ISSN 2347 - 3983
International Journal of Emerging Trends in Engineering Research (IJETER), Vol. 3 No.6, Pages : 117- 123 (2015)
Special Issue of NCTET 2K15 - Held on June 13, 2015 in SV College of Engineering, Tirupati
http://warse.org/IJETER/static/pdf/Issue/NCTET2015sp24.pdf

Detecting Suspicious Nodes Using IntTest For SaaS Clouds

K.Renuka[1] , N.Phani Kumar[2]
[1]M.Tech CSE Student, S.V. College of Engineering, Tirupati, AP, India
kandrarenuka@gmail.com
[2]Assistant Professor, Dept. of CSE, S.V. College of Engineering, Tirupati, AP, India,
Phani6@gmail.com

ABSTRACT

SaaS provides a flexible environment through which application service providers allowed to host their applications in a distributed environment, so that users can access the hosted services in a easier way. As the environment is sharing in nature there is more scope for SaaS clouds vulnerable to malicious attackers. In this paper IntTest a new integrity attestation scheme is introduced that can use graph analysis scheme to achieve higher Pinpointing of attackers. Here also automatically correcting the results of malicious attackers with results provided by benign service providers technique called auto correction is introduced. Also we implemented IntTest and tested on a production cloud infrastructure, the experimental results show that this scheme achieved higher accuracy than previous schemes. IntTest does not require any secure kernel support and hardware and it also supports for large scale cloud computing infrastructure.

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

1. INTRODUCTION

Cloud computing is emerged mainly to provide two advantages ease-of-use and cost-effectiveness. It is storing and accessing data and programs over the internet instead of your computer hard-drive. Software-as-a-service (SaaS) clouds mainly build on the concepts of software as a service and service-oriented architecture (SOA), this allows different application service providers (ASPs) to deliver their applications via the large cloud computing infrastructure. Some of SaaS clouds are e.g., Amazon Web Service and Google App Engine. In this, paper mainly 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. As, cloud computing infrastructures are shared by ASPs from different security domains, there is a scope for them vulnerable to malicious attackers. For example, attackers can pretend to be authorized service providers to provide fake service components, and the service components provided by benign service providers may contain security holes that can be use by attackers. Here mainly we focuses on service integrity attacks that cause the user to receive corrupted data processing results.

Previous schemes provided various software integrity attestation solutions but those techniques are difficult to deploy for large scale cloud computing infrastructure because those require special trusted hardware or secure kernel support. One of the traditional technique is Byzantine fault tolerance (BFT), it can detect arbitrary misbehaviors using full-time majority voting (FTMV) over all replicas, however it is high overhead to the cloud system.

Here, we present IntTest, a novel integrated service integrity attestation framework for multitenant cloud systems. IntTest does not assume trusted entities on third-party service provisioning sites or require application modifications rather provides a practical service integrity attestation scheme. IntTest builds upon previous work RunTest and AdapTest but can provide stronger malicious attacker Pinpointing power than previous tests. Specifically, both RunTest and AdapTest as well as traditional majority voting schemes need to assume that benign service providers take majority in every service function, assumption makes the test easier to get the solution. To invalidate this assumption multiple malicious attackers may launch colluding attacks on certain targeted service functions, in large-scale multitenant cloud. In order to overcome this, IntTest 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 which is caused by colluding attackers, while the global inconsistency graph analysis can effectively show those attackers that try to compromise 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

ISSN 2347 - 3983
International Journal of Emerging Trends in Engineering Research (IJETER), Vol. 3 No.6, Pages : 117- 123 (2015)
Special Issue of NCTET 2K15 - Held on June 13, 2015 in SV College of Engineering, Tirupati
http://warse.org/IJETER/static/pdf/Issue/NCTET2015sp24.pdf

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.
- We present a novel integrity attestation scheme that can achieve higher Pinpointing accuracy than previous techniques.
- We describe a result auto correction technique that can automatically correct the corrupted results that are produced by malicious attackers.
- We conduct both analytical study and experimental results to quantify the accuracy and overhead of the scheme.

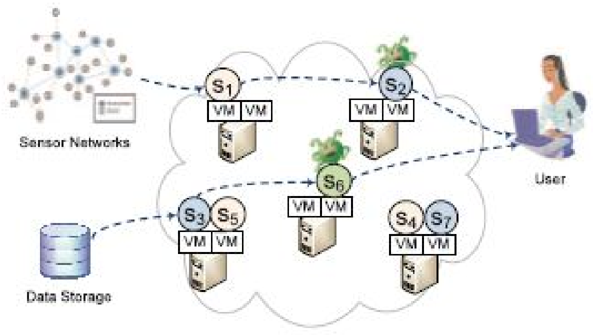


Fig. 1: Service integrity attack in cloud-based data processing.

