VOL. 10, NO. 11, JUNE 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

AN EFFICIENT POLICY BASED SECURITY MECHANISM USING HMAC TO DETECT AND PREVENT UNAUTHORIZED ACCESS IN CLOUD TRANSACTIONS

Judy Jenita S., Justin Samuel S., Abirami S. and R. S. Shalini
Department of Information Technology, Faculty of Computing, Sathyabama University, Chennai, India
E-Mail: drsjustin@gmail.com

ABSTRACT

A cloud computing paradigm accommodates a large number of remotely located servers networked together, by providing access to a centralized resource to all the entities participating within a cloud transaction. Whereas, the virtual cloud is a recent trend in cloud computing in which multiple third party vendors renting a virtual space thereby improving the virtual memory space to accommodate a wide range of resources. In this paper, we propose a secured cloud infrastructure with *HMAC* authentication and policy fixation for individual users. Also, multiple transactions executing on a cloud server is administered by a centrally located transaction manager which deals with the policy fixation engagements to different users participating in that particular transaction. The proofs of authorisations are evaluated for each participant to facilitate the concept of safe and trusted cloud transactions. Policy violations occurring within the cloud a server is termed as policy inconsistency updates which is overcome by the proposed *HMAC* authentication algorithm. Transactions are either committed or aborted with the permission from all the participating cloud servers within certain time periods provided by the application of a Two phase validation commit protocol. Experimental results show a greater improvement in the security of the system using *HMAC*. The outcome of this work shows notable improvement in the security level of transactions.

Keywords: cloud transactions, cloud access policies, cloud security, HMAC.

I. INTRODUCTION

Cloud computing background consists of servers that are remotely located in large groups that are networked for sharing data-processing and to obtain access to the networked service and resources. Many users share the cloud resources which are dynamically reallocated according to the request. The speciality in cloud computing is, a single web server can be accessed by multi-users and data processing can be done without getting licenses for the applications. Precisely, a cloud environment can render infrastructure, platform, software and communication as services provided by multiple vendors. A few multiplicities of cloud models are known as Platform as a service (Paas), Data as a Service (DaaS), Infra-Structure as a Service (IaaS) and Software as a service (SaaS). In IaaS, the vendors afford virtual as well as physical machines with other resources like virtual disk library with images, raw storage block, etc. The resources are retrieved from various data sources as per the user demand. Cloud users can download the images, videos files and executable software applications over cloud infrastructure. The service providers charge the users based on their usage and consumption of resources. Similarly SaaS, also known as software on demand, can be utilized by cloud users when they are inadequate in establishing the requisite infrastructure and application platforms. In order to meet their demands the virtual machines are cloned at the end time. To balance the loads on the cloud servers, all the tasks are split and distributed over multiple virtual machines. This process is hidden from the users. Users perceive that they have only a single entry point to access the cloud services.

Additionally, the applications offered by cloud are multi-tenant which can be served to multiple organizations simultaneously. A centrally hosted application updates its data resources time to time and this can be performed without consuming any software installations on the client end. Due to its elasticity, users' data in the cloud are more susceptible to unauthorized access. Hence, there are many mechanisms available to protect the privacy as well as the content of servers. Virtualization is the technique which allows a physical computing, device copied into multiple devices in such a way that each such device can perform all computing tasks as the source device. Usage of virtual tools facilitates the implementation of a cloud model with available open sources. Xampp [1] is open source webserver package containing Apache server, Interpreters and MySql. Number of occurrences of XAMPP can be generated in a single system which is self confined, and they can copied to other destinations. EyeOS is another web application which is used to establish communication and collaboration among different users. It provides a web cloud desktop with a unique user interface and provides full support for cloud operations to manage files, tools and to integrate client applications.

