Trust Validation of Cloud IaaS: A Customer Centric Approach

Salman Manzoor, Ahmed Taha and Neeraj Suri
Dept of CS, TU Darmstadt, Germany
Email:{salman, ataha, suri}@deeds.informatik.tu-darmstadt.de

Abstract—**A multitude of issues affect the broader adoption of Cloud computing, with the perceived lack of trust on the Cloud Service Providers (CSPs) often listed as a significant concern. To address this, CSPs typically set up Service Level Agreements (SLAs) that contractually list what the CSP is obligated to provide to meet the customer requirements. While SLAs are promising as a concept, the inadequacy of schemes to actually monitor and validate the run-time compliance of SLAs limits the customer's capability to evaluate the offered services to assess the actual trust to put in the CSPs.**

In this paper, we propose a methodology to validate SLAs and detect service violations over the life of the service. The SLA consists of a list of service attributes that are required by the customer and committed by the CSP. In order to validate an SLA, we evaluate each SLA attribute either qualitatively or quantitatively as relevant. The evaluation provides reproducible assurance to a customer for trusting the CSP based on its fulfillment of the customer's requirements. We classify requirement violations into "trust states" according to the customer defined preferences and the CSP can be in varied trust states depending on the severity level of the violations. We demonstrate assessing the trust state of a CSP based on the services involved in launching and migrating a virtual machine in Infrastructure-as-a-Service (IaaS) offering.

I. Introduction

Cloud computing delivers on-demand, scalable and shared resources to customers in the form of three broad types of services offered by Cloud Service Providers (CSPs) [1]. The first being SaaS (Software as a Service), that offers software applications as a service and allows a Cloud customer to control only the application configurations. The second, PaaS (Platform as a Service), provides the customers a platform for developing applications, and the customer controls the hosting environment. Finally, IaaS (Infrastructure as a Service) provides computing and storage resources where the customer controls everything except the data center infrastructure. With IaaS as a highly flexible service, we focus on IaaS in this paper.

While these service types are attractive, the non-transparent architecture of Cloud, the paucity of mechanisms to provide definitive assurance about the fulfillment of user requirement by Cloud Service Providers (CSPs), and the unclear assurance on security facets on service delivery impede many businesses from deploying Cloud services. These often result in Cloud Service Customers (CSCs) being unable to trust the CSPs. Broadly, trust implies reliance in something that is expected to behave or deliver as promised [2]. In the context of the Cloud, we refer to trust as the degree of reliance on the services offered by the CSP with respect to the customer's requirements.

Members of the Cloud community[1] advocate specifying service provisions in Service Level Agreements (SLAs) to establish common semantics to provide and manage assurance. Fundamentally, an SLA represents a formal contract between a customer and a CSP which specifies service provisions with respect to the customer's requirements. The SLA includes a list of attributes which are the measurable elements that specify service levels provided by the CSP in comparison to the customer's requirements along with the agreed upon quality level for each attribute (e.g., latency, throughput etc) along with penalties for non-delivery of services. Thus, a logical way to assess trust of the CSP is to validate the SLA.

Although the state of the art predominantly focuses on the methodologies to negotiate and design Cloud SLAs [3]–[6], most of these methodologies assume that the CSPs are actually providing the services as agreed in the SLAs. The techniques to detect SLA violations are conspicuous by their paucity. A violation happens if an agreed SLA is not fulfilled by the CSP. In other words, the SLA is violated if the CSP is not provisioning the services according to the customer's requirements. Therefore, it is important to provide customers with a comprehensive support in order to (i) validate the SLA through the course of time and operations at the CSP and (ii) enable an automatic detection of SLA violations.

A. Contributions

This paper aims to solve the aforementioned issues by proposing a novel reasoning approach to:

1) Assess the qualitative and quantitative attributes 'to-be-provided' by the CSP and 'required' by the customer. We use set theory for the qualitative attributes, and propose pairwise comparators for quantitative attributes. This comparison forms the basis to validate the SLA and detect violations on the customer's requirements.
2) Classify violations into "trust states" according to the preferences specified by the customer.
3) Validate the trust of Cloud IaaS by applying our methodology to detect SLA violations during the launch and migration of a virtual machine.

[1]For example, standard bodies such as NIST, the European Network and Information Security Agency (ENISA), Cloud Security Alliance (CSA), ISO/IEC, and the European Commission.

The paper is organized as follows. Section II introduces the basic concepts on SLAs. Section III reviews contemporary SLA validation approaches. Subsequently, Section IV describes our proposed methodology, and in Section V we present case studies to evaluate the services involved in a Cloud IaaS. Section VI summaries the methodology.

II. Basic Concepts

A Cloud SLA describes the provided services, and represents a binding commitment between a CSP and a customer. The SLAs outline the desired services, each of which contains a list of attributes. Each attribute is composed of one or more metrics, as relevant, that help in the measurement of the Cloud services by defining parameters and measurement rules. Hence, the SLA contains a list of attributes, with corresponding desired values, that the CSP is committed to fulfill. If any of these committed values is not fulfilled by the CSP, then the SLA is violated. In practice, one way to assess trust of the CSP is through periodically validating the SLA. Thus SLA monitoring schemes are used to quantitatively validate what a CSP is providing and which assurances are actually met.

