ENSURING DISTRIBUTED ACCOUNTABILITYFOR DATA SHARING IN THE CLOUD

[1]Anoop Kumar Gupta,[2]Dhaval Vaghani,[3]Azhar Imam,
[4]Mrs. Purnima S.Mittalkod, Assoc Prof.ISE Dept, Global Academy of Technology, Bangalore
[5]Dr. S. Vagdevi,Prof & HOD,ISE Dept, Global Academy of Technology, Bangalore

Abstract- Cloud computing enables highly scalable services to be easily consumed over the Internet on an as-needed basis. A major feature of the cloud services is that users'data are usually processed remotely in unknown machines that users do not own or operate. While enjoying the convenience brought by this new emerging technology, users' fears of losing control of their own data (particularly, financial and health data) can become a significant barrier to the wide adoption of cloud services[1-3]. To address this problem, here, we propose a novel highly decentralized information accountability framework to keep track of the actual usage of the users' data in the cloud. In particular, we propose an object-centered approach that enables enclosing our logging mechanism together with users' data and policies. We leverage the JAR programmable capabilities to both create a dynamic and traveling object, and to ensure that any access to users' data will trigger authentication and automated logging local to the JARs. To strengthen user's control, we also provide distributed auditing mechanisms. We provide extensive experimental studies that demonstrate the efficiency and effectiveness of the proposed approaches. we categorized this paper into five sections. In the first section introduction ,second and third gives overview of architecture & problem definition, fourth section algorithm, finally we discuss future work and conclusion.

Key words: JAR files, Cloud Information Accountability (CIA), Cloud Service Provider(CSP)

1 INTRODUCTION

To allay users' concerns, it is essential to provide an effective mechanism for users to monitor the usage of their data in the cloud. For example, users need to be able to ensure that their data are handled according to the servicelevel agreements made at the time they sign on for services in the cloud. Conventional access control approaches developed for closed domains such as databases and operating systems, or approaches using a centralized server in distributed environments, are not suitable, due to the following features characterizing cloud environments [4-6].
Data handling in cloud can be outsourced by Cloud Service Provider(CSP) to other entities in the cloud and other entities can also delegate tasks to others and so on. Data handling in the cloud goes through a complex and dynamic hierarchical service chain.

We propose a novel approach, namely Cloud Information Accountability (CIA) framework, based on the notion of information accountability. The design of the CIA framework presents substantial challenges, including uniquely identifying CSPs, ensuring the reliability of the log, adapting to a highly decentralized infrastructure, etc. Our basic approach toward addressing these issues is to leverage and extend the programmable capability of JAR (Java ARchives) files to automatically log the usage of the users' data by any entity in the cloud. Users will send their data along with any policies such as access control policies and logging policies that they want to enforce, enclosed in JAR files, to cloud service providers. Any access to the data will trigger an automated and authenticated logging mechanism local to the JARs. We refer to this type of enforcement as "strong binding" since the policies and the logging mechanism travel with the data. This strong binding exists even when copies of the JARs are created; thus, the user will have control over his data at any location. Such decentralized logging mechanism meets the dynamic nature of the cloud but also imposes challenges on ensuring the integrity of the logging. To cope with this issue, we provide the JARs with a central point of contact which forms a link between them and the user. It records the error correction information sent by the JARs, which allows it to monitor the loss of any logs from any of the JARs. Moreover, if a JAR is not able to contact its central point, any access to its enclosed data will be denied. In summary, our main contributions are as follows:

IJERT

.We propose a novel automatic and enforceable logging mechanism in the cloud. To our knowledge, this is the first time a systematic approach to data accountability through the novel usage of JAR files is proposed.

.Our proposed architecture is platform independent and highly decentralized, in that it does not require any dedicated authentication or storage system in place.

.We go beyond traditional access control in that we provide a certain degree of usage control for the protected data after these are delivered to the receiver.

. We also provide a detailed security analysis and discuss the reliability
and strength of our architecture.

2CLOUD NFORMATION ACCOUNTABILITY ARCHITECURE

1.Cloud Information Accountability (CIA) Framework:

CIA framework lies in its ability of maintaining lightweight and powerful accountability that combines aspects of access control, usage control and authentication. By means of the CIA, data owners can track not only whether or not the service-level agreements are being honored, but also enforce access and usage control rules as needed.

2. Distinct mode for auditing:

Push mode: The push mode refers to logs being periodically sent to the data owner or stakeholder.

Pull mode: Pull mode refers to an alternative approach whereby the user
(Or another authorized party) can retrieve the logs as needed.

