An Approach towards Data Isolation & Security in a Multi-Tenant Cloud Environment

Research Report

***Abstract*—**

In this era, we observe a paradigm shift in which software is used as a service by customers in the cloud. Realistically, we are far ahead of that as of this moment, solving problems faced by enterprises while they embrace cloud technologies is the main concern. One group of such issues comes with multi-tenancy. Multi-tenancy is a resource sharing capability in Cloud Computing that allows one instance of an application to be shared by multiple customers and therefore achieving high operational efficiency alongside saving costs. This gifted technology has certain disadvantages such as multiple security concerns where confidentiality and integrity can be violated. As software on demand services such as Saas becomes more popular, providers must ensure their services offer quick provisioning, up and down scaling, availability, flexible downtimes and a significant quality of service. After achieving these, the provider must ensure the cost of services is optimal to stay ahead in the competition. But the crucial focus points remain to be the security of the data as well as the overall performance of the system in a cloud-based multi-tenant environment. While many approaches towards solving these issues have been proposed, solutions that battle enhanced attacks and emerging performance issues due to limited resources still remain in the dark. This paper is devoted in proposing a new data isolation approach that implements unique security boundaries using ASP and a tenant request-limiting architecture through a customization approach in real time.

Keywords: *Multi-Tenancy, Cloud Computing, SaaS, IaaS, Data Security, Data Isolation, Side-Channel Attacks.*

1. INTRODUCTION

Looking back at how technologies have changed over time, computing has shifted from a very large device covering a room to PDAs. With the invention of networking, the use of computer has increased tremendously and now almost all organization use computer and network to run their daily operations. Growing use of technology has added fuel in the advancement of technology and of which cloud computing is one solid example. Simply put, cloud computing is a resource sharing mechanism where vendor provides computing resources like servers,

storage, databases, networking, software, analytics and intelligence and allows multiple users to share the resources over the internet.

Figure 1: Simple Cloud Computing Environment

Figure 1. represents a simple structure of a cloud environment. Various devices can access the same resources from different locations at the same time while the resources are in a cloud environment. Such remote access of resources can be done using the web browser over the internet. Cloud computing facilitates sharing of these computing resources through multi-tenancy, virtualization and data-isolation mechanisms.

As defined by National Institute of Science and Technology (NIST), cloud computing is a model that enables convenient and ubiquitous access to a shared pool of configurable computing resources that can be rapidly setup or removed on users' demand with little or no service provider intervention. Cloud computing is a mechanism with which users are given remote access to data and program applications using a web browser over the internet. This facilitates organizations to simply rent the resources required for computing the from the service provider rather than owning their own computing infrastructure or data centres. This helps in the rapid deployment, resources flexibility and provides economies of scale by lowering operating cost by avoiding the upfront cost and simplifies the computing process removing all the complexities of the infrastructure set up. From on-demand self-service to effective virtualization to location and device independence, cloud computing has various advantages and multi-tenancy is one of the most prominent advantage that cloud has provided in the modern-day business.

Multi-tenancy, a crucial component of cloud computing, is a resource sharing capability without which cloud services would become far less popular than they are now. It allows users to share computing resources ensuring their data is kept secured from all other users. Multiple instances of same resource are created through virtualization which are then accessed by tenants from various locations at the same time. Multi-tenant applications are distributed applications based on the tenant-isolated services or multi-tenant microservices. Tenants share computing resources in private or public cloud while their data remain isolated from each other and invisible to any unauthorised parties.

Multi-tenancy is designed and incorporated differently in each of the service models of cloud infrastructure:

Multi-tenancy in IaaS: Cloud Service providers such as Amazon, Skytap, Sun etc offers Infrastructure-as-a-service when integrated with multi-tenancy and can provide computing models such as SQL, MR and storage space capability of resources as well as CPU cycles, network bandwidth.

Multi-tenancy in SaaS: SaaS applications in Multi-tenant format can serve several tenants at the same time. Data is segregated in distinct partitions belonging to one different tenant. One application can serve several customers at a time, so data isolation is a critical metric to ensure safety in the cloud.

Multi-tenancy in PaaS: Multi-tenant PaaS contributes to providing Development tools i.e. software code editor, compiler etc., Operating systems, Database management system. Prominent service providers are Google, Salesforce.com, Azure and so on.

2.1 Overview of the Topic

2.1.1 Advantages of Multi-tenant architecture:

1. **Low Cost of Investment**: Customers in a multi-tenant architecture, use the same resources, functionalities and database, so the cost of overall development and management is lowered down extensively while maintaining service for everyone. No reparations needed for data customization and addition of new features.

2. **Smooth access to new releases**: Prior to multi-tenancy, if there was a new version of the software out, the developers had to modify the instances for each tenant by making sure their needs are met. Earlier, each instance needed to be tested by test engineers before deployment and it was overall a very lengthy process. With the integration of multi-tenancy, it is a very smooth process, as the code must be modified at one place and multiple tenants in the system can access it immediately. No chance of competitors getting updated service leaving the uprising tenants behind.

3. **Resource sharing**: Tenants in the system can all benefit from each other by shared hardware and software resources. Tenants get access to shared software and database at an affordable cost. Consequently, the shared components are all utilized properly. While maintaining the Service Level Agreement (SLA), it is crucial for a cloud service provider to ensure balanced resource consumption among the tenants so that performance isolation can be maintained at all hours.

