

Examining Security Certification and Access Control Conflicts Using Deontic Logic

M. Smith, M. Kelkar, R. Gamble

University of Tulsa
Department of Mathematical and Computer Sciences
gamble@utulsa.edu

Abstract. Component-based software has become a mainstream practice as organizations attempt to streamline application development tasks. These applications invariably contain third-party Commercial-off-the-Shelf (COTS) systems with black box functionality. When integrated applications require security certification, COTS components, even if individually certified, may introduce vulnerabilities into the system if their security mechanisms are poorly combined. One cause of improper integration can be found in the access control mappings across COTS component domains. Missing, conflicting, and ambiguous mappings can lead to non-compliance with security certification criteria. In this paper, we discuss certification criteria applicable to COTS integration and their interpretations to access control across domains. Highlighting common conflicts using deontic logic, we indicate how resolution strategies to those conflicts can comply with certification criteria.

Keywords: COTS-based systems, security certification, access control

1 Introduction

Commercial-off-the-shelf (COTS) components are independent pieces of software that can be combined into larger systems, the main goal being that they are reusable. Inserting COTS software into larger integrated applications is a common, inexpensive development approach. Conflicting security mechanisms can cause problems that sometimes affect the potential for complete security certification of the application by a certification and accreditation authority.

Security certification assesses a software system to determine the extent to which the system design and implementation complies with pre-specified security requirements [1, 2]. This process increases the level of confidence in the system's security by ensuring correct implementation, intended operation, and expected output. Certification discloses security vulnerabilities with their associated risk.

Currently, certification is a manual process with no widely-accepted automated solution. Security certification for integrated systems composed of COTS components is

necessary to ensure that the procured components which match the desired functionality is already certified in isolation as an atomic entity [3]. Apart from this, the certification should include the security property compatibility checking between interacting components and effect of external entities (global properties) [4]. Failure to attain this may result in incomplete certification and lead to unidentified vulnerabilities.

Predominant software certification sources are government and military documents, such as the Common Criteria [5], DITSCAP [2] and NIST Information System's certification document [1], which outline the need to assess critical security features for correct and complete policy implementation of an integrated system. The documents are primarily directives for the integrator and certifier to review the interfaces between components and determine their security compliance. Thus, the criteria are used as a basis for compliance checking, ascertaining contributing properties, and identifying conflicts that cause non-compliance.

Given that each COTS software component has its own domain with a consistent and correct security policy within that domain, we consider the problems arising from an interdomain policy that is expressed through role assignment inheritance or global policy statements for all roles, i.e., a single root representing that is inherited by all roles in all components. Applicable certification criteria for integrated COTS-based systems can be interpreted to access control policies as follows.

- A. The interdomain access control policy should be assessed to ensure correct and complete implementation (Verification phase, task 2.2 - Level 2) [2].
- B. Each policy requirement must be defined unambiguously by a complete set of attributes (e.g., initiator of an action, source of the action, the action, the object of the action, constraints) [6].
- C. If policies conflict among component domains, a deny-override strategy should be used to resolve the conflict [7]. Simply stated, if no other conflict resolution strategy can be formulated, access is denied.
- D. Policies should be traced to the originating component to enforce non-repudiation and accountability for any conflicts that may arise [2, 5, 6].

Access control is a major cross-component security related issue for integrated COTS software systems. Conflict detection and conflict resolution are often dealt with separately using different processes, languages, mechanisms, etc. [8-10]. Most approaches define security policies using logic-based languages [9, 11-13], variations on RBAC systems [8, 14, 15], or other specifications such as XACML [16]. Conflicts among access control policies have been intensively studied [10]. One variation of RBAC, the XGTRBAC, specifically focuses on interdomain conflicts [17]. The administrative model shows conflicts primarily related to role mappings across domains as well as temporal constraints on the assignment of roles. Though the conflict resolution technique is automated, conflict detection is a manual process and is done by visually scanning directed sub-graphs.

Certification of integrated systems provides a different perspective on access-control conflict detection and resolution than that used in prior research:

- Integrated systems have an overriding policy with which COTS components must comply.
- Pair-wise comparisons are not sufficient.
- Any conflict resolution must be globally consistent.
- Traceability of role inheritance and localization of conflict resolution reduces the re-certification effort due to evolution.
- Missing interdomain mapping formation can cause anomalies in propagated policies that can lead to vulnerabilities.

In this paper, we use standard deontic logic for property expressions that indicate valid access permissions. We investigate interdomain access control policy statements for security vulnerabilities related to the certification priorities of consistency and correctness. We use deontic logic to highlight common conflicts, their discovery, and resolution to comply with certification criteria.

