Secure Information and Resource Sharing in Cloud IaaS

APPROVED BY SUPERVISING COMMITTEE:

Prof. Ravi Sandhu, Ph.D.

Prof. Ram Krishnan, Ph.D.

Prof. Palden Lama, Ph.D.

Prof Jianwei Niu, Ph.D.

Prof. Gregory B. White, Ph.D.

Accepted:

Dean, Graduate School

DEDICATION

I would like to dedicate this dissertation to my family who support me with great love, especially to my beloved Matt Meyers who supports me with patience, companion and humor. I also dedicate this dissertation to all my friends who supported me with their kindness and encouraged me during hard times of this endeavor.

Secure Information and Resource Sharing in Cloud IaaS

by

YUN ZHANG, Ph.D.

DISSERTATION
Presented to the Graduate Faculty of
The University of Texas at San Antonio
In Partial Fulfillment
Of the Requirements
For the Degree of

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT SAN ANTONIO
College of Sciences
Department of Computer Science
December 2016

ACKNOWLEDGEMENTS

I would like to give my most sincere appreciation to my advisor Dr. Ravi Sandhu for his great guidance and support during my doctoral studies and the process of achieving this dissertation. His wisdoms and professionals have enlightened and guided me all the way through my doctoral studies to accomplishing this achievement today. His great guidance helped me overcome difficuties during the research and studies. He has taught me not only to be a good researcher in acdemia, but also to be a good person in life. Working with him is one of the most precious and invaluable experience in my life.

I would also like to thank Dr. Ram Krishnan, who co-advised me at the beginning stage of my research. He guided and helped me with my first paper. He showed me an interesting and chanllenging research world. I also want to express gratitude to all the other professors in the committee: Dr. Palden Lama, Dr. Jianwei Niu and Dr. Gregory B. White, for their valuable comments and suggestions.

I would like to especially acknowledge the support from Mr. Farhan Patwa, who is the associate director in the Institute for Cyber Security (ICS) of UTSA. I would like to thank Mr. Farhan Patwa for his guidance and expertise regarding practical enforcement techniques in dominant cloud platforms. He has helped me with all the implementations reguarding to models in my papers and dissertation, which is a crucial part of my research. He also enlightened me from practical perspective, which helps me accomplish the models more realistically.

I want to thank all my fellow Ph.D students (some of them already achieved their Ph.D degrees) in ICS. They provided me great help during the process of study new knowledge, tools and programming.

I also want to thank Raildo Mascena, who is an OpenStack Active Technical Contributor. He helped me with his expertise in understanding Hierarchical Multitenancy in OpenStack.

This work is partially supported by the LMI Research Institute's Academic Partnership Program. This work is also partially supported by NSF CNS-1111925 and CNS-1423481.

December 2016

Secure Information and Resource Sharing in Cloud IaaS

Yun Zhang, Ph.D.
The University of Texas at San Antonio, 2016

Supervising Professor: Prof. Ravi Sandhu, Ph.D.

Cloud infrastructure as a service (IaaS) refers to virtualized IT resources such as compute, storage and networking, offered as a service by a cloud service provider on demand to its customers (or tenants). IaaS is the fastest maturing cloud service model today where tenants are typically strictly isolated from each other. Cloud IaaS provides enterprises and organizations a secure and efficient environment to deploy their systems. While organizations and companies benefit from moving to cloud platform, it is likely that similar cyber attacks will happen to organizations that share the same cloud platform and similar infrastructure. One way to mitigate this risk is to securely share cyber security information and resources among these organizations. Contemporary public cloud platforms such as OpenStack, AWS and Microsoft Azure are lacking a widely accepted access control model for such secure information and resource sharing.

A community in a cloud IaaS refers to a group of organizations with similar organizational structures or business models sharing common business interests, utilizing cloud IaaS to realize their infrastructure deployments. Threat analysis and incident response infrastructure and resources can be rapidly shared in a cloud community, whereby the participating organizations save time and cost in handling cyber incidents. A community can establish a mechanism to prevent, detect and respond to cyber attacks, share cyber security information among these organizations, and help member organizations in the community response to and recover from cyber incidents expeditiously.

In this dissertation, we present an access control model to enable organizations to securely share cyber information and resources during cyber collaborations in a community-based isolated environment in cloud IaaS platforms. The model facilitates a tenant to share its IT resources with other tenants in a controlled and secure manner. It enables secure and effective management of information sharing from a community based perspective for both routine and cyber incident response needs. We then define access control models for each of the three dominant cloud IaaS platforms, viz., OpenStack, Amazon AWS and Microsoft Azure, to abstractly represent the access control features of three complex systems. We further develop access control models for sharing between organizations in a community-based isolated environment on these IaaS cloud platforms. Then we formally specify administrative models and discuss enforcement and implementation techniques for each cloud IaaS platform. Finally, we compare these models for these three systems from perspective of enforcing the secure sharing model in different cloud IaaS platforms.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1: INTRODUCTION

Cloud computing is revolutionizing the way businesses obtain IT resources. Infrastructure as a service (IaaS) is a cloud service model [10] where a cloud service provider (CSP) offers compute, storage and networking resources as a service to its tenants, on demand. By tenant, we refer to an organization that is a customer of a CSP. The need to share information between organizations (commercial and governmental) continues to be an important requirement for various reasons including incident response, improved productivity, collaboration, etc. Securely and effectively sharing cyber information across multiple organizations and cooperating on cyber security incidents has been a significant research topic in recent years. Our premise is that as organizations move to cloud, the traditional information sharing activities would also need to move to cloud.

Cloud computing has three service models: infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS), as shown in Figure 1.1. PaaS offer a development environment to application developers. SaaS offers application and software to end-users. We focus on IaaS cloud for two reasons: (1) IaaS is one of the most adopted cloud service models today (as compared to PaaS and SaaS, respectively), and (2) IaaS is also the foundation of cloud with characteristics such as elasticity, self-service, etc. By gaining insights on issues related to sharing at this lower level of abstraction, we can subsequently also develop analogous models for higher levels of abstractions of cloud such as PaaS and SaaS. Note that in the context of IaaS, the unit of sharing comprises virtual resources such as objects in a storage volume, virtual machines (VMs), etc.

The deployment models of clouds can be categorized into public, private, community and hybrid clouds [10], as shown in Figure 1.2. A public cloud provides services for open use by the general public. A private cloud provides services for exclusive use by a single organization. A community cloud provides services for exclusive use by a specific community, which contains organizations with shared concern, such as mission, security requirements, business models, etc. In some cases, a big corporate group with multiple subsidiaries may own one community cloud for

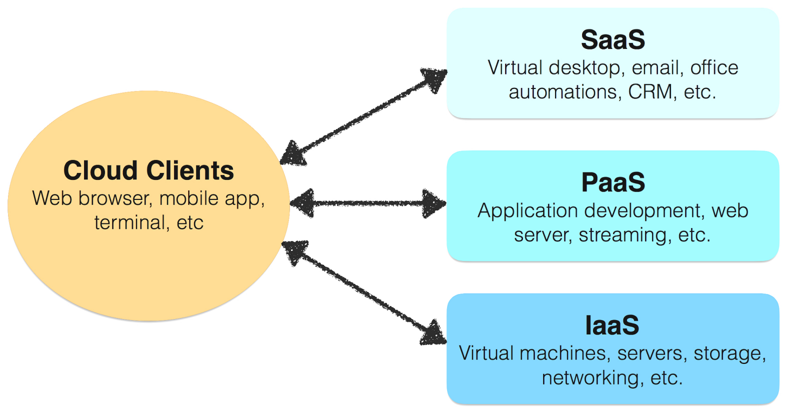

Cloud Clients
Web browser, mobile app, terminal, etc
SaaS
Virtual desktop, email, office automations, CRM, etc.
PaaS
Application development, web server, streaming, etc.
IaaS
Virtual machines, servers, storage, networking, etc.

Figure 1.1: Cloud Service Models

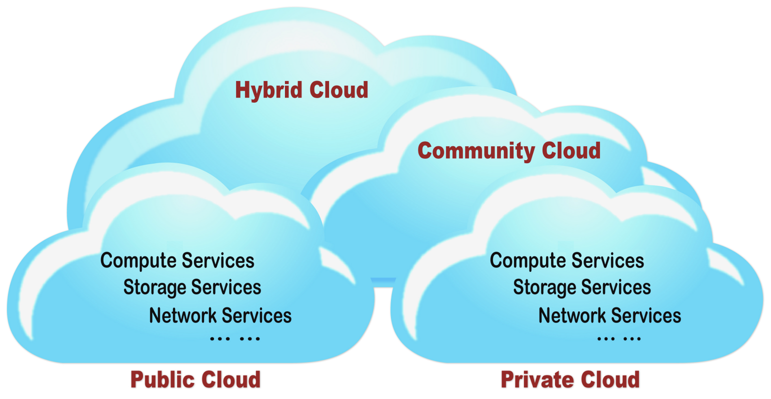


Figure 1.2: Cloud Deployment Models

business needs. A hybrid cloud is a composition of multiple distinct clouds, which may be public, private or community clouds. In this dissertation, we investigate models information sharing in a community in a cloud constructed using OpenStack cloud platform, or as communities in Amazon Web Service (AWS) cloud or Microsoft Azure public clouds.

IaaS providers emphasize strict separation between tenants for obvious reasons. Thus their virtual resources are typically strongly isolated. For instance, in OpenStack, a tenant user does not have the capability to access resources outside her domain. By domain, we refer to the administrative boundary of that tenant. In release of Kilo of OpenStack, each tenant is represented internally as a domain. Similarly, in Amazon AWS and Microsoft Azure, tenants refer to accounts - an administrative boundary of cloud resources. Users from one account (tenant) by default have no rights to access resources outside that account.

The emergence of cloud as a shared infrastructure, significantly improves the efficiency and flexibility of business systems, as well as incident response processes. A critical concern for participating organizations is the level of control that they can maintain over the resources that are shared. In particular, participating organizations will need to ensure that the resources are shared only with users from select other organizations, and can retain the ability to enable and disable the sharing. Thus in order to share resources between tenants, we need to develop access control models that offer precise control to each tenant on what they are willing to share. As we will see, realizing this scenario varies considerably on the different cloud IaaS platform studied in this dissertation.

1.1 Motivation

Cyber security information sharing allows organizations to share threat analysis and incident response information with collaborative communities formed to handle both existing and potential cyber threats, as shown in Figure 1.3. With growing sophisticated cyber attacks every year, defending a single organization on its own becomes increasingly difficult. All organizations, regardless of size, could be the target of a cyber attack putting critical digital assets at risk. A cyber breach can result in substantial economic loss. Establishing a general cyber incident response mechanism for organizations enhances cross-organization coordination, speeds up incident analysis and decision making process, helps identify and understand the attack and facilitates quick response actions. An effective response can minimize the damage caused by cyber incidents on valuable digital assets. Cloud technology puts multiple organizations in a single cloud system infrastructure, giving additional opportunities for cyber adversaries to attack organizations with similar systems in a single cloud. On the other hand, by virtue of sharing same cloud infrastructures it is more likely that organizations will have similar concerns regarding security and privacy. Having suitable cyber security risk management mechanisms in public-cloud communities could be significantly valuable to every organization in that cloud.

While cloud technology significantly improves efficiency and flexibility of business systems,

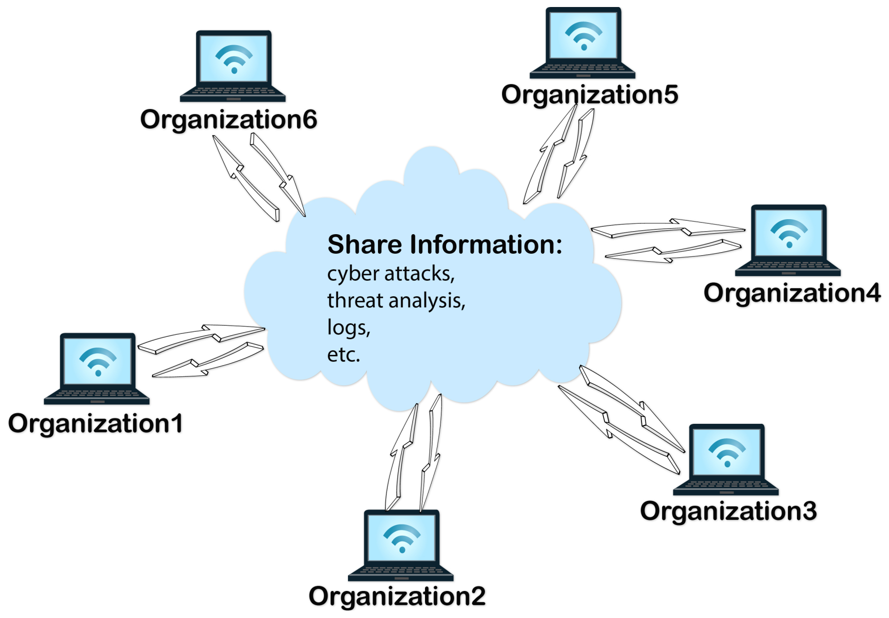


Figure 1.3: Sharing Information in Cloud

it also facilitates cyber collaborations. To our knowledge, current dominant cloud platforms are lacking broadly accepted cyber incident response mechanisms. The traditional approach for cyber security collaboration is mainly through subscription services where organizations can get threat analysis, cyber attack reports and alerts, and so on. Individual organizations submit their security data to a centralized security service center for this purpose. These traditional approaches focus on simply security information exchange and sharing. In our work we seek to develop approaches allowing participating organizations to actively collaborate and interact through the life cycle of a cyber incident which impacts multiple organizations in a well-defined cloud community. We consider two aspects in the context of information and resource sharing which to a certain degree will depend on each other: models and technology. This aligns well with the requirements of cyber incident sharing since organizations can share virtualized snapshots of their IT resources in a community cloud dedicated for, say, electric grid.

Cyber attacks are becoming increasingly sophisticated and difficult to defend by a single organization on its own. Cyber attacks have resulted in significant economic losses. Determined adversaries and organized cyber criminals are aiming at organizations of all sizes putting their valuable digital information at risk. Establishing cyber incident response mechanisms in an organization improves the decision making process and internal and external coordination, which potentially minimize the damage of cyber incidents. By explicitly designating users and roles who are in charge of security issues associated with organization systems, quick decisions can be made if a cyber attack happens. By explicitly establishing a standard cyber security process, organizations can easily identify the problems, schedule the defense process and prevent themselves from further loss caused by improper handling of cyber incidents.

Currently, the way organizations collaborate on cyber security is more like a subscription service they get from a collaboration center. Take FS-ISAC [4] for example. The member organizations submit their security information and get security services like reports and alerts from the collaboration center. This type of cyber collaboration has several limitations. Organizations manually submit security information. Organizations are not actively participating in analyzing and processing the cyber information they submit. Sharing information is mainly by subscription, rather than interactive sharing in a group. Also, to our knowledge, current dominant cloud platforms are lacking broadly accepted cyber incident response mechanisms. With cloud technology development, we believe with organizations transferring to cloud environment, the way they share cyber information will change as well.

1.2 Problem Definition and Solutions

1.2.1 Problems

Threat analysis and incident response information needs to be shared with collaborative groups formed to handle both potential and existing cyber incidents. A key requirement of effective cyber incident sharing and response is that a community needs a capability to share beyond simple data such as log files and documents. In particular, organizations need to replicate a smaller-scale version of their affected IT infrastructure including infected servers with network architectures, routers and firewall configurations, etc., for effective analysis, response and sharing with other organizations.

1.2.2 Solutions

Consider a community cyber incident response scenario where organizations that provide critical infrastructure to a community (such as a city, county or a state) share information related to a cyber incident in a controlled manner [7]. Sharing information amongst such organizations can greatly improve the resilience of increasingly cyber-dependent communities in case of co-ordinated cyber attacks [8]. One domain of a community that can benefit from cyber incident sharing is that of the electric power grid (see Wang et al [17] for a theoretical example of cascading a small scale attack to the entire U.S. power grid).

A community in a cloud shares the infrastructure across multiple organizations from a specific community with common concerns in terms of security, privacy and compliance. We propose a cloud access control model to allow organizations in a cloud community to rapidly and meaningfully share cyber security information and resources. The goal is that organizations in such a well-defined community can rapidly share cyber security information and resources. The community runs a standing Cyber Security Committee, which enables its member organizations executives and technology leaders to provide oversight of privacy and security while enabling effective information sharing. This cross-organizational committee is in constant communication to coordinate such sharing while meeting privacy and security needs. It also runs a Cyber Security Forum, which provides a general place for users from the allied organizations to share security information. The security committee enables organization security leaders to be aware of the overall security and privacy situation, and is limited to select individuals form each member organization. The security forum provides an open forum for security education and awareness for the community, which is limited to individuals from the member organizations who can voluntarily join and leave.

Organizations will collect and analyze their security data as usual, while sharing cyber security information with other members through community cyber security committee, in order to make informed decisions about the community security governance. In most cases, organizations maintain their group of security specialists, who manage security policies, conduct security audits and investigate security related events. A community also maintains a group of external

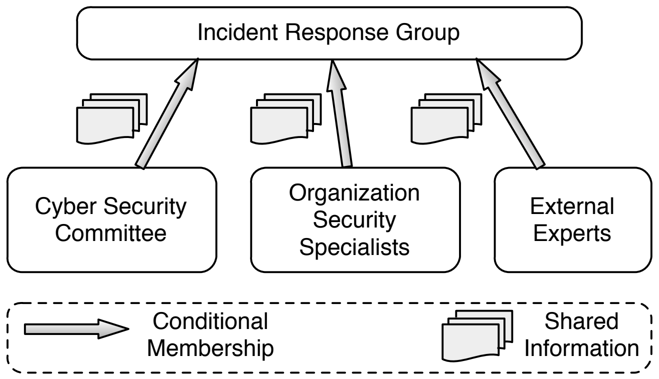


Figure 1.4: Community Cyber Incident Response Governance [21]

security experts, who help organizations with security issues. During the occurrence of cyber security incident, the Cyber Security Committee members start an incident response group with cross-organization security team including organizations' internal security specialists and external security experts, as illustrated in Figure 1.4. Security information about this incident is shared within the incident response group.

When a cyber incident occurs, affected organizations within the community can quickly form a cyber incident response team along with internal and external security specialists. Security information and resources for the incident are shared in the incident response team. A cyber security service is provided in the cloud, which enables organizations having cross-organization collaborations to communicate and coordinate with other organizations during life cycle of a cyber incident. Organizations share their security data with other members in the community.

We discuss models in two aspects: *administrative* models and *operational* models. Administrative models are concerned about managing what resources are to be shared with which users, setting-up and tearing-down platforms for sharing, etc. Examples include a tenant administrator creating a shared secure isolated domain, adding/removing the tenant's users and resources to that domain, inviting other tenants to join the domain, etc. Operational models are concerned about controlling what activities users can perform on the shared platform. Examples include, creating new resources in the domain, modifying objects in storage volumes, importing new resources, etc.

To be concrete, we develop these models based on the facilities exposed by currently dominant

IaaS cloud platforms, including OpenStack [5], a widely adopted open-source cloud IaaS project, Amazon AWS [1] and Microsoft Azure [3], two prominent commercial IaaS providers. These cloud softwares allows creating an IaaS cloud out of conventional hardware. IaaS cloud is a cloud services platform, offering compute power, database storage, networking resources, content delivery and other functionality throughout a datacenter, all managed through a dashboard giving administrators control while empowering their users to provision resources through a web interface. OpenStack is a robust platform for building public, private or hybrid clouds that is developed and maintained by a vibrant community with participation from more than 200 world-leading organizations with a release cycle of 6 months. Amazon AWS provides a highly reliable, scalable, low-cost infrastructure platform in the cloud that powers hundreds of thousands of businesses in 190 countries around the world, with data center locations in the U.S., Europe, Brazil, Singapore, Japan, and Australia, customers across all industries [1]. Millions of customers are currently leveraging AWS cloud products and solutions to build sophisticated applications with increased flexibility, scalability and reliability. Microsoft Azure is a growing collection of integrated cloud services - analytics, computing, database, mobile, networking, storage, and web - for moving faster, achieving more, and saving money [3].

1.3 Scope and Assumption

This research is conducted based on the following assumptions.

- 1. In models we develop in this dissertation, we confine our attention to information and resource sharing among tenants within a single public or community cloud. These issues in the context of multiple/hybrid clouds are important and interesting research problems but out of scope for this dissertation.
- 2. Our scope is to build the access control model in dominant IaaS cloud platforms. Models for information and resources sharing in PaaS and SaaS cloud platforms are beyond our scope. We believe building models in IaaS level will provide solid foundation for setting up

models in higher levels in cloud.

- 3. We assume that multiple communities can exist in a single public cloud. Each community consists of a subset of organizations in the cloud. One community has multiple organizations. For simplicity, we also assume one organization has only one cloud account and a user belongs to one and only one organization in the cloud.

- 4. We confine our attention of cloud services to compute service and object storage service. Focussing on cloud storage service allows us to investigate information sharing requirements between tenants, while compute service allows us to investigate resources sharing requirements between tenants. However, our models equally apply to other services in cloud system such as network service, block storage service, identity service, etc.

These assumptions will be applied to models in each cloud platforms (viz., OpenStack, AWS and Azure) that we are going to discuss in this dissertation.

1.4 Thesis Statement

We can summarize the problems in this dissertation as follows.

- *There is lack of access control models for information and resource sharing within collaborative groups in IaaS cloud platforms.*

The central thesis of this dissertation is as follows.

- *Secure sharing information and resources in IaaS cloud can be achieved by a common access control model that is enforceable in the currently dominant cloud IaaS platforms (viz., OpenStack, AWS and Azure).*

1.5 Summary of contribution

The contribution of this dissertation is that we defined a generic model for secure sharing information and resources in cloud system. We call it Secure Isolated Domain model (SID-model).

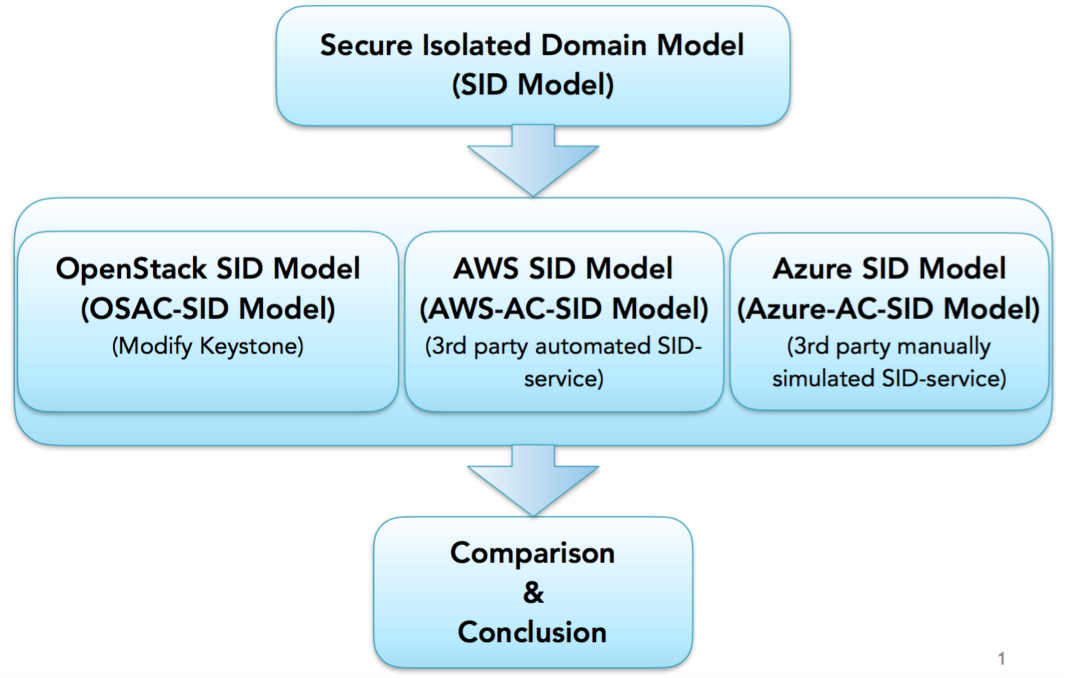


Figure 1.5: Outline of Contributions

We further explore applying this model to dominant cloud IaaS platforms, including open source cloud OpenStack, and two commercial clouds, viz., Amazon AWS and Microsoft Azure. Finally, we compare the application of the SID-model over all three cloud platforms. Figures 1.5 shows the outline of contributions.

- *Model.* We first introduce a generic Secure Isolated Domain model (SID-model) for sharing information and resources in cloud system. We provide a formal characterization of the access control model. We further formalize the concept into access control models, including administrative model and operational model. Then we apply this model to all three dominant cloud platforms (OpenStack, AWS and Azure). We introduce the basic cloud access control models for each of these dominant IaaS cloud platforms. We abstract the cloud access control models for AWS and Azure. For each of those platforms, we specify novel ways to construct SID-model to allow inter-tenant secure information and resource sharing. We extend the cloud access control model to include the capability of secure sharing information and resources.

- *Enforcement.* We discuss the enforcement details for models on each of the dominant cloud platforms. We build each of the model base on the features of each cloud platform itself. For instance, we modified the source code of OpenStack to include the feature of the model, while we add a service to include the feature of the model in AWS and Microsoft Azure. We add functionalities to demonstrate the model in each of those cloud platforms.

- *Comparison.* These cloud IaaS platforms are very different from each other in its manner and realizes IaaS. For instance, OpenStack uses global roles to control permission to access cloud resources in current version. AWS has local roles to access cloud resources and the roles can be customized. Microsoft Azure has more complex way to manage roles, including roles in Active Directory, roles managing subscriptions and roles for accessing cloud resources. We compare the application of the model in those cloud platforms. we discovered the basic resources container in those different cloud platforms and confirm the different needs with settings into the requirements of our model.

1.6 Organization of the Dissertation

In this dissertation, we will introduce the basic cloud access control models for the dominant IaaS cloud platforms, including the open source cloud platform OpenStack, and two commercial cloud platforms - Amazon AWS and Microsoft Azure. We provide a formal characterization of the access control models of these three cloud platforms. For each of the platforms, we specify novel ways to construct inter-tenant secure information and resource sharing. The dissertation outline is as follows. In chapter 2, we will introduce related work on the idea of information and resources sharing. In chapter 3, we give an access control model for secure sharing information and resources in a public cloud. We formalize the model, giving both administrative model and operational model. In chapters 4, 5 and 6, we will introduce the model for OpenStack, Amazon AWS and Microsoft Azure respectively. For each of those platforms, we first give a formal access control model specification of the cloud platform, and then we extend the access control model to include the capability of handling information and resources sharing across tenants. We also give a formal specification of the respective administrative models of information and resources sharing. In chapter 6, we compare models on those three platforms, emphasizing the differences in implementation perspective. Finally, we conclude the dissertation in the last chapter.

Chapter 2: RELATED WORK

Information and resource sharing has been an essential research topic for decades in security research community. In traditional distributed systems, various access control and authorization solutions for sharing information have been proposed. Some of the solutions are similar to our solution.

In paper [14], the authors introduced the Secure Virtual Enclaves (SVE) which allows multiple organizations to share their resources in distributed systems, while retaining their autonomy over local resources. An *enclave* is a collection of computers and networks managed by the same organization and subject to the same security policy [14]. SVE realizes the collaboration by dynamically updating security policies on local computers and networks. Very similar to our model, SVE realized collaborations among a group of organizations. Different from our model, SVE allows the collaborative organizations to have direct access to local resources. Instead, our model allows sharing only by copy of local resources. Another key difference is that SVE defines security policies on local resources, while in our model, we are not changing or adding new security policies to local resources, but only to users. Our approach utilizes the character of cloud to share resources by copying resources to a shared cloud account.

In paper [11], the author proposed an approach for community authorization service to support collaboration in distributed systems. The author introduces a scalable mechanism to represent, maintain and enforce security polices during collaborations. Same as paper [14], while the approach allows resource owners to delegate some authority to communities, it still maintain ultimate control over their resources. The authors introduced a community authorization service (CAS) server to manage the policies that govern access to a community's resources. It mainly solves the fundamental problem of scalable representation and enforcement of access policy within distributed virtual communities. In contrast, our model uses the existing cloud authorization mechanism to authorize users and cloud objects. Expressing policies in terms of direct trust relationship between producers and consumers leads to problems of scalability, flexibility, expressibility and lack of policy hierarchy [11]. We avoid these problems by applying cloud authorization mechanism directly. Moreover, since organizations in the same cloud shares same authentication and authorization center, it is simple and easy for organizations to exchange information and resources with each other. The CSP as a common trusted party, plays an important function in this regard.

In paper [6], the authors proposed a family of coalition-based access control (CBAC) models, to allow member organizations effectively share specific data and functionality in a coalition, while avoiding inappropriate access to their resources. The models capture the entities involved resource sharing in coalition and identify the interrelationships among these entities, as well as present coalition-focused access policies and enforcement mechanisms. Same as previous two papers, this paper is also for distributed system.

More recently a concept of sharing information and resources in a group of users, called Group-Centric Secure Information Sharing (g-SIS) [9] has been developed. The g-SIS model changes the emphasis of the access control unit from individual users and objects to a group of users and objects, which is suitable for collaboration scenarios. Instead of defining policies for each piece of information and resources, g-SIS allow to apply policies on a group of organizations with their users and objects. Instead of giving direct access to organizations' resources, g-SIS model bring resources into the group by copy. Policies are based on each group of organizations.

All in all, the difference between our model with traditional approaches are as follows. First, we proposed our model in a IaaS cloud environment rather than distributed systems, where cloud system facilitate the collaboration in terms of unified user and role set and infrastructure for all the tenants. Second, we don't give the collaboration group direct access over the original data and resources in the organization. Instead, we transfer copies to the collaboration group. Third, we don't use a separate Community Authorization Service (CAS) [11] to manage the access control policies for the collaboration group. Instead, we utilize the setting of roles, users and policies of the cloud to facilitate the access control over the collaboration group.

In the context of cloud, in [15], the authors proposed trust relationships established between tenants to facilitate sharing. This makes collaborations easy to implement by simply adding trust relationships among tenants in cloud. This provides users from one tenant direct access over the resources of another tenant, which in our case we try to avoid by sharing copies.

In paper [16], the author introduced a design and implementation of cloud-based assured information sharing system in SaaS. Recently, Microsoft unveiled a new intelligence-sharing platform: Interflow, which is a PaaS based system for sharing attack information among organizations. There is a lack of IaaS cloud-based information and resource sharing both in the literature and in industry. Our approach is built in IaaS cloud, and aims to provide a formal model and enforcement guide for implementation in real IaaS cloud environments.

Part of the concept we used to build our models comes from Group-Centric Secure Information Sharing (g-SIS) [9], which introduces group-based information and resources sharing - a model to control access among a group of users and objects. G-SIS model is well suited to the collaborative community scenario. In particular, g-SIS enables sharing using copies of original information, versus traditional information and resource sharing approaches which give access to original information and resources [6, 11, 14] to enable sharing. Sharing by copy gives additional security protection over the original information and resources, since access to the copies can be provided in a tightly controlled environment.

We also applied the concept from community cyber security [13], in which the authors present a model for Secure Information Sharing (SIS) in the specific domain of community cyber security. Compared to traditional discretionary access control, mandatory access control and role-based access control, this model gives a new perspective to deal with the scenario of cyber incidents. It can dynamically configure a system to facilitate SIS scenarios during a cyber incident life cycle. The model introduces concept of core group, open group, incident group and domain experts. Core group is for the community constantly exchange cyber informations and monitor the security level of the community, to determine if something “out of the ordinary” occurs in the community [13]. Open group is an open forum for users in the community to be involved with general cyber security issues. Incident group is for specific incident happened in the community. During the life cycle of the cyber incident, users in the community and domain experts from outside could be added to the incident group responding to the incident. Domain experts provide professional expertise to solve problems during the life cycle of a cyber incident. We incorporate these concepts into our model and specify the formalizations.

We have presented several access control models for secure information and resources sharing in a collaborative community of organizations in cloud IaaS. We developed the OpenStack Access Control model with SID extension (OSAC-SID) [18], which is a basic model for organizations sharing information in a OpenStack cloud platform. We also designed the advanced Hierarchical OpenStack Access Control model with SID extension (OSAC-HMT-SID) [21], which provides organizations additional cyber security control with routine cyber information collection and processing, a community security committee and a public security forum in the community. Then we explored the model in current dominant comercial cloud IaaS platfroms AWS and Azure, resulted in two papers [19] and [20]. Our approach aims to provide formal models and enforcement guides for implementation in real environment.

Chapter 3: SECURE ISOLATED DOMAIN (SID) MODEL

In this chapter, we present a Secure Isolated Domain (SID) model, which is an abstraction and consolidation of all the models we have published before in papers [18], [19], [20] and [21], with some modifications. We call it SID-model. SID-model provides an access control solution for securely sharing information and resources in cloud platforms. Figure 3.1 shows the SID-model structure. In this and other figures in this dissertation, the arrows denote binary relations with the single arrowhead indicating the one side and double arrowheads the many side. We use circle to represent entities which can be created multiple times in OpenStack, while rectangle represents entities which can only be created once. SID-model is a common model that will be applied to all three cloud IaaS platforms mentioned earlier, viz., OpenStack, AWS and Microsoft Azure.

In SID-model, we are using the same cloud model concept from previous work [15], including concepts of cloud services, object types, operations, permissions, etc. Also, we present SID-model from two perspective: administrative model and operational model. We cover the basic operations in both models. We confine our operational model to include only two typical cloud services as representatives: computing service and object storage service. The operational model can be extended to other cloud services as well.

In the operational model, we focus on showing how and what operations a normal user can issue in the model. There are lots of services a cloud platform can provide.

3.1 Components

SID-model has thirteen components: Organization Accounts (OA), Users (U), Experts Accounts (EA), Expert Users (EU), Roles (R), Services (S), Object Types (OT), Operations (OPR), Secure Isolated Domains (SID), Secure Isolated Projects (SIP), Core Projects (CP), Open Projects (OP) and Resources (RS) . We also introduce other entities, including policies, credentials, storage containers (SC), storage container objects (CO) and virtual machines (VM), which are implicitly included in the model.

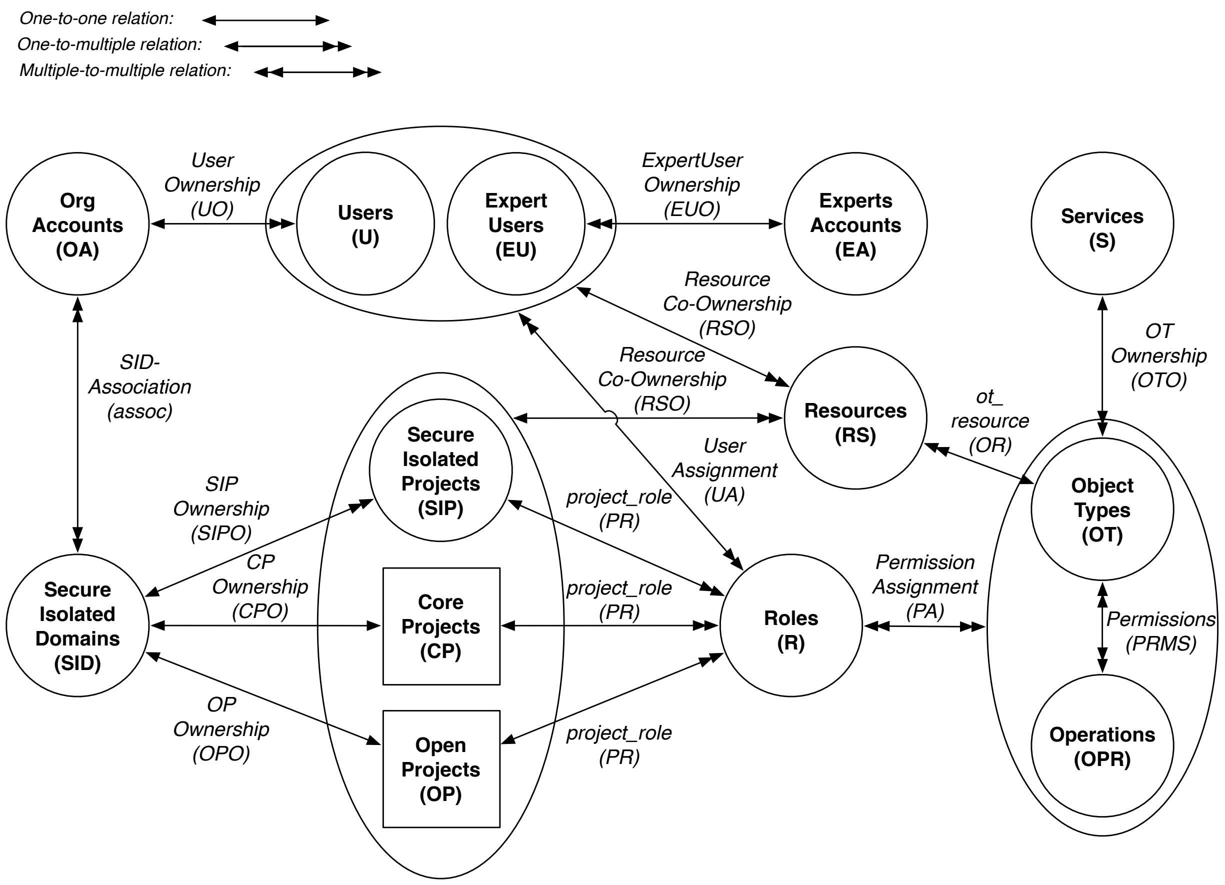


Figure 3.1: Secure Isolated Domain Model (SID-model)

Organization Accounts: To have its own public cloud resources, an organization needs to open a cloud account. A cloud account allows an organization to own specific (virtual) cloud resources that can be accessed through cloud services.

Users and Groups: Users represent individuals who can be authenticated by a cloud system and authorized to access cloud resources through a cloud account. A group is simply a set of users. Users and Groups belong to a cloud account. The existence of groups is for the convenience of managing multiple users as a single unit. Each policy associated to a group will apply to all group members. For simplicity, in SID-model we use term users to represent both users and groups components, since group is just a set of users. Later in the chapter, when we mention groups, we mean to represent a group of organizations.

Experts Accounts: Experts accounts are cloud accounts that exist outside the community of the group of organizations in the same cloud, giving experts access to cloud resources.

Expert Users: To get outside professionals involved, expert users [12] are introduced to SID-model. Expert users don't belong to the community of organizations. They are from other professional security organizations in the same public cloud. These experts bring different cyber security skills. For instance, they may come from an IT consultant company which focusses on specific cyber attacks. They may be cyber security law enforcement officers specializing in cyber crime. The involvement of expert users is to help organizations handle cyber collaborations more effectively. A sid maintains an expert users list which is available for collaborations inside the sid.

Roles: Users are assigned to a role to get permissions to access to cloud resources. Roles allow users to have permissions to access cloud resources, for instance virtual machines (VMs), storage, networking and etc. Roles could be different collections of meta permissions like read and write toward a specific type of resource.

Services [15]: Services refer to cloud services provided to cloud customers. Cloud Service Provider (CSP) leases cloud resources to its customers in terms of services, such as compute, storage, networking, authentications and authorizations, database, etc.

Object Types and Operations [15]: An Object Type represents a specific type of cloud resources objects. From the CSP's viewpoint, objects are essentially services. We define object types as particular service types the cloud provides. For instance, with compute service, the object type is a virtual machine; with the storage service, the object type is a storage container, etc. General operations are create, read, update and delete (CRUD) interacting with object types. For instance, permission of deleting a virtual machine is a combination of delete operation and virtual machines (VM) object type.

Permission Assignment and Policies [15]: In current popular cloud systems, users' permissions over cloud services and resources are defined in policy files. Usually policy files are controlled by a policy engine. In some dominant cloud systems, policies can be attached to a user, a user group, a role or a specific cloud resource. By defining a policy, a user or a group of users gains permissions to corresponding cloud resources. The policy defines the actions which the user will perform and cloud resources on which the actions will apply. Multiple permissions can be defined in one policy file. Multiple policy files can be attached to one entity. Dominant cloud systems achieve permission assignment in a manner via the policies attached to various relevant entities.

Virtual Machines: Virtual Machines is an object type for cloud compute service. Users can launch virtual machines in a cloud account which give them great power of computing.