4. **Reduced Maintenance cost**: Earlier, customers had to pay a huge sum for regular maintenance in order to have a safe and sound platform. With the addition of shared resources, the overall maintenance cost is shared among all the tenants equally. The new feature updates are applicable to all the instances of the component, so everyone is capable of accessing the renovated product while sharing the cost

2.1.2 Issues

Multiple independent instances of one or more applications operate in a shared environment and this mode of operation of software is referenced as multi-tenancy. In such shared environment, instances are logically isolated but integrated physically and although the degree of logical isolation must be complete, the degree of physical integration varies. We can think of the banking system with respect to multi-tenancy in cloud. Customers throughout the world store their assets in banks but are unaware of the integration details about the stored assets. Similarly, tenants can share the same functionalities and infrastructure but should be unaware of each other's presence and commodities. Figure below provides a simple explanation on how multi-tenant services are made up of in cloud.

Figure 2: Multi-tenant Cloud Computing Architecture

The concept of Multi-tenancy is very crucial in a successful cloud computing platform. Hundreds and thousands of business and enterprises across the globe uses similar operational structure and all of them can immensely benefit from each other if they are in a shared platform. The foremost advantage of multi-tenant architecture is that all the tenants have equal opportunity to evolve by accessing the newest functionalities being integrated in the cloud, the industry grows overall, and no-one is left behind. Even though software, hardware and microservices are being shared equally in the cloud, data and specific components to a tenant needs to be isolated from the others. In other words, multi-tenancy must ensure that all the

tenants are unaware of each other's location and presence in the cloud and resources specific to the tenants are confidential and can't be accessed without permission.

Though multi-tenancy has a lot of benefits including resource sharing, cost reduction, on demand self-service, faster deployment and many more, it often must overcome a lot of issues so that users can enjoy a secured service with no disruption. Some of the issues that a multi-tenant application environment could encounter are discussed below:

2.1.3 Limitations

Limited flexibility:

When an enterprise decides to go for the cloud service, they have the option to choose whether to use software as a service (SAAS) or platform as a service (PAAS) or infrastructure as a service (IAAS) depending on their resource need. In one hand, they can make their choice to use either of the service while on the other hand, the choice they have is very limited. Any user wishing to use cloud service are bound to use the limited option provided by the cloud vendors even though it might not meet their customized needs. All organization might have their own customized needs, providing each of them with their varying needs is not what multi-tenancy does. For example, a user might want to use software as a service (SAAS) option while they want their data to be managed in-house. In a multi-tenant environment, they do not have such option as it requires customization. Hence it has limited flexibility in terms of resource availability and services offered.

Security Isolation

Multi-tenancy brings a huge challenge in terms of security. Security breach in multi-tenant environment means exposure of millions of data and information. Multiple users using the same system should have privacy and confidentiality which is why data management is very crucial. Although vendors focus highly on the security of clients' data and information, it is often a challenging factor. In order to ensure security in a multi-tenant environment, they need to isolate data of each client from each other and while doing so they need to make sure that there is not any unauthorized access or that there is no conflict or interference among the tenants.

Data interference

With multi-tenant application in service, providers are required to store data from multiple customers at the same place while isolating each customers' individual data. Due to the large number of data being stored at one location, effective data isolation might become problematic. In absence of proper isolation mechanism, tenants might lose their data and might have to go through multiple attacks and breaches. Lack of proper data isolation in multi-tenancy makes it vulnerable to public attacks.

Performance interference

The main idea behind performance isolation is that the performance of different tenants complies with their service level agreements (SLAs) and preventing one tenant from adversely affecting the performance of another tenant. Due to high level of resource sharing, performance isolation is often complicated in a multi-tenant environment. So, providers must keep track of each of their clients' resource usage summary and monitor continuously to ensure their performance is guided by their SLA. This requires a lot of time and effort and might not be as fruitful.

Side Channel Attack

In a multi-tenant environment, a side channel is an information leakage channel which is exploited by an attacker to gain an access to other tenants' data and applications. These attacks feed on the resource sharing feature of the cloud. By observing the changes i.e. response time, power consumption within the side channel operations happening in shared hardware resources, the attacks are made by creating interruptions in the events and thus obtain sensitive

data such as cryptographic keys. There are variations in different side channel attacks that take place such as: Timing attack, Cache attack, Power-monitoring attack, Electro-magnetic attack etc. Consequently, these attacks usually happen within the system and are very hard to detect which makes them a notorious threat to organizations residing in the system. Recent revelation of attacks called Meltdown and Spectre have caused a frenzy among different organizations. These attacks directly the CPU's health and extracts information based on conjectures while monitoring the CPU functionalities.

Resource Sharing

Resource sharing is the first benefit a cloud service would promise to its users. Multiple instances of single resource are created so that multiple users can access the resources at the same time without having to wait for the resources to be released by other users. This facilitates sharing of resources so that hardware, software and maintenance cost is lowered and that a cloud service provider can operate on lower cost making the services cheaper for its customer. However, the performance of the cloud system depends on how well the resource sharing mechanism is built and how securely clients are protected from each other. Thus, resource sharing could hamper the performance of the cloud at some point of time if the monitoring is not properly managed regarding which client needs which resource at what point of time.

