Volume-4, Issue-7, July-2015 • ISSN No 2277 - 8160

Research Paper Engineering

Identification of Malicious Service Provider Through Clique Cover Algorithm in Hadoop Mapreduce

Dr. Sunita Varma Department of computer Application, SGSITS, Indore, India

Ms. Lata Mahawar Department of Computer Engineering SGSITS Indore, India

ABSTRACT

Cloud computing is a technological advancement that focuses on the way in which we design computing systems, develop applications and leverage existing services for building software resources are made available through the internet offered on a pay per use basis from cloud computing vendors. It provides three types of services which include Infrastructure as a Service(IaaS), platform as a Service(PaaS), Software as a Service(SaaS). Application service provider deliver their application by SaaS cloud systems via massive cloud computing infrastructure. SaaS clouds are vulnerable to malicious attack because cloud infrastructure has a sharing nature.

In this paper an algorithm is proposed for integrity attestation. Our approach provides a modified integrated attestation graph analysis scheme which uses constrained clique cover algorithm that can provide stronger attacker pinpointing power. This technique identify malicious service provider on the basis of size of clique. This approach also detect malicious service provider that perform partial misbehavior.

KEYWORDS : Application service providers (ASPs), virtual machine (VM), secure dataflow processing.

2. INTRODUCTION

Cloud computing introduce due to rapid growth of hardware, networking, middleware, and virtual machine technologies. Cloud computing is a model that provide different services on demand such as infrastructure as a service(IaaS), platform as a service(PaaS) and software as a service(SaaS) through Internet on pay per use basis. Cloud computing recently popular as a resource hosting platform which allows multiple users to share the common physical infrastructure. Software as a service clouds such as Google AppEngine and Amazon Web Service(AWS)[1],[2] build upon the concept of software as a service[3] and service oriented architecture[4],[5] which enable application service provider to deliver their services in cloud computing infrastructure[12]. Application service providers (ASPs) can lease a set of resources such as hardware, middleware from the cloud system to offer software as a service without maintaining their own computing infrastructures, on demand and pay per use basis[6]. Cloud systems are used for processing large amount of data which required large amount of resources. The research work focuses on dataflow processing systems that have many real world applications such as security surveillance and business intelligence. Users can feed data from various data sources into the cloud system to perform various data processing functions and receive final data processing results from the cloud.

However, cloud systems are often shared by multiple tenants either it is users or service providers that belong to different security domains, which make them vulnerable to various malicious attacks [7],[8]. The data processing services are taking large time, which provides more opportunities for attackers to exploit the system vulnerability and perform strategic colluding attacks. Although virtualization ensures certain isolation between users, malicious attackers can still leverage the shared hardware to launch attacks from the VMs they own or by compromising VMs of benign users. One of the top security concerns for cloud users is to verify the integrity of data processing results. Our research focuses on providing application-level attestation framework to dynamically verify the integrity of dataflow processing services provisioned through multi-party cloud computing infrastructures. We aim at achieving a practical integrity attestation technique for large-scale cloud infrastructures without requiring application modifications or assuming a trusted entity at third-party service provider sites.

