in the secSLA. We have proposed a comprehensive model and processes to enable detection of deviations from agreed upon security settings. Most importantly, we have introduced an automated remediation process for potential and actual secSLA violations.

The proposed secSLA model and secSLA management methodology are based on defining a set of measurements for each security metric with which the CSPs are able to evaluate the validity of guaranteed SLOs. For each SLO we have identified a set of detectable alerts and violations, and for each deviation from what is assured and expected, we have defined actions to be taken in order to prevent or recover from secSLA violations. When developing remediation part of the secSLA management framework, we have considered not only how to predict, prevent, and eliminate secSLA violations, but also how to handle corrective and reactive actions in order to avoid causing more damage.

The procedure presented in this paper is focused on the management of security aspects represented by secSLAs. However, the approach is flexible enough to be applied to other contexts and domains (to performance and privacy SLAs or even to non-cloud scenarios where the CSCs run their services on off-premise resources) as long as SLOs, metrics, measurements, and remediation plans are defined. As mentioned in the related work, the cloud security domain has high need for standardized representation for both the security characteristics of cloud services and also security related guarantees offered by CSPs. Moreover, the cloud security domain currently lacks processes for clear and a transparent SLA management. Consequently, the SLA model and monitoring and remediation procedures presented in this paper specifically focus on management of secSLAs.

When referring to cloud monitoring (in security or any other domain), the trustworthiness is the main issue to overcome in our approach. It is well known that the current techniques and available technologies to actually monitor the SLAs are internal to CSPs, and the lack of trustworthiness in the

veracity of the information provided and handled by them might be a stopper to this kind of solutions. Current research activities supported by some working groups (including Cloud Security Alliance) are focused on the use of independent/external auditors and protocols to provide with a common model to get information from CSPs. However, as long as CSPs have the control of the data extracted from their infrastructure, the problem still persists. Developing mechanisms to check the validity and veracity of monitoring information (by enforcing proof-based systems based on attestations like the E2EE security mechanism developed in SPECS) remains as an open challenge and future work that will contribute to reach a higher level of trustworthiness and reliability to the proposed model.

Acknowledgments

The research was supported partly by FP7-ICT-2013-11-610795 (SPECS) and by H2020-ICT-2014-644579 (ESCUDO-CLOUD)

References

[1] C. Coles, J. Yeoh, Cloud adoption practices & priorities survey report, CSA, 2015. Available online, https://downloads.cloudsecurityalliance.org/initiatives/surveys/capp/Cloud_Adoption_Practices_Priorities_Survey_Final.pdf, last accessed in October 2015.

[2] KPMG, Cloud Survey Report: Elevating Business in the Cloud, 2014. Available online, https://www.kpmg.com/PL/pl/IssuesAndInsights/ArticlesPublications/Documents/2015/2014-KPMG-Cloud-Survey-Report-online-secured.pdf, last accessed in October 2015.

[3] Skyway Magazine by Eurocontrol, Why Eurocontrol built a cloud, 2013. Available online, http://www.eurocontrol.int/sites/default/files/content/documents/official-documents/skyway/articles/2013-spring-skyway-focus-why-ecl-built-a-cloud.pdf, last accessed in October 2015.

[4] SESAR project. http://ec.europa.eu/transport/modes/air/sesar/index_en.htm, last accessed in October 2015.

[5] M. Dekker, Security Framework for Governmental Clouds – ENISA, 2015. Available online, https://www.enisa.europa.eu/activities/Resilience-and-CIIP/cloud-computing/governmental-cloud-security/security-framework-for-govenmental-clouds, last accessed in October 2015.

[6] International Organization for Standardization, Guidelines on information security controls for the use of cloud computing services based on ISOIEC 27002, 2014.

[7] Cloud Security Alliance, Cloud Controls Matrix v3.0.1. Available online, https://cloudsecurityalliance.org/download/cloud-controls-matrix-v3-0-1/, last accessed in October 2015.

[8] National Institute of Standards and Technology, Security and privacy controls for federal information systems and organizations, NIST 800-53v4, 2014.

[9] J. Luna, N. Suri, M. Iorga, A. Karmel, Leveraging the Potential of Cloud Security Service-Level Agreements through Standards, IEEE Cloud Computing Magazine 2(3), 2015, pp. 32–40.

[10] European Network and Information Security Agency, Survey and analysis of security parameters in Cloud SLAs across the European public sector, 2011. Available online, https://www.enisa.europa.eu/activities/Resilience-and-CIIP/cloud-computing/survey-and-analysis-of-security-parameters-in-cloud-slas-across-the-european-public-sector, last accessed in October 2015.

[11] International Organization for Standardization, Information Technology -- cloud computing -- Service level agreement (SLA) framework and terminology (Draft), ISO/IEC 19086, 2014.