Capacity Optimization

One of the main issues multi-tenancy brings in cloud computing is capacity optimization. As the concept states that the tenants share resources, it is the duty of database administrator to understand which tenant should be deployed which network in order to maximize capacity and reduce costs. Complicacy lies in the fact that they need to align capacity in accordance to the business demand continuously and are required to efficiently manage the actual and forecasted resource utilization for all their servers. As the modern IT environment is ever changing and dynamic, it is very difficult for the administrator to track the capacity requirement and hence often fail to optimize it properly.

Service delivery and high availability

A cloud provider should maintain high availability in order to deliver required service to the users. While service delivery and high availability has been the important characteristics of cloud, it could be one of the challenges as failures may occur during delivery of service due to abnormal loads generated which could interrupt the service hindering the availability. This could be very disastrous because unavailability of multi-tenant application means loss of services to multiple clients' relying of a single cloud provider for their regular operation. In order to ensure the proper delivery of services and that the resources are available when requested, continuous monitoring of service delivery and its availability is very critical.

2.1.4 Potential Solutions

Monitoring:

The best solution to all the issues that multi-tenancy might face is the effective and continuous monitoring. Proper monitoring helps in locating the problems and ensuring downtimes are reduced. Present day information technology environment is extremely vibrant with complex operations which require sophisticated altering capabilities. While one client might require more data storage, other might need higher server capacity. One client might be uploading their data to the server and at the same time, some other client might be downloading data from the server. The most excellent way of coping through all these changes is to monitor the activities continuously. The vendor should keep track of what clients are doing all the time so whenever they see any disrupting activity, they can swiftly act upon it and overcome the likely damages it could bring in the service delivery, security, privacy and high availability.

Separation of duties:

In the areas of IT, separation of duties refers to the ability of a system to divide a task or function or a component into multiple areas of responsibility which are then assigned to

multiple roles or multiple individuals. This ensures reduction in the conflict of interest and avoids a single person from misusing their authority to play foul to disrupt the service or to breach the security mechanism. For this, the roles have to be defined and appropriately clarified not only for the administrator but also for the customers so that the services offered by the cloud provider are secured and that their customer's authorities are limited not to hamper any particular resource or data domain. In a multi-tenant environment where tenants may have different reliance on the cloud security provider's role in their security management, ambiguity might occur due to the confusion in the role of provider and customer in security of data. The best way to solve the ambiguity that might occur is to introduce enterprise single sign-on (ESSO) or identity and access management (IAM).

2.2 Summary of Multi-tenant Design Approaches

Proposed approaches include optimal placement of a new tenant inside a shared database while keeping the rest of the space for other new tenants in a flexible manner (Kwok & Mohindra 2008). Salesforce, pioneer of multi-tenancy solutions, have also discussed how they handle multi-tenants in their application and the design of their platform Force.com (Weissman & Bobrowski 2009).

Ochei, Petrovski & Bass (2015) implemented three multi-tenancy patterns based on degree of isolation. Among shared component, tenant-isolated component and dedicated component, their study suggests dedicated components provides the highest degree of isolation. A known entity of Database-as-a-service which is a Saas category is Microsoft SQL Server. A study was conducted to find out multiple database patterns to assist the implementation of multi-tenancy in this server by Chong, Carraro & Wolter 2006).

SQLVM, a prototype in Microsoft SQL Azure was developed to find better performance isolation than its competitors (Narasayya et al. 2016). A Saas multi-tenancy technique that derives tenant context from intercepting web requests was provided by Cai, Wang & Zhou (2010). They also implemented separate systems for tenant, Saas application developer, administrator and the operator.

