ISSN:0975-9646

Gohila Priyadharshini.C et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (6) , 2014, 8205-8209

A Survey on Service Integrity in Software as a Service

Gohila Priyadharshini.C[#1], Mohana.S[*2]

[#1]*PG scholar, Computer Science Department,* [*2] *Assistant Proffesor,*
M.I.E.T.Engineering college,
Trichy ,India

***Abstract*— A cloud computing is the recently emerging technology which can provide many service based on "pay as you go" way that can be accessed through internet. The services provided by the cloud are software as a service (SaaS), platform as a service (PaaS), infrastructure as a service (IaaS). Software as a service can provide their applications from the application service provide through massive cloud computing environment. Due to the sharing nature ,it is vulnerable to malicious attacker. To identify the malicious ,there are many techniques that can be compared through the survey with and without any special hardware or kernel support.**

***Keywords*— cloud computing, saas, service integrity attestation.**

I. INTRODUCTION

In recent days the cloud computing technology is popular because it is an attracting technology in the field of computer science. Cloud computing is internet base computing that usually referred the shared configurable resources is provided with computers and other devices as services. Cloud computing delegate services with a customer's data, software and computation over a network. The customer of the cloud can get the services through the network. In other words, users are using or buying computing services from others. Cloud can provide Anything as a Service (AaaS).

Many service model are provided by the cloud they are IaaS,SaaS and PaaS.Infrastructure as a service (IaaS) offer computers physical or virtual machines and other resources. Infrastructure as a service (IaaS) clouds often offer additional resources such as a virtual-machine disk image library, raw block storage, and file or object storage, firewalls, load balancers, IP addresses, virtual local area networks (VLANs), and software bundles. Infrastructure as a service (IaaS) cloud providers supply these resources on-demand from their large pools installed in data centers.

In the Platform as a service (PaaS) models, cloud providers deliver a computing platform, typically including operating system, programming language execution environment, database, and web server. Application developers can develop,run their software solutions on a cloud platform without the cost complexity of buying and managing the underlying hardware,software layers. With some Platform as a service (PaaS) offers like Microsoft Azure and Google App Engine, the underlying computer and storage resources scale automatically to match application demand so that the cloud user does not have to allocate resources manually.

This paper concentrate on software as a service. It is a software licensing and delivery model in which software is licensed on a subscription basis and is centrally hosted. Sometimes referred to as "on-demand software". Software as a service (SaaS) is typically accessed by users using a thin client via a web browser. Software as a service (SaaS) has been incorporated into the strategy of all leading enterprise software companies. One of the biggest selling points for these companies is the potential to reduce Information Technology (IT) support costs by outsourcing hardware and software maintenance and support to the Software as a service (SaaS) provider. The vast majority of SaaS solutions are based on a multi-tenant architecture. To support scalability, the application is installed on multiple machines (called horizontal scaling). In some cases, a second version of the application is set up to offer a select group of customers with access to pre-release versions of the applications (e.g., a beta version) for testing purposes. And contrasted with traditional software, where multiple physical copies of the software each potentially of a different version, with a potentially different configuration, and often customized are installed across various customer sites. While an exception rather than the norm, some Software as a service (SaaS) solutions do not use multi-tenancy, or use other mechanisms such as virtualization to cost-effectively manage a large number of customers in place of multi-tenancy. Whether multi-tenancy is a necessary component for software-as-a-service is a topic of controversy.

Some limitations slow down the acceptance of Software as a service (SaaS) and prohibit from being used in some cases:

- Since data are being stored on the vendor's servers, data security becomes an issue.
- Software as a service (SaaS) applications are hosted in the cloud, far away from the application users. And introduces latency into the environment; so, for example, the Software as a service (SaaS) model is not suitable for applications that demand response times in the milliseconds.
- Multi-tenant architectures, which drive cost efficiency for SaaS solution providers, limit customization of applications for large clients, inhibiting such applications from being used in scenarios (applicable

mostly to large enterprises) for which such customization is necessary.