3. RELATED WORK

Previous research work has provided various software integrity attestation solutions technique that require special third party and those techniques that does not require special trusted hardware or secure kernel support these techniques are Traditional Byzantine Fault Tolerance (BFT) technique, RunTest, AdapTest, IntTest, Hatman[9],[10],[11],[12],[13]. Traditional Byzantine fault tolerance (BFT) technique[9] detect malicious service provider by taking a result of all service provider and verify their result, this technique also called full-time majority voting (FTMV), but this technique produce high overhead to the cloud system. RunTest[10] is a runtime service integrity attestation scheme that uses a novel attestation graph model to capture attestation results among different cloud nodes. They design a clique based attestation graph analysis algorithm to pinpoint malicious service providers and recognize colluding attack patterns. This scheme can achieve runtime integrity attestation for cloud dataflow processing services using a small number of attestation data. AdapTest[11], a novel adaptive runtime service integrity attestation framework for large-scale cloud systems. AdapTest builds on top of previously developed system RunTest that performs randomized probabilistic attestation and employs a clique-based algorithm to pinpoint malicious nodes. However, randomized attestation still imposes significant overhead for high-throughput multi-hop data processing services. In contrast, AdapTest dynamically evaluates the trustiness of different services based on previous attestation results and adaptively selects attested services during attestation. Thus, AdapTest can significantly reduce the attestation overhead and shorten the detection delay and provide a novel adaptive multi-hop integrity attestation framework based on a new weighted attestation graph model. Derive both per-node trust scores and pair-wise trust scores to efficiently guide probabilistic attestation. In IntTest[12], a new integrated service integrity attestation framework for multitenant cloud systems. IntTest provides a practical service integrity attestation scheme that does not assume trusted entities on third-party service provisioning sites or require application modifications. IntTest builds upon our previous work RunTest and AdapTest but can provide stronger malicious attacker pinpointing power than RunTest and AdapTest. Specifically, both RunText and AdapTest as well as traditional majority voting schemes need to assume that benign service providers take majority in every service function. However, in large-scale multitenant cloud systems, multiple malicious attackers may launch colluding attacks on certain targeted service functions to invalidate the assumption. To address the challenge, IntTest takes a holistic approach by systematically examining both consistency and inconsistency relationships among different service providers within the entire cloud system. IntTest examines both per-function consistency graphs and the global inconsistency graph. The per-function consistency graph analysis can limit the scope of damage caused by colluding attackers, while the global inconsistency graph analysis can effectively expose those attackers that try to compromise many service functions. Hence, IntTest can still pinpoint malicious attackers even if they become majority for some service functions. By taking an integrated approach, IntTest can not only pinpoint attackers more efficiently but also can suppress aggressive attackers and limit the scope of the damage caused by colluding attacks. Moreover, IntTest provides result autocorrection that can automatically

replace corrupted data processing results produced by malicious attackers with good results produced by benign service providers. Hatman[13] uses EigenTrust values for identify the malicious service provider.

The proposed approach is a modification of IntTest, this approach only uses the clique cover algorithm, minimum size of clique is used to identify malicious service provider. This approach identify aggressive malicious service provider as well as partial malicious service provider.

4. PRELIMINARY

This section first describe multi tenant cloud infrastructure and describe the data processing in cloud. This section also describes service composition in cloud computing system.

3.1 Multi-Tenant Cloud Infrastructure

Cloud computing infrastructure shared between a multiple tenant, same resources are shared between the multiple tenants. Consider an example for resource leasing infrastructure is Amazon EC2[1] in the figure 1.

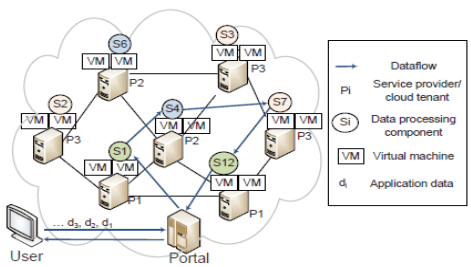

Figure 1: Cloud infrastructure shared by three tenant: p1, p2 and p3.

In EC2, cloud system lease a set of physical hosts running on virtual machine. The user can run application within the virtual machine and will be pay per use basis. Cloud computing provide a facility for user do not need to maintain their physical infrastructure and use virtual infrastructure with pay per use basis. Cloud computing also provides a platform for different service provider for deliver their services, service provider pi lease a resources from cloud infrastructure. Each service divided into service component di and these services are component store on a shared infrastructure. Whenever the user send the access request to cloud, the services divided into component and combine result back to user.

3.2 Dataflow Processing in Clouds

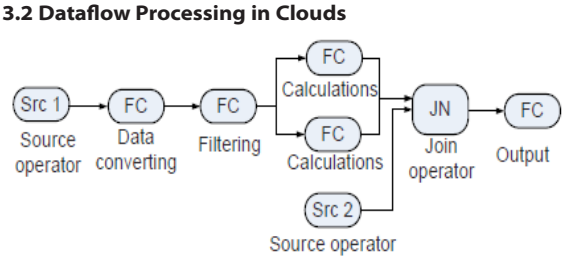

Figure 2: Dataflow processing service.

Data processing system e.g., Google's MapReduce , Yahoo's Hadoop, IBM System S, Microsoft Dryad have become increasingly popular with applications in many domains such as business intelligence, security surveillance, and scientific computing[10]. The research work focus on the data flow processing service, which provides more opportunity to attacker to exploit the system vulnerability. In this data processing system row data is read from the source and next step is to perform some conversion and calculation on row data and produce a final result to user. User requests the dataflow processing services through portal node. Portal node takes input from user and forward final result to user.