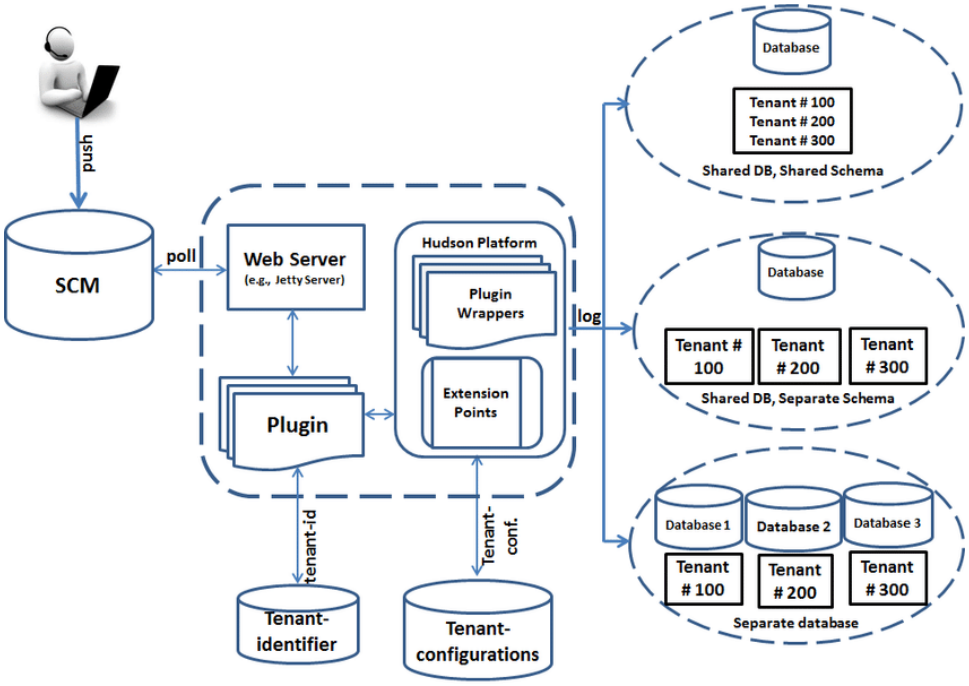

Fig 3. Multitenancy Data Isolation Architecture

Significance << 5 marks need 2 pages more

Multi-tenancy acts as a backbone of cloud computing and is the ideal architecture to make public cloud environment. Multiple tenants share same software infrastructure but the data and configurations that are specific to a tenant are stored in separate and secure container. With sharing comes the reduction in cost. Instead of having to configure everything on their own, organizations can now rent the resource and instead of providing separate resource to each client, vendors can simply create multiple instances of a single resource through virtualization and provide each client with the resource at the same time without burdening their own resource. In addition to this, multi-tenancy helps in obtaining economies of scale. Tenants can share same basic software and if scaling is required, they can obtain it with fewer additional infrastructures thus by lowering the cost with economies of scale.

While multi-tenancy is one of the most significant features of cloud, it has its own significance. It enhances the scalability of public and private clouds and provides a better return on investments for organizations. Furthermore, it also eases the maintenance and updates for the tenants as everything is done on the providers end. Because most of the multi-tenant solutions are designed in a way that they are highly configurable, it makes organizations easier to make application perform the way they want without having to change the underlying code and data structure. And when codes remain unchanged, it is easier to perform upgrades. Multi-tenancy offers ease of operations to the users and reduce their delivery cost.

Partition in multi-tenancy is done internally and needs to track individual configuration for each client. Multi-tenancy can support large number of tenants however, this comes with the price, as the scalability level increase, the isolation level decrease. As the tenants share the same software and possibly the database, this architecture needs to prevent the Quality or Service(QoS) from being affected by other tenants.

We have previously discussed the induction of security threats by multi-tenancy in cloud services. Security attacks are everchanging and no certain approach can commit as an overall

detects to such attacks. Organizations are switching to shared cloud-based systems in order to achieve lesser IT costs. Many risks and countermeasures revolving such systems have been discussed in the past. A significant idea is a "Virtual Private Cloud". This provides a certain degree of isolation between organizations inside a public cloud. Different consumers have a configurable platform where they can choose what they need on-demand from a pool of available resources. Examples of such services provider are Amazon Web Services, Microsoft Azure, etc. (Narula, Jain & Prachi 2015). A virtual private cloud solves many issues regarding the cloud environment but, when the intentions of these consumers are known, auditing all administrative access to the system the wise decision to be made. Although numerous approaches have been introduced in the past in monitoring and controlling such access requests, the grey area of not having a system that do not learn from past security threats and implement them on-the-go needs to be explored substantially (Brown, Anderson & Tan 2012).

4. SUMMARY OF EXISTING SYSTEMS

With the multiple advantages of multi-tenancy, comes its issues. A major challenge can be pointed to data interference between tenants commonly data security breaches. Tenants in such architectures use resources from it. Thus, a higher degree of resources consumed by one tenant can affect the resource consumption of another. This in turn affects performance. Such issues have been addressed by many over time since the first introduction of multi-tenancy in the cloud.

4.1 EXISTING SOLUTIONS

On an investigative research on EC2 instances provided by the pioneer cloud services provider Amazon Web Services, an approach named "Bolt" was tested. 20 users launched their application instances secretly. Bolt detected the resource type of 277/436 jobs while stating their launch status. Moreover, once a victim application is located, performance can be made to deteriorate (Delimitrou & Kozyrakis 2017).

On a separate attempt on Google's trace logs dataset, a model was proposed to achieve better security and performance through resource allocation in a multi-tenant environment. Simultaneously, an attack model was constructed, and suspicious tenants were detected by Aljahdali et al. (2014)

Comprehensive Approaches towards Multi-Tenancy

A popular architectural framework to serve efficient multi-tenant applications with a set of services was presented after investigating various requirements (Guo et al. 2007). This solution was originally developed to reduce the costs of services to implement multi-tenancy among enterprises but nevertheless have contributed to data isolation techniques empowering performance and availability. The main concept behind this architecture is simple. Although clients subscribe to shared resources in the cloud, they naturally expect delivery of services as if they are dedicated ones. Thus, isolation in functional and non-functional level needs to be achieved in all sections. Another concept behind the architecture is while the number of tenants increases, the services to other tenants can get affected in a multi-tenancy-enabled service. To avoid such interruptions created by one tenant while they update their application while overcoming availability issues, an architecture that had various modifications in the realms of security, availability, performance, administration and a customizable interface was presented.

Firstly, the approach states that a communication between the tenant's private domain and the hosted service needs to be established. A unique user ID such as a company email address needs to be provided to every user of that service which is also given access to the particular hosted environment through distinct mapping. A second approach is that a company subtree of the client is stored in hosted service. Upon request from a client, a credential is generated according to the subtree and thus authentication is provided.

Access control isolation in this architecture introduced a filter on what a tenant can see and work on. Another option is to use an Access Control List where the environment of each tenant has been pre-allocated. Information protection encryption is also a part of this architecture where encryption to critical data is only to be used due to the fact that there is a trade-off between encryption and access performance. A strong encryption model may lead to poorer performance and therefor a suitable method needs to be selected considering the frequency of access of the data elements (Chong, Carraro & Wolter 2006).

