IJSEAT International Journal of Science Engineering and Advance Technology

Adaptable conveyed administration uprightness verification for software as a service clouds

[1]Satuluri Om Sri Sai Krishna,[2]G.B.V.Padmanadh,[3]Dr. G Veereswara Swamy

[1]M.Tech(research scholar in Software engineering),[2]Associate Professor,[3]Professor

Dept. of Computer science in Kakinada Institute of Engg. & Tech.,3Dept. of Computer science

Gitam University

Abstract

SaaS gives an adaptable situation through which application administration providers permitted to have their applications in a conveyed domain, with the goal that clients can get to the facilitated administrations in a less demanding manner. As the earth is partaking in nature there is more degree for SaaS mists defenseless against vindictive assailants. In this paper IntTest another trustworthiness verification plan is presented that can utilize diagram investigation plan to accomplish higher Pinpointing of aggressors. Here additionally consequently rectifying the aftereffects of malignant aggressors with results gave by kindhearted administration supplier's method called auto revision is presented. Additionally we actualized IntTest and tried on a creation cloud foundation, the exploratory results demonstrate this plan accomplished higher precision than past plans. IntTest does not require any protected piece backing and equipment and it additionally underpins for huge scale cloud computing foundation.

Keywords: Distributed service integrity attestation, cloud computing, secure distributed data processing.

I. Introduction

Lately the cloud computing innovation is well known in light of the fact that it is a drawing in innovation in the field of software engineering. Cloud computing is web base registering that typically alluded the common configurable assets is furnished with PCs and different gadgets as administrations. Cloud computing agent administrations with a client's information, programming and calculation over a system. The client of the cloud can get the administrations through the system. At the end of the day, clients are utilizing or purchasing figuring administrations from others. Cloud can give anything as a Service (AaaS). Numerous administration model are given by the cloud they are IaaS, SaaS and PaaS. Base as an administration (IaaS) offer PCs physical or virtual machines and different assets. Base as an administration (IaaS) mists regularly offer extra assets, for example, a virtual-machine circle picture library, crude square stockpiling, and record or question stockpiling, firewalls, load balancers, IP addresses, virtual neighborhood (VLANs), and programming groups. Framework as an administration (IaaS) cloud suppliers supply these assets on interest from their substantial pools introduced in server farms. In the Platform as an administration (PaaS) models, cloud suppliers convey a processing stage, commonly including working framework, programming dialect execution environment, database, and web server. Application engineers can create, run their product arrangements on a cloud stage without the expense intricacy of purchasing and dealing with the basic equipment, programming layers. With some Platform as an administration (PaaS) offers like Microsoft Azure and Google App Engine, the hidden PC and capacity assets scale naturally to match application request so that the cloud client does not need to distribute assets physically. This paper focus on programming as an administration. It is a product authorizing and conveyance model in which programming is authorized on a membership premise and is midway facilitated. Once in a while alluded to as "on-interest programming". Programming as an administration (SaaS) is regularly gotten to by clients utilizing a slender customer by means of a web program. Programming as an administration (SaaS) has been fused into the procedure of all driving undertaking programming organizations. One of the greatest offering focuses for these organizations is the possibility to diminish Information Technology (IT) bolster costs by outsourcing equipment and programming upkeep and backing to the Software as an administration (SaaS) supplier. Most by far of SaaS arrangements depend on a multi-occupant building design. To bolster versatility, the application is introduced on various machines (called level scaling). At times, a second form of the application is set up to offer a select gathering of clients with access to pre-discharge variants of the applications (e.g., a beta adaptation) for testing purposes. What's more, appeared differently in relation to customary programming, where numerous physical duplicates of the product each possibly of an alternate form, with a conceivably distinctive arrangement, and regularly redid are introduced crosswise over different client destinations. While an exemption as opposed to the standard, some Software as an administration (SaaS) arrangements don't utilize multitenancy, or use different systems, for example, virtualization to cost-successfully deal with countless set up of multi-occupancy. Whether multi-tenure is a fundamental part for programming as-an administration is a subject of contention.

