A Review on Amusa: Access Control Middleware for Multitenant SaaS Applications

Priyanka S. Mane, Yogesh. B Gurav

1ME Student, Department of Computer engineering, Padmabhooshan Vasantdada Patil Institute of Technology. Savitribai Phule Pune University, Maharashtra, India

2Professor, Department of Computer engineering, Padmabhooshan Vasantdada Patil Institute of Technology. Savitribai Phule Pune University, Maharashtra, India

Abstract: Access control is the most important and often only application-level security mechanism for SaaS applications. Application-level access control is responsible for constraining the actions of authenticated users on the information in the application by using access policies. However, access control policies for multi-tenant SaaS applications is inherently complicated as an access control system for multi-tenant SaaS applications should make sure that tenants can’t access each other’s resources in the application. These policies should enable the application provider to constrain its users based on its own rules. Also enable the tenants to constrain their end-users based on their own rules as per system policy. These policies should enforce the appropriate rules in the centralized application at runtime. In this paper we study Amusa- Access control middleware for multi-tenant SaaS applications, defined to achieve above things. The goal of Amusa is to provide middleware for multi-tenant access control management. It provides efficient access control management to the providers of SaaS applications and their tenants, and is reusable by multiple SaaS providers. Amusa enables both the provider and all its tenants to efficiently declare their access rules on the SaaS application. Amusa securely combines the access rules of all parties and enforces them at run-time with low performance overhead.

Keywords: Access control management, Software as a service, multitenancy, security, access control

1. Introduction

Nowadays Cloud computing is a fast computing platforms that delivers a new generation of internet-based, highly scalable distributed computing platforms. Cloud Computing is defined as a model for enabling convenient, on-demand network access to computing resources such as networks, servers, storage, applications, and services, that can be provisioned rapidly and released with minimum effort or interaction of service provider i.e. the ability for end users to utilize resources and can be acquired quickly and easily. The characteristics of cloud computing are on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service. The Figure 1 depicts the cloud computing stack – three distinct categories within cloud computing: Software as a Service, Platform as a Service and Infrastructure as a Service. SaaS applications are designed for end-users, delivered over the web. PaaS is the set of tools and services designed to make coding and deploying those applications quick and efficient. IaaS is the hardware and software that powers it all servers, storage, networks, operating systems.

Figure 1: The Cloud Computing stack – three distinct categories within Cloud Computing: Software as a Service, Platform as a Service and Infrastructure as a Service.

Software-as-a-Service (SaaS) is one of the three key service delivery models delivered by the cloud computing model. The cloud model, including SaaS, is based on two key characteristics: multi-tenancy, where multiple tenants share the same service instance, and elasticity, where tenants can scale the amount of their allocated resources based on current demands. Multi-tenancy means that a SaaS (Software as a Service) vendor provides a single version of its software for all its customers. This differs from a single-tenant hosted solution, where the application is housed on a vendor’s server but the codebase is unique for each customer. Although all users of the software access the same foundational components, the data and configurations that are specific to a customer are stored in a separate and secure container. Users can access all the capabilities of the software, but their data aren’t shared. Although this characteristic target improving resource utilization, cost and service availability, these gains are threatened by multi-tenancy security implications. Sharing applications that process critical information with different tenants without sufficient proven security isolation, security SLAs or tenant control, results in "loss-of-control and” "lack-of-trust” problem.

Access control is the most important and often only application-level security mechanism for SaaS applications. Application-level access control is responsible for constraining the actions of authenticated users on the information in the application by using access policies. However, access control policies for multi-tenant SaaS applications is inherently complicated as an access control system for multi-tenant SaaS applications should make sure that tenants can’t access each other’s resources in the application. These policies should enable the application provider to constrain its users based on its own rules. Also enable the tenants to constrain their end-users based on their
own rules as per system policy. These policies should enforce the appropriate rules in the centralized application at run-time. To achieve this, we present Amusa - Access control middleware for Multi-tenant SaaS Applications[1]. Amusa allows both the provider and all its tenants to express their access rules for the SaaS application using expressive attribute-based policies. Amusa combines these policies securely and enforces them at run-time. Amusa simplifies uses an incremental three-layered approach and introduces low performance overhead.

2. Literature Survey

In previous work issues and importance for multi-tenancy, access control and policies are addressed in [6][7], identified two high-level requirements: isolating tenants and allowing the saas application to be customized to the specific needs of each tenant. This work focuses on access control, which is both a means to provide (application-specific) tenant isolation and an important source of variability in SaaS. But very little work has been performed to achieve these requirements. The authors are not aware about the solutions that provide strong tenant -specific policy based access control. And the case is many industries strongly insisting for such Technology . This approach of tenant isolation implemented manually most of the time, and builds on strict data isolation in the data store. More specifically, a built-in tenant identifier is used in database queries. This is adopted in gae[4], but this does not allow easy application specific customization of the isolation policy. [2] also focused on multi-tenant authorization for cloud applications. They adopted mechanism for extending role based access control specifically for multi-tenancy. Work of this author is later formalized by Tang et al.[3] amusa extends this approach by moving from rbac to more expressive attribute-based policy trees and extending the architecture into a reusable middleware. [8] Sloman applied policies for declaratively managing access control in distributed systems, which later lead to the definition of the influential ponder specification language for access control policies [10] ,employ policies for information flow control in multi domain applications to track the data access and data confidentiality. Kumar et al.[9] proposed work that describe self-management behavior by applying some policies for user access. the common factor and proposed work is that policies are used to separate semantics from enforcement and describe the semantics declaratively.