3.3 Service composition

Application service providers provide different service on demand pay per use basis. A single service composed from multiple individual service components [8][9]. For example, a disaster assistance claim processing application consists of voice-over-IP (VoIP) analysis component, e-mail analysis component, community discovery component, clustering and join components.

5. DESIGN AND ALGORITHM

In this section we describe the basics of integrity attestation technique: baseline attestation technique, assumptions. Later we describe the integrity attestation technique in detail.

4.1 Assumptions:

This approach first assumes that the total number of malicious service provider is less than the total number of legitimate service provider in the entire cloud system. Without this assumption it is very difficult for any integrity attestation technique to detect malicious service provider because it identify malicious service provider by using the results of legitimate service provider. Second assume that malicious nodes have no knowledge of other nodes except those nodes to whom they interact for data receiving and forwarding. Third assumes that the result inconsistency caused by hardware or software faults can be marked by fault detection schemes and are excluded from our malicious attack detection.

4.2 Baseline attestation scheme:

To detect malicious service provider in a cloud computing system this technique uses replay based consistency check and drive both consistency and inconsistency relationship between service providers. Consider example in figure 3 shows the consistency check scheme for attesting three service provider p1, p2, and p3 that offer the same service function f. The portal sends the original input data d1 to p1 and gets back the result f (d1). Next, the portal sends d1′, a duplicate of d1 to p3 and gets back the result f(d1′).The portal then compares f(d1) and f(d1′) to see whether p1 and p3 are consistent.

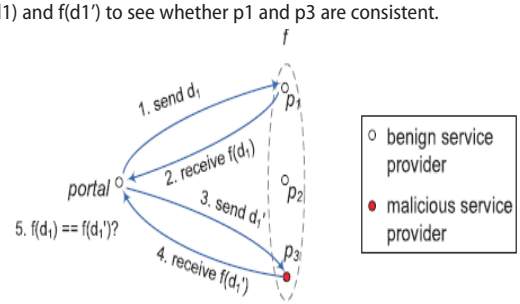

Figure 3: Baseline attestation scheme.

The reason for using this approach is that if two service providers disagree with each other on the processing result of the same input, at least one of them should be malicious. We should not send an input data item and its duplicates (i.e., attestation data) concurrently. After receiving the processing result attestation is performed. Thus, the malicious attackers cannot avoid the risk of being detected when they produce false results on the original data. Integrity attestation is perform in parallel for minimizing the delay for receiving the final result. If two service providers always give consistent output results on all input data, there exists consistency relationship between them. Otherwise, if they give different outputs on at least one input data, there is inconsistency relationship between them. Inconsistency relationship exist also between those service provider which give similar but not same results. If the result of service function is 500 then consistency relationship exist between those service provider whose result is 500 and inconsistency relationship exist when result are different or similar i.e. 500 and 501.

Definition 1. Two service provider give results r1 and r2 , consistency link exist between them only when r1=r2 or between the threshold value defined by a user.

Portal node is responsible for taking a users request and provide a verified result to user. For an incoming tuple di, the portal node may decide to perform integrity attestation with probability pu. If the portal node decides to perform attestation on di, the portal node

first sends di to a pre-defined service path p1->p2 . . . ->pl providing functions f1->f2. . . ->fl. After receiving the processing result for di, the portal replays the duplicate of di on alternative service path(s) such as p1'-> p2'. . . ->pl', where pj' provides the same function fj as pj. The portal may perform data replay on multiple service providers to perform concurrent attestation. After receiving the attestation results, the portal compares each intermediate result between pairs of functionally equivalent service providers pi and pi'. If pi and pi' receive the same input data but produce different output results,

We can say that pi and pi' are inconsistent. Otherwise, we say that pi and pi' are consistent with regard to function fi.

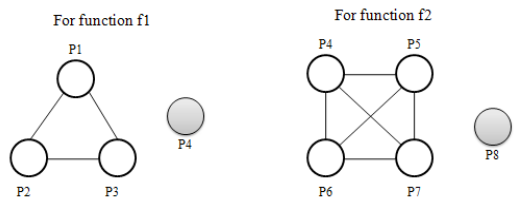

Figure 4: Per function consistency graph.

