4

Access Control in Cloud IaaS

Yun Zhang, Ram Krishnan, Farhan Patwa, and Ravi Sandhu

University of Texas at San Antonio, San Antonio, TX, USA

4.1 Introduction

Cloud computing is revolutionizing the way businesses obtain IT resources. *Cloud computing* refers to Internet-based computing that provides on-demand access to a shared pool of configurable computing resources (Hassan 2011), such as networks, servers, storages, applications, services, etc. Instead of having an application installed on a local PC, applications are hosted in the Cloud. Cloud computing allows users and organizations to conveniently and rapidly get computing resources with minimal management effort, helping organizations avoid focusing on upfront infrastructure costs. Rapid maturity of both commercial and open source cloud platforms greatly contributes to the wider acceptance and application of cloud computing in industry.

Infrastructure-as-a-Service (IaaS) is a cloud service model (Mell and Grance 2011) in which a cloud service provider (CSP) offers compute, storage, and networking resources as a service to its tenants. *Tenant* refers to an organization that is a customer of a CSP. Traditionally, IaaS providers maintain strict separation between tenants, for obvious reasons. Thus their virtual resources are strongly isolated. For instance, in OpenStack (http://openstack.org), a tenant user does not have the capability to access resources outside its domain. *Domain* refers to the administrative boundary of that tenant. Similarly, in AWS (http://aws.amazon.com) and Microsoft Azure (https://azure.microsoft.com), *tenant* refers to an account—an administrative boundary. Users from one account (tenant) by default have no rights to access resources outside that account.

In this chapter, 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: 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 also specify novel ways to construct intertenant secure information and resource sharing. The chapter outline is as follows. In Section 4.2, we present some background knowledge: more details of cloud services and the idea of information and resource sharing. In Sections 4.3, 4.4, and 4.5, we introduce the cloud access-control models for OpenStack, AWS, and Azure, respectively. For each of those platforms, we

Security, Privacy, and Digital Forensics in the Cloud, First Edition. Edited by Lei Chen, Hassan Takabi, and Nhien-An Le-Khac.
© 2019 Higher Education Press. All rights reserved. Published 2019 by John Wiley & Sons Singapore Pte. Ltd.

first give a formal access-control model specification, 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. Section 4.6 concludes the chapter.

4.2 Background

Cloud computing has three service models: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). PaaS offer a development environment to application developers. SaaS offers application and software to end users. We focus on IaaS for two reasons: (i) IaaS is one of the most-adopted cloud service models today (as compared to PaaS and SaaS), and (ii) IaaS is the foundation of the Cloud, with characteristics such as elasticity, self-service, etc. By gaining insights into issues related to sharing at this lower level of abstraction, we can also develop better models for higher levels of abstraction of cloud computing, such as PaaS and SaaS.

Note that in the context of IaaS, the unit of sharing consists of virtual resources such as objects in a storage volume, virtual machines (VMs), etc. For models, we mainly focus on administrative aspects. Administrative models are concerned with managing which users and what resources are to be shared, setting up and tearing down platforms for sharing, etc. Examples include a tenant administrator creating a shared secure isolated domain, adding users and resources to and removing them from that domain, inviting other tenants to join the domain, etc.

While cloud technology provides significant convenience to business systems, it also gives great potential to facilitate cyber-collaborations among organizations. In a cloud community, organizations can share cybersecurity information with other members through a cybersecurity committee to make informed decisions about the community's 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 security experts who help organizations with security issues. When a cybersecurity incident occurs, the cybersecurity committee members start an incident-response group with a cross-organization security team including organizations' internal security specialists and external security experts, as illustrated in Figure 4.1. Security information about this incident is shared within the incident response group.

Models for information sharing in IaaS are lacking. The concept we used to build our models for sharing comes from Group-Centric Secure Information Sharing (g-SIS) (Krishnan et al. 2009), which presents a method to control access among a group of users and objects that is well suited to the collaborative community scenario. In particular, g-SIS enables sharing using copies of the original information, versus traditional sharing that gives access to original information and resources (Cohen et al. 2002; Pearlman et al. 2002; Shands et al. 2000). Sharing by copy gives additional security protection, since access to the copies can be provided in a tightly controlled environment.

We present access-control models in a way that fits our best understanding. We abstract a necessary set of components to describe an access-control model. Based on the cloud platform access-control model, we build models for secure information and resource sharing. Then we formalize the administrative model. When we discuss the

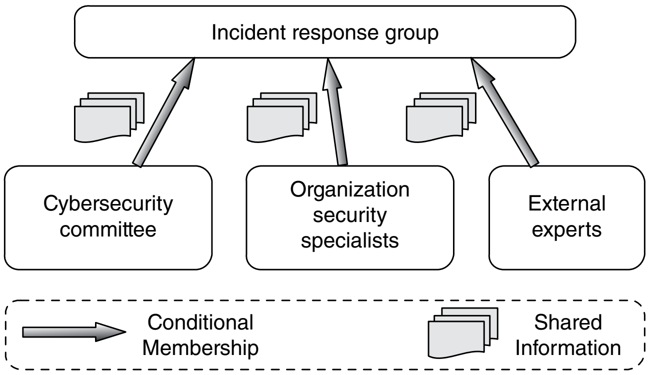

Figure 4.1 Community cyber-incident response governance.

models, we assume one organization has only one tenant in the cloud community. In the discussion of models for sharing, we simply ignore the group entity from cloud access-control models, since it is essentially a convenience to group users and can easily be incorporated in a more complete description. Instead, we use the term *group* to mean a group of organizations.

4.3 Access Control in OpenStack Cloud IaaS

In this section, we will introduce an access-control model for the OpenStack cloud IaaS platform and demonstrate its flexibility by extending it to include information and resource sharing. The content of this section has been published in (Zhang et al. 2015a). From the cloud provider's perspective, each tenant is an independent customer of the Cloud. From an organization's perspective, in general a single organization may have a single or multiple tenants in a single cloud. For simplicity, we assume here that each organization from the cloud community has exactly one tenant.

4.3.1 OpenStack Access-Control Model

A core OpenStack access control (OSAC) model was presented in (Tang and Sandhu 2014), based on the OpenStack Identity API v3 and Havana release. This model comprises nine entities: users, groups, projects, domains, roles, services, object types, operations, and tokens. Hierarchical multitenancy (HMT) (http://openstack.org) is a new feature added to OpenStack since the Juno release. We enhance the OSAC model with HMT, resulting in the OSAC-HMT model shown in Figure 4.2. In this and other figures in this chapter, the arrows denote binary relations, with the single arrowhead indicating one side and double arrowheads many sides.

Users represent people who are authenticated to access OpenStack cloud resources, while *groups* are sets of users. HMT does not change user/group management, which is handled at the domain level:

- **Domains and projects**—*Projects* are resource containers through which users access cloud services such as VMs, storage, networks, identity, and so on. Each project

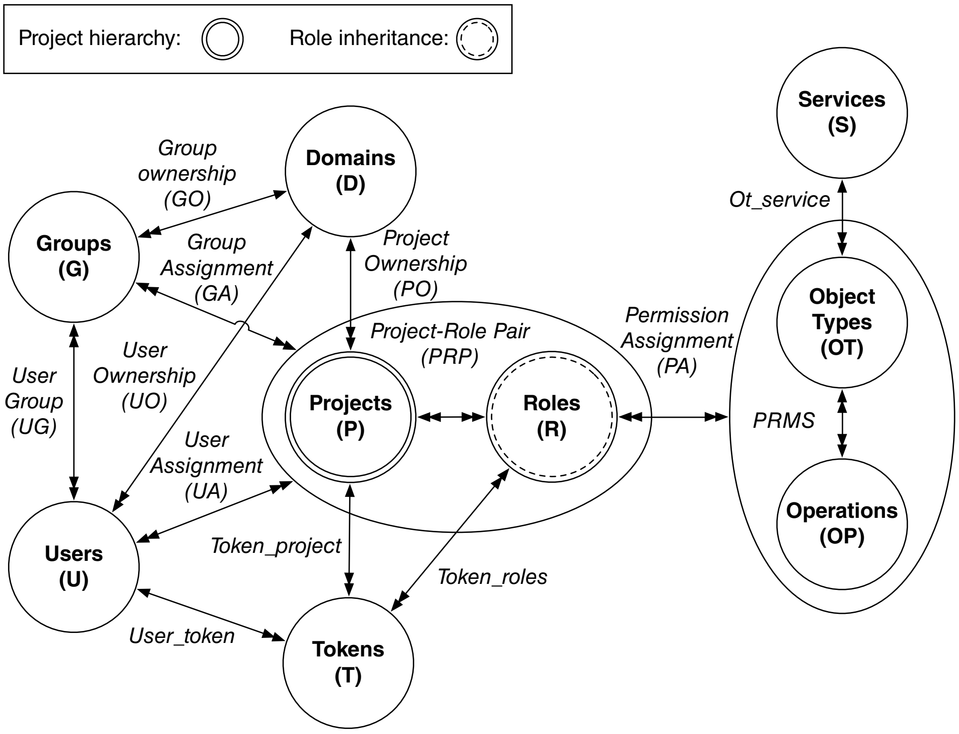

Figure 4.2 OpenStack Access Control (OSAC) model with HMT.

defines a boundary of cloud resources. *Domains* are administrative boundaries of collections of projects, users, and groups. Each domain contains multiple projects, users, and groups. Conversely, each project, user, and group is "owned" by a single domain. However, 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 administrative user of that domain (tenant).

- **Project hierarchy**—The *project hierarchy* enables the resources to be divided into smaller management units, giving tenants more power to control their cloud resources. A domain can have multiple projects in it, each of which is a root project for a hierarchical project tree. A child project has only one parent project. Basically, child projects are a further division of resources of a parent project.
- **Roles**—*Roles* are global in that each role is applicable to every project. Roles are used to specify access levels of users to services in specific projects in a given domain. Roles and their associated permissions are defined by the CSP. Note that users are assigned to projects with a specific set of roles. For instance, by assigning a role of Member to a 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.

