IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

ACCESS CONTROL OF DATA OF THE USERS IN ENCRYPTED CLOUD DATABASES

Ms.V.Mahendar, Miss.P.Mounika

Abstract: Software-as-a-service (SaaS) cloud systems enable application service providers to deliver their applications via massive cloud computing infrastructures. However, due to their sharing nature, SaaS clouds are vulnerable to malicious attacks. In this paper, we present IntTest, a scalable and effective service integrity attestation framework for SaaS clouds. IntTest provides a novel integrated attestation graph analysis scheme that can provide stronger attacker pinpointing power than previous schemes. Moreover, IntTest can automatically enhance result quality by replacing bad results produced by malicious attackers with good results produced by benign service providers. We have implemented a prototype of the IntTest system and tested it on a production cloud computing infrastructure using IBM System S stream processing applications. Our experimental results show that IntTest can achieve higher attacker pinpointing accuracy than existing approaches. IntTest does not require any special hardware or secure kernel support and imposes little performance impact to the application, which makes it practical for large-scale cloud systems.

1INTRODUCTION

Cloud computing has emerged as a cost-effective resource leasing paradigm, which obviates the need for users maintain complex physical computing infrastructures by themselves. Software-as-a-service (SaaS) clouds e build upon the concepts of software as a service and service-oriented architecture (SOA), which enable application service providers (ASPs) to deliver their applications via the massive cloud computing infrastructure. In particular, our work focuses on data stream processing services that are considered to be one class of killer applications for clouds with many real-world applications in security surveillance, scientific computing, and business intelligence. However, cloud

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

computing infrastructures are often shared by ASPs from different security domains, which make them vulnerable to malicious attacks. For example, 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 service integrity attacks that cause the user to receive untruthful data processing results. Although confidentiality and privacy protection problems have been extensively studied by previous research, the service integrity attestation problem has not been properly addressed. Moreover, service integrity is the most prevalent problem, which needs to be addressed no matter whether public or private data are proessed by the cloud system.

Fig. 1. Service integrity attack in cloud-based data processing. S_i denotes different service component and VM denotes virtual machines.

Although previous work has provided various software integrity attestation solutions, those techniques often require special trusted hardware or secure kernel support, which makes them difficult to be deployed on large-scale cloud computing infrastructures. Traditional Byzantine fault tolerance (BFT) techniques can detect arbitrary misbehaviors using full-time majority voting (FTMV) over all replicas, which however incur high overhead to the cloud system. A detailed discussion of the related work can be found the online supplementary material. In this paper, we present IntTest, 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 Run Test and AdapTest but can provide stronger malicious attacker pinpointing power than Run Test and AdapTest. Specifically, Run Text 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

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

malicious attackers may launch colluding attacks on certain targeted service functions to invalidate the assumption. To address the challenge, ntTest takes a holistic approach by systematically examining both consistency and inconsistency relationships among different service providers within the entire cloud system. 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 auto correction that can automatically replace corrupted data processing results produced by malicious attackers with good results produced by benign service providers. Specifically, this paper makes the following contributions: We provide a scalable and efficient distributed service integrity attestation framework for large scale cloud computing infrastructures

2 PRELIMINARY

In this section, we first introduce the software-as-a-service cloud system model. We then describe our problem formulation including the service integrity attack model and our key assumptions.

2.1 SaaS Cloud System Model SaaS

Cloud builds upon the concepts of software as a service and service-oriented architecture, which allows application service providers to deliver their applications via large-scale cloud computing infrastructures. Both Amazon Web Service and Google AppEngine 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 (pi). For example, a disaster assistance claim processing application consists of voice-over-IP (VoIP) analysis component, e-mail analysis component, community discovery component, and clustering and joins components. Our work focuses on data

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

processing services which have become increasingly popular with applications in many 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.

2.2 Problem Formulation

Given an SaaS cloud 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.

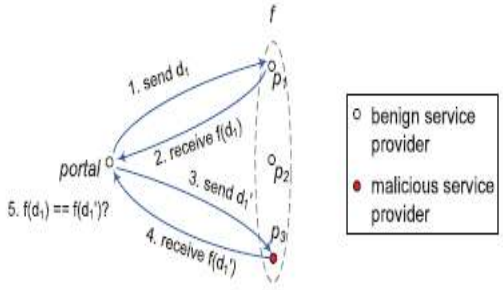

Fig. 2. Replay-based consistency check.

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; and 2) the detection scheme needs to 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

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

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. Assumptions. 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 majorityvoting 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. 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 processin 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.

