A Multi-level Policy Engine to Manage Identities and Control Accesses in Cloud Computing Environment

Faraz Fatemi Moghaddam[1,2(✉)], Süleyman Berk Çemberci[3], Philipp Wieder[1], and Ramin Yahyapour[1,2]

[1] Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG), Göttingen, Germany
{faraz.fatemi-moghaddam, philipp.wieder, ramin.yahyapour}@gwdg.de
[2] Institute of Informatics, Georg-August-Universität, Göttingen, Germany
[3] Systeme, Anwendungen und Produkte in der Datenverarbeitung (SAP), Walldorf, Germany
suleyman.berk.cemberci@sap.com

Abstract. Security challenges are the most important obstacles for the advancement of IT-based on-demand services and cloud computing as an emerging technology. Lack of coincidence in identity management models based on defined policies and various security levels in different cloud servers is one of the most challenging issues in clouds. In this paper, a policy-based user authentication model has been presented to provide a reliable and scalable identity management and to map cloud users' access requests with defined polices of cloud servers. In the proposed schema several components are provided to define access policies by cloud servers, to apply policies based on a structural and reliable ontology, to manage user identities and to semantically map access requests by cloud users with defined polices.

Keywords: Cloud computing · Security · Policy management
Identity management · Access control

1 Introduction

Cloud security issues are mainly classified to three major categories [1]: data protection in cloud-based data centers, isolated and secure resource provisioning and reliable access control by identity management and authentication procedures [2]. These concerns are the most apparent reasons why most of individuals and businesses still have doubt to delegate management of their sensitive data to cloud service providers as third party collaborates [3]. One of the most challenging security issues in clouds that has led to the appearance of several researches and solutions is to ensure reliable accesses to different cloud servers based on various policies in each server. In fact, service providers needs to manage access requests and map them to resources according to defined policies from cloud customers or service providers [4].

K. Kritikos et al. (Eds.): ESOCC 2018, LNCS 11116, pp. 120–129, 2018.
https://doi.org/10.1007/978-3-319-99819-0_9

Using a federated identity management schema is the most popular solution for managing accesses to different cloud servers with single identity. In recent years most cloud services have adopted OpenID [5] or Shibboleth [6] as the most independent and flexible authentication and identity management models in cloud-based platforms. The proliferation of these identity federations has allowed cloud users belonging to one network (known as home organization) to access the services provided by other networks (known as remote organizations), all members of the same federation [7]. Therefore, there isn't any necessity for cloud users to re-introduce their credentials for each access in different cloud servers. The most important characteristic of identity management models is to provide a framework with fast-authentication mechanisms [8], low access time and reduced authentication data exchanges between different service access requests [9]. Although the establishment of multiple security mechanisms in each node enhance the security of resources and reduces considerable processing power for manipulating sensitive and also non-sensitive data [10], the authentication data exchange and access time for cloud users in identity management models are also affected. In particular, two important concerns in cloud-based identity management models are still challenging:

- Managing defined policies in different virtualized nodes according to capabilities of service providers, requirements of resource owner and constraints.
- Mapping access requests to cloud-servers based on established security mechanisms and defined policies of each node.

In this paper, a policy-based user authentication model is presented to provide a reliable identity management mechanism for establishing multiple access policies in different virtualized nodes and mapping access requests to defined policies accordingly capabilities of cloud servers and requirements of resources.

2 Problem Description

As described in previous section, the main aim of proposed model is to manage identities based on defined policies in cloud servers. Each virtualized node in cloud-based data center is associated with set of policies. These polices are classified in several protocols according to Protection Ontology [11]. The classification of security policies are based on three main parts: Resource Protection (including cryptography and key management policies), Confidential Transport (including signature and transport policies) and Identity Management (including authentication and access control policies). The latter, which is the focus of this work, refers to the capabilities that are provided to ensure the reliable access mapping between requests and policies by managing identities based on capabilities of service provider and requirements of cloud users.