2 Role-Based Access Control

Role-Based Access Control (RBAC) is commonly used to define access parameters within COTS components and combinations thereof. This is done by creating rule-sets of permissions, assigning the rules to roles, and then assigning roles to users [18]. The robust, low-maintenance, and efficient nature of RBAC systems allow for simple modeling of many constraints including hierarchical and separation of duty (SoD) [19]. A role hierarchy is a partial order relationship established among roles. SoD constraints define mutual exclusion relations between two entities. Users assigned to a role have access rights defined by the permissions assigned to the same role.

In RBAC, access is defined in terms of user-role mappings that specify a set of security constraints restriction on role access for a set of users. When these mappings are between distinct software components or domains, they are called interdomain mappings. Approaches that examine interdomain conflicts at the design phase use XML or UML to detect interdomain conflicts [10]. For example, recurrence of a conflict template in a UML system representation denotes conflict. Thus, the UML architecture diagram is manually scanned. Similar research is found in the XGTRBAC Admin model, which shows examples of cross-domain problems [17]. A conflict is detected by visually scanning directed sub-graphs. The conflict resolution technique is automated using formal language implementation [17].

Security policies of an integrated COTS system can be defined at three levels:

1. Local policy of a component that specifies the constraints between its roles.
2. For each role which is included in the local policy of a component.
3. Globally for the system, defining interdomain mappings

Conflicts arise when any of the above policies have a contradiction.

RBAC systems have noted limitations that should be addressed in integrated system certification analysis [15, 20]. Inheritance may be ambiguous where it does not correspond to an organization's hierarchy. It is not necessary for all roles to allow user assignments as some roles in the hierarchy are defined solely for convenience and are not intended to have users assigned to them. We call these non-user roles. Context is not included in role assignment constraints, which can be eased if traceable origins of inherited access are maintained across domains of the integrated system.

3 Deontic Logic

Deontic logic, as proposed by vonWright [21], extends modal logic with concepts that define operations on the obligatory, the permitted, and the forbidden. Several variations of deontic logic exist, but we use Standard Deontic Logic (SDL) that historically has been used for reasoning about morality, law, and social norms [22].

An *action* is the antecedent for the deontic operators such that "it is permitted that *action*." If an action is performed (or not performed), nothing can be stated as to its obligatory, permitted, or forbidden character. In other words, actual performance is unrelated to any given permission.

Permission is defined as being allowed to complete some action. Logically, it is represented as PA, which is read as, "Permission to achieve A," or "It is permitted that A." Disjunctive and conjunctive rules of logic apply to permission, such that (1) and (2) necessarily hold. However, it is not necessarily true that the contrapositive of (2) holds.

$$PA \vee PB \equiv P(A \vee B) \quad (1)$$

$$P(A \wedge B) \rightarrow PA \wedge PB \quad (2)$$

Prohibition is the negation of Permission, designated logically as $\neg PA$ ("not permission to achieve A," or "forbidden to achieve A"). For example, we are not allowed to steal hence we must not steal.

Obligation requires a Subject to do an Action and is stated as "it is obligatory that A," which is equivalent to $\neg(P\neg A)$. Obligation implies permission, as in equation (3). For example, in the non-smoking compartment, you are obligated to not smoke, therefore non-smoking is permitted, and smoking is forbidden.

$$OA \rightarrow PA \quad (3)$$

Indifference implies permission and is denoted in equation (4). This occurs when an action and its negation are both permitted. Permission does not imply indifference while it is equivalent to $\neg O\neg A$, or the statement, "You are not obligated to not achieve A." For example, in a smoking compartment, we may smoke, but we may also not smoke. If Action A is indifferent, then $\neg A$ is also permitted, but if A is obligatory (and thus permitted), then $\neg A$ is forbidden, as seen in equation (5).

$$PA \wedge P\neg A \rightarrow PA \quad (4)$$

$$OA \wedge PA \rightarrow \neg(P\neg A) \wedge PA \quad (5)$$

Dispensation is a non-obligation; a waiver from a required action [23]. This is denoted as equation (6). Notice that this is not equivalent to, nor does it imply, $\neg PA$. For this reason, prohibition does not imply non-obligation.

$$\neg OA \equiv P\neg A \quad (6)$$

VonWright defines six laws of commitment (equations (7) through (11)) and one rule that says you can never be obligated to do the forbidden (equation (12)). We extend these to include the converse of (9) to state that if you are not obligated to do A and the obligation of A implies the obligation of B, you are therefore not obligated to do B (equation (14)). These prove important when dealing with the inheritance of different kinds of constraints, as can be seen in the Chinese Wall example. Finally, the *Principle of Permission* is stated as, “you cannot have both permission and not permission at the same time” (equation (15)).