An interesting aspect of the cloud is its elasticity, making it favourable for highly scalable and multi tiered applications. In data replication, multiple copies of original data is duplicated and distributed over networked servers [2]. This offers the facility of scalability and elasticity in cloud. This conveys an account of consistency when the

data is propagated throughout the system. Inconsistencies arise in a database when the single data is observed at multiple sites. There are two types of security inconsistency problems, policy inconsistency in which several versions of a single policy are witnessed at multiple sites within a lifetime of a single transaction leading to a set of inconsistent access difficulties. Secondly, the system may suffer from credential inconsistencies, where a user's login credentials are revoked by the authority or a transaction manager before the transactions commits or rollback. Paper [3] illustrates a mechanism which sense the attacks based on certified characteristics over a multi-tenant cloud model. We introduce the concept of trusted transactions which do not cause policy or credential inconsistencies and at the same time to conform to the ACID properties of a transaction. A two phase validate and two phase validation commit protocol can be suggested to ensure the safety of a transaction by examining the policy, credential and data consistency during the period of a transaction. Considerably, the use of log records in a cloud architecture increases data storage to a large extent [4]. Furthermore, developments may lead to workload balancing between multiple servers participating within a cloud architecture. When a heap of transaction requests is made by a transaction manager through the same cloud server the workload increases on that particular server. This may lead to deadlocks in the system and causes incomplete transactions and rollbacks in the progress of a transaction. Hence it becomes necessary to balance the workload among the multiple servers present in a cloud environment. This results in an efficient distributed transaction system over a cloud environment.

2. RELATED STUDY

Due to the elasticity and relaxed consistency of cloud model a variety of related enhancements can be possibly done in the near future. Outsourcing of cloud resources serves as a major threat to its security considerably due to the potential loop holes in the system. Replicated data stored at remote sites are provided to clients. In the related works involving such cases, data replication process can be done in accordance with the proofs of retrievability in order to provide data integrity, policy and credential consistency to the users. The transaction manager maintains a master policy which contains documents related to the user accesses and authorization policies [5,14]. A recent work provides a certain guarantee level for the interaction between the data and set of policies [6], which assures that the server side policies maps the data stored in the server. Williams *et al*. [7] introduces a technique by which cloud servers can handle encrypted reads, writes, and inserts so that the third party vendors support transaction serialization, backup, and recovery to ensure data confidentiality and correctness of user access patterns. Further, the extension of the system by using HMAC authentication strengthens the security of the virtual cloud architecture [10]. According to [11] the recent research works in the virtual cloud security probably falls into two contrasting categories. It involves the process of determining the security of cloud storage and cloud computation. In [12], authors have proposed a protocol called the Orthogonal Handshaking Authentication Protocol for handling transactions in cloud. This indicates that future enhancements can implement hand shaking mechanisms for executing the transactions in a virtual cloud.

3. ARCHITECTURE OF THE SYSTEM

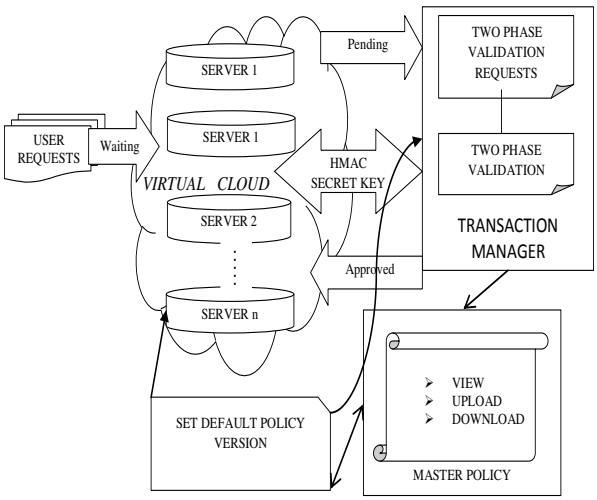

Figure-1. Transaction processing with HMAC authentication

The above Figure illustrates the working of multiple cloud servers enclosed in a runtime environment. Pile of user requests are sent to the transaction manager for authorization and set a default policy for each server. Using *hmac,* a secret key is generated in the transaction manager. The master policy associated with the transaction manager encloses all the policy versions. A virtual run time cloud contains many servers participating in multiple transactions. Users can access the cloud services through a Transaction Manager (TM) which manages the servers. A master policy is maintained by the TM. At regular intervals the existing policies in the cloud servers are replaced by newer versions. It includes view, update, delete and login policies. When an adversary tries to modify the data in the server a policy update request is submitted to the TM.A hash message authentication code is used to ensure the authenticity of the database.