- Some business applications require access to or integration with customer's current data. When such data are large in volume or sensitive (e.g., end users' personal information), integrating them with remotely hosted software can be costly or risky, or can conflict with data governance regulations.
- Constitutional search warrant laws do not protect all forms of Software as a service (SaaS) dynamically stored data. The end result is that a link is added to the chain of security where access to the data, and, by extension, misuse of these data, are limited only by the assumed honesty of 3rd parties or government agencies able to access the data on their own recognizance.
- Switching Software as a service (SaaS) vendors may involve the slow and difficult task of transferring very large data files over the Internet.
- Organizations that adopt SaaS may forced into adopting new versions, which might result in unforeseen training costs or an increase in probability that a user might make an error.

Relying on an Internet connection means that data are transferred to and from a SaaS firm at internet speeds, rather than the potentially higher speeds of a firm's internal network.

Although confidentiality and privacy protection problems have been extensively studied by previous research [6],[7],[8],[9],[10], the service integrity will be discussed in this paper. The rest of the paper is organized as follows: Section 2 presents a techniques available for checking a service integrity. Section 3 provides the types of malicious attacker. Section 4 describes the comparative study.

II. TECHNIQUES FOR VERIFYING SERVICE INTEGRITY

In this paper, we will provide a broad overview of the different techniques for verifying the service integrity. We will provide a broad view of the major algorithms available for each method, and the variations on the different techniques. We will also discuss a combination of different concepts.

1)The BIND Technique

In this section, we will discuss the BIND (Binding information and data)method for the verification of the integrity of services provided by the SaaS cloud system model. It consist of the fine grained attestation framework . It can provide the verification through the secure kernel or third party Auditor.

BIND offers the following properties:1)It attest only to the piece of code we are concerned about. 2) It narrows the gap between time of attestation and time of use. And measures a piece of code immediately before it is executed then uses a sand-boxing mechanism to protect the execution of the attested code.3) It ties the code attestation with data that the code produce ,such that we can pinpoint what code has been run to generate that data.

1.1) Attestation Annotation Mechanism

In this mechanism ,the programmer allowed to identify and annotate the beginning and end of this critical piece of code and every time this piece of code is executed. And this can be validated by checking and verifying checksum value.

1.2 SandBox Mechanism

In this ,the execution of the critical code will be preserved. An integrity proof for a piece of the input data into the integrity statement of the code and output data. This enable us to achieve transitive integrity verification with constant overhead i.e.,we only need to verify one signature to guarantee the integrity of the entire chain of process that transformed the data.

1.3) Verification Of Authenticator Through Hash

It consist of two steps to verify: 1) verify the signature, 2) verify the hash. Since verifying the signature is straightforward , we now explain how to verify the hash and how to enable different software version and software upgrades. BIND allows the application to register one or more legal hash values. We assume that for each application's trusted authority that signs certificate for legal hash values. When an application registers a hash value ,it has to show a correct certificate. The public key of an applications trusted authority is included whenever BIND supports various software versions and software upgrades.

1.4 Conclusion

This BIND system uses the Diffee-Hellman key exchange for providing the integrity attestation.

The existing system of this system is TCG style framework, it uses the coarse grain attestation where it provide attestation for the entire operating system.

In this remote verification is difficult and the software will be compromised at runtime. Then next system is COPILOT system framework where it miss the short lived intrusion. Hence these things can be overcome through the BIND system.

2) The TEAS Technique

A software scheme for protecting the integrity of computing platforms using Timed Executable Agent Systems(TEAS). A trusted challenger issues an authenticated challenge to a perhaps corrupt responder. A new is that the issued challenge is an executable program that can potentially compute any function on the responder. The responder must compare not only the correct value implied by the agent, but also must complete this computation within time bounds prescribed by the challenger. It also need some third party auditor to verify the integrity.

The algorithm used here is agent generation and verification algorithm. In this both agent and the client generate the checksum value. The generated checksum of client and agent's checksum will be compared then verified by the auditor.