Based on the analysis of the state of practice presented in [7], Cloud SLAs are typically modeled using a hierarchical structure as shown in Figure 1. The root of the structure defines the main container for the SLA. The intermediate levels (second and third levels in Figure 1) are the services and the lowest level represents the actual attributes committed by the CSP and consequently offered to the customer. These attributes form the threshold values which are specified in terms of metrics in relation to the customer's requirements.

The concept of a SLA is formalized using the following definition.

Definition. *An SLA consists of a set of services* $S = s_1, \ldots, s_n$*. Each service s consists of a finite positive number n of attributes* k_i*; where* $i = 1 \ldots n$*. Each attribute* k_i *consists of m different values* v_i*; such that* $k_i = v_{i,1}, v_{i,2}, \ldots, v_{i,m}$*. Each value implies a different service level offered by the CSP and required by the customer. Each* k_i *value is mapped to a numerical value according to its type/range.*

Note that, the attributes can have varied types/ranges of qualitative and quantitative values. Hence, a process for comparing different attributes across the customer's requirements is needed in order to detect SLA violations by the CSP. We present this comparison methodology in Section IV.

III. Related Work

With the rapid growth of Cloud services, multiple approaches have emerged to assess the functionality and security of CSPs. In [8], the authors proposed a framework to compare different Cloud providers across performance indicators. In [5], an Analytic Hierarchy Process (AHP) based ranking technique was proposed that utilizes performance data to measure various Quality of Service (QoS) attributes and comparatively ranks the CSPs. In [9], a framework that enables a comparison

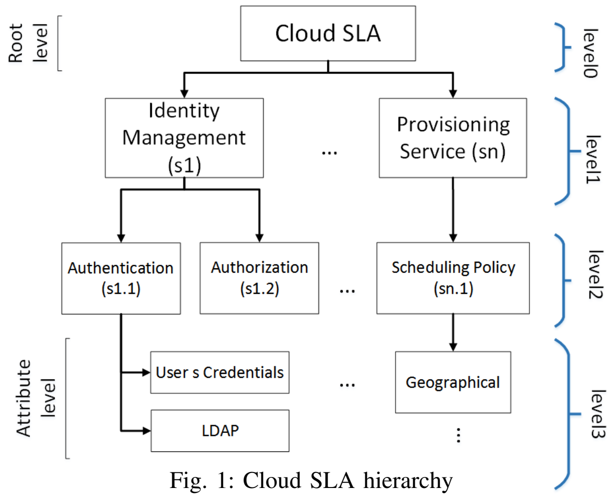

Fig. 1: Cloud SLA hierarchy

of Cloud services based on critical characteristics is presented. However, these studies (i) focused on assessing performance of Cloud services but not their security properties and (ii) did not consider SLA validation for identifying violation-prone service providers.

Security requirements for non-Cloud scenarios have been addressed by Chaves et al. [10] who explored security in SLAs by proposing a monitoring and controlling architecture for web services. In [11] and [12], the authors proposed a technique to aggregate security metrics from web services. However, the authors did not propose any techniques to assess Cloud SLAs or empirically validate the proposed metrics. Luna et al. [3] presented a methodology to quantitatively benchmark Cloud security with respect to the customer's defined requirements (based on control frameworks). In [4], the authors presented a framework to compare, benchmark and rank security levels provided by two or more CSPs. However in both of them, the SLA validation was not covered.

An SLA validation framework can help in identifying violation-prone service providers. However, in order to serve the customer best, a trust model should take into account all the available sources of information including the customer's requirements and feedback. In [13], the authors proposed a multifaceted Trust Management (TM) to identify trustworthy Cloud providers in terms of different attributes. However, their assessment considered trust as a security service level and furthermore they did not manage and maintain dynamically changing trust values. In both [14], [15], the authors considered the SLA validation as the main factor for establishing trust on grid and web service providers. However, they only considered QoS attributes.

The methodology presented in this paper differs from the above mentioned works in respect to the attributes under consideration and maintaining the dynamic state of trust. Trust models for Cloud computing need to take Cloud specific attributes into account and these go beyond the usual QoS parameters. In this paper we consider Cloud specific attributes by evaluating services in launching and migrating a VM. We validate attributes to detect violations periodically and the effect of these violations on the state of trust. We utilize a state transition approach to model different trust states, and demonstrate how violations dictate the transition across them.

IV. Proposed Methodology

In this section, we describe our research methodology to validate an SLA by comparing service provisions of a CSP with the customer requirements over the lifetime of the service. The stages involved in the methodology are shown in Figure 2. In Stage A, the customer specifies his/her requirements and the CSPs specify their service provisions. The customer then selects a CSP that "best" matches his/her requirements. In Stage B, we monitor the selected CSP to acquire the attribute values of the desired services. Stage C validates these attributes with the customer's requirements and assesses the current state of trust. As mentioned in Section I, we refer to trust as the degree of reliance on the offered services with respect to the customer's requirements. Therefore, if a CSP is provisioning the services according to the requirements, the CSP is consequently fulfilling the SLA to be deemed to be in a trusted state. However, if the requirements are violated, consequently, the state of trust is changed to reflect the degree of violation. We apply our methodology periodically[2] to assess the current state of trust which is modeled in Stage D using a state diagram.