Assume that there are N virtualized node (server) in the cloud-based data center, denoted as $\{S_1, S_2, \ldots, S_N\}$, and the current authentication policy set of node S_n with $s \in \{1, 2, \ldots, N\}$ is $P(S_n) = \{p_1, p_2, \ldots, p_M\}$. Given I registered users' access requests waiting to be processed, denoted as $\{U_1, U_2, \ldots, U_I\}$, and each U_i is associated with specific identity set (authentication and authorization set):

$$AA(U_i) = \left\{ \begin{array}{c} ID_i, h(PW_i), (h(ID_i) \oplus h(PW_i)), (AP_1, h(AR_1), (h(AP_1) \oplus h(AR_1)))_i, \ldots, \\ \left(AP_j, h\left(AR_j\right), \left(h\left(AP_j\right) \oplus h\left(AR_j\right)\right)\right)_i \end{array} \right\}$$

where ID, PW, AP and AR are user ID, user basic password, access policy and access response respectively. There are several authentication and authorization (access) policies that are defined for each node to enhance the security level of the node in comparison between other nodes. The authentication policies are focus on confidentiality and integrity of resources, while the authorization policies are based on privacy and access management features of cloud resources. To provide a semantic mapping between requests and policies, each of authentication and authorization policies of a specific node need to be evaluated according to the characteristic of cloud user. The objective of suggested model is to map elements of the policy set for each node to appointed access responses for cloud users to provide decisive access permit. For instance, consider a cloud provider with different services (*e.g.* storage, platform, software, *etc.*) and each service has dedicated security policies (*e.g.* two factor authentication for storage and one-time single pass for software). The main problem is to address the process of mapping security requirements of these cloud services to defined authentication and authorization capabilities of the cloud user in identity set. Overall, the access request of specific node is granted if and only if the following equation is applied to the request:

$$\forall p_i \in P(S_n) : \left(\exists \left(AP_j, AR_j\right)_i : \left\{\left(p_i = \left(AP_j\right)_i\right) \wedge \left(\left(AR_j\right)_i = true\right)\right\}\right) \quad (1)$$

In fact, cloud user needs to provide additional authorization and authentication capabilities for nodes with higher security policies. The proposed model tries to manage access requests and map between access policies and authentication capabilities of cloud users.

3 Proposed Schema

Using an agent-based authentication model [12] to send access requests, to search on policy queues and to match access requests to a specific defined policy may seem like a plausible solution for achieving the goal. However, this agent-based authentication process in not scalable and takes lots of processing power to map between requirements and capabilities. Thus, the design of our proposed model is based on a different manner. Our schema uses a framework with several components to define, store, check and match policies with identity details. Figure 1 shows the overall architecture of our model.

3.1 Policy Engine

The main duty of Policy Engine is to define and generate authentication and authorization policies based on the structural Protection Ontology [10] for cloud customers according to security requirements. Protection Ontology is a policy language based on

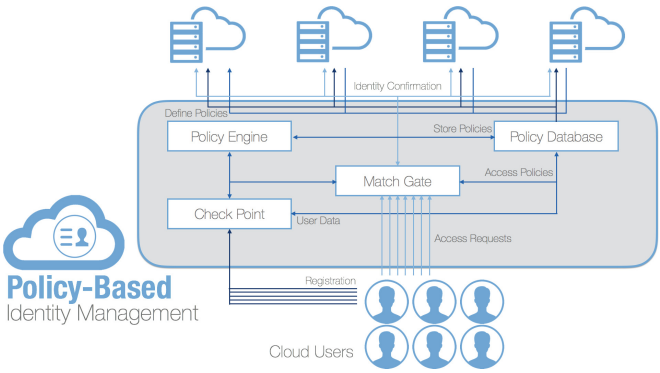

Fig. 1. Architecture of policy-based identity management

WS-Policy [13] as a recommended W3C [23] language for defining various security levels in cloud-based environments. Protection ontology classifies security algorithms to three main levels: Protocol, Mechanism and Algorithm. In the proposed model authentication and authorization capabilities of service provider are offered according to this classification. This structural classification helps to apply different security mechanisms to virtualized nodes and creates security levels based on requirements of cloud users and sensitivity of resources. Each of the offered algorithms is associated to a structural semantic resource for security level establishment according to the concepts of WS-Policy and Protection Ontology. A security-based SLA is the output of security ring (level) establishment and is defined as Security Level Certificate (SLC).

As described the main duty of policy engine is to define and generate authentication and authorization policies for different virtualized nodes according to the sensitivity of nodes and capabilities of service provider. The process of policy application is done by policy engine based on generated SLC as follows:

Step 1. Policy engine sends SLC ID to node n to apply policies of SLC to the node.
Step 2. According to the associated SLC, node n calls semantic resources of SLC to create $P(S_n)$.

$$for\left(i = 0\ to\ \sum_{\alpha=0}^{R}(rdf : Algorithm)_{\alpha}\right)$$

$$\left\{\begin{array}{c} p_i = \left(p_i \parallel add\left(SP_{\mu}\right)\right) \\ X(p_i) = h(p_i) \\ HP(S_n) = add(X(p_i)) \end{array}\right\}$$

where R is the total of semantic algorithm resources and $\mu = \sum_{\varepsilon=0}^{Count(HLSP)} (rdf : HLSP)_{\varepsilon}$ are the defined sub-policies for each algorithm based on the SLC. Also, the hashed value of each policy p_i is stored in the set $HP(S_n)$.