- **Role inheritance**—Without a project hierarchy, a user is explicitly assigned to a project with a role. With a project hierarchy, a user needs to be able to be assigned to a child project, which is enabled by inherited role assignment. By assigning an inherited role to a user in a parent project, the user will automatically have the role in child projects.
- **Object types and operations**—An *object type and operation* pair defines actions that can be performed by end users on cloud services and resources. The concept of object types allows different operations to be specified for different services. For example, in the Nova compute service, an object type is VM, and operations on VM include start, stop, etc.
- **Token**—*Tokens* define the scope of resources that users are authenticated to access. Users authenticate themselves to the Keystone service and obtain a token that they then use to access different services. The token contains various information, including the user's domain and user's roles for specified projects. A token must be scoped to a target project on which the action is performed. Inherited roles allow tokens to be granted for child projects, giving access to the child projects.

We formalize the OSAC-HMT model next. Part of it is the same as the OSAC model (Tang and Sandhu 2014).

Definition 4.1 OSAC-HMT Model Components

- U, G, P, D, R, S, OT, and OP are finite sets of existing users, groups, projects, domains, roles, services, object types, and operations, respectively, in an OpenStack cloud system. We require two roles, so $\{admin, member\} \subseteq R$.
- 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 OT to its owning service. Equivalently viewed as a many-to-one relation $OTO \subseteq OT \times S$.
- $UG \subseteq U \times G$ is a many-to-many relation assigning users to groups, where the user and group must be owned by the same domain.
- $PRP = P \times R$ is the set of project-role pairs.
- $PERMS = OT \times O$ is the set of permissions.
- $PA \subseteq PERMS \times R$ is a many-to-many permission-to-role assignment relation.
- $UA \subseteq U \times PRP$ is a many-to-many user-to-project role assignment relation.
- $GA \subseteq G \times PRP$ is a many-to-many group-to-project role assignment relation.
- Project Hierarchy (PH) is a function PH: $P \rightarrow P$, mapping a project to its parent project. Equivalently viewed as a many-to-one relation $PH \subseteq P \times P$. This is required to be a forest of rooted trees.
- Role Inheritance (RI) allows users' roles to be inherited from domain to project and from parent project to child project, as discussed earlier.
- user_tokens is a function $U \rightarrow 2^T$, mapping a user to a set of tokens; correspondingly, token user is a function token user $T \rightarrow U$, mapping 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: $T \rightarrow 2^R$, mapping a token to its set of roles. Formally, token_roles(t) = $\{r \in R \mid (\text{token\_user}(t),(\text{token\_project}(t),r)) \in UA\} \cup (\bigcup_{g \in \text{user\_groups}(\text{token\_user}(t))} \{r \in R \mid (g, (\text{token\_project}(t), r)) \in GA\})$.
- avail_token_perms is a function avail token perms: $T \rightarrow 2^{PERMS}$, mapping the permissions available to a user through a token. Formally, avail_token_perms(t) = $\bigcup_{r \in \text{token\_roles}(t)} \{\text{perm} \in PERMS \mid (\text{perms},r) \in PA\}$.

4.3.2 Secure Information and Resource-Sharing Model in OpenStack

In this section, we present a model for sharing in OpenStack; we call it the Hierarchical Multitenancy OpenStack Access Control Model with Secure Isolated Domain extension (OSAC-HMT-SID model). In our discussion, we assume that a user belongs to one organization in the community, which is consistent with the user home-domain concept in OpenStack. The concept of a *home domain* requires that a user can belong to only one domain in OpenStack. OpenStack allows a user to be assigned to projects across domains and access those projects separately using the appropriate tokens.

The OSAC-HMT-SID model extends the OSAC-HMT model to include secure isolated domain (SID) (Zhang et al. 2014) functionality. We build the OSAC-HMT-SID model on top of the OSAC-HMT model. We will present the OSAC-HMT-SID model in a way that covers only the additional components compared to the OSAC-HMT model. Figure 4.3 shows the OSAC-HMT-SID model. We use circles to represents entities that can be created multiple times in OpenStack, whereas rectangles represent entities that can be created only once. The additional entity components included in the model are SID, Expert User (EU), Core Project (CP), Secure Isolated Project (SIP), and Open Project (OP):

- **Secure Isolated Domain (SID)**—A SID (Zhang et al. 2014) is a special domain that holds the security information for cross-organization security collaboration in the community cloud. It provides an administrative boundary for cybersecurity information and resource collecting, resource passing, analyzing and exporting results, as well as providing a secure isolated environment for cybersecurity collaborations among organizations.
- **Security Project (SP)**—SPs are hierarchical projects particularly used to collect, store, and analyze cybersecurity information for one organization. A SP provides the same capability of utilizing cloud resources as a normal project. Organizations keep their security information and resources in the SPs, with their security staff/users assigned to the corresponding level of project in the SP hierarchy. This separates an organization's regular projects from its SPs.
- **Core Project (CP)**—A CP is a shared project that holds the community cybersecurity committee (Sandhu et al. 2011). Each organization in the community has at least one user in the security committee, with one as an admin user of the CP and the rest as regular member users. The CP holds all SIPs that are designed for cyber-incident response and cybersecurity collaboration.
- **Open Project (OP)**—An OP is a project where users share public cybersecurity information and resources (Sandhu et al. 2011). Information published in an OP is public to every user who is subscribed to the project.
- **Secure Isolated Project (SIP)**—A SIP (Zhang et al. 2014) is a special project with constraints over its user membership, information, and resource utilization. The SIP

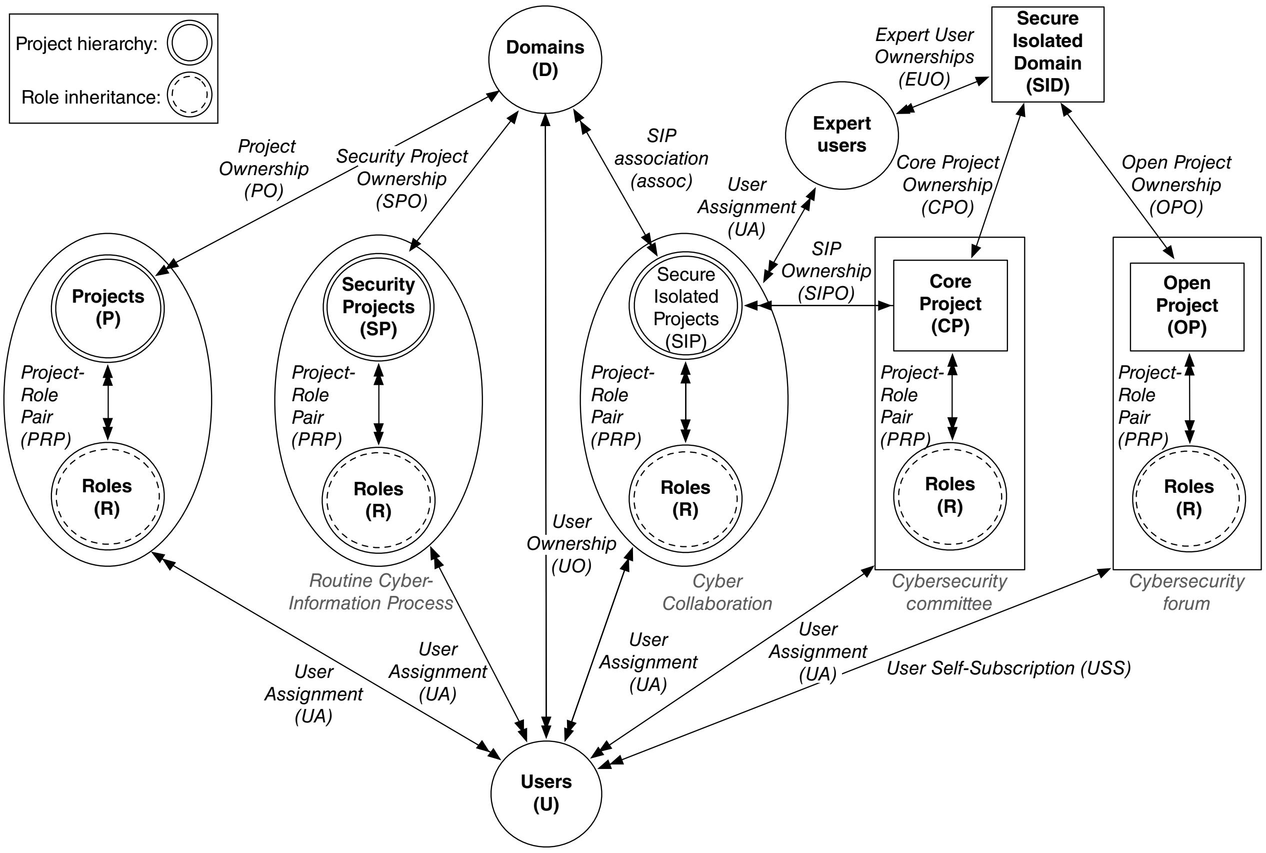

Figure 4.3 Hierarchical multitenancy OSAC model with SID extension (OSAC-HMT-SID) (ignoring the group, token, and services components).

provides a controlled environment for organizations to collaborate on security incidents.

- **Expert Users (EU)**—To get outside community professionals involved, EUs (Sandhu et al. 2011) are introduced to the SID. EUs originally don't belong to the community. They bring expertise from different cybersecurity categories. For instance, they may come from an IT consultant company that focuses on specific cyber attacks. Or they may be cybersecurity law-enforcement officers specializing in cybercrime. The involvement of EUs helps organizations handle cyber collaborations more effectively.

Following are the formalized concepts we just introduced, as well as the relationships among them.

Definition 4.2 OSAC-HMT-SID Model Components in Addition to OSAC-HMT

- SID is an implicitly existing SID, which is transparent to users. SID owns EU, CP, OP, and SIP, correspondingly represented by Expert User Ownership (EOU), CP Ownership (CPO), Open Project Ownership (OPO), and Secure Isolated Project Ownership (SIPO).
- SP, SIP, EU, and SO are finite sets of SPs, SIPs, EUs, and Swift Objects (SOs).
- Security Project Ownership (SPO) is a function SPO: $SP \rightarrow D$, mapping a SP to its owning domain. Equivalently viewed as a one-to-one relation $SPO \subseteq D$.
- Swift Object Ownership (SOO) is a function SOO: $SO \rightarrow P$, mapping a SO to its owning project. Equivalently viewed as a many-to-one relation $SOO \subseteq SO \times P$.
- User Self Subscription (USS) is a function $USS \subseteq U \times \{<OP, \mathit{member}>\}$, a many-to-one user-to-project role assignment relation for the Member role in the single OP.
- SIP association (assoc) is a function assoc: $SIP \rightarrow 2^D$, mapping a SIP to all its member domains/organizations.