3 DESIGN AND ALGORITHMS

We first present the basis of the IntTest system: probabilistic replay-based consistency check and the integrity attestation graph model. We then describe the integrated service integrity attestation scheme in detail. Next, we present the result auto correction scheme.

3.1 Baseline Attestation Scheme

To detect service integrity attack and pinpoint malicious service providers, our algorithm relies on replay-based consistency check to derive the consistency/inconsistency relationships between service providers the consistency check scheme for attesting three service

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

providers offer the same service function f. The portal sends the original input data d1 to p1 and gets back the result. Next, the portal sends d0 1, a duplicate of d1 to p3 and gets back the result. The portal sees whether p1 and p3 are consistent. The intuition behind our 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. Note that we do not send an input data item and its duplicates (i.e., attestation data) concurrently. Instead, we replay the attestation data on different service providers after receiving the processing result of the original data. Thus, the malicious attackers cannot avoid the risk of being detected when they produce false results on the original data. Replay-based consistency check. Single tuple processing, we can overlap the attestation and normal processing of consecutive tuples in the data stream to hide the attestation delay from the user. 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. We do not limit the consistency relationship to equality function since two benign service providers may produce similar but not exactly the same results. The credit scores for the same person may vary by a small difference when obtained from different credit bureaus. We allow the user to define a distance function to quantify the biggest tolerable result difference**.**

Definition 1. For two output results, r1 and r2, which come from two functionally equivalent service providers, respectively, result consistency is defined as either r1 ¼ r2, or the distance between r1 and r2 according to user-defined distance function falls within a threshold _. For scalability, we propose randomized probabilistic attestation, an attestation technique that randomly replays a subset of input data for attestation. For composite data-flow processing services consisting of multiple service hops, each service hop is composed of a set of functionally equivalent service providers. Specifically, for an incoming tuple di, the portal may decide to perform integrity attestation. If the portal decides to perform attestation on di, the portal first sends di to a pre-defined service path p1 ! p2 _ _ _!pl

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

receiving the processing result for di, the portal replays the duplicate(s) of di on alternative service path(s) such as p0 1 ! p0 2 _ _ _!p0 l, where p0 j provides the same function fj .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 p0 i. If pi and p0 i receive the same input data but produce different output results, we say that pi and p0 i are inconsistent.

Definition2. 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. We then construct consistency graphs for each function to capture consistency relationships among the service providers provisioning the same function. the consistency graphs for two functions. Note that two service providers that are consistent for one function are not necessarily consistent for another function. This is the reason why we confine consistency graphs within individual functions.

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. We use a global inconsistency graph to capture inconsistency relationships among all service providers. Two service providers are said to be inconsistent as long as they disagree in any function. Thus, we can derive more comprehensive inconsistency relationships by integrating inconsistency links across functions. the global inconsistency rap. Note that service provider p5 provides both functions f1 and f2. In the inconsistency graph, there is a single node p5 with its links reflecting inconsistency relationships in both functions f1 and f2.

Definition4. 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. The portal node is responsible for constructing and maintaining both per-function consistency graphs and the global

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

inconsistency graph. To generate these graphs, the portal maintains counters for the number of consistency results and counters for the total number of attestation data between each pair of service providers.

3.2 Integrated Attestation Scheme

We now present our integrated attestation graph analysis algorithm. **Step 1**: Consistency graph analysis. We first examine per function consistency graphs to pinpoint suspicious service providers. The consistency links in per-function consistency graphs can tell which set of service providers keep consistent with each other on a specific service function. Given any service function, since benign service providers always keep consistent with each other, benign service providers will form a clique in terms of consistency links. **Step 2:**Given an inconsistency graph containing only the inconsistency links, there may exist different possible combinations of the benign node set and the malicious node set. However, if we assume that the total number of malicious service providers in the whole system is no more than K, we can pinpoint a subset of truly malicious service providers. Intuitively, given two service providers connected by an inconsistency link, we can say that at least one of them is malicious since any two benign service providers should always agree with each other.

3.3 Result Auto corrections

IntTest can not only pinpoint malicious service providers but also automatically correct corrupted data processing results to improve the result quality of the cloud data processing service. Without our attestation scheme, once an original data item is manipulated by any malicious node, the processing result of this data item can be corrupted, which will result in degraded result quality.

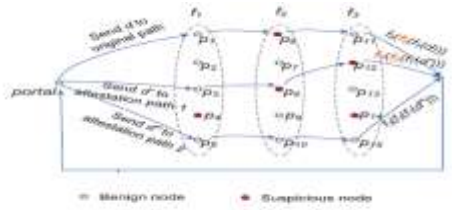

Fig. 7. Automatic data correction using attestation data processing results.