A few constraints moderate down the acknowledgment of Software as an administration (SaaS) and preclude from being utilized as a part of a few cases:

• Since information are being put away on the merchant's servers, information security turns into an issue.

• Software as an administration (SaaS) applications are facilitated in the cloud, far from the application clients. Furthermore, brings dormancy into nature; thus, for instance, the Software as an administration (SaaS) model is not suitable for applications that request reaction times in the milliseconds.

• Multi-occupant architectures, which drive cost productivity for SaaS arrangement suppliers, limit customization of utilizations for huge customers, hindering such applications from being utilized as a part of situations (material generally to extensive endeavors) for which such customization is vital.

• Some business applications oblige access to or joining with client's present information. At the point when such information are substantial in volume or delicate (e.g., end clients' close to home data), incorporating them with remotely facilitated programming can be exorbitant or unsafe, or can strife with information administration regulations.

• Constitutional court order laws don't ensure all types of Software as an administration (SaaS) powerfully put away information. The final result is that a connection is added to the chain of security where access to the information, and, by expansion, abuse of these information, are restricted just by the accepted trustworthiness of third gatherings or government organizations ready to get to the information all alone recognizance.

• Switching Software as an administration (SaaS) sellers may include the moderate and troublesome errand of exchanging extensive information records over the Internet.

• Organizations that embrace SaaS may compel into receiving new forms, which may bring about unexpected preparing expenses or an increment in likelihood that a client may make a blunder. Depending on an Internet association implies that information are exchanged to and from a SaaS firm at web speeds, as opposed to the conceivably higher paces of an association's interior system. In spite of the fact that secrecy and security assurance issues have been widely contemplated by past examination the administration respectability will be talked about in this paper.

II. Problem Formulation

For a given SaaS system, the goal of IntTest is to pinpoint any malicious service provider that offers an untruthful service function. IntTest treats all service components as black-boxes, which does not require any special hardware or secure kernel support on the cloud platform. We now describe our attack model and our key assumptions as follows Attack model: A malicious attacker can pretend to be a legitimate service provider or take control of vulnerable service providers to provide untruthful service functions. Malicious attackers can be stealthy, which means they can misbehave on a selective subset of input data or service functions while pretending to be benign service providers on other input data or functions.

The stealthy behavior makes detection more challenging due to the following reasons:

1) The detection scheme needs to be hidden from the attackers to prevent attackers from gaining knowledge on the set of data processing results that will be verified and therefore easily escaping detection;

2) The detection scheme needsto be scalable while being able to capture misbehavior that may be both unpredictable and occasional. In a large-scale cloud system, we need to consider colluding attack scenarios where multiple malicious attackers collude or multiple service sites are simultaneously compromised and controlled by a single malicious attacker. Attackers could sporadically collude, which means an attacker can collude with an arbitrary subset of its colluders at any time. We assume that malicious nodes have no knowledge of other nodes except those they interact with directly. However, attackers can communicate with their colluders in an arbitrary way. Attackers can also change their attacking and colluding strategies arbitrarily.

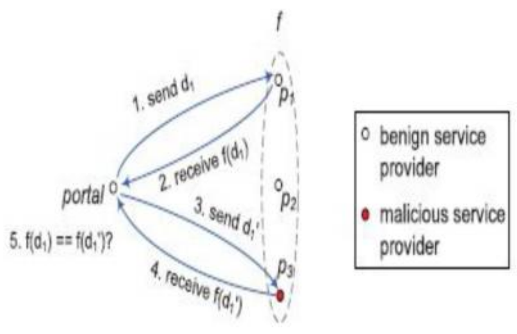

Fig.. Replay-based consistency check

Assumptions:

1. We first assume that the total number of malicious service components is less than the total number of benign ones in the entire cloud system. Without this assumption, it would be very hard, if not totally impossible, for any attack detection scheme to work when comparable ground truth processing results are not available. However, different from RunTest, AdapTest, or any previous majority voting schemes, IntTest does not assume benign service components have to be the majority for every service function, which will greatly enhance our Pinpointing power and limit the scope of service functions that can be compromised by malicious attackers.
2. Second, we assume that the data processing services are input-deterministic, that is, given the same input, a benign service component always produces the same or similar output (based on a user defined similarity function). Many data stream processing functions fall into this category. We can also easily extend our attestation framework to support stateful data processing services, which however is outside the scope of this paper. Third, we also assume 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.