Storage Containers and Storage Container Objects: Storage Containers and Storage Container Objects are object types for cloud storage service. We define a storage container as a common object storage resources boundary for a cloud platform, to allow storing storage container objects like files in the cloud.

Resources: Resources refer to cloud assets which can be owned by users. Cloud assets are cloud resources such as virtual machines, databases, storages, etc. Since the only way for users to access resources is through a project (either a sip, core project or open project), which is a cloud resources container, we also define that a project has ownership over its resources.

Credentials: Cloud system credentials are used for both authentication and authorization. Account owners can create users with their own security credentials to allow these users to access a cloud system services and resources. Account owners can also grant external federated users from other accounts with temporary security credentials to allow them to access the account's a cloud system services and resources.

Secure Isolated Domains: Secure Isolated Domain [18] is a special exclusive isolated domain, holding security information and resources for cross-organizational security collaborations. A sid provides an administrative boundary for cyber security information and resource collection and analysis, and a secure isolated environment for cyber security collaborations in a community of organizations. A sid holds two permanent projects: a core project and an open project. A sid also holds all Secure Isolated Projects (sips) designed for cyber incident response and security collaboration within this community of organizations. Figure 3.2 shows a sid with its composition. One sid serves for one community of organizations.

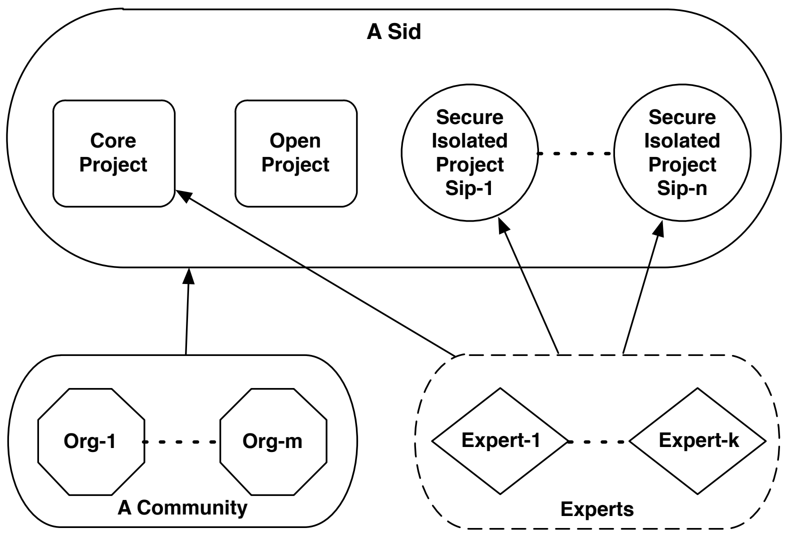


Figure 3.2: SID Composition

Projects: A project is a cloud resources container. It has strict boundaries isolating cloud resources from each other. A user accesses cloud resources though a project. In SID-model, projects include secure isolated projects, core projects and open projects.

Secure Isolated Projects: Secure Isolated Project [18] is a special project with constraints over its user membership. It is used to collect, store and analyze cyber security information for specific security reasons. A sip provides a controlled environment for a group of organizations within the community to collaborate and coordinate on cyber incidents and other security issues.

Core Projects: Core project is a shared project holding cyber security committee [12] for the community of organizations. Each organization in the community has at least one representative security user in the committee.

Open Projects: Open project is an open shared project where users from the community of organizations share common cyber security information and resources [12]. It is a common forum for all community users to share general security information. Information published in open project is simply public to every user who is in the project.

With the concepts described above, we formalize SID-model as follows.

Definition 1 SID-model has the following components.

- OA, U, EA, EU, R, S, OT, OPR, SID, SIP, CP, OP and RS are finite sets of existing organization accounts, users, experts accounts, expert users, roles, services, object types, operations, secure isolated domains, secure isolated projects, core projects, open projects and resources respectively in a cloud system.

- Virtual Machines (VM) is object type for cloud compute service.

- Storage Containers (SC) and Storage Container Objects (CO) are object types for cloud object storage services.

- User Ownership (UO) : is a function UO : $U \rightarrow OA$, mapping a user to its owning account. Equivalently viewed as a many-to-one relation $UO \subseteq U \times OA$.

- Secure Isolated Project Ownership (SIPO) : is a function SIPO : $SIP \rightarrow SID$, mapping a single secure isolated project to its owning sid. Equivalently viewed as a many-to-one relation $SIPO \subseteq SIP \times SID$.

- Core Project Ownership (CPO) : is a function CPO : $CP \rightarrow SID$, mapping a single core project to its owning sid. Equivalently viewed as a one-to-one relation $CPO \subseteq CP \times SID$.

- Open Project Ownership (OPO) : is a function OPO : $OP \rightarrow SID$, mapping a single open project to its owning sid. Equivalently viewed as a one-to-one relation $OPO \subseteq OP \times SID$.

- Resource Co-Ownership (RSO) : is a function RSO : $RS \rightarrow ((SIP \cup CP \cup OP), (U \cup EU))$, mapping resources to its owning project and user. Equivalently viewed as a many-to-one relation $RSO \subseteq RS \times ((SIP \cup CP \cup OP) \times (U \cup EU))$.

- Object Type Owner (OTO) : is a function OTO : $OT \rightarrow S$, mapping an object type to its owning service. Equivalently viewed as a many-to-one relation $OTO \subseteq OT \times S$.

- SID association (assoc) : is a function assoc : $SID \rightarrow 2^{OA}$, mapping a SID to all its member organization accounts.

- Permission Assignment (PA) : is a function PA : $PERMS \rightarrow R$, mapping permissions to roles. Equivalently viewed as a many-to-many relation $PA \subseteq PERMS \times R$.

- User Assignment (UA) : is a function UA : $U \rightarrow PRP$, mapping users to project-role pairs. Equivalently viewed as a many-to-many relation UA $\subseteq$ U $\times$ PRP.

- ot_resource (OR) : is a function OR : OT $\rightarrow$ RS, mapping object types to resources. Equivalently viewed as a one-to-many relation OR $\subseteq$ OT $\times$ RS.

- project_role (PR) : PR = (SIP $\cup$ CP $\cup$ OP) $\times$ R, is a set of project-role combinations.

- Permissions (PERMS) : PERMS = OT $\times$ OPR, is a set of permissions.

3.2 Administrative Model

To make role assignment simple and clear, we constrain roles in two types: administrative roles and member roles, which separately denotes the permission of being able to manage users and permissions only for accessing cloud resources. We use one admin role *SIDadmin* to represent all admin permissions a user can get from the cloud. We use one member role *SIDmember* to represent all normal roles a user can get in a resource container. Admin users have the capability to add and remove other users from their home organizations to core project and sips. Member users can be added/removed from/to a project inside a sid. Member users are the those who actually have access to the real cloud services and resources, like creating or deleting a virtual machine.

One sid is associated with one community in a cloud. The number of organizations associated with the sid is fixed. Let *uSet* denotes the fixed group of security admin users, each of which represent one and only one organization in the community. Each organization in the community has equal limited administrative power in the sid, which is carried through *uSet*. Each sid maintains *uSet* as a core group [12] of sid admin users. Only users from *uSet* later can dynamically create sips in the sid.

A sid is initially set up with a core project and an open project. The core project and open project are created when the sid is created. Each organization can join different sid with different communities of organizations. Each of these sids are isolated from each other. Inside the sid, organizations can request multiple sips for convenience of different cyber collaborations. The number of sips depends on how many collaborations are initialized by the group of organizations. Organizations can automatically request to create and delete sips, as well as add or remove users

Table 3.1: SID Administrative Model

Operation	Authorization Requirement	Update
SidCreate(adminu, uSet, sid) */* An admin user representing uSet creates a sid */*	adminu $\in$ *uSet* $\wedge$ adminu $\in$ U $\wedge$ sid $\notin$ SID	SID′ = SID $\cup$ {sid}; assoc(sid) = $\bigcup_{adminu \in uSet}$ UO(adminu); CP′ = CP $\cup$ {cp}; CPO(cp) = sid; OP′ = OP $\cup$ {op}; OPO(op) = sid; UA′ = (uSet, *SIDadmin*) $\cup$ UA; PR′ = PR $\cup$ {(cp, *SIDadmin*), (op, *SIDadmin*)}.
SidDelete(adminu, uSet, sid) */* An admin user representing uSet deletes the sid*/*	adminu $\in$ *uSet* $\wedge$ adminu $\in$ U $\wedge$ (adminu, *SIDadmin*) $\in$ UA $\wedge$ assoc(sid) = $\bigcup_{adminu \in uSet}$ UO(adminu) $\wedge$ sid $\in$ SID	SID′ = SID - {sid}; assoc(sid) = NULL; CP′ = CP - {cp}; CPO(cp) = NULL; OP′ = OP - {op}; OPO(op) = NULL; UA′ = UA - (uSet, *SIDadmin*); PR′ = PR - {(cp, *SIDadmin*), (op, *SIDadmin*)}; if $\exists$ u $\in$ (U $\cup$ EU).((u, *SIDmember*) $\in$ UA), then UA′ = UA - (u, *SIDmember*); if $\exists$ sip $\in$ SIP.(SIPO(sip) = sid), then SIP′ = SIP - sip $\wedge$ PR′ = PR - {(sip, *SIDadmin*), (sip, *SIDmember*)}.
SipCreate(adminu, sip, sid) */* An admin user representing uSet creates a sip */*	adminu $\in$ U $\wedge$ (adminu, *SIDadmin*) $\in$ UA $\wedge$ UO(adminu) $\in$ assoc(sid) $\wedge$ sip $\notin$ SIP	SIP′ = SIP $\cup$ {sip}; SIPO(sip) = sid; PR′ = PR $\cup$ {(sip, *SIDadmin*)}.
SipDelete(adminu, sip, sid) */* An admin user representing uSet deletes a sip*/*	adminu $\in$ U $\wedge$ (adminu, *SIDadmin*) $\in$ UA $\wedge$ UO(adminu) $\in$ assoc(sid) $\wedge$ SIPO(sip) = sid	SIP′ = SIP - {sip}; SIPO(sip) = NULL; PR′ = PR - {(sip, *SIDadmin*)}.

from or to sips. Admin users from *uSet* also can add or remove users from/to core project. With the creation of a sid, admin users from *uSet* automatically get limited administrative permission in a core project in a sid, which is represented by role *SIDadmin*. A normal user from the community automatically get permissions to open project with role *SIDmember*.

The administrative aspects of SID-model are discussed informally below. A formal specification is given in Table 3.2.

Table 3.2: SID Administrative Model (continued)

Operation	Authorization Requirement	Update
UserAdd(adminu, u, p, sid) */* Admin users add a user from his home domain to a cp, op or sip */*	adminu $\in$ U $\wedge$ (adminu, *SIDadmin*) $\in$ UA $\wedge$ (p, *SIDadmin*) $\in$ PR $\wedge$ u $\in$ U $\wedge$ UO(u) = UO(adminu) $\wedge$ p $\in$ (CP $\cup$ OP $\cup$ SIP) $\wedge$ (CPO(p) = sid $\vee$ OPO(p) = sid $\vee$ SIP(p) = sid)	UA$'$ = UA $\cup$ {(u, *SIDmember*)}.
UserRemove(adminu, u, p, sid) */* Admin users remove a user from a cp, op or sip */*	adminu $\in$ U $\wedge$ (adminu, *SIDadmin*) $\in$ UA $\wedge$ (p, *SIDadmin*) $\in$ PR $\wedge$ u $\in$ U $\wedge$ UO(u) = UO(adminu) $\wedge$ p $\in$ (CP $\cup$ OP $\cup$ SIP) $\wedge$ (CPO(p) = sid $\vee$ OPO(p) = sid $\vee$ SIP(p) = sid) $\wedge$ (u, *SIDmember*) $\in$ UA $\wedge$ (p, *SIDmember*) $\in$ PR	UA$'$ = UA - {(u, *SIDmember*)}.
EUserAdd(adminu, eu, p, sid) */* Admin users add an expert user to a cp or sip */*	adminu $\in$ U $\wedge$ (adminu, *SIDadmin*) $\in$ UA $\wedge$ (p, *SIDadmin*) $\in$ PR $\wedge$ eu $\in$ EU $\wedge$ p $\in$ (CP $\cup$ SIP) $\wedge$ (CPO(p) = sid $\vee$ SIPO(p) = sid)	UA$'$ = UA $\cup$ {(eu, *SIDmember*)}.
EUserRemove(adminu, eu, p, sid) */* Admin users remove an expert user from a cp or sip */*	adminu $\in$ U $\wedge$ (adminu, *SIDadmin*) $\in$ UA $\wedge$ (p, *SIDadmin*) $\in$ PR $\wedge$ eu $\in$ EU $\wedge$ p $\in$ (CP $\cup$ SIP) $\wedge$ (CPO(p) = sid $\vee$ SIPO(p) = sid) $\wedge$ (eu, *SIDmember*) $\in$ UA $\wedge$ (p, *SIDmember*) $\in$ PR	UA$'$ = UA - {(eu, *SIDmember*)}.

Create a sid: For each community of a group of organizations, there is one sid serving for cyber security purpose. For such communities of organizations who are going to have cyber collaborations, one sid is assigned to be associated with one community. The number of organizations associated with a sid is fixed. Let *uSet* denotes the fixed group of security admin users, each of which represent one and only one organization in the community. Each sid is associated with a certain number of organizations accounts. With different combinations (communities) of organizations in the cloud, the total number of possible sids in the cloud is $2^{|A|}$.

Each organization in the community has equal limited administrative power in the sid, which is carried through *uSet*. SID-model maintains each *uSet* as a core group [12] of a corresponding sid admin users. Only users from *uSet* later can dynamically create sips in the sid.

Inside the sid, organizations can request multiple sips for convenience of different cyber collaboration purpose. The number of sips depends on how many collaborations are initialized by the group of organizations. A sid is initially set up with a core project and an open project, while organizations can then automatically request to create and delete sips, as well as add or remove users from/to sips. Admin users from *uSet* also can add or remove users from/to core project and open project. With the initialization of a sid, admin users from *uSet* automatically get limited administrative permission in core project and open project in a sid, which is represented by role *SIDadmin*. A normal user from the community can be added to open project with role *SIDmember*.

Delete a sid: After all collaborations are finished, organizations can choose to delete a existing sid. The delete command is issued by a security admin user from (*uSet*) who represents the group of organizations. All existing sips, cp and op, with information data and resources are securely deleted with sid delete. All users assigned to any sips, cp and op is removed from it.

Create a sip: An organization security admin user representing a set of security admin users *uSet* create a sip for an cyber incident response among the community of organizations. Each organization in the sip has equally limited administrative power, which is represented by role *SIDadmin*, which gives the sip admin users the permission to add and remove other users from their home domain to the sip. Multiple sips for different purpose can be created in one sid.

Delete a sip: After the collaboration is finished, a sip needs to be securely deleted. The delete command is issued by the same security admin user from (*uSet*) who issued the sip creation. All information data and resources are securely deleted from the sip. All users assigned to the sip are removed from it.

Add or remove a user to or from a core project or sips: Core project and sips admin users are the set of security administrative users (*uSet*) from the community of organizations. These limited administrative users can add/remove users of their organizations to/from the core project and sips. All the users added to the core project or sips are existing users from an organization's account. The limited administrative users don't have the permission to create new users or delete an existing user. They can only add existing users to the core project or sips. When users are removed from the core project or a sip, they will lose the access to corresponding information and resources in the core project or the sip, regardless of the ownership of the piece of information in the past. Admin users in core project or a sip can see all users added from the community of organizations, as well as information and resources they bring in, which means there are no hidden users, information and resources in a core project or a sip.

Add or remove a user to or from an open project: Every user in the collaborative community of organizations is allowed to join the open project. Users in open project have equal but limited permissions. They can share cyber data, but have no control over other users. We use role *SIDmember* to represent this limited permission. Organization security admin users add/remove normal users from their organizations to/from the open project. Users will not be able to access and share any data once they leave the open project.

Add or remove an expert user to or from a core project or sips: Expert users are needed when external cyber expertise need to be involved. For instance, a cyber incident needs experts from security consultant companies, government cyber experts, cyber polices, etc. Sid services maintain a relation with external experts. Expert users can be added/remove to/from a core projects and sips as a member. Users from *uSet* can request to add/remove expert users to/from the core project or a sip. There are situations when an existing expert user in the core project or a sip needs to be removed. For instance, at the end of a cyber collaboration, an unneeded expert user will be securely deleted. After the expert user is deleted, the user will lose all access to any information and resources in the core project or a sip.

3.3 Operational Model

In the operational model, we focus on showing how and what operations a normal user can issue in the model. There are lots of services a cloud platform can provide. We choose two typical cloud services as representatives: computing service and object storage service. Correspondingly, computing service has an object type of virtual machines (VM), and storage service has an object type of storage containers (SC) and storage container objects (CO). We will show two core operations including create and delete. Create method allows users to create a new instance of virtual machine or a storage container in a core project, open project or a sip. Delete method allows users to delete an existing instance of a virtual machine or storage container in a project. In Table 3.3, we give the details of operational model.

After a user is assigned to a core project, open project or a secure isolated project in a sid, the user can issue following operations:

Table 3.3: SID Operational Model

Operation	Authorization Requirement	Update
CreateVM(vm, p, u) *$/^*$ A user creates a vm $^*/$*	$vm \notin RS \wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge \exists$ (perms, r) $\in$ PA.(perms = (vm, create) $\wedge$ (p, r) $\in$ PR $\wedge$ (u, (p, r)) $\in$ UA)	RS' = RS $\cup$ {vm}; RSO' = RSO $\cup$ {(vm, (p, u))}; OR(vm) = VM.
DeleteVM(vm, p, u) *$/^*$ A user deletes a vm $^*/$*	$vm \in RS \wedge$ RSO(vm) = {(p, u)} $\wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge \exists$ (perms, r) $\in$ PA.(perms = (vm, delete) $\wedge$ (p, r) $\in$ PR $\wedge$ (u, (p, r)) $\in$ UA	RS' = RS - {vm}; RSO' = RSO - {(vm, (p, u))}; vm = NULL.
CreateSContainer(sc, p, u) *$/^*$ A user creates a storage container $^*/$*	$sc \notin RS \wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge \exists$ (perms, r) $\in$ PA.(perms = (sc, create) $\wedge$ (p, r) $\in$ PR $\wedge$ (u, (p, r)) $\in$ UA)	RS' = RS $\cup$ {sc}; RSO' = RSO $\cup$ {(sc, (p, u))}; OR(sc) = SC.
DeleteSContainer(sc, p, u) *$/^*$ A user deletes a storage container $^*/$*	$sc \in RS \wedge$ RSO(sc) = {(p, u)} $\wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge \exists$ (perms, r) $\in$ PA.(perms = (sc, delete) $\wedge$ (p, r) $\in$ PR $\wedge$ (u, (p, r)) $\in$ UA	RS' = RS - {sc}; RSO' = RSO - {(sc, (p, u))}; sc = NULL.
CreateObject(co, sc, p, u) *$/^*$ A user creates a storage container object $^*/$*	$co \notin RS \wedge sc \in RS \wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge$ RSO(sc) = (p, u) $\wedge \exists$ (perms, r) $\in$ PA.(perms = (co, create) $\wedge$ (p, r) $\in$ PR $\wedge$ (u, (p, r)) $\in$ UA)	RS' = RS $\cup$ {co}; RSO' = RSO $\cup$ {(co, (p, u))}; OR(co) = CO.
DeleteObject(co, sc, p, u) *$/^*$ A user delete a storage container object $^*/$*	$co \in RS \wedge$ RSO(co) = {(p, u)} $\wedge sc \in RS \wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge$ RSO(sc) = (p, u) $\wedge \exists$ (perms, r) $\in$ PA.(perms = (co, create) $\wedge$ (p, r) $\in$ PR $\wedge$ (u, (p, r)) $\in$ UA)	RS' = RS - {co}; RSO' = RSO - {(co, (p, u))}; co = NULL.

Create or delete a vm: A user can create/delete a virtual machines in a core project, open project or sip, to which the user is assigned.

Create or delete a container: A user can create/delete a storage container in a core project, open project or sip, to which the user is assigned. A storage container holds container objects.

Create or delete a storage container object: A user can create/delete a container object in a storage container in a core project, open project or sip, to which the user is assigned.

Chapter 4: SID-MODEL IN OPENSTACK CLOUD IAAS

Part of content from this chapter has been published in paper [18].

OpenStack is a cloud platform which provides a large scale of resources like compute, storage, networking and so on. As the biggest open source cloud infrastructure platform, OpenStack provides a set of core services to facilitate scalability and elasticity of cloud architecture. Those core services include compute service (Nova), identity service (Keystone), block storage service (Cinder), object storage service (Swift), image service (Glance), networking service (Neutron) and Dashboard service (Horizon), as shown in Figure 4.1.

Nova provides the support of management of virtual machines at big scale, multi-tiered application and high performance computing level. It allows users to create their own virtual machines and run the instances. Image service provides users with images (OS, software, configurations, etc.), which are used to create virtual machines. Swift supports storing and retrieving data in the cloud. It allows users to store their data as Swift objects. Cinder provides block storage which is attached to a virtual machine as a storage volume. Neutron provides users with networking services such as security groups, IP address management, DNS, and etc. For instance, different virtual machines can be connected using virtual routers. Keystone provides users with security services, such as authentication and authorization. Horizon is the web-based dashboard where users can access all the services through a graphic user interface. OpenStack also provides shared file systems which supports management of shared file system in a multi-tenant cloud environment. In addition, OpenStack provides command line based clients to interface with each of those services.

4.1 OSAC Model

The OSAC model presented in this chapter has been previously described in paper [15]. The core OpenStack Access Control (OSAC) model based on the OpenStack Identity API v3, is shown in Figure 4.2. The OSAC model consists of eight entities: users, groups, projects, domains, roles, services, operations, and tokens. Users represent people who are authenticated to access OpenStack

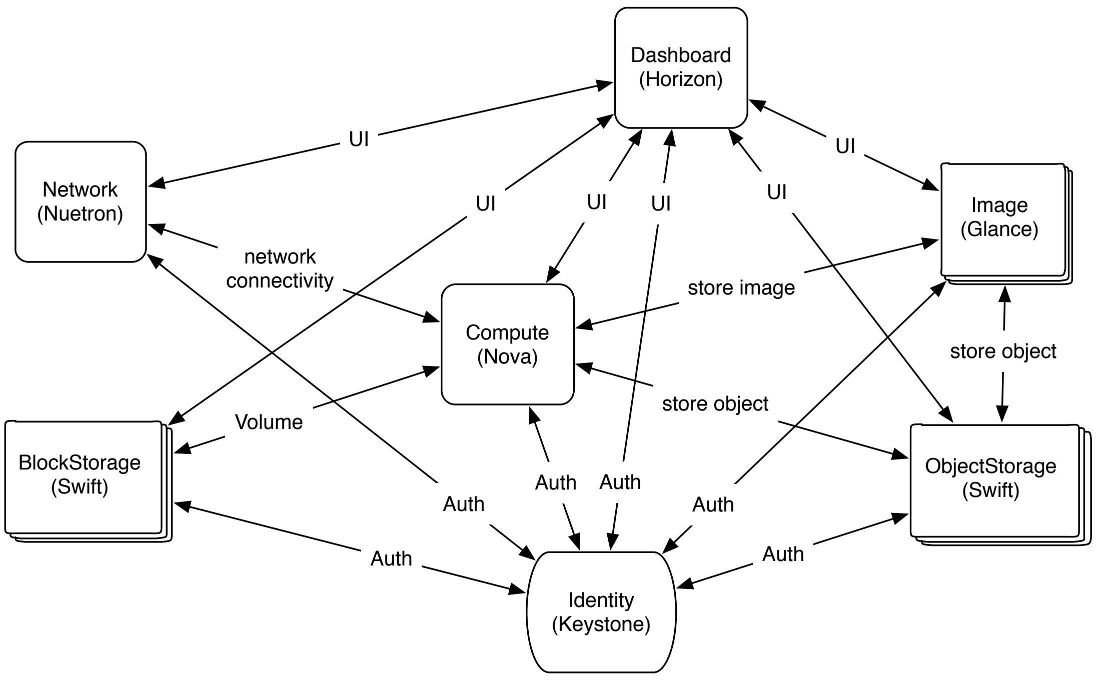


Figure 4.1: Architecture of OpenStack [5]

cloud resources while a group is a set of users. Projects define a boundary of cloud resources—a resource container in which users can get access to the services the cloud provides, such as virtual machines, storages, networks, and so on. Domain is a higher level concept that equates to a tenant of the CSP. The projects in a domain represent the administrative boundary of its users and groups. Projects allow tenants to segment their resources and to manage their users' scope of access to those resources. Roles are global, which are used to specify access levels of users to services in specific projects in a given domain. Note that users are assigned to projects with a specific set of roles. Object type and operations pairs define actions which can be performed by end users on cloud services and resources. Users authenticate themselves to Keystone and obtain a token which they then use to access different services. The token contains various information including the domain the user belongs to, and the roles of the users in specific projects in that domain. We elaborate the model and these concepts further below.

Users/Groups: Users are individuals who have access to OpenStack cloud. A group is a set of users who are grouped together for convenience of management.

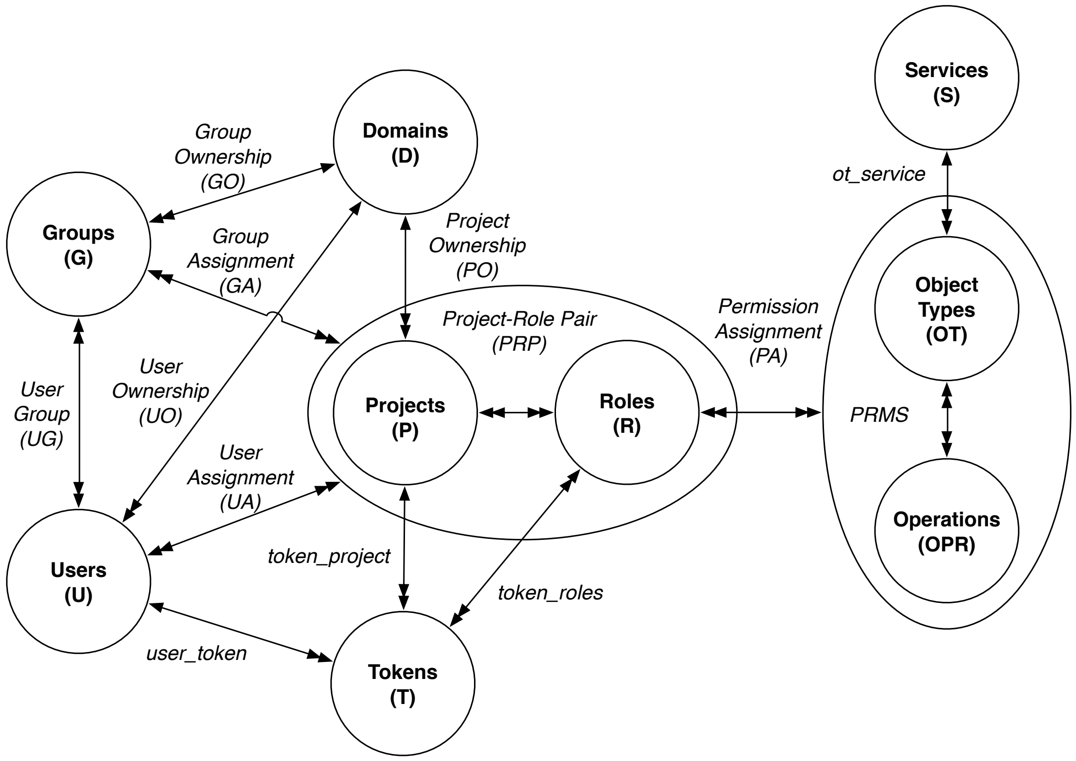


Figure 4.2: OpenStack Access Control (OSAC) Model [15]

Domains and Projects: In OpenStack, a project can only belong to one domain. A user can only belong to one home domain but they can be assigned to multiple projects, which can be distributed in different domains. That is, the ownership of users and projects can be defined by assigning them to a domain. Note that users in a domain are powerless unless they are assigned to a project with a particular role. Typically, domains are created by a CSP for its tenants. A domain admin is an admin user of that domain (tenant).

Roles: Role defines the accesses of cloud services and resources the user can have. By assigning a role to a user, one can specify different access rights for the user. For instance, by assigning a role of $Member$ to user, the user receives all operational rights over the resources in a project; by assigning a role of $admin$ to a user, the user receives admin rights over a project. The accesses defined by roles are enforced by a policy engine in the cloud based on policy files where the roles are defined.

Tokens: There are two types of tokens in OpenStack. One is an unscoped token, which is used for initial authentication to a specific domain. Using the unscoped token, the user can obtain scoped tokens that are project-specific from Keystone. If a user has membership in two domains, the user can obtain two different unscoped tokens and thereby further obtain multiple scoped tokens for projects that belong to those domains. OpenStack clients facilitate this process.

Object Types and Operations. The concept of object types allow specifying different operations for different services. For instance, consider the Nova service. The object type for Nova is VM and operations on VM include start, stop, etc. In contrast, for Swift, the object type is Container and the operations include create, upload object, download object, etc.

The feature of Hierarchical Multitenancy (HMT) [5] was added to OpenStack since Juno release. It changes OpenStack from the flat domain-projects structure to a hierarchical domain-parent project-child project tree structure. Prior to Juno release, OpenStack allows tenants to have domains with flat projects in them. Hierarchical Multitenancy allows tenants to have hierarchical project trees in a domain. However, in this dissertation, HMT feature is not a main concern of the model. For more details of HMT, please refer to paper [21].

The formalization of the OSAC model is as follows. We fully refer this formalization from paper [15], to help understand the following OpenStack SID model, since it was build upon OSAC model.

4.1.1 Components

Definition 1. OSAC model has the following components.

- U, G, P, D, R, S, OT and OPR are finite sets of existing users, groups, projects, domains, roles, services, object types and operations respectively in an OpenStack cloud system.

- User Ownership (UO) : is a function UO : U $\rightarrow$ D, mapping a user to its owning domain. Equivalently viewed as a many-to-one relation UO $\subseteq$ U $\times$ D.

- Group Ownership (GO) : is a function GO : U $\rightarrow$ D, mapping a group to its owning domain. Equivalently viewed as a many-to-one relation GO $\subseteq$ G $\times$ D.

- Object Type Owner (OTO) : is a function OTO : OT $\rightarrow$ S, mapping an object type to its owning service. Equivalently viewed as a many-to-one relation OTO $\subseteq$ OT $\times$ S.

- User Group (UG) : is a function UG : U $\rightarrow$ G, mapping users to groups where the user and group are owned by the same domain. Equivalently viewed as a many-to-many relation UG $\subseteq$ U $\times$ G.

- Permission Assignment (PA) : is a function PA : PERMS $\rightarrow$ R, mapping permissions to roles. Equivalently viewed as a many-to-many relation PA $\subseteq$ PERMS $\times$ R.

- User Assignment (UA) : is a function UA : U $\rightarrow$ PRP, mapping users to project-role pairs. Equivalently viewed as a many-to-many relation UA $\subseteq$ U $\times$ PRP.

- Group Assignment (GA) : is a function GA : G $\rightarrow$ PRP, mapping groups to project-role pairs. Equivalently viewed as a many-to-many relation GA $\subseteq$ G $\times$ PRP.

- PRP = P $\times$ R, the set of project-role pairs.

- PERMS = OT $\times$ O, the set of permissions.

- user_tokens: is a function user_tokens : U $\rightarrow 2^{\mathrm{T}}$, mapping a user to a set of tokens; correspondingly, token_user is a function token_user : T $\rightarrow$ U, mapping of a token to its owning user.

- token_project: is a function token_project : T $\rightarrow$ P , mapping a token to its target project.

- token_roles: is a function token_roles : T $\rightarrow 2^{\mathrm{R}}$, mapping a token to its set of roles. Formally, token_roles(t) = $\{$r $\in$ R$|$(token_user(t),(token_project(t),r)) $\in$ UA$\}$ $\cup$ ($\bigcup_{g \in user\_groups(token\_user(t))}$ $\{$r $\in$ R$|$(g, (token_project(t), r)) $\in$ GA$\}$).

- avail_token_perms: is a function avail_token_perms : T $\rightarrow 2^{\mathrm{PERMS}}$, mapping the permissions available to a user through a token. Formally, avail_token_perms(t) = $\bigcup_{r \in token_roles(t)}$$\{$perm $\in$ PERMS$|$(perms,r) $\in$ PA$\}$.

- ot_service: is a function ot_service : OT $\rightarrow$ S , mapping an object typle to its owning service.

4.2 Discussion of Models Possibilities

Informed by OSAC model, we discuss various alternatives in designing information and resource sharing models in OpenStack IaaS cloud platform. Recall that a domain represents a tenant of the CSP and domains can contain multiple projects, where each project is a resource boundary as specified by the roles assigned to the user in that project. Also recall that domain admin is a super

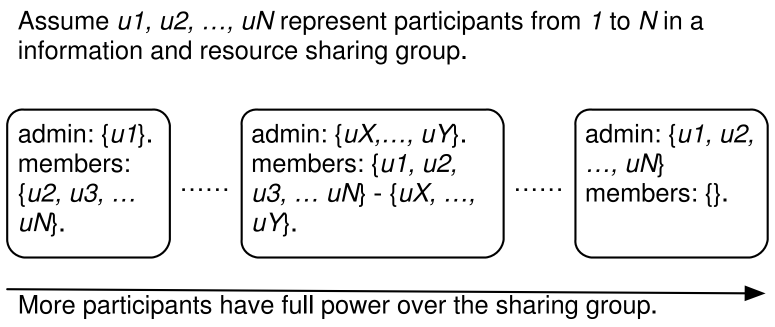

Assume *u1, u2, ..., uN* represent participants from *1* to *N* in a information and resource sharing group.
admin: {*u1*}. members: {*u2, u3, ... uN*}.
......
admin: {*uX,..., uY*}. members: {*u1, u2, u3, ... uN*} - {*uX, ..., uY*}.
......
admin: {*u1, u2, ..., uN*} members: {}.
More participants have full power over the sharing group.

Figure 4.3: From Administrative Perspective of Modeling

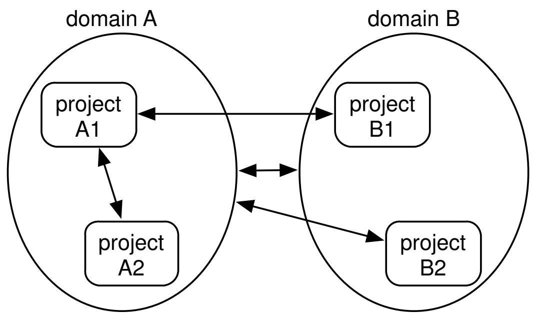


Figure 4.4: From Operational Perspective of Modeling

user who takes charge of operations inside a domain, including creating new projects, creating and adding users to a project, and so on. With the assumption that each domain represents an organization in a cloud platform, each project inside the domain can represent either a department or a temporary program in that organization. Domain admin roles are assigned to people who have the super power over the organization, such as the capability to manage the departments and initiate a new event in the organization. In the following, we conceptually refer to the sharing platform as “group”. We use the term group informally. Specifically, by group we do not mean the OSAC group component unless explicitly stated otherwise. Later, we discuss how exactly we model this group in OpenStack cloud platform.

We can model information and resources sharing from three perspectives: administrative, operational and control. From administrative perspective of modeling, we assume we have n participants in the information and resource sharing group, we can have n different levels of administrative controls, from one participant being in charge of the group to all the participants being

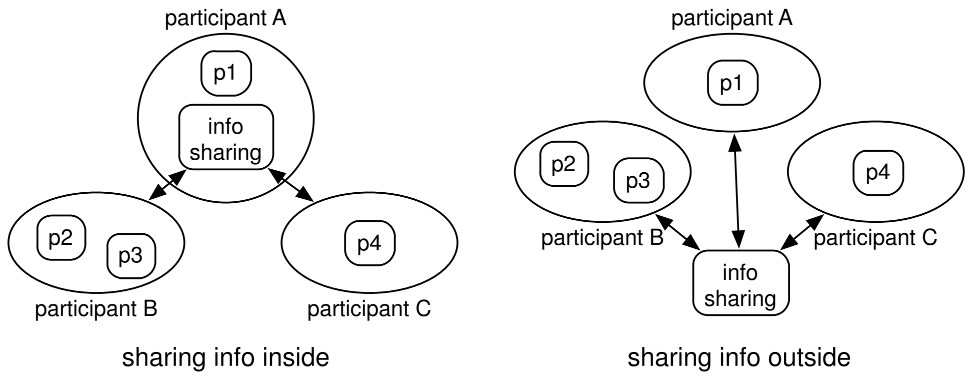


Figure 4.5: From Control Perspective of Modeling

in charge of the group, as shown in Figure 4.3. In the model where one participant has the absolute control over the collaborating group, this participant has full access to all the information and resources in the shared group, as well as having full power in determining shared group members, and which user can have what level of access over what information and resources inside the shared group. In the other extreme case of information and resources sharing administrative control, all the participants have full power over the group, including access to shared information and resources, and management of users in the group. Clearly, there are different degrees of control alternatives over this range as illustrated in Figure 4.3.

From the operational perspective of modeling in OpenStack, we have different levels of collaboration among projects and domains: project-to-project collaboration within the same domain, project-to-project collaboration across different domains, project-to-domain collaboration and domain-to-domain collaboration, as show in Figure 4.4. Project-to-project collaboration involves sharing between several projects either in the same domain or across different domains. Only users who are assigned to roles in these projects can join the collaboration group. Project-to-domain collaboration occurs when a department needs to collaborate with in its home or an external organization. This is useful since not all collaboration scenarios need the whole organization to be involved. Project-to-Domain perspective minimizes a tenant's exposure to other tenants. In scenarios, where two organizations need large-scale collaboration or merge their resources, domain-to-domain collaboration perspective is a useful construct.

From a control perspective of modeling, there are two ways to share information and resources among participants. One approach is to host the group sharing inside one of the existing participants' administrative scope. The other approach is to host the sharing group within a third party's administrative scope, where no single participant maintains a superior control over the group. For the first approach, any participant can set up an information and resources sharing project in its domain and invite others to the sharing group. For the second approach, participants who are willing to share sensitive information can get together to set up a working project outside of any of the member participant domains. The relationship of shared space with the member participants is showed in Figure 4.5.

Based on these perspectives and the levels of administrative controls discussed above, we now provide an overview of three model alternatives using OpenStack constructs. Specifically, we use the terms project and domain as specified earlier by OSAC model.

Model 1

In model 1, one of the collaborating participants hosts a shared project where all the other collaborating participants are invited to this shared project. We call the participant who initiates the collaboration as the shared project holder, and the rest of participants the shared project members. The shared project is hosted inside the holder's domain and is isolated from the rest of the projects in the domain in order to secure the sensitive information shared by collaboration members. In this model, the shared project holder has the full power over the shared information and resource, as well as the member participants users. The sharing occurs either at a domain-to-domain level or a project-to-domain level. Figure 4.6 illustrates this approach.

When collaboration begins, the holder creates an empty shared project for information and resource sharing in its own domain. The holder invites other organizations to join the shared project as members by adding their users to the shared project. The holder decides which users can be added to the shared project and what access rights can be assigned to those users. A user who is added to the shared project is assigned with a role, which gives the user proper access right inside the shared project. Data can be brought into the shared project by member users from their

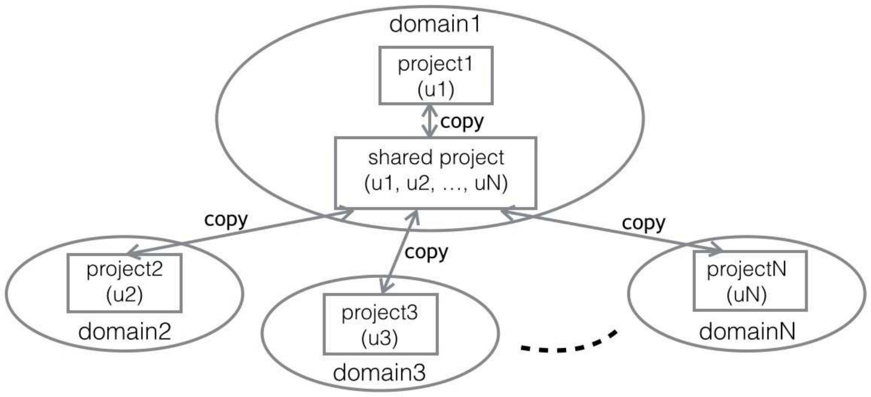


Figure 4.6: Architecture of Model 1

original projects in their home domains. Due to the ownership of the shared project, the holder can decide what data is allowed to bring in and how the information is shared.