IntTest leverages the attestation data and the malicious node pinpointing results to detect and correct compromised data processing results. Specifically, after the portal node receives the of the original data d, the portal node checks whether the data d has been processed by any malicious node that has been pinpointed by our algorithm.

4 SECURITY ANALYSIS

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

We now present a summary of the results of our analytical study about IntTest. Additional details along with a proof of the proposition presented can be found of the online supplemental material. Given an accurate upper bound of the number of malicious service providers K, if malicious service providers always collude together, IntTest has zero false positive. Although our 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, our approach limits the damage colluding attackers can cause if they can evade detection in two ways. Our algorithm limits the number of functions which can be simultaneously attacked. Our approach ensures a single attacker cannot participate in compromising an unlimited number of service functions without being detected.

5 EXPERIMENTAL EVALUATION

In this section, we present the experimental evaluation of the IntTest system. We first describe our experimental setup. We then present and analyze the experimental results.

5.1 Experiment Setup

We have implemented a prototype of the IntTest system and tested it using the NCSU's virtual computing lab a production cloud infrastructure operating in a similar way. We add portal nodes into VCL and deploy IBM System S stream processing middleware to provide distributed data stream processing service. System S is an industry-strength high performance stream processing platform that can analyze massive volumes of continuous data streams and scale to hundreds of processing elements (PEs) for each application.

5.2. Each node runs multiple virtual machines (VMs) on top of Xen 3.0.3.

The data-flow processing application we use in our experiments is adapted from the sample applications provided by System S. This application takes stock information as input, performs windowed aggregation on the input stream according to the specified company name, and then performs calculations on the stock data. We use a

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

trusted portal node to accept the input stream, perform comprehensive integrity attestation on the PEs, and analyze the attestation results. The portal node constructs ne consistency graph for each service function and one global inconsistency graph across all service providers in the system. For comparison, we have also implemented three alternative integrity attestation schemes: 1) the full-time majority voting scheme, which employs all functionally equivalent service providers at all time for attestation and determines malicious service providers through majority voting on the processing results; 2) the part-time majority voting (PTMV) scheme, which employs all functionally equivalent service providers over a subset of input data for attestation and determines malicious service providers using majority voting.

5.3 Results and Analysis

We first investigate the accuracy of our scheme in pinpointing malicious service providers. Fig. 8a compares our scheme with the other alternative schemes (i.e., FTMV, PTMV, and Run Test) when malicious service providers aggressively attack different number of service functions. In this set of experiments, we have 10 service functions and 30 service providers.

6 LIMITATION DISCUSSION

Although we have shown that IntTest can achieve better scalability and higher detection accuracy than existingschemes, IntTest still has a set of limitations that require further study. A detailed limitation discussion can be found the online supplementary material. We now provide a summary of the limitations of our approach. First, malicious attackers can still escape the detection if they only attack a few service functions, take majority in all the compromised service functions, and have less inconsistency links than benign service providers. However, IntTest can effectively limit the attack scope and make it difficult to attack popular service functions. Second, IntTest needs to assume the attested services are input deterministic where benign services will return the same or similar results defined by a distance function for the same input. Thus, IntTest cannot support those service functions whose results vary

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

significantly based on some random numbers or time stamps.

7 CONCLUSION

In this paper, we have presented the design and implementation of IntTest, a novel integrated service integrity attestation framework for multitenant software-as-a-service cloud systems. IntTest employs randomized replay-based consistency check to verify the integrity of distributed service components without imposing high overhead to the cloud infrastructure. IntTest performs integrated analysis over both consistency and inconsistency attestation graphs to pinpoint colluding attackers more efficiently than existing techniques. Furthermore, IntTest provides result auto correction to automatically correct compromised results to improve the result quality. We have implemented IntTest and tested it on a commercial data stream processing platform running inside a production virtualized cloud computing infrastructure.Our experimental results show that IntTest can achieve higher pinpointing accuracy than existing alternative schemes. IntTest is lightweight, which imposes low-performance impact to the data processing services running inside the cloud computing infrastructure.

REFERENCES

[1] Amazon Web Services, http://aws.amazon.com/, 2013.

[2] Google App Engine, http://code.google.com/appengine/, 2013.

[3] Software as a Service, http://en.wikipedia.org/wiki/Software as

a Service, 2013.

[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] T.S. Group, “STREAM: The Stanford Stream Data Manager,” IEEE Data Eng. Bull., vol. 26, no. 1, pp. 19-26, Mar. 2003.

[7] D.J. Abadi et al., “The Design of the Borealis Stream Processing Engine,” Proc.

Second Biennial Conf. Innovative Data Systems Research (CIDR '05), 2005.