III. SaaS Cloud System Model

It develops upon the concepts of Software as a Service (SaaS) and Service Oriented Architecture (SOA) which allows application service providers (ASPs) to deliver their applications via large-scale cloud computing infrastructures. Amazon Web Service (AWS) and Google App Engine are examples to provide a set of application services supporting enterprise applications and big data processing. A distributed application service can be dynamically composed from individual service components provided by different ASPs. For example, a disaster assistance claim processing application consists of voice-over-IP (VoIP) analysis component, email analysis component, community discovery Component, and clustering and joins components. Our work focuses on data processing services which have become increasingly popular with applications in any real world usage domains such as business intelligence, security surveillance, and scientific computing. Each service component, denoted by ci, provides a specific data processing function, denoted by fi, such as sorting, filtering, correlation, or data mining utilities. Each service component can have one or more input ports for receiving input data tuples, denoted by di, and one or more output ports to emit output tuples. In a large-scale SaaS cloud, the same service function can be provided by different ASPs. Those functionally equivalent service components exist because: i) service providers may create replicated service components for load balancing and fault tolerance purposes; and ii) popular services may attract different service providers for profit. To support automatic service composition, we can deploy a set of portal nodes that serve as the gateway for the user to access the composed services in the SaaS cloud. The portal node can aggregate different service components into composite services based on the user’s requirements. For security protection, the portal node can perform authentication on users to avoid malicious users from disturbing normal service provisioning. Different from other open distributed systems such as peer-to-peer networks and volunteer computing environments, SaaS cloud systems possess a set of unique features. First, third-party ASPs typically do not want to reveal the internal implementation details of their software services for intellectual property protection. Thus, it is difficult to only rely on challenge-based attestation scheme where the verifier is assumed to have certain knowledge about the software implementation or have access to the software source code. Second, both the cloud infrastructure provider and third-party service providers are autonomous entities. It is impractical to impose any special hardware or secure kernel support on individual service provisioning sites. Third, for privacy protection, only portal nodes have global information about which service functions are provided by which service providers in the SaaS cloud. Neither cloud users nor individual ASPs have the global knowledge about the SaaS cloud such as the number of ASPs and the identifiers of the ASPs offering a specific service function.

IV. Cloud Security Challenges

When a company mitigates to intense cloud services, and particularly public cloud services, abundant of the automatic data processing system infrastructure can currently below the management of cloud service supplier. These management initiatives can needs clearly delineating the possession and responsibility roles of each the cloud supplier and therefore the organization functioning within the role of client. Security managers should be able to confirm what detective and preventative controls exist to obviously outline security posture of the organization. Though correct security controls should be implement supported quality, threat, and vulnerability risk assessment matrices. Encryption: the sensitivity of information might need that the network traffic to and from the virtual machine be encrypted, victimization encoding at the host OS computer code.
• Physical security: keep the virtual system and cloud management hosts safe and secure behind carded doors, and environmentally safe.

• Authentication and access control: the authentication capabilities among your virtual system ought to copy the approach your different physical systems evidence. Only once watchword and life science ought to all be enforced within the same manner. Conjointly authentication needs whereas you're causing information or message from one cloud to different cloud. To produce message authentication we'll use digital signatures.
• Separation of duties: as system get additional complicated, misconfiguration occur, as a result of lack of experience in addition to meager communication. Take care to enforce least privileges with access controls and responsibleness.
• Configuration, modification management, and patch management: this is often important and typically unnoted in smaller organizations. Configuration, modification management, patch management, and updated processes ought to be maintained within the virtual world moreover as physical world.
• Intrusion detection and prevention: what's returning into and going out of your network must recognize. a bunch primarily based} intrusion bar system in addition to a hypervisor based resolution may examine for virtual network traffic.