4.3.2.1 Administrative OSAC-HMT-SID Model

The administrative aspects of OSAC-HMT-SID are discussed informally next. A formal specification is given in Table 4.1.

Creation of the SID, CP, OP, and SP: A SID with a CP and OP is part of the community cloud functionality the CSP provides to its customers on behalf of organizations responding collaboratively to cyber incidents. The SID, CP, and OP are created when the community cloud is set up. Each domain has one corresponding SP. The creation of a SP is automatically done with the creation of a domain.

Initial user assignment for the SID, CP, OP, and SP: The SID has no admin users assigned on the domain level. The admin users of the CP come from an organization's domain. When a domain is created, the cloud admin assigns a domain admin user as an admin of the CP. We assume there is only one admin user for each domain. Domain admins assign admin users for their SPs. The OP doesn't have an admin user assigned to it. Each user in the Cloud can self-subscribe or unsubscribe as a member of the OP.

Create a SIP: Let *uSet* denote a set of domain admin users. A group of organizations comes together to create a SIP. Each organization in the group has equal administrative power over the SIP. The creation of the SIP succeeds based on agreement among the organizations. Organization membership in the SIP is established with the

Table 4.1 OSAC-HMT-SID administrative model.

Operation	Authorization Requirement	Update
SipCreate(uSet, sip) /* *A subset of Core Project/domain admin users together create a sip* */	$\forall u \in uSet.(u \in U \wedge [u, <CP, admin>] \in UA) \wedge sip \notin SIP$	$assoc(sip) \cup_{u \in uSet} UO(u)$ $SIP' = SIP \cup \{sip\}$ $UA' = UA \cup uSet \times \{<sip, admin>\}$
SipDelete(uSet, sip) /* *The same subset of Core Project/domain admin users together delete a sip* */	$\forall u \in uSet.(u \in U \wedge (u, <sip, admin>) \in UA \wedge (u, <CP, admin>) \in UA) \wedge assoc.(sip) = \cup^{u \in uSet} UO(u) \wedge sip \in SIP$	$assoc(sip) = NULL$ $SIP' = SIP - \{sip\}$ $UA' = UA - uSet \times \{<sip, admin>\}$
ExpertUserCreate(coreadmin, eu) /* *Core Project admin users can create an expert user* */	$coreadmin \in U \wedge (coreadmin, <CP, admin>) \in UA \wedge eu \notin EU$	$EU' = EU \cup \{eu\}$
ExpertUserDelete(coreadmin, eu) /* *Core Project admin users can delete an expert user* */	$coreadmin \in U \wedge (coreadmin, <CP, admin>) \in UA \wedge eu \in EU$	$EU' = EU - \{eu\}$
ExpertUserList(adminuser) /* *Admin users of Core Project and SIPs can list expert users* */	$adminuser \in U \wedge (\exists proj) \{proj \in (\{CP\} \cup SIP) \wedge (adminuser, <proj, admin>) \in UA\}$	
ExpertUserAdd(adminuser, r, eu, proj) /* *Core Project/sip admin can add an expert user to Core Project/sip* */	$adminuser \in U \wedge proj \in (\{CP\} \cup SIP) \wedge (adminuser, <proj, admin>) \in UA \wedge eu \in EU \wedge r \in R$	$UA' = UA \cup (eu, [proj, r])$
ExpertUserRemove(adminuser, r, eu, proj) /* *Core Project/sip admin can remove an expert user from Core Project/sip* */	$adminuser \in U \wedge proj \in (\{CP\} \cup SIP) \wedge (adminuser, <proj, admin>) \in UA \wedge eu \in EU \wedge r \in R \wedge (eu, [proj, r]) \in UA$	$UA' = UA - (eu, [proj, r])$
UserAdd(adminuser, r, u, sp, p) /* *CP/Sip admin can add a user from their home domain Security Project to CP/sip* */	$adminuser \in U \wedge (adminuser, <p, admin>) \in UA \wedge p \in (\{CP\} \cup SIP) \wedge r \in R \wedge u \in U \wedge (u, <sp, r>) \in UA \wedge SPO(sp) = UO(adminuser)$	$UA' = UA \cup (u, [p, r])$
UserRemove(adminuser, r, u, sp, p) /* *CP/Sip admin can remove a user from the Core Project/sip* */	$adminuser \in U \wedge (adminuser, <p, admin>) \in UA \wedge p \in (\{CP\} \cup SIP) \wedge r \in R \wedge u \in U \wedge (u, <sp., r>) \in UA \wedge SPO(sp) = UO(adminuser) \wedge (u, [p, r]) \in UA$	$UA' = UA - (u, [p, r])$
OpenUserSubscribe(u, member, OP) /* *Users subscribe to Open Project* */	$u \in U \wedge (u, <OP, member>) \notin USS$	$USS' = USS \cup (u, <OP, member>)$
OpenUserUnsubscribe(u, member, OP) /* *Users unsubscribe from Open Project* */	$u \in U \wedge (u, <OP, member>) \in USS$	$USS' = USS - (u, <OP, member>)$
CopyObject(u, so1, sp., so2, p) /* *Copy object from Security Project to Core Project/SIP* */	$sol \in SO \wedge sp. \in SP \wedge so2 \notin SO \wedge SOO(so1) = sp \wedge UO(u) = SPO(sp) \wedge u \in U \wedge (\exists r \in R) \{(u, <sp, r>) \in UA \wedge (u, <p, r>) \in UA)\} \wedge p \in (\{CP\} \cup SIP)$	$SO' = SO \cup \{so2\}$ $SOO(so2) = p$
ExportObject(adminuser, so1, p. so2 sp) /* *Export object from Core Project/SIP to Security Project* */	$adminuser \in \cup \wedge (adminuser, <p, admin>) \in UA \wedge p \in (\{CP\} \cup SIP) \wedge so1 \in SO \wedge SOO(so1) = p \wedge so2 \notin SO \wedge sp \in SP \wedge (adminuser, <sp., admin>) \in UA$	$SO' = SO\ U\ \{so2\}$ $SOO(so2) = sp$

creation of the SIP. The size of the group ranges from one organization to the total number of organizations held in the community cloud. The group of organizations sets up a SIP by sending the SIP-creation request to the cloud admin. Users who are allowed to issue a SIP-creation command are admin users in CPs, who are domain admins as well. When a SIP is created, the users who issued the SIP-creation command automatically become the admin users of the SIP.

Delete a SIP: After the collaboration is finished, a SIP needs to be securely deleted. The delete command is issued by the same set of admin users (*uSet*) who issued the SIP-creation command. All information and resources are securely deleted. All users assigned to the SIP are removed from it. Removing information and resources guarantees no information and resources will leak after the SIP has been deleted. Removing users guarantees no users will have access to information and resources that belonged to a SIP.

Create/Delete an EU: New EUs are created when additional cyber expertise is needed, such as when a consultant company is introduced to the community or a new cybersecurity agent is involved with one of the collaboration groups. CP admin users send the EU-creation command to the cloud admin. he cloud admin returns the new EU and adds the user to the EU list. CP admin users can request to delete an EU. After the EU is deleted, the user will lose all access to any information and resources in the community cloud.

List EUs: CP and SIP admin users can list EUs in the SID. EUs are important human resources for cyber-collaboration activities. By listing the EUs in the SID, collaborative groups with SIPs can easily add experts to their SIPs.

Add/remove an EU: An EU is visible to all projects in the SID except the OP. Project admins in the SID can add EUs to their projects due to collaboration. After the cyber collaboration is done, project admins can remove EUs from their projects.

Add/remove a user to/from a CP/SIP: Admin users of a CP/SIP can add/remove users of their home SPs to/from CP or the corresponding SIP due to the need for collaboration. The removed user will lose access to information and resources that they had during collaborations in the CP/SIP.

Subscribe/unsubscribe a user to the **OP:** Every user in the OP is a normal member user. They can share cyber data but have no control over other users. Users subscribe/unsubscribe themselves to/from the OP. They will not be able to access and share any data once they leave the OP.

Copy data between a SP and CP/SIP: Users can copy data from SPs of their home domains to a CP and SIP. Users may be scoped to multiple projects in their home domains, but only data from SPs are allowed to be copied to a CP/SIP. Admin users can export data from CPs and SIPs to SPs of their home domains.

4.4 Access Control in AWS Cloud IaaS

In this section, we investigate a model for the AWS public cloud and demonstrate its flexibility by extending the access-control model to include information and resource sharing. The content of this section has been published in (Zhang et al. 2015b). As we did for OpenStack, for simplicity, we assume that each organization from the cloud community has only one tenant that is an AWS account.

4.4.1 AWS Access-Control Model

As a public CSP, AWS provides web services to its customers through AWS accounts. Customers that own an account have access to cloud resources. They can create users and grant them access to cloud resources in the account. A user belongs to a unique account. Users can also access resources in other accounts with federated permissions. We discuss the AWS access control (AWS-AC) model from 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 4.4 depicts this model within a single account. In this and other figures in this chapter, dotted lines denote virtual relations between entities, whereas solid lines denote explicit relations. Cross-account access will be discussed later in the context of Figure 4.5.