It consist of two adversaries 1) on-line adversaries, 2) off-line adversaries.

2.1) Off-Line Adversaries

In this class we assume that the adversary controlling a client will try to analyze the incoming programs without running them. Recall that in static analysis of program, it can be analyzed in isolation , without inputs and without the state of the machine where they will run. An off-line adversaries will perform a similar type of analysis, except that it might also have access to inputs and state of the client.

2.2) On-Line Adversaries

In this class we assume that the adversary controlling a client will also be able to run the incoming programs.

2.3)Conclusion

Finally it can verify the service integrity with the help of third party auditor or secure kernel .

The existing system of Timed executable Agent System(TEAS) is verifies the agent through the genuinity system. It also uses the checksum composition. This system is corrupted by the substitution attack. After that this drawback can be overcome through the SWATT system and this system is tightly coupled between target system and trusted host. And then it needs some large memory to access is (O(n log n)). So these drawbacks can overcome through the TEAS system.

*3)**The RunTest Technique***

This technique provide a scalable runtime integrity attestation framework. To assure that the integrity of data flow processing in cloud it provides a light weight application level attestation methods.

To identify the untruthful data flow processing results and pinpointing malicious data processing service provider and finally detect a colluding attack behavior.

3.1) Generation of Integrity Attestation Graph

The malicious can be determined through Integrity Attestation Graph. To capture the aggregated cross node integrity attestation results. It also includes the statistical output of consistency and inconsistency information from different data flow processing nodes. It uses the Bron-Kerbosch(Bk) Clique finding algorithm for finding the consistency cliques in the attestation graph.

Proposition 1:

All benign service provider always form the consistency clique in the integrity attestation graph.

Proposition 2:

Any node that is outside of all maximal cliques of size larger than [k/2] in a per-function attestation graph must be a malicious node.

3.2)Security Analysis

Our scheme of pinpointing malicious service provider is based on attestation graph analysis. We claim that the scheme preserves the following properties:

*Property 1***:** No false positive: A benign service provider will not be pinpointed as malicious.

*Property 2***:** Non-Repudiation: For any pinpointed malicious service provider ,the trusted portal node can present evidence to prove it is malicious

3.3) Data Quality

It define data quality as the percentage of proposed data with correct results. This scheme can detect the tempered data results probabilistically and report data quality close to actual data quality.

$$Qr = 1-(c/n)$$

Qr be the data quality and 1-(c/n) be the actual quality.

3.4)Conclusion

Hence this technique can provide the good service provider and also determine malicious behavior. It provide non-repudiation and its demerit is low performance.

*4) **The AdapTest Technique***

This presents a novel adaptive data driven runtime service integrity attestation framework for Multitenant cloud system. It can significantly reduce attestation overhead and shorten delay by adaptively selecting attested node based on dynamically derived trust scores. It treats the attested service as a Black Boxes and does not impose any special hardware or software requirements on the cloud system or application service provider.

4.1)Generation of Weighted Attestation Graph

This graph will provide the trust score for a single node or a pair node.

Definition 1: A weighted Attestation graph is an un-directed graph consisting of all functionality equivalent service instances as nodes. The weight of each edge consist of a pair of counters denoting the number of consistent results respectively.

Definition 2: The trust score of the node Si, is defined by α_i , is defined as the fraction of consistent results returned by the node S_i when attested with all the other nodes. Node trust scores range within [0,1] and are initialized to be 1.

Definition 3: The pairwise trust score between two services instances S_i and S_j, denoted by $\beta_{(i,j)}$,is calculated by the fraction of consistent results when S_i is attested against S_j . the pairwise trust score ranges within[0,1] and are initialized to be -1, which means that S_i and S_j have not been attested with each other yet.

A. Per-Hop Adaptive attestation :

In this ,the attestation will be provided through clique based algorithm. Initially it selects the suspicious nodes that have low trust scores and attest those suspicious node more frequently.

B. Multi-Hope Attestation:

It also provide the attestation through clique based algorithm. Complicated data processing services often from comprise multiple data processing functions called service hops. Malicious attacker can attack any of the service hops to compromise the final data processing results.