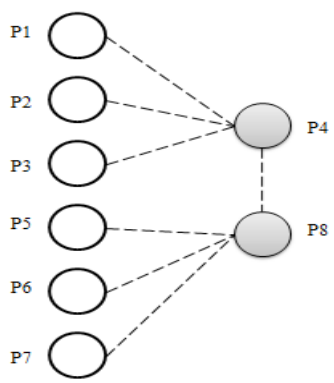

Figure 5: Global inconsistency graph.

Definition 2. A consistency link exists between two service providers who always give consistent output for the same input data during attestation. An inconsistency link exists between two service providers who give at least one inconsistent output for the same input data during attestation.

Construct the per function consistency graph and capture the consistency relationship between service providers. Figure 4 shows the per function consistency graph. Service provider node p1, p2 and p3 are consistent with each other and p4 are inconsistent with all others in function f1. If two service provider consistent for the result of one function, it is not necessary that they are consistent for other function. Service provider node p4 is malicious for function f1 and trusted for function f2.

Definition 3. A per-function consistency graph is an undirected graph, with all the attested service providers that provide the same service function as the vertices and consistency links as the edges.

Global inconsistency graph are used to capture the inconsistent behavior. Inconsistency link exist between those service provider who give different result for the result for same input. We can derive more comprehensive inconsistency relationships by integrating inconsistency links across all functions.

Definition 4. The global inconsistency graph is an undirected graph, with all the attested service providers in the system as the vertex set and inconsistency links as the edges.

Portal node is responsible for creating and maintaining the per function consistency graph and global inconsistency graph. Portal node perform user authentication and communication between user and the service provider perform by portal node. Portal node verify the result of different service provider node and identify the malicious service provider.

4.3 Integrity Attestation Scheme

1. Consistency graph analysis: We first examine per function consistency graphs to pinpoint malicious service providers. The consistency links in per-function consistency graphs represent that the service provider trusted for this function. If the service provider trusted for specific service function they form a clique. Service provider nodes in clique are treated as a trusted service provider for this function. For example, in fig. 4, p1, p2 and p3 are benign service providers and they always form a consistency clique. RunTest[10], developed a clique-based algorithm to pinpoint malicious service providers. If we assume that the number of benign service providers is larger than that of the malicious ones, a benign node will always stay in a clique formed by all benign nodes, which has size larger than k/2, where k is the number of service providers providing the service function. The nodes outside side of a clique are malicious.

2. Inconsistency graph analysis: Inconsistency graph contains all the inconsistency link. Global inconsistency graph is created by all the node which is outside of the clique. Global inconsistency graph shows the behavior of attacker, whether it is aggressive or partial attacker. Global inconsistency graph analysis is done by following condition i.e. service provider node pi is malicious if and only if

Epi>=Cmin……(1)

Where Epi is the incident edges of service provider node i and Cmin is the minimum size of clique into all per function consistency graph. In the figure 5 we check for the node p4, the number of connecting edges are 4 and the minimum size of clique into all function are 3, then p4 is malicious services provider because 4>=3 condition is true. If we check for the node p1 then connecting edges are 1, then 1>=3 condition is false, so the service provider node p1 is trusted.

6. IMPLEMENTATION AND RESULTS

We have implement this proposed approach on a Hadoop MapReduce and tested it on virtual system created by Oracle Solaris Zone technology that uses Intel Core-i5 Series 2.4 GHz processor and RAM is 4 GB and above. Virtual cloud computing environment is created by creating 10 multiple virtual systems within a single system. Figure 6 shows the architecture of Hadoop MapReduce technology in oracle solaris operating system.

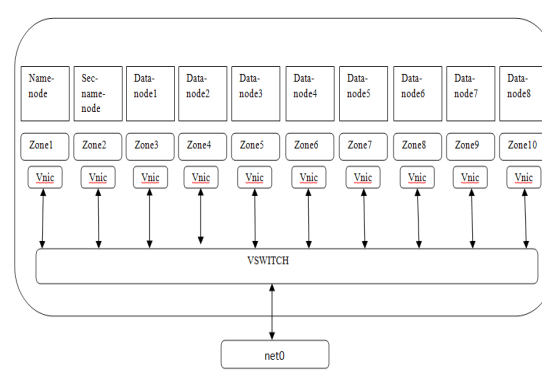

Figure 6: Architecture of Hadoop in Oracle Solaris Zone technology.

