International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Integrity Attestation and Auto-correction of Services for SaaS Clouds

Amulya Rachna[1], John Prakash Veigas[2]

[1]Department of Computer Science and Engineering Guide

[2]Assistant Professor (M. Tech CSE) Dept CSE,P A College,

Abstract: ***SaaS systems helps the Application Service Provider's (ASP's) to convey their service to the users by making use of cloud computing infrastructure. But because of the partake nature of SaaS clouds, they are often exposed and provide liable chance or the attackers to easily accomplish their strategic attacks. In this paper, we present integrity test which is a novel service integrity attestation for SaaS clouds. Integrity test can detect the fake service providers using an integrated graph attestation analysis method which is better than existing methods. And in addition to that integrity test obsolete the auto-correction of the fake services, that is; it automatically rectifies the corrupted result invoked by the fake service providers and replace it with the justified results provided by genuine service providers. Integrity test can attain higher precision in pinpointing fake attackers than existing techniques.***

Keywords: Cloud computing, SaaS, Service, Integrity attestation, Service Providers

1. Introduction

Cloud computing has become known as a capable hosting platform that enables multiple cloud users called multi-tenants to share a common physical computing infrastructure. With the concepts of Software as a Service (SaaS) [1] and Service Oriented Architecture (SOA) [2], the Internet has evolved into an important service delivery infrastructure instead of only providing host connectivity. Software as a service clouds and Google App Engine [3] build upon the concepts of Software-as-a-Service and Service Oriented Architecture which enable application service providers (ASPs) to deliver their applications via the massive cloud computing infrastructure. However, cloud computing infrastructures are often shared by ASPs from different security domains, which make them vulnerable to malicious attacks [4], [5] as shown in Fig.1. The problem is attackers can pretend to be legitimate service providers to provide fake service components, and the service components provided by benign service providers may include security holes that can be exploited by attackers.

Our work focuses on dataflow processing systems [6], [7], [8] that provide high-performance continuous processing over massive data streams. Previous work on distributed dataflow processing mainly focuses on resource and performance management issues. It usually assumes that all data processing components are trustworthy.

In the previous research papers confidentiality and privacy protection problems [9], [10], [11] are studied extensively but the service integrity attestation problem was not properly addressed. In software as a service cloud one of the most important problems that need to be addressed is this service integrity, no matter whether the data processing in cloud is public or private data. Although traditional Byzantine Fault Tolerance (BFT) techniques [12], [13] can detect malicious behaviour using replicated services, those techniques often incur high overhead and impose certain agreement protocol.

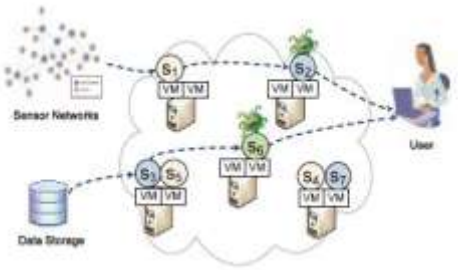

Figure 1: Service integrity attacks in clouds

In this paper, we present **integrity test** a newly integrated service integrity attestation framework for multitenant cloud systems. Integrity test provides a practical service integrity attestation scheme that does not assume trusted entities on third-party service provisioning sites or require application modifications.Integrity test builds upon our previous work RunTest [14] and AdapTest [15] but can provide stronger malicious attacker pinpointing power than 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, Integrity test takes a holistic approach by systematically examining both consistency and inconsistency relationships among different service providers within the entire cloud system. Moreover, Integrity test provides *resultauto-correction* that can automatically replace corrupted data processing results produced by malicious attackers with good results produced by benign service providers.

The rest of this paper is organized as follows. Section 2 presents the literature survey. Section 3 provides the architecture in detail. Section 4 presents the methodology. Finally, the paper concludes in section 5.

