Abstract

Service Oriented Computing (SOC) has provided a promising architectural for business collaboration. However, challenging security issues concerning business collaboration have arisen because of its dynamic and loosely coupling nature. An important problem is how to specify security policies that belong to different application domains and integrate them to satisfy the requirements of the collaboration. In this paper we will provide analysis on authorization policy requirements for business collaboration, collaboration patterns and various security comparability and integration issues. We believe it is a first step toward a framework for modeling, specifying and handling authorization control for business collaboration.

1. Introduction

Nowadays business need to be collaborative and dynamic to stay competitive. In the presence of different security policies from parties in the collaboration, which may not ‘comparable’ with each other, the challenge we are facing is how security policy can be specified, managed and integrated.

Let us take an example in the area of Health Care. Patient grants access right to a physician in a medical centre on patient’s health record. Later on, the physician wants to grant access right to a medical research institute. How can we control this type of service propagation problem with security policies?

Policy inconsistencies among participants are another problem area. For example, a patient who holds an Overseas Students Health Cover wants to book an appointment with a physician, the physician wants to collaborate with a radiology institute for this patient. OSHC is acceptable for physician, but it is not the case with that radiology institute. Are these two organizations able to collaborate in the presence of security policy conflicts? If both parties do want to automate this type of collaboration, how will they come out with an integrated security policy?

How to dynamically achieve a security agreement for different collaboration models is a real challenge. To address these problems, we need a framework that could model and specify security policies from participating parties so that policies can be compared, verified and integrated. In this paper we proposed an initial framework which addresses the modeling issues for security policies in the following aspects:

From User Aspect:
Our framework modeled security policies associate with users by using elements such as role, credential and privilege, which specify access rights granted to a role and requirements for playing the role.

From Service Aspect:
We addressed constraints to access certain services by associate conditions to services and privileges. In the proposed framework, we can specify conditions attached to a service for certain privileges.

From Business Collaboration Aspect:
The framework also identifies different collaboration types, based on which different security policies compatibility and integration issues are thoroughly analyzed.

The following section explains elements that need to be included in security policy.

2. Security Agreement Elements

As we mentioned above, organizations have their own security policies. We suppose that before organizations can start the collaboration with each other, certain forms of agreement must be established among all the participants regarding each other’s
security policies. We call it security policy agreement. It also can be a process of integrating all participants’ security policies. For this purpose, we define some elements included in an agreement:

**Roles**
A role is a job function within the context of an organization with some associated semantics regarding the authority and responsibility conferred on the user assigned to the role. [5] (E.g. “doctor” is a role, John as a user can play the role of “doctor”).

**Privileges**
We use privileges and permissions interchangeably in the paper. Privilege is an approval to perform an operation on one or more objects, such as: read, write, forward, delete etc. In role based access control, permissions are assigned to roles rather than individuals [5], for example, security policy will specify that “Doctor” has “update” privilege on patient health records.

**Credentials**
Credentials are signed assertions describing attributes of the owner [1]. Examples of credentials are digital certificate, identity number etc. In our framework, a user need disclose certain credentials to be able to carry a “role”. In our model, role has permissions and required credentials. The requester has to become certain role to gain privileges on certain services, and the requester has to submit required credentials to become a role.

**Services**
Services are self-contained processed, simple services may provide simple functions such as inventory checking, complex services could provide a whole range of automated processes such as insurance liability services [3]. Services could have conditions associate with it.

**Conditions**
Conditions are special agreements associate with services and required privileges. When system makes decisions about authorization, it does not only check credential of requester, but also conditions that need to be met before access are granted. There are two types of conditions: provision and obligation [4]. Provisions are conditions that need to be satisfied before authorization is granted; obligations are conditions that must be fulfilled after the decision [4]. Not all services have conditions. Moreover, conditions are also associated with certain privileges. The relationships between these elements are illustrated in Figure 1.