NameNode are act as a portal node, all DataNode are serve as application service provider(ASPs) and secondary NameNode are used to maintain logs. All these nodes are connected to virtual switch. Different ASPs(DataNodes) are perform the data processing through MapReduce programming. MapReduce framework is very popular for processing Big data in distributed environment. Various previous techniques are implemented on MapReduce framework for detecting malicious service providers [10][11][14]. MapReduce framework detects malicious misbehavior by log analysis, and other technique such as Hatman: Intra-cloud Trust Management for Hadoop and Trusted MapReduce Infrastructure.

Processing Application runs on the 8 DataNodes and their output are stored in HDFS. We have implemented an algorithm that run on a NameNode that is portal node that take input from HDFS and verify their output. Results of different service provider are stored on the Hadoop Distributed File System, these result are access by portal node for verification purpose. Portal node is responsible for verification of integrity of results. Portal node take the results from different ASPs verify them and identify malicious service provider, then give correct result to user in milliseconds.

Different MapReduce program as a service function run on various DataNodes. Approximately 1000 lines MapReduce code implement for result comparision are run on a NameNode. In this code, output of various service functions is taken from HDFS and compares the result. In this implementation we use matrix implementation of consistency and global inconsistency graph. If two service providers give same result then give 1 otherwisw 0. In figure 7different application service provider provides three different function and 1 represent that the result of DataNode are same as other DataNodes. In this value of different service provider for different function are 1 means there is no malicious service provider and this algorithm only find minimum size of clique in this case. In the figure 8 service provider node p2 and p6 are malicious because its value for different service function is 0, which means it value does not match to other. According to the condition 1 it identify as a malicious. This technique also identify partial malicious service provider. In figure 9 service provider node p3, p5 and p7 identify as a malicious service provider because it trusted for some function and malicious to other function.

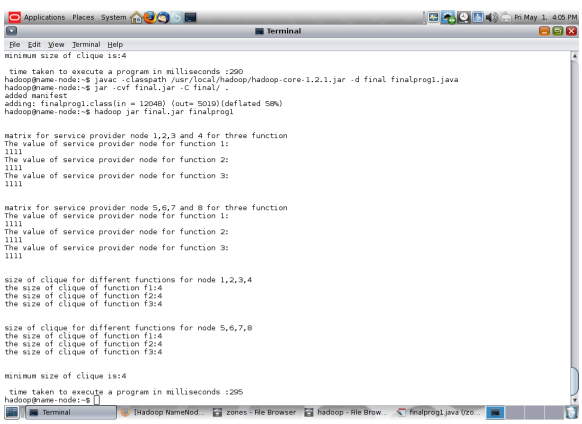

Figure 7: Detection technique when all service provider node are trusted.

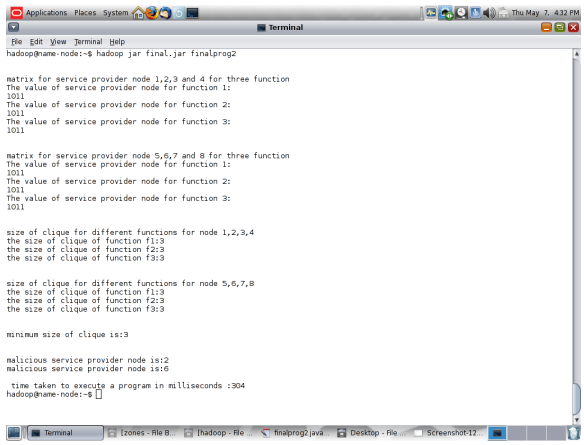

Figure 8: detection scheme detect malicious service providers.

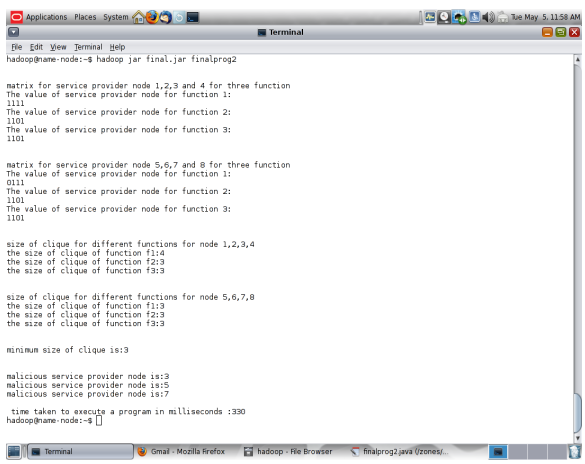

Figure 9: Detection technique detect partial malicious service providers.