During information and resource sharing, member users inside the shared project exchange their data, work on the shared data and finally generate new data, which may be copied back to members' original domains. Members can create, update and delete data based on their roles in the shared project. After the collaboration, the holder is responsible for disbanding the shared project. All the data which is attached to the shared project are deleted, all the processes and sessions which are executing in the shared project are killed.

Model 2

In model 2, all the collaborating participants together hosts a shared project located in an external domain. This domain does not belong to any of the members of the collaborating participants. In order to facilitate information and resource sharing among these organizations, we have the concepts of Secure Isolated Domain and Secure Isolated Project added to OpenStack platform. A sid is a special domain specifically designed for information and resource sharing, while a sip is a secure project set up for each information and/or resource sharing team. In this model, each participant of the collaboration have equal power over the shared information and resource. The sharing happens at domain-to-domain level. Thus the participants are domains of the tenants. Figure 4.7 illustrates this approach.

In this model, we enable a sid for every possible combination of organizations (tenants/domains)

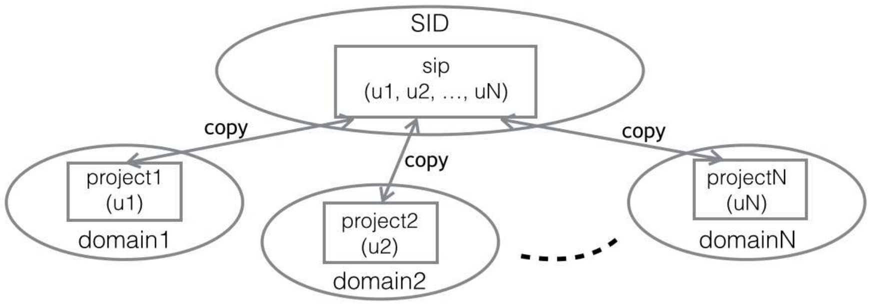


Figure 4.7: Architecture of Model 2

in the cloud. Within each such sid, there can be multiple sips. For instance, different collaborations may occur between different users in a group of organization, which leads to different sips in the same sid. Note that a sid between a set of organizations will only need to be created if a collaboration activity is ever necessary between those organizations.

When the collaboration starts, a group of organizations together create a sid and a sip in the sid. The creation process is complete only after all the members of the group agree to join to the sip. Each organization has the same access control right and priority inside the sid. Inside the sid, each participating organization has an admin user who decides which other users from her home domain can be added to a sip in the sid and with what access. A user who is added to a sip is assigned with a role at joining time. Users inside the sip can bring data into the sip from their original projects in their home domains. Users decide what data will be brought in and how the information is shared. Users inside the sip exchange their data, work on the shared data and finally generate new data, which are allowed to be copied back to their respective original domains. Users can create, update and delete data as per their roles. After the collaboration, the set of admin users are responsible for disbanding the sip. All the data which are attached to the sip are deleted, and all the processes and sessions which are running in the sip are killed.

Model 3

Model 3 is a slight variation of model 2. Similar to model 2, we still utilize sid and sip concepts to design the model. A set of organization admin users are responsible for creating, updating and deleting the sip, and this set of users become the admin users of the sip. After the sid is created, sid admin users determine which users inside of their home domain can be added to a sip in the sid or removed from the sip. The difference is that, instead of multiple sids, we design a single sid with multiple sips for each collaboration activity between organizations. The idea is that we want to hide the domain level administration of the sips which simplifies the implementation of this model.

Model 3 specially applies to a community cloud environment, where the whole cloud is a single community with multiple organizations as members. This has been previously described in paper [21], which demonstrates the model in a community cloud. In this model, we give a default sid to hold all possible sips that users can create. The default sid function is transparent to users. Every time a user issues a collaboration activity create request, a sip is created in the default sid. The system hosts the default sid permanently.

Summary

Model 1 is convenient for information and resource sharing in cases of low-assurance requirements on confidentiality of the shared data. It is easy to deploy model 1 in current OpenStack cloud platform. However, since it gives one of the collaborating organizations overwhelming power on the share project, it can create trust issues between the holder domain and member domains. Moreover, by bring in users outside of its home domain, the holder might be under the risk of exposing itself to other tenants.

Model 2 and model 3 provide all organizations that are involved in information and resources sharing an external secure space to cooperate and work together on the shared data. They avoid the disadvantages of model 1, and provide the organizations equal access control right over the shared space. Such a capability can be valuable in certain scenarios such as information sharing for cyber incident where the data is very sensitive and each participating organization would wish to have equal control on the shared space. Besides, hosting the shared project outside the organization alleviates mutual suspicions that arise in model 1. In this chapter, our model follows the idea of model 2. We give detailed design of the model for information and resources sharing in OpenStack cloud IaaS. We call it the OpenStack Access Control model with SID extension (OSAC-SID).

4.3 OSAC Model with SID Extension (OSAC-SID)

OpenStack SID model extends OSAC model to include Secure Isolated Domain (SID) functionality. We call it OSAC-SID model. We build OSAC-SID model on top of OSAC model. We will present the OSAC-SID model in a way which covers only the additional components compared to OSAC model.

In our discussion, we assume that in an OpenStack cloud, each organization has one and only one domain. In previous chapter, we assumed that a user belongs to one organization in the community. It is consistent with the user home-domain concept in OpenStack. The concept of home-domain requires that a user can only belong to one domain in OpenStack. OpenStack allows a user to be assigned to projects across domains and access those projects separately using appropriate tokens. Given two storage options in OpenStack, here we constrain the storage to object storage only, which is provided by the Swift service. For simplicity we ignore the group mechanism in OpenStack, since it is essentially a convenience to group together a set of users in a domain and can be easily incorporated in a more complete description.

OpenStack has basic resource containers projects and administrative boundary domains, which are respectivley corresponding to projects and secure isolated domains in SID-model. Thus, we use OpenStack projects to realize core project, open project and secure isolated projects in SID-model, and OpenStack domains to realize secure isolated domains in SID-model.

In chapter 3, we have introduced the important concepts and components in SID-model. In the formalization of OSAC-SID model in this chapter, we mainly introduce how SID-model fit into OpenStack platform.

4.3.1 Components

Figure 4.8 shows OSAC-SID model. The additional entity components included in OSAC-SID model are: Secure Isolated Domain (SID), Expert Users (EU), Core Project (CP), Secure Isolated Project (SIP), and Open Project (OP). We require two roles, so $\{admin, member\} \subseteq$ R.

Secure Isolated Domains provides a secure isolated environment for cyber security collabo-

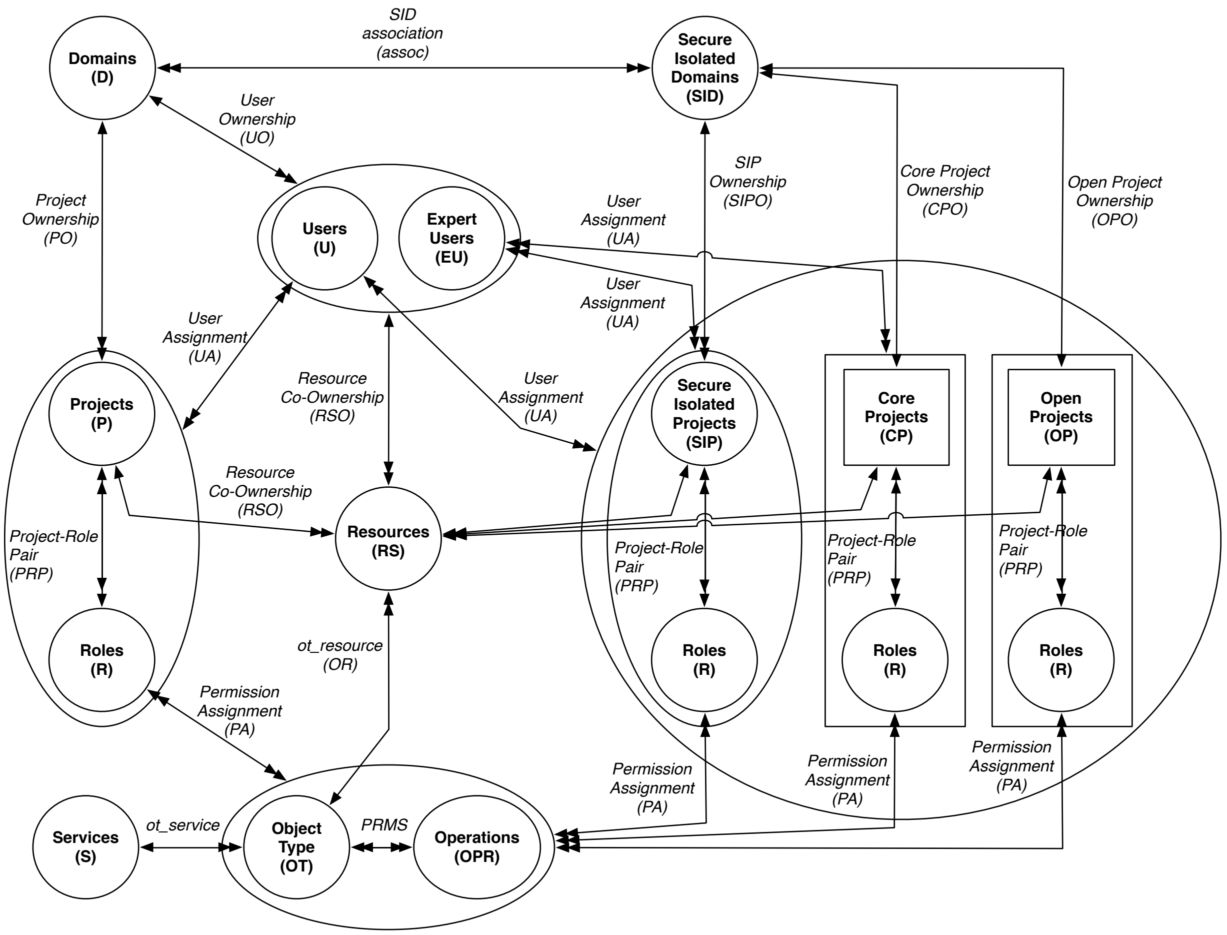


Figure 4.8: OpenStack Access Control Model with SID Extension (OSAC-SID)

rations among organizations. In context of OpenStack, a secure isolated domain is an OpenStack domain with constrains fitting SID requirements. It does not belong to any of the organizations in the community or public cloud. OpenStack project component is used for **Core Projects, Open Projects and Secure Isolated Projects** entities, by adding constrains to fit into SID requirements. **Expert Users** brings the model outside-community professionals and expertise. Expert users don't belong to the community. **Resources** refers to cloud assets which can be owned by users. Cloud assets are cloud resources such as virtual machines, databases, storages, etc. A user and a project which the user is assigned to have the co-ownership over cloud resources inside the project.

In the following, we formalize of concepts introduced above, as well as relations among them.

Definition 2. OSAC-SID model has the following components in addition to OSAC.

- SID, SIP, CP, OP and EU are finite sets of Security Isolated Domain, Secure Isolated Projects, Core Projects, Open Projects and Expert Users. Each sid is a special domain. Each sid owns its own Expert Users, Core Project, Open Project, and Secure Isolated Projects, correspondingly represented by Expert User Ownership (EOU), Core Project Ownership (CPO), Open Project Ownership (OPO) and Secure Isolated Project Ownership (SIPO). Core Project and Open Project are two permanent projects in a sid.

- Virtual Machines (VM) is object type for compute service in OpenStack cloud platform.

- Storage Container (SC) and Storage Container Object (CO) are two object types for storage service in OpenStack cloud platform.

- Core Project Ownership (CPO) : is a function CPO : CP $\rightarrow$ SID, that maps a single core project to its owning sid. Equivalently viewed as a one-to-one relation CPO $\subseteq$ CP $\times$ SID.

- Open Project Ownership (OPO) : is a function OPO : OP $\rightarrow$ SID, that maps a single open project to its owning sid. Equivalently viewed as a one-to-one relation OPO $\subseteq$ OP $\times$ SID.

- Secure Isolated Project Ownership (SIPO) : is a function SIPO : SIP $\rightarrow$ SID, mapping Secure Isolated Projects to its owning sid. Equivalently viewed as a many-to-one relation SPO $\subseteq$ SIP $\times$ SID.

- SID association (assoc): is a function assoc : SID $\rightarrow 2^D$, mapping a SID to all its member domains/organizations.

- Resource Co-Ownership (RO) : is a function RO : RS $\rightarrow$ ((SIP $\cup$ CP $\cup$ OP), (U $\cup$ EU)), mapping resources to its owning project and user. Equivalently viewed as a many-to-one relation RSO $\subseteq$ RS $\times$ ((SIP $\cup$ CP $\cup$ OP) $\times$ (U $\cup$ EU)).

4.3.2 Administrative Model

The administrative aspects of OSAC-SID are discussed informally below. A formal specification is given in Table 4.2 and Table **??**.

Create a sid: The creation of sid is based on agreement among the community of organiza-

Table 4.1: Administrative Model

Operation	Authorization Requirement	Update
SidCreate(adminu, uSet, sid) */* An admin user representing uSet creates a sid */*	adminu $\in$ *uSet* $\wedge$ adminu $\in$ U $\wedge$ sid $\notin$ SID	SID$'$ = SID $\cup$ {sid}; assoc(sid) = $\bigcup_{adminu \in uSet}$ UO(adminu); CP$'$ = CP $\cup$ {cp}; CPO(cp) = sid; OP$'$ = OP $\cup$ {op}; OPO(op) = sid; UA$'$ = UA $\cup$ (uSet, cp, *admin*) $\cup$ (uSet, op, *admin*); PRP$'$ = PRP $\cup$ {(cp, *admin*), (op, *admin*)}; D$'$ = D $\cup$ {sid}; P$'$ = P $\cup$ {cp, op}.
SidDelete(adminu, uSet, sid) */* An admin user representing uSet deletes the sid*/*	adminu $\in$ *uSet* $\wedge$ adminu $\in$ U $\wedge$ (adminu, *admin*) $\in$ UA $\wedge$ assoc(sid) = $\bigcup_{adminu \in uSet}$ UO(adminu) $\wedge$ sid $\in$ SID	SID$'$ = SID - {sid}; assoc(sid) = NULL; CP$'$ = CP - {cp}; CPO(cp) = NULL; OP$'$ = OP - {op}; OPO(op) = NULL; UA$'$ = UA - (uSet, cp, *admin*) - (uSet, op, *admin*); PRP$'$ = PRP - {(cp, *admin*), (op, *admin*)}; D$'$ = D - {sid}; P$'$ = P - {cp, op}; if $\exists$ u $\in$ (U $\cup$ EU).((u, *member*) $\in$ UA), then UA$'$ = UA - (u, *member*); if $\exists$ sip $\in$ SIP.(SIPO(sip) = sid), then SIP$'$ = SIP - sip $\wedge$ PRP$'$ = PRP - {(sip, *admin*), (sip, *member*)}.
SipCreate(adminu, sip, sid) */* An admin user representing uSet creates a sip */*	adminu $\in$ *uSet* $\wedge$ adminu $\in$ U $\wedge$ (adminu, *admin*) $\in$ UA $\wedge$ UO(adminu) $\in$ assoc(sid) $\wedge$ sip $\notin$ SIP	SIP$'$ = SIP $\cup$ {sip}; SIPO(sip) = sid; PRP$'$ = PRP $\cup$ {(sip, *admin*)}; UA$'$ = UA $\cup$ (adminu, sip, *admin*); P$'$ = P $\cup$ {sip}.
SipDelete(adminu, sip, sid) */* An admin user representing uSet deletes a sip*/*	adminu $\in$ *uSet* $\wedge$ adminu $\in$ U $\wedge$ (adminu, *admin*) $\in$ UA $\wedge$ UO(adminu) $\in$ assoc(sid) $\wedge$ SIPO(sip) = sid	SIP$'$ = SIP - {sip}; SIPO(sip) = NULL; PRP$'$ = PRP - {(sip, *admin*)}; UA$'$ = UA - (adminu, sip, *admin*); P$'$ = P - {sip}.

Table 4.2: SID Administrative Model (continued)

Operation	Authorization Requirement	Update
UserAdd(adminu, u, p, sid) */* Admin users add a user from his home domain to a cp, op or sip */*	adminu $\in$ U $\wedge$ (adminu, p, *admin*) $\in$ UA $\wedge$ (p, *admin*) $\in$ PRP$\wedge$ u $\in$ U $\wedge$ UO(u) = UO(adminu) $\wedge$ p $\in$ (CP $\cup$ OP $\cup$ SIP) $\wedge$ (CPO(p) = sid $\vee$ OPO(p) = sid $\vee$ SIP(p) = sid)	PRP$'$ = PRP $\cup$ {(p, *member*)}; UA$'$ = UA $\cup$ {(u, p, *member*)}.
UserRemove(adminu, u, p, sid) */* Admin users remove a user from a cp, op or sip */*	adminu $\in$ U $\wedge$ (adminu, p, *admin*) $\in$ UA $\wedge$ (p, *admin*) $\in$ PRP$\wedge$ u $\in$ U $\wedge$ UO(u) = UO(adminu) $\wedge$ p $\in$ (CP $\cup$ OP $\cup$ SIP) $\wedge$ (CPO(p) = sid $\vee$ OPO(p) = sid $\vee$ SIP(p) = sid) $\wedge$ (u, p, *member*) $\in$ UA $\wedge$ (p, *member*) $\in$ PRP	PRP$'$ = PRP - {(p, *member*)}; UA$'$ = UA - {(u, p, *member*)}.
EUserAdd(adminu, eu, p, sid) */* Admin users add an expert user to a cp or sip */*	adminu $\in$ U $\wedge$ (adminu, p, *admin*) $\in$ UA $\wedge$ (p, *admin*) $\in$ PRP$\wedge$ eu $\in$ EU $\wedge$ p $\in$ (CP $\cup$ SIP) $\wedge$ (CPO(p) = sid $\vee$ SIPO(p) = sid)	PRP$'$ = PRP $\cup$ {(p, *member*)}; UA$'$ = UA $\cup$ {(eu, p, *member*)}.
EUserRemove(adminu, eu, p, sid) */* Admin users remove an expert user from a cp or sip */*	adminu $\in$ U $\wedge$ (adminu, p, *admin*) $\in$ UA $\wedge$ (p, *admin*) $\in$ PRP$\wedge$ eu $\in$ EU $\wedge$ p $\in$ (CP $\cup$ SIP) $\wedge$ (CPO(p) = sid $\vee$ SIPO(p) = sid) $\wedge$ (eu, p, *member*) $\in$ UA $\wedge$ (p, *member*) $\in$ PRP	PRP$'$ = PRP - {(p, *member*)}; UA$'$ = UA - {(eu, p, *member*)}.

tions. One security admin from an organization representing the group of organizations request the creation of a sid with parameters including all the security admin users, each representing one organization in the community. *uSet* denotes a fixed group of security admin users from all organizations of the community, with one admin user representing each organization. When a sid is created, the security admin user who issues sid creation command along with all other security admin users from *uSet* will become the limited administrative users of the sid, in which each organization in the community has equal limited administrative power which is carried through *uSet*. Only users from *uSet* later can dynamically create sips in the sid. Like the normal domain in OpenStack, a sid has domain admin assigned with it, which are users from *uSet*.

Since in OpenStack, secure isolated domains are realized by OpenStack domains, *uSet* will be assigned with domain admin roles on the special domain sid. Each project in the sid are realized by OpenStack projects, *uSet* will be assigned with project admin roles to the core project and open project. With project admin role, core project admin users can add and remove other users from their home domain to the core project. The open project is open for all the users from the community of organizations. Admin users can add normal users to open project with OpenStack member role as a member user.

Delete a sid: One admin user from *uSet* representing the group of organizations initilizes the sid delete request. When the sid domain is deleted, the core project, open project and all the sips inside the sid (if any) will be securely deleted. All the assigned resources and users will be securely revoked from the sid. All the role assignment to the sid domain and projects inside the sid will be removed.

Create a sip: A sip is created for cyber collaborations among a set of organizations in a community. Any security admin user in *uSet* which represents the set of the community of organizations can create a sip. Each organization security admin (*uSet*) get equal limited administrative power in the sip, by being assigned with project admin role, which in OpenStack is *admin* role. The role gives the sip admin users the permission to add and remove other users from their home account to the sip.

Delete a sip: A sip can be securely deleted by any security admin user from *uSet*. Role assignments created for users to join the sip are deleted. All information and resources are securely deleted in the sip.

Add/remove a user to/from a core project: Core project admin users are the set of security admin users (*uSet*) from the community of organizations. These limited admin users can add/remove users of their organizations to/from core project with assigning normal users *member* role. All the users added to the core project are existing users from an organization's domain. When users are removed from the core project, they will lose the access to corresponding information and resources in the core project, regardless of the ownership of the piece of information in the past.

Add/remove a user to/from a sip: Users from *uSet* have limited administrative power in the sip, since they are assigned with *admin* role to the sip. They can add/remove users of their home domains to/from the corresponding sip as appropriate for the collaboration with assigning normal users *member* role.

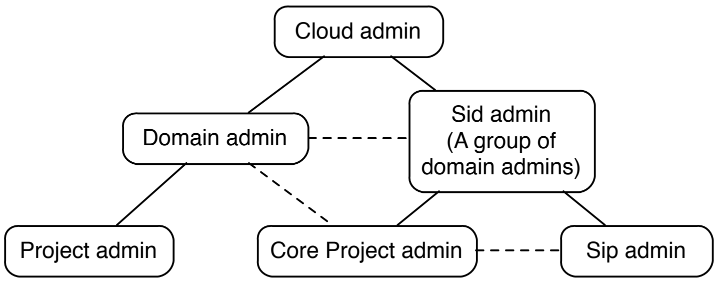


Figure 4.9: Administration Relation

Add/remove a user to an open project: All user in the collaborative community of organizations are allowed to join the open project. Users in open project have equal but limited permissions. They can share cyber data, but have no control over other users. OpenStack role *member* represents this limited permission. Sid admin users add/remove normal users from their organizations to/from open project. Users will not be able to access and share any data once they leave the open project.

Add/remove an expert user to/from a cp/sip: Expert users can be added/remove to/from core projects and sips as a member. Admin users from *uSet* can request to add/remove expert users to/from the core project or a sip.

Additional administration details

Here we give additional explanation of OSAC-SID model from administration perspective, as shown in Figure 4.9. Cloud admin is the super admin user of the cloud who can create domains, users and assign users as admins for domains. Some administrative operations in sid are done by cloud admin, such as creating/deleting/updating a sid domain, though the request is initiated by an organization domain admin user.

A domain admin is the super admin user for an organization. A domain admin can create/delete/update a project and user/group in the domain. Projects can also have admin users assigned to them. The difference is that project admin user cannot create/delete/update users and groups, but they can assign users/groups to the project.

A core project is designed for core group [12], which is a cyber security committee for the

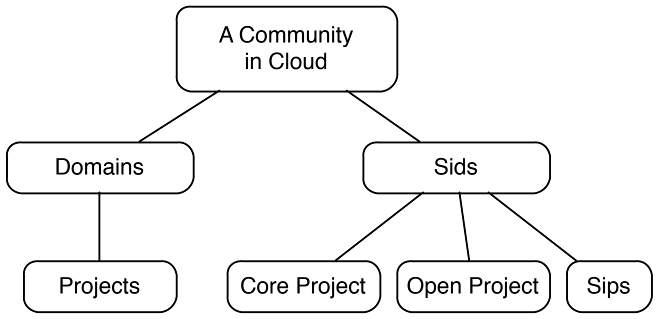


Figure 4.10: Resources Ownership

whole community. Domain admin decides which of the organization's users will be in the cyber security committee. Domain admins are automatically assigned as admin users in core project when a sid domain is created. As core project admin users, they can further add users from their home domains to a core project, create sips and add users to sips.

The administration over a sip is similar to the one in paper [18]. The same set of core project admin users create/delete/manage a sip. Each user in this set has equal admin power over the sip. They can assign users from their organizations to the sip. They can bring in cyber information from their home domains.

Resource ownership

From the perspective of resource ownership, we give a view of the model, as shown in Figure 4.10. Organizations own their resources manifested as domains in the community cloud. An organization has multiple normal projects, some of which could be for security purposes. Inside a domain, resources are divided by projects which for example, represent different departments inside an organization.

A sid securely isolates cloud resources from organization domains for cyber security purpose. All sids are owned by the cloud. A core project belongs to a sid and provides a stable and controlled place for organizations to exchange and share routine cyber security information. An open project belongs to a sid and provides an isolated place for normal users to share security data. A sid holds all sips which are designed for specific cyber security purposes.

Table 4.3: OSAC-SID Operational Model

Operation	Authorization Requirement	Update
CreateVM(vm, p, u) *`/*` A user creates a vm `*/`*	$vm \notin RS \wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge \exists$ (perms, r) $\in$ PA.(perms = (vm, create) $\wedge$ (p, r) $\in$ PRP $\wedge$ (u, p, r) $\in$ UA)	RS' = RS $\cup$ {vm}; RSO' = RSO $\cup$ {(vm, (p, u))}; OR(vm) = VM.
DeleteVM(vm, p, u) *`/*` A user deletes a vm `*/`*	$vm \in RS \wedge$ RSO(vm) = {(p, u)} $\wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge \exists$ (perms, r) $\in$ PA.(perms = (vm, delete) $\wedge$ (p, r) $\in$ PRP $\wedge$ (u, p, r) $\in$ UA	RS' = RS - {vm}; RSO' = RSO - {(vm, (p, u))}; vm = NULL.
CreateSContainer(sc, p, u) *`/*` A user creates a storage container `*/`*	$sc \notin RS \wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge \exists$ (perms, r) $\in$ PA.(perms = (sc, create) $\wedge$ (p, r) $\in$ PRP $\wedge$ (u, p, r) $\in$ UA)	RS' = RS $\cup$ {sc}; RSO' = RSO $\cup$ {(sc, (p, u))}; OR(sc) = SC.
DeleteSContainer(sc, p, u) *`/*` A user deletes a storage container `*/`*	$sc \in RS \wedge$ RSO(sc) = {(p, u)} $\wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge \exists$ (perms, r) $\in$ PA.(perms = (sc, delete) $\wedge$ (p, r) $\in$ PRP $\wedge$ (u, p, r) $\in$ UA	RS' = RS - {sc}; RSO' = RSO - {(sc, (p, u))}; sc = NULL.
CreateObject(co, sc, p, u) *`/*` A user creates a storage container object `*/`*	$co \notin RS \wedge sc \in RS \wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge$ RSO(sc) = (p, u) $\wedge \exists$ (perms, r) $\in$ PA.(perms = (co, create) $\wedge$ (p, r) $\in$ PRP $\wedge$ (u, p, r) $\in$ UA)	RS' = RS $\cup$ {co}; RSO' = RSO $\cup$ {(co, (p, u))}; OR(co) = CO.
DeleteObject(co, sc, p, u) *`/*` A user delete a storage container object `*/`*	$co \in RS \wedge$ RSO(co) = {(p, u)} $\wedge sc \in RS \wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge$ RSO(sc) = (p, u) $\wedge \exists$ (perms, r) $\in$ PA.(perms = (co, create) $\wedge$ (p, r) $\in$ PRP $\wedge$ (u, p, r) $\in$ UA)	RS' = RS - {co}; RSO' = RSO - {(co, (p, u))}; co = NULL.

4.3.3 Operational Model

In the operational model, we mainly show how and what operations a normal user can issue in the model. For simplicity, we only demonstrate the core operations on virtual machines and storage containers, including creation and delete. *Create* method allows users to create a new instance of virtual machine or a storage container in a core project, open project or a sip. *Delete* method allows users to delete an existing instance of a virtual machine or storage container in a project. For objects, we can upload objects and download objects from a storage container. In Table 4.3, we give the details of operational model.

After a user is assigned to the core project, open project or a sip, the user can issue following operations:

CreateVM/DeleteVM: A user can create/delete a virtual machines in the core project, open project or a sip, to which the user is assigned with *member* role.

Createcontainer/Deletecontainer: A user can create/delete a storage container in the core project, open project or a sip, to which the user is assigned with *member* role. A storage container holds Swift container objects.

CreateObject/DeleteObject: A user can create/delete a Swift object in a storage container in the core project, open project or a sip, to which the user is assigned with *member* role.

4.4 Enforcement

We discuss the enforcement of OSAC-SID model on OpenStack Kilo release. In OpenStack, there are three levels of administrative roles: $cloud\_admin$, $domain\_admin$, and $project\_admin$, which have administrative power respectively over the whole cloud, a domain and a project. In OpenStack, domains forms the administrative level of resources boundary. Inside a domain, resources can be further divided by projects, while project is the basic resources container in the cloud. Admin users can create users and assign global roles to a user with a project. Users permissions over services and resources are defined in policy files stored in different services.

We enforced the SID-service as part of OpenStack public cloud platform services, which is quite different with model enforcement in AWS and Azure in the following chapters, where SID-service is provided by a third party outside of the community. In order to deploy the model in OpenStack platform, we modified Keystone server to include functionalities which has specific features that facilitate information and resources sharing using SID-model. Recall that Keystone facilitates authentication and authorization in OpenStack.

SID-service consists of a collection of functionalities added to OpenStack cloud platform, which realize the SID-model. Users can initialize a sid command just like issuing any other OpenStack command. A core project and an open project are created along with sid creation. In OpenStack, only cloud admin is allowed to create a domain. A sid is a special domain, thus, to create a sid needs cloud admin permissions. Cloud admin create a sid with a core project and an open project in it. Cloud admin assigns organization security admin users as admin users for the sid, as well as the core project. For open project, sid admin users have the permissions to assign users from their organizations to it as a member role. All such cloud admin functions can be automated by providing scripts that do these activities on the cloud admin's behalf after verifying appropriate authorization.

4.4.1 Functionalities

Keystone server consists of several modules, such as auth, identity, assignment, resources, catalog and etc. Each of those modules follows a basic protocol to handle request from clients, as shown in Figure 4.11. Request from clients are in http format. With a request coming from a client, the router maps the http request to functions in controller. In another word, the router decides which controller method should match with the client request. Then the controller talks to the core and find the corresponding methods in core to handle the request. The core processes the request and talks to the drive. The drive writes to tables in database. The result is returned to the client in a sequence of drive-core-controller-router-client.

Resource Module

Domains and projects related functions are included in resource module. One can create, delete, update and list domains and projects through resource module. A sid is a special domain and a sip is a special project. We add sid and sip functions in resource module. Sid and sip functions follows the basic protocol of Keystone server module. Take sid creation for example. A user sends a sid creation http request to Keystone server, the router maps the http request to a function in controller, the controller further call methods in core, the core finally talks to the drive and the drive writes the data into database tables and returns the result, as show in Figure 4.12.

In resource controller, we add two classes of functions, one for sid and the other for sip. These functions include creation, update, list and delete. Here, we give more details about creation functions. Figure 4.12 shows the flow of a sid and the sip creation functions in resource controller.

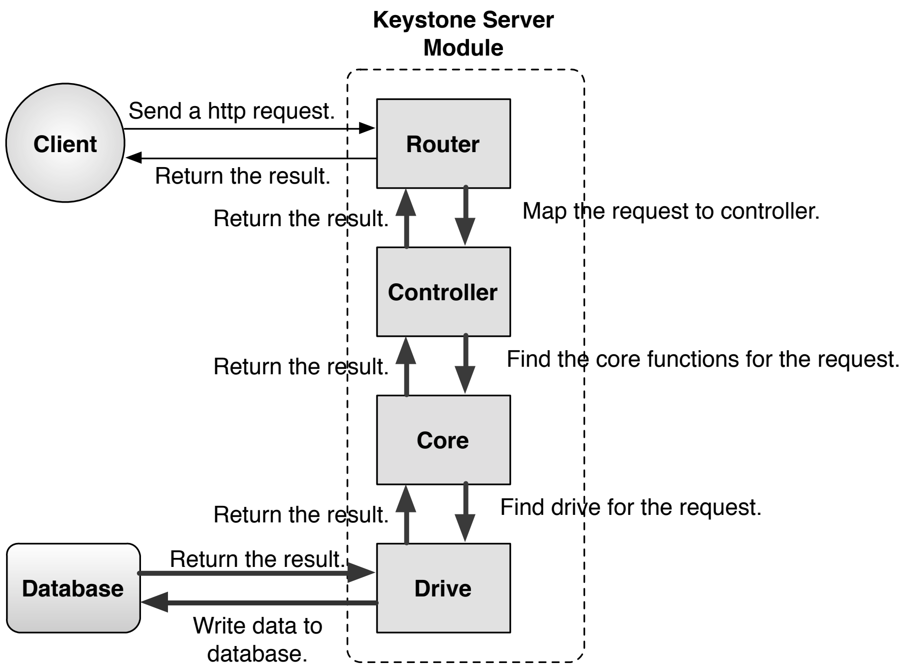
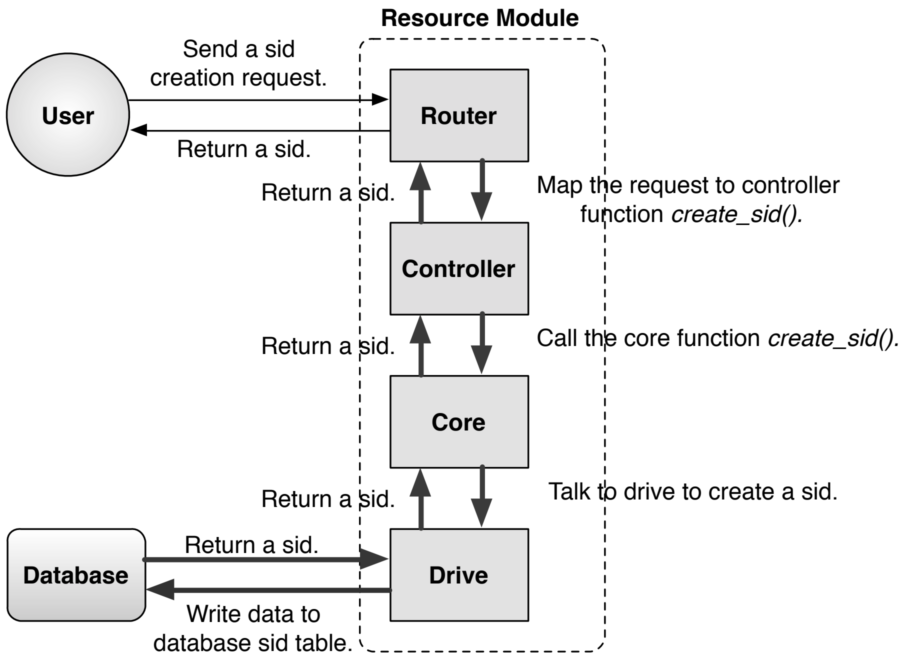


Figure 4.11: OpenStack Keystone Module Protocol

Again, a sid is a real domain, thus, a domain is created as a sid in a sid creation function. A core project and an open project are two projects created in sid domain. All the member organizations security users are assigned to as sid domain admin users on the new sid, as well as project admin users on the core project. Open project assignment is left for later. Sid admin users can assign a normal user to core project, open project and a sip at any time. The last step is to add a sid record into sid table in database, where saves all the sids association information. The sid information table allows distinguishing a normal domain from a sid domain. Sip creation is much like a project creation in controller module, since it is constrained by sid domain boundary already. On the other side, in normal project creation function, we add a sid checking process, which determines whether the creating project belongs to a sid or a normal domain. A sip project cannot be created by a normal project creation function.



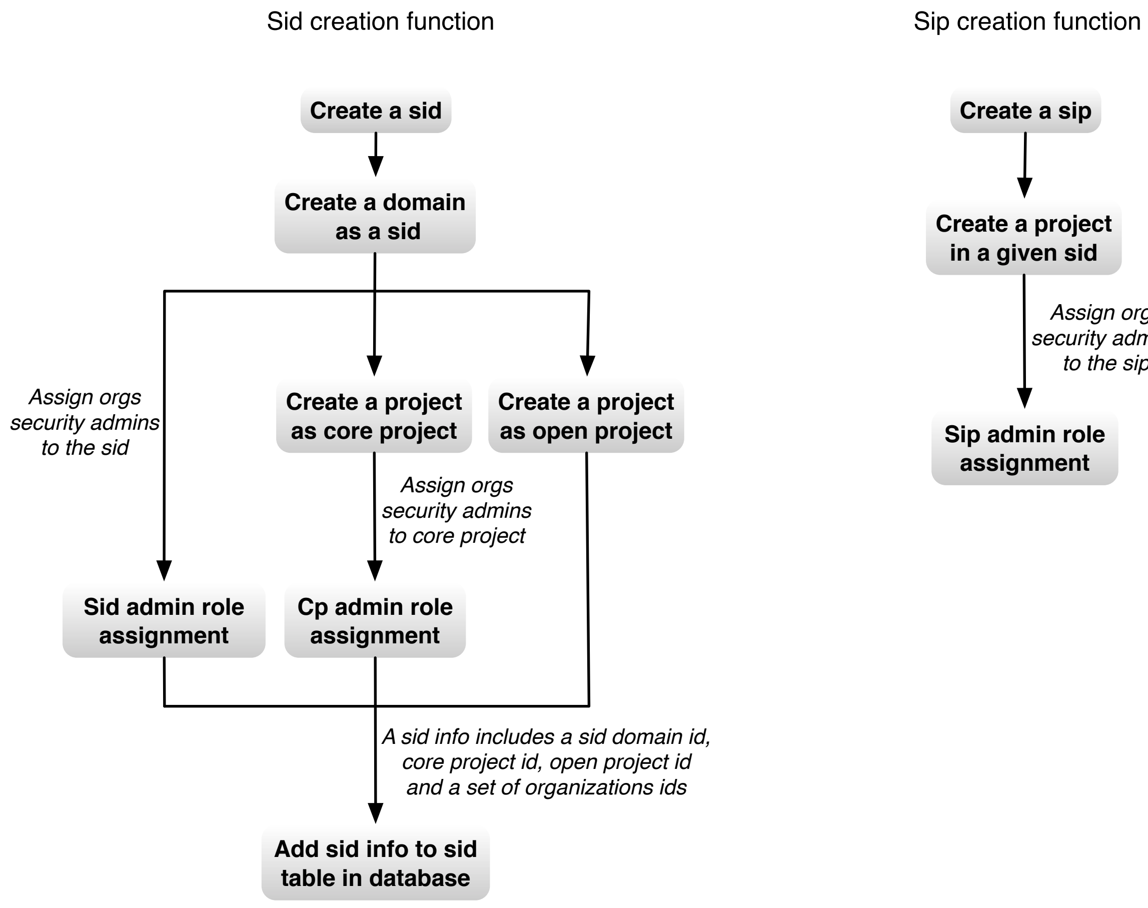
Resource Module
Send a sid creation request.
User
Router
Return a sid.
Return a sid.
Map the request to controller function *create_sid().*
Controller
Return a sid.
Call the core function *create_sid().*
Core
Return a sid.
Talk to drive to create a sid.
Database
Return a sid.
Drive
Write data to database sid table.

Figure 4.12: Process of a Sid Creation

Assignment Module

Assignment module is part of the Keystone server which in charge of user role assignments. To add sid and sip functionality to Keystone, we also need to modify the assignment module. Same as resource module, assignment follows the general Keystone module protocol. We add sid and sip role assignments in the assignment module. Same as in resource module, the request is handled by router, controller, core and drive. Same as assigning a role a domain or a project, a user with proper permission can assign a role to a sid or a project in the sid, including core project, open project and sips. For the model, we constrain only the cloud user has the privilege to assign a domain admin role to a sid and a sid admin user has the privilege to assign a role to a normal user. The flow of a role assignment process is similar to the process of a sid creation as described in proceeding paragraph.