To achieve a fair balance in performance and coincide to SLAs, a number of methods can be implemented. A resource reservation mechanism to ensure minimum SLA compliance is ensured. A load balancer can be used to establish the maximum amount of resources available at a particular time according to the degree of load on the hosted service during the runtime of the application.

A fault isolation process where detection, stopping propagation and repair service for the tenants is a possible solution. Since, a major SLA in such agreements is service availability, establishing such a method is crucial. To stop propagation, one way can be releasing the resources of that specific tenant quicker. Lastly, repairing the ill tenant's service can be editing wrong information and restarting the instances or removing all data of the tenant and replacing it with a backup version from the past. Administration isolation of data suggests backup and restore of one tenant should not switch off the shared database and should be performed when service load is minimal. Also, only the tenant administrator should only have access to the data but nevertheless an encryption on the backup files should be added in case it gets in the wrong hands.

A configuration wizard that lets users take control over their own service requirements during runtime was introduced to assist data isolation solutions. Here, tenants can choose their service e.g. number of instances in a guided and intuitive way. These are then validated by the system later. Such a user interface also provides good customer satisfaction.

The two models discussed in access control isolation have been first proposed in an earlier multi-tenant model. Rather than migrating to multi-tenant model from traditional applications, service solutions were designed with multi-tenancy solutions in mind. The main objective was to protect vital business data in a cost-effective, fast and efficient manner while maintaining the Software-as-a-Service architecture. Nevertheless, organizations must trust a third-party vendor to protect their data.

Approaches in Database Architecture

A scalable, secure and extensible data model design was proposed by Chong, Carraro & Wolter (2006). One choice here is to simply have separate databases. Although, resources and application structure are shared between the tenants, meta-data is used to direct traffic to and from each separate database on tenants' requests. Each database is secured with specific security model to revoke malicious attempts. The databases can now be scaled according the individual client's needs but the hardware costs, database load limitation by the host and backup costs are highly potential shortcomings. These can be great choices for high isolation requiring clients such as medicals and banks.

A second approach is a shared database having separate schemas. Multiple tenants occupy one database where each individual tenant have their group of tables inside a schema that is meant for that tenant. Therefore, they own separate schemas inside one database. Scalability for such

an approach remains intact as tenants can still add columns or even tables but this does not serve a high degree of security as the isolated database approach. On the other hand, this approach can host more tenants by offering more tables within budget as long as tenants do not require a lot of tables made. Also, they have to consider putting their data in a shared database architecture with other tenants.

Thirdly, a shared database and shared schema comes into play. Tenants have separate tenant Ids that are used to access their records from the same table shared by the other tenants. This concept was elaborately investigated in order to find a solution for data interference and its causes (Furda, Fidge & Barros 2018). Out of all approaches, this has the lowest costs related to backup, restore or hardware since the tenants are sharing the same database tables. Although, since tenant data are very closely stored, this approach may need stronger security implications. The ideal situation to go for this approach is when there are minimum resources for a large number of clients. Tenants have to consider their data being stored so closely with others, although at a much lower cost.

Bezemer et al. (2010) conducted research on data interference in multi-tenant environments in order to migrant a single-tenant architecture to a multi-tenancy. They proposed the development of a static analysis tool for software developers. The goal was to make a tool that helps in making secure code in such architecture using a few analysis procedures:

- Configuration: This is where the tool is exposed to data about the application.
- Each individual tenant classes are inserted.
- To help decrease high level analysis input and output channels were identified.
- The final step is where data interference is detected at the statement level of tenants.

A similar concept to multi-tenancy is multi-user where the difference lies in the configuration options. A multi-user environment has limited configuration options where the users are running the same application. In contrast, in multi-tenancy, tenants can configure the application to a much higher degree. Another very popular approach in present times is multi-instance. Tenants receive their own instances of the application, database and other resources required to run them. To create such multi-tenant applications using multi-instance is easier but the cost is higher to conduct maintenance on this model. Thus, it is better suited for low number of tenants (Bezemer & Zaidman 2010).

Approaches in Data Security

It is not shocking to assume that in a shared environment tenants might not trust each other and for good reasons too. Through the cloud a user stores data on the providers system and this can lead to malware attacks. Especially in a shared tenancy system, potential untrusted tenants increase the risk of data leakage through the means of side channel attacks. These attacks can happen quite naturally when the attacker places a virtual machine on the legitimate tenant's system directing all tenant traffic towards the attacker's VM.

"CloudRadar" is a system that was developed to stop side channel attacks in real time. It is a cache-based method that detects common anomalies and access to cryptographic applications. It focuses on the protected virtual machine by detecting typical suspicious behaviour by reading signatures (Zhang, Zhang & Lee 2016)

A number of side channel attack types have been compared by Vanathi & Chokkalingam (2019). A comprehensive research in cache attacks, timing attacks, power-monitoring attacks, etc. Have been discussed along with its respective solutions such as Sandbox, Stopwatch and Game theoretic approach.

4.2 REMAINING ISSUES [NAZ]

After reviewing existing literature, a number of gaps were concluded as our findings. These need to be addressed in order to enhance solutions towards data isolation and security strategies.