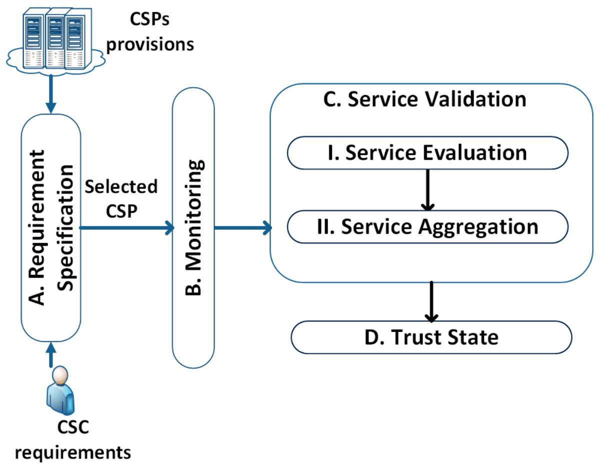

Fig. 2: Stages of the proposed methodology

Stage A. Requirements Specification

In this stage, the customer specifies his/her requirements and the CSPs specify their provisions. The customer evaluates and ranks each CSP according to the requirements and selects a CSP that best matches these requirements. The ranking algorithm was proposed in our previous paper [4]. The target of this paper is to periodically validate the selected CSP provisions. For clarity, we describe the customer's requirements specification that is used to validate the CSP provisions.

[2]A periodic interval can be chosen or an event based schema using a violation threshold as a trigger can be used.

We use the same SLA hierarchical structure (cf., Figure 1) to model requirements. The customer can specify his/her requirements at different levels of granularity and can specify priorities of the requirements by assigning weights to them. Furthermore the customer can specify weights by using linguistic terms (Highly-Critical (HC), Critical (C), Less-Critical (LC) and Not-Critical (NC)) for the attributes and/or services. The highly-critical attributes have high importance for the customer, while not-critical attributes have least importance. Thus, violating higly-critical attributes have severe implications on trust than the rest of the attributes. Critical and less-critical specifies the customer's different degrees of importance regarding these requirements, where he/she can accept varied values depending on the considered scale.

Stage B. Monitoring the Selected CSP

This stage involves monitoring of the selected CSP to capture the values of the attributes. Monitoring plays a significant role in identifying violations as it provides the attribute values from the CSP for comparison. There are various monitoring schemes proposed by industry and academia, e.g., Cloud Watch [16] which is used to monitor Amazon EC2, Cloud Stack [17] as an open source framework for monitoring the Cloud, and Ayad and Dipel [18] proposed an agent based monitoring for virtual machines in an IaaS environment. For this paper, we are primarily interested in the values of the attributes from the CSP to compare with the requirements of the customer. The details of these monitoring schemes are beyond the scope of this paper. We refer the interested reader to [19], a survey on monitoring schemes for the Cloud.

Stage C. Service Validation

The goal of this stage is to validate the service provisions of the CSP in comparison to the requirements of the customer. As mentioned in Section II, each service consists of a set of attributes that are necessary to provide the desired functionality. Therefore, services are validated using the attribute values provided by the CSP and required by the customer.

Service validation forms the basis to detect a requirement violation. If a CSP is fulfilling all the customer's requirements, consequently the CSP is in the trusted state. However, in presence of a requirement violation, we assess the Impact Factor (IF) of the violation, which determines the severity of the violation by measuring the distance between the provided and the required values. In Table I, we define different levels of IF and relate severity of the violation respectively. The levels are normalized between 0 and 1 based on the degree of deviation from the required value i.e., greater the deviation, more severe is the violation. Ideally, IF should be 0, which indicates no violation from the CSP. The second level ($0.1 < IF \leq 0.25$) indicates violations that have minimum severity as IF deviated minimally. The third level ($0.25 < IF \leq 0.5$) indicates medium severity of the violations while the last level ($0.5 < IF \leq 1$) specifies that the distance between the service provision of the CSP and the customer's requirement is farthest and hence, indicates the maximum severity of the violation.

TABLE I: The relation of Impact Factor to Severity of violation(s).

Impact Factor	Severity of violation(s)
$IF = 0$	No Violation
$0.1 < IF \leq 0.25$	Minimum Severity
$0.25 < IF \leq 0.5$	Medium Severity
$0.5 < IF \leq 1$	Maximum Severity

The subsequent Phases I and II evaluate each discrete service to detect requirement violations and to assess the impact of these violations according to the levels described in Table I.

Phase I. *Service Evaluation:* The services can have multiple attributes that can be either quantitative or qualitative in nature. The attributes such as *CPU*, *RAM* and *disk space* are quantitative attributes while *scheduling policy* and *authentication methods* are examples of qualitative attributes. This complicates the process of modeling and comparing values to evaluate a quantitative metric. To address this complexity, we first classify different types of attributes and provide a validation method for each type. The attributes can be classified as either numerical or unordered sets. This classification sets the basis for validating attributes, as the validation method for numerical values differs from the validation of the unordered set.