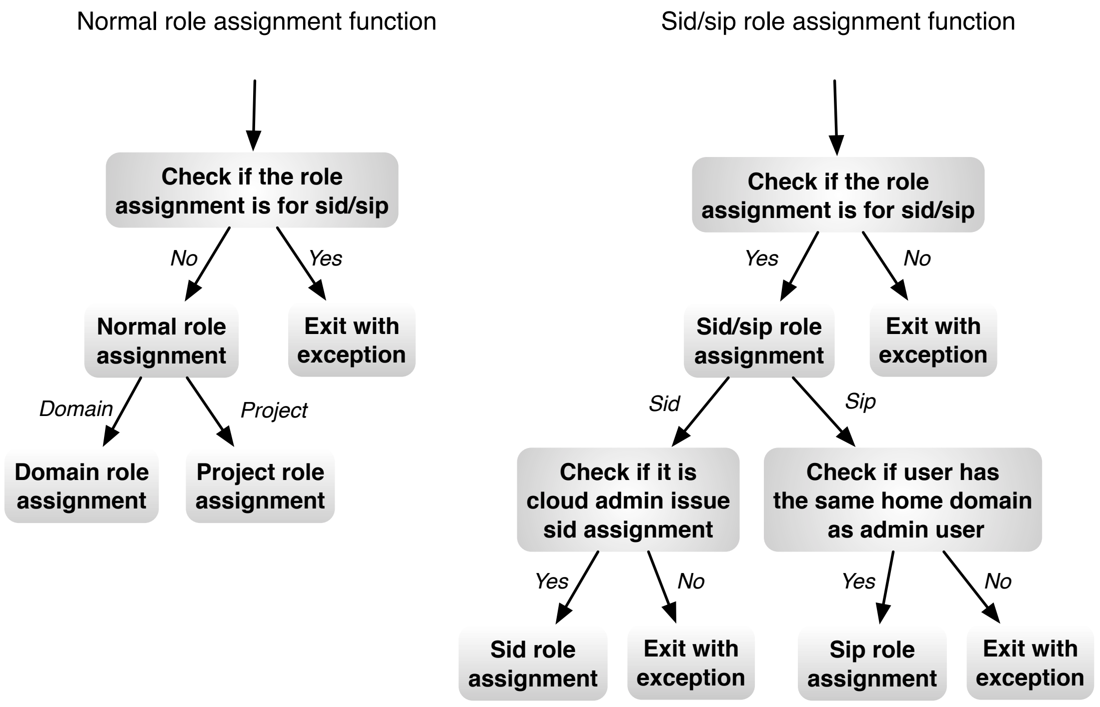
During information and resource sharing activity, if a new user requests to join a sip, the user need to be verified as to whether the user comes from domains which are associated with the sid where the sip is located. Users that do not belong to the set of associated domains will be rejected



Sid creation function
Sip creation function
Create a sid
Create a domain as a sid
Assign orgs security admins to the sid
Create a project as core project
Create a project as open project
Assign orgs security admins to core project
Sid admin role assignment
Cp admin role assignment
A sid info includes a sid domain id, core project id, open project id and a set of organizations ids
Add sid info to sid table in database
Create a sip
Create a project in a given sid
Assign orgs security admins to the sip
Sip admin role assignment

Figure 4.13: Flow of a Sid or Sip Creation

by the server. The crucial part of sid role assignment is to distinguish this from assigning a role to a normal domain and to a sid, and from a normal project to a sip. Since a sid is a real domain and a sip is a real project, assignments to a domain and a project would be applied to a sid and a sip too, which we do not want to happen. Thus, before an assignment is issued, we need to be able to determine whether it is a normal domain or a sid, a normal project or a sip. We add a pre-check process before a role assignment and a sid/sip role assignment in the module, as shown in Figure 4.14. In the controller of the module, there are two functions for assigning roles, one for normal domains and projects, the other for sids and sips. Both functions point to the same core method which issue role assignment in Keystone server. We add the role-assignment pre-check in assignment controller. The pre-check process guarantee that: 1) a domain admin cannot assign roles through sid/sip role assignment function, since a sid is a real domain and a sip is a real project;



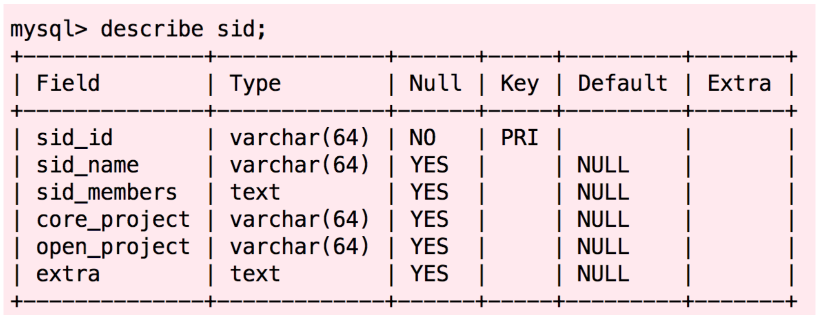
Normal role assignment function
Check if the role assignment is for sid/sip
No
Yes
Normal role assignment
Exit with exception
Domain
Project
Domain role assignment
Project role assignment
Sid/sip role assignment function
Check if the role assignment is for sid/sip
Yes
No
Sid/sip role assignment
Exit with exception
Sid
Sip
Check if it is cloud admin issue sid assignment
Check if user has the same home domain as admin user
Yes
No
Yes
No
Sid role assignment
Exit with exception
Sip role assignment
Exit with exception

Figure 4.14: Pre-Check of Role Assignment

2) a sid admin cannot assign roles through normal role assignment function, for the same reason; 3) only cloud admin is allowed to assign sid admin to a sid; and 4) sid admins can only assign users coming from their own home domain, but they cannot assign arbitrary users to projects in the sid while domain admins can assign any outside users to projects in their home domain.

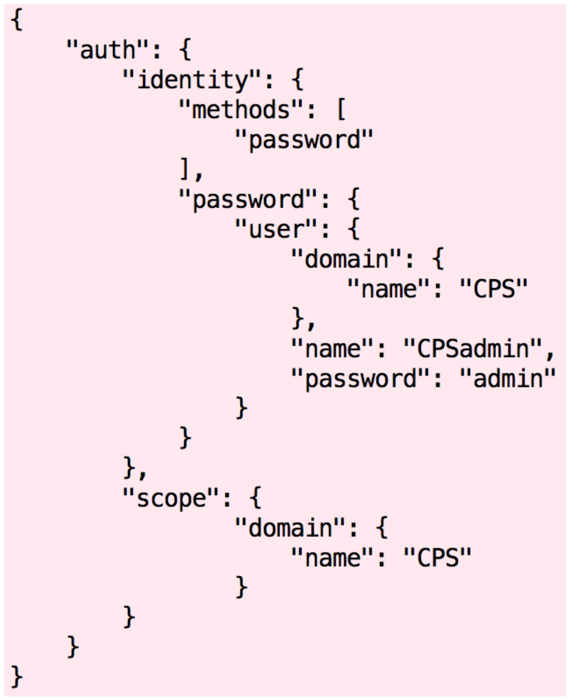
Database

OpenStack uses MySQL database as storage of all meta data. Keystone has one database to store all meta data. To store meta data for sid/sip functionality, we need to add one table “sid" to hold meta data for all sid initiations, which includes sid domain id, all associated organizations names and domain ids, core project id and open project id. Every time a sid creation request is issued, the “sid” table stores the above information as a record in the table. The record will be removed when the sid is deleted. The “sid” table will be queried during all user and sip verification and sid member verification processes. Figure 4.15 shows table “sid” construct in keystone database.



```
mysql> describe sid;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| sid_id       | varchar(64) | NO   | PRI |         |       |
| sid_name     | varchar(64) | YES  |     | NULL    |       |
| sid_members  | text        | YES  |     | NULL    |       |
| core_project | varchar(64) | YES  |     | NULL    |       |
| open_project | varchar(64) | YES  |     | NULL    |       |
| extra        | text        | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
```

Figure 4.15: Sid Table



```
{
    "auth": {
        "identity": {
            "methods": [
                "password"
            ],
            "password": {
                "user": {
                    "domain": {
                        "name": "CPS"
                    },
                    "name": "CPSadmin",
                    "password": "admin"
                }
            }
        },
        "scope": {
                "domain": {
                    "name": "CPS"
                }
        }
    }
}
```

Figure 4.16: A Sample JSON File for Generating a Token

4.4.2 Credentials and Policy

Authentication and Authorization with Tokens

In order to access to resources in OpenStack cloud, a user need to have a token. A token can be unscoped or scoped to a domain or a project. In SID-model, all sid admin users's tokens are scoped to the sid level which is domain level, while all users who have access to core project, open project and sips are scoped to project level token. Domain scoped tokens allow permissions to create/delete/update projects and users in the domain. Project scoped tokens give users permission

```
"identity:get_domain": "rule:cloud_admin",
"identity:list_domains": "rule:cloud_admin",
"identity:create_domain": "rule:cloud_admin",
"identity:update_domain": "rule:cloud_admin",
"identity:delete_domain": "rule:cloud_admin",
```

Figure 4.17: A Sample V3 Policy for Creating or Deleting a Sid

to access cloud resources though certain projects. Figure 4.16 shows an example of Json file which can be used to get a token. This file shows that a domain scoped token can be generated, in which user *CPSadmin* has a home domain *CPS* and a password *admin*, and is scoped to domain CPS. This file is used to get a domain admin role for user *CPSadmin*.

Similar to user-project-role assignment, a user is assigned to a sip with a sip-role pair. After a user is assigned to a sip, the user has access to the resources in the sip. That is, the user can get a sip-scoped token from Keystone and use the token to access the sip. After sid/sip attributes are added to token, Keystone can issue a token which can scoped to a sip in a sid. Users can use this scoped token to access a sip. Cloud services authorize users according to the user credential information stored in a scoped token.

Policy

New methods of SID-service have to be added to policy files in OpenStack Keystone server. In policy file, each method has corresponding rules defined. We need to add all new methods we define in SID-service to policy files in order to make those methods work under proper access control rules. By default, policy version v2 is used. In order to use domain admin role, we use policy v3 instead of v2. More fine-grained access control is defined in policy v3. Figure 4.17 shows a piece of v3 policy, which shows that only cloud admin user is allowed to get, list, create, update and delete a domain.

4.4.3 Demonstrations

In this part, we focus on user's perspective on the model enforcement. From organizations's perspective, their security admin users can start a new group collaboration among a set of organizations in the community, by initiating a sid creation request to OpenStack Keystone server. After the sid setting up, proper roles are assigned to associated member organizations security admin users, who will be representing organizations being in charge of during all collaborations happening in the sid. Those admin users manages the sid together. They have equal permission over core project and any new created sip projects. From organization users' perspective, they are able to join the sid for cyber collaborations once the security admin users give them permissions. They are able to share information and resources in a project in the sid domain with proper roles.

Create a Sid

As an organization security admin user, one can initiate a sid request and create a sid for the group of organizations in the community. From an organization security admin user's perspective, there is a sequence of actions that happens behind the scenes when the sid creation request is issued. The security admin user issues a sid creation request command, which communicates to Keystone server that an organization security admin user wishes to create a sid for a group of organizations in the community. Keystone server accepts the request, creates a domain as a sid domain, a project as core project and another project as open project. Then Keystone server assigns proper roles to the sid domain and core project and create a sid info record. Now a sid is officially created and Keystone server return the sid to the user who issued the request. Figure 4.18 shows the whole process.

After the process of creating a sid is complete, all the organizations security admin users get sid domain admin roles. They can get scoped tokens over the sid domain when they visit the sid domain. With the domain scoped token, admin users can create projects, which will be sips in this situation, and bring normal users to the created projects. Mean time, they also get core project admin roles. They can get project scoped token to access core project. With the project scoped token, admin users can bring normal users to the core project.

When an organization security admin user initiates a sid request and creates a sid for the group of organizations in the community, the admin user sends the request with information including

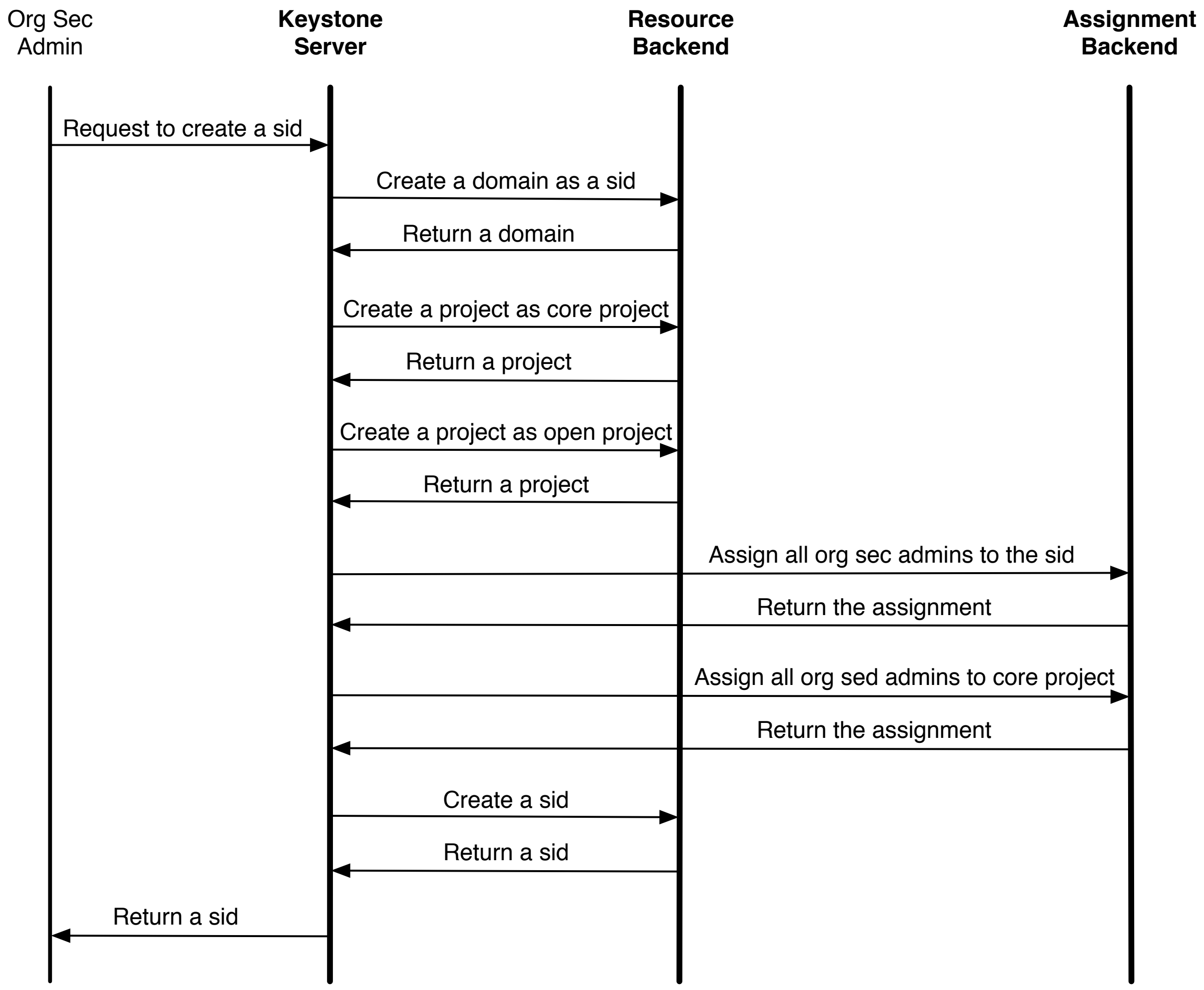


Figure 4.18: Create a Sid

the sid name, sid member organizations names and ids, and sid member organizations security admin users names and ids. Also, the request needs to be sent to Keystone server with proper tokens. Figure 4.19 shows an example of http request during this procedure. A proper token $ADMIN_TOKEN is used for authentication and authorization. Parameter data contains the new created sid name “name”, member organizations information “sid_members”, as well as organizations security admin users information “sid_member_admins”.

```
# create a sid:
curl -i 'http://10.245.121.77:5000/v3/sids' -X POST -H "X-Auth-Token: $ADMIN_TOKE
N" -H "Content-Type: application/json" -H "Accept: application/json" -d '{"sid":
{"enabled": true, "name": "sid1", "sid_members": {"$org1": "$org1_id", "$org2": "
$org2_id", "$org3": "$org3_id"}, "sid_member_admins":{"$org1": "$org1_admin_id",
"$org2": "$org2_admin_id", "$org3": "$org3_admin_id"}}}'
```

Figure 4.19: HTTP Request - Create a Sid

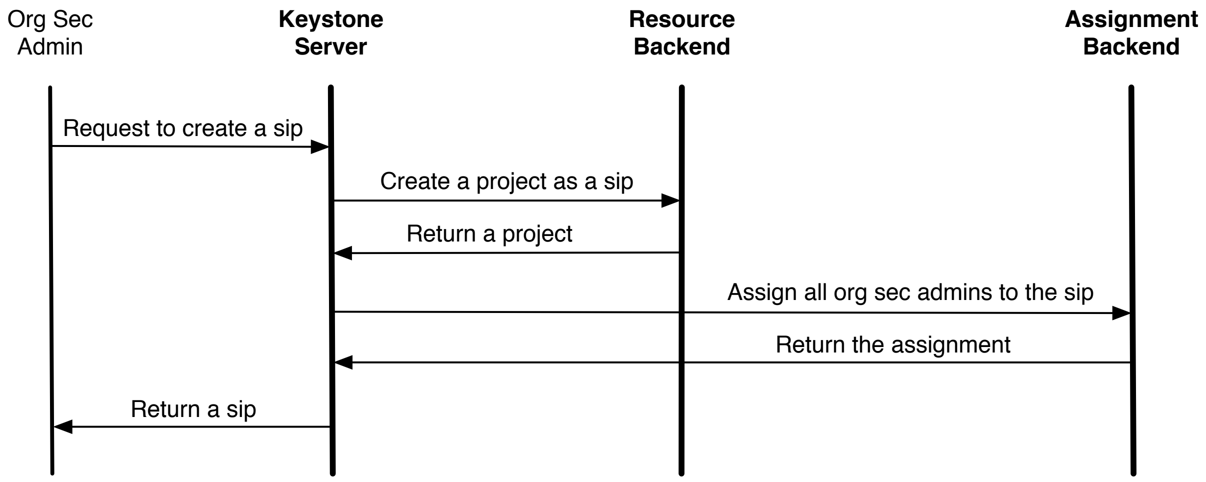


Figure 4.20: Create a Sip

Create a Sip

As we already said that sid domain admin users (organizations security admin users) can create project in the sid domain, which are actually the sips in the model. Figure 4.20 shows the behind scenes actions of creating a sip in a sid domain. Worth to mention here is that, sid domain admin users uses sid domain scoped token to create a sip project. Again, with the creation of a sip, all organizations security admin users get the project admin role over the sip project, which allows them to bring their own users to the sip. With sip project admin roles, organizations security admin users bring their users by assigning them with an operational role like *Member*, which gives users operational permission in the sip project, such as create, update and delete a virtual machine, a storage container, and etc.

When an organization security admin user initiates a sip request and create a sip for a cyber collaboration in the sid, the admin user sends the request with information includes the sip name

```
# create a sip:
curl -i 'http://10.245.121.77:5000/v3/sips' -X POST -H "X-Auth-Token: $TOKEN" -H
"Content-Type: application/json" -H "Accept: application/json" -d '{"sip": {"enab
led": true, "name": "$sip_name", "sid_id": "$sid_id"}}'
```

Figure 4.21: HTTP Request - Create a Sip

```
# assign a role to user on a sip:
curl -i 'http://10.245.121.77:5000/v3/sips/{sip_id}/users/{user_id}/roles/{role_i
d}' -X PUT -H "X-Auth-Token: $TOKEN" -H "Content-Type: application/json" -H "Acce
pt: application/json"
```

Figure 4.22: HTTP Request - Add a User to a Sip

and sid domain id. The request is sent to Keystone server with sid domain scoped token. Figure 4.21 shows an example of http request during this procedure. A proper sid domain scoped token $TOKEN is used for authentication and authorization. Parameter data contains the new created sip name “name” and the sid domain id “sid_id”.

Add a User to a Sip

Organization security admin users can add normal users from their home domain to any project (core project, open project and sips) in the sid domain. Figure 4.22 shows an example of http request during this procedure. A proper token $TOKEN (either a sid domain scoped token a sip project scoped token) is used for authentication and authorization. Parameter data contains the sip project id “sip_id”, the normal user’s id “user_id” and the role’s id “role_id”. This allows user “user_id” access sip project “sip_id” by permissions defined in role “role_id”.

Access to a Sip

Normal users are assigned by their organizations security admin users to a sip project in a sid. This process is transparent to normal users. After the assignment is done, normal users get an operational role over the sip project, which allows them to access cloud resources in the sip project. From a normal user perspective, we have a sequence of actions requesting to access a sip project. Normal users get a project scoped token by requesting Keystone server to access to the sip project.

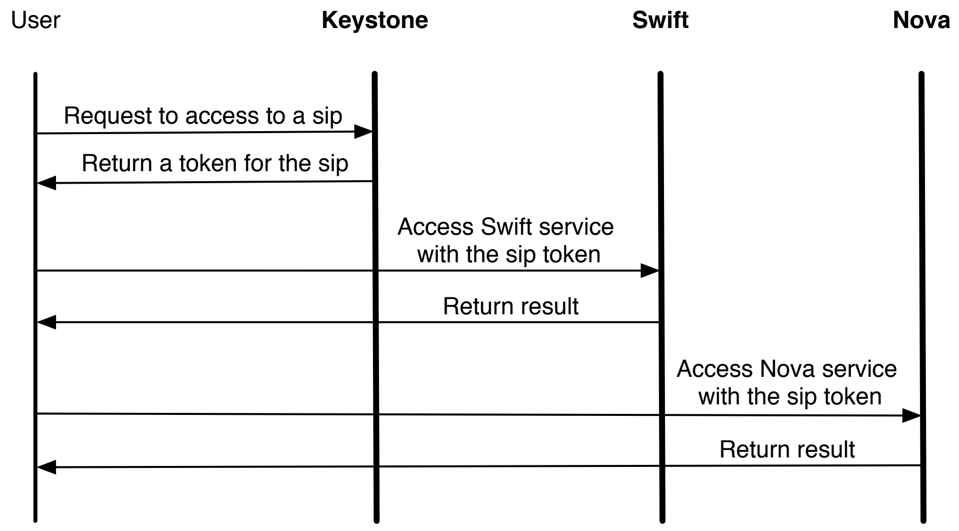

User
Keystone
Swift
Nova
Request to access to a sip
Return a token for the sip
Access Swift service with the sip token
Return result
Access Nova service with the sip token
Return result

Figure 4.23: A User Access to a Sip

Then they can use the token to access resources in the sip project, for example virtual machine resources from Nova service, storage resources from Swift service, and so on. By accessing to the sip project, users can share information and resources with others inside the sip project. In Figure 4.23, we give the sequence of actions that happens when a normal user request to access a sip project. For simplicity, we limited cloud services to include only computing resources service Nova and storage resources service Swift. For other cloud services, the actions follow the same sequence.

Chapter 5: SID-MODEL IN AWS CLOUD IAAS

Part of content from this chapter has been published in paper [19].

Amazon Web Services (AWS) started to offer cloud infrastructure services to enterprises and businesses in the form of web services in 2006. As the lowest level of cloud computing service, infrastructure as a service provides enterprises the opportunity to replace expensive computing infrastructure with low variable cost virtual computing resources in the cloud. With cloud computing, enterprises can easily boot up thousands of servers instantly without deploying real computer servers. It is a great convenience for enterprises and businesses with large quantity of infrastructure to manage and maintain.

With development of over 10 years experience, Amazon Web Services now offers a highly scalable, reliable and low-cost cloud infrastructure platform to the public, with a global infrastructure of 11 regions, 28 availability zones and 52 edge locations, powering thousands of businesses in more than 190 countries around the world. Keeping expanding global infrastructure, AWS commits to help its customers achieve their business goals and meet their global requirements of cloud infrastructure. Each location of AWS is composed of regions and availability zones. Each region is completely independent and isolated from other regions. Availability zones are isolated from each other, while in the same region they are connected through low-latency links.

With respect to security, AWS provides its customers with guidance and expertise through online documents resources and customer services. Customers have access to hundreds of security-specific tools and features across network security, access control, configuration management and data encryption, which help them to meet their security objectives. AWS builds secure data center architectures and networks to satisfy the security requirements of organizations and enterprises, which allows customers to meet their security needs with much lower operational overhead.

AWS consists of many cloud services which customers can use to deploy and realize their business needs. To access the cloud services, customers can use either the AWS Management Console or the Command Line Interface, where the former provides a simple and intuitive user interface, and the latter gives a quick view to resources and can be unified with other tools together to automate management of multiple AWS services though automated scripts. Amazon Elastic Compute Cloud (Amazon EC2) is a cloud service that provides web-scale computing capacity to customers. Amazon EC2 gives customers complete control of the cloud computing resources and allow them to obtain and configure computing capacity easily on demand. Customers can quickly boot new servers instances and scale computing capacity up and down according to their requirements. By using cloud computing, customers only pay for the capacity that they actually use. Amazon Simple Storage Service (Amazon S3) offers a secure and highly scalable object storage for cloud users to manage their data. Policies can be used to manage the data throughout its lifecycle, which provides automatically migration to most storage data without changes to customers' applications. Amazon Relational Database Services (Amazon RDS) offers scalable relational database services in the cloud, which release customers from setting up and operating databases, instead allowing them to focus on application and administration level. It's fast, efficient and allows customers to launch a database in minutes. Amazon Virtual Private Cloud (Amazon VPC) allows customers to define their own virtual network, which is logically isolated from other section of the AWS cloud. Customers have complete control over the virtual networking. They can customize the network configuration as appropriate for their deployment. AWS Identity and Access Management (IAM) enables customers to securely control access to cloud services and resources. By using IAM, customers can define permissions to allow and deny their users' access to AWS services and resources. AWS also offers other services, such as developer tools, management tools, application services, mobile services, and etc.

The material presented in this chapter has been partially published previously in [19]. In this chapter, we first define an access control model based on our understanding of the AWS framework and our need to abstract the main components for subsequent modeling. Then we develop the model of securely sharing information and resources in context of AWS cloud IaaS platform. The latter model realize the abstract model of chapter 2 in AWS. We formalize the administrative model in the next section. Then we give implementation details in the following section.

We assume that each organization has one and only one AWS account. Each user belongs to one and only one organization in AWS public cloud. For simplicity, we constrain operational model to include only two cloud services: Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3).

5.1 AWS Access Control (AWS-AC) Model

We present the Amazon Web Service Access Control (AWS-AC) model in this section. As a public cloud service provider, AWS provides web services to its customers through AWS accounts. Customers who own an account have access to cloud resources. They can create users and grant them access to cloud resources in the account. Each user belongs to a unique account. Users can also access resources in other accounts with federated permissions. We discuss AWS Access Control in two perspectives: within a single account and across accounts.

AWS offers a form of policy-based access control, wherein permissions are defined over cloud resources in a policy file and policies are attached to entities such as users, groups, roles and resources. Figure 5.1 depicts this model within a single account. In this and other figures, the dotted lines denote virtual relations between entities while the solid lines denote explicit relations. Cross-account access will be discussed later in context of Figure 5.2.

AWS-AC has seven components: Accounts (A), Users (U), Groups (G), Roles (R), Services (S), Object Types (OT), and Operations (OPR). We also introduce other entities such as policies and credentials, which are implicitly included in the model.

Accounts: In AWS, Accounts are basic resource containers, which allows customers to own specific amount of (virtual) cloud resources. Accounts are the units of usages of cloud resources and billing. Customers get public cloud services through an AWS account.

Users and Groups: Users represent individuals who can be authenticated by AWS and authorized to access cloud resources through an account. A group is simply a set of users. Users and groups belong to an account. The existence of groups is for the convenience of managing multiple users as a single unit. Each policy attached to a group will apply to all group members. For

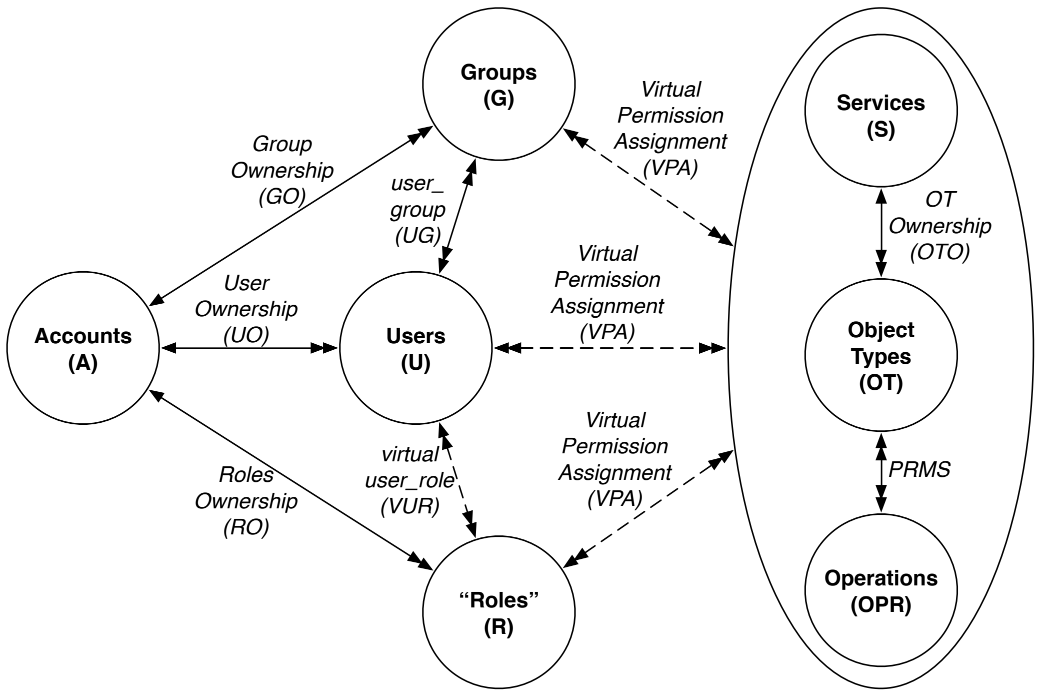


Figure 5.1: AWS Access Control within a Single Account [19]

simplicity, we use the term users to represent both users and groups in the rest of the chapter.

Virtual Permission Assignment: In AWS, users' permissions over services and resources are defined in policy files. Policy files can be attached to a user, a group, a role or a specific cloud resource. By attaching a policy to a user, a group or a role, users gain permissions to corresponding cloud resources. The policy defines the actions the user will perform and cloud resources on which the actions will function. Multiple permissions can be defined in one policy file. Multiple policy files can be attached to one entity. AWS achieves permission assignment in a virtual manner via the policies attached to various relevant entities.

Roles: Unlike roles in RBAC, roles in AWS are used for both permission assignment and trust relationship definition of cross-account access. Permissions are given to a role by attaching policy files that define the role's permissions over cloud srevices and resource. A role defines the trust relation between principals which can be either another AWS account and its users or the owner account and internal users. Users use roles though the *AssumeRole* action to access to corresponding cloud resources. To emphasize the difference between the usual concept of roles in

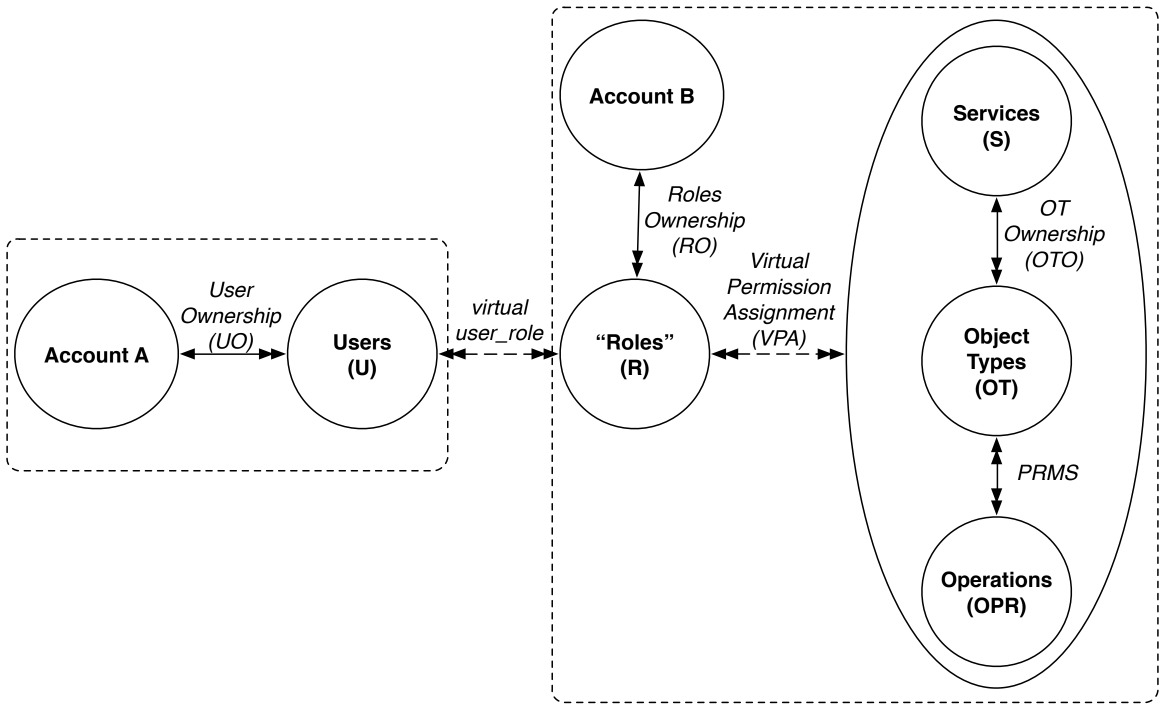


Figure 5.2: AWS Access Control accross Accounts [Users in Account A Access Services and Resources in Account B] [19]

RBAC and the term roles in AWS, we use quotation marks around Roles in our figures.

Services: Services refer to cloud services AWS provides to its customers. Cloud Service Provider (CSP) leases cloud resources to its customers in terms of services. AWS provides customers with services such as compute, storage, networking, administration, and database.

Object Types and Operations: An Object Type represents a specific type of object. From the CSP's viewpoint, objects are more like services. We define object types as particular service types the cloud provides. For instance, with compute service EC2, the object type is a virtual machine; with storage service S3, the object type is a bucket; etc.

Credentials: AWS credentials are used for both authentication and authorization. Account owners can create IAM users with their own security credentials to allow these users to access AWS services and resources. Account owners can also grant external federated users from other accounts with temporary security credentials to allow them to access the account's AWS services and resources.

Cross-account access: Users in one AWS account can access services and resources in another AWS account through the action *AssumeRole* with temporary security credentials, as shown in Figure 5.2. Users from account A access services and resources in account B through roles created in account B, by being attached with policies of the action *AssumeRole* and a target resource.

With the concepts described above, we formalize AWS-AC model as follows.

Definition 1. AWS-AC model has the following components.

- A, U, G, R, S, OT and OPR are finite sets of existing accounts, users, groups, roles, services, object types and operations respectively in the AWS public cloud system.

- User Ownership (UO) : is a function UO : U $\rightarrow$ A, mapping each user to its owning account. Equivalently viewed as a many-to-one relation UO $\subseteq$ U $\times$ A.

- Group Ownership (GO) : is a function GO : G $\rightarrow$ A, mapping each group to its owning account. Equivalently viewed as a many-to-one relation GO $\subseteq$ G $\times$ A.

- Roles Ownership (RO) : is a function RO : R $\rightarrow$ A, mapping each role to its owning account. Equivalently viewed as a many-to-one relation GO $\subseteq$ R $\times$ A.

- Object Type Owner (OTO) : is a function OTO : OT $\rightarrow$ S, mapping each object type to its owning service. Equivalently viewed as a many-to-one relation OTO $\subseteq$ OT $\times$ S.

- Virtual Permission Assignment (VPA) : is a function VPA : PERMS $\rightarrow$ R, mapping permissions to roles. Equivalently viewed as a many-to-many relation VPA $\subseteq$ PERMS $\times$ R, resulting from policies attached to user, groups and roles entities.

- user_group (UG) : is a function UG : U $\rightarrow$ G, mapping users to groups where the user and group are owned by the same account. Equivalently viewed as a many-to-many relation UG $\subseteq$ U $\times$ G.

- virtual user_role (VUR) : is a function VUR : U $\rightarrow$ R, mapping users to roles. Equivalently viewed as a many-to-manuy virtual relation VUR $\subseteq$ U $\times$ R, resulting from policies attached to user entity. *AssumeRole* is an action allowing users to activate a role authorized in VUR.

- PERMS = OT $\times$ OPR, is the set of permissions.

5.2 AWS-AC Model with SID Extension (AWS-AC-SID)

In this section, we present an access control model for AWS with the Secure Isolated Domain extension (AWS-AC-SID). We build the AWS-AC-SID model on top of the AWS-AC model to include Secure Isolated Domain (SID) functionality [18].

In SID-model, the basic resource container is projects such as core projects, open projects and secure isolated projects. Secure isolated domains is the administrative boundary of resource containers. In AWS, the basic resource container is AWS accounts. which is also the only level of resource containers. To apply SID-model to AWS platform, we use AWS accounts to carry on the basic resource containers projects in SID-model. Each project inside a sid will be a AWS account in AWS-AC-SID model.

We present the AWS-AC-SID model so as to cover only the additional components added to the AWS-AC model. Figure 5.3 shows the AWS-AC-SID model, where we ignore groups for simplicity. In the rest of the chapter, group is used to denote a group of organizations, rather than the groups component of AWS-AC model.

5.2.1 Components

Definition 2. AWS-AC-SID model has the following components in addition to AWS-AC model: Secure Isolated Domain (SID), Secure Isolated Project (SIP), Expert Users (EU), Core Project (CP), Open Project (OP) and Resources (RS). We have introduced these concepts and described these components when we introduced SID-model in chapter 3. We will give a simple review of these components.

Secure Isolated Domain is a special domain, holding security information and resources for cross-organizational security collaborations. Sid provides a secure isolated environment for cyber security collaborations in a community of organizations. Each sid holds a core project, an open project and multiple secure isolated projects. **Secure Isolated Project** provides a controlled environment for a group of organizations within the community to collaborate and coordinate on cyber incidents. **Core Project** is a shared project holding cyber security committee for the community

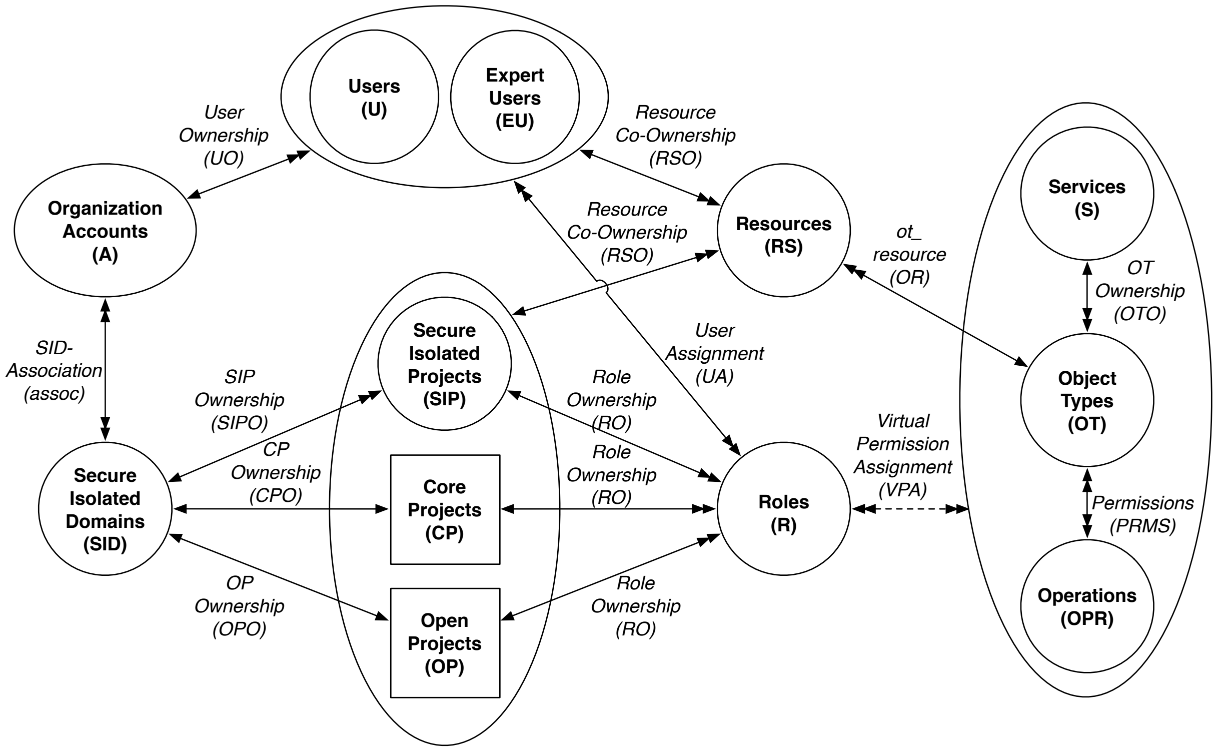


Figure 5.3: AWS Access Control Model with SID Extension (AWS-AC-SID)

of organizations. **Open Project** is an open shared project where users from the community of organizations share common cyber security information and resources. **Expert Users** brings professionals cyber security skills to the community. **Resources** refers to cloud assets such as virtual machines, databases, storages, etc.