SECURITY ISSUES

In such a search for vulnerability, an attack was launches on the popular Amazon Web Services EC2 service. To observe the miscarriage of information, network probing was carried out and brute force was implemented while installing the attackers Virtual Machine beside the VM of the legitimate tenant. By taking advantage of the how the system works, it was seen that an attacker can install a false Virtual Machine beside the tenant's Virtual Machine at a staggering 40% possibility. The victim's data was naturally at stake and any tenant can possibly attack its neighbour (Aljahdali et al. 2014).

A Role Based Multi-Tenancy Access Control Scheme was developed to battle these attacks. In their research, they provided an augmentation of Role-Based Multi-Tenancy Access Control by introducing a server-based authentication model. In this system, before users provide their identification information, genuine users will get a server credential upon request. The system will have users whose roles will be defined by role-based access control procedure. This is an enhanced RBAC. In this system, a trigger is switched whenever a side channel attack is detected. Moving forward, it analyses the network performances and applies another protocol called "KAMAN" to authenticate the user and VM (Kaur & Sapra 2015). This is certainly valid for preloaded side channel attack types. But, attackers can still channel attacks that is unknown to this mechanism.

Yang, Lai & Lin (2013) provided a model in which they have designed a Role-Based Multitenancy Access Control (RB-MTAC). They have integrated identity management and role-based access control (RBAC) of multitenancy to provide safe opportunities to have secure access to hardware resources and safeguard their data. In this, tenants can have their separate database or can be on a shared data system. User authentication through identity management and RBAC defines the roles to preserve data in the database and authenticate legitimate users. The whole system gives good prevention against side channel attacks but overall proves to be time consuming.

A similar problem exists in the "CloudRadar" system where the system can go no further than looking for built-in typical anomalies. Although many defence mechanisms have been discussed in previous approaches, various attack generation and effectiveness of defence mechanisms need to be explored further.

Braun, Jana & Boneh (2015) introduces a timing side channel attack prevention method, in which they have modified the operating system's data segregation methods, time padding and cache scrubbing. Changes in the time-padding prevents the attacker to scrap the execution time of a functionality as it will be kept separated from the function's secret inputs. Changes in the cache cleaning will mitigate the risk of attackers knowing the state of the cache.

Limitations due to data isolation

Inside a shared database but different schema methodology, too many schemas can increase load on the system. Therefore, a filter pattern is the optimal choice over a permission pattern with a number of schemas. But, even in a filter pattern, the tenant data is closely stored, creating an unsafe tension in the clients' minds.

An existing problem in prevailing data isolation technique is when a client's private domain is used to map into the hosted cloud service. If the client's user base is large, it may take huge time and effort to map them into the cloud service provider.

Although a potential solution to the above problem can be encryption of all data, but the issue remains that performance will be impacted when encryption large amount of data and this trade-off between security and performance is unattended.

The number of databases limits the number of tenants on a given database server. In other words, the Daas application can only host as many databases that it can support for its tenants and thus increasing hardware costs. If the allocation is not done correctly, this problem can result in even higher costs and performance difficulties. An established solution is using "Autoclose" that unloads the databases from the memory when the connections are inactive. This can however create issues such as the users may simply not want to deactivate their instances.

In a case where performance isolation prevails and encryption is encouraged, the update, backup and restore procedures must only be performed under certain control policies I.e. at a time when the load on the service is low. This is a drawback since clients need to wait till a time to restore their data once their system has subsided. Otherwise, it will affect every other tenant inside the resource-sharing system.

Furthermore, another shortcoming in a separate-schema approach is when tenants have individual dedicated databases, restoring data can be troublesome. When one tenant schema crashes, naturally restoring the data from the last backup can be a solution. But, needless to say, the tenants are sharing the database. Thus, restoring the DB will overwrite every tenant's data from the last logged backup. This can lead to lost updates. One potential solution is to manually store the data into a temporary server and bring it back after restoring the individual client's data. This is a high-effort and time-consuming task. One potential solution to this is using the shared schema approach where removing and updating individual tenant's rows can do the required task but nevertheless when a large number of rows is affected, this will affect the performance significantly.

5. Our Approach: *The AI Cloud Administrator*

Machine Learning architecture to ensure data isolation & security

In this part, there will be a discussion on our approach that consists of a number of procedures that we can put through to tackle data isolation and side channel attack threats in cloud computing.

Motivation towards approach

Our findings suggest that cloud providers may go overboard on the number of tenants that their systems can hold (Vicentini et al. 2018). As a result, existing tenants can suffer from performance inadequacy. Our assumption for the approach is that the metrics provided by the provider is biased. In order to monitor efficient performance isolation procedures, it is necessary to implement a system that finds the correct measures for the client and the provider so that steps can be taken to fix it promptly.

According to general Service Level Agreements (SLA) in cloud computing, service providers must provide desired performance, dependability and security within the system. As security levels are being incorporated, the resource consumption increases likewise. The basic security declaration involve data security, scalability, reliability and overall performance but cloud service providers should opt for more adjustable security measures so that performance isolation can always be provided in a new or old complex threats arising situations (Almorsy, Grundy & Müller 2010).

Description

Our solution learns from the provider resource metrics and uses that information to derive performance isolation measures and eventually takes countermeasures. It also uses artificially intelligent algorithm to investigate security attacks in order to battle newly shaped intrusions. The algorithm updates itself every time it learns from a new threat and thus potential hackers are unable to breach it using existing security system-breaking protocols.