4.2)Conclusion

Finally from the experimental results , it will reduce the attestation overhead by upto 60% and the detection reduced by 40% compared to the previous approach. And it does not need any third party auditor as the previous system. Hence it can provide the better result to the user and it does not provide 100% detection of malicious node.

*5)**The IntTest Technique***

This technique provides the novel integrated attestation graph analysis scheme that can provide a stronger attacker pinpointing power than the previous schemes. It can automatically enhance the result quality by replacing the bad results. This can achieve higher attacker pinpointing accuracy than existing approaches.

5.1) Generation of Integrity Attestation Graph

It consist of two attestation graph , 1) per-function consistency and 2) global-function inconsistency.

Definition 1: For two output results r_1 and r_2 , which come from two functionality equivalent service providers, respectively result consistency is defined as either $r_{1=} r_2$ according to user – defined distance function $D(r_1, r_2)$falls within a threshold δ

*Definition 2***:** A per-function consistency graph is an undirected graph ,with all the attested service provider that provide the same service function as the vertices and consistency link as the edge.

Definition 3: The global inconsistency graph is an undirected graph,with all the attested service provider in the system as the vertex set and inconsistency link as the edges.

This attestation determine the malicious through the clique finding algorithm. By combining the definition 1 and 2, it can evaluate the malicious service provider without using the third party auditor or secure kernel.

5.2)Security Analysis

A summary of the result of our analytical study about IntTest, additional details along with a proof the proposition presented in this section.

*Proposition1:*Given an accurate upper bound of the no.of malicious service providers k , if malicious service providers always collude together , IntTest has zero false positive.

Although the clique finding algorithm cannot guarantee zero false positive when there are multiple independent colluding groups, it will be difficult for attackers to escape the detection with multiple independent colluding groups since attackers will have inconsistency links not only with benign node. Additionally,this approach limits the damage of colluding attacker that can cause if they can evade detection in two ways. Initially , our algorithm limits the number of functions which can simultaneously attacked. Second ,our algorithm ensure a single attacker cannot participate in compromising an unlimited number of service functions without being detected.

5.3)Conclusion

Finally ,this mechanism ca determine the malicious one without any third party auditor or secure kernel hardware or software. It limit the attack scope and make difficult to attack the popular service provider and finally it automatically replace the bad results with good results.

III. Different Form Of Malicious Attacker In Service Provider

In a shared cloud infrastructure ,malicious attacker can pretend to be legitimate service provider to provide fake service instance or compromised vulnerable benign service instance by exploiting their security roles. It consist of different form of malicious which are described below.

1) Malicious Intermediary

A malicious intermediary may arbitrarily alter and inject protocol data. To prevent such attacks, we can employ cryptographic construction such as message authentication codes or digital signatures.

2) The Data Misuse Attack

It uses authenticated protocol data in a malicious way. For instance , a malicious intermediary can perform a data suppression attack by effusing to forward any data. Then the attacker can perform the replay attack by replaying data that have been authenticated but are outdated.

3) Malicious process and the Data Falsification Attack

In a highly adversarial environment , an attacker may corrupt one or more process in the system. A malicious process is capable of injection bogus data into distributed system. We refer to this attack as the data falsification attack.

4) Non-collusion Always Misbehave(NCAM)

Malicious component always act independently and always give incorrect results. It correspond to $b_i = 1$ and $c_i = 0$.

5) Non-collusion Probabilistically Misbehave(NCPM)

Malicious components always act independently and give incorrect results probabilistically with probability less than 1. Different malicious components may have different misbehaving probability b_i . It corresponds to $0 < b_i < 1$ and $c_i = 0$.

6) Full time Full Collusion(FTFC)

Malicious component always collude and always give the same incorrect results,corresponding to $b_i = 1$ and $c_i = 1$.

7) Partial Time Full Collusion(PTFC)

Malicious components always collude and give the same incorrect results on selected tuples, corresponding to $0 < b_i < 1$ and $c_i = 1$.

