Preliminary Design of a Platform-as-a-Service to Provide Security in Cloud

Valentina Casola[1], Alessandra De Benedictis[1], Massimiliano Rak[2] and Umberto Villano[3]
[1]*Università "Federico II" di Napoli, Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione, Napoli, Italy*
[2]*Seconda Università di Napoli, Dipartimento di Ingegneria dell'Informazione, Aversa, Italy*
[3]*Università del Sannio, Dipartimento di Ingegneria, Benevento, Italy*

Keywords: Cloud, Cloud Security, SLA, Requirements.

Abstract: Cloud computing is an emerging paradigm, widely adopted in distributed and business computing. Nevertheless, the biggest issue with the large adoption of cloud computing is the perception of loss of security and control over resources that are dynamically acquired in the cloud and that reside on remote providers, and the strong integration of security mechanisms into system architectures. This paper deals with the integration of security features into cloud applications by an as-a-service approach, using Service Level Agreements as a means to clearly define rights and constraints of both customers and providers. The goal is to show the main requirements of a platform dedicated to security and to present the global architecture, in terms of components and their interactions, devoted to negotiate, monitor and enforce the security mechanisms to be applied over existing cloud providers.

1 INTRODUCTION

Cloud computing, based on features as *on-demand self-service* and on the *pay-per-use* business model, is currently the emerging paradigm for distributed computation and has been recently indicated as a strategic asset by the European Commission (Commission, 2011), which is assuming a set of initiatives (e.g. European Cloud Partnership (Commission,)) aimed at supporting its adoption in both private and public bodies.

The biggest issue with the large adoption of cloud computing is related to security (Dekker, 2012; Catteddu, 2011), as the paradigm assumes that all resources are *delegated to the cloud* (i.e., hosted and managed by a remote cloud provider, and can be hardly ever physically located), while typically cloud providers have their own policies for the management of reserved data and privacy. The approach almost universally followed to define guarantees on the provisioning of a service is the introduction of *Service level Agreements* (SLAs). SLAs should specify *Service Level Objectives* (SLOs) dedicated to security, but there is still a lack of standards for the definition of the security terms in a SLA, even if a lot of work is ongoing in this field by dedicated standard groups (as the SLA C-SIG from the European Commission, the CSCC SLA group(CSCC,)) and research projects (see CUMULUS (Pannetrat et al., 2013), A4Cloud (Pearson, 2011), and SPECS (Rak et al., 2013)).

In addition to the above discussed issues, a further problem is linked to the "traditional" design of security systems, which often requires *full control* of the systems to be protected, in contrast with the cloud principles that would require an approach based on *modular* security (Battista et al., 2013). To this end, the FP7 has funded the *SPECS* Project (Rak et al., 2013), which proposes a cloud Platform-as-a-Service (PaaS) dedicated to provide Security-as-a-Service using SLAs. In this paper, we present a preliminary requirement analysis for platforms like SPECS and identify the main involved components. In order to clarify the whole analysis process, we illustrate the simple methodology adopted to derive requirements, which can be successfully applied to the design of any PaaS.

The remainder of this paper is organized as follows. Section 2 illustrates the context of our work, while in Section 3, we present an overview of most relevant PaaS solutions and of the proposed platform. In Section 4 we show, for the most representative functionalities we are interested at offering, the whole extraction process of requirements. Finally, in Section 5 a preliminary platform architecture is presented, and

Casola V., De Benedictis A., Rak M. and Villano U..
Preliminary Design of a Platform-as-a-Service to Provide Security in Cloud.
DOI: 10.5220/0004979307520757
In *Proceedings of the 4th International Conference on Cloud Computing and Services Science* (MultiCloud-2014), pages 752-757
ISBN: 978-989-758-019-2

in Section 6 we draw our conclusions and identify future work directions.