Following Table [1] shows the different existing Access Control Systems for Multitenant SaaS applications

3. Existing System

The existing SaaS application level access control system constrain the actions of authenticated subjects on the resources by enforcing access rules. Access control for multi-tenant SaaS applications is complex because:

- Firstly, the provider's policies about tenants are often hard-coded in the application.
- Secondly, the tenant specific access control rules are often supported using a simple role-based scheme.
- Thirdly, tenant isolation is either hard-coded in the application or relies on isolation at the level of the data-store such as using namespaced silos in Google App Engine.

As a result, the provider policies are hard to customize, tenants are able to express only a limited set of access rules, and tenant separation is inflexible. This access control system does not meet these challenges. To solve this problem the system do the combination of policy-based access control with attribute-based , tree-structured policies. The combination of policy-based access control with attribute-based, tree-structured policies supports the declarative specification of a wide range of rules outside of the application and binding these at runtime. However, these bare technologies still require each SaaS provider to build a multi-tenant access control layer on top of them. And even when employing these technologies, addressing the requirements stated above is not an easy task.

Limitations of Existing System

- The provider policies are hard to customize.
- Tenants are able to express only a limited set of access rules.
- Tenant separation is inflexible.
4. Proposed System

To overcome the limitations of the existing system a new system is proposed named Amusa. Amusa provides incremental three-layered management based on attribute-based tree-structured policies. Amusa securely combines the access rules of all parties and enforces them at run-time with low performance overhead. Amusa enables both the provider and all its tenants to efficiently declare their access rules on the SaaS application.

Three-layered access control management

In terms of attribute-based access control, access control management consists of managing attributes and managing policies that employ these attributes. Amusa divides this management over the three kinds of stakeholders involved: the provider, the tenants and the Amusa middleware itself.

Firstly, Amusa precedence’s common attributes and policies that can be reused across applications, providers and tenants. Secondly, the provider overs the SaaS application. Thirdly, the tenants manage their own users based on organization specific attributes and policies. Three-layered attribute management.

Managing attributes need to (1) defining possible attributes for subjects and resources and (2) assigning values to the attributes. In this case, the three layer management applies to attribute definition: Amusa itself pre-defines a fixed set of attributes, which the provider can extend for its own application and which the tenants in turn can extend for their organization. More precisely, Amusa pre-defines the attributes it requires for its correct functioning and a number of frequently occurring attributes. The provider then assigns the appropriate attributes to its resources. In essence, these attributes are already present in the SaaS application itself. Finally, the tenant assigns the subject attributes defined by Amusa, the provider or itself to its subjects.

Three-layered policy management

The policies of each layer can be specified in terms of attributes that are available in that layer. Firstly, Amusa itself has some policies built-in that apply to the provider and all tenants. The most important of these is the default tenant isolation policy. These policies can only reason about the attributes pre-defined by Amusa. Secondly, the provider can specify policies about the tenants as a whole. These policies can employ the attributes specified by Amusa and the provider itself. For example, eDocs specifies that users belonging to a certain tenant cannot send any document if the credit of that tenant is not sufficient. Thirdly, the tenants can specify policies that apply to their own users. These policies can employ the attributes specified by Amusa, the provider and the tenant itself.

Amusa builds on three state-of-the-art access control technologies that support part of these requirements, i.e. policy-based access control, attribute-based access control and tree-structured policies. Policy-based access control. Policy-based Access Control is a recent model to express access rules in terms of key-value properties of the subject, the resource, the action and the environment. These properties are called attributes and include for example the subject identifier, subject roles, resource type and the time. Figure[2] shows the Amusa middleware facilitates incremental three-layered of multi-tenant SaaS applications.

Architecture of the Amusa middleware

In Amusa the application consists of two components: application logic and the database containing the application resources. The Amusa middleware itself consists of six major components: Policy decision point, Authentication endpoint, Provider administrator dashboard, Tenant administrator dashboard, Database containing attribute definitions, and Database containing subject attribute.
Algorithm used
1. User authentication using Hash function.
2. Data security while transferring data using Cryptography algorithm as AES, Triple DES.
3. Combination algorithm for sub-tree PermitOverrides.
4. Complete policy-tree algorithm DenyOverrides.

5. Advantages And Disadvantages

Advantages
1) Such access control enable the provider to constrain its tenants, e.g., make sure that only paying tenants can access the application.
2) Enables the SaaS applications provider that the tenants cannot access each other's resources in the shared application.
3) So the tenants are be able to constrain their own users of the SaaS application.

Disadvantages
1) As growing number of tenants in SaaS, Amusa grows with the number of tenants, also grows the size of the complete policy tree. Each new tenant adds a branch to the policy tree of which the applicability will be checked during each policy evaluation. This increased the work as well the operation time for generation of tree node.
2) As the no of tenants exceeds than some limit, overall performance overhead also increases.

6. Conclusion and Future Work

In this paper we study previous access control system and Amusa. Amusa is an access control middleware that provide SaaS-specific layer that enables flexible and secure multi-tenancy on attribute based tree-structured access control policies. These techniques are used in a three-layered approach, which simplifies access control management through gradual refinement. In future need to focus on work for implementation of a test methodology and a real-time monitoring mechanism for multi-tenant applications. As the no of tenants increases need to manage the performance overhead.

References