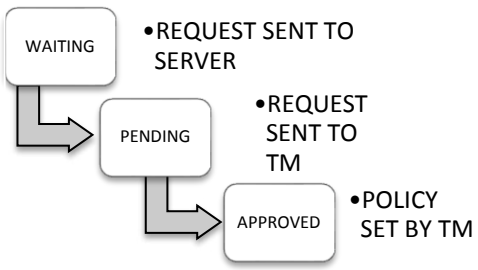

Figure-2. Various states of user's transaction request.

3.1. Policy updates

The relation $t = T_s(1 - P_u) + Tl P_u$, represents the average execution time of a transaction, where $P_u$is the probability function of a policy update. Given, (T_s) represents the average commit time of a short transaction with no policy changes and (T_l) represents the average commit time of a long transaction, where the policy updations, force the proofs of authorizations to be revaluated multiple number of times [1].

3.2. Trusted transactions

We present a concept of trusted transactions which do not violate both the policy and credential consistencies during the lifetime of a transaction. [13]. Given a transaction T and its view $V(T)$ where $T = (q1, q2, q3, \ldots., q_n)$, represents the number of queries evaluated within that particular transaction. Tcan be called as a trusted transaction if, at time*t:* $\alpha(T) \leq t \leq \beta(T) \wedge (\Phi - Consistent\ (V(T)) \vee (\psi - Consistent\ (V(T)))$.

3.3. Proof of authorizations

The relation $P(A) = (q_n, s_n, P_{s_n}(d(q_n)), t_n, c)$ denotes a proof of authorization that is being evaluated at the server $s_n$where $q_n$is the query to be evaluated at that server at time t. Ps_n denotes the proofs of authorization enforced by the server s_n, d indicates the set of data items being that are included in the query q_n. All the proofs of authorizations are evaluated at the time t_n, and finally c is a set of credentials that are applied by the query processor to complete all the existing proofs of authorization.

3.3.1. Authorization mechanisms

I) Deferred proofs of authorisation: In a transaction T with its view V(T), all the available proof of authorisations are evaluated only during the commit time to decide whether the transaction is a trusted one. In case weak authorisations the deferred proof of authorisation mechanism provides a positive approach.

II) Punctual proofs of authorisation: In a transaction T with its view V (T), all the available proof of authorisations are evaluated at once a query processing is initiated in the cloud server. At commit time, the proofs are again evaluated which makes it easy to detect the unsafe and complicated transactions at its early stage of execution. This reduces the amount of consistency in the cloud servers as they may falsely block access to valid data. Hence we need some obstructive approaches to enforce consistency among the participants.

III) Incremental punctual proofs of authorisation: A view instance V→T can be defined as a subset of all the proofs of authorisations evaluated by the cloud servers involved in a transaction T at time t. With the incremental proofs of authorisation a transaction T is highly trusted since it is not allowed to commit until the server achieves the specified level of policy consistency with all other participating servers. By applying view consistent model, all the servers will be consistent at the commit time.

IV) Continuous proofs of authorization: In a transaction T with its view V(T), all the available proof of authorisations are evaluated on accordance to the previous proofs if any policy change is encountered at the server entities. It comprehends two cases:

a) All servers are expected to update their policy versions in consistent with the new set of policies.
b) Re-evaluate all the existing proofs.

Table-1. Comparison table between the proofs of authorizations

Category	Deferred	Punctual	Incremental	Continuous
Efficiency	High	High	Moderate	Moderate
Performance	High	High	Low	Low
Accuracy	Moderate	Moderate	High	Moderate
Precision	High	Moderate	Moderate	Low

The above table illustrates a comparison between the different proofs of authorization. The deferred proofs may be less accurate when compared to other approaches, because while executing the deferred proofs the policies used by the transaction manager are not updated during regular intervals. Whereas, the punctual proofs of authorizations are monitored locally throughout a transaction. Continuous and Incremental proofs of authorizations provide a low performance in comparison with the other approaches, especially when the transaction manager frequently updates the policies present in the

master policy records. However, the deferred proofs provide a much better performance overall.