In the following, we give formalization of concepts introduced above, as well as the relations among them.

- SID, SIP, CP, OP, EU, RS and BO are finite sets of existing secure isolated domain, secure isolated projects, core projects, open projects, expert users, resources, and storage bucket objects respectively in a AWS cloud system.

- Virtual Machines (VM) is object type for compute service in AWS cloud platform.

- Storage Buckets (SB) and Storage Bucket Objects (BO) are two object types for storage service in AWS cloud platform.

- Secure Isolated Project Ownership (SIPO) : is a function SIPO : SIP $\rightarrow$ SID, mapping a single secure isolated project to its owning sid. Equivalently viewed as a many-to-one relation $\mathrm{SIPO} \subseteq \mathrm{SIP} \times \mathrm{SID}$.

- Core Project Ownership (CPO) : is a function $\mathrm{CPO} : \mathrm{CP} \rightarrow \mathrm{SID}$, mapping a single core project to its owning sid. Equivalently viewed as a one-to-one relation $\mathrm{CPO} \subseteq \mathrm{CP} \times \mathrm{SID}$.

- Open Project Ownership (OPO) : is a function $\mathrm{OPO} : \mathrm{OP} \rightarrow \mathrm{SID}$, mapping a single open project to its owning sid. Equivalently viewed as a one-to-one relation $\mathrm{OPO} \subseteq \mathrm{OP} \times \mathrm{SID}$.

- Role Ownership (RO) : is a function $\mathrm{RO} : \mathrm{R} \rightarrow (\mathrm{SIP} \cup \mathrm{CP} \cup \mathrm{OP})$, mapping a role to its owning project (core/open/secure isolated project).

- Resource Co-Ownership (RSO) : is a function $\mathrm{RSO} : \mathrm{RS} \rightarrow ((\mathrm{SIP} \cup \mathrm{CP} \cup \mathrm{OP}), (\mathrm{U} \cup \mathrm{EU}))$, mapping resources to its owning project and user. Equivalently viewed as a many-to-one relation $\mathrm{RSO} \subseteq \mathrm{RS} \times ((\mathrm{SIP} \cup \mathrm{CP} \cup \mathrm{OP}) \times (\mathrm{U} \cup \mathrm{EU}))$.

- Bucket Object Owner(BOO): $\mathrm{BO} \rightarrow \mathrm{SB}$, a function mapping a storage object to its owning bucket. Equivalently viewed as a many-to-one relation $\mathrm{BOO} \subseteq \mathrm{BO} \times \mathrm{SB}$.

- SID association (assoc): is a function assoc : $\mathrm{SID} \rightarrow 2^{\mathrm{A}}$, mapping a SID to all its member organization accounts.

- ot_resource (OR) : is a function $\mathrm{OR} : \mathrm{OT} \rightarrow \mathrm{RS}$, mapping object types to resources. Equivalently viewed as a one-to-many relation $\mathrm{OR} \subseteq \mathrm{OT} \times \mathrm{RS}$.

5.2.2 Administrative Model

The general concept of a sid is a secure isolated container for a community of organizations to share their security data. In context of public cloud, there are two approaches to build sid functionality. One is to integrate sid function as part of the cloud functionality. The other is to build it as a service provided by a third party in the cloud to its customers. The first approach is applicable only if the cloud provider is willing to do so. The second approach requires to build additional services on the cloud platform. For proprietary products such as AWS, sid functionality can be provided as a service by a third party who is a customer of AWS.

An AWS account is a secure isolated place in the cloud system (within the limits of isolation assurance enforced by Amazon). An account is the smallest unit of secure isolated space. Bigger spaces for isolated sharing can be built by using multiple accounts in AWS.

A SID-manager is an automated agent that manages sids and their constituent components through their life cycle. It is built using AWS standard functionality. SID-manager processes SID-requests from communities of organizations and constructs a separate sid for each community. Within each sid it facilitates the creation and deletion of sips for the set of organizations within the community of organizations of that sid. Each time a cyber collaboration request is sent to SID-manager, it picks up an available sip account and assign it to a community of organizations in a sid. After the collaboration is done, the sip will be cleaned up and be available for the next collaboration.

As a security service provided for customers in AWS public cloud, SID-service might belong to a cyber security company who has the trust of the organizations in AWS. In general there may be multiple sid providers. Each sid has a core project and an open project as a security service provided to all organizations in the sid community. Secure isolated domain, core project and open project are created when the SID-request is sent by a community of organizations. Each organization can join several sids with different communities of organizations. Each of these sids are isolated from each other.

We constrain the roles in two types: administrative role and member role, which separately denotes the permission of being able to manage users and permissions only for resources respectively. We use roles *SIDadmin* to respectively represent limited administrative power in the core project, open project or a sip, which gives the core project, open project or sip admin users permission to add and remove other users from their home account to the core project, open project or a sip. We use roles *SIDmember* to represent operative permissions which can be given to normal users to access to the core project, open project or a sip. Since roles in AWS are local, these roles are sets of roles, *SIPadmin* represents the set of admin roles in core projects, open projects and all sips; while *SIDmember* represents the set of member roles in these projects.

The administrative aspects of AWS-AC-SID model are discussed informally below. A formal specification is given in Table 5.1 and Table 5.2.

Create a sid: Remind again that the creation of sid is based on agreement among the community of organizations. After all the organizations in the community have agreed, one security admin from an organization representing the group of organizations request the creation of a sid with parameters including all the security admin users, each representing one organization in the community. Remind again that *uSet* denotes a fixed group of security admin users from all organizations of the community, with one admin user for each organization. When a sid is created, the security admin user who issues sid creation command along with all other security admin users from *uSet* will become the limited administrative users of the sid, in which each organization in the community has equal limited administrative power.

Since in AWS, each project in a sid are realized by AWS accounts, *uSet* will be assigned with admin roles (belonging to *SIDadmin*) to AWS accounts which carry the core project and open project. One admin role for core project and the other for open project. With these two roles, admin users (from *uSet*) can add and remove other users from their home account to the core project and open project. Two member roles (belonging to *SIDmember*) will also be created respectively in core project and open project for normal users to join into these two projects. The open project is open for all the users from the community of organizations.

Delete a sid: One admin user from *uSet* representing the group of organizations initilizes the sid delete request. When the sid is deleted, the core project, open project and all the sips inside the sid (if any) need to be securely deleted. All the assigned resources and users will be securely revoked from the sid. The created admin roles (from *SIDadmin*) and member roles (from *SIDmember*) will be deleted from core project and open project or any secure isolated projects. All projects inside the sid will be removed from the sid and the sid will be deleted.

Create a sip: A sip is created whenever there is a need for cyber collaborations among a set of the community organizations. Any organization security admin user from *uSet* can represent the set of the community of organizations to create a sip. All these admin users are assigned to the sip

Table 5.1: AWS-AC-SID Administrative Model

Operation	Authorization Requirement	Update
SidCreate(adminu, uSet, sid) *[/* An admin user representing uSet creates a sid */]*	adminu $\in$ *uSet* $\wedge$ adminu $\in$ U $\wedge$ sid $\notin$ SID	SID′ = SID $\cup$ {sid}; assoc(sid) = $\bigcup_{adminu \in uSet}$ UO(adminu); CP′ = CP $\cup$ {cp}; CPO(cp) = sid; RO(cpra) = cp; RO(cprm) = cp; OP′ = OP $\cup$ {op}; OPO(op) = sid; RO(opra) = op; RO(oprm) = op; R′ = R $\cup$ {cpra, cprm, opra, oprm}; *SIDadmin*′ = *SIDadmin* $\cup$ {cpra, opra}; *SIDmember*′ = *SIDmember* $\cup$ {cprm, oprm}; UA′ = UA $\cup$ {(uSet, cpra), (uSet, opra)}.
SidDelete(adminu, uSet, sid) */* An admin user representing uSet deletes the sid*/*	adminu $\in$ *uSet* $\wedge$ adminu $\in$ U $\wedge$ $\exists$ r $\in$ *SIDadmin*.((adminu, r) $\in$ UA) $\wedge$ assoc(sid) = $\bigcup_{adminu \in uSet}$ UO(adminu) $\wedge$ sid $\in$ SID	SID′ = SID - {sid}; assoc(sid) = NULL; CP′ = CP - {cp}; CPO(cp) = NULL; RO(cpra) = NULL; RO(cprm) = NULL; OP′ = OP - {op}; OPO(op) = NULL; RO(opra) = NULL; RO(oprm) = NULL; R′ = R - {cpra, cprm, opra, oprm}; *SIDadmin*′ = *SIDadmin* - {cpra, opra}; *SIDmember*′ = *SIDmember* - {cprm, oprm}; UA′ = UA - {(uSet, cpra), (uSet, opra)}; $\forall$u $\in$ U.(UA′ = UA - {(u, cprm), (u, oprm)}); if $\exists$(u, sip, r) $\in$ ((U $\cup$ EU), (SIP), (*SIDadmin* $\vee$ *SIDmember*)).({(u, r)} $\in$ UA $\wedge$ RO(r) $\in$ = sip), then UA′ = UA - {(u, r)} $\wedge$ SIP′ = SIP - {sip} $\wedge$ R′ = R - {r}.
SipCreate(adminu, sip, sid) */* An admin user representing uSet creates a sip */*	adminu $\in$ *uSet* $\wedge$ adminu $\in$ U $\wedge$ UO(adminu) $\in$ assoc(sid) $\wedge$ sip $\notin$ SIP	SIP′ = SIP $\cup$ {sip}; SIPO(sip) = sid; RO(sipra) = sip; RO(siprm) = sip; R′ = R $\cup$ {sipra, siprm}; *SIDadmin*′ = *SIDadmin* $\cup$ {sipra}; *SIDmember*′ = *SIDmember* $\cup$ {siprm}; UA′ = UA $\cup$ {(uSet, sipra)}.
SipDelete(adminu, sip, sid) */* An admin user representing uSet deletes a sip*/*	adminu $\in$ *uSet* $\wedge$ adminu $\in$ U $\wedge$ UO(adminu) $\in$ assoc(sid) $\wedge$ SIPO(sip) = sid $\wedge$ sip $\in$ SIP	SIP′ = SIP - {sip}; SIPO(sip) = NULL; RO(sipra) = NULL; RO(siprm) = NULL; R′ = R - {sipra, siprm}; *SIDadmin*′ = *SIDadmin* - {sipra}; *SIDmember*′ = *SIDmember* - {siprm}; UA′ = UA - {(uSet, sipra)}.

Table 5.2: AWS-AC-SID Administrative Model (continued)

Operation	Authorization Requirement	Update
UserAdd(adminu, u, p, sid) */* Admin users add a user from his home domain to a cp, op or sip */*	adminu $\in$ U $\wedge$ (adminu, sidra) $\in$ UA $\wedge$ RO(sidra) = p $\wedge$ u $\in$ U $\wedge$ UO(u) = UO(adminu) $\wedge$ p $\in$ (CP $\cup$ OP $\cup$ SIP) $\wedge$ (CPO(p) = sid $\vee$ OPO(p) = sid $\vee$ SIP(p) = sid)	R$'$ = R $\cup$ {sidrm}; RO(sidrm) = p; UA$'$ = UA $\cup$ {(u, sidrm)}.
UserRemove(adminu, u, p, sid) */* Admin users remove a user from a cp, op or sip */*	adminu $\in$ U $\wedge$ (adminu, sidra) $\in$ UA $\wedge$ RO(sidra) = p $\wedge$ u $\in$ U $\wedge$ UO(u) = UO(adminu) $\wedge$ p $\in$ (CP $\cup$ OP $\cup$ SIP) $\wedge$ (CPO(p) = sid $\vee$ OPO(p) = sid $\vee$ SIP(p) = sid) $\wedge$ (u, sidrm) $\in$ UA $\wedge$ RO(sidrm) = p	R$'$ = R - {sidrm}; RO(sidrm) = NULL; UA$'$ = UA - {(u, sidrm)}.
EUserAdd(adminu, eu, p, sid) */* Admin users add an expert user to a cp or sip */*	adminu $\in$ U $\wedge$ (adminu, sidra) $\in$ UA $\wedge$ RO(sidra) = p $\wedge$ eu $\in$ EU $\wedge$ p $\in$ (CP $\cup$ SIP) $\wedge$ (CPO(p) = sid $\vee$ SIP(p) = sid)	R$'$ = R $\cup$ {sidrm}; RO(sidrm) = p; UA$'$ = UA $\cup$ {(eu, sidrm)}.
EUserRemove(adminu, eu, p, sid) */* Admin users remove an expert user from a cp or sip */*	adminu $\in$ U $\wedge$ (adminu, sidra) $\in$ UA $\wedge$ RO(sidra) = p $\wedge$ eu $\in$ EU $\wedge$ p $\in$ (CP $\cup$ SIP) $\wedge$ (CPO(p) = sid $\vee$ SIP(p) = sid) $\wedge$ (eu, sidrm) $\in$ UA $\wedge$ RO(sidrm) = p	R$'$ = R - {sidrm}; RO(sidrm) = NULL; UA$'$ = UA - {(eu, sidrm)}.

with limited admintrative permissions, which defined by role set *SIDadmin*. These roles give sip admin users permissions to add and remove other users from their home account to the sip.

Delete a sip: After the collaboration is finished, a sip can be securely deleted. The delete command is issued by the any security admin user (*uSet*). All information and resources are securely deleted in the sip. Roles created for users to join the sip are deleted, which result in users unassignment from the sip.

Add/remove a user to/from a core project: Remind that core project admin users are the set of security administrative users (*uSet*) from the community of organizations. These limited administrative users can add/remove existing users of their organizations to/from core project by add/remove normal users to/from a member role in the core project.

Add/remove a user to/from a sip: Remind that users from *uSet* who are assigned with an admin role (belonging to *SIDadmin*) has the limited administrative power in the sip. They can add/remove users of their home accounts to/from the corresponding sip due to the need of collaboration add/remove normal users to/from a member role in the sip.

Table 5.3: AWS-AC-SID Operational Model

Operation	Authorization Requirement	Update
CreateVM(vm, p, u) *`/*` A user creates a vm `*/`*	$vm \notin RS \wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge \exists (perms, prm) \in VPA.(perms = (vm, create) \wedge RO(prm) = p \wedge (u, prm) \in UA \wedge (prm \in \mathit{SIDmember}))$	$RS' = RS \cup \{vm\}$; $RSO' = RSO \cup \{(vm, (p, u))\}$; $OR(vm) = VM$.
DeleteVM(vm, p, u) *`/*` A user deletes a vm `*/`*	$vm \in RS \wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge \exists (perms, prm) \in VPA.(perms = (vm, delete) \wedge RO(prm) = p \wedge (u, prm) \in UA \wedge (prm \in \mathit{SIDmember})) \wedge RSO(vm) = \{(p, u)\}$	$RS' = RS - \{vm\}$; $RSO' = RSO - \{(vm, (p, u))\}$; $vm = NULL$.
CreateSBucket(sb, p, u) *`/*` A user creates a storage bucket `*/`*	$sb \notin RS \wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge \exists (perms, prm) \in VPA.(perms = (sb, create) \wedge RO(prm) = p \wedge (u, prm) \in UA \wedge (prm \in \mathit{SIDmember}))$	$RS' = RS \cup \{sb\}$; $RSO' = RSO \cup \{(sb, (p, u))\}$; $OR(sb) = SB$.
DeleteSBucket(sb, p, u) *`/*` A user deletes a storage bucket `*/`*	$sb \in RS \wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge \exists (perms, prm) \in VPA.(perms = (sb, delete) \wedge RO(prm) = p \wedge (u, prm) \in UA \wedge (prm \in \mathit{SIDmember})) \wedge RSO(sb) = \{(p, u)\}$	$RS' = RS - \{sb\}$; $RSO' = RSO - \{(sb, (p, u))\}$; $sb = NULL$.
CreateObject(co, sb, p, u) *`/*` A user creates a storage container object `*/`*	$co \notin RS \wedge sb \in RS \wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge \exists (perms, prm) \in VPA.(perms = (co, create) \wedge RO(prm) = p \wedge (u, prm) \in UA \wedge (prm \in \mathit{SIDmember})) \wedge RSO(sb) = (p, u)$	$RS' = RS \cup \{co\}$; $RSO' = RSO \cup \{(co, (p, u))\}$; $OR(co) = CO$.
DeleteObject(co, sb, p, u) *`/*` A user delete a storage container object `*/`*	$co \in RS \wedge RSO(co) = \{(p, u)\} \wedge sb \in RS \wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge \exists (perms, prm) \in VPA.(perms = (co, delete) \wedge RO(prm) = p \wedge (u, prm) \in UA \wedge (prm \in \mathit{SIDmember})) \wedge RSO(sb) = (p, u)$	$RS' = RS - \{co\}$; $RSO' = RSO - \{(co, (p, u))\}$; $co = NULL$.

Add/remove a user to an open project: Every user in the collaborative community of organizations is allowed to join open projects. Users in open projects have equal but limited permissions, which is carried on by a member role (belonging to *SIDmember*) in the open project. They can share cyber data, but have no control over other users. Organization security admin users add/remove normal users from their organizations to/from open projects.

Add/remove an expert user to/from a sip or core project: Expert users are introduced for their expertise and professionals. Expert users can be added/remove to/from core projects and sips as a member through a member role in the project. Users from *uSet* can request to add/remove expert users to/from the core project or a sip.

5.2.3 Operational Model

In the operational model, we mainly show how and what operations a normal user can issue in the model. For simplicity, we only demonstrate the core operations on virtual machines and storage buckets, including creation and delete. *Create* method allows users to create a new instance of virtual machine or a storage bucket in a core project, open project or a sip. *Delete* method allows users to delete an existing instance of a virtual machine or storage bucket in a project. For objects, we can upload objects and download objects from a storage bucket. In Table 5.3, we give the details of operational model.

After a user is assigned to the core project, open project or a sip, the user can issue following operations:

CreateVM/DeleteVM: A user can create/delete a virtual machines in the core project, open project or a sip, to which the user is assigned through the specific member role inside the project.

CreateBucket/DeleteBucket: A user can create/delete a storage bucket in the core project, open project or a sip, to which the user is assigned through the specific member role inside the project.

CreateObject/DeleteObject: A user can create/delete a bucket object in a storage bucket in the core project, open project or a sip, to which the user is assigned through the specific member role inside the project.

5.3 Enforcement

We implemented AWS-AC-SID model on the current AWS release. Accounts form the basic resource boundary in AWS. Inside one account, there is no clear further sub-division of resources. Owning an AWS account give a customer root user privilege over the account. Root user has complete access power over all the services and resources in the account. Root user can create users of all level of access permissions in the account. Root user can create administrative user for the account, who can have the full access permissions except owning the account. Administrative users can further create roles and other users. Users permissions over services and resources are defined in policy files which are attached to users (among other entities).

One way to enforce SID-model in AWS public cloud is to design it as a service, which is provided by a third trusted party in AWS public cloud. We implemented our own SID-server to provide SID-service to organizations in AWS public cloud.

One thing need to mention is that since AWS roles are local, we use multiple roles to realize the concept of role *SIDadmin* and *SIDmember*. We use roles *CPadmin*, *OPadmin* and *SIPadmin* to respectively represent limited administrative power in the core project, open project or a sip, which gives the core project, open project or sip admin users permission to add and remove other users from their home account to the core project, open project or a sip. We use roles *CPmember*, *OPmember* and *SIPmember* to represent operative permissions which can be given to normal users to access to the core project, open project or a sip. Since roles in AWS are local, these roles are sets of roles, *CPadmin*, *OPadmin* and *SIPadmin*, are separately representing the set of admin roles in core projects, open projects and all sips; while *CPmember*, *OPmember* and *SIPmember* are separately representing the set of member roles in core projects, open projects and all sips.

5.3.1 Functionalities

SID-server:

As a SID-server, all we need is to be able to handle requests from organizations and maintain sid related information in the server backend. We use a REST API module running on apache service as our SID-server. Organizations send REST API requests to SID-server and get response from the server.

SID-server consists of a running backend, a SID-manager AWS account and a number of SID-operational accounts. The running backend accept and process SID-requests sent from community organizations. SID-manager account manages the SID-operational accounts in response to the requests. SID-operational accounts are for sips, whenever a sip creation request is sent to SID-server, an available SID-operational account will be used and marked as a sip account associated with a sid of a group of organizations. For each sid, with the creation of the sid, a core project

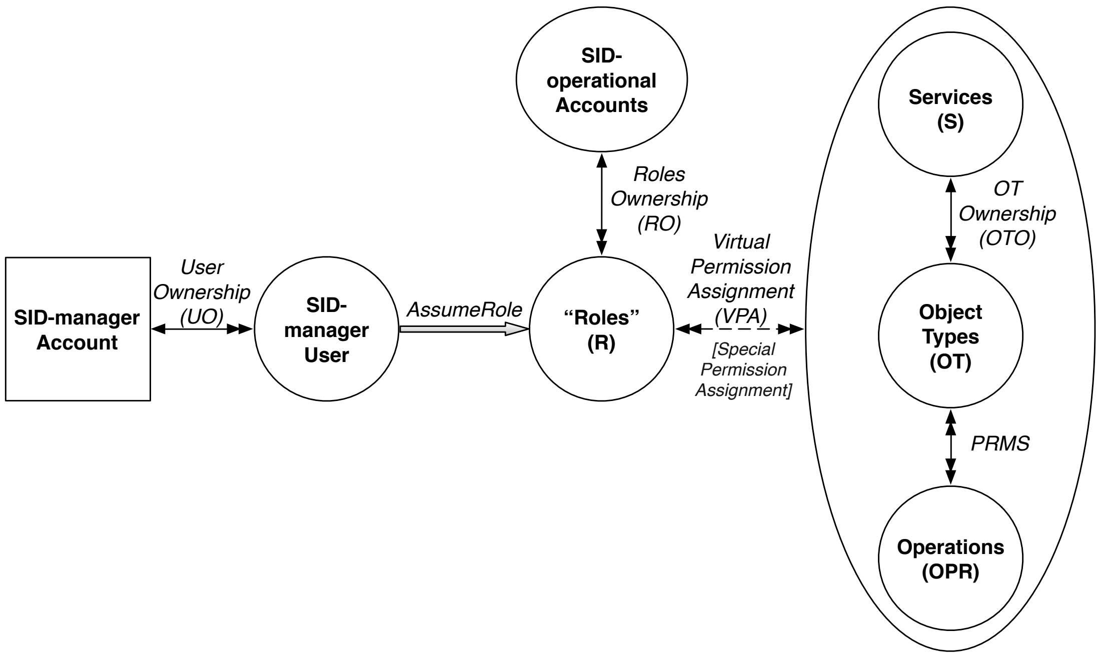
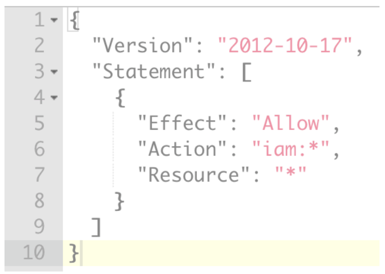


Figure 5.4: Setup SID-service



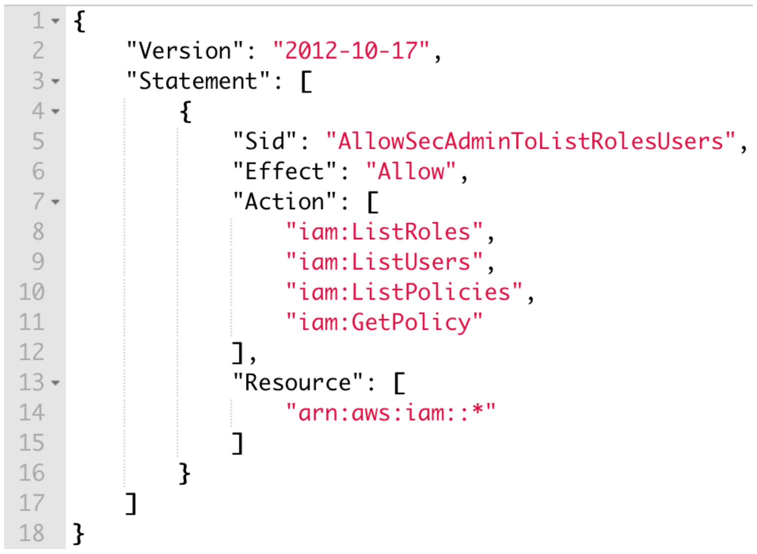
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:*",
      "Resource": "*"
    }
  ]
}
```

Figure 5.5: Policy of SID-manager User

and an open project are created. Those two accounts are also selected from the pool of available SID-operational accounts.

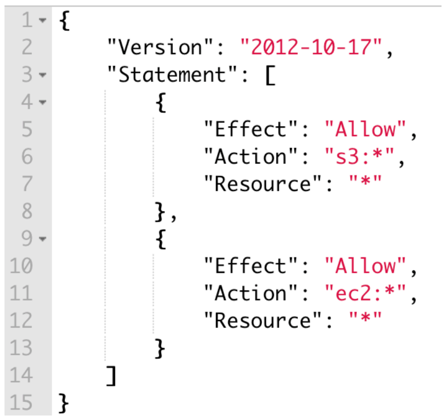
SID Special Manager User:

For each SID-operational account, we have a special manager user for the account with specific policy attached to it which allows the manager user to create roles for upcoming SID-requests. Figure 5.4 illustrates this process. The special manager user access to SID-operational accounts though *AssumeRole* function. We pre-setup administrative policy for the special SID-manager user in each SID-operational accounts, as show in figure 5.5, which give the manager user permissions



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowSecAdminToListRolesUsers",
            "Effect": "Allow",
            "Action": [
                "iam:ListRoles",
                "iam:ListUsers",
                "iam:ListPolicies",
                "iam:GetPolicy"
            ],
            "Resource": [
                "arn:aws:iam::*"
            ]
        }
    ]
}
```

Figure 5.6: Core Project Admin User Policy

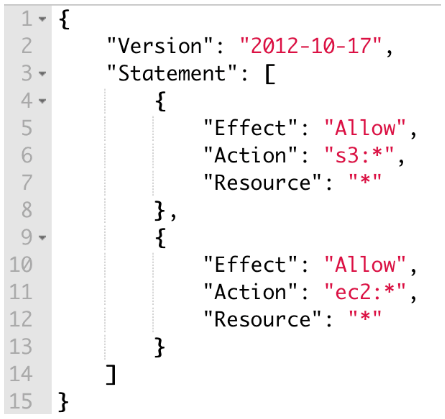


```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:*",
            "Resource": "*"
        }
    ]
}
```

Figure 5.7: Core Project Member User Policy

to full access of IAM permissions. During sip request processing, the special SID-manager user will automatically respond to requests from organizations.

Since each SID-operational accounts will be possessed by different community of organizations during different times, the special manager user is in charge of creating new roles for a collaborative community and erase everything after the collaboration and prepare the operational accounts available for next sip request.



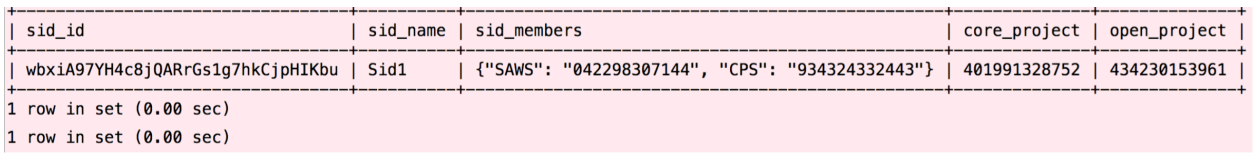
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:*",
            "Resource": "*"
        }
    ]
}
```

Figure 5.8: Open Project Member User Policy

Roles:

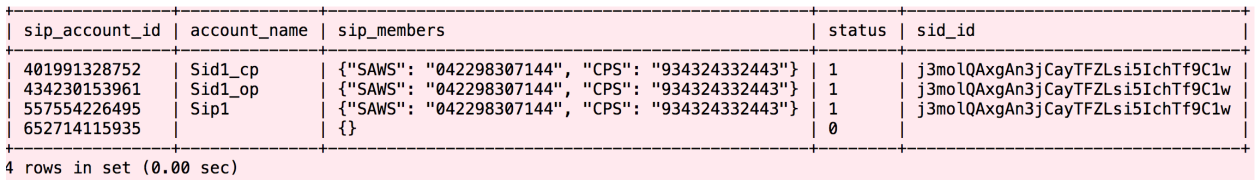
The core project has two types of roles created in the account: *CPadmin* and *CPmember*. Role *CPadmin* has a policy specifying the permission which allows the admin user have limited administrative power toward core project account, which is limited to add/remove member users from its own organization. Figure 5.6 gives an example of the policy of a cp admin user. Role *CPmember* specifies permissions for member users from organizations to have operational rights in the core project, such as access to S3, EC2, database, and etc. For our implementation situation, we simply give member role S3 and EC2 permissions, as shown in Figure 5.7. In your implementation, you can always give more permissions according to your case and scenario. The open project has the role *OPmember* created with the sid creation, which has the policies allows all users from the community to access the open project account, as shown in Figure 5.8.

A sip has similar roles setting as core project. A sip has two types of roles created in the account: *SIPPadmin* and *SIPmember*. Role *SIPadmin* has a policy specifying the permission which allows the security admin user have limited administrative power toward sip account, which is limited to add/remove member users from its own organization. Role *SIPmember* specifies permissions for member users from organizations to have operational rights in the sip.



```
+----------------------------------+----------+----------------------------------------------+--------------+--------------+
| sid_id                           | sid_name | sid_members                                  | core_project | open_project |
+----------------------------------+----------+----------------------------------------------+--------------+--------------+
| wbxiA97YH4c8jQARrGs1g7hkCjpHIKbu | Sid1     | {"SAWS": "042298307144", "CPS": "934324332443"} | 401991328752 | 434230153961 |
+----------------------------------+----------+----------------------------------------------+--------------+--------------+
1 row in set (0.00 sec)
1 row in set (0.00 sec)
```

Figure 5.9: SIDs Table



```
+----------------+--------------+-------------------------------------------------+--------+----------------------------------+
| sip_account_id | account_name | sip_members                                     | status | sid_id                           |
+----------------+--------------+-------------------------------------------------+--------+----------------------------------+
| 401991328752   | Sid1_cp      | {"SAWS": "042298307144", "CPS": "934324332443"} | 1      | j3molQAxgAn3jCayTFZLsi5IchTf9C1w |
| 434230153961   | Sid1_op      | {"SAWS": "042298307144", "CPS": "934324332443"} | 1      | j3molQAxgAn3jCayTFZLsi5IchTf9C1w |
| 557554226495   | Sip1         | {"SAWS": "042298307144", "CPS": "934324332443"} | 1      | j3molQAxgAn3jCayTFZLsi5IchTf9C1w |
| 652714115935   |              | {}                                              | 0      |                                  |
+----------------+--------------+-------------------------------------------------+--------+----------------------------------+
4 rows in set (0.00 sec)
```

Figure 5.10: SIPs Table

Database:

SID-manager maintains the association information for each sid with its member organizations, core project, open project and sip accounts in the community. With security administrative users (*uSet*) from organizations, each organization in the community has one and only one security administrative user in *uSet*. In the association SID-manager mains a list of organizations accounts name and number.

We have two tables in database to maintain the association information for all sids and sips. One is *SIDs* table to have basic information for each sid, include sid id, sid name, sid members, core project id, open project id. The other is *SIPs* table, having information for all core projects, open projects and sips. Figure 5.9 and figure 5.10 shows examples of these two tables. Status in SIPs table shows the availability of a SID-operational AWS account, whether it is possessed by a core project, open project or sip, or not. Status 1 means it is currently possessed while 0 means it is available to be chose.

SID-requests Handling:

A representative organization from a community of organizations can send SID-request to SID-server. An administrative user from *uSet* sends a SID-request to SID-server. SID-manager creates a sid by adding the sid association information to database table and create a core project and open

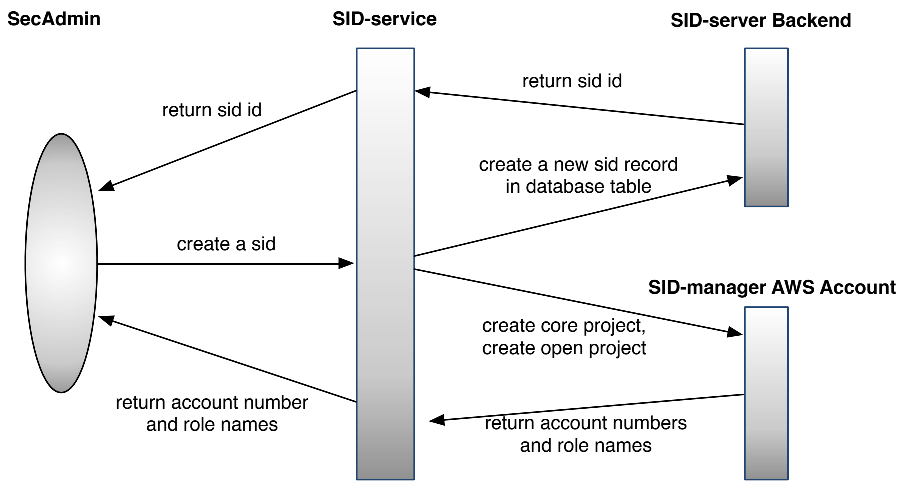

SecAdmin
SID-service
SID-server Backend
return sid id
return sid id
create a new sid record in database table
create a sid
SID-manager AWS Account
create core project, create open project
return account number and role names
return account numbers and role names

Figure 5.11: Process of Creating a Sid

project as part of initialization of the sid, and creates all the roles that needed for each organization to access to the sid. The whole process is completed automatically, as shown in Figure 5.11. After the request, the community of organization should have their sid with a core project and open project. The set of security administrative users *uSet* should have limited admin access to core project and all users from the community should have member access to open project. Admin users from *uSet* should be able to send sip creation request to SID-server.

SID-manager chooses available AWS opeational accounts as core project and open project accounts. SID-manager create roles *CPadmin*, *CPmember* and *OPmember* roles for each organization in the community. Then SID-manager returns core project and open project account number with all new created roles for each organization in the community. Security admin users from *uSet* and normal users from organizations then can access to core project through these roles. We call users who got *CPadmin* role CP admin users. Knowing the role names, sip account numbers, security administrative users can login to the core project and bring normal users from their organization account to the core project and open project.

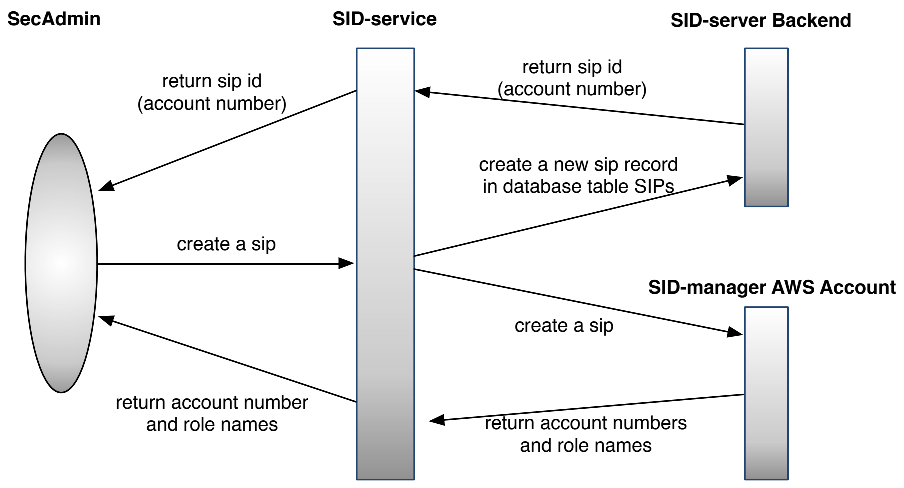

SecAdmin
SID-service
SID-server Backend
return sip id (account number)
return sip id (account number)
create a new sip record in database table SIPs
create a sip
SID-manager AWS Account
create a sip
return account number and role names
return account numbers and role names

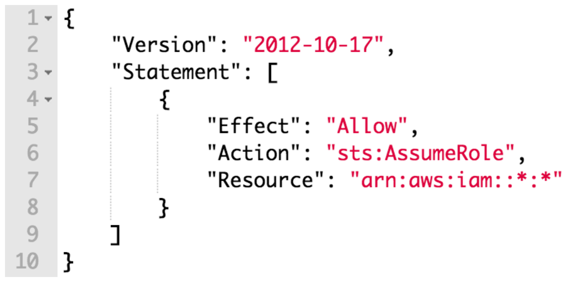
Figure 5.12: Process of Creating a Sip

SIP-requests Handling:

When a representative security administrative users from *uSet* send a sip request to SID-server, SID-manager creates a sip by assigning a non-possessed SID-operational account to the sid, and associates the group of organizations to the sip. Figure 5.12 shows this process. The new sip has two roles created in the account: *SIPadmin* and *SIPmember*. Role *SIPadmin* has a policy specifying permissions which allow security users have limited administrative power to bring and remove member users to and from the sip from its own organization. Role *SIPmember* specifies policies for normal users from organizations to join the sip and have operational permissions.

SID-manager returns a sip account number with roles *SIPadmin* and *SIPmember* for each organization from the community. Security admin users from *uSet* and normal users from organizations then can access to the sip through these two types of roles. We call users who got *SIPadmin* role SIP admin users, and users who got *SIPmember* role member users. Knowing the role names, sip account numbers, security administrative users can access to the sip and assign normal users from their organization account to the sip. Normal users access to the sip in a similar way.

For cyber security collaborations among a community of organizations, they can request multiple sips for collaborations with in a sid from the SID security service. The number of sips depends on how many collaborations are going on among these organizations in the community.



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::*:*"
        }
    ]
}
```

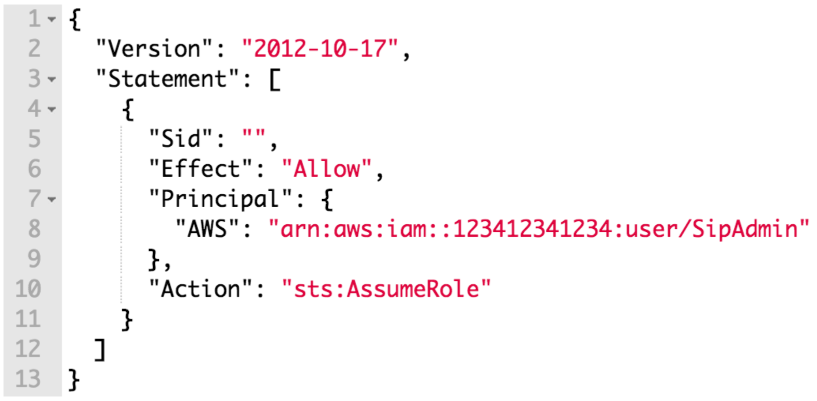
Figure 5.13: AssumeRole

AssumeRole:

Organizations can simply put a policy for their users to be able to use *AssumeRole* action, which gives users ability to access accounts outsides of their home accounts, as well as accounts in a sid. Figure 5.13 shows a sample of a policy attached to a user which allows the user to have access to any other account through *AssumeRole* action. By attached with this policy, users have the potential ability to access accounts in a sid. We say potential ability, because on the other side, the account in the sid also have to have a trust relationship specify who is trust worth to have the access to it. With AssumeRole policy and trust relationship together, a user can have access to resources in another AWS account.

