Security-by-design in clouds: a Security-SLA driven methodology to build secure cloud applications

Valentina Casolaa, Alessandra De Benedictisb, Massimiliano Rakb,*, Erkuden Riosc

aUniversity of Naples Federico II, Napoli, Italy
bSecond University of Naples, Caserta, Italy
cTECNALIA, Bilbao, Spain

Abstract

This paper presents a security-by-design methodology for the development of cloud applications, which relies on Security SLAs as a means to express their security requirements. The process followed to build such Security SLAs entails the application of a risk analysis procedure aimed at identifying the main vulnerabilities affecting a cloud application and allows to determine the countermeasures to consider at design time in order to thwart the main existing threats.

The paper illustrates a proof-of-concept application that founds on standard risk assessment tools and adopts state-of-art Security Control Frameworks and a novel Security SLA model for the security requirements representation.

Keywords: Security by design; Secure Cloud Applications; Secure Multi-cloud Applications; Security SLA; Threat analysis

1. Introduction

Cloud security is considered one of the main inhibitors for cloud adoption nowadays1,2: delegating resources and data to the cloud has the effect of a loss of control that makes risk analysis and mitigation more complex and prevents prospective customers from cloudifying their applications. In order to overcome this reluctance, the cloud application development process should take into account the potential security issues from the beginning, and should adapt to the flexibility offered by the cloud paradigm while also considering the security constraints posed by developers and cloud customers. The above expressed need is tackled by security-by-design approaches, which aim to make systems as free from vulnerabilities as possible by taking into account security from the very early stages of the design process. However, security-by-design is a complex practice that is not always addressed in common software, even in enterprise solutions, and which needs the definition of proper methodologies and tools, especially in contexts such as the cloud

* Corresponding author. Tel.: +39-081-5010-336.
E-mail address: massimiliano.rak@unina2.it
where providers often do not state clearly the security features they are able to offer with their services, thus making it more complex for developers to estimate possible risks.

Regarding this point, Service Level Agreements (SLAs) including security-related terms would represent a powerful and effective means to express security guarantees offered by Cloud Service Providers (CSPs), and may be also used by customers to negotiate, if possible, the (security) features of the acquired services. Security SLAs and their management have been the focus of the SPECS EU FP7 project, just closed, whose main achievements were (i) the definition of a novel Security SLA model and the implementation of a (ii) platform and a framework for the development, deployment and execution of secure cloud applications offering services covered by Security SLAs. The results of the SPECS project have been leveraged by the MUSA EU H2020 project, recently started, which focuses on the security-intelligent management of multi-cloud applications via a security-by-design development approach based on Security SLAs and on a runtime security assurance platform for their monitoring.

This paper is the result of a joint research activity carried out within the SPECS and MUSA projects. In particular, it presents the security-by-design methodology proposed in MUSA for the development of multi-cloud applications, which strongly relies on the adoption of Security SLAs. With respect to such methodology, the paper illustrates the SLA Generation process that allows to obtain a Security SLA for each (multi-)cloud application component, allowing to define their security requirements from the very early stages of their development. In our discussion, a cloud application component is a software component that implements a specific functionality of the application (e.g., a web container, a database) and that may offer/use a (cloud) service. The proposed SLA Generation process (of which we implemented a proof-of-concept application), entails the adoption of risk analysis techniques aimed at identifying the main vulnerabilities affecting a cloud application based on the nature of its software components, and allows to determine the countermeasures to take into account at the design stage in order to thwart the main existing threats. In particular, countermeasures are defined in terms of the security controls to apply (based on state-of-art Security Control Frameworks) and can be put in place by implementing specific security mechanisms to be integrated into the cloud application under development.

The reminder of this paper is organized as follows. Section 2 provides an overview of the research background behind this paper, represented by the objectives and results of the SPECS and MUSA projects, while Section 3 illustrates the high-level security-by-design methodology proposed in MUSA for the development of multi-cloud applications. Section 4 provides the details on the SLA Generation process and discusses the conceptual models on which it founds. Section 5 describes the SLA Generation proof-of-concept application, which evidences the applicability of the approach, and illustrates the underlying adopted tools. Finally, Section 6 illustrates the related work on SbD approaches in cloud and Section 7 summarizes our conclusions.