2. Preliminary

Here we introduce the software-as-a service (SaaS) cloud system model. Then we describe our problem formulation including the service integrity attack model and our key assumptions.

2.1 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.

2.2 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 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 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.

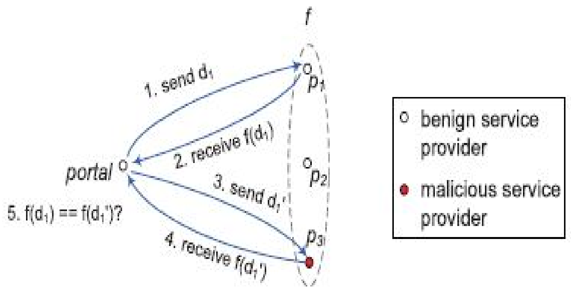


Fig. 2. Replay-based consistency check

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.

3. DESIGN AND ALGORITHMS

In this, we present the basis of the IntTest system: probabilistic replay-based consistency check and the integrity attestation graph model, then describe the integrated service integrity attestation scheme in detail. Also we present the result autocorrection scheme.

3.1 Baseline Attestation Scheme

Our algorithm uses replay-based consistency check to derive the consistency/inconsistency relationships of service providers to Pinpoint the malicious attackers. Fig. 2 shows the consistency check scheme for attesting three service providers p1, p2, and p3 that offer the same service function f. The portal sends the original input data d1 to p1 and gets back the result f(d1). Next, the portal sends d1', a duplicate of d1 to p3 and gets back the result f(d1'). The portal then compares f(d1) and f(d1') to see whether p1 and p3 are consistent."

The suspicion behind our approach is that if two service providers disagree with each other then at least one should be malicious in that note that we do not send an input data item and its duplicates (i.e., attestation data) concurrently. After receiving the processing result of the original data we replay the attestation data on different service providers. Thus, the malicious attackers cannot avoid the risk of being detected when they produce false results on the original data. Although the replay scheme may cause delay in a 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.For all the input data if two service providers give the same output results then, there exists consistency relationship between them. Otherwise, if they generate 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 results but not exactly the same results. For example, the credit scores for the same person may give small difference when obtained from different credit bureaus. To quantify the biggest tolerable result difference we allow the user to define a distance function.

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=r2, or the distance
between r1 and r2 according to user-defined distance function D(r1, r2) falls within a threshold δ

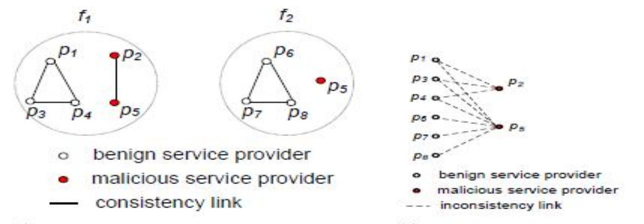
we propose randomized probabilistic attestation, an attestation technique that randomly replays a subset of input data for attestation for scalability. For composite data-flow processing services consisting of multiple service hops, each service hop is composed of a set of functionally equivalent service providers. Specifically, for an incoming tuple di, the portal may decide to perform integrity attestation with probability pu. If the portal decides to perform attestation on di, the portal first sends di to a pre-defined service path p1→ p2 → ..pl providing functions fl → f2 →....→ fl. After receiving the processing result for di, the portal replays the duplicate(s) of di on alternative service path(s) such as p1' → p2'.... → pl',

where pj' provides the same function fj as pj. The portal may perform data replay on multiple service providers to perform concurrent attestation..

After receiving the attestation results, the portal compares each intermediate result between pairs of functionally equivalent service providers Pi and Pi. If Pi and Pi' receive the same input data but produce different output results, we say that Pi and Pi' are inconsistent. Otherwise, we say that Pi and Pi' are consistent with regard to function fi. For example, let us consider two different credit score service providers p1 and p0

1. Suppose the distance function is defined as two credit score difference is no more than 10. If p1 outputs 500 and p1' outputs 505 for the same person, we say p1 and p1' are consistent. However, if p1 outputs 500 and p1' outputs 550 for the same person, we would consider p1 and p1' to be inconsistent. We evaluate both intermediate and final data processing results between functionally equivalent service providers to derive the consistency/ inconsistency relationships. For example, if data processing involves a sub query to a database, we evaluate both the final data processing result along with the intermediate sub query result. Note that although we do not attest all service providers at the same time, all service providers will be covered by the randomized probabilistic attestation over a period of time.