Trust Relationship:

As we just said above, a user can have access to resource in another AWS account only when he has a policy to assume a role in another AWS account, mean time, that account has a trust relationship declaimed to trust this user. Thus, after sid creation and sip creation requests, roles like *CPadmin/SIPadmin* and *CPmember/OPmember/SIPmember* are created. In these roles, a trust relationship defines who from which AWS account can access current account with the role. Figure 5.14 shows a sample of trust relationship, which gives user SipAdmin from account with account number *123412341234* access to current account.



```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123412341234:user/SipAdmin"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Figure 5.14: Trust Relationship in a Role

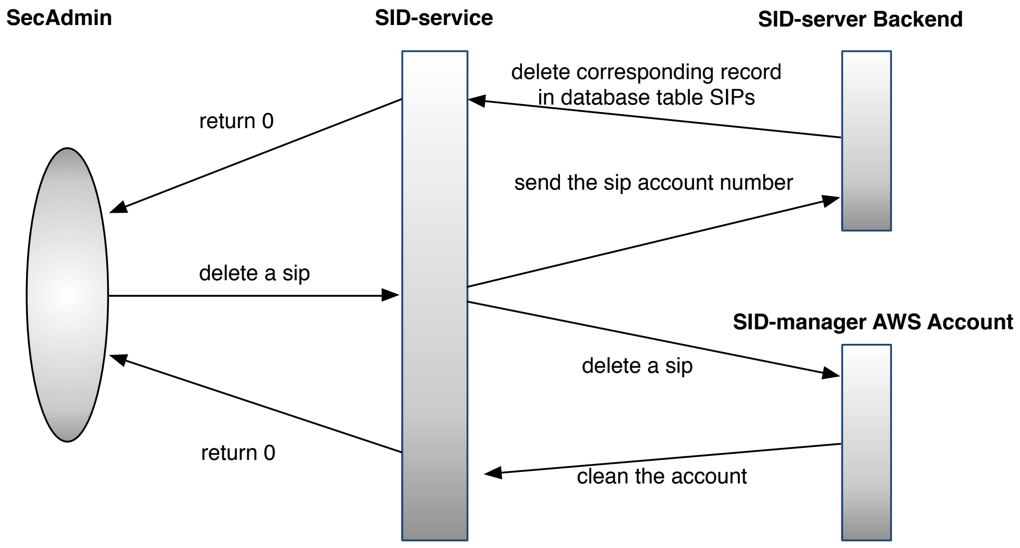


Figure 5.15: Process of Deleting a Sip

Add and Remove Users:

Since trust relationship controls the access of users to a sip account, by updating trust relationship for a role, we can add a user to a sip or remove the user from the sip. Organizations security admin users send add and remove users request to SID-server to perform the actions.

Delete a Sip:

After a collaboration is completed, organizations can request to delete a sip. All the roles which are created for the specific group of organizations will be deleted. All the information and resources

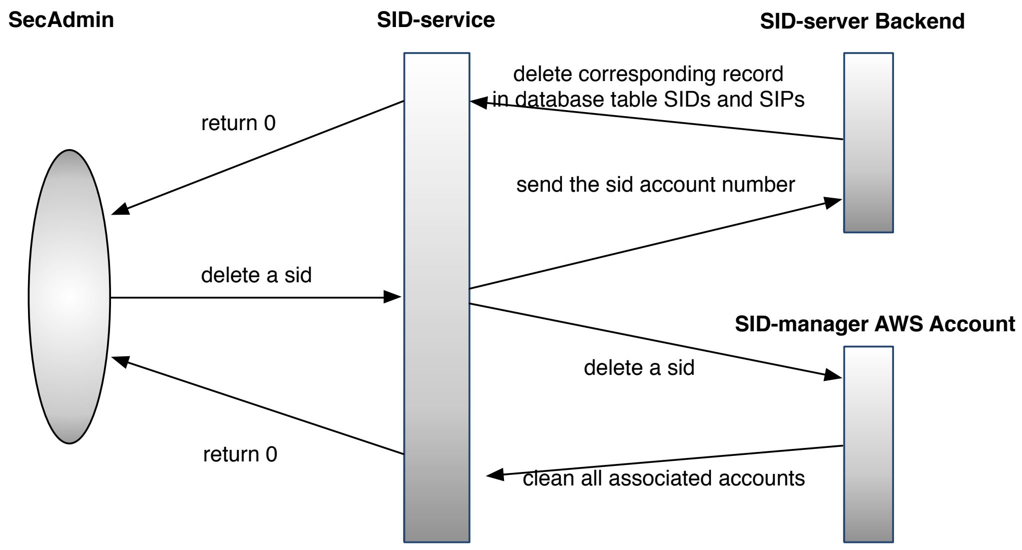

SecAdmin
SID-service
SID-server Backend
delete corresponding record in database table SIDs and SIPs
return 0
send the sid account number
delete a sid
SID-manager AWS Account
delete a sid
return 0
clean all associated accounts

Figure 5.16: Process of Deleting a Sid

that is created during the collaboration will be cleaned up. Any users that are still assigned to the sip will be removed. Figure 5.15 shows the process of deleting a sip. Deleting a sip doesn't mean to delete the sip AWS account, it means to delete all information and resources created during the collaboration in the sip account, making the sip account clean and available for next sip use.

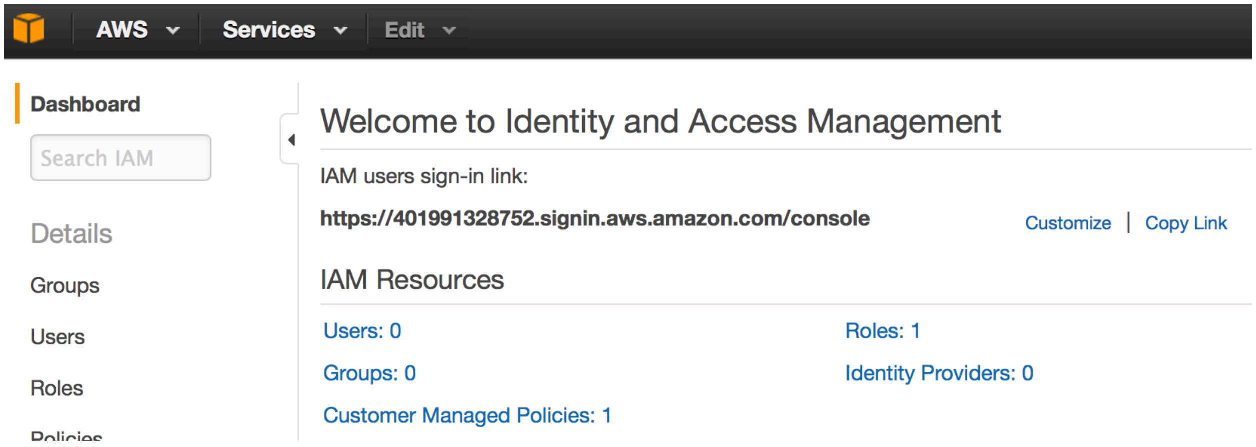
Delete a Sid:

When a sid is deleted, all the projects associated to it will be deleted too, including core project, open project and any sips. With deleting the project, any roles and policies that are created during the sid or sip creation will be removed, as shown in Figure 5.16.

5.3.2 Demonstrations

In this section, demonstrations of screenshots are given to show how SID-server is implemented and presents in AWS cloud platform.

Before a sip is created, any of available AWS operational accounts can be used for creating a sip. We have pre-settings in all operational accounts, with a *SIDmanager* role setting up in each of them, through which the SID-manager manages the operational account. I have some screen



Welcome to Identity and Access Management

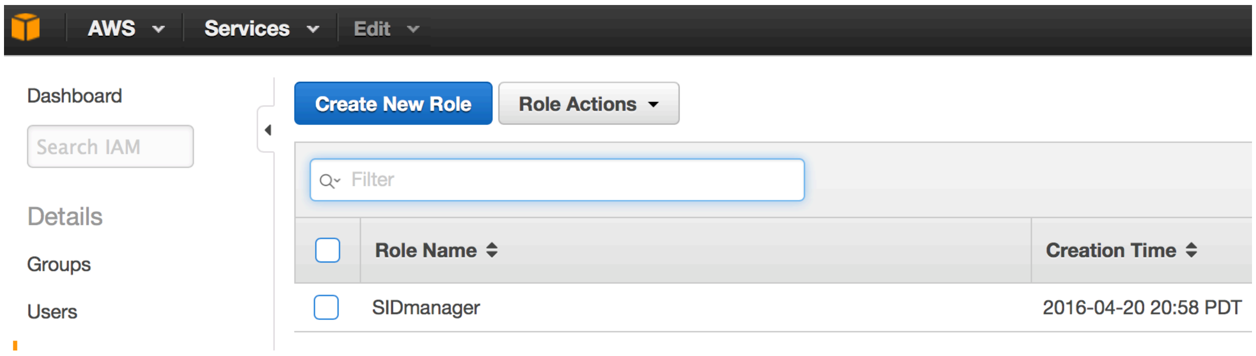
IAM users sign-in link:

https://401991328752.signin.aws.amazon.com/console Customize | Copy Link

IAM Resources

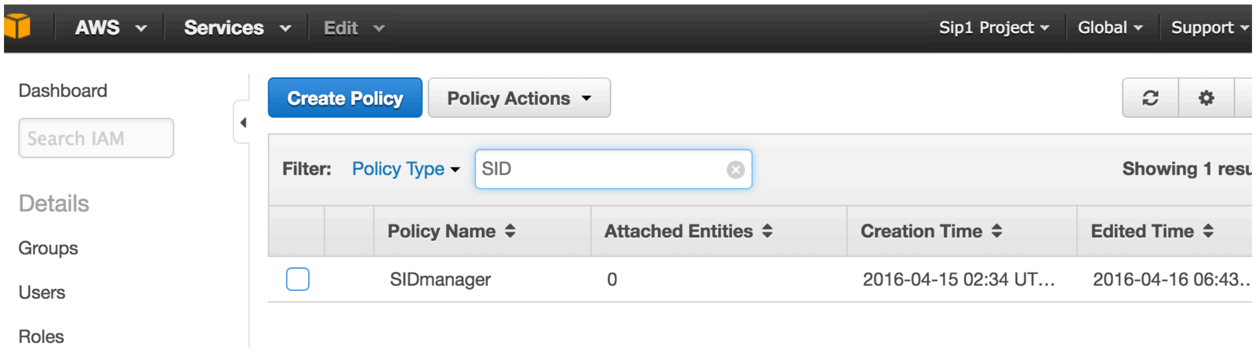
Users: 0	Roles: 1
Groups: 0	Identity Providers: 0
Customer Managed Policies: 1	

Figure 5.17: IAM Dashboard in an Operational AWS Account



	Role Name	Creation Time
☐	SIDmanager	2016-04-20 20:58 PDT

Figure 5.18: IAM Roles in an Operational AWS Account

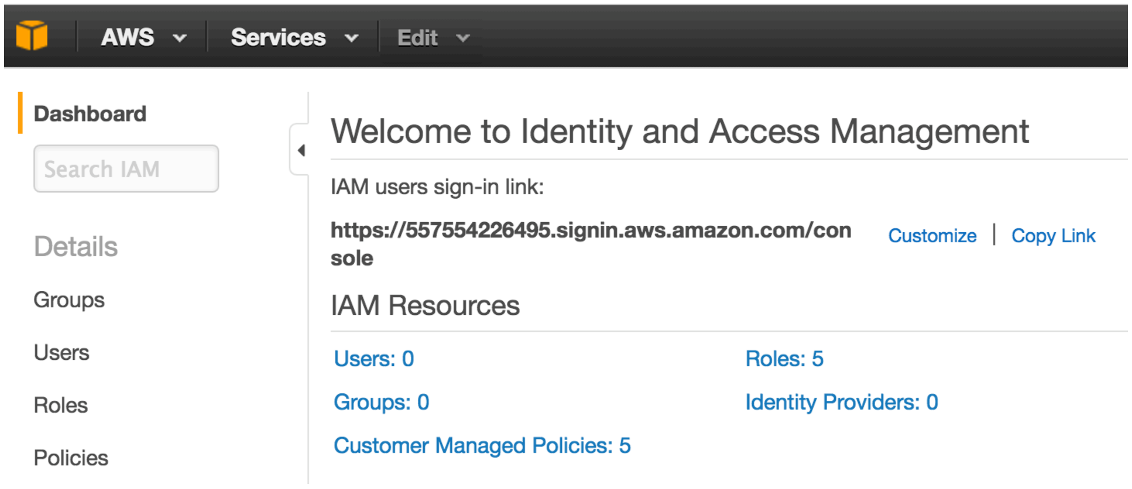


Filter: Policy Type SID Showing 1 resu

	Policy Name	Attached Entities	Creation Time	Edited Time
☐	SIDmanager	0	2016-04-15 02:34 UT…	2016-04-16 06:43..

Figure 5.19: IAM Policies in an Operational AWS Account

shots to show how the IAM panel looks like in an operational AWS account, without a sip, a core project or an open project being created. Figure 5.17 shows how it looks like in IAM panel of an operational AWS account. We can see that there is one role and one customer managed policy already existing in the account, which are *SIDmanager* role and *SIDmanager* policy. Figure 5.18 and Figure 5.19 further show how it looks like in IAM Roles and Policies panel of an operational



Welcome to Identity and Access Management

IAM users sign-in link:

https://557554226495.signin.aws.amazon.com/console

Customize | Copy Link

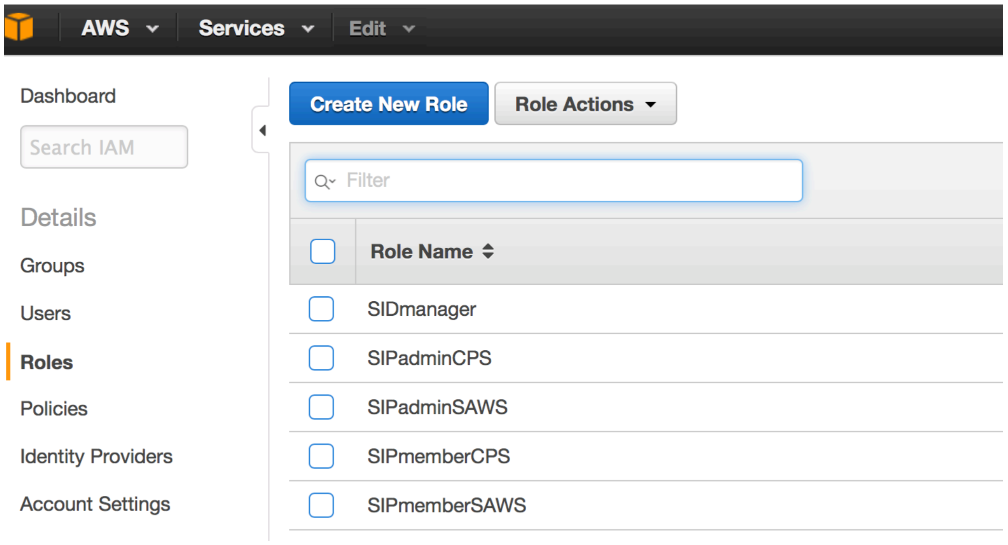
IAM Resources

Users: 0 Roles: 5

Groups: 0 Identity Providers: 0

Customer Managed Policies: 5

Figure 5.20: IAM Dashboard in a Taken Operational AWS Account



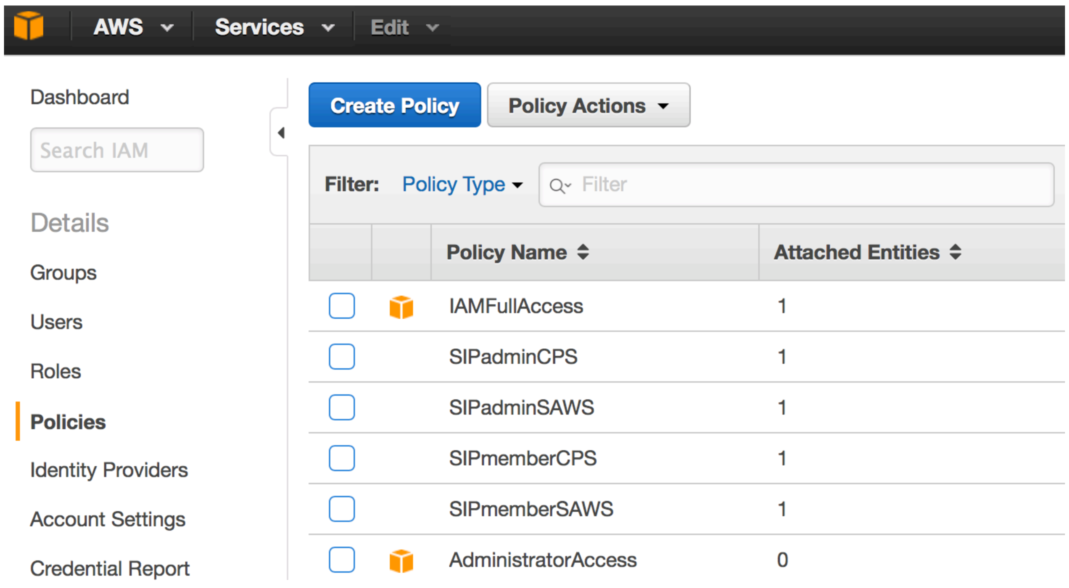
Role Name
SIDmanager
SIPadminCPS
SIPadminSAWS
SIPmemberCPS
SIPmemberSAWS

Figure 5.21: IAM Roles in a Taken Operational AWS Account

AWS account separately. In Figure 5.18, we can see there is a existing role *SIDmanager* and in Figure 5.19, we see an existing policy *SIDmanager*, which is for role *SIDmanager*.

After a sip is created, the taken AWS operational accounts will have corresponding roles and polices created for SID functionalities. Here we assume two organizations are going to collaborate and created a sid for the collaboration. Figure 5.20 shows how IAM panel presents after a sip is created. We can see that, now it has 5 roles and 5 customer managed policies.

Figure 5.21 further shows how it looks like in IAM Roles panel of a taken operational AWS account. We see there are two roles for each organization, one for admin user and one for member



		Policy Name	Attached Entities
☐	📦	IAMFullAccess	1
☐		SIPadminCPS	1
☐		SIPadminSAWS	1
☐		SIPmemberCPS	1
☐		SIPmemberSAWS	1
☐	📦	AdministratorAccess	0

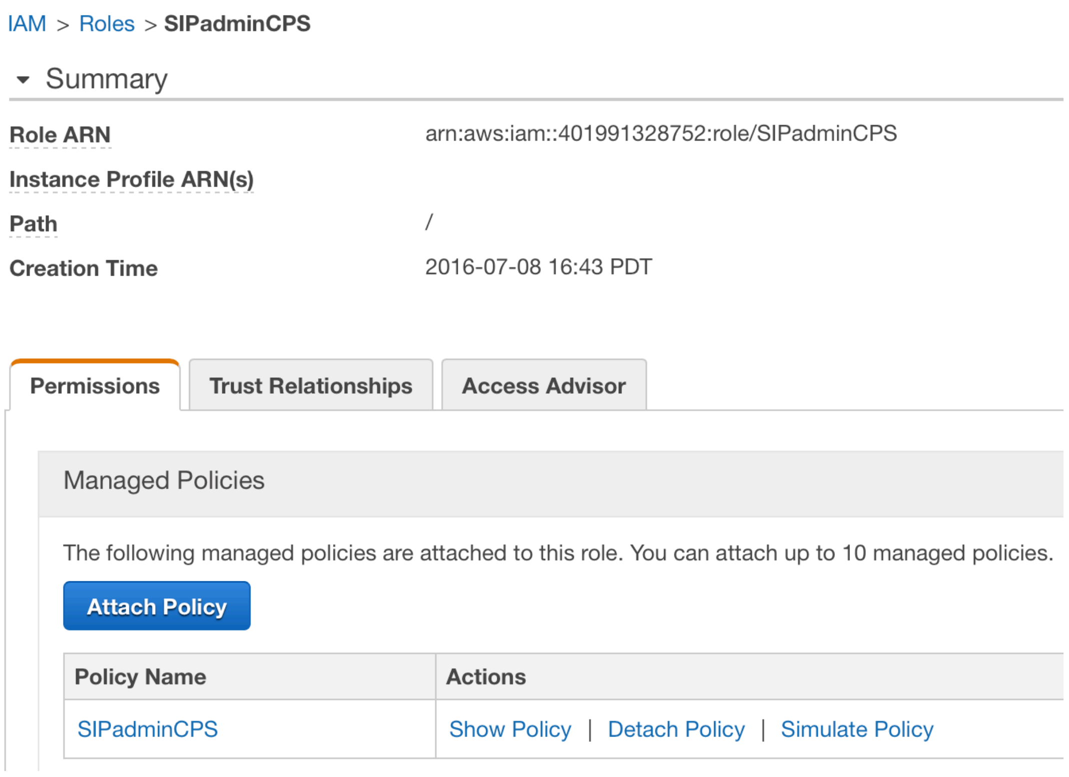
Figure 5.22: IAM Policies in a Taken Operational AWS Account

users. For example, security admin user from organization CPS will use role *SIPadminCPS* to access the sip, while security admin user from organization SAWS will use role *SIPadminSAWS* to access the sip. Similar, normal users from CPS will use role *SIPmemberCPS* to access the sip, while normal users from organization SAWS will use role *SIPmemberSAWS* to access the sip.

Figure 5.22 shows how it looks like in IAM Policies panel of a taken operational AWS account. Policies for each roles in the sip are specified only for that role. For example, policy *SIPadminCPS* is attached to role *SIPadminCPS*, which means organization CPS security admin user has a role of *SIPadminCPS* with policy *SIPadminCPS* set. Figure 5.23 shows role *SIPadminCPS* with attached policy *SIPadminCPS*.

Also, in order for users from organizations to access to the sip, each role has a trust relationship setting up for the corresponding user from a certain organizations. For example, in role *SIPadminCPS*, the trust relationship set the trust entity for the sip is organization CPS (in this case, CPS has a AWS account number 934324332443), as shown in Figure 5.24.

For member roles in a sip, the way for organizations to add more normal users to a sip is to add trust relationships in the corresponding member role. For example, organization CPS has a member role in a sip as of *SIPmemberCPS*, Figure 5.25 shows users with trust relationships. In


IAM > Roles > SIPadminCPS

Summary

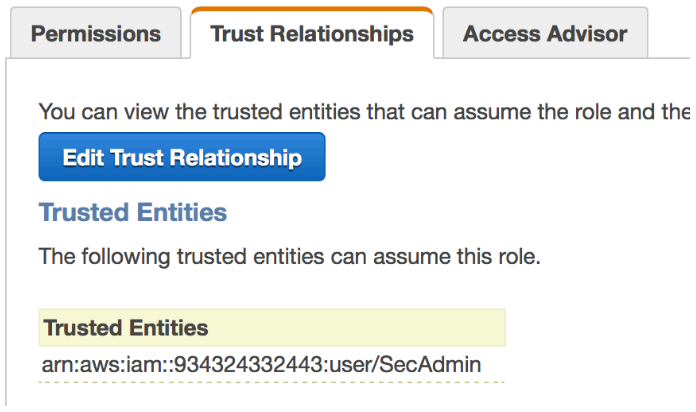
Role ARN	arn:aws:iam::401991328752:role/SIPadminCPS
Instance Profile ARN(s)	
Path	/
Creation Time	2016-07-08 16:43 PDT

Managed Policies

The following managed policies are attached to this role. You can attach up to 10 managed policies.

Policy Name	Actions
SIPadminCPS	Show Policy | Detach Policy | Simulate Policy

Figure 5.23: IAM Roles with Policies Attached in a Taken Operational AWS Account



You can view the trusted entities that can assume the role and the

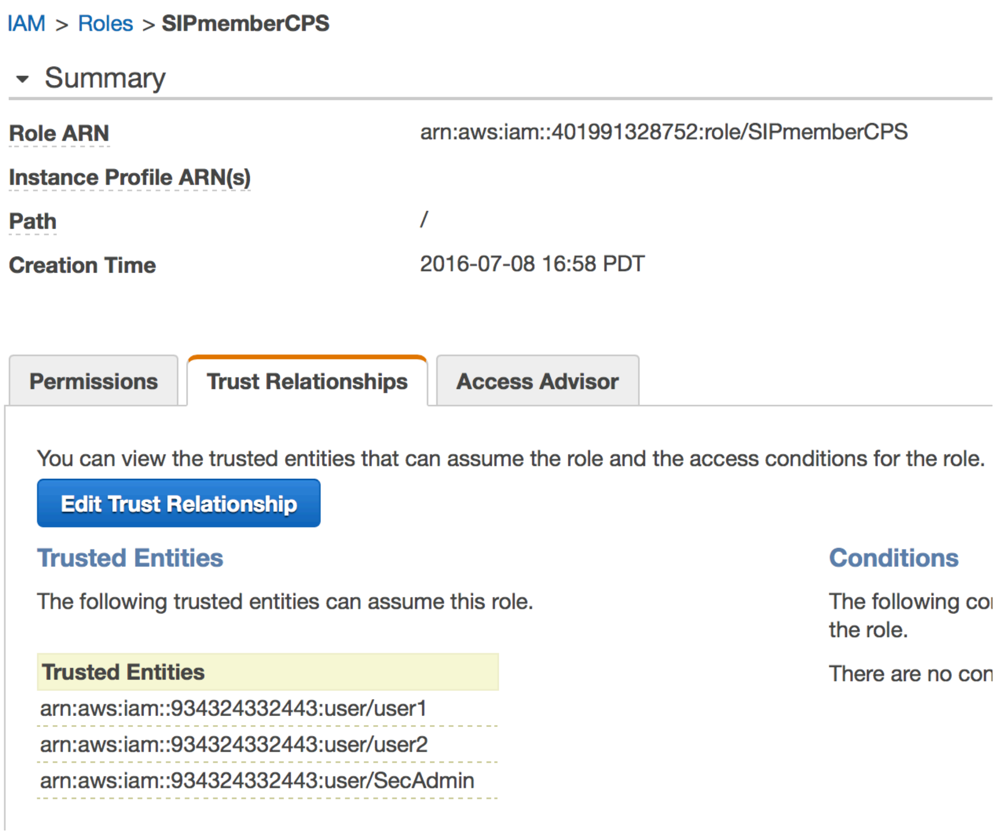
Trusted Entities

The following trusted entities can assume this role.

Trusted Entities
arn:aws:iam::934324332443:user/SecAdmin

Figure 5.24: Trust Relationship for Role *SIPadminCPS*

the figure, each user from organization CPS (AWS account number 934324332443) has a piece of trust relationship policy specified in role *SIPmemberCPS*. Organization security admin users add and remove their normal users to and from a sip by adding and remove a piece of relationship policy attached to the role *SIPmemberCPS*.


IAM > Roles > SIPmemberCPS

Summary

Role ARN	arn:aws:iam::401991328752:role/SIPmemberCPS
Instance Profile ARN(s)	
Path	/
Creation Time	2016-07-08 16:58 PDT

You can view the trusted entities that can assume the role and the access conditions for the role.

Trusted Entities

The following trusted entities can assume this role.

Trusted Entities
arn:aws:iam::934324332443:user/user1
arn:aws:iam::934324332443:user/user2
arn:aws:iam::934324332443:user/SecAdmin

Conditions

The following co
the role.

There are no con

Figure 5.25: Trust Relationship for Role *SIPmemberCPS*

When a sip is deleted, all the roles and policies will be deleted, left the AWS account the same as any other available AWS accounts for next use of a sip creation.

Chapter 6: SID-MODEL IN AZURE CLOUD IAAS

Part of content from this chapter has been published in paper [20].

Microsoft Azure is one of the dominant cloud IaaS platforms for enterprises. Similar to AWS and OpenStack, as an IaaS provider, Azure's core features include compute, storage, database and networking. Azure divides the basic features into four categories: build infrastructure, develop modern applications, gain insights from data, and manage identity and access. You can easily get a virtual machine for development, as well as storage, database and networking.

In Azure, any user has the capability to create an Azure account. The user who creates an Azure account will be the owner and super administrative user of that account. Meanwhile, he/she can access to resources in other accounts with proper roles. Unlike AWS, local users created in an Azure Active Directory can create their own Azure account which is completely isolated from the parent account.

Azure has two main components to manage users' access to resources in the cloud: Azure Active Directory (AAD) and Subscriptions (Sub). In order for a user to use resources in Azure, the user has be assigned to a subscription. Azure Active Directory helps to manage users, including both local Azure AD users and other valid Microsoft users.

6.1 Azure Access Control Model

Azure offers a form of role-based access control, wherein permissions are defined over cloud resources within a role in resource groups. Roles can then be assigned to users. Roles are predefined in Azure. In this part, we introduce Microsoft Azure cloud platform and present Azure Access Control model. Figure 6.1 depicts the Azure Access Control model. In this and other figures, the arrows denote binary relations with the single arrowhead indicating one side and double arrowheads many sides.

Azure Access Control (Azure-AC) model has fourteen components: Accounts (A), Azure Active Directory (AAD), Subscriptions (Sub), Azure Active Directory Roles (AADR), Azure Active

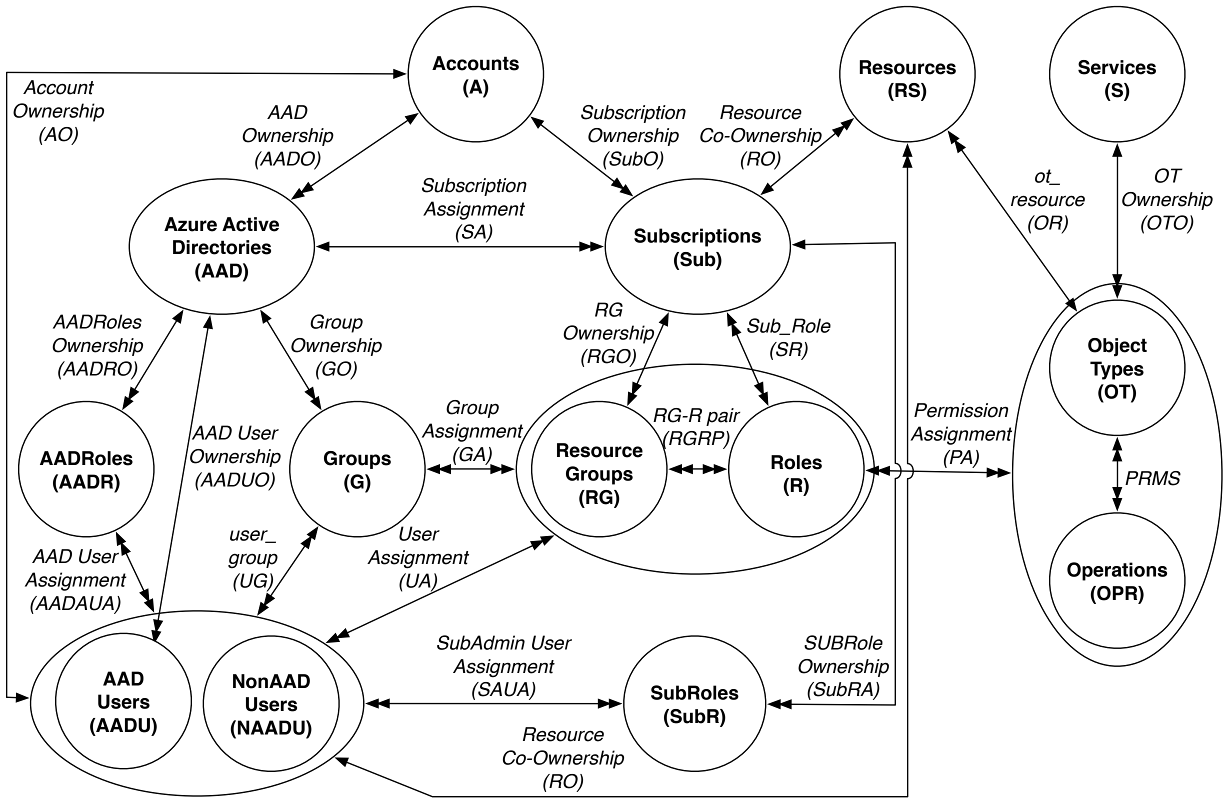


Figure 6.1: Azure Access Control (Azure-AC) Model

Directory Users (AADU), Non Azure Active Directory Users (NAADU), Groups (G), Resource Groups (RG), Roles (R), Subscription Roles (SubR), Resources (R), Services (S), Object Types (OT), and Operations (OPR). We also introduce other entities policies, which are implicitly included in the model.

Accounts: To have its own public cloud resources, an organization need to open an Azure account. An Azure account allows an organization to own specific (virtual) cloud resources that can be accessed through Azure cloud services.

Azure Active Directory: Azure Active Directory (Azure AD) is Microsoft's multi-tenant cloud based directory and identity management service. It provides a full suite of identity management capabilities including multi-factor authentication, device registration, self-service password management, privileged account management, role based access control, security monitoring and so on. Azure AD also provides single sign-on (SSO) access to cloud SaaS Applications. It can also integrate with other identity management solutions used in industry.

Subscriptions: Users have access to cloud resources via subscriptions. Subscriptions are the units of usage and billing for cloud resources. In order to have access to cloud resources, users must be assigned to at least one subscription.

Azure Active Directory Roles: Azure AD roles allow one to manage the directory and identity-related features. Azure AD has a set of administrative roles, including billing administrator, global administrator, password administrator, service administrator and user administrator. Each of these administrative roles are designed for a different specific administrative purpose. It also has a normal user role, which has no administrative power.

Subscription Roles: Subscription roles are a separate role set from Azure Active Directory Roles. Subscription Roles are administrative roles which give users permissions to manage cloud resources via a subscription. Subscription roles include service administrator and co-administrators, both of which can give users access to cloud services. The services administrator and co-administrators can be either Microsoft accounts or Azure AD users. A service administrator cannot be a local Azure AD user from the same Azure AD assigned to that subscription.

Azure Active Directory Users and Non-Azure Active Directory Users: Users represent individuals who can be authenticated by Azure and authorized to access cloud resources through an Azure account. Users from both Microsoft accounts and partner organization accounts are allowed to access to cloud resources in Azure. Azure Active Directory users are users created in Azure AD. They can be administrative users of the directory or normal users. Non-Azure AD users refer to users not from the local Azure AD, but from partner organizations and other Microsoft users.

Groups: A group is simply a set of users and it can include both Azure AD users and Non-Azure AD users. Groups belong to an Azure AD account. The existence of groups is to conveniently mange multiple users as a single unit. Each policy attached to a group will apply to all group members. For simplicity, we will ignore discussion group concept in the following part of the paper, since a set of users can represents a group.

Resource Groups: Resource groups is a logical resource container which allows customers to add various cloud resources like database, virtual machine etc. Resource groups provides a way to monitor and control users' access to collections of cloud resources.

Roles: Users are assigned to a resource group with roles to get permissions to access to cloud resources. Roles allow users to have permissions to access cloud resources, for instance virtual machines (VMs), storage, networking and etc. Roles could be different collections of meta permissions like read and write toward a specific piece of resource. Roles are only able to be assigned to users inside a resource group.

Resources: Resources refer to cloud assets which can be owned by users. Cloud assets are cloud resources such as virtual machines, databases, storages, etc. Since the only way for users to access resources is through subscriptions, we require that the subscription has ownership over the resources too.

Services: Services refer to cloud services Azure provides to its customers. Cloud Service Provider (CSP) leases cloud resources to its customers in terms of services. Azure provides customers with services such as compute, storage, networking, administration, and database.

Object Types and Operations: An object type represents a specific type of object. From the CSP's viewpoint, objects are more like services. We define object types as particular service types the cloud provides. For instance, with the Compute service, the object type is a virtual machine; with the storage service, the object type is a storage container; etc.

Policy: In Azure, users' permissions over services and resources are defined in policy files. Policy files can be attached to a user, a group, a role or a specific cloud resource. By attaching a policy to a user, a group or a role, users gain permissions to corresponding cloud resources. The policy defines the actions the user will perform and cloud resources on which the actions will function. Multiple permissions can be defined in one policy file. Multiple policy files can be attached to one entity. Azure achieves permission assignment in a virtual manner via the policies attached to various relevant entities.

With the concepts described above, we formalize Azure-AC model as follows.

Definition 1. Azure-AC model has the following components.

- A, AAD, Sub, RG, R, AADR, SubR, AADU, NAADU, G, RS, S, OT and OPR are finite sets of existing accounts, Azure active directories, subscriptions, resource groups, roles, Azure AD roles, Subscription roles, Azure AD users, Non-Azure AD users, groups, resources, services, object types and operations respectively in the Azure cloud system.

- Account Ownership (AO) : is a function AO : A $\rightarrow$ U, mapping an account to its owning user.

- AAD Ownership (AADO) : is a function AADO : AAD $\rightarrow$ A, mapping an Azure AD to its owning account. Equivalently viewed as a many-to-one relation AADO $\subseteq$ AAD $\times$ A.

- Subscription Ownership (SubO) : is a function SubO : Sub $\rightarrow$ A, mapping a subscription to its owning account. Equivalently viewed as a many-to-one relation SubO $\subseteq$ Sub $\times$ A.

- Resource Group Ownership (RGO) : is a function RGO : U $\rightarrow$ Sub, mapping a resource group to its owning subscription. Equivalently viewed as a many-to-one relation GRO $\subseteq$ RG $\times$ Sub.

- AAD User Ownership (AADUO) : is a function AADUO : AADU $\rightarrow$ AAD, mapping a user to its owning Azure AD. Equivalently viewed as a many-to-one relation AADUO $\subseteq$ AADU $\times$ AAD.

- Group Ownership (GO) : is a function GO : G $\rightarrow$ AAD, mapping a group to its owning Azure AD. Equivalently viewed as a many-to-one relation GO $\subseteq$ G $\times$ AAD.

- Azure AD Roles Ownership (AADRO) : is a function AADRO : AADR $\rightarrow$ AAD, mapping a Azure AD role to its owning Azure AD. Equivalently viewed as a many-to-one relation AADRO $\subseteq$ U $\times$ A.

- Resource Co-Ownership (RSO) : is a function RSO : RS $\rightarrow$ Sub $\cup$ RS $\rightarrow$ (AADU $\cup$ NAAUD), mapping a piece of resource to its owning subscription and user. Equivalently viewed as a many-to-one relation RSO $\subseteq$ RS $\times$ Sub $\cup$ RS $\times$ (AADU $\cup$ NAAUD).

- Object Type Owner (OTO) : is a function OTO : OT $\rightarrow$ S, mapping an object type to its owning service. Equivalently viewed as a many-to-one relation OTO $\subseteq$ OT $\times$ S.

- Resource Group Role pair (RGRP) : is a function RGRP : RG $\rightarrow$ R, mapping a resource group to a role. Equivalently viewed as a many-to-many relation RGRP $\subseteq$ GR $\times$ R.

- Subscription Assignment (SA) : is a function SA : Sub $\rightarrow$ AAD, mapping subscriptions to its associated Azure active directory. Equivalently viewed as a many-to-one relation SubA $\subseteq$ Sub $\times$ AAD.

- Subscription Roles Assignment (SubRA) : is a function SubRA : SubR $\rightarrow$ Sub, mapping subscription roles to subscriptions. Equivalently viewed as a many-to-many relation SubRA $\subseteq$ Sub $\times$ SubR.

- AAD User Assignment (AADUA) : is a function AADUA : (AADU $\cup$ NonAADU) $\rightarrow$ AADR, mapping users to Azure activedirectory. Equivalently viewed as a many-to-many relation AADUA $\subseteq$ (AADU $\cup$ NonAADU) $\times$ AADR.

- SubAdmin User Assignment (SAUA) : is a function SAUA : (AADU $\cup$ NonAADU) $\rightarrow$ SubR, mapping users to subscription roles. Equivalently viewed as a many-to-many relation SAUA $\subseteq$ (AADU $\cup$ NonAADU) $\times$ SubR.

- User Assignment (UA) : is a function UA : U $\rightarrow$ RGRP, mapping users to resource group-role pairs. Equivalently viewed as a many-to-many relation UA $\subseteq$ U $\times$ RGRP.

- Group Assignment (GA) : is a function GA : G $\rightarrow$ RGRP, mapping groups to resource group-role pairs. Equivalently viewed as a many-to-many relation UA $\subseteq$ G $\times$ RGRP.

- Permission Assignment (PA) : is a function PA : PERMS $\rightarrow$ R, mapping permissions to roles. Equivalently viewed as a many-to-many relation PA $\subseteq$ PERMS $\times$ R.

- user_group (UG) : is a function UG : U $\rightarrow$ G, mapping users to groups where the user and group are owned by the same account. Equivalently viewed as a many-to-many relation UG $\subseteq$ U $\times$ G.

- ot_resource (OR) : is a function OR : OT $\rightarrow$ RS, mapping object types to resources. Equivalently viewed as a one-to-many relation OR $\subseteq$ OT $\times$ RS.

- PRMS = OT $\times$ OPR, is the set of permissions.

6.2 Azure-AC Model with SID extension (Azure-AC-SID)

In this section, we present an access control model for Azure with the Secure Isolated Domain extension (Azure-AC-SID). We extend the Azure-AC-SID model from Azure-AC model to include SID-service functionality [18]. We present the Azure-AC-SID model so as to cover only the additional components added to the Azure-AC model. Figure 6.2 shows the Azure-AC-SID model, where we ignore groups for simplicity. In the rest of the paper, group is used to represent a group of organizations, rather than the groups component of Azure-AC model. In our discussion, we assume that a user belongs to only one organization in cloud. For simplicity, we also assume one organization has only one Azure account.

Azure Active Directory and subscriptions give a great convenience to design the Azure-AC-SID model. Each sid is associated with one Azure AD and one subscription to manage all the users and cloud resources. Resources groups then are created in the subscription as a core project, an open project and sips. We constrain cloud services to only include Azure compute service and Azure object storage service. The corresponding object types are Virtual Machines (VM), Storage Containers (SC) and Storage Container Objects (CO).

Microsoft Azure has resource groups as basic resource containers, which are isolated with each other. Above resource groups, there are subscriptions which act as administrative boundary of multiple basic resource containers. An Azure cloud account can have multiple subscriptions to manage its resources. We use resource groups to realize projects while subscriptions to realize secure isolated domains.

In the following part, we present Azure-AC-SID model. The additional components included in Azure-AC-SID model are: Secure Isolated Domain (SID), Secure Isolated Project (SIP), Expert Users (EU), Users(U), Core Project (CP), and Open Project (OP). These are described below.

We have introduce all these concepts in previous chapters. In Azure, **Secure Isolated Domains** is carried by a subscription, holding security information and resources for cross-organizational security collaborations. **Secure Isolated Project** are resources groups inside the sid subscription, which collect, store and analyze cyber security information for specific cyber incidents. **Core Project** and **Open Project** are two other resources groups inside the sid subscription. **Expert Users** are external non-organizational professionals. They don't belong to the group of organizations. **Users** include both Azure AD users and Non-Azure AD users, which refer to either Microsoft users or partner organization users. We use one entity *Users* to represent all users that are allowed to access cloud resources, since from the stand point of SID-model functionality, as long as the user is associated to the organization's Azure AD, it does not care where the users

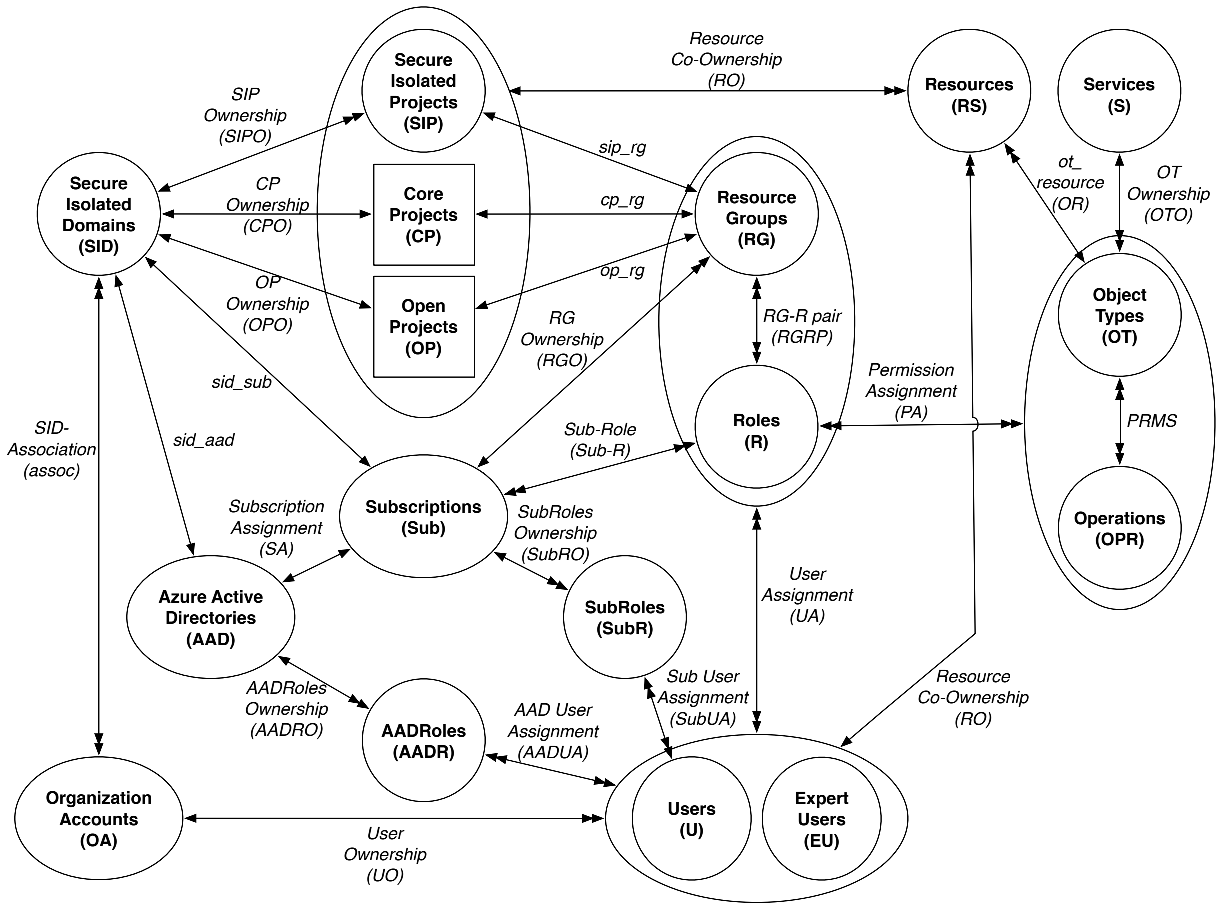


Figure 6.2: Azure Access Control Model with SID Extension (Azure-AC-SID)

come from. **Organization Accounts** represent organizations in the community. They could be either Azure AD accounts or organizations enterprise accounts which are identified by Azure AD. Organization accounts allows organizations to own specific amount of (virtual) cloud resources.

In the following, we give formalization of concepts introduced above, as well as the relation among them.

Definition 2. Azure-AC-SID model has the following components in addition to Azure-AC model.

- SID, SIP, CP, OP, EU and U are finite sets of Secure Isolated Domains, Secure Isolated Projects, Core Projects, Open Projects, Expert Users and Users.