2 MOTIVATION AND CONTEXT

In this Section, we provide some representative scenarios that outline the main open issues related to the management of security requirements in a cloud environment, and help to evidence the main functionalities required from a security-oriented platform. As a first scenario, let us consider a cloud federation whose set-up is explicitly subject to security requirements (i.e. the federated providers must ensure a set of security features). At the state of the art, the evaluation of security attributes related to a cloud provider is carried out *by hand* (for example, based on the provider's existing ISO/IEC 27000-series certifications), as there are no automatic tools to search, among the available cloud offerings, those matching specific security attributes. This is mostly due to the lack of standards to specify the requested/provided security parameters. Recently, some steps have been taken for improving transparency and assurance in the cloud by the Cloud Security Alliance, with its STAR (Security, Trust & Assurance Registry) initiative. However, even if STAR is a powerful means to help users assess the security of cloud providers, there is still a lack of mechanisms to effectively evaluate and compare different offerings, given a specific request.

Similar issues can show up even in the traditional interactions between cloud providers and end-users. Let us consider, for example, a cloud end-user that wishes to acquire a secure cloud storage service to remotely store data with specific data confidentiality requirements. With the currently available approaches, the user has first to manually search for cloud providers with the desired storage features, then check their SLAs in order to verify the kind of confidentiality features offered, study the different available offerings, and finally select the one that fulfills as much as possible its requirements. As for the previous case, it would be desirable to provide the end-users with a single access point to specify their functional and non-functional (security) requirements and with automatic tools to search for and to select the best offering. Moreover, in both scenarios, once a security control has been guaranteed from a provider, the customer should be provided with proper tools to monitor the correct service provisioning while, at the state of art, the customer must accept the provider monitoring services and metrics, and can only create local measurements to obtain further information (e.g., custom local measurements of response time).

3 DEFINING A SECURITY-ORIENTED PAAS

As stated, our main goal is the design and the implementation of a PaaS dedicated to security services, based on an SLA approach. At the state of the art, PaaS solutions are very different from one another, and there is no standard interpretation of how to build them. The most common solution is based on adapting application servers (like Apache Tomcat), in order to decouple application deploying from virtual machines acquired over Infrastructure-as-a-Service solutions. Examples of such solutions are offered by FP7 projects as cloud4SOA (Zeginis et al., 2013), OPTIMIS (OPTIMIS,) or Contrail (in the ConPaaS module (Pierre and Stratan, 2012)). Such PaaS solutions are usually dedicated to a single technology (in the most common case, web applications), which is cloud-enabled in a way completely transparent to application developers. This approach is based on the idea of porting well-known middlewares to the cloud, and of making transparent to middleware users the adoption of the cloud paradigm for lower level resource acquisition. In other words, applications (and their developers) are not aware of the cloud.

An alternative approach is the one proposed by integrated stacks as Google App Engine (GAE) or Microsoft Azure, where developers must adopt dedicated APIs to build cloud-enabled applications. Such platforms are integrated with the offering of the CSP, who manages the full stack of services (SaaS, PaaS and IaaS). Applications are developed using dedicated toolkits: in this case, application developers are aware of the adoption of the cloud paradigm and have a set of dedicated APIs that enable them to exploit the cloud flexibility in their applications. As a drawback, applications are dependent on the platform, and cannot be used over different cloud providers without modifications.

Finally, a further approach has been proposed in solutions as mOSAIC (Petcu et al., 2013) or openshift (RedHat,). These adopt a dedicated *cloudware*, which can be deployed over infrastructure resources and offers APIs similar to the ones offered by GAE or Azure, but independent of the CSP that delivers the (infrastructure-level) resources. Our goal is to start from such solutions and build a PaaS dedicated to offering security (Security-as-a-Service) through an SLA-based approach.