$$PA \wedge O(A \rightarrow B) \rightarrow PB \quad (7)$$

$$\neg PA \wedge O(A \rightarrow B) \rightarrow \neg PB \quad (8)$$

$$OA \wedge O(A \rightarrow B) \rightarrow OB. \quad (9)$$

$$O(A \rightarrow (B \vee C)) \wedge \neg PB \wedge \neg PC \rightarrow \neg PA \quad (10)$$

$$OA \wedge O((A \wedge B) \rightarrow C) \rightarrow O(B \rightarrow C) \quad (11)$$

$$O(\neg A \rightarrow A) \rightarrow OA \quad (12)$$

$$\neg O(A \vee B) \wedge \neg PA \wedge \neg PB \quad (13)$$

$$\neg OA \wedge O(A \rightarrow B) \rightarrow \neg OB \quad (14)$$

$$PA \vee \neg PA. \quad (15)$$

3.2 Access Control Usage

Cuppens, et al. [24] represented access control policies using deontic logic. He denotes a difference between having the right to know something and actually knowing it, which brings in the concept of having a memory. This affects all separation of duty constraints

and requires including a temporal concept to the model. Organizational-based Access Control is another approach that specifies a logical model for policy management. Here, a permission is denoted as $Permission(org, r, v, a, c)$ that is read as “in organization org , within context c , role r is permitted to perform activity a on view v ” [15]. This approach is tailored to alleviate the concerns regarding organizational policy hierarchies and the context limitations of the RBAC model.

In another approach, Free Variable Tableaux is also used to analyze policy conflicts [9]. Deontic logic is used in the form of $Obli+$, $Obli-$, $Auth+$, and $Auth-$. They also introduce Chinese Wall and SoD constraints to express mutual exclusion of different policies. Chinese Wall defines two mutually exclusive target roles or objects.

4 Interdomain Access Control Certification Issues

For our use, an RBAC access control policy from a deontic logic perspective is divided into a set of permissions and a set of constraints for each role. The policies defined in a permission set should be complete and unambiguous [2]. Constraints are applied to the permission set that should not contradict each other. In this paper, we address constraints in the form of role SoDs, User based SoDs, and ChineseWall.

4.1 Applying Certification Criteria to Change Deontic Logic Specification

Looking at the conflict definitions and model, we adopt the triplet representation from [25]: (u, o, a) , where the user, u , completes an action, a , on the object, o . The Principle of Permission must be altered to remove the possibility of ambiguity—if you only have permission to do A , how do you know if you have permission to not do A ? Or in the absence of permission to not do A , are you then obligated to do A ? Therefore, we clarify this principle as stated in equation (16). $O(u, o, a)$ is an unambiguous statement, so it follows that we must specify that $O(u, o, a) \rightarrow P(u, o, a)$. The derived policy of $P(u, o, a)$ and $O(u, o, a)$, as seen in equation (17), explains the absence of a stand-alone $O(u, o, a)$ in equation (16).

$$(P(u, o, a) \wedge O(u, o, a)) \vee (P(u, o, a) \wedge \neg P(u, o, a)) \vee (\neg P(u, o, a)) \quad (16)$$

$$O(u, o, a) \rightarrow P(u, o, a) \wedge O(u, o, a) \quad (17)$$

Policies are attached to roles and can be divided into a set of permissions and a set of constraints (see equation (18)). Permissions are passed downward through the hierarchy.

$$Policy_{r1}: \{Permissions\} \cup \{Constraints\} \quad (18)$$

Now, in the case of related actions, we follow the law of commitment in equations (7) or (9), depending on the permission applied to the action. Then we add the implied permission to the policy's set of permissions and add the implication, $O(a_1 \rightarrow a_2)$, to the set of constraints. For example, if we are required to write and writing implies reading, we must also read, meaning both actions (read and write) must be permitted or obligated.