- Virtual Machines (VM), Storage Containers (SC) and Storage Container Objects (CO) are object types respectively for compute service and object storage service in Azure cloud platform.

- Core Project Ownership (CPO) : is a function CPO : CP $\rightarrow$ SID, mapping a single core project to its owning sid. Equivalently viewed as a one-to-one relation CPO $\subseteq$ CP $\times$ SID.

- Open Project Ownership (OPO) : is a function OPO : OP $\rightarrow$ SID, mapping a single open project to its owning sid. Equivalently viewed as a one-to-one relation OPO $\subseteq$ OP $\times$ SID.

- Secure Isolated Project Ownership (SIPO) : is a function SIPO : SIP $\rightarrow$ SID, mapping a single secure isolated project to its owning sid. Equivalently viewed as a many-to-one relation SIPO $\subseteq$ SIP $\times$ SID.

- SID association (assoc): is a function assoc : SID $\rightarrow 2^{\mathrm{A}}$, mapping a SID to all its member organization accounts.

- User Ownership (UO) : is a function UO : U $\rightarrow$ OA, mapping a user to its owning organization account. Equivalently viewed as a many-to-one relation UO $\subseteq$ U $\times$ OA.

- Subscription Assignment (SubA) : is a function SubA : Sub $\rightarrow$ AAD, mapping a single subscription to a single Azure active directory.

- sid_sub $\subseteq$ SID $\times$ Sub, is a one-to-one relation mapping a single sid to a single subscription.

- sid_aad $\subseteq$ SID $\times$ AAD, is a one-to-one relation mapping a single sid to a single Azure active directory.

- cp_rg $\subseteq$ CP $\times$ RG, is a one-to-one relation mapping a single core project to a single resource group.

- op_rg $\subseteq$ OP $\times$ RG, is a one-to-one relation mapping a single open project to a single resource group.

- sip_rg $\subseteq$ SIP $\times$ RG, is a one-to-one relation mapping a single sip to a single resource group.

- p_rg $\subseteq$ P $\times$ RG, is a one-to-one relation mapping a single project (core project, open project or a sip in a sid) to a single resource group.

6.2.1 Administrative Model

To make role assignment simple and clear, we constrain roles in two types: administrative roles and member roles, which separately denotes the permission of being able to manage users and permissions only for accessing cloud resources. We use one admin role *SIDadmin* to represent all admin permissions a user can get from Azure AD and subscriptions. We use one member role *SIDmember* to represent all normal roles a user can get in a resource group. Admin users have the capability to add and remove other users from their home organizations to a sid subscription, which in terms of resource groups viz., core project resource group or a sip resource group. Member users can be added/removed from/to a project resource group inside a sid subscription. Member users are those who have access to the cloud services and resources, for example, creating or deleting a virtual machine.

The administrative aspects of AzureAC-SID model are discussed informally below. A formal specification is given in Table 6.1 and Table 6.2.

Create a sid: Remind again that the creation of sid is based on agreement among the community of organizations. After all the organizations in the community have agreed, one security admin from an organization representing the group of organizations request the creation of a sid with parameters including all the security admin users, each representing one organization in the community. *uSet* denotes a fixed group of security admin users from all organizations of the community, with one admin user for each organization. When a sid is created, the security admin user who issues sid creation command along with all other security admin users from *uSet* will become the limited administrative users of the sid, in which each organization in the community has equal limited administrative power.

In Azure, in order for a user to access resource to a subscription, the user need to be assigned to both the subscription and associated Azure active diretory. Since each projects in a sid is realized by Azure subscription in a Azure management account, admin users from *uSet* will be assigned with co-admin to a sid subscription and admin roles in Azure active diretory which is associated to that sid subscription. After the core project and open project resource groups are created, admin users from *w*ill be assigned to the resource groups with *Owner* roles. All of these role assignment together give an admin user from *uSet* an administrative permission over the sid.

Table 6.1: Azure-AC-SID Administrative Model

Operation	Authorization Requirement	Update

SidCreate(adminu, uSet, sid)
/ An admin user representing uSet creates a sid */*
	adminu $\in$ *uSet* $\wedge$ adminu $\in$ U $\wedge$ sid $\notin$ SID	SID$'$ = SID $\cup$ {sid};
		assoc(sid) = $\bigcup_{adminu \in uSet}$ UO(adminu);
		CP$'$ = CP $\cup$ {cp};
		CPO(cp) = sid;
		OP$'$ = OP $\cup$ {op};
		OPO(op) = sid;
		Sub$'$ = Sub $\cup$ {sid_sub(sid)};
		AAD$'$ = AAD $\cup$ {aad};
		SA(sid_sub(sid)) = aad;
		RG$'$ = RG $\cup$ {cp_rg(cp)} $\cup$ {op_rg(op)};
		RGO(cp_rg(cp)) = sid_sub(sid);
		RGO(op_rg(op)) = sid_sub(sid);
		AADUA$'$ = AADUA $\cup$ {(uSet, *admin*)};
		SubUA$'$ = SubUA $\cup$ {(uSet, *co-admin*)};
		UA$'$ = UA $\cup$ {(uSet, cp_rg(cp), *Owner*), (uSet, op_rg(op), *Owner*)}.

SidDelete(adminu, uSet, sid)
/ An admin user representing uSet deletes the sid*/*
	adminu $\in$ *uSet* $\wedge$ adminu $\in$ U $\wedge$ assoc(sid) = $\bigcup_{adminu \in uSet}$ UO(adminu) $\wedge$ sid $\in$ SID	SID$'$ = SID - {sid};
		assoc(sid) = NULL;
		CP$'$ = CP - {cp};
		CPO(cp) = NULL;
		OP$'$ = OP - {op};
		OPO(op) = NULL;
		Sub$'$ = Sub - {sid_sub(sid)};
		AAD$'$ = AAD - {aad};
		SA(sid_sub(sid)) = NULL;
		RG$'$ = RG - {cp_rg(cp)} - {op_rg(op)};
		RGO(cp_rg(cp)) = NULL;
		RGO(op_rg(op)) = NULL;
		AADUA$'$ = AADUA - {(uSet, *admin*)};
		SubUA$'$ = SubUA - {(uSet, *co-admin*)};
		UA$'$ = UA - {(uSet, cp_rg(cp), *Owner*), (uSet, op_rg(op), *Owner*)};
		$\forall$u $\in$ U.(UA$'$ = UA - {(u, cp_rg(cp), *Contributer*), (u, op_rg(op), *Contributer*)});
		if $\exists$(u, sip_rg(sip), r) $\in$ ((U $\cup$ EU), RG, (*Owner* $\vee$ *Contributer*)).({(u, sip_rg(sip), r)} $\in$ UA $\wedge$ RGO(sip_rg(sip)) $\in$ = sid_sub(sid)), then UA$'$ = UA - {(u, sip_rg(sip), r)} $\wedge$ SIP$'$ = SIP - {sip}.

Table 6.2: Azure-AC-SID Administrative Model (continued)

Operation	Authorization Requirement	Update
SipCreate(adminu, sip, sid) */* An admin user representing uSet creates a sip */*	adminu $\in$ *uSet* $\wedge$ adminu $\in$ U $\wedge$ UO(adminu) $\in$ assoc(sid) $\wedge$ sip $\notin$ SIP $\wedge$ (adminu, *admin*) $\in$ AADUA $\wedge$ (adminu, *co-admin*) $\in$ SubUA	SIP′ = SIP $\cup$ {sip}; SIPO(sip) = sid; RG′ = RG $\cup$ {sip_rg(sip)}; RGO(sip_rg(sip)) = sid_sub(sid); UA′ = UA $\cup$ {(uSet, sip_rg(sip), *Owner*)}.
SipDelete(adminu, sip, sid) */* An admin user representing uSet deletes a sip*/*	adminu $\in$ *uSet* $\wedge$ adminu $\in$ U $\wedge$ UO(adminu) $\in$ assoc(sid) $\wedge$ SIPO(sip) = sid $\wedge$ sip $\in$ SIP $\wedge$ (adminu, *admin*) $\in$ AADUA $\wedge$ (adminu, *co-admin*) $\in$ SubUA	SIP′ = SIP - {sip}; SIPO(sip) = NULL; RG′ = RG - {sip_rg(sip)}; RGO(sip_rg(sip)) = NULL; UA′ = UA - {(uSet, sip_rg(sip), *Owner*)}.
UserAdd(adminu, u, p, sid) */* Admin users add a user from his home domain to a cp, op or sip */*	adminu $\in$ U $\wedge$ (adminu, *admin*) $\in$ AADUA $\wedge$ (adminu, *co-admin*) $\in$ SubUA $\wedge$ (adminu, p_rg(p), *Owner*) $\in$ UA $\wedge$ u $\in$ U $\wedge$ UO(u) = UO(adminu) $\wedge$ p $\in$ (CP $\cup$ OP $\cup$ SIP) $\wedge$ (CPO(p) = sid $\vee$ OPO(p) = sid $\vee$ SIP(p) = sid)	AADUA′ = AADUA $\cup$ {(u, *user*)}; UA′ = UA $\cup$ {(u, p_rg(p), *Contributer*)}.
UserRemove(adminu, u, p, sid) */* Admin users remove a user from a cp, op or sip */*	adminu $\in$ U $\wedge$ (adminu, *admin*) $\in$ AADUA $\wedge$ (adminu, *co-admin*) $\in$ SubUA $\wedge$ (adminu, p_rg(p), *Owner*) $\in$ UA $\wedge$ u $\in$ U $\wedge$ UO(u) = UO(adminu) $\wedge$ p $\in$ (CP $\cup$ OP $\cup$ SIP) $\wedge$ (CPO(p) = sid $\vee$ OPO(p) = sid $\vee$ SIP(p) = sid) $\wedge$ (u, p_rg(p), *Contributer*) $\in$ UA	AADUA′ = AADUA - {(u, *user*)}; UA′ = UA - ({(u, cp_rg(p), *Contributer*)} $\vee$ {(u, op_rg(p), *Contributer*)} $\vee$ {(u, sip_rg(p), *Contributer*)}).
EUserAdd(adminu, eu, p, sid) */* Admin users add an expert user to a cp or sip */*	adminu $\in$ U $\wedge$ (adminu, *admin*) $\in$ AADUA $\wedge$ (adminu, *co-admin*) $\in$ SubUA $\wedge$ (admin, p_rg(p), *Owner*) $\in$ UA $\wedge$ eu $\in$ EU $\wedge$ p $\in$ (CP $\cup$ SIP) $\wedge$ (CPO(p) = sid $\vee$ SIP(p) = sid)	AADUA′ = AADUA $\cup$ {(eu, *user*)}; UA′ = UA $\cup$ {(eu, p_rg(p), *Contributer*)}.
EUserRemove(adminu, eu, p, sid) */* Admin users remove an expert user from a cp or sip */*	adminu $\in$ U $\wedge$ (adminu, *admin*) $\in$ AADUA $\wedge$ (adminu, *co-admin*) $\in$ SubUA $\wedge$ (uSet, sip_rg(sip), *Owner*) $\in$ UA $\wedge$ eu $\in$ EU $\wedge$ p $\in$ (CP $\cup$ SIP) $\wedge$ (CPO(p) = sid $\vee$ SIP(p) = sid) $\wedge$ (eu, p_rg(p), *Contributer*) $\in$ UA	AADUA′ = AADUA - {(eu, *user*)}; UA′ = UA - {(eu, p_rg(p), *Contributer*)}.

Delete a sid: Remind that one admin user from *uSet* representing the group of organizations initilizes the sid delete request. To delete the sid, the admin user need to delete all the projects resource groups inside the sid subscriptions and unassign all the users in these resource groups.

Create a sip: A security admin user representing *uSet* creates a sip resource group in the sid subscription for an cyber collaboration among the community of organizations, assign role *Owner* to all admin users in *uSet*.

Delete a sip: Any security admin user from (*uSet*) can delete a sip resource group in a sid subscription. All information data and resources will be securely deleted from the sip resource group. All users assigned to the sip resource group will be removed from it.

Add/remove a user to/from a core project or sips: Remind that core project and sips admin users are the set of security administrative users (*uSet*) from the community of organizations. These limited administrative users can add/remove users of their organizations to/from the core project and sips. Every time a normal user is brought to a project resource group inside a sid subscription, the normal user need to be added to the associated Azure active directory. Then the user can be assigned to a project resource group with role *Contribution*.

Add/remove a user to an open project: Every user in the collaborative community of organizations is allowed to join the open project. Users in open project have equal but limited permissions. Similar as adding users to core project, users need to both Azure active directory roles and roles in resource group to have permissions to open project. Removing is the opposite as adding action, the user will be unassigned with both roles from active directory and resource group.

Add/remove an expert user to/from a core project or sips: Expert Users are needed when external cyber expertise need to be involved. Adding and removing an expert user in Azure is the same way as adding and removing a normal user in Azure.

6.2.2 Operational Model

In the operational model, we mainly show how and what operations a normal user can issue in the model. Again, for simplicity, we only demonstrate the core operations on virtual machines and storage containers, including creation and delete. *Create* method allows users to create a new instance of virtual machine or a storage container in a core project, open project or a sip. *Delete* method allows users to delete an existing instance of a virtual machine or storage container in a project. In table 6.3, we give the details of operational model.

Table 6.3: Azure-AC-SID Operational Model

Operation	Authorization Requirement	Update
CreateVM(vm, p, u) */* A user creates a vm */*	$vm \notin RS \wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge \exists$ (perms, r) $\in$ PA.(perms = (vm, create) $\wedge$ (p_rg(p), r) $\in$ RGRP $\wedge$ (u, (p_rg(p), r)) $\in$ UA)	RS' = RS $\cup$ {vm}; RSO' = RSO $\cup$ {(vm, (p, u))}; OR(vm) = VM.
DeleteVM(vm, p, u) */* A user deletes a vm */*	$vm \in RS \wedge$ RSO(vm) = {(p, u)} $\wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge \exists$ (perms, r) $\in$ PA.(perms = (vm, delete) $\wedge$ (p_rg(p), r) $\in$ PR $\wedge$ (u, (p_rg(p), r)) $\in$ UA	RS' = RS - {vm}; RSO' = RSO - {(vm, (p, u))}; vm = NULL.
CreateSContainer(sc, p, u) */* A user creates a storage container */*	$sc \notin RS \wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge \exists$ (perms, r) $\in$ PA.(perms = (sc, create) $\wedge$ (p_rg(p), r) $\in$ PR $\wedge$ (u, (p_rg(p), r)) $\in$ UA)	RS' = RS $\cup$ {sc}; RSO' = RSO $\cup$ {(sc, (p, u))}; OR(sc) = SC.
DeleteSContainer(sc, p, u) */* A user deletes a storage container */*	$sc \in RS \wedge$ RSO(sc) = {(p, u)} $\wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge \exists$ (perms, r) $\in$ PA.(perms = (sc, delete) $\wedge$ (p_rg(p), r) $\in$ PR $\wedge$ (u, (p_rg(p), r)) $\in$ UA	RS' = RS - {sc}; RSO' = RSO - {(sc, (p, u))}; sc = NULL.
CreateObject(co, sc, p, u) */* A user creates a storage container object */*	$co \notin RS \wedge sc \in RS \wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge$ RSO(sc) = (p, u) $\wedge \exists$ (perms, r) $\in$ PA.(perms = (co, create) $\wedge$ (p_rg(p), r) $\in$ PR $\wedge$ (u, (p_rg(p), r)) $\in$ UA)	RS' = RS $\cup$ {co}; RSO' = RSO $\cup$ {(co, (p, u))}; OR(co) = CO.
DeleteObject(co, sc, p, u) */* A user delete a storage container object */*	$co \in RS \wedge$ RSO(co) = {(p, u)} $\wedge sc \in RS \wedge p \in (CP \cup OP \cup SIP) \wedge u \in U \wedge$ RSO(sc) = (p, u) $\wedge \exists$ (perms, r) $\in$ PA.(perms = (co, create) $\wedge$ (p_rg(p), r) $\in$ PR $\wedge$ (u, (p_rg(p), r)) $\in$ UA)	RS' = RS - {co}; RSO' = RSO - {(co, (p, u))}; co = NULL.

After a user is assigned to a core project, open project or sip, the user can issue following open projecterations:

CreateVM/DeleteVM: A user can create/delete a virtual machines in a core project, open project or sip resource group, to which the user is assigned.

CreateContainer/DeleteContainer: A user can create/delete a storage container in a core project, open project or sip resource group, to which the user is assigned. A storage container holds container objects.

CreateObject/DeleteObject: A user can create/delete a container object in a storage container in a core project, open project or sip resource group, to which the user is assigned.

6.3 Enforcement

Microsoft Azure has been undated to new releases from time to time. We discuss the enforcement of Azure-AC-SID model on the current Azure release. Azure accounts form the basic resource boundary in the cloud. To sign in to the Azure cloud, a user has to have either a Microsoft account or an Azure AD account which stores the organization accounts information. In this chapter, we uniformly call both of these types of accounts Azure accounts.

For a user to use resources in the Azure cloud, the user has to be assigned to a subscription. A user can be assigned to multiple subscriptions. Each subscription has a trust relationship with one and only one Azure AD, which gives the Azure AD power to authenticate users, services and devices for that subscription. Users have one Azure AD as their home directory to authenticate them, meanwhile they can be guest users in other Azure ADs. One Azure AD can be associated with multiple subscriptions. Subscriptions can change their associations with Azure AD any time they want. When a subscription de-associates with an Azure AD, users will lose their access to resources through the subscription but still exist in that Azure AD.

Inside an Azure account, subscriptions are sub-divisions of cloud resources while they are separated with each other. Resource groups are further sub-divisions of cloud resources under a subscription. Figure 6.3 simply shows the perspective of resources divisions within an Azure account. Azure account owners can create different subscriptions for different management and billing purpose. Azure account owners also assign a service administrator for a subscription, who has full permissions to assign any cloud services and resources to users though the subscription. Each subscription only has one service administrator. Service administrators can further assign co-administrators to the subscription to help assigning cloud services and resources to users.

Azure AD has its own set of administrative roles, including global administrator, billing administrator, password administrator, service administrator and user administrator. For simplicity, we are going to use global administrator role only in our model. Azure AD global administrator can create, edit and delete users and manage user licenses. An admin user has to have both Azure AD global administrator role and subscription administrator to be able to grant a user the access to

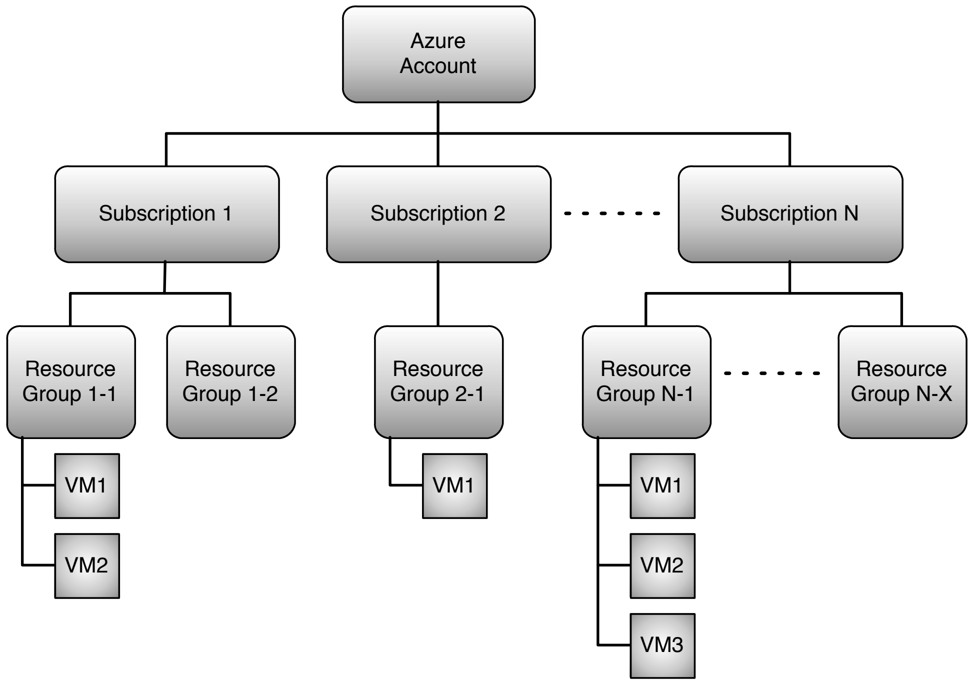


Figure 6.3: Azure Account Resource Division

cloud resources in an Azure account.

As a public commercial cloud, Microsoft Azure provides featured APIs for users to use its functions. Since we can't modify Azure itself, one way to approach SID-service function in Azure is to build it as a service provided by a third party in the cloud to customers. It requires to build additional services on the cloud platform. In the enforcement, we use a python web server to communicate with Azure API to provide SID-service, which is the same web server we used in AWS-AC-SID model enforcement. SID-service consists of two parts, one is the web server, the other is a SID-manager Azure account with subscriptions for all the potential sids created in the cloud. SID-service acts as an automated agent that manages all the sids and their constituent components through their life cycle. SID-service web server is the interface which respond to organizations' requests. SID-server processes all sids requests from communities of organizations and maintains a separate sid for each community. Within each sid, it facilitates the creation and deletion of sips. SID-manager account manages all subscriptions in response to the requests.

Considering that Azure already has its dedicated roles for managing subscriptions and Azure

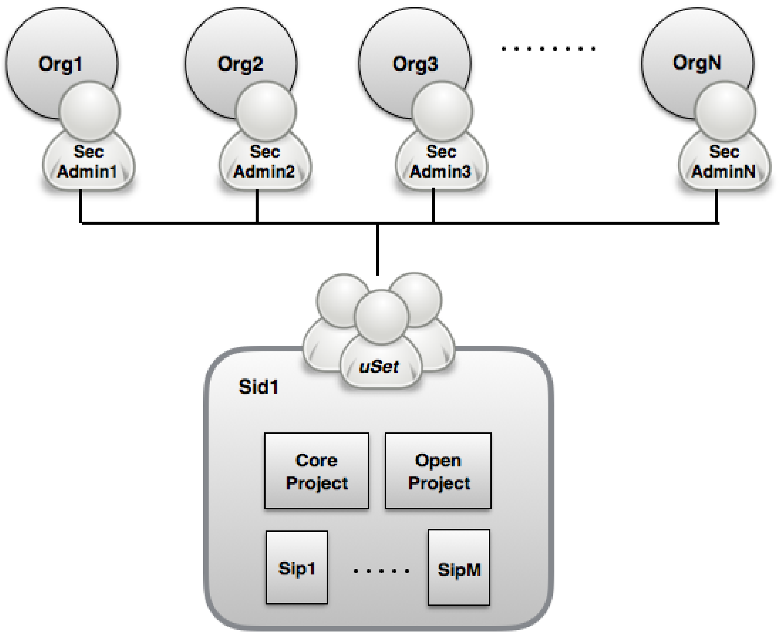


Figure 6.4: A Sid for a Community

Active Directory, we are going to use those existing administrative roles from Azure AD roles and subscription Roles to manage sips, core project and open project in a sid. Azure provides us five Azure AD admin roles and two subscription admin roles. As the SID-service, the SID-manager needs administrative roles to include Azure AD global admin role and subscription service admin role. Azure also provides a great set of operative roles in resource groups, which allows users to have permission to access cloud resources.

SID-manager maintains a list of security administrative users (*uSet*) from each community of organizations. Each organization in the community has one and only one security administrative user in *uSet*, which represents the organization in the sid, as shown in figure 6.4. SID-manager also maintains the associations for each sid with its member organizations in the community.

6.3.1 Functionalities

SID-server:

SID-server is the same web sesrver as used in AWS-AC-SID model enforcement. SID-server processes requests from organizations and maintains sid related information in the server backend.

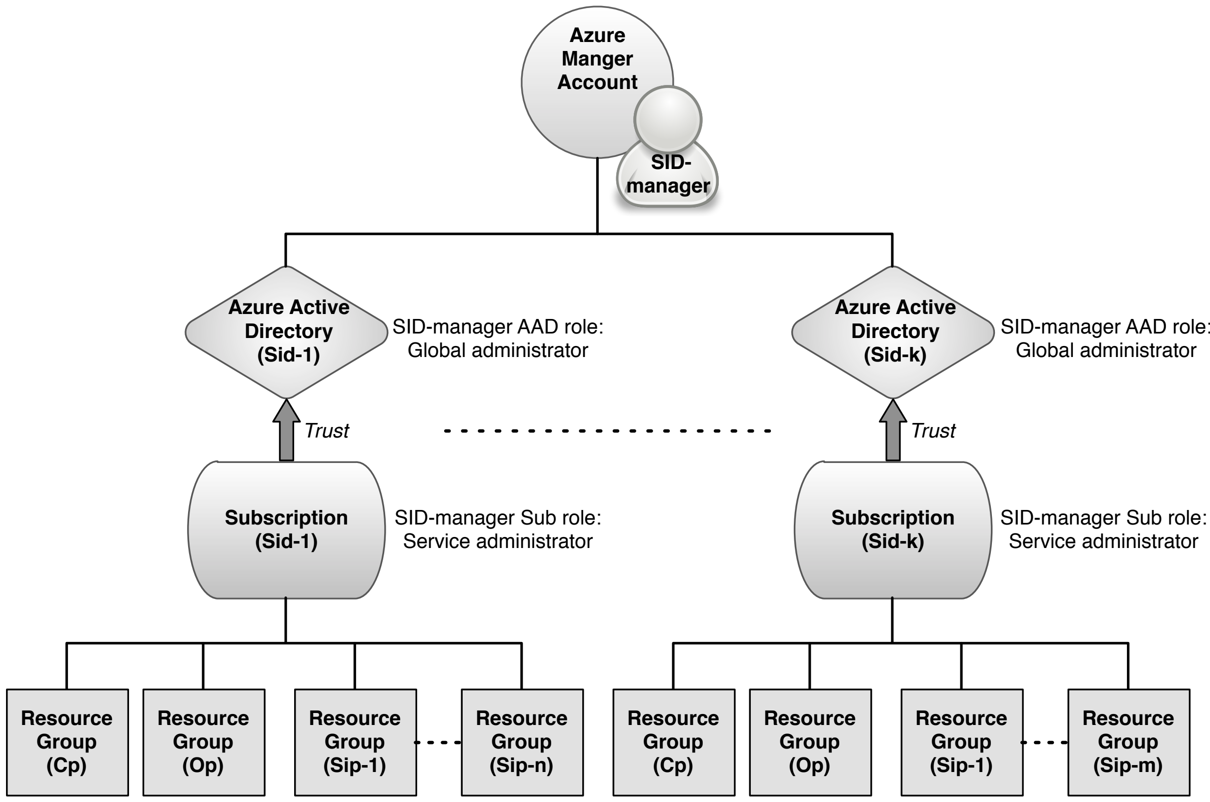


Figure 6.5: SID-manager User

The server uses a rest api module running on apache service in Linux. Organizations send rest api requests to SID-server and get response from the server.

SID-server consists of a running backend and a SID-manager Azure account. The running backend accepts and processes SID-requests of community organizations. SID-manager account manages subscriptions in response to SID-requests. Subscriptions inside SID-manager account are for sids, whenever a sid creation request is sent to SID-server, a new subscription will be created as a sid and associated with a group of organizations. With the creation of a sid, a core project and an open project are created. Resource groups are used for core project, open project and sips. Whenever a sip creation request is sent to SID-server, a new resource group will be created as a sip inside the sid.

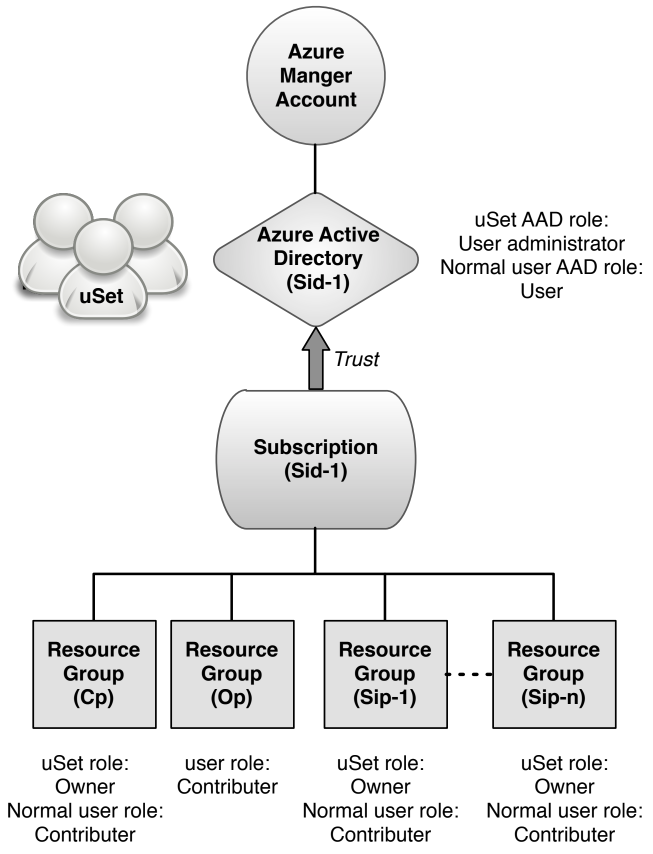

Azure Manger Account
uSet
Azure Active Directory (Sid-1)
uSet AAD role: User administrator Normal user AAD role: User
Trust
Subscription (Sid-1)
Resource Group (Cp)
Resource Group (Op)
Resource Group (Sip-1)
Resource Group (Sip-n)
uSet role: Owner Normal user role: Contributer
user role: Contributer
uSet role: Owner Normal user role: Contributer
uSet role: Owner Normal user role: Contributer

Figure 6.6: Create a Sid

SID-manager User:

For SID-manager Azure account, the owner of the account is the special manager user which have the permissions to process upcoming SID-requests. SID-manager creates active directories, subscriptions and resource groups for sids creation requirements, as shown in Figure 6.5. During SID-requests and SIP-request processing, the special SID-manager user will automatically respond to requests from organizations. The special SID-manager user is in charge of assigning proper roles for users from the collaborative community and delete the active directory and subscription associated with the sid.

Setting up a sid for organizations:

A Sid is set up for a community of organizations in the cloud. A core project and an open project are created with the creation of a sid. Sips then are created due to different collaboration purposes. Figure 6.6 shows a sid creation, with role assignments during the creation process. The security admin users group *uSet* are added to the sid subscription as administrators. These roles are inherited down to each resource group in the subscription, which gives admin users in *uSet* the administrative power over core project, open project and all sips. Each admin user from *uSet* is assigned with Azure active directory role *User Admin* and *Owner* role for the subscription. This allows admin users from *uSet* assign users from their own organizations to the sid active directory and resource groups in the subscription. Normal users are assigned with role *User* in sid active directory and role *Contributer* in resource group.

Roles:

In the model, we constrains roles into two roles: *SIDadmin* and *SIDmember*. Specifically in enforcement, role *SIDadmin* is realized by Azure active directory role *User Admin* and role *Owner* in subscriptions. Role *SIDmember* is realized by Azure active directory role *User* and role *Contributer* in subscriptions.

Azure activedirectory role *User Admin* allows a user managing other users in the active directory. Role *Owner* in subscriptions allows a user to manage resources in a resource group, as well as assign other users to the subscription or a resource group in the subscription. It requires both AAD role *User Admin* and *Owner* role for a security admin user to manage both users and cloud resources. Azure active directory role *User* has permissions which allows normal users be able to access a subscription associated with the active directory. Role *Contributer* in subscriptions gives users only permissions to access to resources, but managing other users. It requires both AAD role *User Contributer* role for a normal user to have operational permissions in a subscription. These two sets of roles are used in core project and all sips.

Database:

Same as in AWS-AC-SID model enforcement, SID-server maintains the association information for each sid with its member organizations in the community. With security administrative users (*uSet*) from organizations, each organization in the community has one and only one security administrative user in *uSet*. SID-server mains a list of organizations accounts name and accounts ids. We have one table in database to maintain the association information for all sids. Different with AWS-AC-SID model enforcement, we dont need a table to keep association information for sips. *SIDs* table holds basic information for each sid, include sid id, sid name and sid members.

SID Request Handling