[8] B. Gedik et al., "SPADE: The System S Declarative Stream Processing Engine," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '08), Apr. 2008.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] H. Zhang, M. Savoie, S. Campbell, S. Figuerola, G. von Bochmann, and B.S. Arnaud, "Service-Oriented Virtual Private Networks for Grid Applications," Proc. IEEE Int'l Conf. Web Services, pp. 944-951, July 2007.

[15] M. Burnside and A.D. Keromytis, "F3ildCrypt: End-to-End Protection of Sensitive Information in Web Services," Proc. 12th Int'l Conf. Information Security (ISC), pp. 491-506, 2009.

[16] I. Roy et al., "Airavat: Security and Privacy for MapReduce," Proc. Seventh USENIX Conf. Networked Systems Design and Implementation (NSDI), Apr. 2010.

738 IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 25, NO. 3, MARCH 2014

Fig. 11. Attestation overhead comparison.

[17] J. Garay and L. Huelsbergen, "Software Integrity Protection Using Timed Executable Agents," Proc. ACM Symp. Information, Computer and Comm. Security (ASIACCS), Mar. 2006.

[18] T. Garfinkel et al., "Terra: A Virtual Machine-Based Platform for Trusted Computing," Proc. 19th ACM Symp. Operating Systems Principles (SOSP), Oct. 2003.

[19] A. Seshadri, M. Luk, E. Shi, A. Perrig, L. van Doorn, and P. Khosla, "Pioneer: Verifying Code Integrity and Enforcing Untampered Code Execution on Legacy Systems," Proc. 20th ACM Symp. Operating Systems Principles (SOSP), Oct. 2005.

[20] E. Shi, A. Perrig, and L.V. Doorn, "Bind: A Fine-Grained Attestation Service for Secure Distributed Systems," Proc. IEEE Symp. Security and Privacy, 2005.

[21] Trusted Computing Group, https://www.trustedcomputing group.org/home, 2013.

[22] "TPM Specifications Version 1.2," TPM, https://www. trustedcomputinggroup.org/downloads/specifications/tpm/ tpm, 2013.

[23] J.L. Griffin, T. Jaeger, R. Perez, and R. Sailer, "Trusted Virtual Domains: Toward Secure Distributed Services," Proc. First Workshop Hot Topics in System Dependability, June 2005.

[24] 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.

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

[26] 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.

[27] 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.

[28] Virtual Computing Lab, http://vcl.ncsu.edu/, 2013.

[29] Amazon Elastic Compute Cloud, http://aws.amazon.com/ec2/,

2013.

[30] N. Jain et al., "Design, Implementation, and Evaluation of the Linear Road Benchmark on the Stream Processing Core," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '06), 2006.

[31] B. Raman et al., "The SAHARA Model for Service Composition Across Multiple Providers," Proc. First Int'l Conf. Pervasive Computing, Aug. 2002.

[32] 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.

[33] K.-L. Wu, P.S. Yu, B. Gedik, K. Hildrum, C.C. Aggarwal, E. Bouillet, W. Fan, D. George, X. Gu, G. Luo, and H. Wang, "Challenges and Experience in Prototyping a Multi-Modal Stream Analytic and Monitoring Application on System S," Proc. 33rd Int'l Conf. Very Large Data Bases (VLDB), pp. 1185-1196, 2007.

[34] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Proc. USENIX Symp. Operating System Design and Implementation, 2004.

[35] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks," Proc. European Conf. Computer Systems (EuroSys), 2007.

[36] A. Seshadri, A. Perrig, L.V. Doorn, and P. Khosla, "SWATT: Software-Based Attestation for Embedded Devices," Proc. IEEE Symp. Security and Privacy, May 2004.

[37] A. Haeberlen, P. Kuznetsov, and P. Druschel, "Peerreview: Practical Accountability for Distributed Systems," Proc. 21st ACM SIGOPS Symp. Operating Systems Principles, 2007

Author's profile:

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

Ms. P.Mounika received M.Tech degree from Gokaraju Rangaraju Institute of Engineering & Technology affiliated to JNTUH, Hyderabad. He is currently working as Assistant professor, Department of CSE, in Nalgonda Institute of Technology &Sceince ,Nalgonda, Telangana, India. Her interests includes Web Technologies, Java Programming, Data Base Management Systems.

Ms.V.MAHENDAR received B.Tech Degree from Nalgonda Institute of Technology & Science in Nalgonda. He is currently pursuing M.tech Degree in Computer Science and Engineering specialization in Nalgonda Instituite of Technology & Science in Nalgonda, Telangana, India.