Cloud Security

Privacy through Data Isolation in Multi-tenancy Environment

Soumadip Biswas
School of Information technology
Indian Institute of Technology
Kharagpur, India
Soumadip.cse@gmail.com

Vaibhav Vikas Neharkar
School of Information Technology
Indian Institute of Technology
Kharagpur, India
nvaibhavv@yahoo.com

Affiliation
[Cloud computing, security, privacy, multi-tenancy]

Abstract
[This paper takes one very important aspect of cloud computing namely multi-tenancy and talks about some security/ privacy issues during implementation in the problem definition segment, then we have talked about one existing solution to the problem in the existing solution segment, then we have taken one particular privacy issue that had been seen while providing data access isolation to the tenants of the system in that proposed architecture and we have provided one proposal that actually encounters the problem and provided a model of the solution]

I. Introduction

In Cloud we can use of collection of services, application, and infrastructure (e.g. servers, storage devices) to develop and/or store the data over the Internet. We can access this data from anywhere, anytime using any device (desktop, laptop, mobile etc.) which is having Internet facility on it. These components of cloud can be extended or shrink i.e. scaled up or down at anytime on demand, also in cloud we pay for what we use on periodic basis depends on Cloud Service Provider (CSP) like electricity.

Purchasing, monitoring, managing and controlling H/W or S/W to start up the business is costly task. Cloud Computing provides solution to this problem using Multi-tenancy. Multi-tenancy allows single copy of S/W or H/W to be shared by multiple users thus bringing down the overall cost of IT infrastructure. Cloud Providers uses virtualization technique (also called as multi-tenancy) to effectively manage resources. In these technique different users assigned different virtual instance which may refer to same copy of application (S/W or H/W) located on same physical machine.

There are several other issues in cloud, but in this paper we have focused on architecture of achieving multi-tenancy at the SOA level, which enables users to run their services in isolation in multi-tenant SOA framework [1].

Rest of the paper is organised as follows. In section II we have presented our problem definition in which we have discussed about Data isolation problem in multi-tenant SOA framework. Then in section III we have discussed about existing solution to solve this problem. In section IV we have discussed issues with existing solution and proposed some new solution to overcome these issues.

II. Problem Definition

In SOA multi-tenancy problem, as identified by Chang et al. [3] in his architecture there are metadata services, data services, process and business services, security services, and presentation components as different aspects of the architecture. These services are grouped in to Execution, Security, and Data [1]. In which execution involves developing, deploying and running services, which are often implemented as Web services, and composing those services together to create higher level artefacts like Business Processes, Work-flows and Mash-ups [1]. These execution use or store the data from registry or database or metadata in registry. The structure of this architecture is as shown in fig.

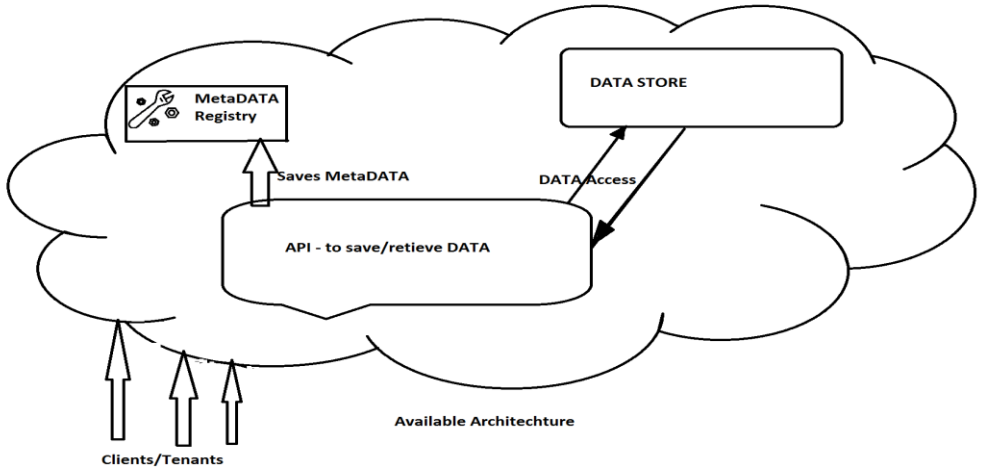

Figure - 1

The second and most important thing is security service, which defines authorisation and ownership for data. The concept of multi-tenancy is to provide each user its own tenant i.e. isolation in shared environment with maximum utilization of resources. But in the multi-tenant system there is always risk of compromising of tenant isolation of user by some attacker due to malfunctioning component or an inadvertent programming error. Also in multi-tenant user should be able to configure its work like add/modify/delete its own users, data and services. But these users should be kept away from admin functionality. Thus in short application should allow each user to configure its own resources without compromising isolation of other.