CONCLUSION

In this paper we have proposed an integrity attestation technique for detecting malicious service provider in Hadoop MapReduce. This technique employs attestation graph model and consistency inconsistency relationship for detecting malicious service providers. This technique can detect malicious service provider as well as partial service provider efficiently and effectively in milliseconds.

REFERENCES

1. Amazon Web Services, http://aws.amazon.com/, 2015. | 2. Google App Engine, http://code.google.com/appengine/, 2015. | 3. Software as a Service, http://en.wikipedia.org/wiki/Software as a Service, 2015. | 4. G. Alonso, F. Casati, H. Kuno, and V. Machiraju, Web Services Concepts, Architectures and Applications (Data-Centric Systems and Applications). Addison-Wesley Professional, 2002. | 5. T. Erl, Service-Oriented Architecture (SOA): Concepts, Technology, and Design. Prentice Hall, 2005. | 6. Rajkumar Buyya, Christian Vecchiola, S. Thamarai Selvi, Mastering cloud computing McGrow Hill Education Pvt limited. | 7. S. Berger et al., "TVDc: Managing Security in the Trusted Virtual Datacenter," ACM SIGOPS Operating Systems Rev., vol. 42, no. 1, pp. 40-47, 2008. | 8. T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, You Get Off My Cloud! Exploring Information Leakage in Third-Party Compute Clouds," Proc. 16th ACM Conf. Computer and Communications Security (CCS), 2009. | 9. L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," ACM Trans. programming Languages and Systems, vol. 4,no. 3, pp. 382-401, 1982. | 10. J. Du, W. Wei, X. Gu, and T. Yu, "Runtest:Assuring Integrity of Dataflow Processing in Cloud Computing Infrastructures," Proc. ACM Symp. Information, Computer and Comm. Security (ASIACCS),2010. | 11. J. Du, N. Shah, and X. Gu, "Adaptive Data-Driven Service Integrity Attestation for Multi-Tenant Cloud Systems," Proc. Int'l Workshop Quality of Service (IWQoS), 2011. | 12. Scalable Distributed Service Integrity Attestation for Software-as-a-Service Clouds by Juan Du, Member, IEEE, Daniel J. Dean, Student Member, IEEE, Yongmin Tan, Member, IEEE, Xiaohui Gu, Senior Member, IEEE, and Ting Yu, Member, IEEE.,IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 25, NO. 3, MARCH 2014. | 13. Hatman: Intra-cloud Trust Management for Hadoop by Safwan Mahmud Khan Kevin W.Hamlen, Department of Computer Science, University of Texas at Dallas USA. | 14. B. Raman et al., "The SAHARA Model for Service Composition Across Multiple Providers," Proc. First Int'l Conf. Pervasive Computing, Aug. 2002. | 15. X. Gu et al., "QoS-Assured Service Composition in Managed Service Overlay Networks," Proc. 23rd Int'l Conf. Distributed Computing Systems (ICDCS '03), pp. 194-202, 2003. | 16. TMR: Towards a Trusted MapReduce Infrastructure, Anbang Ruan, Andrew Martin Department of Computer Science 2012 IEEE Eighth World Congress on Services. | 17. Toward Detecting Compromised MapReduce Workers through Log Analysis, Eunjung Yoon and Anna Squicciarini Department of Computer Science and Engineering Pennsylvania State University University Park, USA. | 18. ANALYSIS OF SECURITY THREATS AND PREVENTION IN CLOUD STORAGE: REVIEW REPORT Prakash Kuppuswamy and Saeed Q Y Al-Khalidi International Journal of Advanced Research in Engineering and Applied Sciences ISSN: 2278, Vol. 3, No. 1, January 2014. | 19. Data Storage Security in Cloud Computing: A survey, Maulik Dave, Department of Masters in Computer Engineering, LJIET Gujarat Technological University, Ahmadabad- India. International Journal of Advanced Research in Computer Science and Software Engineering Volume 3, Issue 10, October 2013. | 20. Security as a Service Model for Cloud Environment Vijay Varadharajan, Senior Member, IEEE, and Udaya Tupakula, Member, IEEE IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, VOL. 11, NO. 1, MARCH 2014