Algorithm 1: Two phase validation commit protocol (TM)

Prepare to Validate ()
{
Send validation request to servers;
Select latest version of a policy;
If(reply=False)
{ Abort transaction; }
Else
Call Prepare to commit ( )
{
Send commit request to servers;
If (all servers accept the policy version){

For each unique policy
{
If (reply=False)
Call Abort () ;}
Else
{
Commit transaction ();
}
}}
For each server with old policies
{
Send (update message);
Wait time for reply from all servers;
End;
}}

In the two phase validation commit algorithm, a validation request is sent to all the server participants. If all the participants reply 'yes' the transaction manager sends a commit request, based on the latest policy versions. For each policy if the participants reply 'yes' then commit the transaction otherwise the transaction is aborted. The policy versions are updated in all the servers during an update request from the transaction manager.

Algorithm 2: Hash message authentication algorithm

Procedure (HMAC (key, text))
{
while key=b, then k_o =key;
//key is the secret key
// b is the hash value of key
IF key>b then key=H(k) ;
//H is the hash function
IF key<b then zeros appended after k (k_o);
}
do
{
Perform $k_o \oplus$ipad;
Append, $k_o \oplus$ipad || m;
//m is the message or text to be authenticated
By applying H perform ipad,
Compute, H (($k_o \oplus$ipad)|| m)
Perform, $k_o \oplus$opad;
Append, ($k_o \oplus$opad) || H (($k_o \oplus$ipad) || text)
By applying the value of H Calculate,
H (($k_o \oplus$opad) || H(($k_o \oplus$ipad) || text))
Select 't' bytes of result as HMAC
}
}

4. RESULTS AND DISCUSSIONS

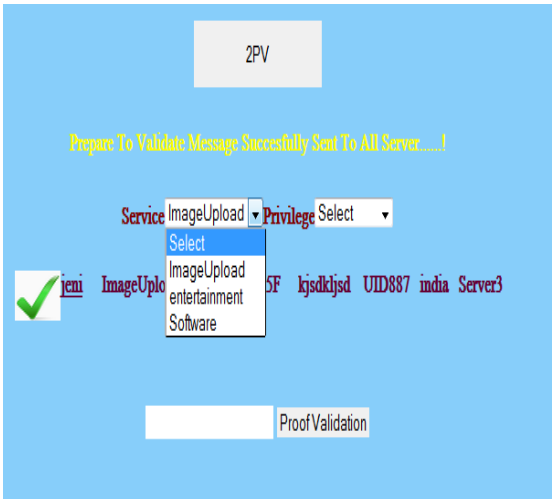

Figure-3. Assigning services to users.

The transaction manager monitors the servers on a timely basis during which it checks the policy versions at each server participant. If any discrepancy is detected in the policy versions, it updates the hacked server with the newest policy versions held by the master policy. The possibility of committing a transaction varies as the policy update probability changes. If the user request violates any of the policies then the transaction is prohibited and a rollback occurs. The master policy associated with the transaction manager stores the latest policy details from where the data is retrieved for updation. This is done intermittently to maintain security and consistency to the users. Before assigning a default policy to a user their proofs of authorization are validated by the transaction manager in three different stages.

During this process, a specific service is allocated to the user based on their request.A unique user ID is automatically generated to all users.

