Self-Adaptive Access Control & Delegation in Cloud Computing

Ali Ahmad Malik
Department of Computing, SEECS
National University of Sciences & Technology
Islamabad, Pakistan
13msccsamalik@seecs.edu.pk

Hirra Anwar
Department of Computing, SEECS
National University of Sciences & Technology
Islamabad, Pakistan
hirra.anwar@seecs.edu.pk

Muhammad Awais Shibli
Department of Computing, SEECS
National University of Sciences & Technology
Islamabad, Pakistan
awais.shibli@seecs.edu.pk

***Abstract*— Cloud computing has seen significant progress in the last decade, and is still on the rise. A big number of companies are shifting their infrastructure to the cloud because of the numerous advantages it offers, such as rapid elasticity and scalability. However, moving to the cloud also presents a number of challenges. In this paper we focus on one of the major challenges, i.e., access control and delegation in a cloud environment. There exist many frameworks and models that solve issues pertinent to this area and allow dynamic delegation of access rights. However, none of them are robust enough to handle unprecedented changes, i.e., situations not handled at the time of requirements engineering. We've presented a risk-based, self-adaptive access control and delegation framework that caters to such situations by continuously monitoring and evaluating the user risk score and environmental factors, and automatically adapting to the changes incurred by reconfiguring the system. The proposed framework is able to adapt to unprecedented change as it can delegate access rights to previously unauthorized users in an emergency situation and revoke users' access rights based on environmental factors, while running in a continuous feedback loop.**

Keywords: Self-Adaptive Systems, Access Control & Delegation, Continuous Monitoring, Adaptation to Unanticipated Events

I. INTRODUCTION

In a cloud-computing environment, different Cloud Service Providers (CSPs) have to establish trusted relationships with each other at runtime in order to share each other's resources. Through this relationship, users can not only use resources of other trusted CSPs, but can also delegate access rights. Let's say there is a user U1, belonging to CSP A, who has access on a certain resource R. The user needs to avail his annual vacation, and there's no other senior member who can take his position for that time interval. Fortunately, the CSP the user U1 belongs to, has formed a federation with another CSP, B. Both the CSPs represent different departments of the same organization in this case. The user U1 knows that he can delegate his access rights to another user U2 who belongs to CSP B, as both the CSPs have formed a federation between them. So while U1 is on leave, the user U2 can take care of critical tasks, essentially access to critical resources, in his absence. Delegating access control is defined as delegating another user the access rights to perform some action on a given resource by a user who is the owner of that resource or meets some pre-defined criteria to do so. There exist many frameworks and models to assist in delegating access rights in a secure manner, but none of them handle uncertainty, i.e., situations or circumstances that weren't identified at the time of requirements engineering, and there is no automatic adaptation in system configuration either when the attributes of a user or a CSP change. Let's say there is an emergency situation and a person who doesn't have requisite attributes needs to be delegated the access rights of a Head of a Department (HoD). There doesn't exist any framework in the literature that handles this scenario but our proposed framework can as it is designed to adapt to changing user needs and environmental factors.

The proposed framework provides the users with a secure access control & dynamic delegation framework that continuously monitors, evaluates and adapts to the user risk score, based on the reputation of the user and the CSP it belongs to and situational factors such as time of access and change in IP address or location. Our proposed framework caters for unprecedented changes, i.e., enables delegation of access rights to previously unauthorized personnel in an emergency state. It continuously monitors and evaluates all the users. So any malicious activity on part of the user will revoke his access for the current session and his overall risk score will drop as well. It is also capable to automatically adapt to changing conditions or situations without any human intervention.

The rest of the paper is divided as follows. Section II presents a brief overview on the different domains the proposed framework comprises of. In section III, we've given a brief overview of some of the existing access delegation models/ frameworks, and highlighted their shortcomings. In section IV we present our self-adaptive access control and delegation framework that adapts to unprecedented changes. Section V highlights a detailed workflow for our proposed framework. We have evaluated

978-1-5090-2239-7/16/$31.00 copyright 2016 IEEE
SNPD 2016, May 30-June 1, 2016, Shanghai, China

the framework and presented our results in Section VI. The paper is concluded in section VII, with references in the last section.

II. Background Study

Our proposed framework combines aspects from different domains, such as cloud computing, access control and delegation, and self-adaptive systems. In this section we present a brief overview of these domains.

As discussed in [1], cloud computing utilizes virtualization technology to provide on-demand and elastic computing services. There have been comparisons among cloud computing, grid computing, utility computing and autonomic computing [2], but cloud computing differs in many aspects and leverages some aspects of these technologies as well [1]. The NIST definition of cloud computing seems to cover all aspects of cloud computing [3]:

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." [4]

Access control mediates users' requests to access a resource and determines allowed activities on a resource [16]. For example, in organization only staff from the finance department can view the employees' payroll. With an access control mechanism in place, it can be made sure that only the employees who work in the finance department or possess some specific attribute(s) are allowed access to payroll data of all the employees.