A representative organization from a community of organizations can send sid request to SID-server. An administrative user from *uSet* sends a sid request to SID-server. SID-manager creates a sid subscription and adds the sid association information to database table and then create a core project resource group and open project resource group as part of initialization of the sid, and assigns all the roles that needed for each organization to access to the sid. The whole process is completed automatically, as shown in Figure 6.7. After the request, the community of organization should have their sid with a core project and open project. The set of security administrative users *uSet* should have limited admin access to core project and all users from the community should have member access to open project. Admin users from *uSet* should be able to send sip creation request to SID-server. SID-manager returns a sid subscription with core project resource group and open project resource group, as well as all the role assignment.

When a sid is deleted, all the project resource groups inside it will be deleted, including core project, open project and all sips. Notice that in Azure, in order to delete a subscription, any resource group inside the subscription are required to be deleted first. With deleting the project, any roles that are assigned to the sid subscription and project resource groups will be unassigned.

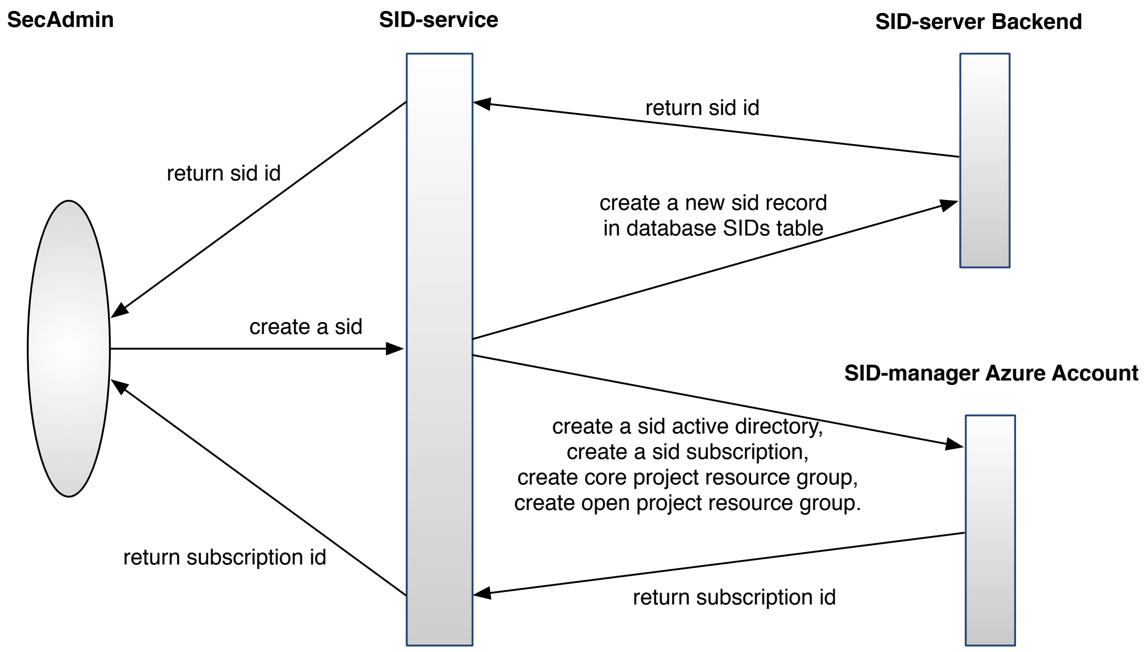

SecAdmin
SID-service
SID-server Backend
return sid id
return sid id
create a new sid record in database SIDs table
create a sid
SID-manager Azure Account
create a sid active directory, create a sid subscription, create core project resource group, create open project resource group.
return subscription id
return subscription id

Figure 6.7: SID-request Process

SIP-requests handling

When a representative security administrative user from *uSet* sends a sip request to SID-server, SID-manager creates a sip by creating a resource group in the corresponding subscription in the SID-manager account, and grant the group of users from *uSet* with role *Owner* to be admin users of the sip resource group. Role *Owner* with active directory role *User Admin* together gives admin users the permission to add other users from their home organizations to the sid active directory and sip resource group. From perspective for a security admin user, we have Figure 6.8 showing the process of a sip request.

After a collaboration is completed, organizations can request to delete the sip. The sip resource group will be deleted with all the information and resources that is created during the collaboration will be cleaned up. All users who are granted access to the sip will be removed from the sip resource group.

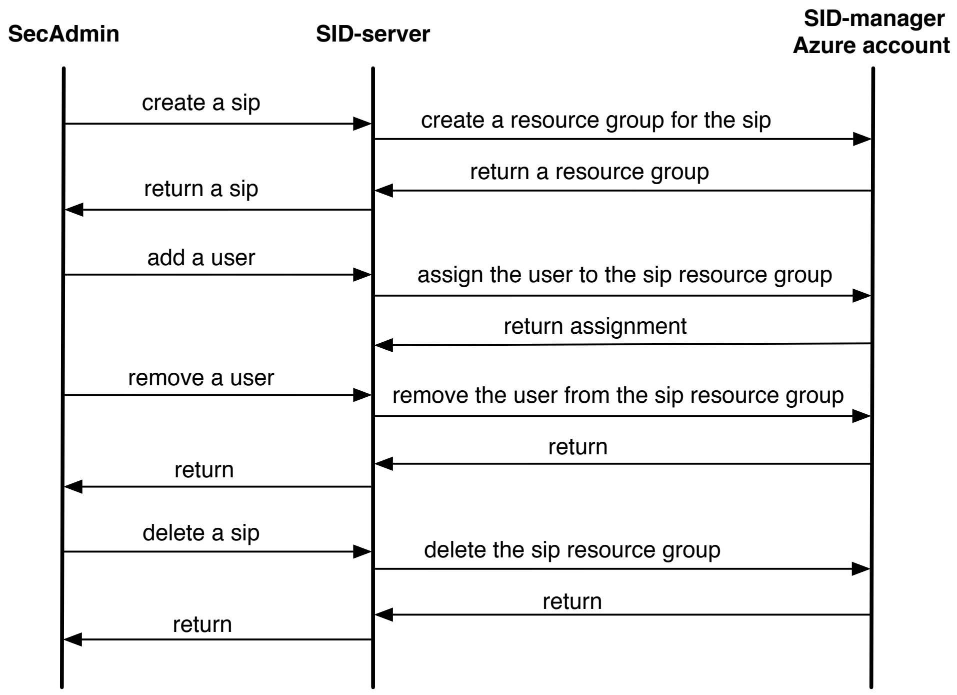

SecAdmin
SID-server
SID-manager Azure account
create a sip
create a resource group for the sip
return a resource group
return a sip
add a user
assign the user to the sip resource group
return assignment
remove a user
remove the user from the sip resource group
return
return
delete a sip
delete the sip resource group
return
return

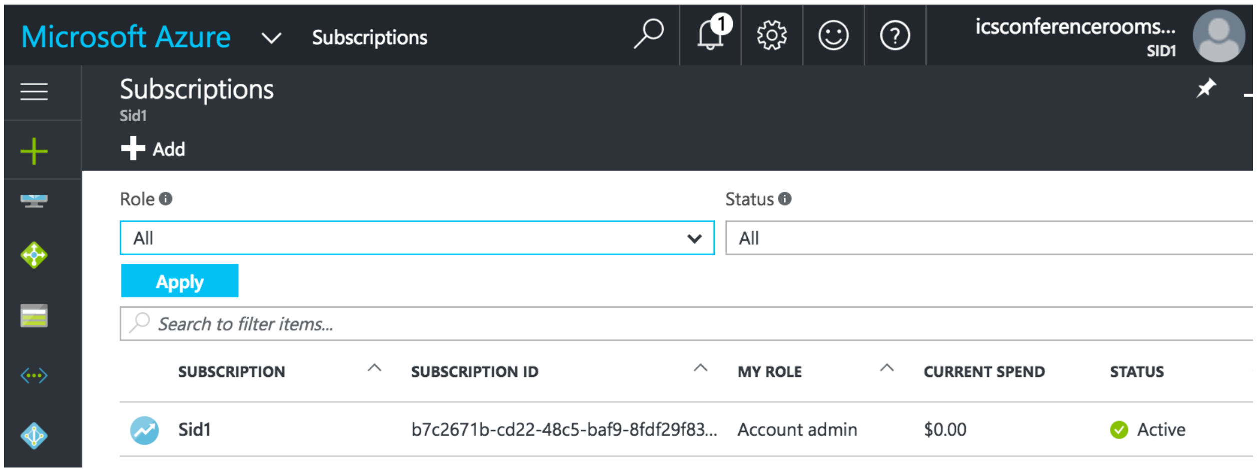
Figure 6.8: Sending SIP-requests

6.3.2 Demonstrations

In this part, demonstrations are given to show how the model enforcement presents in Microsoft Azure cloud platform. One issue we counter is that Azure doesn't offer a complete set of API calls for python sdk package. Thus, we couldn't realize the whole automation of Azure-AC-SID model. However, we can still manually set up sids for groups of organizations. In the following part, we are showing the result we got from manually setting up sids for communities.

Create a Sid:

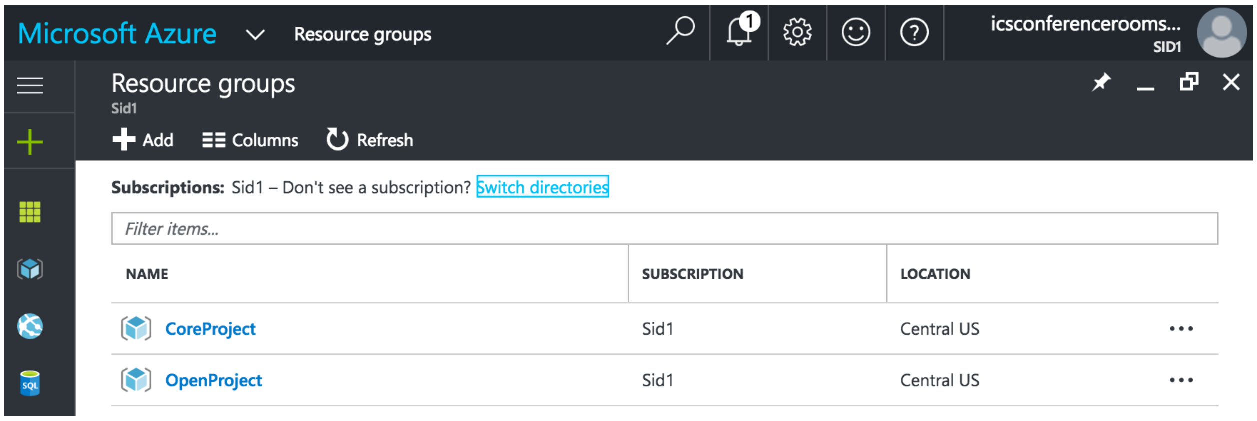
After a sid creation request is sent to SID-server by organizations, a sid instance is created in SID-manager Azure account. Figure 6.9 shows a sid named *Sid1* is created in SID-manager account, with Azure active directory SID1 and subscription Sid1 associated with the active directory. A sid is always created with two routine projects: core project and open project. We can find these two projects in resource group tab, as show in Figure 6.10. They are resource groups.



SUBSCRIPTION	SUBSCRIPTION ID	MY ROLE	CURRENT SPEND	STATUS
Sid1	b7c2671b-cd22-48c5-baf9-8fdf29f83...	Account admin	$0.00	Active

Role: All
Status: All

Figure 6.9: A Sid is Created

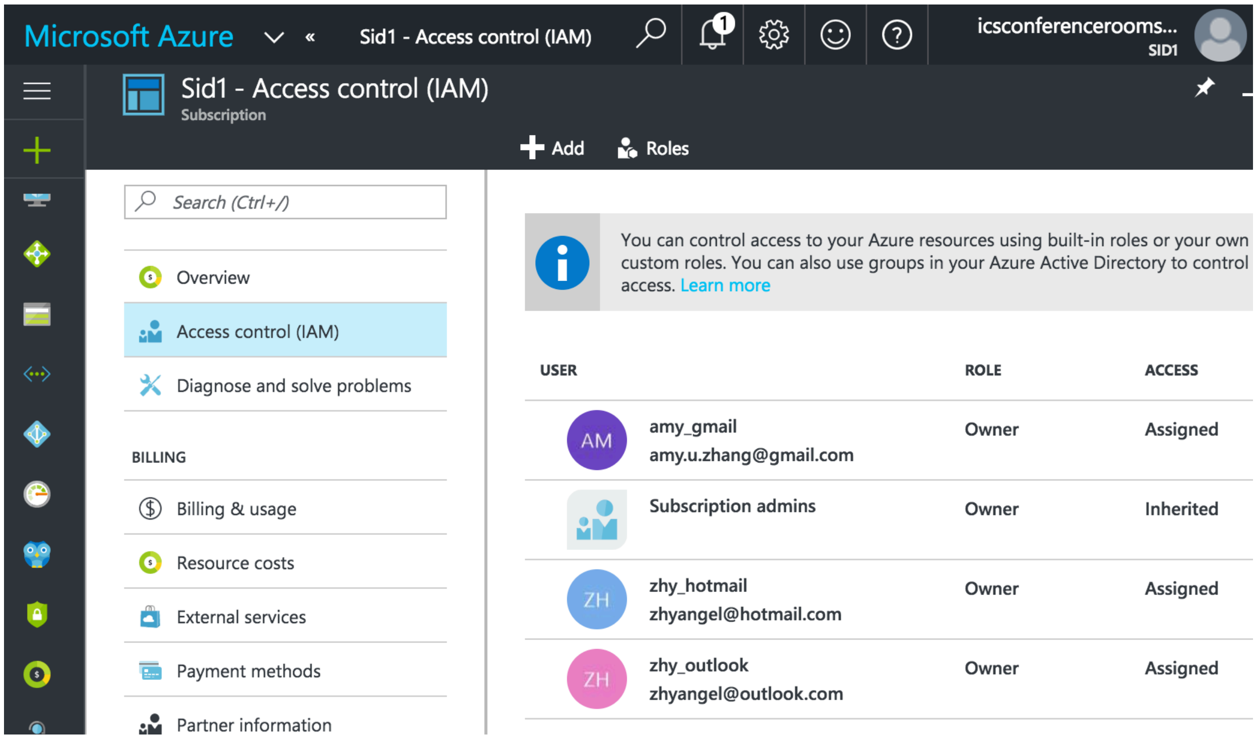


Subscriptions: Sid1 – Don't see a subscription? Switch directories

NAME	SUBSCRIPTION	LOCATION
CoreProject	Sid1	Central US
OpenProject	Sid1	Central US

Figure 6.10: Projects after A Sid is Created

In access control pannel of a sid subscription, we can check out the group organizations who are associated with the sid, each of which is an Azure account. Figure 6.11 shows how to check the group of organizations. "Subscription admins" as show in the figure, is the Azure account owner. "Subscription admins" is the inherited owner of the sid subscription, which inherits the permissions from the acccount owner. The rest three acts like three organizations. They are Azure account owners as well. Their ownership over the sid subscription comes from the assignment process during the sid creation request.


Microsoft Azure « Sid1 - Access control (IAM)

icsconferencerooms... SID1

Sid1 - Access control (IAM)
Subscription

Add Roles

Search (Ctrl+/)

Overview
Access control (IAM)
Diagnose and solve problems

BILLING

Billing & usage
Resource costs
External services
Payment methods
Partner information

You can control access to your Azure resources using built-in roles or your own custom roles. You can also use groups in your Azure Active Directory to control access. Learn more

USER	ROLE	ACCESS
amy_gmail amy.u.zhang@gmail.com	Owner	Assigned
Subscription admins	Owner	Inherited
zhy_hotmail zhyangel@hotmail.com	Owner	Assigned
zhy_outlook zhyangel@outlook.com	Owner	Assigned

Figure 6.11: A Sid Associated Organizations

Create a Sip:

After a sip creation request is sent to SID-server by organizations, a sip resource group is created in the sid subscription in the SID-manager Azure account. As an member organization of the sid, the organization admin user is able to see the sid subscription under the organization's Azure account after the sid creation request is finished. The organization admin user is able to login to the sid directory as well. After a sip is created successfully, it will show up in resource groups in the sid subscription. Figure 6.12 shows an organization admin user logins to the sid directory and is able to manage the sid with all the created projects resource groups in it, viz., *CoreProject*, *OpenProject* and the new created sip *Sip1*.

Add a User to a Sip:

A normal user needs to added to the sid active directory in order to access to resources in the sid subscription, as shown in Figure 6.13. The normal user "zhyhotmail_u1" is added to the sid active

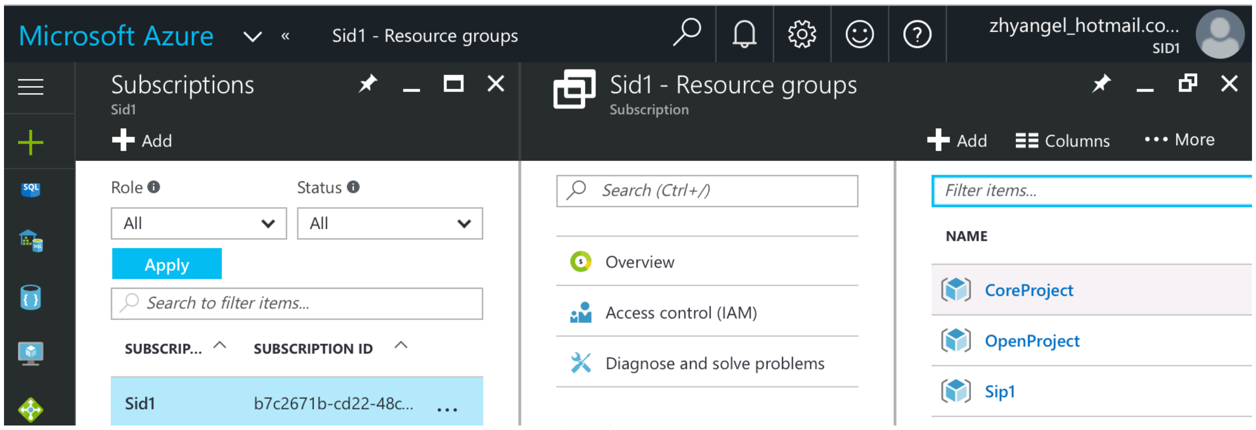
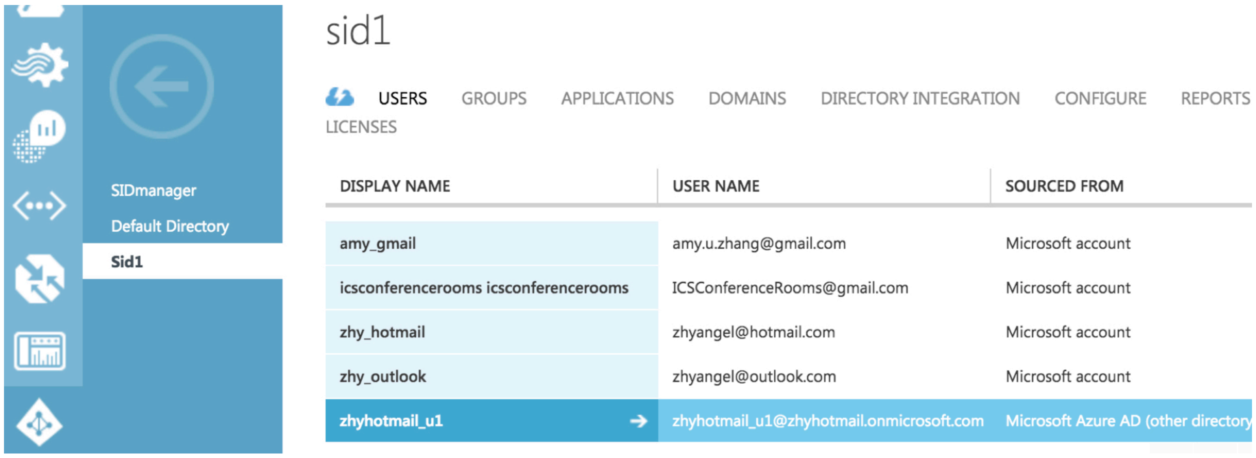


Figure 6.12: A Sip is Created



DISPLAY NAME	USER NAME	SOURCED FROM
amy_gmail	amy.u.zhang@gmail.com	Microsoft account
icsconferencerooms icsconferencerooms	ICSConferenceRooms@gmail.com	Microsoft account
zhy_hotmail	zhyangel@hotmail.com	Microsoft account
zhy_outlook	zhyangel@outlook.com	Microsoft account
zhyhotmail_u1	zhyhotmail_u1@zhyhotmail.onmicrosoft.com	Microsoft Azure AD (other directory

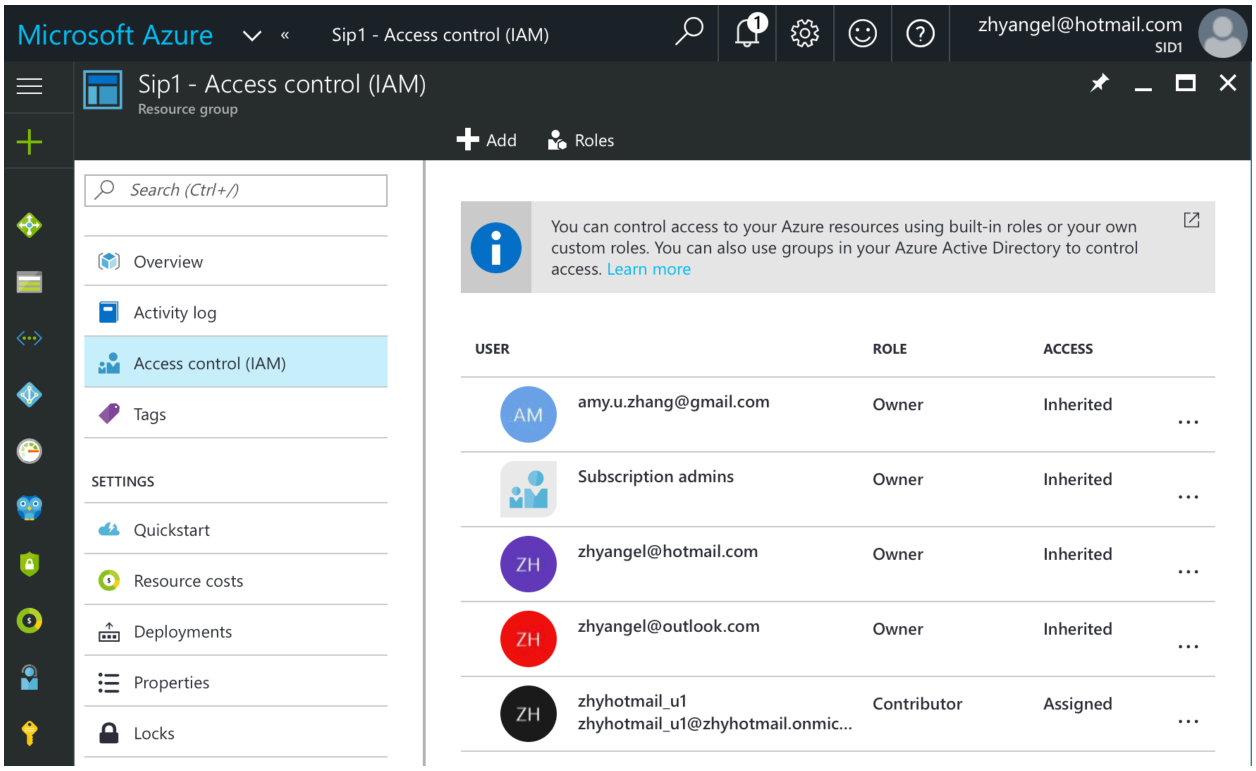
Figure 6.13: Assign a User to a Sid Active Directory

diretory as a recoganized user from other directory. After the normal user becoming a user of the sid active directory, then the user is allowed to add to resource groups in the sid subscription. Figure 6.14 shows user “zhyhotmail_u1” is added to *Sip1* resource group with role *Contributor*, which gives the normal user permission to access resources except managing users.

Login to a Sip:

After the normal user is added to a sip resource group, the user can login to the sip. Figure 6.15 shows user “zhyhotmail_u1” login to *Sip1* resource group in sid subscription *Sid1*.

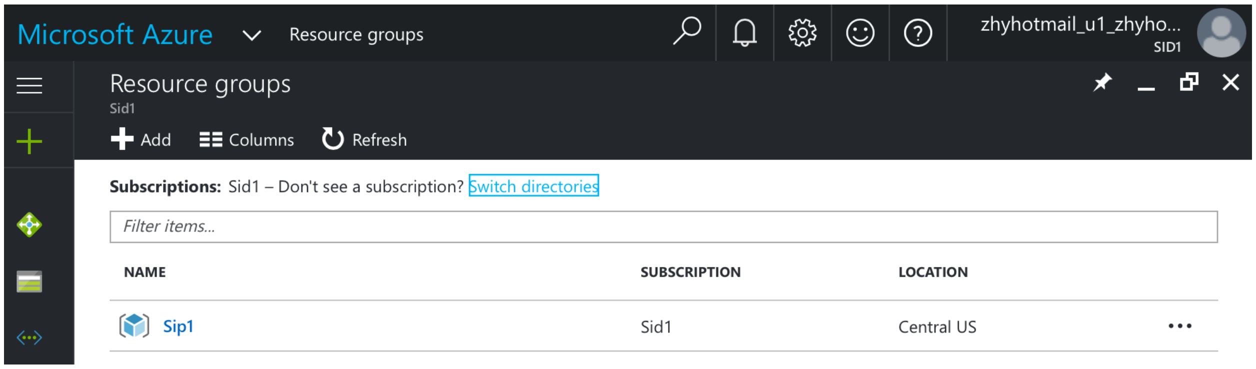
Eventually, at one point, the created sip resource groups or even the whole sid subscription will



You can control access to your Azure resources using built-in roles or your own custom roles. You can also use groups in your Azure Active Directory to control access. Learn more

USER	ROLE	ACCESS
amy.u.zhang@gmail.com	Owner	Inherited
Subscription admins	Owner	Inherited
zhyangel@hotmail.com	Owner	Inherited
zhyangel@outlook.com	Owner	Inherited
zhyhotmail_u1 zhyhotmail_u1@zhyhotmail.onmic...	Contributor	Assigned

Figure 6.14: Add a User to a Sip



Subscriptions: Sid1 – Don't see a subscription? Switch directories

Filter items...

NAME	SUBSCRIPTION	LOCATION
Sip1	Sid1	Central US

Figure 6.15: A User Login to a Sip

be deleted due to security reasons. All the resources created in resource groups will be deleted and users assigned to the active directory will be unassigned.

Chapter 7: CONCLUSION AND FUTURE WORK

In previous chapters, we have introduced SID-model into three dominant cloud systems: OpenStack, Amazon AWS and Microsoft Azure. We give access control models for each of these cloud systems according to our understanding of the systems and the needs for our model. We then construct SID-model in these cloud systems. In this chapter, we first compare SID-model in these three cloud platforms. We then conclude the work that has been done and state the future work.

7.1 Models Comparison

Resource Containers:

When we created the concept of secure isolated domain, we refer to a secure isolated space in a system, where information and resources can be securely shared. It is further divided into projects, which are sub-divisions of the isolated space. In cloud system, a sid can be either a real resource container or an administrative boundary of a couple of resource containers.

The three dominant cloud IaaS systems give different solutions to realize resource containers. Amazon AWS offers one-level resource containers, which are realized by AWS accounts. By owning an AWS account, an isolated tenant of the cloud is offered. Accounts are the only resources isolation divisions in AWS cloud. Current version of OpenStack offers two-level resource containers realized by projects and domains. Projects are the real cloud resource container, while domain is an administrative boundary of cloud resource container, which consists of multiple projects. With feature hierarchical multitenancy, OpenStack will be able to offer multiple-level resource containers. Microsoft Azure offers three-level resource containers. The basic cloud resource container is resource group. Accounts and subscriptions are two upper levels of administrative boundaries of resource container. An Azure account consists of multiple subscriptions, while a subscription consists of multiple resource groups.

The SID-model has two-level resource containers, the basic resource container projects and the administrative boundary level sids. To achieve SID-model in these three cloud systems, proper

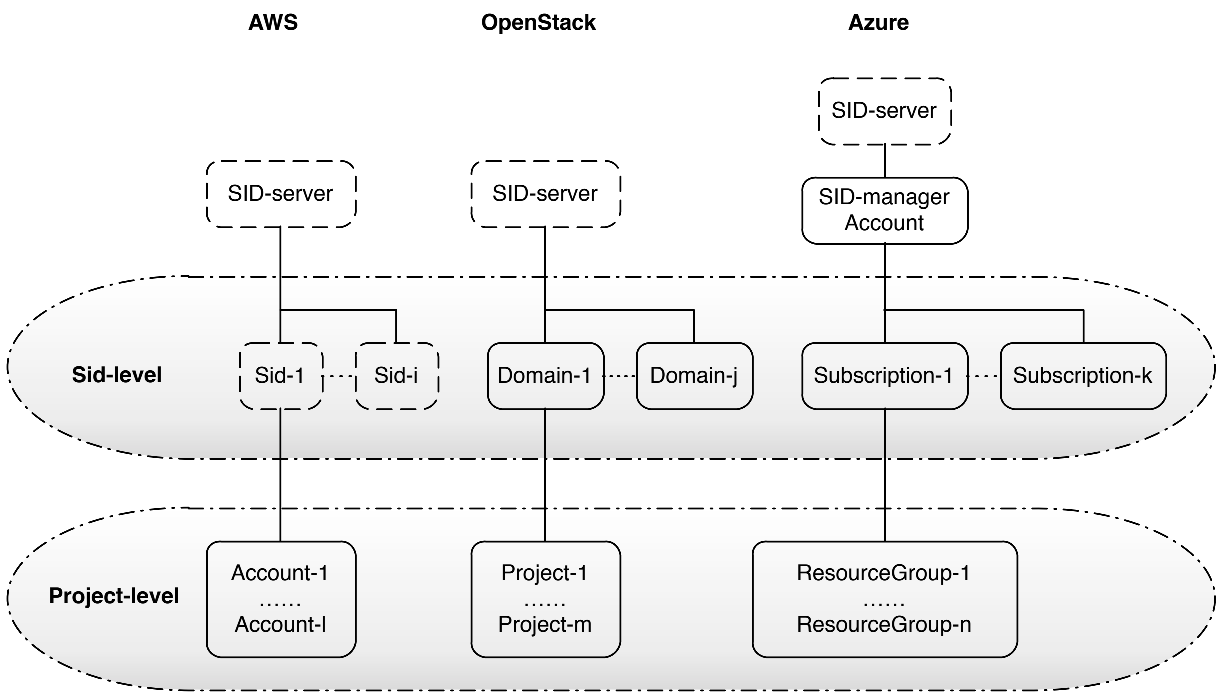


Figure 7.1: Resource Containers

adjustments are required. For AWS cloud platform, we use AWS accounts as the basic resource container projects in SID-model. We then add an administrative boundary level sids on top of it. The structure of current version of OpenStack suits SID-model very well. Thus, domains are used for sids and projects are used for projects in sids. For Microsoft Azure, we use subscription as administrative boundary level sids and resource groups as basic resource containers projects inside sids. Figure 7.1 shows the relation of different levels of resource containers in these three cloud systems toward SID-model's level of resource containers.

SID-service:

Amazon AWS and Microsoft Azure are commercial cloud platforms. To enforce the SID-model into these two systems and realize the automation, an extra service is required. SID-service serves as a third party service processing SID-requests from organizations. This allows the two commercial cloud systems to be able to provide SID-service without adding functionalities in the systems themselves. The same method can be used in OpenStack platform. However, since OpenStack is an open source system, SID-service can be directly added as modules in OpenStack system. This gives OpenStack platform capability to securely share information and resources among a group of organizations (tenants in context of cloud system).

The common thing of applying SID-model to these three cloud systems is that, there must be some database to store sids association information. Each sid has some association information which record the sid member organizations information. During lifetime of the sid, the sid association information is used for authorization purpose, for example, verifying users home organization. In OpenStack, we add a table in Keystone database to record sids association information. The same table is added to SID-service web server backends in AWS-AC-SID model enforcement, as well as Azure-AC-SID model. In AWS-AC-SID model enforcement, we also add one more table to record all projects (viz,. sips, core projects, open projects) information, because AWS has only one level resource containers and it is unable to manage the structure of two levels of resource containers.

Compared to AWS, OpenStack and Azure are quite different. Both of them have at least two level resource containers and each level is well isolated. With these two platforms, no table needs for record projects information, since the cloud platforms themselves provides the capability to manage projects resource container level. For instance, in OpenStack, after a sid is created, to create a sip in the sid is under the control of Keystone authorizations. It is same as creating a project in a domain. With the inherited roles features in OpenStack, the group of sid admins will automatically get admin roles in any new created project in the sid. Even without this feature, sid admin can still manually assign themselves project admin role. Azure has the similar mechanism to control roles. Users who are assigned as owner of a subscription will inherit the owner roles in all new created resource groups in the subscription. This allows the group of sid admin users automatically get owner roles in new created sips.

Roles:

SID-model has two types of roles: administrative role and operational role. Administrative role gives permissions of managing users and accessing to cloud resources. Operational role gives permissions of accessing to cloud resources. Thus, to enforce SID-model to these three systems, proper role assignments are necessary.

Roles are used in different ways in these three cloud platforms. In OpenStack platform, role *Admin* and role *Member* match roles in SID-model. Role *Admin* is used as administrative role and role *Member* as operational role in OSAC-SID model. In AWS platform, roles not only define the permissions of accessing cloud resources, but also the trust relationship between accounts. Besides the existing role set, one can create customized roles on his own needs. Customized roles are used in AWS-AC-SID model, viz., a set of administrative roles and a set of operational roles. Similar to AWS, Azure has its own existing set of roles, but also support customized roles. More differently, Azure has separate role sets for subscriptions and active directories. To get permissions of the administrative role in SID-model, all three types of roles are required, viz., active directory role, subscription role and roles in subscriptions and resource groups.To get permissions of the operational role in SID-model, both active directory role and roles in subscriptions and resource groups are required. In Azure platform, the existing roles are good enough for forming the administrative and operational roles in SID-model. Role *User Admin* in Azure active directory, role *co-administrator* for managing a subscription and role *Owner* in subscriptions together form administrative role in Azure-AC-SID model. Role *User* in Azure active directory and role *Contributor* in subscriptions together form operational role in Azure-AC-SID model.

Although all these three cloud systems give admin users permission to manage other users inside a cloud account, the *SIDmember* role assignment becomes very interesting in different cloud system. For OpenStack and Azure platforms, admin users can directly assign normal users to their territory resource containers (domain and projects in OpenStack, and subscription and resource groups in Azure). Still Azure has more complex way to control permissions. A normal user need both active directory role and roles inside a subscription to gain the access. For AWS-AC-SID model, it requires across account role assignment. The inside-a-account role assignment cannot give a user access to another account, which is a project in a sid. Thus it becomes a cross-account role assignment issue. Not only a policy has to be attached to a user, a trust relationship between accounts is required too before cross-account role assignments.

7.2 Summary and Future Work

In this dissertation, we identified fundamental elements of information and resource sharing in cloud IaaS. We suggested one sharing model to consider and build up various models on different cloud IaaS systems to facilitate secure information and resource sharing among multiple participants during collaborations. We formalized the model. We gave both administrative model and operational model, which gives a clear specification of how the model works from administrative and operational perspectives.

We further explore models for information and resource sharing in cyber security in dominant cloud systems, viz., open source cloud platform OpenStack, commercial cloud platform Amazon AWS and Microsoft Azure, and show the flexibility of those cloud access control models. For each of these cloud system in the dissertation, we first suggested an access control model and give the formalization of the model. We designed these models mainly based on the concept and architecture of these cloud platforms. We gave formal description of these models. We further gave formalizations of the sharing models in both administrative and operational perspective, which gives a clear specification of how the users and resources are managed and controlled in the model.

We also discussed the enforcement issues of the SID-model in the OpenStack, Amazon AWS and Microsoft Azure cloud platforms, specifying the components that need to be updated to support information and resource sharing. We specified the detail of data structures and functionalities realizations for the enforcement of our models.

Currently, we have finished the automation of main functionalities of SID-model in both Amazon AWS and OpenStack platform. We manually set up SID-service in Microsoft Azure platform due to uncompleted APIs. In OpenStack platform, these include adding SID and SIP functionalities to keystone server, and adding the specified tables to store sids association information. In AWS platform, we build a SID-server to serve SID-requests. We put SID and SIP functionalities in SID-server. The server maintains tables in backend database to store sids and sips related information. In Microsoft Azure, we manually set up SID-service and assign proper roles to users to realized SID-model.

For the future work, more perspective of thinking can be done in both modeling of information and resource sharing and enforcement of SID-model. From modeling perspective, the formation of a sid association is not emphasized in SID-model. The control of users' permission inside a sip is simple too. To enrich these two aspects would bring many flexibility to the model. For instance, more control on the group of organizations in the community could be added. Also, to investigate fine-grained access control within a sip gives great value to the sharing model. From perspective of enforcement of SID-model, exploring different ways of enforcement can be very interesting. It can add useful functionalities to open source cloud system OpenStack. It can provide a decent service for AWS and Azure customers. Especially, when more fine-grained access control are added to project level in a sid, the SID-service will not only serve for cyber security reason, but also for any business information and resource sharing purpose.

Finally, to investigate collaboration issues that arise when tenants belong to different cloud service providers is always an interesting topic. The sharing model introduced in this dissertation works in a single cloud system. However, same model can be extended to hybrid cloud system or multiple cloud systems too.

BIBLIOGRAPHY

[1] http://aws.amazon.com/.

[2] http://ken.people.info.

[3] https://azure.microsoft.com/.

[4] https://www.fsisac.com/.

[5] https://www.openstack.org/.

[6] Eve Cohen, Roshan K Thomas, William Winsborough, and Deborah Shands. Models for coalition-based access control (CBAC). In *Proc. 7th ACM SACMAT*, 2002.

[7] K. Harrison and G. White. Information sharing requirements and framework needed for community cyber incident detection and response. In *Homeland Security (HST), 2012 IEEE Conference on Technologies for*, pages 463–469, Nov 2012.

[8] Keith Harrison and Gregory B. White. Anonymous and distributed community cyberincident detection. *IEEE Security and Privacy*, 11(5):20–27, 2013.

[9] Ram Krishnan, Ravi Sandhu, Jianwei Niu, and William Winsborough. Towards a framework for group-centric secure collaboration. In *5th IEEE CollaborateCom*, pages 1–10, 2009.

[10] Peter Mell and Timothy Grance. The NIST definition of cloud computing. *NIST Sp. Pub. 800-145*, Sept. 2011.

[11] Laura Pearlman, Von Welch, Ian Foster, Carl Kesselman, and Steven Tuecke. A community authorization service for group collaboration. In *3rd IEEE International Workshop on Policies for Distributed Systems and Networks*, 2002.

[12] Ravi Sandhu, Khalid Zaman Bijon, Xin Jin, and Ram Krishnan. RT-based administrative models for community cyber security information sharing. In *7th IEEE CollaborateCom*, 2011.

[13] Ravi Sandhu, Ram Krishnan, and Gregory B White. Towards secure information sharing models for community cyber security. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2010 6th International Conference on*, pages 1–6. IEEE, 2010.

[14] Deborah Shands, Richard Yee, Jay Jacobs, and E John Sebes. Secure virtual enclaves: Supporting coalition use of distributed application technologies. In *IEEE DARPA Information Survivability Conference and Exposition*, volume 1, pages 335–350, 2000.

[15] Bo Tang and Ravi Sandhu. Extending OpenStack access control with domain trust. In *Proceedings 8th International Conference on Network and System Security (NSS 2014)*, October 15-17 2014.

[16] Bhavani Thuraisingham, Vaibhav Khadilkar, Jyothsna Rachapalli, Tyrone Cadenhead, Murat Kantarcioglu, Kevin Hamlen, Latifur Khan, and Farhan Husain. Cloud-centric assured information sharing. In *Intelligence and Security Informatics*, pages 1–26. Springer, 2012.

[17] Jian-Wei Wang and Li-Li Rong. Cascade-based attack vulnerability on the US power grid. *Safety Science*, 47(10):1332–1336, 2009.

[18] Yun Zhang, Ram Krishnan, and Ravi Sandhu. Secure information and resource sharing in cloud infrastructure as a service. In *Proceedings of ACM Workshop on Information Sharing and Collaborative Security (WISCS)*, pages 81–90, 2014.

[19] Yun Zhang, Farhan Patwa, and Ravi Sandhu. Community-based secure information and resource sharing in AWS public cloud. In *1st IEEE International Conference on Collaboration and Internet Computing (CIC)*, 2015.

[20] Yun Zhang, Farhan Patwa, and Ravi Sandhu. Community-based secure information and resource sharing in Azure public cloud. In *4th CCS International Workshop on Security in Cloud Computing (SCC)*, 2015.

[21] Yun Zhang, Farhan Patwa, Ravi Sandhu, and Bo Tang. Hierarchical secure information and resource sharing in OpenStack community cloud. In *IEEE Conference on Information Reuse and Integration (IRI)*, 2015.

VITA

Yun Zhang grew up in Gansu, China. She earned a Bachelor's degree in Software Engineering from Beijing Institute of Technology (BIT). She earned a Master's and Doctoral degree in Computer Science from The University of Texas at San Antonio. Her future plan is to pursue a career of cloud computing in industry.