Proposed Solution Architecture

The Application Auditor:

The proposed solution first audits the performance metrics of the application instances from the initiated units in the last 30 seconds. As each tenant has a set of executors attached to its application, we will derive the units from these executors. This will give the provider a view of how much the tenant application is consuming. But for the tenant-side monitoring, the metrics provided by the provider might be biased. Therefore, we use another auditor.

The Resources Auditor

Our initial approach measures the metrics of resources of the specific tenant periodically. To obtain a high-level view of the resource consumption, the system sends one resource metric along with three application metrics to the AI Learner.

The AI Learner

This is the heart of the system that collects the data from the Application and Resources auditors. Using these performance metric data as inputs, the learner builds feature vectors out of each combination instance. The next step is the classification process where an artificial neural network will be used to add weight to each component of the feature vectors. The algorithm learns from the event classes provided for regular tenant performance measures and multi-tenant metrics. To detect if the resources provided to the tenant are optimal, we label them correct only if it matches with the performance metrics mentioned in the SLAs. Otherwise, we move on to the next step by labelling it as problematic multi-tenancy.

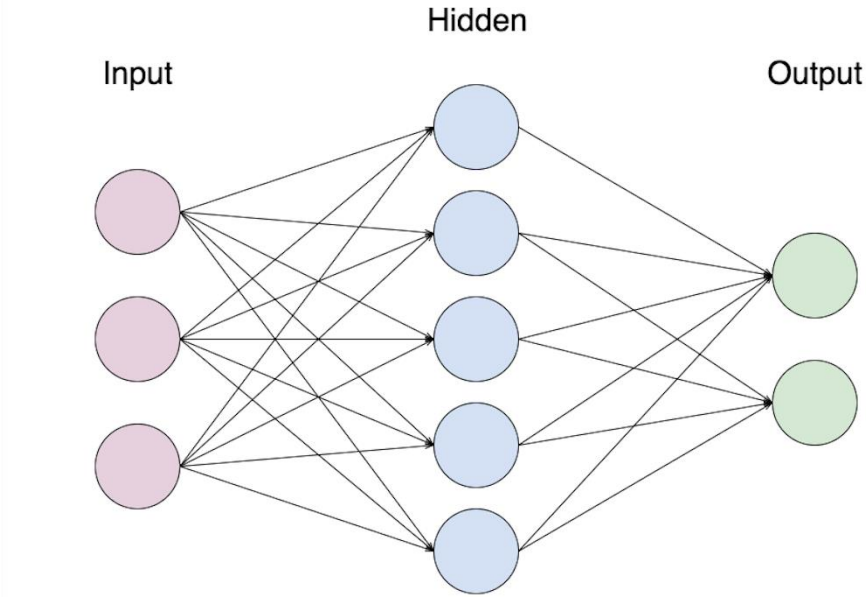

Fig 4: An artificial Neural network structure

Performance deficiencies can be explained by improper data isolation methods and security threats that may be driving the tenant data traffic towards the attacker's own machine mainly through side channel attacks.

To battle this, we implement the following for data security & isolation techniques:

1. Encryption of tenant data with authentication protocols.

2. Removing out-of-order executions

3. Aspect oriented programming protocols &

4. Client request limitations during multi-tenancy issue detections by the AI Learner.

5. Finally, we provision the enhanced resources using Linear Regression algorithm.

Potential other solutions to tackle side channel attacks:

1. Runtime encryption of data with Intel SGX:

The enclave memory system provided by Intel SGX can ensure a trusted environment for data to be accessed and perform computation functionality. Intel Software Guard Extension (Intel SGX) provides regulations around system code and data access environment while it gets exposed to any kind of adjustments. Through using this, developers can make use of the enclave memory to segregate sensitive data.

Data will be kept surrounding the enclave and thus runtime encryption will decrease the attacks as it's almost impossible to penetrate through an enclave memory. Attackers will only be able to get encrypted data which needs decryption keys to be decipherable.

The encrypted data will reside in enclave and it will be computed there. There will verification process which will include a tenant to have report sent and after successful authentication the next step can be to provide the CPU generated "sealing key" to access stored data (Fig 4)

Fig 4. Intel SGX Enclave memory and sealing key creation

All encryption algorithms merged with runtime encryption in the mentioned way we can provide safety against different side channel attacks.

2. **Cut off the out-of-order execution functionality:**

The recent frenzy around side channel attack in Meltdown and Spectre has caused a great deal of havoc for the victim organizations and the root cause for it is out-of-order execution functionality. Depending on the cloud infrastructure of the provider, it can be possible to exploit data from these applications that run in the background of many personal computers, mobile devices and cloud systems. It essentially refers to the procedure of a CPU processor predicting response time, CPU cycles, jump address and branching ways way ahead the related real values can be computed. It certainly peaks the performance graph while it can prove to be vulnerable against side channel attacks. So, in order to reduce the rate of speculation, out-of-order automated execution should be avoided as they can lead to vulnerability.

Potential other solutions to ensure data isolation:

1. Aspect oriented Programming (ASP): In this approach, there is a distinct differentiation between fundamental functionality designs of a database and advocacy of security measurements. As security is a separate segregation, there will be no concern about failure of the entire system. Through ASP implementation, it is possible to integrate discrete code segments to facilitate or address different non-functional issues such as data security. Moreover, the security strategies are assembled in one place so that it can be reviewed and revised easily.