III. Existing solutions

Initial works that has been done on multi-tenancy is actually in the context of Application Service Providers i.e. ASPs during late 1990s and early 2000s. The multi-tenancy concept is actually talks about providing multiple tenants to access at the same time with effective resource sharing and efficiency, which indeed provides maximum utilization of available resources.

In this context some models has been proposed earlier, some of them are

- Model by Chang et.al.
- Model by Jacobs et. al.
- Model by Guo et.al. etc.

These models all talk about multi-tenancy implemented over a database. Jacobs et. al. and Chang et. al. models have outlined three main approaches for data management : separate schemas, shared databases with separate schemas and shared databases with shared schemas. Amongst only the approach with shared database gives the tenant a separate database of its own.

Chong et. al. in their model have talked about maturity levels of the multi-tenancy implementations where the higher the level is the higher the level of resource sharing.[3]

- Level 1 → this level allows one instance per user but its ad-hoc i.e. randomly done, no relation between two users.
- Level 2 → this level again allows one instance per user but in more controlled way, they all are provided same type of instances but at the same time they are given the ability to configure
- Level 3 → this level actually allows to create only one instance and all user will be connected to this only instance, this broadly can be applied for only SaaS and PaaS type of services.
- Level 4 → finally this level offers a load-balancer which acually the same with the level 3 but the load-balancer module gives dynamic load balancing whenever the load on one instance is high.

One related solution is provided by virtualization technologies, where separation is done at the hardware level. Although use of virtualization enables different virtual hosts to share a common set of resources, each host has to run its own operating system, and therefore, the solution incurs much overhead than server level multi-tenancy. Similar arguments can be made against running isolated processes within a single operating system - such as by running multiple JVM instances for Java applications.

Here is another approach taken by Afkham Azeez et.al. in their paper “Multi-tenant SOA Middleware for Cloud Computing” [1] where their approach addresses scalability and load balancing with a load balanced elastic approach. Configurability is provided through a management portal and is server-side tenant specific customizations. Here in high level

multi-tenancy Architecture they have grouped different aspects of the architecture under Execution, Security and Data.

This solution was implemented on top of the WSO2 Carbon platform which is a componentized open source middleware framework for building scalable, high performance servers. Here is a brief of the architecture [1]

- A WSo2 Carbon Platform → WSO2 Carbon is a Java based componentized platform for building servers which offers an extremely powerful framework for building middleware servers.
- Service Deployment → All service hosting components of WSO2 Carbon (including JAX-WS and Apache Axis2 services and BPEL processes), have Apache Axis2 as its underlying execution engine. The architecture of Apache Axis2 (Axis2) was inherently designed to support multi-tenant deployments.
- Message dispatching → Any and every message are handled by some means of dispatchers defined in master configuration. Cross-tenant communications are possible by using unique URL for every tenant.
- Security → User management, Authentication and Authorization, and Super-tenant authorization are three security policies that they have used.
- Service Execution → This section actually implements to achieve execution isolation among the users of the system, so that no malicious code from any user can access other user's data. This has been implemented by providing unique URL to every user's application and thus separating them out.
- Data Access → The WSO2 Carbon platform provides a registry/repository that abstracts out the relational database layer by providing a set of APIs of its own, which can store and retrieve persistent data with tenant-wise data-separation logic. But here it doesn't allow the tenants to access the database directly, the access is somewhat restrictive which is quite apt for some scenario but in general it is required to give the tenants direct access to the database.

IV. New suggestion

In the above last stated solution as proposed by Afkham Azeez et.al. in their paper "Multi-tenant SOA Middleware for Cloud Computing" to the picked problem statement of multi-tenancy it is clear enough that the clients/tenants are not given permission to access the database directly, but through WSO2 Carbon APIs which is sufficient for many scenarios, it is still necessary to provide direct database access in a tenant-safe manner.

To encounter this problem we are proposing a modification to one aspect of the architecture which is definitely the database access module.

Here the following figure shows our proposed model.