[12] National Institute of Standards and Technology, Cloud Computing: Cloud Service Metrics Description, NIST Public RATAX EG draft document, 2014. Available online, http://www.nist.gov/itl/cloud/upload/RATAX-CloudServiceMetricsDescription-DRAFT-20141111.pdf, last accessed in October 2015.

[13] European Commission, Cloud service level agreement standardization guidelines, C-SIG SLA 2014, 2014. Available online, http://ec.europa.eu/digital-agenda/en/news/cloud-service-level-agreement-standardisation-guidelines, last accessed in October 2015.
[14] S. Diver, Information security policy – A development guide for large and small companies, SANS Institute, 2007. Available online, https://www.sans.org/reading-room/whitepapers/policyissues/information-security-policy-development-guide-large-small-companies-1331, last accessed in October 2015.
[15] B. Monahan, M. Yearwort, Meaningful Security SLAs, technical report, HP Laboratories, 2008. Available online, http://www.hpl.hp.com/techreports/2005/HPL-2005-218R1.pdf, last accessed in October 2015.
[16] A. Undheim, K. Bernsmed, M. G. Jaatun, Security in Service Level Agreements for Cloud Computing, in: Proceedings of the 1st International Conference on Cloud Computing and Services Science (CLOSER 2011), 2011, pp. 636–642.
[17] Cloud Security Alliance, Security, Trust & Assurance Registry (STAR). Available online, https://cloudsecurityalliance.org/star/, last accessed in October 2015.
[18] V. Casola, A. Mazzeo, N. Mazzocca, M. Rak, A SLA Evaluation Methodology in Service Oriented Architectures, in: Proceedings of the 2nd ACM Workshop on Quality of Protection (QOP 2006), 2006, pp. 119–130.
[19] J. Luna, H. Ghani, D. Germanus, N. Suri, A Security Metrics Framework for the Cloud, in: Proceedings of the 6th International Conference on Security and Cryptography (SECRYPT 2011), 2011, pp. 245–250.
[20] A. Taha, R. Trapero, J. Luna, N. Suri, AHP-Based Quantitative Approach for Assessing and Comparing Cloud Security, in: Proceedings of Trust, Security and Privacy in Computing and Communications, 2014, pp. 284–291.
[21] R. M. Savola, On the Feasibility of Utilizing Security Metrics in Software-Intensive Systems, in: International Journal of Computer Science and Network Security (IJCSNS), 10 (1), 2010, pp. 230–239.
[22] Cloud Security Alliance, The Consensus Assessments Initiative Questionnaire, 2011. Available online, https://cloudsecurityalliance.org/group/consensus-assessments/, last accessed in October 2015.
[23] J. Luna, R. Langenberg, N. Suri, Benchmarking cloud security level agreements using quantitative policy trees, ACM Workshop on Cloud computing security, 2012, pp. 103–112.
[24] V. Casola, R. Preziosi, M. Rak, L. Troiano, A reference model for security level evaluation: Policy and fuzzy techniques, Journal of Universal Computer Science, 2005, pp. 150–174.
[25] E. Novikoff, The role of remote monitoring in Cloud Computing, 2014. Available online, http://enki.co/blog/the-role-of-remote-monitoring-in-cloud-computing.html, last accessed in October 2015.
[26] A. Grabner, Proof of Concept: dynaTrace provides Cloud Service Monitoring and Root Cause Analysis for GigaSpaces, 2009. Available online, http://apmblog.dynatrace.com/2009/05/07/proof-of-concept-dynatrace-provides-cloud-service-monitoring-and-root-cause-analysis-for-gigaspaces/, last accessed in October 2015.
[27] Nagios, http://www.nagios.org/, last accessed in October 2015.
[28] V. C. Emeakaroha, R. N. Calheiros, M. A. S. Netto, I. Brandic, C. A. F. De Rose, DeSVi: An Architecture for Detecting SLA Violations in Cloud Computing Infrastructures, in: Proceedings of the 2nd International ICST Conference on Cloud Computing (CloudComp 2010), 2010.
[29] Amazon CloudWatch, http://aws.amazon.com/cloudwatch/, last accessed in October 2015.
[30] J. Brower, The security Onion Cloud Client - Network Security Monitoring for the Cloud, SANS Institute, 2013. Available online, https://www.sans.org/reading-room/whitepapers/cloud/security-onion-cloud-client-network-security-monitoring-cloud-34335, last accessed in October 2015.
[31] N. Kroes (EC), Cyber Security – A shared responsibility, 2012. Available online, http://europa.eu/rapid/press-release_SPEECH-12-774_en.htm, last accessed in October 2015.