In the proposed framework, we identify three types of policies in an organization, each of which focuses on different security aspect, and should be defined separately. We defined policies as follows:

- **Policy → RSP | SSP | OSP**
- **RSP → Privileges + Service + Credentials**
- **SSP → Service + Privileges + Conditions**
- **OP → Organizational general security policy**

Role security policy (RSP), which specify privileges and required credentials of each role.

Service security policy (SSP), which specifies which roles have access right to the service; this information is corresponding to service information in role security policy. Most importantly, it specifies conditions associated with the specific privileges of the service access.

Organization general security policy (OSP): since organizations as a whole, might have some generic security rules applied to all circumstance as well, such as: X.509 is the only accepted certificate format in the organization.

3. **Categorization of Collaboration Types**

Business collaborations consist of complex relationships and interactions among organizations. Different collaboration types may require different policy integration strategy. In Figure 2, we conclude four types of collaboration between organizations under the context of Web Services and provided simple examples in Health Care domain.
Type A depicts the most basic scenario in collaboration that involves two organizations. In figure 2, health Cover Company wants to access patient treatment record from a medical center for claim validation purpose. We call this type as a simple service.

Collaboration Type B is a scenario of service propagation: organization B granted the access right to a service provided by organization A, and later on, organization B forwards this access right granted from organization A to organization C.

Types C1 and C2 are both service composition. It is the situation when organization A wants to combine their own services or parts of services with services provided by organization B, and then provides this combined service to third parties. C1 and C2 depict two possibilities exist in service composition:

**C1: Service collaboration with an agent**

Organization C works as an agent, accessing services from different providers. For instance, Medical Information Portal provides various IS services to Health Care staff. (E.g. Electronically Assisted Prescription system) Services available in the portal are from different health care units or research institutes. Each unit containing medical resources is autonomous and has controls over its data. Each service provider and the Portal have their own security policies for different services. They collaborated via the Portal.

**C2. Joined collaboration without an agent:**

Organization A and B provide a composite service by combine their services or part of their services together directly without involvement of an agent.

**4. Compatibility Check**

Compatibility check is the first step in security policy integration since organizations have their own interest and priorities, which may cause inconsistency when it comes to collaboration. In order to resolve inconsistencies in security policies, we have to firstly identify the possible relationships between them.

We identified different collaboration types in previous section. Collaboration type could affect requirements for compatibility. We introduce two concepts here to identify different requirements for compatibility check.

**4.1 Compatible and Integratable**

Compatible:

Compatible we discussed here means security policies of participate organizations are congenial and...
potential violation to security policies of service owner are not exist. Suppose P1 and P2 are two set of security policies, we say P2 is compatible to P1 if the relationships between P1, P2 are equivalent or P1 Includes P2.

**Integratable:**
Different to Compatible, potential violation may acceptable for “Integratable”, as long as security policies of different organizations have some common properties. Integration still can be achieved based on some common points and collaboration may proceed. We say P1 and P2 are Integratable as long as they share some common properties.

4.2 Compatibility Check for Types of Collaborations

**Type A. Basic Access**
This is a simple request – access relationship, the focus is to satisfy related security policies of service owner. Basic Access type needs properties of requester match requirements of service owner, security policies between two organizations should be compatible.

**Type B. Service Propagation**
Service Propagation normally requires collaborating parties have compatible security policies. We can illustrate this by an example: Medical centre may have security policy stating that Patient Records can only be accessed by physician and attending Pathologist, forwarding is only allowed to Emergency room in Hospital. Collaboration can only happen if the pathology institute has equivalent security policy regarding to Patient Records or their policy is a subset of policy of Medical centre. The other way round, if the policy in pathology institute states that forwarding is also allowed to researchers. Service propagation collaboration unlikely success since as long as the Medical centre grant access right to pathologist, the pathologist could forward patient health records to researchers. This is not acceptable for the Medical centre. Any possible actions that may violate the policies of service owner should be restricted.

In order to prevent unwanted “forwarding” from happening, participant’s security policy should be stricter than the policies of service owner. Therefore, enforcing policies of collaborating parties compatible with policy of service owner is essential.