2. Research background: the MUSA and SPECS projects

As anticipated, the piece of research described herein was born in the context of the FP7 project SPECS (Secure Provisioning of cloud services based on SLA management) and the Horizon 2020 project MUSA (Multi-cloud Secure Applications). In regards to the Security SLA-driven methodology proposed, MUSA is the successor of SPECS and enhances its work from a single to multi-cloud approach. In the following, we present a brief overview of the main objectives and results of the two projects, in order to provide the reader with the needed background.

The SPECS project, started in 2013 and closed in April 2016, aimed at designing and developing (i) an open-source framework offering a number of techniques and tools for the systematic management and assurance of cloud Security SLAs, and (ii) a platform devoted to providing secure cloud services based on a Security-as-a-Service approach. In particular, the SPECS framework offers the techniques and tools supporting the (re-)negotiation, enforcement, monitoring and remediation phases of the Security SLA life cycle, while the SPECS platform provides the services for the automatic management of these activities. The SPECS platform includes five modules, namely the (i) Negotiation module, the (ii) Enforcement module and the (iii) Monitoring module, devoted to the management of the Security SLA life cycle phases, the (iv) SLA Platform, responsible for the management of Security SLAs and of their status, and the (v) Enabling Platform, which provides a seamless deployment and execution environment for the other modules.

In the typical SPECS operation scenario, the SPECS platform is used to build applications (SPECS applications) offering cloud services provided with specific security features, negotiated by customers and stated in a Security SLA.
The negotiation is based on templates, which summarize the set of available features in terms of enforceable security capabilities (8) and monitorable security metrics. The services made available by the SPECS Enforcement module are used, during the SLA enforcement phase, to enhance unsecure cloud services provided by external CSPs with the automatic installation and configuration of proper security mechanisms and of related monitoring systems, in order to provide the customer with a means to verify that the negotiated Security SLA is respected during service operation.

The Security SLA model adopted in SPECS is based on and extends the WS-Agreement standard9, and was built by taking into account the latest directives and standards in the fields of cloud security10,11,12. It is described in detail in the paper13, while the paper14 illustrates the process of developing secure cloud applications with SPECS.

The MUSA project, started in January 2015, builds on top of some of the results of SPECS to develop a framework for the security-intelligent management of multi-cloud applications, which includes methods and tools for all the phases in the application life-cycle. Similarly to SPECS, the MUSA framework will include a Security-as-a-Service platform for the Security SLA assurance, but in multi-cloud environments. The MUSA framework combines a preventive security approach, promoting security-by-design practices in the development of multi-cloud applications, and a reactive security approach, for monitoring the application runtime in order to mitigate security incidents. One of the first achievements of the project was the definition of the initial methods and tools supporting the security-by-design development of multi-cloud applications. The SLA Generation process proposed in this paper, along with the conceptual models on which it founds, are part of these activities and are described in detail in the next sections. However, before going into the details of the SLA Generation activity, it is worth illustrating the MUSA Security SLA-based security-by-design multi-cloud application development process, which is sketched in the next section.

3. The MUSA security-by-design development process

The MUSA Security SLA-based security-by-design multi-cloud application development process consists of the five activities summarized in Figure 1:

1. **Design Modelling**: the cloud application is modelled in terms of its Cloud Provider Independent Model (CPIM), which details the information about the cloud services used by the application components.

2. **SLA Generation**: (focus of this paper) the security requirements at component level are identified and formalized in terms of Security SLAs. Each component is analysed based on a standard threat modelling procedure, and a risk analysis process is carried out with focus on each component, in order to identify the main risks associated with the security issues that may arise. This activity will be discussed in detail in the next section.