4.1.1 Complete, Unambiguous, and Correct Policies

Certification criteria assert that the security requirement specifications should be complete (Section 1 -Cert Criteria A, B), and thus, unambiguous. A policy statement is unambiguous if and only if there is only one interpretation of the statement. For example, $P(u,o,a)$ is ambiguous because you cannot know if $P\neg(u,o,a)$ or $O(u,o,a)$ is assumed. Thus, we require the role permissions to have complete meaning as follows: equation (19) states that obligation assumes permission. Equation (20) shows indifference that gives the user the choice to achieve A. Equation (21) denotes prohibition.

$$O(u,o,a) \rightarrow P(u,o,a) \quad (19)$$

$$P(u,o,a) \wedge P\neg(u,o,a) \quad (20)$$

$$\neg P(u,o,a) \quad (21)$$

For certification purposes, the security requirement specification must be correct (Section 1 Cert Criteria A), and thus, not in conflict. For integrated system certification, the interdomain policy, as a combination of local policies, should be free of conflicting policy statements. Direct conflict of permissions is shown in (22)

$$P(u,o,a) \wedge \neg P(u,o,a) \quad (22)$$

4.1.2 Need of Deny-Overrides

In case of conflict between the combined policies of interacting components that must cooperate, the deny-overrides strategy should be used (Section 1 - Cert Criteria C). In a combined policy set, if any statement denies access, then deny-overrides allows that statement to prevail over the other contradicting statements. Thus, the access is denied in the resulting policy. Deny-overrides can also be specified as a revocation of permission. We use a function, called the Policy Revocation function (equation (23)) to apply the global policy to conflicting policy statements. This function makes false all permissions of the type that is being overridden by the global policy. Thus this function would derive the permissions as shown in (24), (25), and (26).

$$\text{PR}(o,a): W(o,a) \rightarrow \neg W(o,a) \quad (23)$$

$$\text{OA} \wedge \text{P}\neg\text{A} \xrightarrow{\text{deny override}} \text{OA} \quad (24)$$

$$\text{PA} \wedge \neg\text{PA} \xrightarrow{\text{deny override}} \text{PA} \quad (25)$$

$$\text{OA} \wedge \neg\text{PA} \xrightarrow{\text{deny override}} \text{OA} \quad (26)$$

4.1.3 Using Origin Tags

Redundant policy statements that are propagated from different roles through role inheritance are allowed if managed properly. We maintain an origin tag as denoted by a superscript on the permissions in order to trace it back to its original role. This aids in conflict resolution by providing a path back to an earlier culprit. Keeping the role-association with the users that propagates through role inheritance helps detect the conflicting redundant permissions that might have come from roles of distinct origins. The ability to trace the information flow helps in maintaining accountability and takes care of non-repudiation, as addressed by certification (Section 1 Cert Criteria D).

4.2 Exemplifying Incomplete Interdomain Mapping Policy Information

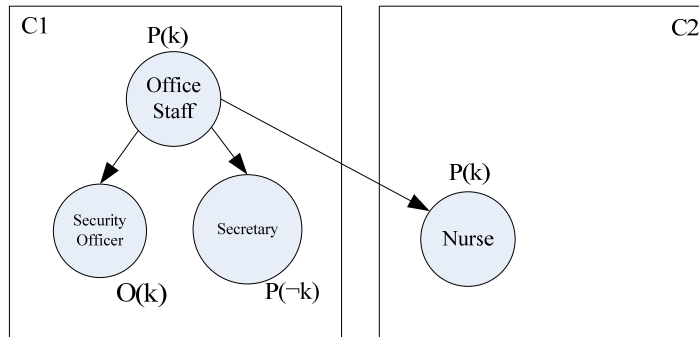


Fig. 1. Incomplete Interdomain Mapping Example

Interdomain role inheritance can propagate incomplete policy information. In Fig. 1, component C1 has three roles: office staff, security officer, and secretary (we do not use the user, object, action triplet in this example for simplicity). Office staff is a non-user role

meaning no user is directly assigned to that role and gives permission to have keys $P(k)$. This is inherited by user-role security officer who is obligated to have keys ($O(k)$) and user-role secretary who has permission not to have keys ($P\neg k$). C2 is added to the system with user-role nurse. Nurse inherits all the policies from office staff, which is $P(k)$ and is a ambiguous. It is evident that there is an interdomain inheritance mapping missing that would allow nurse to have complete and unambiguous permissions.

4.3 Direct Conflict Detection and Resolution

Direct conflicts occur when one role's policy includes contradicting statements making the resulting policy incorrect (Section 4.1.1), as seen in Fig. 2 where r_3 inherits both obligation and prohibition for the same object and action. This creates a logically contradicting statement of the form $A \wedge \neg A$.

One resolution strategy is to define a global policy combining rule, similar to XACML where each $\langle \text{Policy} \rangle$ and $\langle \text{Policy Set} \rangle$ can respectively define a rule and policy combining algorithm [16] as in Fig. 3 below. However, this must also resolve the logical policy for r_3 in order for it to be a conjunctive tautology. We use to apply the global policy to conflicting policy statements. This function makes false all permissions of the type that is being overridden by the global policy.

