Role and Access Based Data Segregator for Security of Big Data

Aayush Gupta
Department of Computer Science
NITK Surathkal
aayushgupta.nitk11@gmail.com

Ketan Pandhi
Department of Computer Science
NITK Surathkal
ketanpandhi13@gmail.com

P. Santhi Thilagam
Department of Computer Science
NITK Surathkal
santhi@nitk.ac.in

***Abstract*—In order to solve existing security problems of distributed networks, such as transmission, data storage, and data verification etc, a storage policy based on Hadoop has been proposed. It is meant from private cloud of the organization and allows the organization to store data ranging from confidential to public on the same cloud. The proposed methodology of cloud segregates the data on the basis of roles in hierarchy of organization having access to the data as well as how valuable the file is for organization and how many times the file has been accessed within a certain time period. The model is a combination of RBAC and Hadoop file distribution system.**

Keywords— RBAC, HDFS, Hadoop, Big-data, storage criticality

I. Introduction

Cloud computing is internet based development which provides IT-based services over the internet. Cloud computing is an extension of grid computing and distributed computing, which is a software concept[1]. It mainly relies on resource sharing which makes it highly scalable over large scale distributed system. Cloud computing is realized mainly through virtual technology, which can be divided into single virtualization and multiple virtualization [2]. The single virtualization makes a single machine to virtually work as multiple machines working together, such as VMware, while multiple machine virtualization links the machine through control center and makes them work like one machine. Hadoop is the relative example for multiple machine virtualization.

Because of rapid growth of cloud services[3][4][5], big data has become ubiquitous and crucial for numerous application domain, thus leading to significant challenges from point of view of data management perspective[6][7].

Cloud computing is evolving towards complex cloud ecosystem from a single private cloud, bringing in picture new scenarios such as cloud federations or multiple cloud. It has resulted in many new challenges in terms of risk, trust, eco-efficiency, security, cost and legal issues which influence their use from different perspectives which involve end users, Service Providers (SP), and Infrastructure Provider (IP)[8]. It is common for cloud storage systems to provide application-level security, in which components that authenticate and process user requests run with sufficient privileges to access any tenant's data; the code of each component is responsible for authorizing requests based on the requester's credentials. This architecture is used by OpenStack Swift [29] and other publicly available cloud storage systems. Further, logged user can store data in cloud without any additional storage media and can have access to the same data whenever and wherever. Cloud disk can make storage simple, fast and convenient but users are not allowed to upload confidential data because of security reasons. The security problems of cloud disk are not only traditional problems but also new problem in cloud computing[9]. Security challenges exist for three services of cloud SaaS, PaaS, IaaS. [10][11][12] have analyzed and summarized the threats being faced from different aspects. Cloud' storage weak security mainly occurs in following aspects:

1. Transmission security which says data can be intercepted while being transmitted in cloud and because of use of weak encryption protection, data can be intercepted.
2. Access control which says user data is stored in cloud without setting access authority, the user loses absolute right to monitor the data.
3. Data storage which says once data has been uploaded it is stored in a distributed manner and user is unaware of specific position of data. And confidential data and non-confidential data stored is not classified, it may cause data leakage.
4. Data verification which says cloud make no verification and inspection on the data uploaded. It can't guarantee that the uploaded data is corresponding to right user's data or the original data form the user.

Furthermore, we refer [13] for an overview and analysis of top ten big data security and privacy challenges. The scientific challenges faced by cloud computing security have been discussed in survey articles [14][15]. Going through all these security threats, we proposed a model for private cloud of an organization which stores the data by considering in role hierarchy of the organization and many other parameters. The section II of the paper discusses the related work done in this area. Section III carries on with the description of conceptual model describing all assumptions made and input required by the model. Section IV explains the algorithm proposed for storing data in cloud and Section V concludes the result and analysis of the implementation of model over different data sets. Section VI and section VII explains the conclusion and future work associated with the model.