Each tenant can have their own security measurements i.e. encryption methodologies, access mechanism based on their need while using the same body code. By using ASP language or libraries such as AspectJ, Spring, the functionality is coded once i.e. data isolation measurements and can be traversed through different objects.

2. Limiting requests per tenant: Tenants in the multi-tenant architecture share hardware and software resources via a mutual data storage. With the increase of tenants and demands, there are often overload situation being created due to many tenants trying to access similar SaaS services concurrently.

Through implementation of rate-limiting algorithms which will monitor the requests per tenant, if a maximum number of access has been reached, the tenant will get an error message. This process can ensure reasonable quality of services to every user on the platform.

Linear Regression in Resource provisioning

From previous work, we have seen how Reinforcement Learning works an automated resource provisioning model. This model learns from experience where it observes the current state many times until action is taken and then transitions to the next. The result is observed. This can take a long time because of its iterative methods. Keeping the current cloud computing speeds in mind, we have chosen LR which observes the output based on inputs while deriving prediction value of the classes. Therefore, the main job of the LR here is to find the degree of priority according to estimated deadline based on which resources must be placed after facing multi-tenancy issues. The priority is based on the number of highest combination metric from the AI Learner that has the highest latency.

Figure 5: Sample comparison between LR and RL

In this approach the first three jobs are allocated to their resources using FIFO and the requests that come after are explored using LR. If the priority value is high, it is allocated next. Since machine learning can deal with complex patterns, this model is more capable in getting better estimate of resources required than self-provisioning by the provider's system.

5. REFLECTION

It's safe to say that multi-tenancy has made cloud computing immensely popular by ensuring maximum resource utilization through leveraging share of hardware and software resources among different tenants. While it is extremely beneficial to have tenants' access and share each other's databases, resources and application through multi-tenancy, significantly worrying security risks can overwhelm the gains altogether.

The security risks in cloud computing's multi-tenant architecture depends on the level of multi-tenancy that is being provided. Although, the providers of multi-tenancy service put in a lot of proficiency and resources to ensure that the foundation is solid and can be trusted, but poor access management system could lead to unethical breach of shared resources within the infrastructure.

SAAS applications is based on the premise of having multiple tenants in a single server accessing multiple instances of a software which is being hosted in that server. Through multi-tenancy, different tenants can have exclusive data storage system also can distribute information among each other. But each tenant needs to feel like it is their own platform meaning they cannot see other tenants and their information if it is not shared. It must be ensured that the performance and resource utilization of one entity in the server does not affect the others. The whole process becomes tricky and prone to infringement of data with higher level data and tenant integration. Accordingly, tenant databases need to have elevated table scalability.

Isolation among tenants is the key driver to ensure secure multi-tenancy platform. When tenants are trying to access similar SaaS, PaaS or IaaS services, there needs to have a clear and distinctive approach dedicated to each tenant like it's their own platform. Successful separation of tenants means higher level memory and CPU power utilization. The problem that exhibits a great deal of trouble is when a tenant is consuming too much space in a shared fragment. This eventually hampers performance isolation as the distributed queries get interrupted. Safeguarding multi-tenancy involves location transparency, which means tenants will have no idea where each other's resources are located so the attackers will have a hard time tracing their desired tenants.

Side channel attacks in a multi-tenant environment can be catastrophic as these strikes are hard to trace and goes beyond the traditional I/O channels. In the conventional way, an attack can happen through a malevolent input which corresponds to unwanted output indulged with classified information. On the other hand, side channel attack is conducted by analysing differences in functionalities i.e. response time and take a guess about the confidential information.

In spite of recent advancements in the field of cloud computing, service providers still struggle to secure sensitive data provided by tenants and manage the performance isolation all together.

As a cloud provider, the main responsibility is to ensure safety of the data and services being shared in the system. Developers must continuously come up with different solutions as attackers very quickly can penetrate through the existing security depths and attack to gain sacred information. Side channels existing in hardware resources are most prone to these attacks. The attacks are mostly fuelled by hardware caches that are part of CPU's memory sub-structure.

6. CONCLUSION

Multi-tenancy provides a centralized administrative platform where computing resources are shared equally among all the organizations residing under the cloud and reducing the overall modification and management cost accordingly. Data belonging to different tenants are residing side by side. While in a single tenant architecture the risk for information exposure to other organizations are low, in a multi-tenant platform, the first and foremost risk is the failure of data isolation. In addition, side channel attack is another prominent threat as the attacks are hardly detectable and harms the system from within by analysing response time, time padding patterns and out-of-order execution functions.

Amazon cloud outage that happened in 2011 causing the whole system shut down for two days and organizations facing data loss, had everyone question about the cloud services being ready or not to provide such crucial responsibility of securing data stored by multiple tenants. In our paper, we have tried to outline the prevailing risks around Multi-tenancy and specifically focused on risks of data isolation and side-channel attack prevention existing methodologies while trying to provide solutions of our own. The terms and use conditions of cloud providers should be reviewed rigorously by focusing on service level agreements, ownership (IP) agreements, termination clause as well as put clauses for disaster recovery, data transfer whereabouts and mitigation strategies for unpredicted security threats.