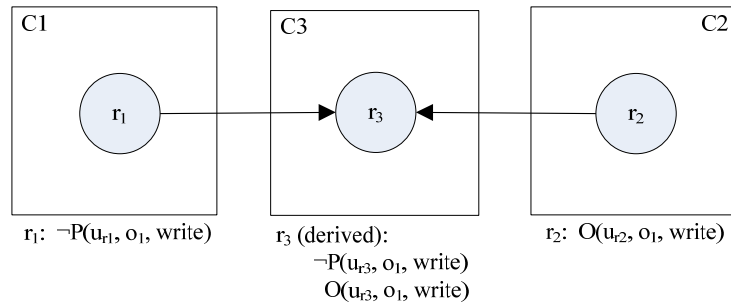


Fig. 2. Direct Conflict Example

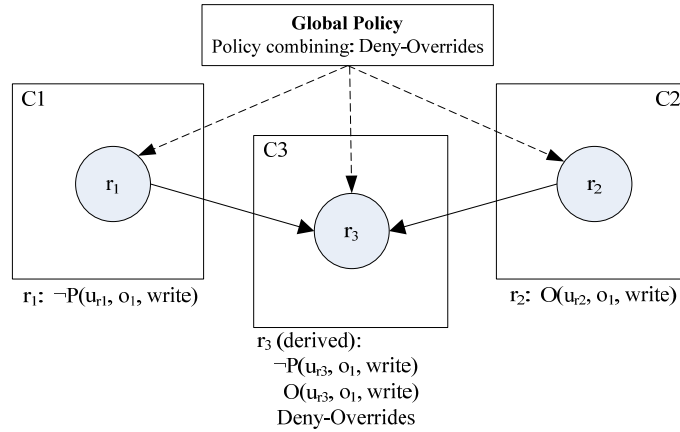


Fig. 3. Global Policy Resolution Strategy

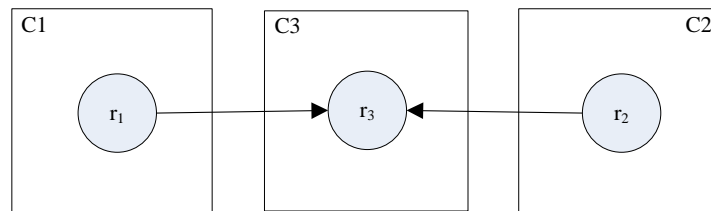


Fig. 4. Chinese Wall Example

4.4 Chinese Wall Conflict Detection and Resolution

In Fig. 4, access control policies for roles r_1 , r_2 , r_3 form the local policies of component C1, C2 and C3 respective and the global policy is specified as follows.

Global Policy:

$$\forall u, o: [O(u, o, write) \rightarrow O(u, o, read)] \quad (27)$$

Role r_1 policy:

$$O(u_{r_1}, o_2, read) \quad (28)$$

$$O(u_{r_1}, o_3, read) \quad (29)$$

$$cw(u_{r_1}, \{o_3, o_2\}, read) \rightarrow \neg(O(u_{r_1}, o_2, read) \wedge O(u_{r_1}, o_3, read)) \quad (30)$$

The label cw stands for Chinese Wall. As specified earlier, an access control policy specified for a role is divided into permission set of mutually exclusive permissions and

constraints. Given the Chinese Wall constraint, r_1 has two sets of permissions. This yields:

Role r_1 policy:

$$\text{Permission Set} = \{ \{ O(u_{r_1}, o_2, \text{read}) \}, \{ O(u_{r_1}, o_3, \text{read}) \} \} \quad (31)$$

$$\text{Constraint: } \{ \neg(O(u_{r_1}, o_2, \text{read}) \wedge O(u_{r_1}, o_3, \text{read})) \} \quad (32)$$

Role r_2 policy:

$$\text{Permission Set} = \{ O(u_{r_2}, o_3, \text{read}) \} \quad (33)$$

Role r_3 does not have any local policies but inherits the permission sets and constraints from r_1 and r_2 , denoted by their origin tags. Given role inheritance properties, we form the cross product of permission sets across distinct roles. The resulting ordered pairs are unioned for new permissions sets for r_3 . The constraint is also propagated. Thus, the resulting policy of r_3 is as follows:

$$\{ O^{r_1}(u_{r_1}, o_2, \text{read}), O^{r_2}(u_{r_2}, o_3, \text{read}) \} \quad (34)$$

$$\{ O^{r_1}(u_{r_1}, o_3, \text{read}), O^{r_2}(u_{r_2}, o_3, \text{read}) \} \quad (35)$$

$$Cw^{r_1}: \neg(O^{r_1}(u_{r_1}, o_2, \text{read}) \wedge O^{r_1}(u_{r_1}, o_3, \text{read})) \quad (36)$$