V. Proposed System

Software as a service and service oriented architecture are the basic concepts of SaaSclouds and this will allow the application service provider to deliver their application via cloud computing infrastructure. In our proposed method we are introducing a new concept called IntTest. The main goal of IntTest is, it can pinpoint all the malicious service providers. IntTest will treat all the service providers as black boxes and this does not need any special hardware or secure kernel support. When we are considering the large scale cloud system multiple service providers may simultaneously compromised by a single malicious attacker. In this we assume that the malicious nodes are not having any knowledge about the other nodes except those which they are directly interacting. 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 input that are giving by a benign service component will always produce the same output. And finally we assume that the inconsistency caused by hardware or software faults can be excluded from malicious attacks. Fig. shows the overall architecture of the proposed system. In this the user give request to cloud the serve 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 IntTest 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. The architecture shows this IntTest module in detail.

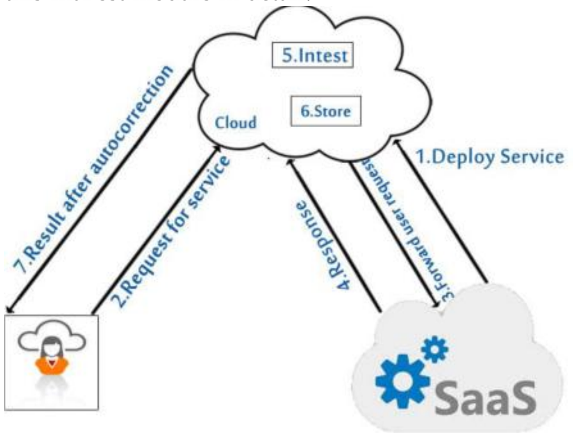

Figure: Over all architecture of the proposed method

VI. 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 bi =1 and ci =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 bi. It corresponds to 0< bi < bi < 1 and ci =1. 8) Partial Time Partial Collusion Malicious component sometimes collude and sometimes act independently. It corresponds to 0< bi< ci<1.

VII. Signature verification algorithm

For receiver to authenticate senders signature, he must have a copy of her public-key curve point Q_A. Receiver can verify Q_A is a valid curve point as follows:

1. Check that is Q_A not equal to the identity element O, and its coordinates are otherwise valid
2. Check that Q_A lies on the curve
3. Check that $n \times Q_A = O$

After that, Bob follows these steps:

1. Verify that r and s are integers in $[1, n-1]$. If not, the signature is invalid.
2. Calculate $e = \mathrm{HASH}(m)$, where HASH is the same function used in the signature generation.
3. Let z be the L_n leftmost bits of e.
4. Calculate $w = s^{-1} \bmod n$.
5. Calculate $u_1 = zw \bmod n$ and $u_2 = rw \bmod n$.
6. Calculate the curve point $(x_1, y_1) = u_1 \times G + u_2 \times Q_A$.
7. The signature is valid if $r \equiv x_1 \pmod n$, invalid otherwise.

VIII. Security Analysis

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

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.

Future Wok

In the future, allow verifying functional properties of cloud services, they have not yet matured. Multiple recent works have tackled specific concerns that arise in the context of cloud storage, and promising techniques have emerged.

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 Softwareas-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.

Authors:

Mr. Satuluri Om Sri Sai Krishna is a student of Kakinada Institute of Engineering and Technology, Kakinada. Presently I am pursuing my M.Tech [Software Engineering] from this college.My area of interest includes Computer networks and Programing languages.

G.B.V.Padmanadh working as Associate Professor,Dept. of Computer science Kakinada Institute of Engg. & Tech.He so many published national and international papers.His area of internest inimage mining and Data mining.

Dr. G. Veereswara swamy Received M.Tech (C.S & S.E) in Computer Science Engineering and Ph.D from Andhra University during 1999 and 1993 respectively. He is working as faculty member in GITAM University from the last 20 years. Presently working as a Professor in the Department of Engineering Physics. His research interest includes Mobile computing, Operating Systems, and Networks. He Published more than 20 papers in various National and International Conference and Journals