Definition 2: Two service providers who always give consistent output for the same input data during attestation then a consistency link exists between them. In the same way if two service providers who give at least one inconsistent output for the same input data during attestation then an inconsistency link exists between them. After that we then construct consistency graphs for each function to capture consistency relationships among the service providers provisioning the same function. Figure 3(a) shows the consistency graphs for two functions and the service providers are giving the results. 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.



(a) Per-function consistency graphs. **(b) Global inconsistency graph.**

Fig. 3: Attestation graphs.

.

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 solution of detecting and eliminating the resources on the cloud is possible.

3.2 Integrated Attestation Scheme

We now present our integrated attestation graph analysis algorithm.

Step 1: Consistency graph analysis. In order to find suspicious service providers, we first check the per-function consistency graph. In a specific service function, to tell which set of service providers are consistent with each other; we use the consistency links which are available in per-function consistency graph. Benign service providers will not be available in the consistency graph, it is because, the benign service providers will always be consistent for any service provider. For instance, in Figure 3(a), p1, p3 and p4 are always forming a consistency group. So, we can call p1, p3 and p4 as benign service providers.

In traditional schemes, an algorithm has been developed to find out malicious service providers. If the number of benign service providers is greater than that of the malicious ones, a benign node will always stay in a network formed by all benign nodes, which has size larger than ⌊k/2⌋, where k is the number of service providers provisioning the service function. Thus, we can find out malicious nodes by identifying nodes that are outside of all groups whose size is larger than ⌊k/2⌋. To understand it clearly, consider Figure 3(a), where, p2 and p5 are identified as suspicious because they are excluded from the clique of size 3.

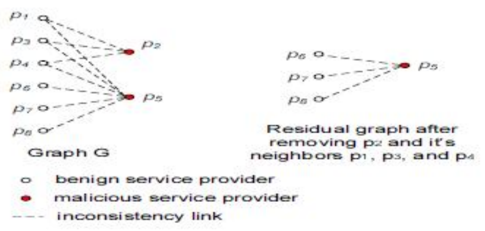

p_1
p_3
p_4
p_6
p_7
p_8
p_2
p_5
Graph G
p_6
p_7
p_8
p_5
Residual graph after removing p_2 and it's neighbors p_1, p_3, and p_4
○ benign service provider
● malicious service provider
--- inconsistency link

Fig. 4: Inconsistency graph G and its residual graph.

Strategically saying, In order to escape the detection, the attackers will try to take the majority in specific service function after the colluding. Therefore, it is not very efficient to examine the per-function graph alone. In order to overcome this drawback, we have to integrate the consistency graph analysis with the inconsistency graph analysis so that we can get wholesome on integrity attestation.

Step 2: Inconsistency graph analysis. There may be different combinations of the benign node set and the malicious node set if there is any inconsistency graph, which contains inconsistency links. If the total number of malicious service providers in the whole system is not greater than K, then, a subset of truly malicious service providers can be identified. If we want to find any one among the given two service providers as malicious, then, we have to connect them by inconsistency link. Since the two service providers should always agree with each other By examining the minimum vertex cover of the inconsistency graph. we can derive the lower bound about the number of malicious service providers. The minimum set of vertices such as each edge of the graph is incident to atleast one vertex in the set, can be identified through minimum vertex cover graph.. For instance, in Figure 3(b), The minimum vertex cover can be formed by p2 and p5. As to get this approach clearly, we present two propositions. These prepositions are explained clearly as follows:

Proposition 1: Given an inconsistency graph G, let CG be a minimum vertex cover of G. Then the number of malicious service providers is no less than |CG|. We now define the residual inconsistency graph for anode Pi as follows.

Definition 5: The residual inconsistency graph of node Pi is the inconsistency graph after removing the node Pi and all of links adjacent to Pi.

For example, Figure 4 shows the residual inconsistency graph after removing the node p2. Based on the lower bound of the number of malicious service providers and Definition 5, we have the following proposition for pinpointing a subset of malicious nodes.

Proposition 2: Given an integrated inconsistency graph G and the upper bound of the number of malicious service providers K, a node p must be a malicious service provider if and only if
|Np|+| CG′ p'| > K

Where |Np| is the neighbour size of p, and |CG′p | is the size of the minimum vertex cover of the residual inconsistency graph after removing p and its neighbours from G.