Delegation is defined as delegating the access rights one has over a resource to another user. Also, delegation is a convenient mechanism in a dynamic environment where it is not known in advance what privileges a user may need to accomplish a task. It also provides means to express and enforce access control policies [5].

A self-adaptive system (SAS) analyses changing user requirements, operating environment and resource availability, and adapts accordingly. Self-adaptive systems change their behavior whenever the evaluation shows the system isn't accomplishing what it was supposed to, or the system isn't performing up to its full capacity. With self-adaptive systems, situations can be handled that weren't defined at the time of requirements engineering [6-7]. Self-adaptive systems normally use a feedback loop for self-adaptation. The feedback loop typically consists of four activities [8]. The first activity collects the requisite data for self-adaptation. Then an analysis is made on the collected data, e.g., applying a model on the collected data. In the third activity, a decision is made on how the system should be adapted to achieve the desired state. The final activity implements re-configured system settings based on the decision made.

III. Literature Review

There have been proposed many frameworks and models in the literature to allow users to delegate their access rights to other users, but each of the proposed solutions have their own shortcomings as highlighted below.

The authors have presented an access control model in [9] in which every user is assigned to a certain domain, and each domain is associated with the role and actual job of the user. Each role of the user is assigned a task so as to practice his job role. The access and data flow is secured by labeling the data as per its sensitivity level, which could only be accessed by user roles that have been assigned a task that has a security classification that dominates the targeted data's security labels. In [10], the authors have devised a generic framework that provides a unified experience to the user to delegate his/ her access rights across different/ multiple domains. All the service providers have their own trust domains and don't necessarily trust each other, hence the lack of a common trusted anchor. This is resolved by what the authors call, "Abstract Delegation". It provides a unified interface that maps this abstract delegation to a concrete mechanism.

Ruan and Varadharajan in [11] have presented a logic-based framework that provides access and role delegation in a distributed environment. The authors have incorporated trust as a factor for access delegation in this framework and depicted how to reason with degree of trust for a certain delegation, privileges a certain role has, the level till which delegation could be propagated and conflict resolution through Extended Logic Programs, called the Role Based Authorization Program. The framework also enforces cardinality constraints and provides separation of privileges and role composition. In [12], Tamassia et al. have devised a protocol called Role Based Cascaded Delegation (RBCD) that provides cross-domain delegation efficiently and could be verified independently without the intervention of an administrator via aggregate signatures that store the authentication information of a role based delegation in a short constant sized signature. In this model, a delegation recipient may further extend the delegation to another entity along with the delegation certificate signed by the first delegation recipient. The delegation credential includes the public key of the next recipient that inherently creates a chain of trust. The framework also supports accountability. The authors in [13] have proposed to use DOMAIN RBAC including the Object Isolation feature. Object Isolation is defined as a method to allow access to a resource only to selected users from a set of authorized users. In other words, a user can access a resource only if he's a member of the domain the resource belongs to. This model limits the authorized users' authority to a restricted set of resources and provides a fine-grained access control mechanism.

Although problems such as lack of accountability (Delegator or Delegatee Accountability), and infinite access to a resource (No revocation of access rights) have been handled in part in different frameworks, but the problems we've identified in the literature have not been addressed before. The problems we've highlighted in the literature are: Continuous Monitoring and Automatic Adaptation. We discuss these problems/ factors below.

A. Continuous Monitoring

This is a very important issue in the area of access control and delegation. It often happens that the person being delegated access rights to a resource, misuses his rights. This is because once the access rights have been granted, there is no continuous monitoring on the delegated resource or the delegatee and the delegatee might start using the resource for malicious activities.

B. Automatic Adaptation

This factor is vital to the dynamic needs of users in a cloud paradigm. There may arise cases that weren't thought of beforehand. This factor is concerned with whether the framework or model handles a change previously unknown to the system without any human intervention, or simply ignores it.

C. Revocation of Access Rights

It is of paramount importance that whenever a user is seen to be misusing his/ her rights, whether or not the rights are revoked to avoid further damage to the system, or he has been granted access to the resource for an infinite time without any check and balance.

D. Delegatee Accountability

This factor highlights whether or not there has been made any distinction on who's the actual user of a resource at any given time, i.e., the delegator or the delegatee. As otherwise it may make accountability a challenging task.