3. **SLA Feasibility Verification**: an analysis on the feasibility of resulting Security SLAs is conducted, in order to check whether any existing provider can fulfil desired security requirements. The SLA feasibility phase takes as input the set of Security SLAs and produces as output the list of possible available offers (i.e., the list of available services provided by CSPs) along with the list of (possibly) missing security controls. The developer has to analyse the result of the SLA Feasibility process: if the Security SLAs cannot be satisfied with existing services, a set of security libraries must be added to fulfil the uncovered security requirements, thus requiring an update in the application model. If instead the Security SLAs are feasible, the SLA Composition activity takes place.
4. **SLA Composition**: A Security SLA for the whole cloud application is derived by composing the Security SLAs of single components. This SLA includes all the security guarantees that can be offered on top of the application to the cloud customers. Note that, even if each component satisfies its own security requirements, this does not grant that the overall cloud application is able to guarantee the desired level of security. As an example, consider the case of a simple application made up of two components, namely a web server and a database. When executing the SLA Generation activity for both components separately, we may not take into account for example that, when using them together, also the communication channel should be protected (as a general requirement of the application). In general, if the composed Security SLA fulfils the initial cloud application requirements, the implementation can take place, otherwise a new design modelling activity is required. It is worth noticing that, in a second iteration, the analysis made in the first round would be available in the form of an already elaborated security SLA, which now should only be updated according to higher-level requirements.

5. **Add Security libraries**: If any of the components Security SLAs results unfeasible, i.e., if there is no provider that already offers the required components with the needed security features, the missing security features are added, if possible, by means of the activation of proper security mechanisms available commonly in form of libraries, whose integration may imply the deployment and configuration of additional components. Let us consider for example the case of a cloud application component represented by a storage service, and let us assume that, as a result of the risk analysis process, the end-to-end encryption of user data is recommended. If none of the known cloud storage providers is able to provide such a feature, this may be added in form of a proper library, embedded in the client, which is responsible for the encryption of data before its submission to the remote storage service. In this case, the design activity must be repeated, in order to take into account the possible new security components to add to the application.

It is worth mentioning that MUSA and SPECS share several common points. The enhancement of a cloud component with additional security libraries introduced in the last phase of the MUSA development process is very similar to what happens in SPECS during the Enforcement phase, when proper security mechanisms are activated to provide the missing security features. In MUSA, however, this process is addressed at design time since it enables to consider the security libraries as additional components to take into account during the design of the application, while in SPECS the enhancement is carried out at runtime. Even the SLA Feasibility activity defined above was originally devised by SPECS, but it is carried out before, during the negotiation phase, when the Enforcement module is invoked to verify if the SLA under negotiation can be actually implemented (the details of this process can be found in 13). Moreover in the next section, we will show how the proposed SLA Generation process can be integrated within the SPECS flow to simplify the definition of security requirements.

### 4. SLA Generation activity

In this section, we illustrate the methodology proposed to build the **required** Security SLAs for the components of a cloud application. This methodology enables not only to clearly define the components’ security requirements according to the MUSA flow, but it also allows to start a negotiation process (as the one proposed in SPECS) for the acquisition of the cloud services having the desired security features.

As anticipated in the introduction, the SLA Generation process adopts a risk analysis procedure to identify the security requirements of a cloud application component, which are then expressed by means of the Security SLA model proposed in SPECS. How are Security SLAs linked to the concept of risk? In order to answer this question and provide the reader with the needed background to understand our proposal, in the following section we illustrate the underlying concepts used to build the SLA Generation process, and after we illustrate the process itself.

#### 4.1. Security SLAs and risk analysis

Figure 2 shows a simple conceptual model that illustrates the relationships among the concepts of Security SLA, security controls, vulnerabilities, threats and risk. As shown, cloud application components (**App component** in the figure) use, in general, **cloud services**, which may belong to several **cloud service types** (i.e., IaaS, SaaS, PaaS) and which constitute the **tangible assets** of the application under design. Each component belongs to a general **component**
type, which mainly refers to its functional behaviour (i.e., the component acts as a database, as a web application, etc.).