Numerical: The attributes such as are *CPU*, *network latency* and *bandwidth* are classified as numerical since their values can monotonically increase or decrease. We validate numerical attributes by comparing values provided by the CSP with the values required by the Cloud Service Customer (CSC). The relationship between the *CSP* and the *CSC* with respect to attribute *k* and value *V* is represented using a pairwise comparison such that:

$$CSP_k/CSC_k = \frac{V_1}{V_2} \tag{1}$$

i.e., assume a *CSP* and a *CSC*, with values V_1 and V_2 for *network latency* attribute respectively, such that: the CSC's required value for network latency is $100ms$ (i.e., $V_2 = 100ms$) and assume $100ms$ is provided by the CSP (i.e., $V_1 = 100ms$). The pairwise comparison relation between V_1, V_2 is defined as: $\frac{V_1}{V_2} = 1$. Therefore, *CSP* is fulfilling the requirement.

If the result of the pairwise comparison is not equal to 1, this indicates a violation. This violation could be due to over-provisioning when the result is greater than 1, or under-provisioning when the result is less than 1. We consider over-provisioning as a violation, since a malicious administrator or an inside attacker could over-provision the attribute and the customer would have to pay for this additional provisioning.

In case of a violation, we calculate the impact factor of the violation as:

$$IF_k = |1 - \frac{CSP_k}{CSC_k}| \tag{2}$$

Equation 2 calculates the impact factor of the violation as an absolute value. We use the levels as mentioned in Table I to indicate the severity level of the violation. An impact factor IF_k of 1 indicates that the CSP has violated the attribute with maximum severity while IF_k of 0 indicates no violation.

Unordered Set: We define an unordered set for the attributes that are qualitative in nature. These attributes include *access policy* and *authentication methods* and validating these attributes comparatively is not possible. For qualitative attributes set theory is utilized to detect violations and estimate the impact factor of violations. The advantages of set theory are twofold, firstly, its ability to generalize logic behavior, i.e., the same operations work for *access policy* and *scheduling techniques* although they belong to different services and have a different set of values. Secondly, using sets in our case empowers us to evaluate the impact factor of the violations by calculating dissimilarity between sets. This dissimilarity could be a result of adding or removing a value in the set. Assuming that for a *CSC* the required set for *access policy* is $\{read, write\}$. A violation is detected whenever a *CSP* adds/deletes any value to/from the access policy and as a result its impact factor should be assessed.

The violations are identified by calculating symmetric set difference between the *CSP* provided set and the *CSC* requested set. If the symmetric difference results in a *null* set, this implies that the *CSP* provisions and the *CSC* requests are the same and hence no violation. However if the result is not a *null* set, then a violation has occurred. Lets suppose the following *sets* list an attribute *k* values of a *CSP* and a *CSC*.

$$CSP_k = \{v1, v3, v5\}$$
$$CSC_k = \{v1, v3\}$$

To find out the violation(s), we calculate the symmetric difference between the values provided by the *CSP* and those requested by the *CSC*, such that:

$$CSP_k - CSC_k \neq CSC_k - CSP_k \neq \{\emptyset\} \tag{3}$$

If the result of the symmetric difference is not a *null* set, then the *CSP* is violating the customer's requirement(s). We use $CSP_k - CSC_k = \{v5\} \neq \{\emptyset\}$ to detect a violation due to the addition of a value and $CSC_k - CSP_k \neq \{\emptyset\}$ to detect the removal of the attribute value.

After identifying the violations, we calculate the impact factor of these violations similar to the numerical type. For sets, we use the Jaccard Index [20] which calculates dissimilarity between the sets by estimating the distance between the provided and the required values. This distance measurement is equivalent to our definition of the impact factor and calculated as:

$$IF_k = 1 - \frac{|CSP_k \cap CSC_k|}{|CSP_k \cup CSC_k|} \tag{4}$$

Equation 4 evaluates the impact factor of the violation. We use the same levels as mentioned in Table I, i.e., $IF = 0$ indicates no violation while $IF = 1$ indicates the maximum severity of the violation.

Using the above comparison metrics for each attribute, we obtain the impact factor(s) of the violating attribute(s).

This results in a matrix of size N if there are N attributes in a service. In order to evaluate the service assurance, we aggregate the impact factors of all attributes belonging to a service.

Phase II. *Service Aggregation*

After validating the lowest (attribute) level of the SLA, we move up in the hierarchy (cf., Figure 1) and assess the aggregated assurance of the service provided by the CSP. In Phase I, we evaluate the impact factors of each attribute and these are further used as input in this phase for an aggregation method. Equation 5 is used to aggregate the impact factors of attributes IF_i along with their weights to evaluate the service impact factor.

$$IF_{service} = \sum_{i=1}^{n} \left(\frac{IF_i * weight_i}{n} \right) \quad (5)$$

By using this equation, $IF_{service}$ results in a value between 0 and 1 and uses the same levels as described in Table I. Thus, $IF_{service}$ of value 0 indicates that the service is provisioned as required by the customer while value 1 indicates that every requirement was violated with maximum severity level.

Stage D. Trust State

After assessing services individually, the next step is to aggregate impact factors of the services ($IF_{service}$) to calculate the impact factor at root level IF_{root}. Services are aggregated according to Equation 5 and IF_{root} uses the same levels as described in Table I. We use IF_{root} to assess the state of trust and implications of the violations using a state diagram which is shown in Figure 3. Each state represents the severity level of the violations, i.e., state 1 represents no violation while states 2, 3 and 4 represents minimum, medium and maximum severity of the violations respectively.