Our literature review shows that none of the frameworks or models provides continuous monitoring or automatic system adaptation in case of a previously unidentified event. Only in [12], the access of a certain user can be revoked in case his behavior is below par. The models in [9], [11], and [12] have a provision of identifying whether the delegator or one of the delegatees exercised the access rights. The review is done to show that none of the existing frameworks have the capacity to handle complex scenarios in a dynamic cloud-computing environment. We present our proposed framework in the next section to address these challenges.

IV. Architecture Of The Self-Adaptive Framework

In this section, we've proposed a self-adaptive access control delegation framework that caters to the limitations in existing frameworks and models. It offers the features of continuous monitoring and handling uncertainty with automatic system adaptation. Our proposed framework comprises of two major components: Trust as a Service (TaaS) layer and Access Control as a Service (ACaaS) layer. These two components are further subdivided in to sub-modules, as shown in the architecture block diagram in Fig. 1.

A. Trust as a Service Layer

Ayehsa et al. have comprehensively described the Trust as a Service (TaaS) layer and its modules in [14]. However, for the sake of completion and to show our enhancements, we have given an overview of this layer in this section. This layer is used whenever two or more CSPs need to evaluate each other's trust level to form a cloud federation. This layer uses the Trust Evaluation Module (TEM) to evaluate the trust level of a CSP. We've enhanced the TEM module in [14] so that it also evaluates the trust level of Cloud Consumer (CC)/ end-user, which will be used in conjunction with CSP's trust level in the Access Control as a Service (ACaaS) layer for delegation of access rights. TEM comprises of the following modules.

1) Registration Management (RM) Module: All the Cloud Service Providers (CSPs) and Cloud Consumers (CCs) / end users have to register with the TEM. The CSPs provide their SLAs to the RM module, and the CCs provide their requirements from CSPs at the time of registration. The CSPs also have to send metadata, i.e., endpoint URL, service URL and service type to this module. The SLAs of CSPs are used by the SLA Management Module to derive their trust level.

2) SLA Management (SM) Module: The Service Level Agreement (SLA) Management Module calculates the SLA score of a CSP based on the parameters identified in the SLA. This score is combined with the score of other modules to derive the trust level of a CSP. It comprises of the following three sub-modules.

1) SLA Repository (SR) that stores and maintains all the SLAs provided by CSPs at the time of registration. 2) Parameter Extraction (PE), the module that retrieves the SLA for the CSP in question from the SLA Repository (SR) and, parses and extracts the essential Quality of Protection (QoP) attributes the CSP offers. 3) SLA based Trust Evaluation (STE), this module receives the Extracted Parameters (EP) from the Parameter Extraction module to calculate the SLA score of the CSP. The EPs are compared against a set of security features / parameters represented by $S = \{C, I, A, AC, AU\}$ that denote confidentiality, integrity, availability, authentication and authorization, respectively. Each of these parameters can be assigned dynamic weights to achieve a custom security level, by any of the participating CSPs, represented by the set $W = \{wC, wI, wA, wAC, wAU\}$. Equation (1) is used to evaluate the SLA score of a CSP.