Step 3: Combining consistency and inconsistency graph analysis results. Let Gi be the consistency graph generated for service function fi, and G be the global inconsistency graph. Let Mi denote the list of malicious nodes by analyzing per function consistency graph Gi, and Ω denotes the list of suspicious nodes by analyzing the global inconsistency graph G, given a particular upper bound of the number of malicious nodes K. We examine per-function consistency graphs one by one. Let i denote the subset of that serves function fi. If Ω∩ Mi≠∅, we add nodes in Mito the identified malicious node set. The idea is that since the majority of nodes serving function fi have successfully excluded malicious nodes in i, we could trust their decision on proposing Mi as malicious nodes.

3.3 Result Auto-Correction

Instead of Pinpointing malicious service providers the IntTest automatically correct the corrupted data processing results in order to improve the result quality of the cloud data processing service. There is no need of our attestation scheme, because if the original data item is manipulated by any malicious node then the processing result of that data item can be corrupted, it will affect the result by degrading the result quality. IntTest gives the advantages that it take the result of attestation data and the malicious node compromised data it takes and automatically correct the processing result.

Mainly, after receiving the result f(d) of the original data d by the portal node, it needs to check the data d which has been Pinpointed by our algorithm whether it has been processed by any malicious node or not ? The received data is checked by the portal node. If the data d has been processed by any Pinpointed malicious node, it is labelled f (d) as "suspicious result". Again the portal node needs to check that that d has been chosen for attestation, if yes, then again we will check the attestation copy of d only traverse good nodes. To replace f(d), we use the result of attestation data but only then if attestation copy of d traverse good node. For example you can clearly observe in Figure 5 the Pinpointed malicious node s6 is processing original data on the other hand benign node is processing one of its attestation data d.The original result is replaced by the attestation data result f(d").

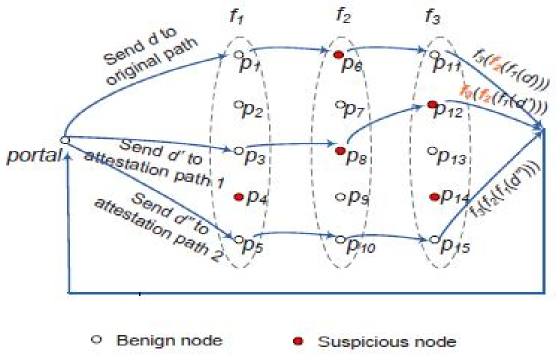


Fig. 5: Automatic data correction using attestation data processing results.

4. CONCLUSION

In this paper, we have discussed about the design and implementation of IntTest, an innovative way of integrated service integrity attestation framework for multi-tenant software-as-a-service cloud systems. Without distinguishing high overhead to the cloud infrastructure, IntTest handles randomized replay-based consistency check to verify the integrity of distributed service components. In order to identify the intriguing attackers more adequately than existing techniques, IntTest performs integrated analysis on both consistency and inconsistency attestation graphs. Apart from that, IntTest also provides auto correction technique, which is used to correct the compromised results mechanically and spontaneously. IntTest has also been implemented and tested on a commercial data stream processing platform, which has been running inside a virtualized cloud computing infrastructure. These tests showed that IntTest has achieved greater Pinpointing accuracy than most of the existing alternative schemes. Finally, from this paper, we conclude that IntTest is light-weight, which assess an impact to the data processing processing services which runs inside the cloud computing infrastructure.

5. REFERENCES

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

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

[3]SoftwareasaService,http://en.wikipedia.org/wiki/Software asa Service, 2013.

[4] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, Web ServicesConcepts, Architectures and Applications (Data-Centric Systems andApplications). Addison-Wesley Professional, 2002.

[5] T. Erl, Service-Oriented Architecture (SOA): Concepts, Technology, andDesign. Prentice Hall, 2005.

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

[7] D.J. Abadi et al., “The Design of the Borealis Stream ProcessingEngine,” Proc. Second Biennial Conf. Innovative Data SystemsResearch (CIDR ’05), 2005.

[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.

ABOUT AUTHORS

K.Renuka received the B.Tech Degree in Computer Science and Engineering from SV College of Engineering, University of JNTUA in 2013. She is currently working towards the Master’s Degree in Computer Science and Engineering, in SV College of Engineering University of JNTUA. Her interest lies in the areas of Cloud Computing,

N.Phani Kumar Received M.Tech Degrees in Computer Science and Engineering from in Siddhartha Institute of Engineering and Sciences, puttur, JNTUA, in 2011 and Master of Computer Application in 2008 respectively. Currently he is an Assistant Professor in the Department of Computer Science and Engineering at SV College of Engineering-Tirupati

.

He has published a paper titled **“A Flexible Rollback Recovery in Dynamic Heterogeneous Grid Computing”.** in Journals and refereed Conference Proceedings. Her current interests include Data Mining, Big Data, Computer Networks and Information.