The AWS-AC model has seven components: accounts (A), users (U), groups (G), roles (R), services (S), object types (OT), and operations (OP). 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 that allow customers to own specific amounts 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* are 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

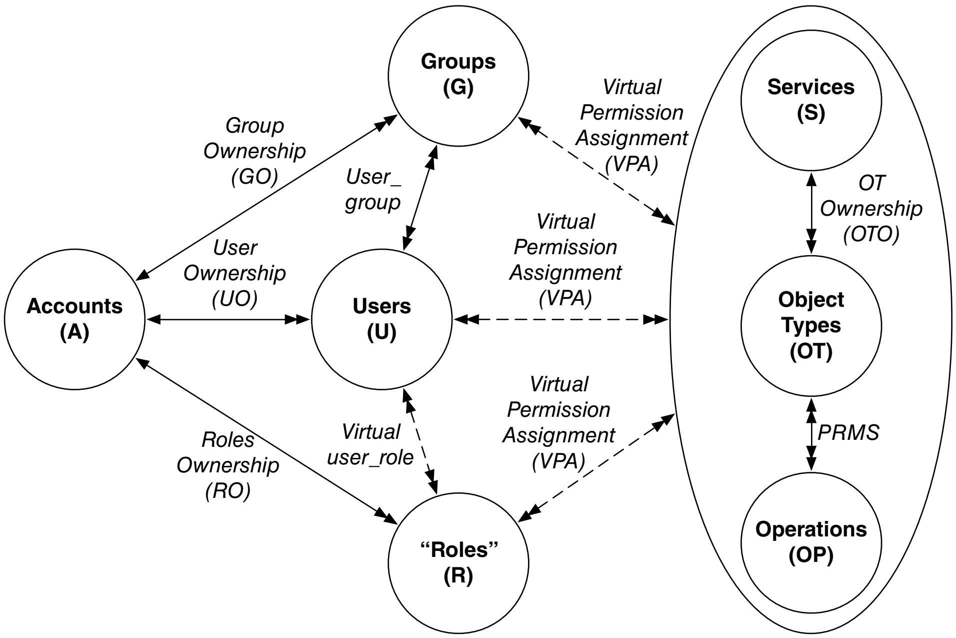

Figure 4.4 AWS access control within a single account.

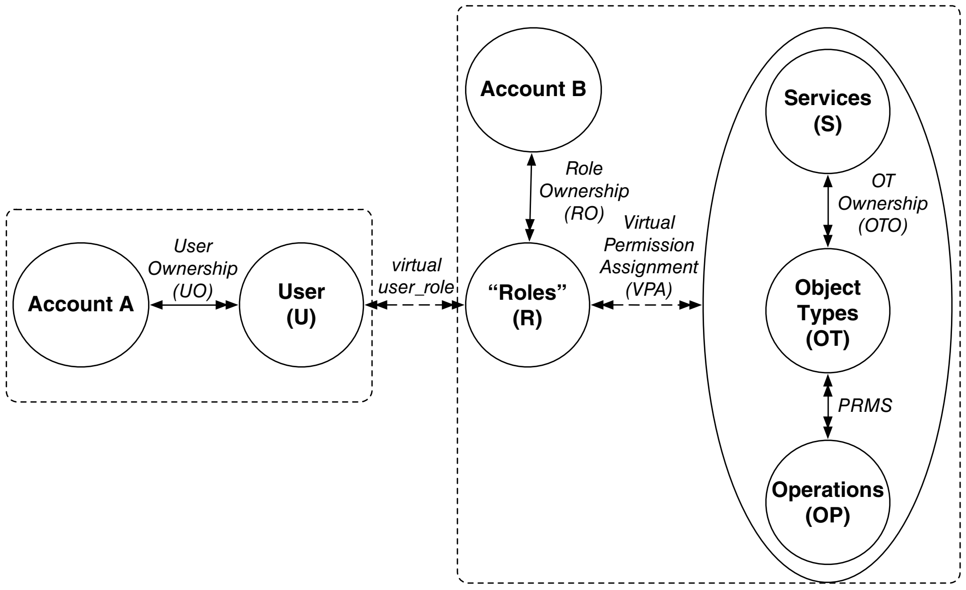

Figure 4.5 AWS access control across accounts (users in account A access services and resources in account B).

applies to all group members. For simplicity, we use the term *users* to represents both users and groups in the rest of this discussion.

- **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**—*Roles* in AWS are mainly used for cross-account permission purposes. However, roles can also be used for internal users in an account. Policy files can be attached to a role. Roles also define the trust relation between principals, which can be either AWS accounts or users. Users use roles through the *AssumeRole* action to access corresponding resources. To emphasize the difference between the usual concept of roles in role-based access control (RBAC) and roles in AWS, we use quotation marks around "Roles" in the figures.
- **Services**—*Services* refer to cloud services AWS provides to its customers. A 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 the compute service EC2, the object type is a VM; with the 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 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 4.5. In this and other figures, a thick arrow represents an action taken by a user to assume a role. 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 defined target resource.

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

Definition 4.3 AWS-AC Model Components

- A, U, G, R, S, OT, and OP are finite sets of existing accounts, users, groups, roles, services, object types, and operations, respectively, in an AWS public cloud system.
- User Ownership (UO) is a function UO: U → A, mapping a user to its owning account. Equivalently viewed as a many-to-one relation UO ⊆ U × A.
- Group Ownership (GO) is a function GO: G → A, mapping a group to its owning account. Equivalently viewed as a many-to-one relation GO ⊆ G × A.
- Roles Ownership (RO) is a function RO: R → A, mapping a role to its owning account. Equivalently viewed as a many-to-one relation GO ⊆ R × A.
- Object Type Owner (OTO) is a function OTO: OT → S, mapping an object type to its owning service. Equivalently viewed as a many-to-one relation OTO ⊆ OT × S.
- PERMS = OT × OP is the set of permissions.
- Virtual Permission Assignment (VPA) is a many-to-many virtual relation VPA ⊆ (U ∪ G ∪ R) × PERMS, resulting from policies attached to users, groups, roles, and resources.
- user_group ⊆ U × G is a many-to-many relation assigning users to groups, where the user and group must be owned by the same account.
- virtual_user_role (VUR) is a virtual relation VUR ⊆ U × R, resulting from policies attached to various entities. *AssumeRole* is an action allowing users to activate a role authorized in the VUR.

4.4.2 Secure Information and Resource-Sharing Model in AWS

In this section, we present an access-control model for AWS with the SID extension (AWS-AC-SID). We build the AWS-AC-SID model on top of the AWS-AC model to include SID functionality (Zhang et al. 2014). We present the AWSAC-SID model so as to cover only the additional components added to the AWS-AC model. Figure 4.6 shows the AWS-AC-SID model.

The additional components included in AWSAC-SID model are SID, SIP, EU, CP, and OP. These are described next:

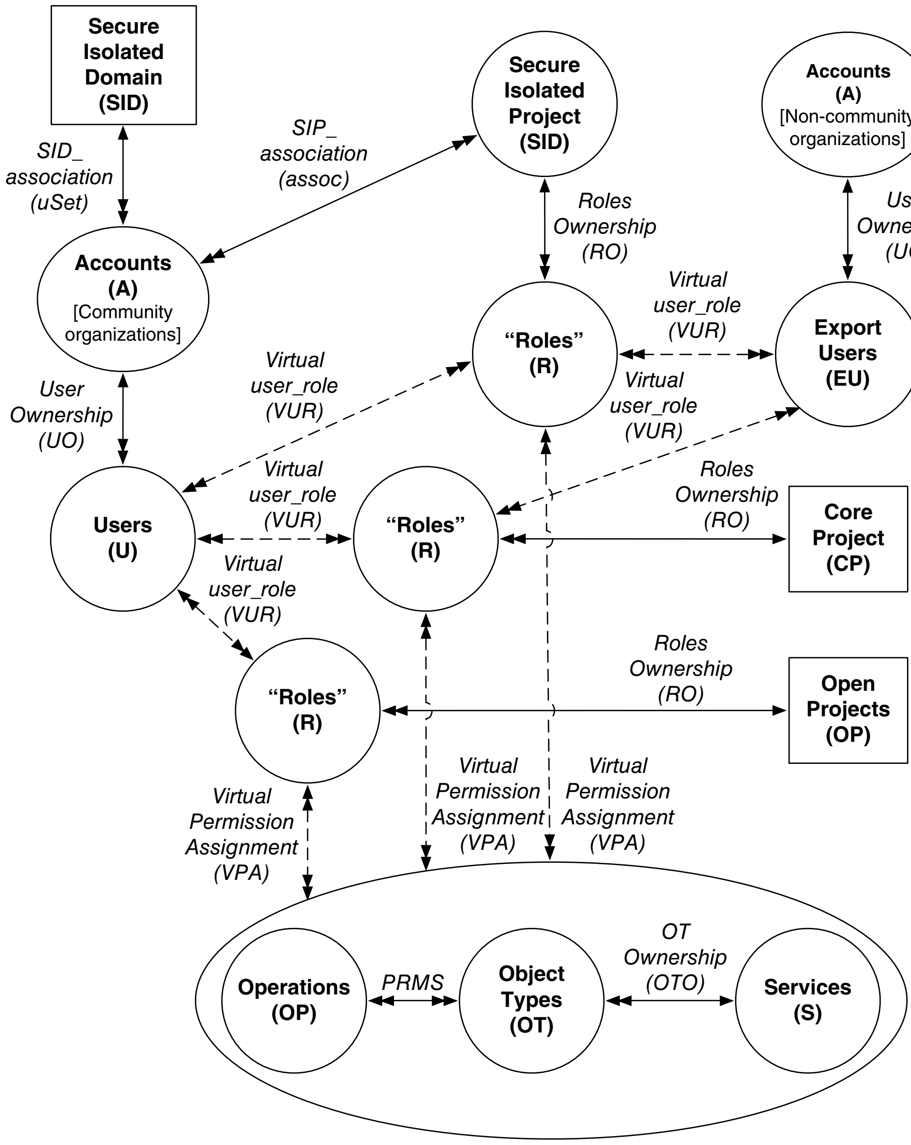

Figure 4.6 Amazon Web Services (AWS) Access Control model with SID extension (AWSAC-SID) (ignoring the groups entity).

- **Secure Isolated Domain (SID)**—A SID (Zhang et al. 2014) is a special domain holding security information and resources for cross-organizational security collaborations. The SID provides an administrative boundary for cybersecurity information and resource collection and analysis, and a secure isolated environment for cybersecurity collaborations in a community of organizations. The SID holds all SIPs designed for cyber-incident response and security collaboration within this community of organizations. SID also holds a CP and an OP, as shown in Figure 4.7.

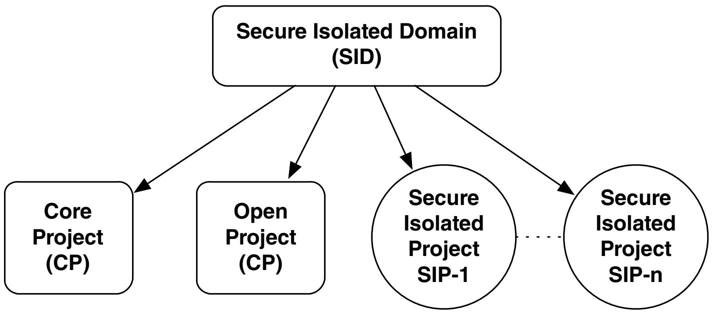
Secure Isolated Domain (SID)
Core Project (CP)
Open Project (CP)
Secure Isolated Project SIP-1
Secure Isolated Project SIP-n

Figure 4.7 SID composition.