$$T_{SLA} = \sum_{i=0}^{n} \frac{W_i * EP_i}{|S|} \quad (1)$$

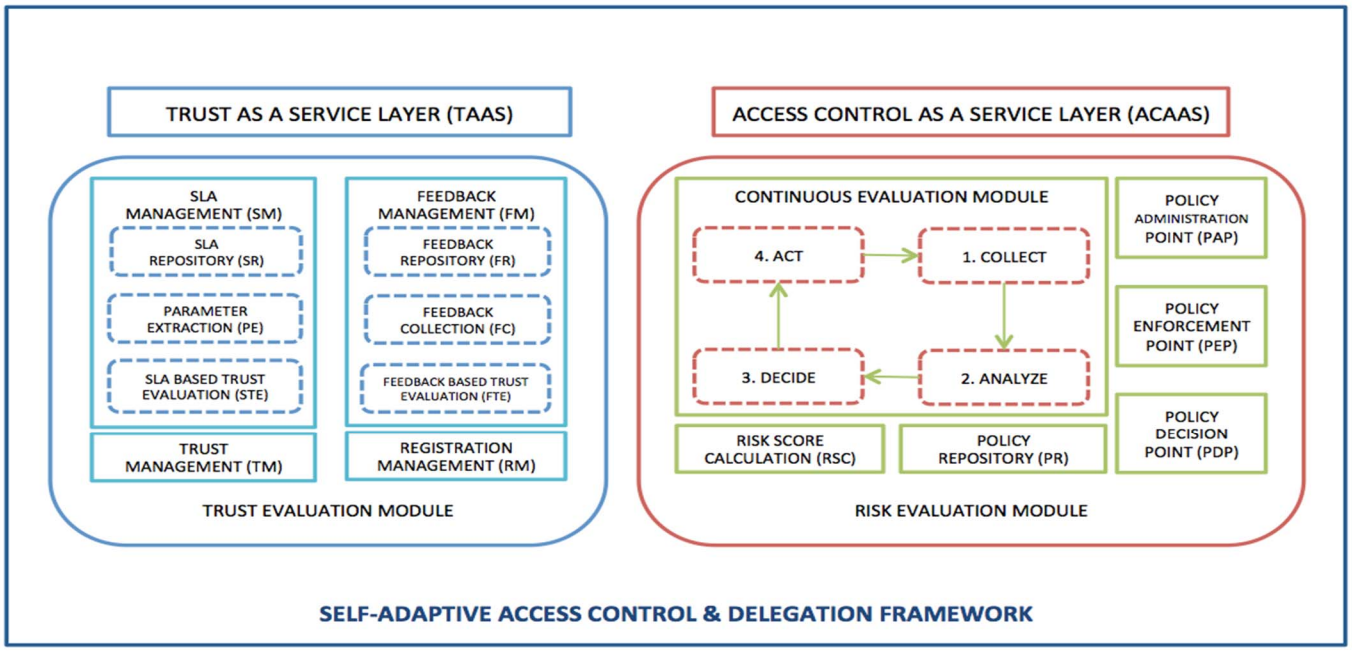

Fig. 1: Framework Architecture Block Diagram

3) Feedback Management (FM) Module: This module is responsible for calculating the feedback score of a CSP or a CC based on the feedback given by other CSPs or by the system. It consists of a Feedback Repository (FR) module that is responsible for storing and maintaining the feedback collected in the backend database. It also has a Feedback Collection (FC) that is responsible for collecting the feedback of CSPs and CCs. This feedback is collected either from CSPs when they interact with each other or the system itself updates it after observing the behavior of a CSP or a CC. The third sub-module, Feedback based Trust Evaluation (FTE), gets the feedback from the FR module, and uses the subjective logic algorithm from [15] for feedback evaluation. The subjective logic technique is based on subjective opinions pertinent to trust of a certain proposition. The proposition in this case is the trustworthiness of the Cloud Service Provider (CSP) or the Cloud Consumer (CC), represented by *C*. Each proposition is assigned a weight by a certain user, represented by W_C^{User}. The opinions of all the users about a CSP or a CC are represented as follows.

$$W_C^{User1}, W_C^{User2}, W_C^{User3}, \ldots, W_C^{UserN}$$

Where $W_C^{User} = (b,d,u,a)$,

$$b = \frac{Positive_{feedback}}{Collected_{feedback}+N},$$

$$d = \frac{Negative_{feedback}}{Collected_{feedback}+N},$$

$$u = \frac{N}{Collected_{feedback}+N}, \quad a = \frac{1}{N}$$

The variable *b* represents belief about the proposition being true and *d* represents the disbelief, with their values being derived from the positive feedback and negative feedback respectively. The variable *u* represents the uncertainty and the variable *a* represents the base probability. We use the fusion operator to aggregate the opinions of all the users, as follows.

$$W_C^{User1} + W_C^{User1} = (b_{new}, d_{new}, u_{new}, a_{new}) \quad (2)$$

$$b_{new} = \frac{(b_C^{User1} * u_C^{User2}) + (b_C^{User2} * u_C^{User1})}{(u_C^{User1} + u_C^{User2} - u_C^{User1} * u_C^{User2})}$$

$$d_{new} = \frac{(d_C^{User1} * u_C^{User2}) + (d_C^{User2} * u_C^{User1})}{(u_C^{User1} + u_C^{User2} - u_C^{User1} * u_C^{User2})}$$

$$u_{new} = \frac{(u_C^{User1} * u_C^{User2})}{(u_C^{User1} + u_C^{User2} - u_C^{User1} * u_C^{User2})}$$

In (2) we've shown how to aggregate opinions of two users using the fusion operator. The same way it could be aggregated for N users' opinions by iterating it N number of times. The aggregation of all the users' opinions represents an expected value E, as shown in (3).

$$T_C^{feedback} = E = b + a * u \quad (3)$$

4) Trust Management (TM) Module: The TEM passes the requests received for trust evaluation of a CSP or a CC to this module. This module collects and combines the scores from the SLA Management (SM) and Feedback Management (FM) module in case of trust evaluation of a CSP as per the following equation.

$$T_{CSP} = \frac{T_{SLA} + T_C^{feedback}}{2} = \frac{T_{SLA} + T_{CSP}^{feedback}}{2} \quad (4)$$

In case of trust evaluation of a CC, the feedback score is used as the trust score, as per the following equation.

$$T_{CSP} = T_C^{feedback} = T_{CC}^{feedback} \quad (5)$$

The trust level of a CSP is calculated as per the above equation. The values of this equation are categorized in Table I, where Level 1 is the lowest trust a CSP or CC can achieve and Level 5 is the highest.