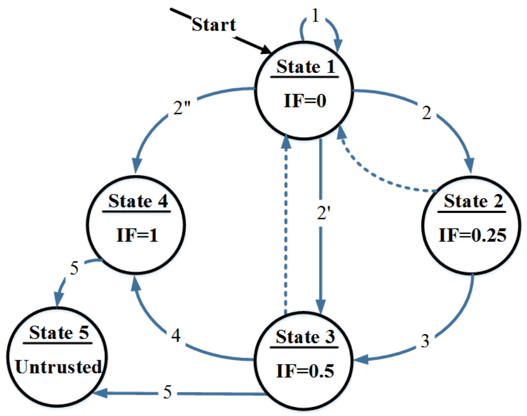

Fig. 3: Effect of impact factor on states of trust

- State 1: We evaluate IF at the root level and if $IF = 0$, this implies that the CSP has not violated any requirement and consequently, it is in the trusted state. This is shown as transition 1 in the state diagram. However, if violations are detected, then the state of the CSP is changed to either state 2, state 3 or state 4 (as indicated by the respective transitions 2, 2' and 2") corresponding to the severity of the violations.
- State 2: The transition 2 in the state diagram illustrates that the state of the CSP is changed to state 2 as the violations result in an IF value of between 0.1 and 0.25. We evaluate IF again in state 2 to deduce the next state of the CSP. If the CSP has not violated any requirement ($IF = 0$) then the state is changed back to state 1. However if there are again violations with minimum severity ($0.1 < IF \leq 0.25$) then the state is changed to state 3 to indicate the aggregated impact of violations.
- State 3: The CSP state is changed to this state from state 1 if the detected violations in state 1, resulted in the medium severity. Thus, the CSP is moved to state 3 to indicate this behavior. We use a counter in states 3 and 4 to ascertain how many times these states have been transited. The threshold is a value specified by the customer to signify how many times the customer can endure violations. This counter plays an integral role in deciding the next state of the CSP. From state 3, the CSP can recover to state 1 if no further violations are detected, and the count is below a specified threshold.
- State 4: This state indicates the maximum impact of violations and the state of the CSP is changed to this state if the (aggregated) impact factor of violations results in a value of between 0.5 and 1. From this state CSP cannot recover to state 1.
- State 5: This is the untrusted state and the state of the CSP is permanently changed to this state if the CSP violates requirements more than the threshold specified by the customer.

The state diagram is useful in determining the current state of trust of the CSP based on the customer's requirement violations. In the next section we apply our methodology to detect violations and assess trust of the CSP during launching and migrating a VM.

V. Case Study: Trust Assessment of CSP during Launching and Migrating a Virtual Machine

This initial validation scenario demonstrates how a Cloud customer can apply the methodology presented in this paper to assess the state of trust of the CSP during the course of operations. We evaluate trust of the CSP by considering two scenarios: (1) launching a VM and (2) migrating a VM. To start a VM, the customer requests the CSP to boot an instance of a VM according to his/her requirements. We evaluate IF_{root} to determine the state of trust during the course of launching a VM. The migration scenario considers moving a VM from one physical host to another in compliance to the customer's requirements. We evaluate IF_{root} again to ascertain violation(s) during the migration phase and the impact of these violations on the state of trust.

The services involved in launching and migrating a VM are shown in the respective sections of the case studies while

Table II presents a sample dataset used for the scenarios, where the values associated for 9 attribute are presented. In order to perform a comprehensive validation, the selected attributes include both qualitative and quantitative attributes. Furthermore, weights assigned by the customer to indicate his/her priorities are specified as numerical value such that Highly-Critical *(HC)* indicates a relative value of *1*. Critical *(C)* and Not-Critical *(NC)* can be considered any intermediate values between *1* and *0*. In this analysis they indicate a relative values of *0.7* and *0.3* respectively.

In the rest of the section, we outline the computation process to assess the state of trust of the CSP with respect to the requirements defined in Table II.

A. Case I: Launching a VM

We now detail our methodology to detect user requirement violation in the services involved in launching a VM. For brevity, the process of launching a VM and interactions among the services is shown in Figure 4. This is requisite to understand CSP operations and, hence, the services validation process.

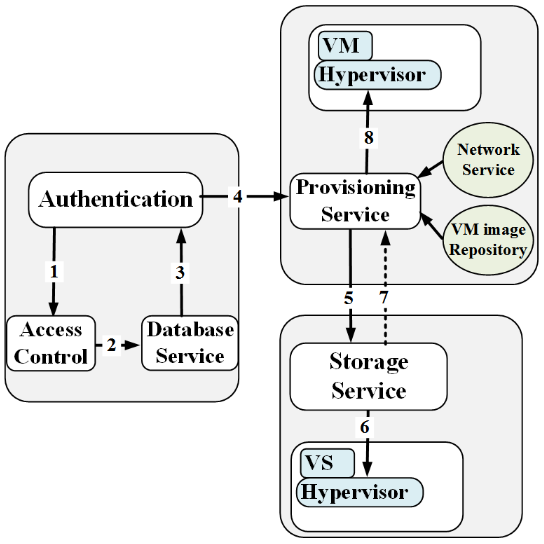

Fig. 4: Services and their communication in launching a VM

- Transitions 1, 2, and 3: A customer is authenticated and, based on the role granted by access control service, gets access to his/her list of VMs in the database.
- Transition 4: The customer requests launching a VM through the provisioning service.
- Transition 5: The provisioning service communicates with the storage service for allocating storage for the VM.
- Transitions 6 and 7: The storage service finds the host server to instantiate a virtual storage, and reports back to the provisioning service with the storage volume. Transitions 6-7 are optional in case the customer does not select the storage facility for the VM. At this point, the image repository service provides the operating system image for the VM, and the network service provides the networking facilities (e.g., virtual NIC, IP addresses).
- Transition 8: The provisioning service requests the hypervisor to instantiate the VM. The hypervisor reports back to the provisioning service after instantiating the VM, and is added to the database by the provisioning service.