**Type C. Service Composition**
Combining existing services to provide value added services is the main drive of SOC. The premise of service composition is that there are common properties among participants’ policies. It unlikely we can combine services that have completely different policies. However, service composition is different to service propagation since collaboration could establish based on common properties of services. For example, Physician and attending pathologist can access patient records in medical centre; physician, attending pathologist and researchers can access patient records in Pathology institute. If two units collaborate to provide joint services, it is only reasonable to assume that only physician and attending pathologist can access patient records, not researchers.

Therefore, what we need check for composite services is not whether they are compatible, but rather whether they are Integratable, which requires policy relationships be Equivalent, Inclusive, and Intersection since integration could base on some common properties.

We analyzed security policy requirements for different collaboration patterns in this section. Now the question is how can we determine relationships between security policies? We discuss this in more detail in the following section.

5. Relationships in Policy Groups

As we identified in section 2, there are three types of security policies that should be included in an organization and each of them consist of basic elements. To identify relationship of security policies between two organizations, we need to first identify relationships between these policies and elements that make up of these policies. We use bags and sets to represent policies and defined four types of relationships for each policy type based on criteria we need for policy compatibility and integratability check. The relationships between policies are built on the relationships between the basic elements.

**Definition5.1 Organization General Security Policy (OSP)**

OSP specify organization level general security requirements, which are the ones that all requests must meet in order to access any services provided by the organization. OSP is a bag of tuples which contains multiset of general security requirements. There are two components included in OSP, opn and opv.
• opn is parameter of a security requirement, e.g. Digital certificate, encryption method...
• opv is specification of the parameter, e.g. X.509 etc. Each security requirement (opn) could have many specifications (opv).

Example:
osp = [{Digital certificate {X.509}}, {Encryption methods {RSA, DES}}]

This example depicts organizations ABC has an organization level general security policy: anyone wants to access ABC must have a valid X.509 certificate.

Let osp1 and osp2 be Organization general security policies of two organization and osp1 is osp of service owner.
osp1 = {opni, opvi}; osp2 = {opnj, opvj}
opvi ∈ {opv1, ..., opvn}, i ∈ N
opvj ∈ {opv1, ..., opvn}, j ∈ N

Definition 5.1.1 Equivalent (osp). We say osp1 and osp2 are Equivalent, if all the policy parameters (opn) in osp1 have equivalent parameters in osp2; all the specifications (opv) of the same parameter are equivalent and represent the same world concept. Assume corropn is correlation between opni and opnj:
coropni = opni/ropni; For ∀ corropni=1,
osp1.opni.opvi = osp2.opnj.opvj

Definition 5.1.2 Inclusive (osp). We say osp1 and osp2 are Inclusive if policy parameters of osp2 are subset of osp1’s, and specifications of same parameters in osp2 are subset of osp1’s:
osp1.opni ⊇ osp2.opnj;
coropni = opni/ropni; For ∀ coropni=1,
osp1.opni.opvi ⊇ osp2.opnj.opvj

Definition 5.1.3 Intersection (osp). osp1 and osp2 are Intersection if osp1 and osp2 have some common parameter and specifications of the same parameter are equivalent or there are some common specifications; OR Policy parameters of both policies are equivalent; there are some common specifications for same parameter:
osp1.opni = osp2.opnj;
coropni = opni/ropni; For ∀ coropni=1,
osp1.opni.opvi ∩ osp2.opnj.opvj ≠ ∅; OR
osp1.opni ∩ osp2.opnj ≠ ∅
coropn = opn1/opn2. For ∀ coropn=1,
osp1.opn1.opv1 ∩ osp2.opn2.opv2 ≠ ∅; OR
osp1.opn1.opv1 = osp2.opn2.opv2

Definition 5.1.4 Disjoint (osp). osp1 and osp2 are Disjoint if there are no any common policy parameters between them.
For ∀ osp1.opni, i = 1, ..., n, ∃ k, k = 1, ..., n:
osp1.opni ≠ osp2.opnk;

Definition 5.2 Role Security Policy (RSP)

RSP is a bag of tuples which contains multiset of role security requirements. Role security policies clarify the requirements to become a certain role and privileges assigned to the role for specific service. Role security policy is a bag rsp = (r, PS, CR) where:
• r is a role identifier;
• PS is a subbag which includes privilege pr (e.g. browse, forward…) assigned to the role and services set Si associated with each privilege;
• CR is another subbag which includes credential parameters crn (e.g. ID, membership number…) required by that role and set of specifications CV (e.g. p passport, driver’s license for “ID”) associated with each credential parameter.