2. Literature Survey

Techniques	*Merits*	*Demerits*
BIND system framework	It use the fine grain attestation to verify the integrity ofservice, where it checks the attestation for particular or necessary corrupted node only.	It needs a third party auditor to verify the service.
TEAS system Framework	Demerit of both genuinity and SWATT can be overcome. It automatically generate the agent program	This system also needs a secure kernel hardware or software for verification.
RunTest system framework	It generate integrity attestation graph to verify service provider. It provides non-repudiation results.	The performance is low.
AdapTest system framework	It generates the weighted attestation graph to verify the services. It can reduce the attestation overhead upto 60% and detection delay upto 40%.	It does not provide 100% detection of malicious node.
Integrity test system framework	It also generates the integrity weighted graph to detect the malicious.	

3. Architecture

In this proposed system we are making some assumptions. First of all we are assuming that the total number malicious service components are less than that of the total number of benign service providers in the entire cloud. This assumptions is very important because without this assumption, it would be difficult for any attack detecting scheme to work successfully. The second assumption is the data processing services are important deterministic. That is, the same inputs that are giving by a benign service component will always produce the same output. Fig.2. shows the overall architecture of the proposed system. In this the user give request to cloud the service will be deployed in the cloud the cloud will forward the user request to the SaaS and the response will be send to the cloud by the SaaS. And then the Integrity test process will be done. After that the result auto correction will be done. After that the result will be send to the user by the cloud.

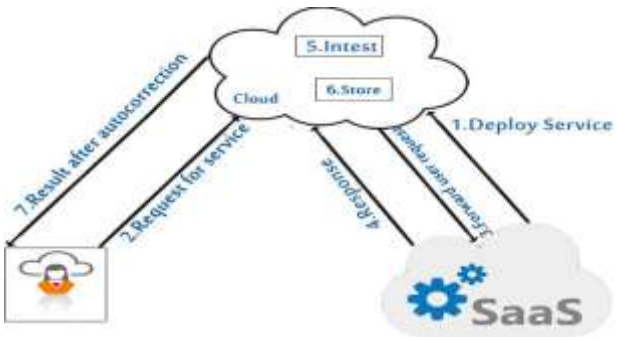

Figure 2: Overall architecture of Integrity test

4. Methodology

Pinpointing malicious service provider

Initially, all nodes are treated as benign nodes and stay in a single clique. As a malicious node keeps misbehaving, it will produce inconsistent results with that of benign nodes sooner or later through attestation, and thus gets excluded from the clique it stayed before. The malicious node either remains in a downsized clique or becomes an isolated node. When the malicious node is pushed away from any of the cliques with size larger than ⌊k/2⌋, it will be pinpointed as malicious. Ultimately, there will be only one clique with size larger than ⌊k/2⌋in the per-function integrity attestation graph, which is formed by all benign nodes. This clique is the maximum clique in the attestation graph. All other cliques, if there is any, should have size less than ⌈k/2⌉. Thus, pinpointing malicious nodes becomes the problem of finding consistency cliques in the attestation graph. We adapt the well-known Bron-Kerbosch (BK) clique finding algorithm as shown in fig.3 for finding consistency cliques in the attestation graph. We maintain three disjoint sets of nodes R, P, and X: The set R stands for the currently growing clique and is initialized to be ∅; The set P stands for prospective nodes which are connected to all nodes in R and using which R can be expanded, and P is initializedto contain all nodes; The set X contains nodes already processed, is initialized to be ∅. The algorithm runs as traversing the recursion tree by moving nodes from P to R and updating the R, P, X sets recursively. A maximal clique is reported when both P and X are empty. The heuristic of the pivot selection is based on the identification and elimination of equal sub-trees appearing in different branches of the recursion tree which lead to the formation of non-maximal cliques.

AdaptiveBK(G)
1. Initialization 1: Mark any two nodes with w <1 edge as unconnected, and with w = 1 edge as connected;
2. Initialization 2: Eliminate nodes that do not have any edge of w = 1
3. FindConsistencyClique(∅, V (G), ∅), where V (G) is the node set of G
FindConsistencyClique(R, P,X)
1. if (P == ∅and X == ∅and size of R >1)
2. Report R as a maximal clique
3. else
4. Let up be the pivot node
5. Assume P = u1, u2, ..., uk
6. for i =1 to k do
7. if ui is not a neighbor of up
8. P = P −ui
9. Rnew = R ∪ui
10. Pnew = P ∩N[ui], where N[ui] is neighbor set of ui
11. Xnew = X ∩N[ui]
12. FindConsistencyClique(Rnew, Pnew,Xnew)
13. X = X ∪ui

Figure 3: Consistency clique discovery algorithm

Identifying attacking patterns:
We characterize all possible attack scenarios using different combinations of parameters (bi, ci) and classify those attacks into five attack patterns.