If viewed as purely inherited (no origin tags) it is clear that (34) conflicts with (36). This induces a new Chinese Wall for r_3 that explicitly focuses compliance given the permission origins.

$$Cw^{r_3}: \neg(O^{r_1}(u_{r_1}, o_2, \text{read}) \wedge O^{r_2}(u_{r_2}, o_3, \text{read})) \quad (37)$$

(37) is added to the constraint set of role r_3 as a conflict resolution strategy. This causes (34) to be partitioned into

$$O^{r_1}(u_{r_1}, o_2, \text{read}) \quad (38)$$

$$O^{r_2}(u_{r_2}, o_3, \text{read}) \quad (39)$$

However, it is clear that (35) is not in conflict with and subsumes (39). Thus, the resulting permissions and constraints for r_3 are:

$$O^{r_1}(u_{r_1}, o_2, \text{read}) \quad (40)$$

$$O^{r_1}(u_{r_1}, o_3, \text{read}) \wedge O^{r_2}(u_{r_2}, o_3, \text{read}) \quad (41)$$

$$Cw^{r_1}: \neg(O^{r_1}(u_{r_1}, o_2, \text{read}) \wedge O^{r_1}(u_{r_1}, o_3, \text{read})) \quad (42)$$

$$Cw^{r3}: \neg(O^{r1}(u_{r1}, o_2, \text{read}) \wedge O^{r2}(u_{r2}, o_3, \text{read})) \quad (43)$$

From (27), we have:

$$O^{r2}(u_{r2}, o_3, \text{write}) \rightarrow O^{r2}(u_{r2}, o_3, \text{read}) \quad (44)$$

Hence, if $O(u_{r2}, o_3, \text{read})$ is allowed then $O(u_{r2}, o_3, \text{write})$ should also be allowed. Thus, $r3$'s inherited policy is:

$$\text{Permission Set} = \{ \{ O^{r1}(u_{r1}, o_2, \text{read}) \}, \{ O^{r1}(u_{r1}, o_3, \text{read}), O^{r2}(u_{r2}, o_3, \text{write}) \} \} \quad (45)$$

$$\text{Constraint Set} = \{ cw^{r1}(u_{r1}, \{ o_3, o_2 \}, \text{read}), cw^{r3}(u_{r3}, \{ o_3, o_2 \}, \text{read}) \} \quad (46)$$

4.5 User Based SoD Conflict Detection and Resolution

In Fig. 5, there is an equivalence relation between roles r_1 and r_2 . Equivalence is bidirectional inheritance, in which both roles inherit permissions and constraints from each other and thus become equal. Access control policy for roles r_1 and r_2 which form the local policies of component C1 and C2, respectively, are specified below. Let m, n, p, q be integers to denote different users. For brevity we define the user based SoD between users as:

$$\neg(u^1 \wedge u^2) \quad (47)$$

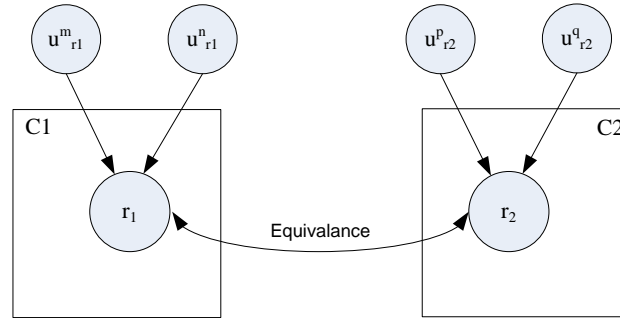


Fig. 5. Original user based SoD example

The user-based SoDs in roles r_1 and r_2 are found in (48) and (49), respectively.

$$\text{usod}^{r1}(\{u_{r1}^m, u_{r1}^n\}, o, a) : \forall o, a, m, n, u_{r1}^m, u_{r1}^n, m \neq n : \neg(O(u_{r1}^m, o, a) \wedge O(u_{r1}^n, o, a)) \quad (48)$$

$$\text{usod}^{r2}(\{u_{r2}^p, u_{r2}^q\}, o, a) : \forall o, a, p, q, u_{r2}^p, u_{r2}^q, p \neq q : \neg(O(u_{r2}^p, o, a) \wedge O(u_{r2}^q, o, a)) \quad (49)$$