Example:
rsp = (Physician, {{browse, {patient health record, drug database}}}, {update, {patient health record}}, {forward, {patient health record}}}, {{ID, {driver’s license number, Medicare number}}, {professional number, {physician registration number}}})

This role security policy states out: to become a role of physician, requester needs to provide valid driver’s license number, Medicare number and a physician registration number; a physician role has privilege to read, update and forward patient health records, he can also read drug database.
Definition 5.2.1 Equivalent (rsp). We say rsp1 and rsp2 are equivalent if:

- All the roles in rsp1 can find same roles in rsp2 and visa versa;
- Privileges assigned to same role are same in rsp1 and rsp2;
- Services associate with each privilege are same in both organizations;
- Credentials required for each role are same in both organizations and specifications for each credential requirement are same as well.

For $\forall r_{sp2,j}, j = 1, \ldots, n$, $\exists k, k = 1, \ldots, n$:
$$r_{sp2,j} = r_{sp1,k};$$

For $\forall r_{sp1,i}, i = 1, \ldots, n$, $\exists k, k = 1, \ldots, n$:
$$r_{sp1,i} = r_{sp2,k};$$

$\overline{cor_{rr}} = r_{sp1,i}/r_{sp2,k}$; for $\forall \overline{cor_{rr}} = 1$,
$$PS_{i}.pr_{i} \supseteq PS_{j}.pr_{j} \oplus PS_{i}.pr_{i} \subseteq PS_{j}.pr_{j};$$

$\overline{cor_{P}} = PS_{i}.pr_{i}/PS_{j}.pr_{j}$; for $\forall \overline{cor_{P}} = 1$,
$$PS_{i}.Si \supseteq PS_{j}.Sj \oplus PS_{i}.Si \subseteq PS_{j}.Sj;$$

$\overline{cor_{C}} = CR_{i}.crn_{i}/CR_{j}.crn_{j}$; for $\forall \overline{cor_{C}} = 1$,
$$CR_{i}.CV_{i} \supseteq CR_{j}.CV_{j} \oplus CR_{i}.CV_{i} \subseteq CR_{j}.CV_{j}.$$

Definition 5.2.2 Inclusive (rsp). We say rsp1 includes rsp2 if:

- Roles in rsp2 are proper subset of Roles in rsp1;
- Privileges for same role in rsp1 are equivalent to or include privileges for that role in rsp2;
- Services associate with each privilege in rsp1 are equivalent to or include respective services in rsp2;
- Credentials required for each role in rsp2 are equivalent to or include respective requirements for same role in rsp1; specifications for each credential requirement in rsp1 are subsets of rsp2’s.

For $\exists r_{sp1,i}, i = 1, \ldots, n$, $\exists k, k = 1, \ldots, n$:
$$r_{sp1,i} \neq r_{sp2,k};$$

$\overline{cor_{rr}} = r_{sp1,i}/r_{sp2,k}$; for $\forall \overline{cor_{rr}} = 1$,
$$PS_{i}.pr_{i} \supseteq PS_{j}.pr_{j} \oplus PS_{i}.pr_{i} \subseteq PS_{j}.pr_{j};$$

$\overline{cor_{P}} = PS_{i}.pr_{i}/PS_{j}.pr_{j}$; for $\forall \overline{cor_{P}} = 1$,
$$PS_{i}.Si \supseteq PS_{j}.Sj \oplus PS_{i}.Si \subseteq PS_{j}.Sj;$$

$\overline{cor_{C}} = CR_{i}.crn_{i}/CR_{j}.crn_{j}$; for $\forall \overline{cor_{C}} = 1$,
$$CR_{i}.CV_{i} \subseteq CR_{j}.CV_{j}.$$

Definition 5.2.3 Intersection (rsp). We say rsp1 and rsp2 are Intersection if:

- rsp1 and rsp2 share some common Roles;
- Privileges and associated services of these common roles from rsp1 are subsets of rsp2’s or vice versa, but not both.
- Credentials and associated specifications of rsp1 are subsets of rsp2’s.