TABLE I
TRUST LEVELS AND THEIR VALUES

Trust Value	Level of Trust
0.0 < T < 0.2	**Level 1**
0.2 < T < 0.4	**Level 2**
0.4 < T < 0.6	**Level 3**
0.6 < T < 0.8	**Level 4**
0.8 < T < 1.0	**Level 5**

B. Access Control as a Service Layer

This layer is the heart of the framework, and is responsible for delegating access rights to end-users by continually monitoring, evaluating and automatically adapting to variables such as user the risk score and environmental factors. It consists of the following modules.

1) Risk Score Calculation (RSC): This module calculates the associated Risk Score (RS) of the CC who is to be delegated a set of rights, as per (5). The equation is computed by fetching the trust score of the Cloud Service Provider (CSP) and Cloud Consumer (CC) in question from the Trust Evaluation Module (TEM) of the Trust as a Service (TaaS) layer.

$$RS = \frac{(1 - T_{CC}) + (1 - T_{CSP})}{2} \quad (6)$$

Where $0 < T_{CC} \leq 1$ *and* $0 < T_{CSP} \leq 1$

Here T_{CC} represents the trust score of the Cloud Consumer (CC), and T_{CSP} represents the trust score of the Cloud Service Provider (CSP) to which the CC belongs. The risk score of a CC is dependent on both T_{CC} and T_{CSP}.

The levels defined in Table II are used in the Policy Administration Point (PAP) at the time of creating/ updating policies to define the level of risk associated with a resource. The Risk Level 1 is associated with the most critical resources, and Risk Level 5 is associated with the least critical resources, using same categorization as of [14].

2) Policy Administration Point: This module is responsible for the creation, management and debugging of access control policies. A policy administrator uses this module to create or update policies. The major inputs required by this module for creating a policy are Policy Name, Resource for which the policy is being made, Required Risk Level of subjects for access, Delegation Level, i.e., whether the delegatees can further delegate access and Time Usage Interval.

TABLE II
RISK LEVELS AND THEIR VALUES

Risk Score	Level of Risk
0.0 < RS < 0.2	**Level 1**
0.2 < RS < 0.4	**Level 2**
0.4 < RS < 0.6	**Level 3**
0.6 < RS < 0.8	**Level 4**
0.8 < RS < 1.0	**Level 5**

3) Policy Enforcement Point (PEP): The role of this module is to enforce the decisions made by the Policy Decision Point (PDP). It acts as a logical entity that controls access to a resource. Whenever a user makes request to access a resource, the request is intercepted by the PEP. The PEP forwards this request to the PDP. The PDP evaluates the request and sends back a response to the PEP. The PEP finally enforces the decision it received as a response from the PDP.

4) Policy Decision Point (PDP): This module evaluates the request for accessing the resource in question. The module fetches the record from the Access Control Module (ACM) for the said resource and the user trying to access it. If an entry is found, a grant decision is returned; a deny decision otherwise.

5) Policy Repository (PR): This module is used for the storage and maintenance of the access policies created by the PAP. Each of the policies created is uniquely identifiable.

6) Access Control Module (ACM): The ACM module is used to grant access to individual users, by maintaining an Access Control List (ACL). The access rights are either granted by an administrator or delegated by another end-user, if allowed. The administrator can grant access rights only to those users who fulfil the policy criteria of the resource in question. If the policy allows, end-users can also delegate their access rights to others via this module. There could be an emergency situation in which access has to be granted to a user who doesn't fulfil the policy criteria, a case we will see in the next section. The ACM will grant access to such a user only if he hasn't been involved in any malicious activity for a pre-defined time interval *t*.

7) Continuous Evaluation Module (CEM): The Continuous Evaluation Module (CEM) is the most vital module of the system. This module continually fetches the trust score from the Trust Evaluation Module, and passes it to the Risk Score Calculation Module to evaluate the risk score. It also maintains the access history of each Cloud Consumer (CC) by noting his/ her usage timings and change in location, i.e., situational factors. It can also revoke the access rights of a user if his/ her activities are found to be

malicious. A user is marked malicious if he/ she is exercising his/ her access rights at an unusual time and/ or for longer intervals on a critical resource, e.g., a malicious employee trying to access the company's financial database after close of business hours. A user is also marked malicious when there is seen a sudden change in the user's location, especially if he'd previously been granted access on a critical resource or he/ she was filling in for someone else in an emergency situation.