This forces the following permissions sets on roles r_1 (50) and r_2 (51), respectively.

$$\forall o, a, m, n, u_{r1}^m, u_{r1}^n, m \neq n, \text{Permission Set} = \{ \{ O(u_{r1}^m, o, a) \}, \{ O(u_{r1}^n, o, a) \} \} \quad (50)$$

$$\forall o, a, p, q, u_{r2}^p, u_{r2}^q, p \neq q, \text{Permission Set} = \{ \{ O(u_{r2}^p, o, a) \}, \{ O(u_{r2}^q, o, a) \} \} \quad (51)$$

The cross product of permission sets the following six policy statements:

$$\forall o, a, m, p, u_{r1}^m, u_{r2}^p, m \neq p, \{ \{ O(u_{r1}^m, o, a) \}, \{ O(u_{r2}^p, o, a) \} \} \quad (52)$$

$$\forall o, a, m, q, u_{r1}^m, u_{r2}^q, m \neq q, \{ O(u_{r1}^m, o, a) \}, \{ O(u_{r2}^q, o, a) \} \quad (53)$$

$$\forall o, a, n, p, u_{r1}^n, u_{r2}^p, n \neq p, \{ O(u_{r1}^n, o, a) \}, \{ O(u_{r2}^p, o, a) \} \quad (54)$$

$$\forall o, a, n, q, u_{r1}^n, u_{r2}^q, n \neq q, \{ O(u_{r1}^n, o, a) \}, \{ O(u_{r2}^q, o, a) \} \quad (55)$$

$$\text{usod}^{r1}: \forall o, a, m, n, u_{r1}^m, u_{r1}^n, m \neq n, \neg(O(u_{r1}^m, o, a) \wedge O(u_{r1}^n, o, a)). \quad (56)$$

$$\text{usod}^{r2}: \forall o, a, p, q, u_{r2}^p, u_{r2}^q, p \neq q, \neg(O(u_{r2}^p, o, a) \wedge O(u_{r2}^q, o, a)). \quad (57)$$

If viewed as purely inherited (no origin tags), it is clear that (52), (53), (54) and (55) conflicts with (56) and (57). This induces a new global usod for both r_1 and r_2 that explicitly focuses compliance given the original permissions.

$$\text{usod}^{r3}: \forall o, a, m, p, u_{r1}^m, u_{r2}^p, m \neq p, \neg(O(u_{r1}^m, o, a) \wedge O(u_{r2}^p, o, a)) \quad (58)$$

The above (58) usod is added to the constraint set of both the roles r_1 and r_2 as a conflict resolution strategy. This causes (52), (53), (54) and (55) to be partitioned into mutually exclusive obligations. Thus, the permissions and constraints for r_1 and r_2 are:

$$\forall o, a, m, u_{r1}^m : O(u_{r1}^m, o, a) \quad (59)$$

$$\forall o, a, p, u_{r2}^p : O(u_{r2}^p, o, a) \quad (60)$$

$$\forall o, a, m, u_{r1}^n : O(u_{r1}^n, o, a) \quad (61)$$

$$\forall o, a, p, u_{r2}^q : O(u_{r2}^q, o, a) \quad (62)$$

$$\forall o, a, m, n, u_{r1}^m, u_{r1}^n, m \neq n, \text{usod}^{r1}: \neg(O(u_{r1}^m, o, a) \wedge O(u_{r1}^n, o, a)) . \quad (63)$$

$$\forall o, a, p, q, u_{r2}^p, u_{r2}^q, p \neq q, \text{usod}^{r2}: \neg(O(u_{r2}^p, o, a) \wedge O(u_{r2}^q, o, a)) . \quad (64)$$

$$\forall o, a, m, p, u_{r1}^m, u_{r2}^p, m \neq p, \text{usod}^{r3}: \neg(O(u_{r1}^m, o, a) \wedge O(u_{r2}^p, o, a)) . \quad (65)$$

Out of which (59), (60), (61) and (62) are mutually exclusive elements of the permission set and (63), (64) and (65) are constraints.

5 Conclusion

In this paper, we have interpreted security certification criteria for access control issues across multiple COTS components in an integrated system. We have take commonly found conflicts, applied them to interdomain mappings, evaluating the resulting policies, and analyzing them using deontic logic operations. Deontic logic allows the permissions to be represented unambiguously as well as indicates when permissions are in conflict. We exemplified SoD and Chinese Wall constraints and their inheritance issues with respect to certification criteria.