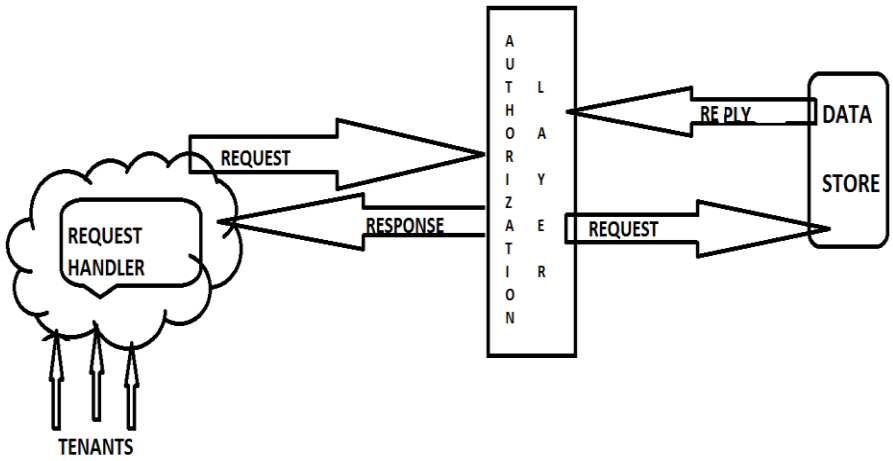

Figure - 2

This is actually about adding a new layer to the given architecture, we call it "Authorization Layer" and this layer actually sits between the service provider and the data store and handles request / response to the data store. We have one module also added to the service provider which actually communicates with this layer; we call this module "Request Handler", details about this module is provided shortly. This module broadly sends data service request to this layer. This layer actually checks the authorization of the user associated with the request, and then serves accordingly i.e. creates proper query, activates proper area of the data store and fires the query, then the request again returned as a service response to the "Request Handler" module.

Broadly we can consider this Authorization layer along with the data store and the services provided them constitutes another service called Data as a Service. Thus the data server can be separated from the main service provider and can be handled much more efficiently and effectively.

Here in the following figure the model of the request handler is explained.

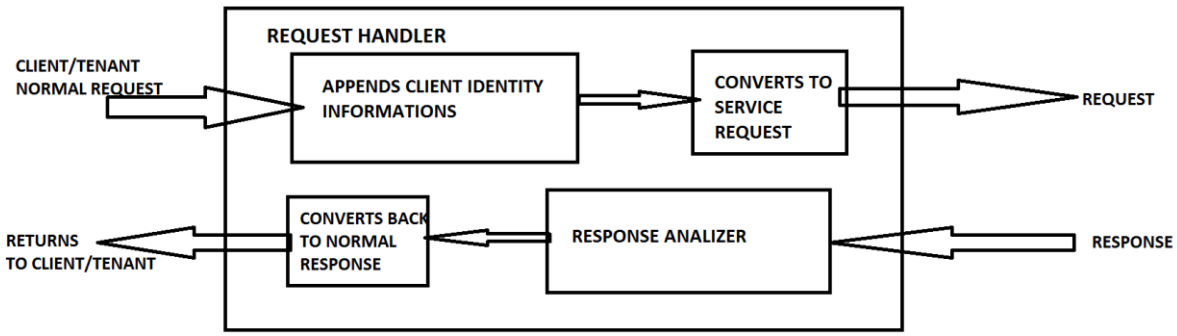

Figure - 3

Basically this request handler works as a transparent layer to the tenants/ clients. This module consists of four sub-modules which functionalities are as follows

- First sub-module captures any request from the user for data base which will be input from the user as normal query. Then this module adds user authentication information with the query for the serving module to identify the actual user who is querying and then it will be passed to the next sub-module.
- Second sub-module will take output from the previous module and convert this query to another service request and send this outside of this module.
- Third sub-module actually fed the response generated from the Authorization Layer, which is analyzed by this entity and forwarded to next section in the row.
- Fourth and last sub-module takes the response from the previous sub-module and convert this response to normal query-response and simply returns to the user.

Thus this module will actually stay invisible to the user and will give client a normal and isolated access to the database as desired.

V. References

[1] Afkham Azeez, Srinath Perera, Dimuthu Gamage, Ruwan Linton, Prabath Siriwardana, Dimuthu Leelaratne, Sanjiva Weerawarana, Paul Fremantle WSO2 Inc. Mountain View, CA, USA, "Multi-Tenant SOA Middleware for Cloud Computing", "2010 IEEE 3rd International Conference on Cloud Computing", pp 468-465.

[2]Dean Jacobs, Stefan Aulbach Technische Universität München Institut für Informatik - Lehrstuhl III (I3) Boltzmannstr. 3 D-85748 Garching bei München, "Ruminations on Multi-Tenant Databases".

[3] F. Chong and G. Carraro, "Architecture strategies for catching the long tail," MSDN Library, Microsoft Corporation, 2006.

[4]Dean Jacobs, Stefan Aulbach Technische Universität München Institut für Informatik - Lehrstuhl III (I3) Boltzmannstr. 3 D-85748 Garching bei München, "Ruminations on Multi-Tenant Databases", "Ruminations on Multi-Tenant Databases", pp 1-5.

[5]Amarnath Jasti, Payal Shah, Rajeev Nagaraj and Ravi Pendse,Department of Electrical Engineering and Computer Science, Wichita State University 1845 N Fairmount, Wichita, Kansas 67260 USA, "Security in Multi-Tenancy Cloud", "Security Technology (ICCST), 2010 IEEE International Carnahan Conference ", pp 35.