This module has further four sub-modules that run in a closed loop. The first module, the Collect module, fetches the Trust score for the Cloud Consumer (CC) and his Cloud Service Provider (CSP) from the Trust Evaluation Module (TEM). The Analyze module then passes the trust score of the user and his CSP's to the RSC module for evaluating the user's Risk Score (RS). It also calculates the user's access history. If the activities of the user are found to be malicious, or if the user risk score is lower than the risk level defined in the policy of a certain resource, the Decide module makes a decision to revoke the user's access on the said resource. If the decision has been made by the *Decide* module to revoke a user's access, then the Act module removes the malicious user's entry from the ACL in the ACM. The Collect module is invoked when the Act module completes its tasks, and the cycle restarts.

V. Framework Workflow

In this section, we present a real life use case to better present the internal working of our framework, and highlight the advantages it has over existing frameworks and models.

In our use case, we have taken the example of a university which has a number of schools. Each of the schools is self-sufficient, and is using services of a certain Cloud Service Provider (CSP). The faculty, staff and students are the Cloud Consumers (CC). These consumers need to access resources of other schools quite often, e.g., one of the schools might not be equipped with powerful servers that students or faculty need to use for their experimentation. In order for the resources to be shared, the CSPs providing services to different schools need to form a federation. Our framework provides a Policy Administration Point (PAP), which the administrators can use to create or update access control policies in the Policy Repository (PR).

Now, let's assume there's an emergency situation where the university has to decide whether or not it wants to be a part of the strike against the government for increasing funds of schools in the rural areas, which is scheduled to take place in two days. The university has a set policy in such matters that it will only take part in such strikes only if all the Head of Departments (HoDs) of all the schools agree to do so. But unfortunately, one of the HoDs is on a leave of absence, and he can't sign off the requisite document to show his school's willingness. However, it is mandatory for the school to show its willingness otherwise the university will not be able to take a decision, but none of the other staff members have access on the requisite document to sign it off. If any other access control and delegation framework was being used, there'd be no any other way around this unprecedented situation, but our framework handles these unanticipated changes by the adapting to the user requirements and needs. Any HoD of a different school can delegate his access to another senior member of the school who's HoD is on leave, and the senior member can make the decision on the absent HoD's behalf. The HoD (Delegator) may or may not trust him (Delegatee), but the senior member needs to have a record of no malicious activities for a pre-defined interval of time *t*, which is specified in the policy specification. Even if the senior member isn't trusted by the HoD (Delegator), the HoD needs not to worry for allowing the senior member the access to the resource, as the system continually monitors activities of all users in a feedback loop. If any user's activities are found to be malicious, their access rights are revoked and their risk scores increased as well. A malicious activity can be anything ranging from usage at unusual times or longer intervals to sudden change in user location. The scenario is depicted in Fig. 2.

The delegator delegates access in step 2 via interacting with ACM in step 1. The delegatee tries to access a resource when his request is captured by the PEP in step 3. The PEP passes the request to PDP in step 4. The PDP fetches the user's current risk score from the RSC module in 5a, and the required risk score to access the resource in question from the PR in 5b. The RSC fetches the user's trust score from TEM in 6, and returns the user's risk score after calculations in 7a. The PR returns the required risk score for accessing the resource in 7b. The PDP then fetches the user's entry from the ACL list from ACM in step 8. If no entry is found, or the current risk score is so less than the one present in the ACL entry so that it doesn't match the risk score returned in 7b, the access is denied and granted otherwise and the result is passed onto the PEP in 9, and then passed to the user in step 10. It demonstrates the system's automatic adaptation capability, while the Continuous Evaluation Module is running in the background that updates user/ CSP's trust score based on their behavior.

VI. Evaluation and Results

In this section we present an evaluation of our framework. The framework generates access policies in the XACML format and we have validated these policies with WSO2 Identity Server [17]. For verifying the framework's conformance against factors not addressed in existing frameworks/ models, namely continuous evaluation and automatic adaptation for granting and/ or revoking access rights, we have validated it by designing unit test cases with Junit [18] and evaluated the framework with the NetLogo

[19] simulation tool. Due to space limitation, we will only discuss NetLogo simulation here.