Acknowledgements. This work is supported in part by a US AFOSR award FA9550-05-1-0374.

References

1. Ross, R., Guide for the Security Certification and Accreditation of Federal Information Systems, in NIST Special Publication 800-37. 2004.
2. Department of Defense Information Technology Security Certification and Accreditation Process. 2000.
3. Brenner, E. and I. Derado. Specifying a Certification Process for COTS Software Components Using UML. in Fourth International Symposium on Object-Oriented Real-Time Distributed Computing. 2001. Magdeburg, Germany.
4. Khan, K.M. and J. Han. Deriving Systems Level Security Properties of Component Based Composite Systems. in Australian Software Engineering Conference (ASWEC'05). 2005.
5. Common Criteria for Information Technology Security Evaluation. 2005.
6. Wallace, D.R., L.M. Ippolito, and B. Cuthill, Reference Information for the Software Verification and Validation Process, in NIST Special Publication 500-234. 1996.

*First International Workshop on Software Certification 2006 (CERTSOFT 06),
a satellite event of Formal Methods 2006 (FM06), August 26-27, 2006.*

7. Allen, J.H., CERT Guide to Systems and Network Security Practices. 2 ed. 2001: Addison Wesley.
8. Benferhat, S. and R.E. Baida. Conflicts Resolutions of Prioritized Security Policies. in 9th ACM Symposium on Access Control Models and Technology (SACMAT '04). 2004. Yorktown Heights, New York, USA.
9. Kamoda, H., M. Yamaoka, and S. Matsuda. Policy Conflict Analysis Using Free Variable Tableaux for Access Control in Web Services Environments. in 14th International World Wide Web Conference. 2005. Chiba, Japan.
10. Ray, I., et al. Using uml to visualize role-based access control constraints. in ninth ACM symposium on Access control models and technology. 2004. York town, New York, USA.
11. Benferhat, S., E. Baida, and F. Cuppens. A stratification-based approach for handling conflicts in access control. in eighth ACM symposium on Access control models and technologies. 2003. Como, Italy.
12. Bieber, P. and F. Cuppens. A Definition of Secure Dependencies using the Logic of Security. in Computer Security Foundations Workshop IV, 1991. Proceedings. 1991. Franconia, NH, USA.
13. Cuppens, F. and R. Demolombe. A Modal Logical Framework for Security Policies. in 10th International Symposium on Foundations of Intelligent Systems. 1997. Charlotte, North Carolina, Usa.
14. Al-Kahtani, M.A. and R. Sandhu. Induced Role Hierarchies with Attribute-Based RBAC. in 8th ACM Symposium on Access Control Models and Technologies (SACMAT). 2003. Como, Italy.
15. El Kalam, A.A., et al. Organization based access control. in Proceedings of the 4th International Workshop on Policies for Distributed Systems and Networks (POLICY '03). 2003. Toulouse, France.
16. Anderson, A. Core and hierarchical role based access control (RBAC) profile of XACML v2.0. in Oasis. 2005.
17. Bhatti, R., et al., X-GTRBAC Admin: A Decentralized Administration Model for Enterprise-Wide Access Control. ACM Trans. On Information and System Security, 2005. **8**(4): p. 388-423.
18. Jensen, K., Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use. Vol. 1. 1992: Springer-Verlag.
19. Shafiq, B., et al., Secure Interoperation in a Multidomain Environment Employing RBAC Policies. IEEE Transactions on Knowledge and Data Engineering, 2005. **17**(11): p. 1557-1577.
20. Bhatti, R., et al., X-GTRBAC Admin: A Decentralized Administration Model for Enterprise-Wide Access Control. ACM Transactions on Information and System Security, 2005. **8**(4): p. 388-423.
21. von Wright, G.H., Deontic Logic. Mind, New Series, 1951. **60**(237): p. 1-15.
22. McNamara, P., Deontic Logic. Spring Edition ed. The Stanford Encyclopedia of Philosophy, ed. E.N. Zalta. 2006.
23. Kagal, L. and T. Finin, Modeling Conversation Policies Using Permissions and Obligations. Journal of Autonomous Agents and Multi-Agent Systems, 2006.
24. Cuppens, F., et al., Merging Regulations: Analysis of a Aractical Example. 16th International Journal of Intelligent Systems, 2001.
25. Kamoda, H., et al. Policy Conflict Analysis Using Tableaux for On Demand VPN Framework. in Proceedings of the Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks. 2005.