As the first step towards the design of the platform, we analyzed the possible interactions with prospective customers, identifying three interaction models illustrated in details in Section 3.1. After that, in order to define the platform requirements, we

adopted a simple methodology based on the following steps:

- we extracted the required macro-functionalities from some example scenarios described in Section 2 and, for each macro-functionality, we identified the involved actors and roles, and the different usage perspective deriving from the different interaction models defined in the previous phase;
- we described each usage perspective by means of an high-level interaction diagram, pointing out the main actions performed by the platform; from this functionalities we identified a preliminary set of high-level use cases from the previous interaction diagrams;
- we classified the resulting use cases into different service layers and carried out a detailed analysis of each service to identify lower level use cases.

Starting from the above described requirement analysis, we identified a preliminary architecture that subdivides the services offered by the platform in different modules and outlines their interactions. Section 5 will describe such preliminary design.

3.1 Actors, Roles and Interaction Models

The platform is intended for different users and can serve different purposes: both end-users and CSPs can access it in order to (i) specify their cloud security requirements by means of SLAs, (ii) automatically evaluate and compare security features offered by remote CSPs (brokering of security services), (iii) enhance a remote service based on specific security requirements, (iv) manage the full SLA life-cycle (negotiation, monitoring and enforcement), and (v) develop and deploy security services that are *cloud SLA-aware*, implemented as a flexible open-source PaaS.

Such functionalities imply that the involved actors can play different roles depending on the specific functionality they invoke (e.g., developer, administrator, customer ...). Moreover, based on the specific hosting configuration and on the resulting different security and responsibility issues, we can identify three different ways to interact with the platform, which introduce further role classifications. In order to have a clear and shared role definition, we will adopt the high-level stakeholder taxonomy proposed in the Cloud Standards Coordination (CSC) initiative launched by ETSI (ETSI, 2013), which basically considers the roles of **customers**, **providers**, and **partners** for each party (represented by either an individual or an organization). While customers and providers are those asking for a service and offering

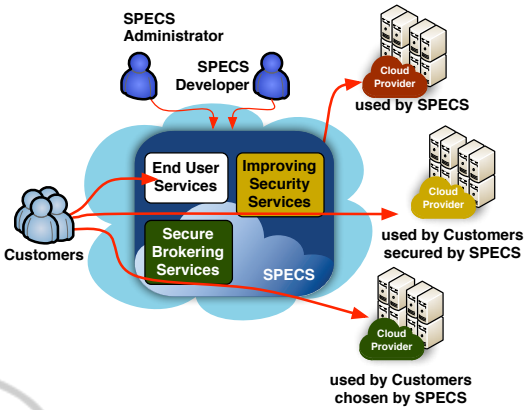

Figure 1: Interaction Model 1: Third Party Security Platform.

the service, respectively, partners are those providing support to the provisioning and/or the consumption of services. To avoid ambiguity, in our discussion we will refer to the end-users as the parties located (as customers) at the beginning of the service supply chain, while the external providers will represent those CSPs located at the end of the supply chain. We do not care about the way these CSPs offer their services, which are in the following considered atomic.

Even if this approach is very general, and covers all the different aspects of cloud service *supply chains*, it has the side effect that in different scenarios the same actor can play different roles, based on the considered perspective, generating confusion in the requirement analysis of a single cloud product that can be deployed in different ways. In particular, the platform can be offered in three different ways as shown in the following interaction models.

The main interaction model, named **Third Party Security Platform** (IM1), is proposed in Figure 1. The platform runs as an independent third-party component, consuming its own resources – acquired from a public or private cloud provider and managed by the administrator – and offering its services to end users. In such model, the platform needs to authenticate its own users and to manage their credentials in a secure way. As for responsibility taking, it depends on the role played by the platform. Indeed, the platform can play both the ETSI roles of CSP and partner: as a CSP, it signs a contract with the end user and takes care of the entire remote service invocation process on his behalf. When it acts as a partner, instead, the user only exploits its brokering capabilities to find the service that bests suits his needs, but then signs a contract with the selected remote service provider to have his requirements granted.