- **Secure Isolated Project (SIP)**—A SIP (Zhang et al. 2014) is a special project with constraints over its user membership. It is used to collect, store, and analyze cybersecurity 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 Project (CP)**—A CP is a shared project holding the cybersecurity committee (Sandhu et al. 2011) for the community of organizations. Each organization in the community has at least one representative security user in the committee.
- **Open Project (OP)**—An OP is an open shared project where users from the community of organizations share common cybersecurity information and resources (Sandhu et al. 2011). It is a common forum for all community users to share general security information. Information published in the OP is public to every user in the project.
- **Expert Users (EU)**—To involve outside professionals, EUs (Sandhu et al. 2011) are introduced to the SID. EU don't belong to the community of organizations. They are from other professional security organizations in the same public cloud. These experts bring different cybersecurity skills. For instance, they may come from an IT consultant company that focuses on specific cyber attacks. They may be cybersecurity law-enforcement officers specializing in cybercrime. The involvement of EUs helps organizations handle cyber collaborations more effectively. The SID maintains an EU list that is available for collaboration inside the SID.

The following formalizes these concepts as well as the relationships among them.

Definition 4.4 AWS-AC-SID Model Components in Addition to the AWS-AC Model

- SIP, EU, and O are finite sets of SIPs, EUs, and objects.
- SID is a unique SID serving a community of organizations. The SID owns a CP, an OP, and a number of SIPs. The SID also maintains EU resources.
- SIP association (assoc) is a function assoc: SIP $\rightarrow$ 2A, mapping a SIP to all its member accounts/organizations.
- Object Ownership (OO) is a function OO: O $\rightarrow$ A, mapping an object to its owning account. Equivalently viewed as a many-to-one relation OO $\subseteq$ O $\times$ A. O is a resource that belongs to an account. We didn't include Object (O) and Object Ownership (OO) in Figure 4.6, since it mainly shows the administrative perspective of the model.

4.4.2.1 Administrative AWS-AC-SID Model

For proprietary products such as AWS, we cannot modify the cloud platform. SID functionality can be provided as a security service to all organizations in the SID community by a third party in AWS. The CP and OP are created with the SID. Each organization can join several SIDs with different communities of organizations. Each of these SIDs is isolated from the others.

The roles can be two types, administrative and member, which denote the permission of being able to manage users and permissions only for resources, respectively. The roles *CPadmin* and *SIPadmin* represent limited administrative power in the CP or a SIP, respectively, which gives the CP or SIP admin users permission to add and remove other users from their home account to the CP or a SIP. The roles *CPmember*, *OPmember*, and *SIPmember* represent operative permissions that can be given to normal users to access the CP, the OP, or a SIP. Since roles in AWS are local, *SIPadmin* and *SIPmember* are two sets of roles, separately representing the set of admin roles and the set of member roles in all SIPs; while *CPadmin*, *CPmember*, and *OPmember* are single roles in an account.

The administrative aspects of the AWS-AC-SID model are discussed informally below. A formal specification is given in Table 4.2.

Initially set up the SID: In the case of one SID serving one community of organizations, we can initially set up the SID with one CP and one OP. The member organizations of the SID are fixed. Let *uSet* denote a fixed group of security admin users from all organizations of the community, with one admin user for one organization. Each organization in the community has equal limited administrative power in the SID, which is carried through *uSet*. The SID maintains *uSet* as a core group of admin users in the SID. Only users from *uSet* later can dynamically create SIPs in the SID.

With the setting up of the SID, users in *uSet* automatically get limited administrative permission in the CP, represented by the role *CPadmin*. With this role, CP admin users can add and remove other users from their home account to the CP. The OP is open for all users from the community of organizations. No admin users are needed for the OP. All users can add themselves to the OP with the role *OPmember* as a normal member user.

Create a SIP—A SIP is created whenever there is a need for cyber collaboration among a subset of the community organizations. It might be because of a cyber incident, a collaborative security program, or a secure information-sharing program. A subset of the community of organizations' representative security admin users *subuSet* together creates a SIP. The creation of a SIP succeeds based on agreement among the subset of the community of organizations. Each organization in the SIP has equal limited administrative power, represented by a role in *SIPadmin*. The role gives SIP admin users permission to add and remove other users from their home account to the SIP. Organizations set up a SIP by sending the SIP-creation request to the SID manager account.

Delete a SIP—After the collaboration is finished, a SIP needs to be securely deleted. The delete command is issued by the same set of the security admin users (*subuSet*) who issue the SIP-creation request. All information and resources are securely deleted in the SIP. All users assigned to the SIP are removed from it.

Table 4.2 AWS-AC-SID administrative model.

Operation	Authorization Requirement	Update
SipCreate(subuSet, sip) /* *A subset of organization security admin users together create a sip* */	∀ u ∈ subuSet.(u ∈ *uSet*) ∧ sip ∉ SIP	assoc(sip) = $\cup_{u \in subuSet}$ UO(u) SIP′ = SIP ∪ {sip}
SipDelete(subuSet, sip) /* *The same subset of security admin users together delete a sip* */	∀ u ∈ subuSet.(u ∈ *uSet*) ∧ sip ∈ SIP ∧ assoc(sip) = $\cup_{u \in subuSet}$UO(u)	assoc(sip) = NULL SIP′ = SIP − {sip}
CpUserAdd(adminu, u) /* *CP admin adds a user from their home account to CP* */	adminu ∈ *uSet* ∧ u ∈ U ∧ UO(u) = UO(adminu)	VUR′ = VUR ∪ {(u, *CPmember*)}
CpUserRemove(adminu, u) /* *CP admin removes a user from CP* */	adminu ∈ *uSet* ∧ u ∈ U ∧ UO(u) = UO(adminu) ∧ (u, *CPmember*) ∈ VUR	VIR′ = VIR − {(u, *CPmember*)}
SIPUserAdd(adminu, u, r, sip) /* *Sip admin adds a user from their home account to SIP* */	adminu ∈ *uSet* ∧ UO(adminu) ∈ assoc.(sip) ∧ u ∈ U ∧ r ∈ *SIPmember* ∧ RO(r) = sip ∧ sip ∈ SIP ∧ UO(u) = UO(adminu)	VUR′ = VUR ∪ {(u, r)}
SIPUserRemove(adminu, u, r, sip) /* *Sip admin removes a user from SIP* */	adminu ∈ *uSet* ∧ UO(adminu) ∈ assoc.(sip) ∧ u ∈ U ∧ r ∈ *SIPmember* ∧ RO(r) = sip ∧ (u, r) ∈ VUR ∧ sip ∈ SIP ∧ UO(u) = UO(adminu)	VUR′ = VUR − {(u, r)}
OpenUserAdd(u) /* *Users add themselves to OP* */	u ∈ U ∧ UO(u) ∈ UO(*uSet*)	VUR′ = VUR ∪ {(u, *OPmember*)}
OpenUserRemove(u) /* *Users remove themselves from OP* */	u ∈ U ∧ UO(u) ∈ UO(*uSet*) ∧ (u, *OPmember*) ∈ VUR	VUR′ = VUR − {(u, *OPmember*)}
CpEUserAdd(adminu, eu) /* *CP admin adds an expert user to CP* */	adminu ∈ *uSet* ∧ eu ∈ EU	VUR′ = VUR ∪ {(eu, *CPmember*)}
CpEUserRemove(adminu, eu) /* *CP admin removes an expert user from CP* */	adminu ∈ *uSet* ∧ eu ∈ EU ∧ (eu, *CPmember*) ∈ VUR	VUR′ = VUR − {(eu, *CPmember*)}
SipEUserAdd(adminu, eu, r, sip) /* *SIP admin adds an expert user to SIP* */	adminu ∈ *uSet* ∧ UO(adminu) ∈ assoc(sip) ∧ eu ∈ EU ∧ r ∈ *SIPmember* ∧ RO(r) = sip ∧ sip ∈ SIP	VUR′ = VUR ∪ {(eu, r)}
SipEUserRemove(adminu, eu, r, sip) /* *SIP admin removes an expert user from SIP* */	adminu ∈ *uSet* ∧ UO(adminu) ∈ assoc(sip) ∧ eu ∈ EU ∧ r ∈ *SIPmember* ∧ RO(r) = sip ∧ (eu, r) ∈ VUR ∧ sip ∈ SIP	VUR′ = VUR − {(eu, r)}
CpCopyObject(u, o1, o2) /**Users copy objects from organization accounts to CP* */	o1 ∈ O ∧ 02 ∉ O ∧ UO(u) = 00(o1) ∧ u ∈ U ∧ (u, *CPmember*) ∈ VUR	O′ = O ∪ {o2} OO(o2) = CP
CpExportObject(adminu, o1, o2) /* *Admin users export objects from CP to organizations accounts* */	adminu ∈ *uSet* ∧ o1 ∈ O ∧ OO(o1) = CP ∧ o2 ∉ O	O′ = O ∪ {o2} OO(o2) = UO(adminu)

(*Continued*)

Table 4.2 (Continued)

Operation	Authorization Requirement	Update
SipCopyObject(u, r, o1, o2, sip) /* *Users copy objects from organization accounts to a SIP* */	o1 ∈ O ∧ o2 ∉ O ∧ UO(u) = OO(o1) ∧ u ∈ U ∧ r ∈ *SIPmember* ∧ RO(r) = sip ∧ (u, r) ∈ VUR ∧ sip ∈ SIP	O′ = O ∪ {o2} OO(o2) = sip
SipExportObject(adminu, o1, o2, sip) /* *Admin users export objects from SIP to organization accounts* */	adminu ∈ *uSet* ∧ UO(adminu) ∈ assoc(sip) ∧ o1 ∈ O ∧ OO(o1) = sip ∧ o2 ∉ O	O′ = O ∪ {o2} OO(o2) = UO(adminu)

Add/remove a user to/from a CP—CP 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 CP. All users added to the CP are existing users from an organization's account. The limited administrative users don't have permission to create new users. They can only add existing users to the CP. When users are removed from the CP, they lose access to corresponding information and resources in the CP, regardless of the ownership of the piece of information in the past.