With respect to these services, we use the data shown in Table II for assessing state of the trust of the CSP. We start our evaluation from the attribute level. We assess $IM1.1$ to check if there is any violation in *authentication* during the course of launching a VM at the CSP. Equation 3 is used to detect violation by calculating the symmetric difference between the sets provided by the CSP and requested by the CSC, such that:

$$CSP_{IM1.1} - CSC_{IM1.1} = CSC_{IM1.1} - CSP_{IM1.1} = \{\emptyset\}$$

The result of the symmetric difference is the empty set which indicates that the CSP is fulfilling the customer's requirement and hence the impact factor is 0. As $IM1$ service consists of only one attribute and, therefore, the service impact factor is the same as the attribute, i.e., $IF_{IM1} = IF_{IM1.1} = 0$. Next we evaluate *authorization* by validating $IM2.1$ and $IM2.2$ using symmetric difference calculation.

$$CSP_{IM2.1} - CSC_{IM2.1} \neq CSC_{IM2.1} - CSP_{IM2.1} \neq \{\emptyset\}$$
$$CSP_{IM2.1} - CSC_{IM2.1} = \{delete\}$$

Since the symmetric difference is not the empty set, this indicates a violation and, therefore, we assess its impact factor using the Jaccard Index (cf., Equation 4) such that:

$$IF_{IM2.1} \quad = 1 - J(CSP_{IM2.1}, CSC_{IM2.1})$$

Where,

$$J(CSP_{IM2.1}, CSC_{IM2.1}) = \frac{|\ CSP_{IM2.1} \cap CSC_{IM2.1}\ |}{|\ CSP_{IM2.1} \cup CSC_{IM2.1}\ |} = \frac{2}{3}$$

Thus $IF_{IM2.1} = 1 - \frac{2}{3} = \frac{1}{3}$ In a similar way, $IM2.2$ impact factor is calculated. Using Equation 3, we premeditate the symmetric difference for $IM2.2$ which detects no violation.

$$CSP_{IM2.2} - CSC_{IM2.2} = CSC_{IM2.2} - CSP_{IM2.2} = \{\emptyset\}$$

The impact factor of $IM2.1$ and $IM2.2$ is 0.66 and 0 respectively. We aggregate these impact factors to assess service provision using Equation 5.

$$IF_{IM2} = \frac{IF_{IM2.1} * weight_{IM2.1} + IF_{IM2.2} * weight_{IM2.1}}{2}$$
$$IF_{IM2} = \frac{0.66 * 0.7 + 0 * 0.7}{2} = 0.23$$

After each service calculation, we move up in the hierarchy to calculate the impact factor of the domain that contains these

TABLE II: Excerpt of SLA's from CSPs and customer's requirements.

Cloud secSLA					Customer (CSC)		CSP	
Services			Attributes	Values	*req*	*weight*	Case I	Case II
Root	Identity Management *IM*	Authentication *IM*1	Auth *IM*1.1	Unordered set	User's credentials	HC	User's credentials	User's credentials
		Authorization *IM*2	Policy *IM*2.1	Unordered set	launch, restart	C	launch, restart, delete	launch, restart
			Roles *IM*2.2		user		user	user
	IaaS *IS*	Provisioning *IS*1	CPU *IS*1.1	Numeric	6.4 GHZ	LC	2.4 GHZ	6.4 GHZ
			RAM *IS*1.2		8 GB		8 GB	8 GB
			DISK *IS*1.3		1 TB		1 TB	1 TB
			Scheduling technique *IS*1.4	Unordered set	Location based	HC	Location based	Random
	Storage *SO*	Storage *SO*1	Type *SO*1.1	Numeric	Persistent	C	Persistent	Persistent
			Location *SO*1.2		Locally attached		Locally attached	Locally attached
	Network *NW*	Network *NW*1	BW *NW*1.1	Numeric	100 Mbps	NC	10 Mbps	100 Mbps
			Latency *NW*1.2		100 ms		10 ms	100 ms

services. We aggregate IF_{IM1} and IF_{IM2} to get the IF_{IM} such that:

$$IF_{IM} = \frac{IF_{IM1} + IF_{IM2}}{2} = 0.11$$

As expected IF_{IM} is above 0, which implies that the CSP violated requirements for the identity management domain IM.