The second interaction model is named **Hosted**

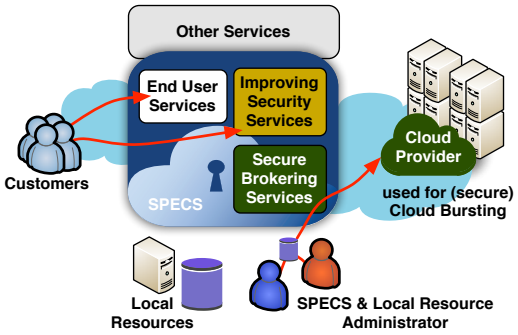

Figure 2: Interaction Model 2: Hosted Platform.

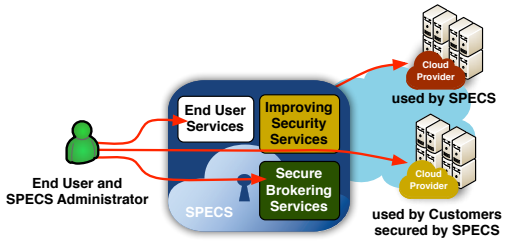

Figure 3: Interaction Model 3: User Software.

Platform (IM2) and is shown in Figure 2. In such interaction scheme, the administrators are co-located within the Hosting CSP, which internally hosts the platform. A hosting CSP can exploit the platform in order to (i) enhance security features of the services and resources being normally offered to end users, (ii) acquire external CSP services to fulfill users' requests (in this case, secure brokering services can be used also to perform cloud bursting – note that such services are invisible to end users), and offer additional functionalities to users by re-selling the end-uses services implemented by the platform. Differently from the first interaction model, in this case the platform acts only as a CSP, and never as a partner, because it directly provides the hosting CSP with the requested services taking on all responsibilities for them.

The third interaction model is referred to as **User Software** (IM3) and is shown in Figure 3. It assumes that the platform is dedicated to a single end user, who installs and runs it to manage his own activities. In this case, the platform does not offer services to other end users, but it is just adopted by its administrator (the end user who owns it) in order to manage security features over remote services. For this reason, internal services are almost useless (no need for authentication, or for offering something internally dedicated), while brokering and security enhancement are the only exploited functionalities. In this interaction model, the platform acts only as a partner, because the end user adopts its features only to build his own services.

4 PRELIMINARY REQUIREMENT ANALYSIS

In this Section we show the whole process followed to analyze some the main functionalities offered by the platform to derive a preliminary set of requirements. In particular, for the sake of brevity, we focus on one of the macro-functionalities identified in Section 3, that is *Brokering of an IaaS (Service) with granted security parameters*.

Table 1: Brokering usage perspectives.

Interaction Models	Brokering
IM1	(as Partner)IM1-P / (as CSP)IM1-CSP
IM2	(as CSP) IM2-CSP
IM3	(as Partner)IM3-P

With respect to this macro-functionality, we analyzed the behavior of the platform in the three interaction models proposed in the project. In particular, for each interaction model, we considered the possible roles played by the platform (CSP or Partner-P), deriving a set of usage perspectives, shown in Table 1. As illustrated in Section 3.1, not all the combinations are allowed: for example, in the interaction model 2, the platform acts only as a CSP and never as a Partner, because it directly provides the Hosting CSP with the requested services taking on all responsibilities for them. In the following, we are going to detail a couple of usage perspectives, and report high level interaction diagrams from which to derive a preliminary set of use cases.

In Figure 4, we report the high level interaction diagram for the usage perspective named *IM1-CSP*: the aim is to identify the basic steps related to the execu-

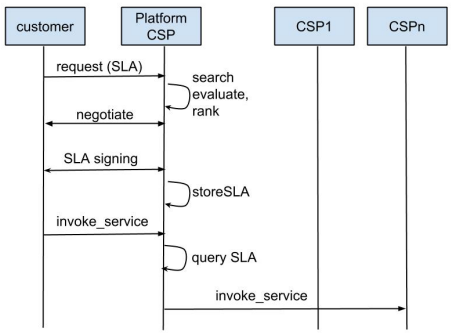

Figure 4: The platform acts as a broker in IM1.