3. Major components of CIA:

There are two major components of the CIA, the first being the logger, and the second being the log harmonizer.The logger is strongly coupled with user's data (either single or multiple data items). Its main tasks include automatically logging access to data items that it contains, encrypting the log record using the public key of the content owner, and periodically sending them to the log harmonizer. It may also be configured to ensure that access and usage control policies associated with the data are honored. For example, a data owner can specify that user X is only allowed to view but not to modify the data. The logger will control the data access even after it is downloaded by user X. The log harmonizer forms the central component which allows the user access to the log files. The log harmonizer is responsible for auditing.

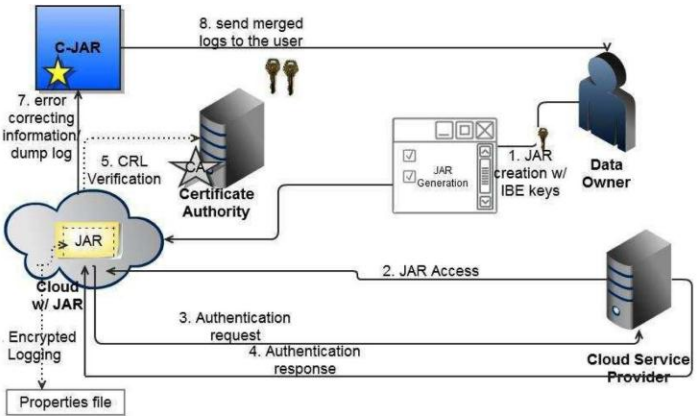

FIG:2.1 Overview of cloud information accountability

3 PROBLEM STATEMENT

While working with cloud users' fears of losing control of their own data can become a significant barrier to the wide adoption of cloud. In order to track the actual usage of the data[7], we aim to develop novel logging and auditing techniques which satisfy the following requirements:

1. The logging should be decentralized in order to adapt to the dynamic nature of the cloud. More specifically, log files should be tightly bounded with the corresponding data being controlled, and require minimal infrastructural support from any server.

2. Every access to the user's data should be correctly and automatically logged. This requires integrated techniques to authenticate the entity who accesses the data, verify, and record the actual operations on the data as well as the time that the data have been accessed.

3. Log files should be reliable and tamper proof to avoid illegal insertion, deletion, and modification by malicious parties. Recovery mechanisms are also desirable to restore damaged log files caused by technical problems.

4. Log files should be sent back to their data owners periodically to inform them of the current usage of their data. More importantly, log files should be retrievable anytime by their data owners when

IJERT

needed regardless the location where the files are stored.

5.The proposed technique should not intrusively monitor data recipients' systems, nor it should introduce heavy communication and computation overhead, which otherwise will hinder its feasibility and adoption in practice.

4 AUTOMATED LOGGING MECHANISM

In this we elaborate on the automated logging mechanism.

4.1 THE LOGGER STRUCTURE

We leverage the programmable capability of JARs to conductautomated logging. A logger component is a nested Java JARfile which stores a user's data items and corresponding logfiles. As shown in Fig. 4.1, our proposed JAR file consists of oneouter JAR enclosing one or more inner JARs.

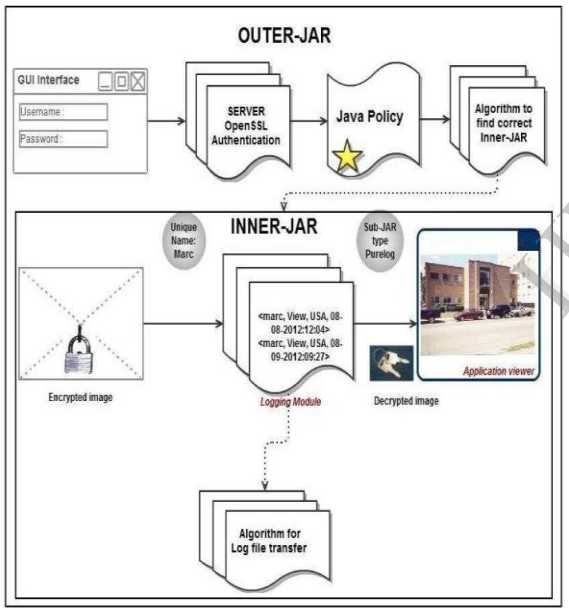

Fig 4.1 JAR enclosing one or more inner JARs

The main responsibility of the outer JAR is to handle authentication of entities which want to access the data stored in the JAR file. In our context, the data owners may not know the exact CSPs that are going to handle the data.