Depending on the cloud service type it belongs to and on its component type, a component may be subject to specific vulnerabilities. If these vulnerabilities are actually exposed, specific threats may exploit them, therefore posing a risk. Let us consider as an example the case of a web application. Web applications are typically subject to code injection, but this vulnerability can be exploited only if the application evaluates the input from customers, while if user input is never evaluated the code cannot be injected. This means that the exposure of a component to vulnerabilities depends on its behaviour and on how it is implemented. The threats pose risks that can be mitigated through the enforcement of proper treatments, which represent specific security controls to apply. Security controls are included in a Security SLA together with associated security metrics, which can be used to measure the level at which they are actually enforced. Security controls belong to a selected Security Control Framework (e.g., the NIST Security Control Framework or the Cloud Security Alliance’s Cloud Control Matrix), while security metrics are defined within a Security Metric Catalogue. In this regard, note that in SPECS a Security Metric Catalogue was introduced, adopted and extended by MUSA with the metrics relevant to its case study applications. The above discussed model suggests that, in order to produce the components Security SLAs, it is possible to carry out a risk analysis process for each component (based on its type and behaviour) and to identify the security controls that can be applied to mitigate the existing risks.

For what regards the formalization of Security SLAs, as said we refer to the Security SLA model introduced by the SPECS project. Such model was slightly updated to include the concept of component type and all the threat-related information (not taken into account in the original SPECS model). The resulting model, adopted to support the security-by-design approach, is depicted in Figure 3. As shown, a Security SLA includes (i) a declarative section, where the application component features (both functional and non functional) are declared, which reports the component type, the associated threats, the required security capabilities (i.e., sets of security controls) and related security metrics, and (ii) a measurable section, where the desired security service level objectives (SLOs) for the component are specified (SLOs are built from security metrics declared above). The information related to the existing threats and to the required security capabilities is obtained as a result of the risk analysis process, while SLOs are defined by the developer based on his specific requirements. The whole process is illustrated in the next section.

4.2. The SLA Generation process

The SLA Generation process is summarized in Figure 4. As shown, it consists in the following steps:

1. **Design analysis**: this phase takes in input the description of the application in terms of its components (i.e., the CPIM model produced in the Design Modelling activity of the developed process described in Section 3.
In this phase, the description of each component of the application is enriched with a set of properties, which characterize the component from the internal and external behaviour point of view. These properties are related to specific implementation and usage choices related to the component, known by the developer at design time and having an impact on the exposure to attacks and vulnerabilities.

2. **Risk analysis**: in this phase, based on the components’ type and on their properties (defined at the previous step), the main threats affecting each of the application’s components are identified. After, threats are classified and a risk assessment procedure is applied to identify their potential impact if exploited. Finally, for each threat, based on the risk previously associated, a set of treatments (i.e., security controls) that mitigate the threat are identified.

3. **SLA generation**: in this phase, for each threat encountered at the previous step, it is possible to select one or more security controls to enforce among those identified at the previous step. Based on the model outlined in Section 4.1, security controls are associated with security metrics, which are used to enforce/monitor them. In this phase then, it is also possible to select the metrics of interest and to set related SLOs, used to build up a Security SLA for each component of the application. Such SLAs represent the requested SLAs, namely the security requirements of each component.

It is worth noticing that the SLA Generation process presented above can be adopted as an alternative to the current way of building SLAs in SPECS for negotiation. In its original definition in fact, the SPECS flow requires that a SPECS Customer (e.g., a cloud developer) is able to express his security requirements directly in terms of the controls to enforce and of the metrics to monitor. The approach presented here, instead, allows a developer to simply define the behaviour of the components of the cloud (or multi-cloud) application he is going to develop, while the SLA terms are automatically derived from an underlying threat analysis process.
5. A proof-of-concept SLA Generation application

In this section, a proof-of-concept application implementing the described SLA Generation process is presented. The application implements the steps discussed in Section 4.2 as described in the following subsections.