tion of the brokering macro-functionality in the case when the platform takes on all responsibilities on the *quality* (i.e. level of assured security) of the invoked services. As depicted, the involved actors are the Customer, the Platform-CSP and one or more External CSPs. We assume that the Platform-CSP manages a repository of existing CSPs and related SLAs and a customer submits to the Platform-CSP a SLA request with a list of security requirements. The basic flow of events can be described as follows:

1. a customer submits to the Platform CSP his security requirements in terms of a SLA,
2. the Platform CSP compares the SLA's security requirements against those specified in its CSP SLA repository and possibly applies a ranking algorithm,
3. the Platform CSP negotiates with the customer the target CSP to invoke based on its SLA,
4. the Platform CSP and the customer sign the SLA, and the Platform CSP stores the signed SLA in a signed SLA repository,
5. the customer invokes one of the negotiated services on the Platform CSP, which forwards the invocation request to the proper access point and retrieves possible results, to return them back to the customer.

With respect to the other interaction models, the brokering function is basically carried out in a similar way, except that no SLA is signed between the customer and the platform whenever the latter acts as a Partner.

From the analysis of this scenario, we were able to derive a first set of functionalities, which can be logically split into three service layers: the *application service layer* includes all services related to the implementation of the business logic of the application built on top of the platform (in this case, the application that offers a broker of secure services). The *core service layer* includes all services related to negotiation of SLAs and search, evaluation and ranking of providers based on their SLAs. Finally, the *Platform service layer* includes all basic enabling services (e.g.set-up, activation, deployment of services on the cloud infrastructure) and services related to SLA lifecycle management.

5 PRELIMINARY ARCHITECTURE DESIGN

A preliminary architecture design is shown in Figure 5: as illustrated in the previous Section, the services offered through the platform can be classified into three service layers: the Application layer, the Core Services layer and the Platform layer. The applications built on top of the platform belong to the application layer and are implemented by using the services offered by both Core and Platform layers. The main concept behind the platform is that every functionality is offered as a service and all applications are managed through SLAs. For this reason, the interactions among an application hosted by the platform and its customers are subject to negotiation and controlled via enforcement and monitoring functionalities. According to this view, the Core Services layer is composed by the following modules:

- Negotiation Module: manages all the services dedicate to guide customers and providers to reach the agreement on SLAs.
- Enforcement Module: manages all the services dedicated to apply security mechanisms and correctly configure and use services, on the basis of the agreed SLAs.
- Monitoring Module: manages all the services dedicated to collect and evaluate the state of the services acquired.

At the Platform layer, as discussed in the previous Section, we locate both low-level (enabling) services, to support the correct execution of higher-level services in the cloud environment and offer basic functionalities (repository management, SLA management), and services for interoperability, which implement the actual communication among services located at the core layer. Indeed, architecture modules must be usable independently one of the others, provided interfaces among modules will be as simple as possible and all modules will support the SLA-based philosophy. This implies that the communication among services of the Core layer will be possible only through the Platform layer services, which will also support the interoperability among modules. From the architecture point of view, the Platform layer will be then composed of two sub-layers:

- Enabling Platform: enables the execution of components (configuration, starting, stopping, packaging) on-top of a cloud-like (IaaS-like) infrastructure and offers other generic support services(component configuration, communication, storage).
- SLA Platform: offers services to enable the management of SLA life cycle, including the management of SLA repositories and the communication among services dedicated to different phases of SLA life cycle, acting as the SLA-based glue

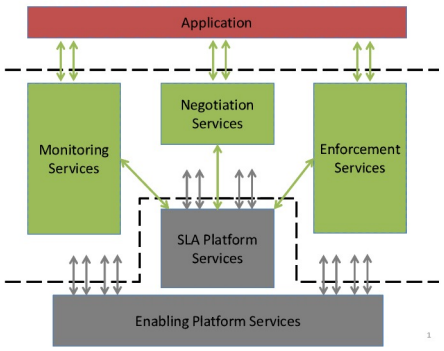

Figure 5: Architecture.

for the security services we are interested to offer over the platform.