Each inner JAR contains the encrypted data, class files to facilitate retrieval of log files and display enclosed data in a suitable format, and a log file for each encrypted item. We support two options:

1. PureLog. Its main task is to record every access to the data. The log files are used for pure auditing purpose.

2. AccessLog. It has two functions: logging actions and enforcing access control. In case an access request is denied, the JAR will record the time when the request is made. If the access request is granted, the JAR will additionally record the access information along wit the duration for which the access is allowed.

4.2 LOG RECORD GENERATION

Log records are generated by the logger component. Logging occurs at any access to the data in the JAR, and new log entries are appended sequentially, in order of creation LR =(r_1; . . . ; r_k) Each record r_i is encrypted individually and appended to the log file. In particular, a log record takes the following form:

r_i=(ID,Act,T,Loc,h((ID,Act,T,Loc)|r_i-1|…|r_1),sig).

Here, r_i indicates that an entity identified by I D has performed an action Act on the user's data at time T at location Loc. The component h(ID,Act,T,Loc|r_i - 1| . . . |r_1) corresponds to the checksum of the records preceding the newly inserted one, concatenated with the main content of the record itself (we use I to denote concatenation).

In the current system we support four types of actions i.e.,Act has one of the following four values:view,download,timed_access and location based access.

.View:

The entity can only read the data but is not alloed to save a raw copy of it anywhere permanently.

.Download

The entity is allowed to save a raw copy of the data and the entity will have no control over this copy neither log records regarding access to the copy.

.Timed_access

This action is combined with the the view only access and it indicates that the data are made available only for a certain period of time

.Location based access

In this case the pure log will record the location of the entities.the access log will verify the location of the entities.

4.3 ALGORITHM:

1. Suppose TS(NTP) be the network time protocol timestamp
2. Command from data owner is not received
3. rec=UID,OID,AccessType,Result,Time,Loc
4. Curtime := TS(NTP)
5. Log := log + ENCRYPT(rec) //ENCRYPT is the encryption function used to encrypt the record.
6. send a PING to the harmonizer to check if it is alive.
7. If PING–CJAR then
8. PUSH RS(rec) //write the error correcting bits
9. Else
10. EXIT(1) // error if no PING is received.
11. end if
12. write the log file to the harmonizer so that it passes to owner
13. RS(log) := NULL //reset the error correction records
14. tbeg := TS(NTP) //reset the tbeg correction

5 CONCLUSION & FUTURE RESEARCH

We proposed innovative approaches for automatically logging any access to the data in the cloud together with an auditing mechanism. Our approach allows the data owner to not only audit his content but also enforce strong back-end protection if needed. Moreover, one of the main features of our work is that it enables the data owner to audit even those copies of its data that were made without his knowledge. we plan to design a comprehensive and more generic object-oriented approach to facilitate autonomous protection of traveling content. We would like to support a variety of security policies, like indexing policies for text files, usage control for executables, and generic accountability and provenance controls.

6 REFERENCES

1. P. Ammann and S. Jajodia, "Distributed Timestamp Generation in Planar Lattice Networks," ACM Trans. Computer Systems, vol. 11, pp. 205-225, Aug. 1993.

2.G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proc. ACM Conf. Computer and Comm. Security, pp. 598- 609, 2007.

3. E. Barka and A. Lakas, "Integrating Usage Control with SIP-Based Communications," J. Computer Systems, Networks, and Comm., vol. 2008, pp. 1-8, 2008.

4. D. Boneh and M.K. Franklin, "Identity-Based Encryption from the Weil Pairing," Proc. Int'l Cryptology Conf. Advances in Cryptology, pp. 213-229, 2001.

5. R. Bose and J. Frew, "Lineage Retrieval for Scientific Data Processing: A Survey," ACM Computing Surveys, vol. 37, pp. 1- 28, Mar. 2005.

6. P. Buneman, A. Chapman, and J. Cheney, "Provenance Management in Curated Databases," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '06), pp. 539-550, 2006.

7. B. Chun and A.C. Bavier, "Decentralized Trust Management and Accountability in Federated Systems," Proc. Ann. Hawaii Int'l Conf. System Sciences (HICSS), 2004.

8. R. Corin, S. Etalle, J.I. den Hartog, G. Lenzini, and I. Staicu, "A Logic for Auditing Accountability in Decentralized Systems," Proc. IFIP TC1 WG1.7 Workshop Formal Aspects in Security and Trust, pp. 187-201, 2005.

9. B. Crispo and G. Ruffo, "Reasoning about Accountability within Delegation," Proc. Third Int'l Conf. Information and Comm. Security(ICICS), pp. 251-260, 2001.

IJERT