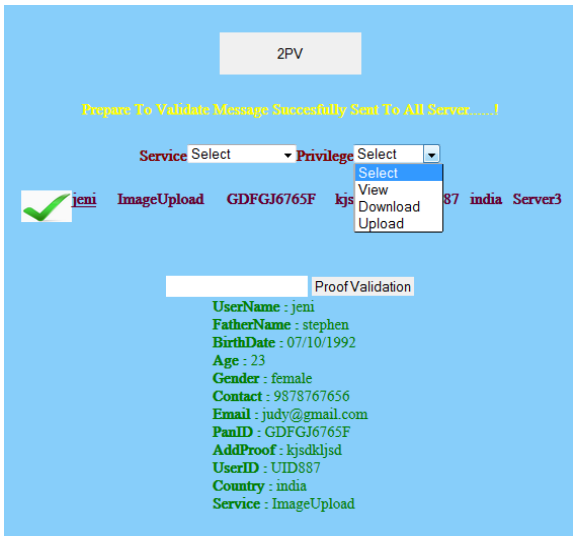

Figure-4. Assigning privilege and default policy to users.

In the above fFigure, allocation of services to the users are depicted. The transaction maanger sends a validation request to all the participating servers, to validate the proofs of authorization of each user.

Once the validation of user data is completed the transaction manager assigns an initial privilege to the user which includes viewing, uploading and downloading of resources in the cloud. The transaction manager then sends a commit request to all the server participants. As a result, a default policy is set for that particular user and a unique policy version is updated in the server.

5. PERFORMANCE ANALYSIS

Experimental studies show the overall performance of the proposed method in accordance with total number of user requests made. Without applying HMAC authentication algorithm the system suffers from inconsistencies due to the transactions abort or rollback at commit time. But, with the application of HMAC authentication code the performance of the system increases with a majority of transactions committing at the run time. While the existing system is more vulnerable to security attacks, the proposed system with HMAC overcomes the system's security loop holes by providing a concept of trusted transactions.

The graph shows a comparative study of transactions failure rate with the proposed method over the existing method. In this case the Figure explains how the unauthorized user requests are filtered. For example, out of 800 requests 660 trusted transactions fail in the existing method.

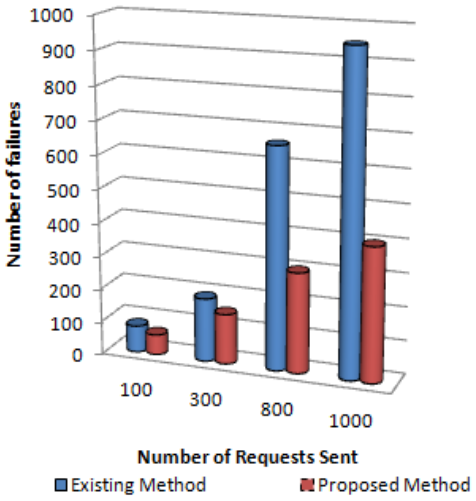

Figure-5. Ratio of trusted transactions failure.

The graph shows a comparative study of transactions failure rate with the proposed method over the existing method. In this case the figure explains how the unauthorized user requests are filtered and 300 trusted transactions fail in the proposed method. Thus it proves the efficiency of the proposed method.

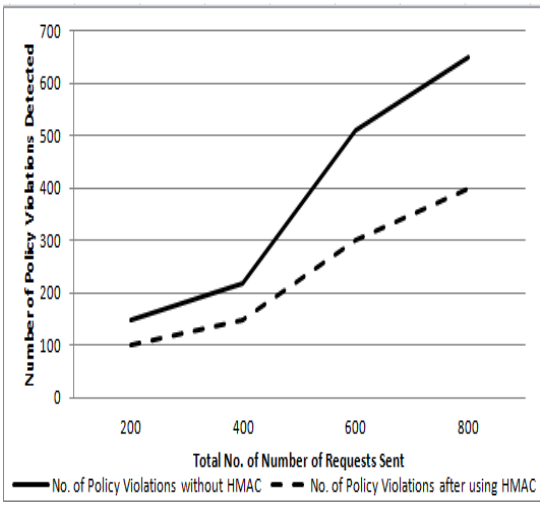

Figure-6. A comparison graph for detection of policy violations.