II. RELATED WORK

Hadoop is an Apache open source project which consists of HDFS, MapRedcue, HBase, Hive, ZooKeeper and other projects [16]. Its main parts are HDFS and MapReduce. MapReduce aims at paralleling and dealing with tasks on a large scale, which would make the MapReduce scheduler become particularly important[17]. HDFS is an open source project of Google distributed file system(GFS)[18]. It has a high fault tolerance and certain data access control. We mainly use Hadoop's HDFS (Hadoop Distributed File System) for cloud storage. As Hadoop also lacks safety measures, Kerberos was integrated into the Hadoop in 2009 by yahoo. The user have to obtain access certification from the third party center for key issues before access to Hadoop cluster first, and it greatly reduced the risk of user's data leakages caused by identification. The paper [19] is looking at a cloud data storage preventing harvesting. The problem concerns the cloud customers, the cloud business centre which provides services, and the cloud data storage centre. Data stored in the data storage centre comes from a variety of customers and some of these customers may compete with each other in the market place or may own data which comprises confidential information about their own clients. Cloud staff have access to data in the data storage centre which could be used to steal identities or to compromise cloud customers. The paper proposes an efficient method of data storage which prevents staff from accessing data which can be abused as described above. In [20], Large Iterative Multiplier Ensemble (LIME) classifier model has been proposed for information security of Big Data. In this paper LIME classifier is a four-tier classifier for the detection of malware using Big Data. LIME classifiers are automatically generated as iterative multitier ensembles of ensembles. The paper [21] investigates an iterative hierarchical key exchange scheme for secure scheduling of big data applications in cloud computing. The privacy preservation over big data on cloud is considered in [22].
[23] proposed the use of the HDFS to build a private enterprise cloud, which combine the Hadoop's fault tolerance and suitable for big data attribute. In [24],SSL secure connection and secure virtual machine monitor are evaluated. [25] encrypted the cloud data using attribute encryption (ABE) scheme, using the property as a public key to encrypt the data before it is uploaded, this limits the data access user to have K attributes to decrypt the data, in which the K is the number of threshold to decrypt the data. This scheme ensured the safety of data storage, and at the same time, the server has no need to keep a public key for each user, the users' attributes are used to be the user's public key, but they could decrypt the data completely when different users hold their attributes together and get all attributes, [26] identifies the users with image processing methods, such as face recognition, fingerprint recognition, etc. And [27] made the transmission, encryption transmission and storage of the key files by Symmetric encryption and asymmetric encryption features complementary. [28] proposes a cloud disk storage based on Hadoop, the program draw lessons from Kerberos' authorization process. We utilize the classic algorithms such as AES, RSA to realize encryption, and authentication, and we also check the time to inspect if it can complete the encryption and transmission in an acceptable period. [30] proposes Secure Logical Isolation for Multi-tenancy (SLIM) model. SLIM adds an orthogonal tenant isolation mechanism over existing application-level security by leveraging the Linux process isolation mechanisms. SLIM enables resource pooling while decreasing the likelihood that a single vulnerability could jeopardize all tenants data.

III. CONCEPTUAL MODEL

We have proposed a model which aims segregating data in cloud into five different classes by evaluating storage criticality of each file. The model is mainly meant for unstructured data which may include text files, binary files, image, audio etc. The proposed model is meant for private cloud of an organization and the model requires the hierarchy model of the organization as well as the weight associated to each role in the hierarchy. For each file being stored following information is associated with it:

- Lowest role in the hierarchy which has access to the file
- External criticality of the file which shows how valuable the file is for organization.
- Number of times the file is being accessed in time period of 24 hours.

To evaluate the storage criticality(SC) of a file, we have following proportionalities of storage criticality with above mentioned inputs:

- Greater the value if external criticality (EC), more security the file needs as file is more valuable for the organization.

$$SC \propto EC$$

- All the roles lying above the lowest role having access to a file will also have access to the file. Thus, file needs less security in order to make the intra-organization access easier. Hence, SC is inversely proportional to the sum of role weights

(RW) of roles from the top of the hierarchy to the lowest role in hierarchy having access to the file.

$$SC \propto 1/\sum_{i=1}^{L} RW(i)$$

- More the access number (AN) of the file, more the security the file needs as it is being accessed frequently in a time period of 24 hours. Thus,

$$SC \propto AN$$

- To normalize the proportionality of SC with AN, we evaluate maximum of access number (MAN) and the relation between SC and MAN is as follows:

$$SC \propto 1/MAN$$

Combining the above four proportionalities, the final value of storage criticality (SC) is given as follows:

$$SC(x) = {EC(x)}/{SR(x)} + {NA(x)}/{MNA(x)}$$

$$\text{Where } SR(x) = \sum_{i=1}^{L} RW(i)$$

Let SC obtained by using above formula lie in range [0, N], and if we have K security classes, their boundary values will be as follows [0, N/K], (N/K, 2N/K], (2N/K, 3N/K], . . . , ((K-1)N/K, N]. However, number of files will be more in lower classes and less in higher classes, leading to relatively over-utilization of resources in lower classes and underutilization of resources in higher classes. Thus, to overcome this shortcoming we came up with the idea of normalizing the storage criticalities and we used cumulative probability distribution function for that. The normalization works as follows:

- For storage criticality (SC) = x, evaluate the total number of files with SC = x. Let N_x be the number of files and let N be the total number of files being stored in server.
- Find probability of SC=x.

$$P(x) = N_x/N$$

- Evaluate the cumulative probability for SC=x:

$$CP(x) = \sum_{i=0}^{x} P(i)$$

- Let K be number of storage classes to be build, thus replace SC=x with CP(x) * K.