Similarly, the remaining services are evaluated using their attributes. Since $IS1.1$ metric value is represented by numeric as shown in Table II, we use Equation 2 to calculate $IS1.1$ impact factor such that:

$$IF_{IS1.1} = \sqrt{\left(1 - \frac{CSP_{IS1.1}}{CSC_{IS1.1}}\right)^2} = \sqrt{\left(1 - \frac{2.4}{6.4}\right)^2} = 0.625$$

This means that the CSP is under provisioning the customer's requirement for $IS1.1$ and thus violating the requirement. We calculate the impact factors for $IS1.2$, $IS1.3$ and $IS1.4$ in a similar way and aggregate to calculate impact factor of the service IF_{IS}.

$$IF_{IS} = IF_{IS1} = \frac{IF_{IS1.1} + IF_{IS1.2} + IF_{IS1.3} + IF_{IS1.4}}{4} = 0.07$$

Similarly, we calculate impact factors of storage SO and network NO. After evaluating each domain of services we aggregate impact factors to calculate IF_{root} as:

$$\begin{aligned} IF_{Root} &= \frac{I_{IM} + I_{IS} + I_{SO} + I_{NW}}{4} \\ &= \frac{0.11 + 0.07 + 0 + 0.45}{4} = 0.15 \end{aligned}$$

The IF_{root} indicates that CSP has violated requirements and the overall impact of these violations is minimum. Consequently, the state of trust is changed from trusted ($IF_{root} = 0$) to ($IF_{root} = 0.25$). Figure 6 shows the aggregated impact factors of attributes belonging to a single service. In the figure, the root impact factor depicts the current "Trust State" for the CSP during launching and migrating a VM.

B. Case II: VM Migration

In Case I, we assessed the state of trust of the CSP by evaluating the IF_{root}. The IF value of 0.15 indicated that the CSP violated requirement(s) and the aggregated impact of these violations was minimal and hence the state is changed to $IF_{root} = 0.25$. From this state, we again evaluate the IF_{root} to assess the CSP trust level by considering the migration process. The services involved in the VM migration are shown in Figure 5. After the user has been authenticated, the provisioning service starts a new instance of the VM and provides details of the instance to the migration service. The role of the migration service is to migrate data from the old VM to the new VM. Therefore, the migration service accesses old VM and transfers data over the network to new instance of the VM as shown in transitions 4-6 of the figure.

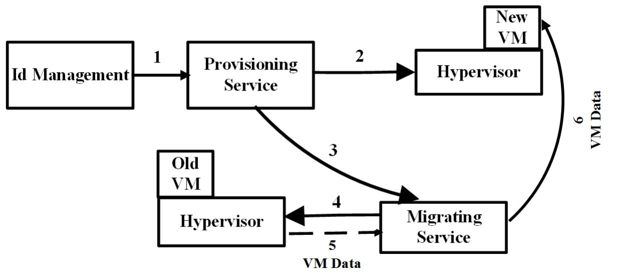

Fig. 5: Services and their communication in migrating a VM

We assess services $IM1.1$ to check if requirements are violated by the CSP during the VM migration process. We use Equation 3 to calculate the symmetric difference between $IM1.1$ values provided by the CSP and requested by the CSC so that:

$$CSP_{IM1.1} - CSC_{IM1.1} = CSC_{IM1.1} - CSP_{IM1.1} = \{\emptyset\}$$

The *null* set indicates that there are no violations. Using Equation 5, we calculate the impact factor of the service:

$$\begin{aligned} IF_{IM1.1} &= 1 - J(CSP_{IM1.1}, CSC_{IM1.1}) \\ &= 1 - \frac{\mid CSP_{IM1.1} \cap CSC_{IM1.1} \mid}{\mid CSP_{IM1.1} \cup CSC_{IM1.1} \mid} = 1 - 1 = 0 \end{aligned}$$

Similarly, both $IM2.1$ and $IM2.2$ impact factors are calculated. Using Equation 3, we premeditate the symmetric difference for both $IM2.1$ and $IM2.2$ such that:

$$IF_{IM2.1} = 1 - 1 = 0$$
$$IF_{IM2.2} = 1 - 1 = 0$$

This means that the CSP is offering $IM2.2$ and $IM2.1$ as agreed with the customer. Thus the $IM2$ impact factor is then premeditated by aggregating $IF_{IM2.1}$ and $IF_{IM2.2}$ using Equation 5 such that:

$$IF_{IM2} = \frac{IF_{IM2.1} + IF_{IM2.2}}{2} = 0$$

Subsequently, we aggregate IF_{IM1} and IF_{IM2} to get IF_{IM} such that:

$$IF_{IM} = \frac{IF_{IM1} + IF_{IM2}}{2} = 0$$

Similar to Case I, the values of IS, SO and NW are calculated. Finally, the IF_{root} impact factor is calculated such that:

$$IF_{Root} = \frac{IF_{IM} + IF_{IS} + IF_{SO} + IF_{NW}}{4} = \frac{0 + 0.25 + 0 + 0}{4} = 0.06$$

As the current trust value is evaluated to be 0.06, this indicates that the state of the trust should be changed to the trusted state ($IF = 0$) state (cf., Figure 3). Thus our methodology enables the customers to (a) assess the state of trust of the CSP during the course of operations, and (b) also provide traceable justification for trusting the CSP.