5.1. Design analysis

As said when discussing the general development process, during the Design Modelling phase the developer has to specify the main components of the application in terms of their nature and behaviour. Our proof-of-concept application supports the developer in this preliminary phase by enabling him to specify the application components and to assign them a type. Three types of components are currently supported:

- **custom web app**: the component is a custom web application, accessible via a network (either private or public). The web application offers specific functionalities through an application server, which can be consumed by generic clients represented by a web browser;
- **software-as-a-service (SaaS)**: the component is an external service that is not under the control of the developer but is provided by a third party;
- **storage-as-a-service**: the component is a storage service offered by an external provider.

During the design phase, the developer is asked to specify the component name, description and type among those discussed above (cf. Figure 5). Once this information is provided, the design analysis phase can take place, where the application model is enriched with the assignment of specific properties to each application component, useful to assess the associated risk and determine the related security requirements.

In order to enable the subsequent threat analysis, our proof-of-concept application has been preloaded with a set of associations <threat - component type>, which basically define the threats a component is subject to based on its type. These associations have been built by following several existing guidelines for the development of secure applications. In particular, the OWASP Top 10\(^\text{16}\) was considered to identify the most critical web application security flaws, and several security bulletins and other relevant sources carefully analysed to find the main threats against other component types.

The result of this process was a Threat Catalogue, adopted by the SLA Generation application to identify the list of possible threats for each component of the application. An extract of the Threat Catalogue is provided in Table 1, which reports, for each threat, the affected component types. Moreover, for each threat, we also provide the mapping with the STRIDE categorization methodology\(^\text{17}\), an approach developed by Microsoft for identifying and classifying computer security threats. STRIDE considers six threat categories: spoofing, tampering, repudiation, information...
Table 1. Extract of the Threat Catalogue used by the SLA Generation application

| Threat                        | Description                                                                 | STRIDE cat. | Component type          | Condition                                                                 |
|-------------------------------|-----------------------------------------------------------------------------|-------------|-------------------------|---------------------------------------------------------------------------|
| Account Hijacking             | Account hijacking is a process through which an individual’s email account, computer account or any other account associated with a computing device or service is stolen or hijacked by a hacker. It is a type of identity theft in which the hacker uses the stolen account information to carry out malicious or unauthorized activity. | SPOOFING    | custom web app, SaaS    | The component manages user accounts                                        |
| Advanced Persistent Threats (APTs) | An advanced persistent threat (APT) is a network attack in which an unauthorized person gains access to a network and stays there undetected for a long period of time. The intention of an APT attack is to steal data rather than to cause damage to the network or organization. APT attacks target organizations in sectors with high-value information, such as national defense, manufacturing and the financial industry. | REPUDIATION | custom web app          | The component has a state and computation resources                       |
| Cross-Site Request Forgery (CSRF) | A CSRF attack forces a logged-on victims browser to send a forged HTTP request, including the victims session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victims browser to generate requests the vulnerable application thinks are legitimate requests from the victim. | TAMPERING    | custom web app, SaaS, storage-as-service | The component exposes an interface through which it is possible to send malicious requests |

**disclosure, denial of service and elevation of privileges.** These categories will be used later, in the risk analysis phase, to group risk values related to each threat. Finally, for each threat, we report the condition that, if verified, enables exploiting it. Indeed, whether the component is actually subject to a certain threat may strongly depend on how the component has been developed or on how it is meant to be used. For this reason, based on the description of the threat and on the enabling conditions elicited from it, we prepared a questionnaire aimed at guiding the developer in identifying, for each threat, the real issues related to the implementation and usage of the component, so to discard threats that are not actually solicited (cf. Figure 6). The questions were simply built by analysing the description of each threat, and by extracting the relevant information. The set of relevant threats resulting from the filtering process is the input for the risk analysis phase.
5.2. Risk analysis

In the risk analysis phase of our proof-of-concept application, the developer is guided through a risk classification process based on the OWASP Risk Rating Methodology\textsuperscript{18}. For each threat, the developer has to assign a score (in the range [0-9]) to each of the available indicators belonging to the probability of occurrence (likelihood) and impact categories, which are combined to obtain a final risk value (risk is calculated or defined as the product of likelihood and impact: \textit{Risk} = \textit{Likelihood} \times \textit{Impact}). The risk values associated with each threat are then grouped according to the categories identified by the STRIDE methodology, producing 6 different risk values as result, one for each category. This grouping is done to give the user a comprehensive view, so to be able to identify which of the six aspects is more risky (this usually depends on the nature of the component and of the offered functionalities).