[32] CSA, Cloud Trust Protocol. Available online, https://cloudsecurityalliance.org/research/ctp/, last ccessed in October 2015.
[33] I. Brandic, V. C. Emeakaroha, M. Maurer, S. Dustdar, S. Acs, A. Kertesz, G. Kecskemeti, Laysi: A layered approach for sla-violation propagation in self-manageable cloud infrastructures, in: Proceedings of the 2010 IEEE 34th Annual IEEE Computer Software and Applications Conference Workshops (COMPSACW), 2010, pp. 365–370.
[34] O. F. Rana, M. Warnier, T. B. Quillinan, F. Brazier, D. Cojocarasu, Managing violations in service level agreements, Grid Middleware and Services, Springer US, 2008, pp. 349–358.
[35] Z. Zhang, L. Liao, H. Liu, G. Li, Policy-based adaptive service level agreement management for cloud services, in: Proceedings of the 2014 5th IEEE International Conference on Software Engineering and Service Science (ICSESS), IEEE, 2014, pp. 496–499.
[36] SLA@SOI project (IST-216556; Empowering the Service Economy with SLA-aware Infra-structures), http://www.sla-at-soi.org, last accessed in October 2015.
[37] M. Rak, N. Suri, J. Luna, D. Petcu, V. Casola, U. Villano, Security as a Service Using an SLA-Based Approach via SPECS, in: Proceedings of IEEE 5th International Conference on Cloud Computing Technology and Science (CloudComp), IEEE, 2013, pp. 1–6.
[38] SPECS project (ICT-610795; Secure Provisioning of Cloud Services based on SLA Management), http://www.specs-project.eu/, last accessed in October 2015.
[39] Micro Focus, OVAL Information, 2015. Available online, https://support.novell.com/security/oval/, last accessed in October 2015.
[40] The Open Vulnerability and Assessment Language (OVAL) repository. Available online, https://oval.cisecurity.org/repository, last accessed in October 2015.
[41] P. Mell, K. Scarfone, S. Romanosky, A complete guide to the Common Vulnerability Scoring System Version 2.0, 2007. Available online, https://www.first.org/cvss/cvss-v2-guide.pdf, last accessed in October 2015.
[42] Chef, An Overview of Chef, 2009. Available online, https://docs.chef.io/chef_overview.html, last accessed in October 2015.
[43] Puppet Labs, Puppet Enterprise Overview, 2005. Available online, http://docs.puppetlabs.com/pe/latest/overview_about_pe.html, last accessed in October 2015.
[44] SaltStack, SaltStack Walkthrough, 2011. Available online, https://docs.saltstack.com/en/latest/topics/tutorials/walkthrough.html, last accessed in October 2015.
[45] Ansible, Ansible documentation, 2015. Available online, http://docs.ansible.com/ansible/index.html, last accessed in October 2015.
[46] G. Davis, The Heartbleed Vulnerability: What It Is and How It Affects You, 2014. Available online, https://blogs.mcafee.com/consumer/what-is-heartbleed/, last accessed in October 2015.
[47] J. Luna, A. Taha, R. Trapero, N. Suri, Quantitative Reasoning About Cloud Security Using Service Level Agreements, IEEE Transactions on Cloud Computing, 99, 2015. Available online, http://www1.deeds.informatik.tu-darmstadt.de/External/PublicationData/0/TCC_secSLA_2014.pdf, last accessed in April 2016.
[48] SPECS, SPECS Team Bitbucket Account, 2015. Available online, https://bitbucket.org/specs-team/, last accessed in April 2016.
[49] M. Stopar, J. Modic, D. Petcu, M. Rak, Towards a Proof-Based SLA Management Framework – The SPECS Approach, in: Proceedings of the 2016 6th International Conference on Cloud Computing and Services Science (CLOSER), 2016, to appear.
[50] Amazon Web Services (AWS), https://aws.amazon.com/, last accessed in April 2016.
[51] Microsoft Azure, https://azure.microsoft.com/en-us/, last accessed in April 2016.

Ruben Trapero received his Ph.D. from Universidad Politécnica of Madrid and was an assistant professor at Universidad Carlos III of Madrid. Since 2014 he is a lead researcher at the Technische Universität of Darmstadt, Germany. His research interests include privacy, identity management, cloud security and service engineering.

Jolanda Modic received her Ph.D. in 2014 from the Faculty of Mathematics and Physics, University of Ljubljana. She is currently a researcher at XLAB, Ljubljana, Slovenia, and is focused on automated SLA management frameworks, and evaluating, monitoring, and enforcing security in cloud environments.

Miha Stopar received his B.Sc. in 2007 from the Faculty of Mathematics and Physics, University of Ljubljana. He is currently a researcher and a software developer at XLAB, Ljubljana, Slovenia. His research interests include cloud computing, machine learning, and applied cryptography.

Ahmed Taha is currently a Ph.D. student at DEEDS group in the Department of Computer Science at Technische Universität of Darmstadt, Germany. His research interest includes Cloud security metrics, security assessment and quantification.

Neeraj Suri received his Ph.D. from the University of Massachusetts at Amherst and is a Chair Professor at the Technische Universität of Darmstadt, Germany. His research addresses the design, analysis, and assessment of trustworthy cloud services.