Step 3. $X(S_n) = h(x_n) \oplus h(p_1) \oplus h(p_2) \ldots \oplus h(p_n)$
where x_n and $X(S_n)$ are the secret key and the secret value for node n respectively.

Step 4. Send $\{(P(S_n), HP(S_n)), X(S_n)\}$ to Policy Database.
The SLC, policy set, hashed policy set and secret value of node n are sent to policy database.

3.2 Policy Check-Point

The check point component creates, updates and manages identities for accessing to different nodes. Identities are defined in registration phase, updated in checking phase and managed in access control phase. In recent years, two types of registration progresses are performed in web-based models:

- Normal Registration: The creation of personal identity within cloud provider with User ID, Password and other personal details.
- 3rd-Party Registration: Using identities in social media or other providers.

During the registration phase by each of these models, an Identity set (Authentication and Authorization) object $AA(U_i)$ is created from identity set class for user U_i. The basic identity set with the lowest identity details is associated with the ID and password:

$$I(U_i) = \{ID_i, h(PW_i), (h(ID_i) \oplus h(PW_i))\}$$

By the basic identity set, cloud users can access to the nodes with the lowest security level in cloud environment. However, three types of authentication and authorization access policies need to be defined and added to the identity set based on polices and capabilities of service provider:

- User Access Policies (UAP): These types of policies are defined by cloud users according to capabilities of cloud provider. For instance, cloud user can establish second password with an authenticator application or email.
- Cloud Access Policies (CAP): These types of policies are awarded to cloud users by the provider or admin after an identity validation (*e.g.* RBAC in a university).
- Temporary Access Policies (TAP): These types of policies are based on dynamic parameters such as location, hardware and time.

An access policy is defined in identity set according to the characteristics of policy by a triplex set as follows:

$$\left(AP_j, h\left(AR_j\right), \left(h\left(AP_j\right) \oplus h\left(AR_j\right)\right)\right)_i$$

where AP_j and AR_j refer to semantic resource access policy (*e.g.* two factor authentication by Email) and access responses (*e.g.* confirmed email address) respectively. Therefore, the authentication set for U_i are updated based on defined UAP, CAP and TAPs as:

$$AA(U_i) = \left\{ \begin{array}{c} ID_i, h(PW_i), (h(ID_i) \oplus h(PW_i)), (AP_1, h(AR_1), (h(AP_1) \oplus h(AR_1)))_i, \ldots, \\ \left(AP_j, h\left(AR_j\right), \left(h\left(AP_j\right) \oplus h\left(AR_j\right)\right)\right)_i \end{array} \right\}$$

3.3 Policy Match-Gate

The proposed identity management model for mapping accesses requests to defined policies is based on the performance of policy match gate. Given I registered users' access requests waiting to be processed, denoted as $\{U_1, U_2, \ldots, U_I\}$, and each U_i is associated with a specific authentication set $AA(U_i)$. The main aim of Match Gate is to process access requests and to map between these requests and defined polices for each node according to the identity set. To provide an efficient policy mapping algorithm, a session class is defined by policy match gate for creation of access session objects according to the capabilities of cloud users. The objects from this class (*AccessSession* class) use several security functions and parameters to ensure about the reliable mapping between capabilities and security policies. After the registration phase in the check point component, cloud users are able to sign in to cloud computing environment by their basic internal or external login information. A successful basic login lets the policy match gate to create a session object from the access session class for basic or additional security checking. The process of using this object for identity management is in number of steps as follows:

Step 1. An object is created from *AccessSession* class with basic parameters.

$$AccessSession\, ASU_i = new\, AccessSession(ID_i, h(PW_i), TS, (h(ID_i) \oplus h(PW_i)), TK_i, e)$$

where TK_i is a basic token for U_i and is valid if login details are matched with $A(U_i)$ and e is a Boolean property that shows the status of TK_i whether is enabled or disabled.

Step 2. The basic value of TK_i after the first login lets the cloud user to access basic nodes with lowest security level. In this level policy match gate checks if ΔTS and e are still valid, the access of cloud user to the root nodes are granted. The basic value of TK_i is calculated as follows:

$$TK_i = (h(h(ID_i \parallel TS) \oplus h(PW_i)))$$

Step 3. When the cloud user requests for accessing to basic nodes, the match gate calculates Node Access Request (*NAR*) as follows:

$$if\ ((e = true) \wedge (\Delta TS = Valid))\ then\ \left\{NAR_{(i,n)} = (TK_i, Enc(TK_i, x_n))\right\}$$

where *Enc* is AES-256 func. with the secret key for node n. The checking phase confirms the user identity and the value of *NAR* is sent from match gate to requested node.