Add/remove a user to/from a SIP—Users from *subuSet* who are assigned the role *SIPadmin* have limited administrative power in the SIP. They can add/remove users of their home accounts to/from the corresponding SIP due to a need for collaboration. Users lose access to information and resources after they are removed from the SIP. Administrative users in 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, or resources in a SIP.

Add/remove a user to an OP—Every user in the collaborative community of organizations is allowed to join the OP. Users in the OP have equal but limited permissions. They can share cyber data but have no control over other users. We use the role *OPmember* to represent this limited permission. Users add/remove themselves from their organizations to/from OP. Users cannot access and share any data once they leave the OP.

Add/remove an EU to/from a SIP—EUs are required when external cyber expertise needs to be involved. For instance, a cyber incident requires experts from security consultant companies, government cyber experts, cyber police, etc. SID services maintain a relationship with external expertise. EUs can be added/remove to/from CPs and SIPs as members. Users from *uSet* can request to add/remove EUs to/from the CP, while users from *subuSet* can request to add/remove EUs to/from a SIP. There are situations in which an existing EU in a SIP needs to be removed. For instance, the contract with a cyber-consultant company ends, or a cybersecurity agent finishes their task as part of cyber collaboration. In such cases, securely deleting an EU is necessary. After the EU is deleted, the user loses all access to information and resource in the SIP.

Copy data between organization accounts and a CP/SIP—Users can copy data from their home accounts to the CP or a SIP. Administrative users from *uSet* or *subuSet* can export data from the CP or a SIP to their home accounts.

4.5 Access Control in Azure Cloud IaaS

In this section, we introduce a model for the Microsoft Azure cloud and demonstrate its flexibility by extending the access-control model to include information and resource sharing. As we did for AWS, we assume that each organization from the cloud community has only one tenant that is an Azure account.

4.5.1 Azure Access-Control Model

In Azure, any user has the ability to create an Azure account. The user who creates an Azure account will be the owner and super-administrative user of that account. Local users created in an Azure Active Directory (AAD) can create their own Azure accounts that are isolated from the parent account. Azure has two main components to manage users' access to resources in the Cloud: AAD and Subscriptions (Sub). To use resources in Azure, a user has to be assigned to a subscription. AAD helps to manage users, including both local AAD users and other valid Microsoft users. Azure offers a form of RBAC wherein permissions are defined over cloud resources within roles in resource groups. Roles can then be assigned to users. Roles are predefined in Azure.

The Azure Access Control (Azure-AC) model has 14 entities: Accounts (A), Azure Active Directory (AAD), Subscription (Sub), Azure Active Directory Role (AADR), Azure Active Directory User (AADU), Non-Azure Active Directory User (NAADU), Group (G), Resource Group (RG), Role (R), Subscription Role (SubR), Resource (RS), Service (S), Object Type (OT), and Operation (OP), as shown in Figure 4.8:

- **Account (A)**—To have its own public cloud resources, an organization needs 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 (AAD)**—AAD is Microsoft's multitenant cloud-based directory and identity-management service. It provides a full suite of identity-management capabilities including multifactor authentication, device registration, self-service password management, privileged account management, RBAC, security monitoring, and so on. AAD also provides single sign-on (SSO) access to cloud SaaS applications. In addition, it can integrate with other identity-management solutions used in industry.
- **Subscription (Sub)**—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 Role (AADR)**—AADRs allow users to manage the directory and identity-related features. AAD has a set of administrative roles, including billing, global, password, service, and user administrator. Each of these administrative roles is designed for a different specific administrative purpose. It also has a normal user role, which has no administrative power.
- **Subscription Role (SubR)**—SubRs are a separate role set from AADRs. SubRs are administrative roles that give users permissions to manage cloud resources via a subscription. SubRs 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 AAD users. A service administrator cannot be a local AAD user from the same AAD assigned to that subscription.

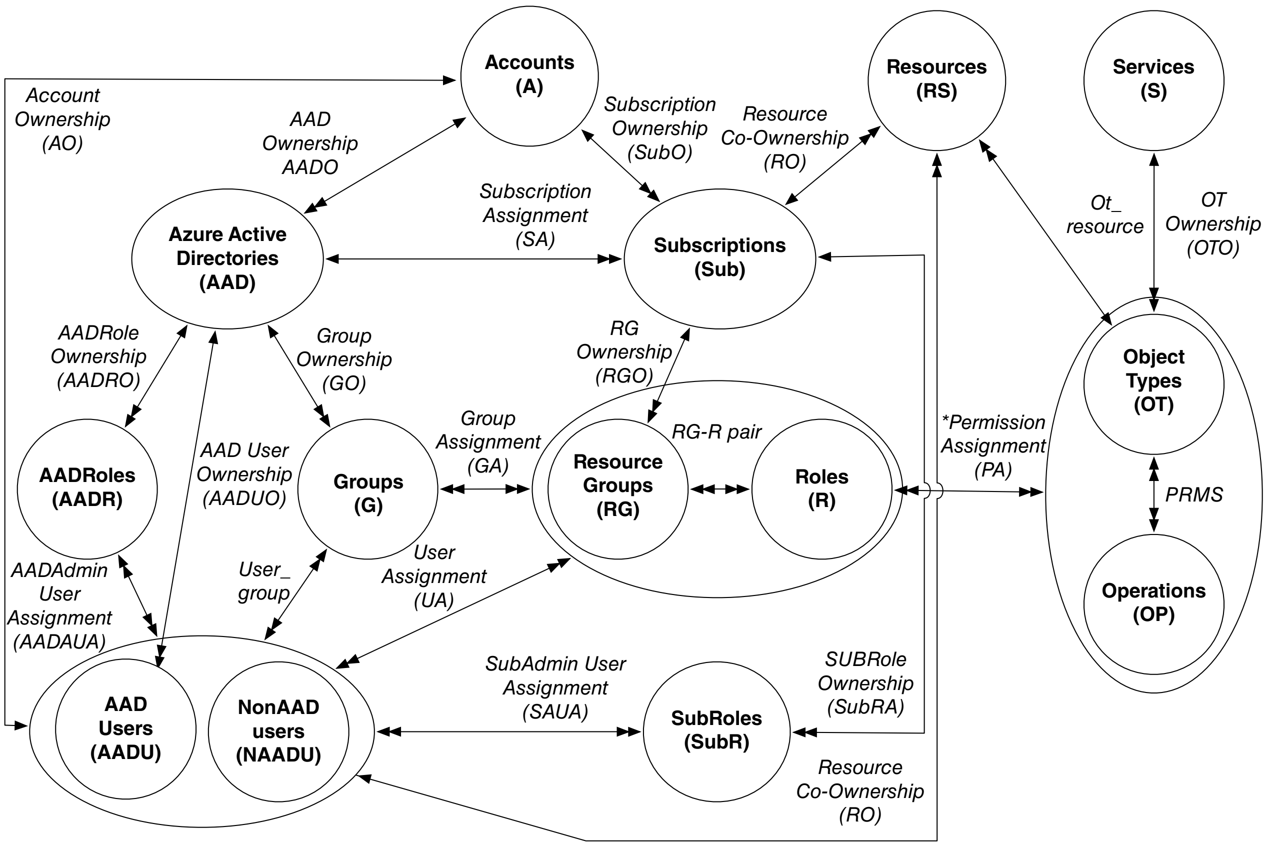

Figure 4.8 Azure Access Control (Azure-AC) model.

- **Azure Active Directory User (AADU)** and **Non-Azure Active Directory User (NAADU)**—These are 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. AADUs are users created in AAD. They can be administrative users of the directory or normal users. NAADUs are users not from the local AAD, but from partner organizations and other Microsoft users.
- **Group (G)**—A *group* is simply a set of users; it can include both AADUs and NAADUs. Groups belong to an AAD account. The existence of groups serves to allow the convenient management of multiple users as a single unit. Each policy attached to a group applies to all group members.
- **Resource Group (RG)**—RGs are logical resource containers that allow customers to add various cloud resources like databases, VMs, etc. RGs provides a way to monitor and control users' access to collections of cloud resources.
- **Role (R)**—Users are assigned to a RG with *roles* to get permissions to access cloud resources. Roles allow users to have permissions to access cloud resources: for instance, VMs, storage, networking, etc. Roles can be different collections of meta-permissions like read and write toward a specific resource. Roles can only be assigned to users inside a RG.
- **Resource (RS)**—*Resources* refer to cloud assets that can be owned by users. Cloud assets are cloud resources such as VMs, databases, storage, etc. Since the only way for users to access resources is through subscriptions, we also define that the subscription has ownership over the resources.
- **Service (S)**—*Services* refer to cloud services Azure provides to its customers. A CSP leases cloud resources to its customers in terms of services. Azure provides customers with services such as compute, storage, networking, administration, and databases.
- **Object Type (OT)** and **Operation (OP)**—An OT represents a specific type of object. From the CSP's viewpoint, objects are more like services. We define OTs as particular service types the Cloud provides. For instance, with the compute service, the OT is a VM; with the storage service, the OT is a storage container; etc.