The layers also put in evidence that SPECS is able to provide both Software-as-a-Service (SaaS) and PaaS service models: as a SaaS, it offers a deployed Platform, hosting a set of applications, which provides application-specific services to its own customers; as a PaaS, it offers a deployed Platform that enables the deployment of new applications. As an example, from one side the customers can use the platform to invoke (use) an Improved Security Service, consisting in a service either locally developed in the platform or obtained by improving an existing service offered by a remote CSP, which has specific security features. From the other side, the customers can use the platform to develop, for example, an Improved Security Service, based on basic enabling services (those offered by the Enabling Platform and the SLA Platform).

6 CONCLUSIONS

In this paper we presented the preliminary analysis of a *cloudware* dedicated to create a PaaS that offers security services through an SLA approach. We proposed a methodology to design such kind of platforms, and sketched the preliminary results of the requirement analysis, starting from some of the usage scenarios identified in the context of the SPECS FP7 project. We argue that the proposed methodology is of general interest for the design of *cloudwares* and dedicated PaaS security solutions. The requirement analysis presented here can be reused independently of the technology which will can be involved, offering a high-level view over the provision of security as-a-service using SLAs. Moreover, we proposed a preliminary design of the Platform, based on the considerations and the classification of services made in our preliminary analysis. We plan to upgrade this requirement analysis in the future, collecting feedbacks from cloud stakeholders to refine the requirements and the preliminary design, and to start the prototyping phase of the platform.

REFERENCES

Battista, E., Casola, V., Mazzocca, N., Ficco, M., and Rak, M. (2013). Developing secure cloud applications:a case study.

Catteddu, D. (2011). Security & resilience in governmental clouds. Technical report.

Commission, E. European cloud partnership. http://ec.europa.eu/digital-agenda/en/european-cloud-partnership.

Commission, E. (2011). Unleashing the potential of cloud computing in europe. Technical report.

CSCC. The cscc practical guide to cloud service level agreements. Technical report.

Dekker, M. (2012). Critical cloud computing a ciip perspective on cloud computing services. Technical report.

ETSI (2013). Cloud standards coordination. Technical report.

OPTIMIS. The optimis project web site. http://www.optimis-project.eu/.

Pannetrat, A., Hogben, G., Katopodis, S., Spanoudakis, G., and Cazorla, C. (2013). D2.1: Security-aware sla specification language and cloud security dependency model. technical report, certification infrastructure for multi-layer cloud services (cumulus).

Pearson, S. (2011). Toward accountability in the cloud. *Internet Computing, IEEE*, 15(4):64–69.

Petcu, D., Martino, B. D., S., V., Rak, M., Mhr, T., Lopez, G. E., Brito, F., Cossu, R., Stopar, M., Sperka, S., and Stankovski, V. (2013). Experiences in building a mosaic of clouds. *JOURNAL OF CLOUD COMPUTING*, 2:–.

Pierre, G. and Stratan, C. (2012). Conpaas: a platform for hosting elastic cloud applications. *IEEE Internet Computing*, 16(5):88–92.

Rak, M., Suri, N., Luna, J., Petcu, D., Casola, V., and Villano, U. (2013). Security as a service using an sla-based approach via specs. In IEEE, editor, *Proceedings of IEEE CloudCom Conference 2013*.

RedHat. Openshift web site. https://www.openshift.com/.

Zeginis, D., D'Andria, F., Bocconi, S., Cruz, J. G., Martin, O. C., Gouvas, P., Ledakis, G., and Tarabanis, K. A. (2013). A user-centric multi-paas application management solution for hybrid multi-cloud scenarios. *Scalable Computing: Practice and Experience*, 14(1).