Following the above algorithm, the new boundary value of storage criticalities of security classes are [0, 1], (1, 2],. . ., (k-1, k]. The algorithm further ensures evenly distribution of files in each security class thus leading to optimal utilization of resources.

IV. EXPERIMENTAL SETUP

For the experimental purpose we build our own name node having functionalities quiet similar to NameNode of HDFS. We performed experiments over data sets of 1000, 10,000 and 100,000 files where each file had following values of the above mentioned parameters:

- External Criticality (EC): [1, 1000]
- Number of Roles in Hierarchy model: 20
- Weight of each role (RW): [1, 100]
- Access Number for a file: [0, 2000]

Number of security class in which data to be stored was taken as 5. Thus name node had 5 servers, with each server having 1000 racks and each rack having 1000 data nodes. For experiment, a buffer was also required whose threshold capacity was one-tenth the total number of files present in the server. The experiment was done on a system with dual core processor with 2.2 GHz clock speed, 3GB RAM and 64-bit windows 7 OS.

Algorithm 1 is the algorithm used for normalizing the storage criticality using cumulative distribution function

Algorithm 1 Cumulative Distribution Normalization Function

```
file_dir ← Directory of files where key is file name and
        value is file's storage_criticality.
criticality_count ← Directory where key is a
        storage_criticality, say S and value is number of
        files having storage_criticaly equal to S.
ciritcs ← list to store all the storage_critcality value
N ← Total number of files in fiel_dir
for file in file_dir.keys() do
    if file_dir[file] not in keys of criticality_count.keys()
    then
        criticality_count[file_dir[file]]=1
    else:
        criticality_count[file_dir[file]]+=1
    end if
end for
for k in criticality_count.keys() do
    criticality_count[k]=criticality_count[k]/N
    critics.append(criticality_count[k])
end for
Sort(critics)
criticality_count[critics[0]] = criticality_count[critics[0]]
for i=1 to critics.length() – 1 do
  criticality_count[critics[i]] = criticality_count[critics[i]]
                  +criticality_counr[critics[i-1]]
end for
for k in criticality_count.keys() do
```

```
    criticality_count[k] = criticality_count[k]*5
end for
return criticality_count.
```

Algorithm 2 is used to evaluate the storage criticality and it calls algorithm 1 for normalizing the same.

Algorithm 2: Role and Access Based Data Segregation

```
f ← file to be stored in the server
file_dir ← directory holding name of each file and its
        normalized criticality and server allocated to it
external_critic ← external criticality of file f.
buffer_threshold ← threshold size of buffer
buffer_count ← Number of files in buffer
master_node ← masterNode of proposed file distribution
        system
if buffer_count < buffer_threshold
then
    server = (external_critic/500) mod 5
    master_node.add_file(f, server)
    buffer_count+=1
else
    normalized_files = cumulative_distribution_function
    for file in file_dir do
        if 0 ≤normalized_files[file ] ≤ 1
        then
            server=0
        else if 1 < normalize_files[file] ≤ 2
        then
            server=1
        else if 2 < normalize_files[file] ≤ 3
        then
            server=2
        else if 3< normalize_files[file] ≤ 4
        then
            server=3
        else if 4 < normalize_files[file] ≤ 5
        then
            server=4
        end if
        if file_dir[file].server!= server
        then
            master_node.shift_file(file, server)
        else
            file_dir[file].criticality =
normalized_files[file]
        end if
end for
```

V. RESULTS

We ran the algorithm 2 mentioned above on the data sets of 1000, 10,000 and 100,000 files and evaluated the number of files stored under 5 security classes prior to the normalization of storage criticality as well as after normalizing it. Following results were obtained:

Level of Security class	Lower boundary value of SC	Upper boundary value of SC	Number of files stored
Level V	>80	≤ 100	20
Level IV	>60	≤ 80	20
Level III	>40	≤ 60	19
Level II	>20	≤ 40	133
Level I	≥ 0	≤ 20	808