The graph represents the number of policy violations detected in the system with and without using the HMAC authentication. The transaction manager updates the policies to each user. As given in Figure-4 each user is assigned a unique policy while registering to the server and a unique policy version is updated in the server. If any user violates the policies assigned to them, our proposed HMAC algorithm will detect and prevent the policy violations from affecting the efficiency of the system. For example, out of 600 user requests 500 requests undergo policy violations in

the existing method and only 300 requests undergo policy violations in the proposed method.

6. CONCLUSIONS

The services in cloud are widely adopted by many organisations for the purpose of resource sharing. Even though they are popular, the vendors of cloud services generally lack security as well as consistency in policy and data storage. All these consistency problems arising due to weak consistency models in the cloud hosted transactions are identified in this paper. The user access mechanism is controlled by the policy-based authorization systems stored in the master policy. To overcome the inconsistent policies, we have developed a proof based authorization and consistency models such as the Deferred and Punctual consistency models, Whereas, the Incremental and Continuous models can apply increasingly strong protections with minimal runtime outlays. Also, we have used simulated jobs to experimentally evaluate the operations of our projected consistency models relative to performance of transaction processing, security and authentication. Furthermore, the implementation of hash message authentication code generates a secret key between the transaction manager and the cloud servers, such that any violations in the server policies is notified to the transaction manager which updates the policy versions enhancing the security of the proposed system. Results show that the failure rates of trusted transactions are reduced in the proposed system in comparison with the existing system. Moreover, with the proposed method users violating the assigned policies to access the resources are identified and prevented. Hence, the security level of the trusted transactions and the overall efficiency of the system is considerably improved.

REFERENCES

[1] Iskander M., *et al*. 2014. Balancing Performance, Accuracy, and Precision for Secure Cloud Transactions. IEEE Transactions on Parallel and Distributed Systems. 25(2): 417-426.

[2] Wang Boyang, Baochun Li and Hui Li. 2012. Oruta: Privacy-preserving public auditing for shared data in the cloud. Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. IEEE.

[3] Varadharajan Vijay and UdayaTupakula. 2014. Counteracting security attacks in virtual machines in the cloud using property based attestation. Journal of Network and Computer Applications. 40: 31-45.

[4] Ray Indrajit, *et al*. 2013. Secure Logging as a Service Delegating Log Management to the Cloud. IEEE systems journal. 7: 323-334.

[5] Justin Samuel S, Koundinya RVP, Kotha Sashidhar, 2015. Service Oriented Secured Privacy Enhancement for Health Care Applications. International Journal of Applied Engineering Research. 10(3): 6207-6216.

[6] Hsiao, Hung-Chang, *et al*. 2013. Load rebalancing for distributed file systems in clouds. Parallel and Distributed Systems, IEEE Transactions on 24.5. 951-962.

[7] Williams, Peter, RaduSion, and Dennis Shasha. 2009. The Blind Stone Tablet: Outsourcing Durability to Un trusted Parties. NDSS.

[8] http://csrc.nist.gov/publications/fips/dfips-HMAC.pdf.

[9] http://en.wikipedia.org/wiki/Hashbased_message_authentication_code

[10] Arasu S., Ezhil B., Gowri and S. Ananthi. 2013. Privacy-Preserving Public Auditing In Cloud Using HMAC Algorithm. International Journal of Recent Technology and Engineering. 2(1).

[11] Wei Lifei, *et al*. 2014. Security and privacy for storage and computation in cloud computing. Information Sciences. 258: 371-386.

[12] Mohammed M, Subramanian. 2014. Enhancement of the Private Cloud Data Transaction by using an Orthogonal Handshaking Authentication Protocol (OHSAP). International Journal of computer Applications. 96(23).

[13] M. K. Iskander, D. W. Wilkinson, A. J. Lee and P. K. Chrysanthis. 2011. Enforcing policy and data consistency of cloud transactions. In Proceedings of the Second International Workshop on Security and Privacy in Cloud Computing, ser. ICDCS-SPCC 2011. Washington, DC, USA: IEEE Computer Society.

[14] Wobber Ted, Thomas L. Rodeheffer and Douglas B. Terry. 2010. Policy based access control for weakly consistent replication. Proceedings of the 5th European conference on Computer systems. ACM.