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

Definition 4.5 Azure-AC Model Components

- A, AAD, Sub, RG, R, AADR, SubR, AADU, NAADU, G, RS, S, OT, and OP are finite sets of existing accounts, Azure Active Directories, subscriptions, resource groups, roles, Azure AD roles, subscription roles, AAD users, non-Azure AD users, groups, resources, services, object types, and operations, respectively, in an 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 AAD 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 → Sub, mapping a RG to its owning subscription. Equivalently viewed as a many-to-one relation GRO ⊆ RG × Sub.
- AAD User Ownership (AADUO) is a function AADUO: AADU → AAD, mapping a user to its owning AAD. Equivalently viewed as a many-to-one relation AADUO ⊆ AADU × AAD.
- Group Ownership (GO) is a function GO: G → AAD, mapping a group to its owning AAD. Equivalently viewed as a many-to-one relation GO ⊆ G × AAD.
- Azure AD Roles Ownership (AADRO) is a function AADRO: AADR → AAD, mapping a Azure AD role to its owning AAD. Equivalently viewed as a many-to-one relation AADRO ⊆ U × A.
- Resource Co-Ownership (RSO) is a function RSO: RS → Sub ∨ RS → (AADU ∪ NAAUD), mapping a piece of a resource to its owning subscription and user. Equivalently viewed as a many-to-one relation RSO ⊆ RS × Sub ∪ RS × (AADU ∪ NAAUD).
- Object Type Owner (OTO) is a function OTO: OT → S, mapping an OT to its owning service. Equivalently viewed as a many-to-one relation OTO ⊆ OT × S.
- Resource Group Role (RG-R) pair ⊆ GR × R is a many-to-many relation mapping RGs to roles.
- Subscription Assignment (SubA) is a many-to-one relation SubA ⊆ Sub × AAD.
- Subscription Roles Assignment (SubRA) is a many-to- many relation SubRA ⊆ Sub × SubR.
- AADAdmin User Assignment (AADAUA) is a many-to-many relation AADAUA ⊆ (AADU ∪ NonAADU) × AADR, mapping a user to a AAD.
- SubAdmin User Assignment (SAUA) is a many-to-many relation SAUA ⊆ (AADU ∪ NonAADU) × SubR. There is one exception to the SAUA relation in assigning a service admin to a subscription. Every subscription has only one service admin user assigned to it, while it can have up to 200 co-admin users assigned to it.
- User Assignment (UA) is a many-to-many relation UA ⊆ (AADU ∪ NonAADU) × RG-R, mapping a user to a RG role pair.
- Group Assignment (GA) is a many-to-many relation GA ⊆ G × RG.
- Permission Assignment (PA) is a many-to-many relation PA ⊆ (RG × R) × PREM, assigning RG role pairs to permissions. One thing we need to mention is that Azure has fixed sets of collections of permissions that users can choose from, instead of giving users the capability to define their own permission sets.
- user_group ⊆ U × G is a many-to-many relation assigning users to groups, where the user and group must be owned by the same account.
- ot_resource ⊆ OT × RS is a one-to-many relation mapping resources to OTs.
- PRMS = OT × OP is the set of permissions.

4.5.2 Secure Information and Resource-Sharing Model in Azure

In this section, we present an access-control model for Azure with the SID extension (Azure-AC-SID). We extend the Azure-AC model to include SID functionality (Zhang et al. 2014). We present the Azure-AC-SID model so as to cover only the additional components added to the Azure-AC model. Figure 4.9 shows the Azure-AC-SID model.

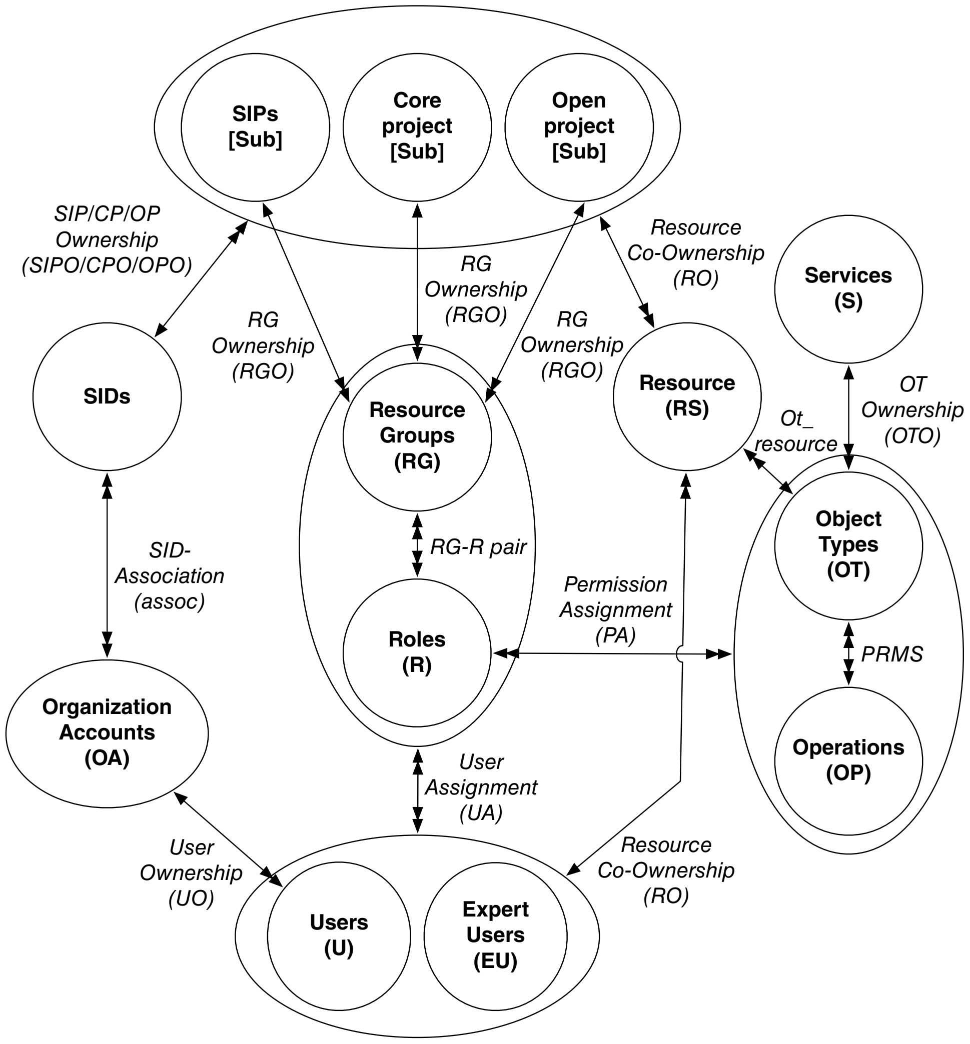

Figure 4.9 Azure Access Control model with SID extension (Azure-AC-SID) (ignoring the groups entity).

The following introduces the Azure-AC-SID model. The additional components included are Secure Isolated Domain (SID), Secure Isolated Project (SIP), Expert User (EU), User (U), Core Project (CP), and Open Project (OP):

- **Secure Isolated Domain (SID)**—The SID (Zhang et al. 2014) is a special domain, holding security information and resources for cross-organizational security collaboration. The SID provides an administrative boundary and a secure isolated environment for cybersecurity collaborations in a community of organizations. Each SID holds several SIPs designed for cyber-incident response and security collaboration among a group of organizations, a CP, and an OP for general secure information and resource sharing.

- **Secure Isolated Project (SIP)**—The SIP (Zhang et al. 2014) is a special project with limited user membership. It is used to collect, store, and analyze cybersecurity information for specific cyber incidents. A SIP provides an isolated, controlled environment for a group of organizations within the community to collaborate and coordinate on cyber incidents and other security issues. Subscriptions provide isolated resource containers for different projects to use. Thus, we design projects using subscriptions.
- **Core Project (CP)**—The CP is a shared project holding the cybersecurity committee (Sandhu et al. 2011) for the community of organizations. Each organization in the community has representative security users in the committee. CPs handle routine security tasks for the community.
- **Open Project (OP)**—The OP is an open shared project where users from the community of organizations share common cybersecurity information and resources (Sandhu et al. 2011). It is a common forum for all organizational users in the community to share general security information. Information published in the OP is public to every user associated with the subscription.
- **Expert User (EU)**—EUs (Sandhu et al. 2011) are external non-organizational professionals. They don't belong to the group of organizations. They are from other professional security organizations that bring different cybersecurity skills. They could be from IT consultant companies or from government cybersecurity law-enforcement departments. A SID maintains an EU list that is available to any project inside the SID.
- **User (U)**—Users include both AADUs and NAADUs, which refer to either Microsoft users or partner organization users. We use one User entity to represents all users that are allowed to access cloud resources, since from the standpoint of SID functionality, as long as the user is associated with the organization's AAD, it does not care where the user comes from.
- **Organization accounts**—Organization accounts represent organizations in the community. They can be either AAD accounts or organizations enterprise accounts that are identified by AAD. Organization accounts allow organizations to own a specific amount of (virtual) cloud resources.

The following formalizes these concepts, as well as the relationships among them.

Definition 4.6 Azure-AC-SID Model Components in Addition to the Azure-AC Model

- SID, SIP, CP, OP, EU, U, and O are finite sets of SIDs, SIPs, CPs, OPs, EU, users, and objects. The SID serves communities of organizations. A SID owns a CP, an OP, and a number of SIPs. A SID also maintains EU resources.
- CP/OP/SIP ownership (CPO/OPO/SIPO) is a function CPO/OPO/SIPO: CP/OP/SIPO $\rightarrow$ SID, mapping a single CP/OP/SIP to its owning SID, which equals mapping a specific subscription to a SID.
- SID association (assoc) is a function assoc: SID $\rightarrow 2^{A}$, 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 $UO \subseteq U \times OA$.
- Object Ownership (OO) is a function OO: $O \rightarrow OA$, mapping an object to its owning organization account. Equivalently viewed as a many-to-one relation $OO \subseteq O \times OA$.

4.5.2.1 Administrative Azure-AC-SID Model

Similar to the AWS-AC-SID model, each SID has a CP and an OP as a security service provided to all organizations in the SID community. The CP and OP are created with the SID. Each organization can join different SIDs with different communities of organizations. Each of these SIDs is isolated from the others. We only discuss the model in which the SIDs are manually set up, serving different communities of organizations in the Azure public cloud.

We design a SID manager as an automated agent that serves cloud communities of organizations and manages SIDs and their constituent components throughout their life cycle. The SID manager processes SID requests from communities of organizations and maintains a separate SID for each community. Within each SID, it facilitates the creation and deletion of SIPs. Each time a cyber-collaboration request is sent to the SID manager, it creates a new subscription, assigning the subscription to the group of organizations that made the request. After the collaboration is done, the SIP is deleted.

Considering that Azure already has dedicated roles for managing subscriptions and AAD, we will use those existing AAD administrative roles and subscription roles to manage SIPs, the CP, and the OP in a SID. Azure provides five AAD admin roles and two subscription admin roles. For simplicity, we will constrain the administrative roles to include only the AAD global admin role and subscription co-admin role. Azure also provides a set of operative roles in RGs, which allows users to have permission to access cloud resources.