Table 1*File distribution for un-normalized SC for 1000 files data set*

Level of Security class	Lower boundary value of SC	Upper boundary value of SC	Number of files stored
Level V	>4	≤ 5	200
Level IV	>3	≤ 4	200
Level III	>2	≤ 3	202
Level II	>1	≤ 2	198
Level I	≥ 0	≤ 1	200

Table 2 *File distribution for normalized SC for 1000 files data set*

Level of Security class	Lower boundary value of SC	Upper boundary value of SC	Number of files stored
Level V	>80	≤ 100	230
Level IV	>60	≤ 80	216
Level III	>40	≤ 60	193
Level II	>20	≤ 40	1402
Level I	≥ 0	≤ 20	7959

Table 3 *File distribution for un-normalized SC for 10,000 files data set*

Level of Security class	Lower boundary value of SC	Upper boundary value of SC	Number of files stored
Level V	>4	≤ 5	2001
Level IV	>3	≤ 4	2000
Level III	>2	≤ 3	2000
Level II	>1	≤ 2	1999
Level I	≥ 0	≤ 1	2000

Table 4 *File distribution for normalized SC for 10,000 files data set*

Level of Security class	Lower boundary value of SC	Upper boundary value of SC	Number of files stored
Level V	>80	≤ 100	2006
Level IV	>60	≤ 80	1956
Level III	>40	≤ 60	1966
Level II	>20	≤ 40	13962
Level I	≥ 0	≤ 20	80110

Table 5 *File distribution for un-normalized SC for 100,000 files data set*

Level of Security class	Lower boundary value of SC	Upper boundary value of SC	Number of files stored
Level V	>4	≤ 5	20015
Level IV	>3	≤ 4	19982
Level III	>2	≤ 3	19981
Level II	>1	≤ 2	20011
Level I	≥ 0	≤ 1	20011

Table 6 *File distribution for normalized SC for 100,000 files data set*

Table 1, table 3, table 5 shows the file distribution under the servers belonging to different security classes when the storage criticality was not normalized and one can conclude that servers of security class I and security class II are over utilized.
Whereas the result of storage criticality normalization are visible in tables 2, 4 and 6. It ensures even distribution of files.

VI. CONCLUSION

Performing the experiments with different data sets following conclusion can be made:

- The normalization ensured evenly distribution of files over different security servers, thus resulting in efficient utilization of resources.
- Secondly, whenever the buffer gets full, the normalization function is called again, thus resulting in overhead of re-evaluation of storage criticality when the data set is quiet large.
- Thirdly, as access number of a file changes after every 24 hours, hence the storage criticality of the file also might change after 24 hours, resulting in dynamic storage of file, thus ensuring more security to the file system.

VII. FURTURE WORK

Future work for the model proposed is to go for its practical implementation in Hadoop Framework. Secondly, periodic normalization of storage criticality results in an overhead, to overcome that we will be experimenting with different values of threshold capacity of buffer so as to perform normalization for minimum number of times. Also, future work will involve to bring upon more parameters for evaluation of storage criticality which will make the formula more reliable and accurate. We can also look for optimization of MapReduce task so as to overcome the overhead induced.

VIII.REFERENCES

[1] HongBo Zhou. Cloud computing: technology, application, standar, Electronic Industry Press.2011

[2] http://en.wikipedia.org/wiki/Virtualization

[3] X. Liu et al., The Design of CloudWorkow Systems. New York, NY, USA: Springer-Verlag, 2012.

[4] L. Wang, R. Ranjan, J. Chen, and B. Benatallah, Cloud Computing: Methodology, System, and Applications. Boca Raton, FL, USA: CRC Press, 2011.

[5] X. Liu, J. Chen, and Y. Yang, Temporal QoS Management in Scientific Cloud Workflow Systems. Amsterdam, The Netherlands: Elsevier, 2012.

[6] J. Chen et al., "Big data challenge: A data management perspective," Frontiers Comput. Sci., vol. 7, pp. 157-164, Apr. 2013.

[7] L. Liu, "Computing infrastructure for big data processing,"' Frontiers Comput. Sci., vol. 7, no. 2, pp. 165-170, 2013.

[8] Secure Logical Isolation for Multi – tenancy in Cloud Storage, Michael Factor, Anner Hammama, Nadav Har'el, Elliot K. Kolodner, Anil Kurums, Alexandra Shulman-Peleg, Alessendro Sorniotti, IBM Haifa Research Lab.

[9] MUNIER M.Self-Protecting Documents for Cloud Storage Security[C].Trust, Security and Privacy in

Computing and Commu. Liverpool,2012: 1231-1238.