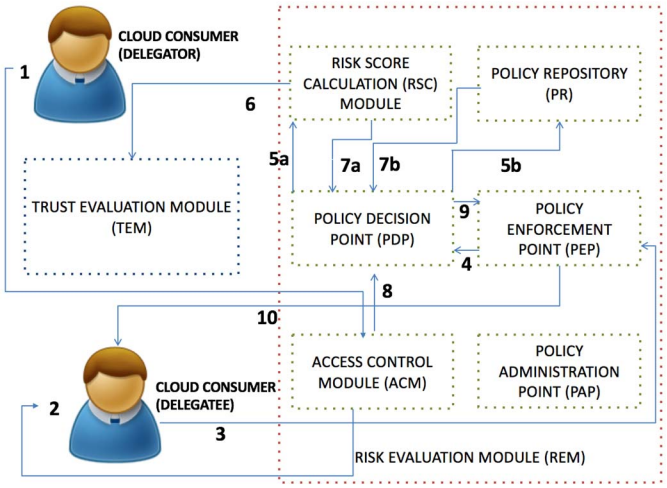

Fig. 2: Framework Workflow

A. NetLogo Simulation

In this simulation, N_{TOTAL} number of users take part for gaining access to a resource. N_A number of users are marked as authorized out of N_{TOTAL} and the rest are marked as N_U with a probability P_A. We have termed authorized users as users who already have access to the resource in question, and the rest as unauthorized users. Over the period of time, there may arise cases where some user has to take place of another user to perform an urgent operation, but the user may not have the requisite risk score to take place of the other user. To simulate this scenario, every time an authorized user tries access the resource, the system delegates his access rights to an unauthorized user with a probability P_E, and the delegatee then accesses the resource in the authorized user's place. With this simulation, we show our framework's capability to adapt to unidentified scenarios and delegates access to previously unauthorized users in a case of emergency. The following Fig. 3 shows that 21 such cases were handled in a time span of over 300 seconds for N_A number of users. We can also observe from the graph that the need for such scenarios also decreases over time.

We have also simulated our framework functionality when a user's behavior becomes malicious. Another use case for our framework's continuous monitoring and automatic adaptation comes in to play when the user behavior changes at runtime and his/ her access may need to be revoked. To simulate this case, on each resource access the user is marked as a malicious user with a probability P_B. As soon as a malicious user is found trying to access a resource, our framework identifies the malicious user and revokes his/ her access on the resource(s). Over time, the users have a chance to improve their behavior and their risk score to regain access on any resource. In Fig. 4, we have shown the number of malicious access requests handled by our system, which is 12 over a period of 300 seconds

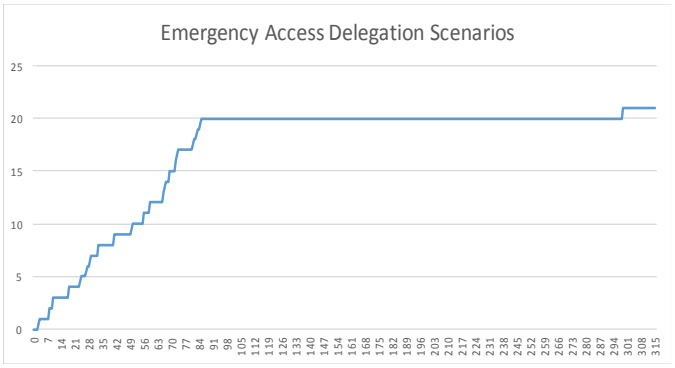

Fig. 3: Emergency Access Delegation Scenario. X-axis shows time in seconds, Y-axis shows the number of Emergency Access Delegation Scenarios Handled by the framework

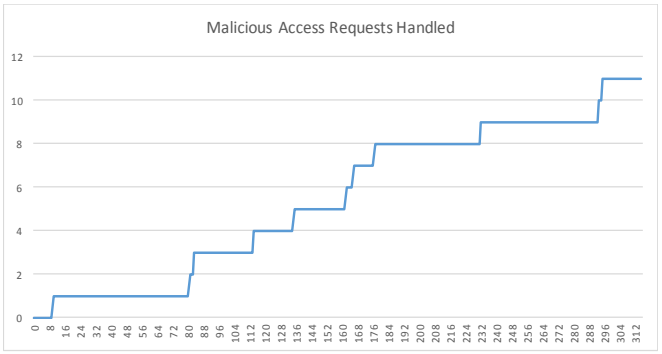

Fig. 4: Malicious Access Requests Handled. X-axis shows time in seconds, Y-axis shows the number of Malicious Requests handled by the framework

B. Results

Access delegation is a common practice to delegate one user's access rights to another user and has been implemented by many frameworks and models. However, these frameworks and models allow access delegation only in a predefined way, e.g., people with specific attributes or roles. Our proposed framework solves this dilemma as with our framework users can delegate their access rights to others who don't have the requisite roles or attributes. The framework automatically adapts to users' needs and allows to delegate their access rights in an emergency situation to previously unauthorized users.

The ability to grant access rights to previously unauthorized users may sound insecure, as those users may turn out to be malicious and harm other users. This holds true with the existing frameworks/ models where there is no continuous monitoring or evaluation. Our framework not only continually monitors and evaluates all the users, but also the environmental factors such as time of access and change in IP/ location. This makes sure that users who were granted access rights in an emergency situation can't harm other users even if they were malicious as the framework will identify such users from their activities and revoke their access. The framework also makes sure that such users can't access other critical resources until they have improved

their risk score by not indulging in such activities. However, increasing the security does impact the overall system performance, as explained in [20], you can't increase the security of a system without some performance degradation and vice versa.