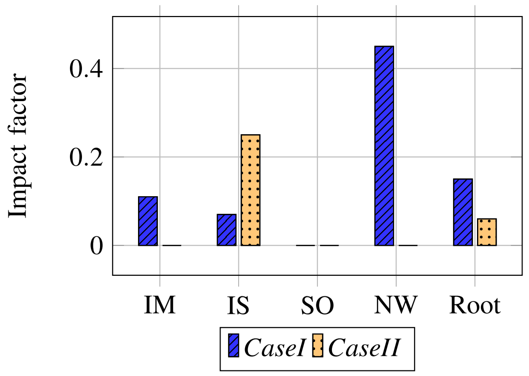

Fig. 6: Service and root Impact factors of Cloud IaaS

VI. Conclusion

Cloud service providers expect their customers to simply "trust" the CSP services offered to them but not every customer is willing to grant this trust without justification. It should be possible for a customer to establish that his/her SLA requirements are actually fulfilled by the CSP's provisioning of the specified services. With this aim, we have proposed a methodology to enable the customer to identify the CSP violation of requirements over the life of the service. For violation detection, we compare and validate each SLA attribute with respect to the customer requirements. In case of a requirement violation, we calculate the severity of the violation by calculating an impact factor. The impact factor determines the degree of deviation of the provided value from the required value, and consequently dictates the change in the level/state of trust in the CSP. The customer can periodically apply the proposed methodology to assess the behavior of the CSP over the life of the service. The application of the proposed trust assessment methodology was established via two actual use cases of CSPs offering IaaS.

Acknowledgment

Research supported by grants H2020-MSCA-ITN NECS GA # 675320 and H2020 ESCUDO-CLOUD GA # 644579

References

[1] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," *Tech. Rep. 800-145*, 2011.
[2] M. Lund, B. Solhaug, and K. Stolen, "Evolution in relation to risk and trust management," *In IEEE Computer*, vol. 43, no. 5, pp. 49–55, 2010.
[3] J. Luna, R. Langenberg, and N. Suri, "Benchmarking Cloud Security Level Agreements Using Quantitative Policy Trees," *Proc. of Cloud Computing Security Workshop*, pp. 103–112, 2012.
[4] A. Taha, R. Trapero, J. Luna, and N. Suri, "AHP-Based Quantitative Approach for Assessing and Comparing Cloud Security," *Proc. of Trust, Security and Privacy in Computing and Communications*, pp. 284–291, 2014.
[5] K. Garg, S. Versteeg, and R. Buyya, "A framework for ranking of cloud computing services," *In Future Generation Computer Systems*, vol. 29, no. 4, pp. 1012–1023, 2013.
[6] Z. Rehman, F. Hussain, and O. Hussain, "Towards multi-criteria cloud service selection," *Proc. of Innovative Mobile and Internet Services in Ubiquitous Computing*, pp. 44–48, 2011.
[7] J. Luna, A. Taha, R. Trapero, and N. Suri, "Quantitative reasoning about cloud security using service level agreements," *In Trans. on Cloud Computing*, no. 99, 2015.
[8] A. Li, X. Yang, S. Kandula, and M. Zhang, "Cloudcmp: comparing public cloud providers," *Proc. of Internet Measurement*, pp. 1–14, 2010.
[9] J. Siegel and J. Perdue, "Cloud services measures for global use: the service measurement index (smi)," *Proc. of Global Conference*, pp. 411–415, 2012.
[10] S. Chaves, C. Westphall, and F. Lamin, "SLA perspective in security management for cloud computing," *Proc. of Networking and Services*, pp. 212–217, 2010.
[11] G. Frankova and A. Yautsiukhin, "Service and protection level agreements for business processes," *Proc. of European Young Researchers Workshop on Service Oriented Computing*, pp. 38–43, 2007.
[12] L. Krautsevich, F. Martinelli, and A. Yautsiukhin, "A general method for assessment of security in complex services," *Proc. of Towards a Service-Based Internet*, pp. 153–164, 2011.
[13] S. Habib, S. Ries, and M. Mühlhäuser, "Towards a trust management system for cloud computing," *Proc. of Trust, Security and Privacy in Computing and Communications*, pp. 933–939, 2011.
[14] S. Wang, L. Zhang, S. Wang, and X. Qiu, "A cloud-based trust model for evaluating quality of web services," *In Computer Science and Technology*, vol. 25, no. 6, pp. 1130–1142, 2010.
[15] I. Haq, R. Alnemr, A. Paschke, E. Schikuta, H. Boley, and C. Meinel, "Distributed trust management for validating sla choreographies," *In Grids and service-oriented architectures for service level agreements*, pp. 45–55, 2010.
[16] "Amazon CloudWatch," http://awsdocs.s3.amazonaws.com/AmazonCloudWatch/latest/acw-dg.pdf.
[17] CloudStack, "Open source cloud computing," 2013.
[18] A. Ayad and U. Dippel, "Agent-based monitoring of virtual machines," *In Information Technology*, vol. 1, pp. 1–6, 2010.
[19] G. Aceto, A. Botta, W. Donato, and A. Pescapè, "Cloud monitoring: A survey," *Computer Networks*, vol. 57, no. 9, pp. 2093 – 2115, 2013.
[20] L. Hamers, Y. Hemeryck, G. Herweyers, M. Janssen, H. Keters, R. Rousseau, and A. Vanhoutte, "Similarity measures in scientometric research: the jaccard index versus salton's cosine formula," *In Information Processing & Management*, vol. 25, no. 3, pp. 315–318, 1989.