5.3. SLA generation

The first step of the SLA Generation phase consists in associating a set of security controls to each threats identified at the previous step. In our proof-of-concept application, the NIST security Control Framework\textsuperscript{8} has been considered as the source of security controls. In practice, the application suggests the enforcement of a set of security controls in order to cope with the existing threats, based on the associated risk. The association is done statically, based on a mapping built ad-hoc. Moreover, the application also presents the developer with the controls associated with each of the six STRIDE categories, in order to give him/her a wider choice. The developer can select the security controls of interest, which will be included in the SLA of the component being analysed. After the selection of the security controls, the relevant security metrics must be chosen and related SLOs must be defined. Security metrics are statically mapped to security controls, therefore the developer is prompted with applicable metrics and is asked to select those of interest and to define related objectives. Finally, all selected controls and metrics with related SLOs are included in an SLA. For each component, an SLA is prepared according to the model presented in Section 2.

6. Related work

Security-by-design requires that security be taken into account from the very early stages of the design process. Kreizman and Robertson, with their Gartner whitepaper\textsuperscript{19}, were probably the first to position security-by-design principles in the enterprise context, but the need for a deep security analysis from early development stages is well outlined in literature\textsuperscript{20,21,22} even if\textsuperscript{23} outlines the immaturity of the field and the lack of tools and methods for formal threat modeling. In the processes proposed by such papers, security threat modeling, which is the process of identifying, documenting and mitigating security threats affecting a software system, assumes a key role. In the context of cloud computing, where security is considered one of the main inhibitors, the role of such techniques becomes very relevant, despite the lack of concrete techniques and tools to address such solutions. Risk analysis for cloud adoption was proposed by some recent papers in literature\textsuperscript{24,25}, but no one clearly addressed the process of cloud application development according to such risk analysis process.

For what regards the adoption of Security SLAs to represent the security features offered by CSPs, a lot of activities exists today on this topic, supported by ENISA considerations\textsuperscript{2}. Moreover, as already outlined, several projects have addressed the use of Security SLAs, such as SPECS, MUSA, SLA-Ready\textsuperscript{26} and SLALOM\textsuperscript{27}.

7. Conclusions

This paper discussed the results of the joint research activity carried out in the context of two European projects, SPECS and MUSA, both focused on providing techniques and tools for the development of secure applications in the cloud based on the adoption of Security SLAs. In particular, the paper illustrated one of the preliminary results of the MUSA project, consisting in a security-by-design methodology for the development of multi-cloud applications, strongly relying on Security SLAs as a means to specify the security requirements of the applications and of their components. With respect to such process, the paper focused on the steps leading to the generation of per-component Security SLAs, and presented a SLA Generation application and a number of supporting tools for its implementation.
Our SLA generation application currently manages a reduced set of component types, that we plan to extend to take into account, as much as possible, how current cloud and multi-cloud applications are built. Moreover, with respect to the general security-by-design process, we plan to investigate on the SLA composition phase, in order to obtain a feasible Security SLA of the whole application from the Security SLAs built for each of its components.

Acknowledgements

The project leading to this paper has received funding from the European Union’s Horizon 2020 Research and Innovation Programme under Grant Agreement No. 644429 (MUSA).