In essence, our framework adapts to users' needs and changes its behavior to meet those needs, and continually evaluates and monitors not only users' activities but the environmental factors such as time of access and IP/ location, too that none of the existing frameworks/ models have the capability to do.

VII. CONCLUSION

In this paper we discussed issues pertinent to access control and delegation in a dynamic cloud-computing environment, as enterprises are adapting cloud services rapidly. We also highlighted shortcomings of the existing frameworks and models. We then proposed a risk-based, self-adaptive access control and delegation framework that takes into account factors including continuous monitoring of the delegated resources and the corresponding users, automatic system adaptation to unprecedented and/ or environmental changes in the system and revocation of malicious users' access from the system on change of behavior. The framework calculates the risk score of a user based on the user and his/ her's CSP's feedback and the CSP's SLA score. The framework continually monitors and keeps evaluating the users that have gained access to a resource, and also rescinds their access rights if they make any malicious advances. The framework also self-adapts according to the situational and environmental changes. One of the directions for the future of this framework could be taking additional environmental factors into account, rather than just time and location.

REFERENCES

[1] Shahzad, Farrukh. "State-of-the-art survey on cloud computing security Challenges, approaches and solutions." *Procedia Computer Science* 37 (2014): 357-362.

[2] Zhang, Qi, Lu Cheng, and Raouf Boutaba. "Cloud computing: state-of-the-art and research challenges." *Journal of internet services and applications* 1.1 (2010): 7-18.

[3] Takabi, Hassan, James BD Joshi, and Gail-Joon Ahn. "Security and privacy challenges in cloud computing environments." *IEEE Security & Privacy* 6 (2010): 24-31.

[4] Mell, Peter, and Tim Grance. "The NIST definition of cloud computing." (2011).

[5] Pham, Quan, et al. "On a taxonomy of delegation." *computers & security* 29.5 (2010): 565-579.

[6] Qureshi, Nauman A., Ivan J. Jureta, and Anna Perini. "Towards a requirements modeling language for self-adaptive systems." *Requirements Engineering: Foundation for Software Quality*. Springer Berlin Heidelberg, 2012. 263-279.

[7] Qureshi, Nauman A., Ivan J. Jureta, and Anna Perini. "Requirements engineering for self-adaptive systems: Core ontology and problem statement." *Advanced Information Systems Engineering*. Springer Berlin Heidelberg, 2011.

[8] Brun, Yuriy, et al. "Engineering self-adaptive systems through feedback loops." *Software engineering for self-adaptive systems*. Springer Berlin Heidelberg, 2009. 48-70.

[9] Younis, Younis A., Kashif Kifayat, and Madjid Merabti. "An access control model for cloud computing." *Journal of Information Security and Applications* 19.1 (2014): 45-60.

[10] Bussard, Laurent, Anna Nano, and Ulrich Pinsdorf. "Delegation of access rights in multi-domain service compositions." *Identity in the Information Society* 2.2 (2009): 137-154.

[11] Ruan, Chun, and Vijay Varadharajan. "Dynamic delegation framework for role based access control in distributed data management systems." *Distributed and Parallel Databases* 32.2 (2014): 245-269.

[12] Tamassia, Roberto, Danfeng Yao, and William H. Winsborough. "Independently verifiable decentralized role-based delegation." *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* 40.6 (2010): 1206-1219.

[13] Ranganathan, Vidya, and Guha P. Venkataraman. "Object Isolation for Cloud with DOMAIN RBAC." *Cloud Computing in Emerging Markets (CCEM), 2012 IEEE International Conference on*. IEEE, 2012.

[14] Kanwal, Ayesha, Rahat Masood, and Muhammad Awais Shibli. "Evaluation and establishment of trust in cloud federation." *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication*. ACM, 2014.

[15] Jøsang, Audun, and David McAnally. "Multiplication and comultiplication of beliefs." *International Journal of Approximate Reasoning* 38.1 (2005): 19-51.

[16] Ferraiolo, David F., Vincent C. Hu, and D. Rick Kuhn. "Assessment of Access Control Systems." (2007).

[17] WSO2, "WSO2 Identity Server", http://wso2.com/products/identity-server/, accessed: 2016-01-08.

[18] JUnit, "JUnit", http://junit.org/ accessed: 2016-01-08.

[19] NetLogo, "NetLogo", https://ccl.northwestern.edu/netlogo/ accessed: 2016-01-08.

[20] Wolter, Katinka, and Philipp Reinecke. "Performance and security tradeoff." Formal methods for quantitative aspects of programming languages. Springer Berlin Heidelberg, 2010. 135-167.