For $\exists r_{sp1,i}, i = 1, \ldots, n$, $\exists k, k = 1, \ldots, n$:
$$r_{sp1,i} = r_{sp2,k};$$

For $\forall r_{sp2,j}, j = 1, \ldots, n$:
$$r_{sp2,j} \neq r_{sp1,k};$$

$\overline{cor_{rr}} = r_{sp1,i}/r_{sp2,k}$; for $\forall \overline{cor_{rr}} = 1$,
$$PS_{i}.pr_{i} \supseteq PS_{j}.pr_{j} \oplus PS_{i}.pr_{i} \subseteq PS_{j}.pr_{j};$$

$\overline{cor_{P}} = PS_{i}.pr_{i}/PS_{j}.pr_{j}$; for $\forall \overline{cor_{P}} = 1$,
$$PS_{i}.Si \supseteq PS_{j}.Sj \oplus PS_{i}.Si \subseteq PS_{j}.Sj;$$

$\overline{cor_{C}} = CR_{i}.crn_{i}/CR_{j}.crn_{j}$; for $\forall \overline{cor_{C}} = 1$,
$$CR_{i}.CV_{i} \subseteq CR_{j}.CV_{j}.$$

Definition 5.2.4 Disjoint (rsp). We say rsp1 and rsp2 are Disjoint if there are no common Roles in rsp1 and rsp2:

Definition 5.3 Service Security Policy (SSP)

SSP is a bag of tuples which contains multiset of service related security requirements. It specifies conditions (provisions/obligations) attached to certain access request for each service.

Service security policy $ssp = (s, CD)$ where:

- $s$ is service identifier;
- $CD$ is a subbag which includes condition cd (e.g. Agree to protect patient privacy) and related privileges sets PR. Each condition could require for several privileges (e.g. anyone who request browse, forwarding need agree to predefined condition).

Example:
$$ssp = (\text{Patient records}, \{(\text{protect privacy policy}, \{\text{forward, browse}\})\})$$

This is an example of service security policy for Patient Records; anyone wants to “browse” or “forward” patient records must agree to protect patient policy. We can define relationships between two Service Security Policies as: equivalent, inclusive,
intersection and disjoint by same approach we used for Role Security Policy (See section 5.2).

6. Security Controls for Types of Collaboration

We identified four types of collaboration patterns in this paper. Requirements for compatibility and integration could be different for different patterns. Integration we discuss here is a process of combining and consolidating policies. The purpose of integration is to generate a consistent view of security policies for collaborating parties, which will be carried out after compatibility checking. In the following section, we will identify integration rules or security controls for each collaboration type based on policy relationships we identified in previous section.

Type A. Basic Access

Basic Access does not need policy integration. For Basic Access, security policies the requester needs to meet are: OSP of service owner & required credential for specific role & Agree with provisions and obligations for required services (if there are any).