- **Non-Collusion Always Misbehave (NCAM).**Malicious components always act independently and always give incorrect results. It corresponds to bi = 1 and ci = 0.

- **Non-Collusion Probabilistically Misbehave (NCPM)**.Malicious components always act independently and give incorrect results probabilistically with probability less than 1.
- It corresponds to 0 < bi < 1 and ci = 0.
- **Full Time Full Collusion (FTFC)**. Malicious components always collude and always give the same incorrect results, corresponding to bi = 1, and ci = 1.
- **Partial Time Full Collusion (PTFC)**.Malicious components always collude and give the same incorrect results on selected tuples, corresponding to 0 < bi < 1 and ci = 1.
- **Partial Time Partial Collusion (PTPC)**.Malicious components sometimes collude and sometimes act independently. It corresponds to 0 < bi < 1 and 0 < ci < 1.

IdentifyMaliciousNodes(G)
1. find all maximal cliques CLi ($1 \leq i < k$) in G using adapted
Bron-Kerbosch algorithm with pivot selection
2. in CLi, find those maximal cliques with size larger than $\lfloor k/2 \rfloor$,
CLb ($b \leq i$), where k is the total number of nodes in G
3. check all nodes Nodej in G against nodes in all CLb
4. **if** (Nodej is not in any of CLb)
5. Nodej is malicious
6. **if** (only one maximal clique in CLb, i.e.,the maximum clique)
7. **if** (numCliques == 1)
8. nodes in the clique is identified as Ni
9. **if** (weights of edges from nodes in Ni to rest of nodes not
in Ni are all 0s)
10. attack model NCAM
11. **else**
12. attack model NCPM or PTPC
13. **if** (numCliques ≥ 2)
14. nodes in the maximum clique are Ni, in the rest cliques are N1i, N2i, ...
15. check each clique other than the maximum clique
16. **if** (weights from Ni to any of N1i, N2i, ... are all 0s)
17. attack model FTFC
18. **else if** (all links between Ni and Nji have same weight)
19. attack model PTFC

Figure 4: Integrity attack detection algorithm

Fig.4. shows the pseudo code of our algorithm to identify attack patterns and malicious service nodes.

In this section we present the main modules in the proposed system. Mainly it consists of three modules that are described below

4.1 Baseline Attestation Scheme

Consider the Fig.5 it shows the consistency check method. In that p1, p2 and p3 are the service providers. All of them offer the same function f. The portal sends the original data d1 to the service providers p1 and gets the processing result f(d1). Then the portal sends the duplicate of d1 to p3 and gets the result f(d1'). And if both of them are same means it is consistent and if not means they are inconsistent, that is if two service providers disagree with each other, when processing the same input then any one of them will be malicious. Thus the malicious attackers cannot escape from detecting when they are providing bad results with good results.

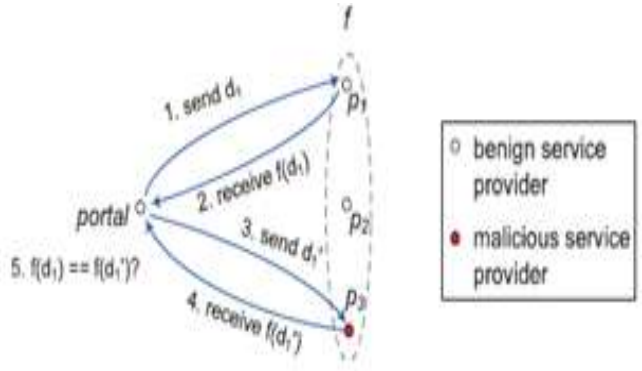

Figure 5: Consistency check

4.2 Integrated Attestation Scheme

Here we present an integrated attestation graph analysis algorithm.