8) Partial Time Partial Collusion

Malicious component sometimes collude and sometimes act independently. It corresponds to $0 < b_i < 1$ and $0 < c_i < 1$.

IV. COMPARISON STUDY

TECHNIQUES	MERITS	DEMERITS
TCG-style system framework	It use the coarse grain attestation to verify the integrity of service,where it provide the attestation for entire os.	In this remote verification is difficult and software will be compromised at runtime.
COPILOT system framework	It also use the same coarse grain attestation to verify integrity of service provider.	It can miss the short lived intrusions.
BIND system framework	It use the fine grain attestation to verify the integrity of service ,where it checks the attestation fo particular or necessary corrupted node only.	It need a third party auditor to verify the service.
Genuinity system	It use the checksum verification for integrity.	This system can be corrupted by the substitution attack.
SWATT system	It overcome the substitution attack.	It needs large memory to access (O(n log n))
TEAS system Framework	Demerit of both genuinity and SWATT can be overcome. It automatically generate the agent program	This system also need a secure kernel hardware or software for verification.
RunTest system framework	It generate integrity attestation graph to verify service provider. It provide 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.
IntTest system framework	It also generate the integrity weighted graph to detect the malicious.	It replaces the bad service results with the good service result.

V. CONCLUSION

This paper, discussed about various approaches and techniques used in providing the service integrity of SaaS cloud model. Each techniques has its own advantages and dis-advantages. Most integrity attacks can be effectively destroyed by the advanced techniques and approaches. All methods are approximate to our goal of providing the service or search results with integrity,we need to further perfect those approaches or develop some efficient methods.

REFERENCES

[1] Garay.J and Huelsbergen.L, "Software integrity protection using timed executable agents," in Proceedings of ACM Symposium on Information, Computer and Communications Security (ASIACCS), Taiwan, Mar. 2006.

[2] Juan Du Daniel J. Dean, Yongmin Tan, Xiaohui Gu, Senior and Ting Yu Scalable Distributed Service Integrity Attestation for Software-as-a-Service Clouds .

[3] Du.J, Wei.W, Gu.X, and Yu.T, "Runtest: Assuring Integrity of Dataflow Processing in Cloud Computing Infrastructures," Proc.ACM Symp. Information, Computer and Comm. Security (ASIACCS),2010.

[4] Du.J, Shah.N, and Gu.X, "Adaptive Data-Driven Service Integrity Attestation for Multi-Tenant Cloud Systems," Proc. Int'l Workshop Quality of Service (IWQoS), 2011. Virtual Computing Lab, http://vcl.ncsu.edu/, 2013.

[5] Ho et al.T, "Byzantine Modification Detection in Multicast Networks Using Randomized Network Coding," Proc. IEEE Int'l Symp. Information Theory (ISIT), 2004.

[6] Hwang.I, "A Survey of Fault Detection, Isolation, and Reconfiguration Methods," IEEE Trans. Control System Technology, vol. 18,no. 3, pp. 636-653, May 2010.

[7] Lamport.L, Shostak.R, and Pease.M, "The Byzantine Generals Problem," ACM Trans. Programming Languages and Systems, vol. 4,no. 3, pp. 382-401, 1982

[8] Shi.E, Perrig.A, and Doorn.L.V, "Bind: A fine-grained attestation service for secure distributed systems," in Proceedings of the IEEE Symposium on Security and Privacy, 2005.

[9] Xu.W, Venkatakrishnan.V. N, Sekar.R, and Ramakrishnan .I. V, "A framework for building privacy-conscious composite web services," in IEEE International Conference on Web Services, Chicago, IL, Sep. 2006, pp. 655–662.

[10] Zhang.H, Savoie.M,Campbell.S, Figuerola.S, von Bochmann.G, and Arnaud.B.S, "Service-oriented virtual private networks for grid applications," in IEEE International Conference on Web Services, Salt Lake City, UT, Jul. 2007, pp. 944–951.