Type B. Service Propagation

As we discussed in Section 4, service propagation requires policies of participants equivalent or compatible with security policies of the service owner. According to relationships we defined in previous section, organization B compatible to A while A is service owner when:

- OSPB equivalent or include OSP_A &
- RSP_A equivalent or include RSP_B &
- SSP_A equivalent or include SSP_B

Based on above basic requirements for this type of collaboration, Security controls for service propagation focus on requirements of service owner. Let Policyfinal be integrated security policy, security policies for a propagated service should be:

\[
\text{OSP}_{\text{final}} = \text{OSP}_A \cup \text{OSP}_B \\
\text{RSP}_{\text{role}} = \text{RSP}_A \cap \text{RSP}_B \\
\text{RSP}_{\text{privilege}} = \text{RSP}_A \cap \text{RSP}_B \\
\text{RSP}_{\text{credential}} = \text{RSP}_A \cup \text{RSP}_B \\
\text{SSP}_{\text{condition}} = \text{SSP}_A \cup \text{SSP}_B
\]

Type C. Service Composition

We say organizations are Integratable as long as there are some common properties:

- OSP_A equivalent or include or intersect OSP_B
- RSP_A equivalent or include or intersect RSP_B
- SSP_A equivalent or include or intersect SSP_B

Based on above discussion, integrated security policy for composite service should be:

- OSP of both organizations since requester actually interact with two business units;
- Roles formed by common roles in two units;
- Privileges formed by common privileges assigned to same role;
- Credentials required formed by the union of both credential requirements for same role, requirements become stricter;
- Conditions for same service in both units should be satisfied to protect interest of both units.

\[
\text{OSP}_{\text{final}} = \text{OSP}_A \cup \text{OSP}_B \\
\text{RSP}_{\text{role}} = \text{RSP}_A \cap \text{RSP}_B \\
\text{RSP}_{\text{privilege}} = \text{RSP}_A \cap \text{RSP}_B \\
\text{RSP}_{\text{credential}} = \text{RSP}_A \cup \text{RSP}_B \\
\text{SSP}_{\text{condition}} = \text{SSP}_A \cup \text{SSP}_B
\]

7. PD-AC Framework

The core function of our PD-AC (Policy driven authorization control) framework is realized by a Policy Controller, which is associated with every
organization that provides services in business collaboration, see Figure 3. Policy Controller is used to analyze the nature of collaboration, check compatibility and determine the integrated security policies. Once receiving a request for a service, the Policy Controller firstly identifies the type of requested service, which is a process that identify whether the requested service is a collaborative service. If it is a simple service that does not involve any collaboration, the controller will perform normal comparison: looking up security policy database, check which role in the database has the requested privilege; compare requester’s credential with security requirement; make access decision. If it is a collaborated service, the controller will perform the following tasks:

- Identify collaboration type;
- Check policy compatibility or integratability;
- Generate integrated security policies that the requester need to meet according to policy integration rules;
- Compare requester credential with integrated security policies requirements;
- Make access decision.

8. Related Work

A few works has been done in security issues. [1] Proposed a model-driven trust negotiation framework for web services – Trust-Serv, author modeled trust negotiation policies by using state machines and each state associate with different permissions. We adopted this idea in our study, but we defined provisions and obligations in different way. The concept of provisions and obligations came from [5], but we associate them to services instead of transitions between roles since we think provisions and obligations are actually associated with objects for certain operation in real world.

[2] suggested a brokered architecture to build composite Web services according to the specified security constraints, which is relate to our study. They used security matchmaker to find right collaboration partners who have compatible security policies. It addressed security concerns of participator of composite services. But it did not address integration of security policies for cross-organization collaborated services. Nothing mentioned about how security policies from different partners could be combined and how to solve the conflicts between policies.

None of these studies go depth into different patterns of cross-organization collaboration. We believe different collaboration patterns can raise different integration issues. Our goal is thus to provide a first step to identify different cross-organization collaboration patterns, analysis compatibility of security policies and define policy integration rules, propose a framework that could generate integrated security requirements the requester has to fulfill based on different collaboration patterns. Our study focus on relationships between dynamically collaborated organizations other than relationships between individual requester and single service provider.

9. Conclusion

In this paper, we focus on security policies issues in cross-organization collaboration in the context of web services. We identified different collaborations types, analyzed relationships between policies, checked compatibility for collaboration and defined rules for generating integrated security policies. We have proposed a framework for handling authorization control for business collaboration. The work presented here is only at its initial stage. Future work will include policy consistency checking, policy negotiation, and security policy engagement.

10. References