Step 1: Consistency analysis: In the first step it will examine the per-function consistency graph and will pinpoint suspicious service providers. The consistency links in the consistency graph will provide a set of service providers. It will keep consistent with each other on a specific service function. The benign service providers will always keep consistent with each other and will form a clique in terms of consistency links. The colluding attackers can try to escape from being detected. Then next we must examine the per-function in consistency graph too.

Step 2: Inconsistency analysis: This inconsistency graph will contain only the inconsistency links, this may exist in different possible combinations of the benign node and the malicious node set. First we assume that the total number of malicious service providers in the cloud system is not more than the benign service providers, and then we can pinpoint a set of malicious service providers. If two service providers are connected by an inconsistency link, we can say that any one of them is malicious.

4.3 Result Auto Correction for Attacks

Integrity test can not only pinpoint malicious service providers but also it will autocorrect the corrupted data processing results with good results to improve the result quality of the cloud data processing service. Without our attestation scheme, once if an original data input is changed by any malicious attacker, then the processing result of that input will be corrupted and which will result in degraded result quality.

As an illustration for the integrity test, consider the information of some patients from a hospital. The details of patients include their SSN number, name, disease, date of birth, gender, zip code, salary, and DNA. The main objective of integrity test is to maintain the integrity between services being provided and pinpoint any service which is attacked. We consider one original copy of information and three attested copies of same information and also a test data

which has disease and SSN to be tested. When all the four set has consistency relationship between them then it means that they are genuine set of providers so it just passes on the service to the user. Now for illustrating the meaning of integrity test, we consider the third set of attested copy as attacked which doesn't give proper result as other three services. We perform two kinds of tests as functional and combinational test where functional test is for consistency check and combinational test is for inconsistency check. In the functional test, it just forms cliques for genuine and malicious services and in the combinational test it makes the different combination of services and again forms cliques for genuine and malicious services. When the service is attacked, it starts misbehaving and gives improper result. The set which gives exact results forms a clique of being consistent. And the service which is attacked gives corrupted result and so it forms other clique by the help of the two algorithms shown in the above figures. Next, this service which gave the corrupted result is autocorrected by taking help of the set of genuine services.

5. Conclusion

In this paper we introduced a novel integrated service integrity attestation graph analysis scheme for multitenant software-as-a-service cloud system. Integrity test uses a reply based consistency check to verify the service providers. Integrity test will analyses both the consistency and inconsistency graphs to find the malicious attackers efficiently than any other existing techniques. And also it will provide a result auto correction to improve the result quality.

References

[1] Software as a Service, http://en.wikipedia.org/wiki/Software as a Service, 2013.
[2] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, Web Services Concepts, Architectures and Applications (Data-Centric Systems and Applications). Addison Wesley Professional, 2002.
[3] Google App Engine, http://code.google.com/appengine/, 2013.
[4] 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.
[5] 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.
[6] D. J. Abadi and et al. The Design of the Borealis Stream Processing Engine. Proc. of CIDR, 2005.
[7] B. Gedik, H. Andrade, K.-L. Wu, P. S. Yu, and M. Doo. SPADE: the system s declarative stream processing engine. Proc. of SIGMOD, April 2008.
[8] The STREAM Group. STREAM: The Stanford Stream Data Manager. IEEE Data Engineering Bulletin, 26(1):19-26, March 2003.
[9] W. Xu, V.N. Venkatakrishnan, R. Sekar, and I.V. Ramakrishnan, "A Framework for Building Privacy-Conscious Composite Web Services," Proc. IEEE Int'l Conf. Web Services, pp. 655-662, Sept. 2006.
[10] P.C.K. Hung, E. Ferrari, and B. Carminati, "Towards Standardized Web Services Privacy Technologies," IEEE Int'l Conf. Web Services, pp. 174-183, June 2004.
[11] L. Alchaal, V. Roca, and M. Habert, "Managing and Securing Web Services with VPNs," Proc. IEEE Int'l Conf. Web Services, pp. 236- 243, June 2004.
[12] 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.
[13] T. Ho et al., "Byzantine Modification Detection in Multicast Networks Using Randomized Network Coding," Proc. IEEE Int'l Symp. Information Theory (ISIT), 2004.
[14] 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.
[15] 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.