References

1. Deloitte, . Executive forum - cloud computing: risks, mitigation strategies, and the role of internal audit. 2010. URL: http://www.deloitte.com.
2. European Network and Information Security Agency (ENISA), . Procure secure. a guide to monitoring of security service levels in cloud contracts. 2012.
3. SPECS project web site. 2016. URL: http://www.specs-project.eu.
4. MUSA project web site. 2016. URL: http://www.musa-project.eu.
5. The proof-of-concept SLA Generation application. 2016. URL: http://37.48.247.125/TESI-0.0.1-SNAPSHOT/welcome.
6. Casola, V., De Benedictis, A., Rak, M., Villano, U.. Preliminary Design of a Platform-as-a-Service to Provide Security in Cloud. In: CLOSER 2014 - Proc. of the 4th Int. Conf. on Cloud Computing and Services Science, Barcelona, Spain, April 3-5, 2014. 2014, p. 752–757.
7. Rak, M., Suri, N., Luna, J., Petcu, D., Casola, V., Villano, U.. Security as a service using an SLA-based approach via SPECS. In: Proc. of CloudCom, 2013 IEEE 5th Int. Conf. on; vol., 2. 2013, p. 1–6. doi:10.1109/CloudCom.2013.165.
8. National Institute of Standards and Technology, . Recommended Security and Privacy Controls for Federal Information Systems and Organizations. Tech. Rep.; 2013. URL: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf.
9. Czarnecki, K., Aoto, R., Vukolic, K., Waczka, R.. Web services agreement specification (WS-Agreement). In: Global Grid Forum. The Global Grid Forum (GGF); 2004.
10. 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.
11. European Commission – C-SIG (Cloud Select Industry Group) subgroup. Cloud Service Level Agreement Standardisation Guidelines. 2014.
12. Cloud Standard Customer Council - CSCC,. The CSCC practical guide to cloud service level agreements. April 2012.
13. Casola, V., De Benedictis, A., M.Rak, , Modic, J., Erascu, M.. Automatically enforcing Security SLAs in the cloud. IEEE Transactions on Services Computing (PrePrints) 2016.
14. Casola, V., De Benedictis, A., Rak, M., Villano, U.. SLA-Based Secure Cloud Application Development: The SPECS Framework. In: 2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC). 2015, p. 337–344. doi:10.1109/SYNASC.2015.59.
15. Cloud Security Alliance. Cloud Control Matrix v3.0. https://cloudsecurityalliance.org/download/cloud-controls-matrix-v3/; 2016.
16. The OWASP Top 10 2013. Project Home Page. 2016. URL: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.
17. The STRIDE Threat Model. 2016. URL: https://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).asp.
18. The OWASP Risk Rating Methodology Wiki Page. 2016. URL: https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology.
19. Kreitzman, G., Robertson, B.. Incorporating security into the enterprise architecture process. 2006. URL: https://www.gartner.com/doc/488757/incorporating-security-enterprise-architecture-process.
20. Crook, R., Ince, D., Lin, L., Nuseibeh, B.. Security requirements engineering: When anti-requirements hit the fan. In: Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on. IEEE; 2002, p. 203–205.
21. Rushby, J.. Security requirements specifications: How and what. In: Symposium on Requirements Engineering for Information Security (SREIS); vol. 441. Citeseer; 2001.
22. Myagmar, S.. Threat Modeling as a Basis for Security Requirements. In StorageSS ‘05: Proceedings of the 2005 ACM workshop on Storage security and survivability 2005;:94–102.
23. Oladimeji, E., Supakkul, S., Chung, L.. Security threat modeling and analysis: A goal-oriented approach. Proc. of the 10th IASTED International Conference on Software Engineering and Applications SEA 2006 2006;:13–15URL: https://pipeline.utdallas.edu/~eao151000/documents/SecurityThreatModeling.pdf.
24. Gupta, S., Mantes-Mulero, V., Mathews, P., Dominik, J., Omorovic, A., Aranda, J., et al. Risk-driven framework for decision support in cloud service selection. In: Cluster, Cloud and Grid Computing (CCGRID), 2015 15th IEEE/ACM International Symposium on. 2015, p. 545–554. doi:10.1109/CCGRID.2015.111.
25. Martens, B., Teuteberg, F.. Decision-making in cloud computing environments: A cost and risk based approach. Information Systems Frontiers 2012;14(4):871–893. URL: http://dx.doi.org/10.1007/s10796-011-9317-x. doi:10.1007/s10796-011-9317-x.
26. SLA-Ready project web site. 2016. URL: http://www.sla-ready.eu.
27. SLALOM project web site. 2016. URL: http://slalom-project.eu.