To make role assignment simple and clear, we constrain roles to be two types, administrative roles and member roles, which denote the permission of being able to manage users and permissions only for accessing cloud resources. We use the admin role *SIDAdmin* to represent all admin permissions a user can get from AAD and subscriptions. We use the member role *SIDmember* to represent all normal roles a user can have in a RG. Admin users have the capability to add and remove other users from their home organizations to a CP subscription or a SIP subscription. Member users can be added/removed from/to a project subscription inside a SID. Member users are those who have access to real cloud services and resources, like creating or deleting a VM.

The administrative aspects of the Azure-AC-SID model are discussed informally next. A formal specification is given in Table 4.3.

Initially set up the SID: For every community of organizations that will have cyber collaboration, we offer one SID associated with the community. The number of organizations associated with the SID is fixed. Let *uSet* denotes the fixed group of security admin users, each of which represents 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*. The SID maintains *uSet* as a core group (Sandhu et al. 2011) of SID admin users. Only users from *uSet* later can dynamically create SIPs in the SID.

Inside the SID, organizations can request multiple SIPs for the convenience of different cyber collaborations. The number of SIPs depends on how much collaboration is initialized by the group of organizations. A SID is initially set up with a CP and an OP, and organizations can then automatically request to create and delete SIPs, as well as add or remove users to/from SIPs. With the initialization of a SID, admin users from *uSet* automatically get limited administrative permission in a CP in a SID, which is represented by role *SIDadmin*. Normal users from the community automatically get permissions to be able to add them to the OP with role the *SIDmember*.

Table 4.3 Azure-AC-SID administrative model.

Operation	Authorization Requirement	Update
SipCreate(uSet, sip, sid) /* *A set of organization security admin users together create a sip* */	$\forall$ u $\in$ uSet.(u $\in$ *uSet*) $\wedge$ sip $\notin$ SIP	assoc(sid) = $\cup_{u \in uSet}$UO(u) SIPO(sip) = sid SIP′ = SIP $\cup$ {sip}
SipDelete(subuSct, sip, sid) /* *The same subset of security admin users together delete a sip* */	$\forall$ u $\in$ subuSet.(u $\in$ *uSet*) $\wedge$ sip $\in$ SIP $\wedge$ assoc(sid) = $\cup_{u \in subuSet}$UO(u) $\wedge$ SIPO(sip) = sid	assoc(sid) = NULL SIPO(sip) = NULL SIP′ = SIP − {sip}
UserAdd(adminu, u, p, sid) /* *Admin users add a user from their home account to a Cp/Sip* */	adminu $\in$ *uSet* $\wedge$ u $\in$ U $\wedge$ UO(u) = UO(adminu) $\wedge$ p $\in$ (CP $\cup$ SIP) $\wedge$ (CPO(p) = sid $\cup$ SIP(p) = sid)	UA′ = $\exists$ rg $\in$ p.(UA $\cup$ {(u, [rg, *SIDmember*])})
UserRemove(adminu, u, p, sid) /* *Admin users remove a user from a Cp/Sip* */	adminu $\in$ *uSet* $\wedge$ u $\in$ U $\wedge$ UO(u) = UO(adminu) $\wedge$ p $\in$ (CP $\cup$ SIP) $\wedge$ (CPO(p) = sid $\cup$ SIP(p) = sid) $\wedge$ $\exists$ rg $\in$ p.(UA $\cup$ {(u, [rg, *SIDmember*])})	UA′ = UA − {(u, [rg, *SIDmember*])}
OpenUserAdd(u, op, sid) /* *Users add themselves to a Op* */	u $\in$ U $\wedge$ UO(u) $\in$ UO(*uSet*) $\wedge$ op $\in$ OP $\wedge$ OPO(op) = sid	UA′ = $\exists$ rg $\in$ op.(UA $\cup$ {(u, [rg, *SIDmember*])})
OpenUserRemove(u, op sid) /* *Users remove themselves from a Op* */	u $\in$ U $\wedge$ UO(u) $\in$ UO(*uSet*) $\wedge$ op $\in$ OP $\wedge$ OPO(op) = sid $\wedge$ $\exists$ rg $\in$ op.(UA $\cup$ {(u, [rg, *SIDmember*])})	UA′ = UA − {(u, [rg, *SIDmember*])}
ExpertUserAdd(adminu, eu, p, sid) /* *Admin users add an expert user to a Cp/Sip* */	adminu $\in$ *uSet* $\wedge$ eu $\in$ EU $\wedge$ p $\in$ (CP $\cup$ SIP) $\wedge$ (CPO(p) = sid $\cup$ SIPO(p) = sid)	UA′ = $\exists$ rg $\in$ p.(UA $\cup$ {(eu, [rg, *SIDmember*])})
ExpertUserRemove(adminu, eu, p, sid) /* *Admin users remove an expert user from a Cp/Sip* */	adminu $\in$ *uSet* $\wedge$ eu $\in$ EU $\wedge$ p $\in$ (CP $\cup$ SIP) $\wedge$ (CPO(p) = sid $\cup$ SIPO(p) = sid) $\wedge$ $\exists$ rg $\in$ p.(UA $\cup$ {(eu, [rg, *SIDmember*])})	UA′ = UA − {(eu, [rg, *SIDmember*])}
CopyObject(u, o1, o2, p) /* *Users copy objects from organization accounts to a Cp/Sip* */	o1 $\in$ O $\wedge$ o2 $\notin$ O $\wedge$ UO(u) = OO(o1) $\wedge$ u $\in$ U $\wedge$ p $\in$ (CP $\cup$ SIP) $\wedge$ 3 rg.((u, [rg, *SIDmember*]) $\in$ UA)	O′ = O $\cup$ {o2} OO(o2) = p
ExportObject(adminu, o1, o2, p) /* *Admin users export objects from a Cp/Sip to organization accounts* */	adminu $\in$ *uSet* $\wedge$ o1 $\in$ O $\wedge$ o2 $\notin$ O $\wedge$ OO(o1) = p $\wedge$ p $\in$ (CP $\cup$ SIP) $\wedge$ $\exists$ rg.((adminu, [rg, *SIDadmin*]) $\in$ UA)	O′ = O $\cup$ {o2} OO(o2) = UO(adminu)

Create a SIP: A set of security admin users *uSet* together creates a SIP for cyber collaboration among the community of organizations. The creation of a SIP succeeds based on agreement among the community of organizations. Each organization in the SIP has equally limited administrative power, which is represented by the role *SIDadmin*.

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

Add/remove a user to/from a CP or SIPs: CP 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 CP and SIPs. All the users added to the CP or SIPs are existing users from an organization's account. The limited administrative users don't have permission to create new users or delete an existing user. They can only add existing users to the CP or SIPs. When users are removed from the CP or a SIP, they lose access to corresponding information and resources in the CP or the SIP, regardless of the ownership of the piece of information in the past. Admin users in the CP 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, or resources in a CP or a SIP.

Add/remove a user to/from an OP: Every user in the collaborative community of organizations is allowed to join the OP. Users in the OP have equal but limited permissions. They can share cyber data but have no control over other users. We use the role *SIDmember* to represent this limited permission. Users add/remove themselves from their organizations to/from the OP. Users cannot access and share any data once they leave the OP.

Add/remove an EU to/from a CP or SIPs: EUs are required when external cyber expertise needs to be involved. For instance, a cyber incident needs experts from security consultant companies, government cyber experts, cyber police, etc. SID services maintain a relationship with external experts. EUs can be added/removed to/from a CP and SIPs as members. Users from *uSet* can request to add/remove EUs to/from the CP or a SIP. An existing EU in the CP or a SIP can also be removed. For instance, at the end of a cyber collaboration, an unneeded EU is securely deleted. After the EU is deleted, the user loses all access to any information and resources in the CP or a SIP.

Copy data between organization accounts and a CP or SIPs: Users can copy data from their home accounts to the CP or a SIP. The administrative users from *uSet* can export data from the CP or a SIP to their home accounts.

4.6 Conclusions

In this chapter, we introduced access-control models for OpenStack, AWS, and Microsoft Azure cloud IaaS platforms. We identified fundamental elements of access control in cloud IaaS. We also explored models for information and resource sharing in cybersecurity to show the flexibility of those cloud access-control models. We designed these models mainly based on the concept and architecture of the cloud platforms. We gave formal descriptions of administrative models, which provide a clear specification of how the users and resources are managed and controlled in the model.

References

E. Cohen, R. K. Thomas, W. Winsborough, and D. Shands. (2002). Models for coalition-based access control (CBAC). In: Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies, 97–106. New York: ACM.

Hassan, Q. (2011). Demystifying cloud computing. *The Journal of Defense Software Engineering (CrossTalk)* 24 (1): 16–21.

R. Krishnan, R. Sandhu, J. Niu, and W. Winsborough. (2009). Towards a framework for group-centric secure collaboration. In: 5th International Conference on Collaborative Computing, Networking, Applications and Worksharing, 1–10. Piscataway, NJ: IEEE.

P. Mell and T. Grance. (2011). The NIST definition of cloud computing. NIST Sp. Pub. 800-145.

L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke. (2002). A community authorization service for group collaboration. In: 3rd IEEE International Workshop on Policies for Distributed Systems and Networks, 50–59. IEEE.

R. Sandhu, K. Z. Bijon, X. Jin, and R. Krishnan. (2011). RT-based administrative models for community cyber security information sharing. In: 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing, 473–478. IEEE.

Shands, D., Yee, R., Jacobs, J., and Sebes, E.J. (2000). Secure virtual enclaves: Supporting coalition use of distributed application technologies. In: *IEEE DARPA Information Survivability Conference and Exposition*, 335–350. IEEE.

B. Tang and R. Sandhu. (2014). Extending OpenStack access control with domain trust. In: 8th International Conference on Network and System Security (NSS), 54–69.

Y. Zhang, R. Krishnan, and R. Sandhu. (2014). Secure information and resource sharing in cloud infrastructure as a service. In: Proceedings of the 2014 ACM Workshop on Information Sharing & Collaborative Security, 81–90. New York: ACM.

Y. Zhang, F. Patwa, R. Sandhu, and B. Tang. (2015). Hierarchical secure information and resource sharing in openstack community cloud. In: IEEE Conference on Information Reuse and Integration (IRI), 419–426. IEEE.

Y. Zhang, F. Patwa, and R. Sandhu. (2015). Community-based secure information and resource sharing in AWS public cloud. In: IEEE Conference on Collaboration and Internet Computing (CIC), 46–53, IEEE.