Step 4. Server n receives the request from Match Gate and access is granted if the difference between timestamps and the following equation is valid:

$$if\ ((\Delta TS = Valid) \wedge (TK_i = Dec(TK_i, x_n)))\ then\ Access\ is\ Granted$$

This calculation helps to check if the secret key of node n is still valid or not. In fact, the validated identity from match gate can access to request node if the secret value of node is valid. If the validity of the equation is not confirmed, Match Gate should update the secret key of server n in database.

Step 5. If the cloud user requests for accessing to nodes with the defined security policies and higher privacy levels, further identity details are requested from Match Gate based on the defined policies. Thus, Match Gate checks $P(S_n)$ from policy database and asks U_i if UAP or TAP policies are needed for authentication and authorization checking. Also, the user database is checked by Match Gate for CAP policies for only authorization checking if needed. Each of the requested access details should be provided by the cloud user (*i.e.* UAP and TAP) or the user database (*i.e.* CAP) and ASU_i is updated according to the provided details:

$$\begin{aligned}
&ASU_i.AddAccessCapability(AP_1, h(AR_1), (h(AP_1) \oplus h(AR_1)), AST_1);\\
&ASU_i.AddAccessCapability(AP_2, h(AR_2), (h(AP_2) \oplus h(AR_2)), AST_2);\\
&\vdots\\
&ASU_i.AddAccessCapability\big(AP_j, h\big(AR_j\big), \big(h\big(AP_j\big) \oplus h\big(AR_j\big)\big), AST_j\big);
\end{aligned}$$

where *AST* is the Algorithm Session Time that shows the maximum validity of confirmed access response. For instance, the valid time for confirmed second password is longer that 1-time password. By each of the additional identity details the value of TK_i is updated as follows:

$$TK_i = TK_i \oplus \big(h\big(AP_j \parallel TS\big) \oplus h\big(AR_j\big)\big)$$

Step 6. After updating the value of Tk_i and confirming the identity of cloud user by additional identity request and according to the capabilities of user, the match gate sends server access requests to the requested node as follows:

$$X'(S_n) = \big(h(x_n) \oplus h(AP_1) \oplus h(AP_2) \ldots \oplus h\big(AP_j\big)\big)$$

$$if\ ((e = true) \wedge (\Delta TS = Valid))\ then\ \left\{NAR_{(i,n)} = (TK_i, Enc(TK_i, x_n), h(X'(S_n)), TS')\right\}$$

Step 7. Server n receives the request from Match Gate and access is granted if the ΔTS and the following equations are valid:

$$if\ ((\Delta TS = Valid)) \wedge (TK_i = Dec(TK_i, x_n)) \wedge (h(X(S_n)) = h(X'(S_n)))\ then\ Access\ is\ Granted$$

This calculation checks the validity of timestamp, the validity of secret key and finally the confirmed application and mapping process of defined policies by checking the validity of secret value.

Step 8. If the user requests to access to a node with common policies that were confirmed by match gate before and the Algorithm Session Time for the access response is still valid for the policy, just un-checked policies are evaluated and the is no necessity to re-check previous policies. In fact, every functions and properties of ASU_i are confirmed and stay reusable until the algorithm session time for that authentication or authorization algorithm is still valid. For instance, the session time for double authentication is less than single authentication and U_i needs to be double-authenticated again after the session time for double authentication is over while the session time for basic authentication is still valid. Also, re-authentication for some authorization access policies (*e.g.* Geographical or Software authenticators) or One-Time passwords need to be checked periodically or continuously. These valid session times are defined as sub-policies in the ring establishment stage based on Protection Ontology [11].

4 Discussion and Conclusion

In order to incarnate the superiorities of this schema in cloud-based environments, we give a performance analysis of the proposed model in this section. In this experiment the performance of match gate in different types of workloads was evaluated. Accordingly, the total process time for processing 500 access requests to VMs with high secure VMs with more authentication and authorization policies was examined in the first step. The aim of this case study is to examine the effects of static, continuously changing, dramatic increase and predictable increase workloads on the performance of match gate task management. The experiment was in 6 rounds based on different types of workloads. Figure 2 shows the results in details. In the static workload, the number of user accesses was same in all rounds. However, the total processing time was decreased slightly due to the common policies in different VMs. Thus, there was not any necessity to re-check common policies. As expected, in the dramatic increase of users requests, the total processing time was risen dramatically and in the respective rounds the total processing time was reduced considerably to the normal range. This change was less in predictable change due to the predictable scheduling in associated task processing. Finally, the rate of change in continuously increase of requests is

significantly slight. That was because two different effects: increase due to the number of requests and decrease due to the common policies in different VMs. Overall, the results show the performance of match gate task management for semantic mapping of access polices to request was scalable enough in different types of workloads.