[10]SHAIKH F B. Security threats in cloud computing[C].Internet Technology and Secured Transactions (ICIT • DAbu Dhabi,2011: 214-219.

[11]“Security of Cloud Computing Providers Study,” Ponemon Institute, 2011.

[12]V. Winkler, “Securing the Cloud Computer: Security Techniques and Tactics,” Elsevier Inc., ISBN: 978-1-59749-592-9, 2011.

[13]S. Rajan et al. (2013, Oct. 12). Expanded Top Ten Big Data Security and Privacy Challenges. Cloud Security Alliance, Los Angeles, CA, USA [Online]. Available: http://cloudsecurityalliance.org/research/big-data/

[14]M. D. Ryan, ``Cloud computing security: The scientic challenge, and a survey of solutions,'' J. Syst. Softw., vol. 86, no. 9, pp. 2263-2268, 2013.

[15]C. M. Rong, S. T. Nguyen, and M. G. Jaatun, “Beyond lightning: A survey on security challenges in cloud computing,” Comput. Electr. Eng., vol. 39, no. 1, pp. 47-54, 2013.

[16]XU Guang-hui.Deploying and researching Hadoop in virtual machines[C]//Automation and Logistics (ICAL), 2012 IEEE International Conference on. Zhengzhou, 2012: 395-399.

[17]ZHUO Tang, ZHOU Jun-qing, LI Ken-li, et al. Contact Administrator; MTSD: A Task Scheduling Algorithm forMapReduce Base on Deadline Constraints [C]. Parallel and Distributed Processing Symposium Work.Shanghai, 2012: 2012-2018.

[18]Ghemawat S,Gobioff H,Leung S T.The Gooogle file systm[C].Proc of the 19th ACM Symp on Operating Systems Principles.New York:ACM,2003:29-43

[19]L. Batten, J. Abawajy, and R. Dose, ``Prevention of information harvesting in a cloud services environment,'' in Proc. 1st Int. Conf. Cloud Comput. Services Science, 2011, pp. 66-72.

[20]JEMAL H. ABAWAJY, ANDREI KELAREV, AND MORSHED CHOWDHURY, “Large Iterative Multitier Ensemble Classifiers for Security of Big Data”, IEEE transaction on emerging topics in computing, volume 2, no. 3, page 352-363, 2014

[21]C. Liu et al., “An iterative hierarchical key exchange scheme for secure scheduling of big data applications in cloud computing,” in Proc. 12th IEEE Int. Conf. Trust Security Privacy Comput. Commun., Melbourne, Australia, Jul. 2013, pp. 916.

[22]X. Zhang, C. Liu, S. Nepal, C. Yang, and J. Chen, “Privacy preservation over big data in cloud systems,” in Security, Privacy and Trust in Cloud Systems. Berlin, Germany: Springer-Verlag, 2013, pp. 239-257.

[23]ZHANG Da-wei. Research on hadoop-based enterprise file cloud storage system[C].Awareness Science and Technology (iCAST), 2011 3rd. Dalian, 2011: 434-437.

[24]Hou Qinhua,Wu Yongwei,Zheng Weimin etc.A Method on Protection of User Data Privacy in Cloud Storage Platform.Journal of Computer Research and Development, 2011,48（7）: 1146-1154

[25]YU Shu-cheng.Contact Administrator; Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing[C]//INFOCOM, 2010 Proceedings IEEE. San Diego, CA, 2010: 1-9.

[26]BAO Rong-chang. Access Security on Cloud Computing Implemented in Hadoop System[C].Genetic and Evolutionary Computing (ICGEC), 2011.

[27]WeiWei Zhang, Research of user data transmission and storage security in cloud computing slotuion:[Master degree theses].BEIJING : Beijing university of posts and telecommunications , 2011.

[28]A. Huang Jing, B. LI Renfa, and C. Tang Zhuo, “The Research of the Data Security for Cloud Disk Based on the Hadoop Framework,” Fourth International Conference on Intelligent Control and Information Processing (ICICIP, Beijing, China, 2013

[29]“Welcome to Swift’s documentation!” September, 2012. [Online]. Available: http://http://docs.openstack.org/developer/swift/

[30]Michael Factor, David Hadas, Aner Hamama, Nadav Har’el, Elliot K. Kolodner, Anil Kurmusy, Alexandra Shulman-Peleg, Alessandro Sorniotti, “Secure Logical Isolation for Multi-tenancy in Cloud Storage,” IBM Research Labs, Haifa and Zurich