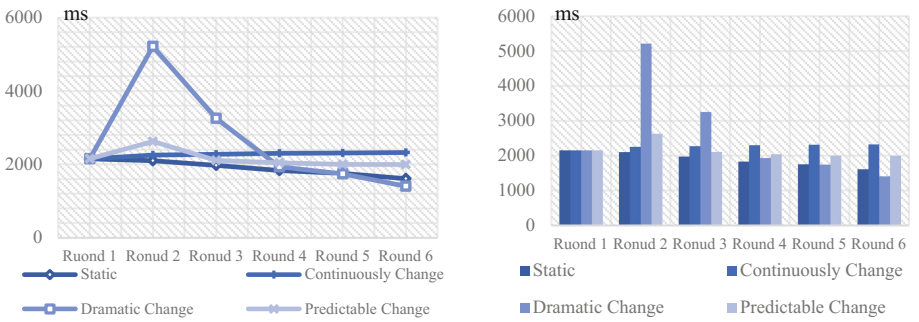

Fig. 2. Effects of different workload on the performance of match gate task management

Acknowledgement. This research has been supported by CleanSky project (No. 607584) funded by the Marie-Curie-Actions within the 7th Framework Program of the European Union (EU FP7).

References

1. Fatemi Moghaddam, F., Ahmadi, M., Sarvari, S., Eslami, M., Golkar, A.: Cloud computing challenges and opportunities: a survey. In: 1st International Conference on Telematics and Future Generation Networks (TAFGEN), pp. 34–38 (2015)
2. Sadiku, M.N.O., Musa, S.M., Momoh, O.D.: Cloud computing: opportunities and challenges. IEEE Potentials **33**(1), 34–36 (2014)
3. Wang, C., Ren, K., Lou, W., Li, J.: Toward publicly auditable secure cloud data storage services. IEEE Netw. **24**(4), 19–24 (2010)
4. Coppolino, L., D'Antonio, S., Mazzeo, G., Romano, L.: Cloud security: emerging threats and current solutions. Comput. Electr. Eng. **59**, 126–140 (2017)
5. Recordon, D., Reed, D.: OpenID 2.0: a platform for user-centric identity management. In: Proceedings of the Second ACM Workshop on Digital Identity Management - DIM 2006, p. 11 (2006)
6. Morgan, R.L., Cantor, S., Carmody, S., Hoehn, W., Klingenstein, K.: Federated security: the Shibboleth approach. Educ. Q. **27**(4), 12–17 (2004)
7. Pérez Méndez, A., Marín López, R., López Millán, G.: Providing efficient SSO to cloud service access in AAA-based identity federations. Futur. Gener. Comput. Syst. **58**, 13–28 (2016)
8. de Carvalho, C.A.B., de Castro Andrade, R.M., de Castro, M.F., Coutinho, E.F., Agoulmine, N.: State of the art and challenges of security SLA for cloud computing. Comput. Electr. Eng. **59**, 141–152 (2017)
9. Liu, Z., Yan, H., Li, Z.: Server-aided anonymous attribute-based authentication in cloud computing. Futur. Gener. Comput. Syst. **52**, 61–66 (2015)

10. Fatemi Moghaddam, F., Wieder, P., Yahyapour, R.: Policy Engine as a Service (PEaaS): an approach to a reliable policy management framework in cloud computing environments. In: IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), pp. 137–144 (2016)
11. Fatemi Moghaddam, F.: Multi-layered policy generation and management in clouds. University of Göttingen (2018)
12. Hajivali, M., Fatemi Moghaddam, F., Alrashdan, M.T., Alothmani, A.Z.M.: Applying an agent-based user authentication and access control model for cloud servers. In: International Conference on ICT Convergence (ICTC), pp. 807–812 (2013)
13. Bajaj, S., Box, D., Chappell, D., Curbera, F., Daniels, G., Hallam-Baker, P., Hondo, M., Kaler, C., Langworthy, D., Malhotra, A.: Web Services Policy Framework (WS-Policy). Specif. IBM, BEA, Microsoft, SAP AG, Sonic Software, VeriSign (2004)