Scientia Iranica D (2017) 24(6), 3101–3116

Sharif University of Technology
Scientia Iranica
Transactions D: Computer Science & Engineering and Electrical Engineering
www.scientiairanica.com

A semantic access control model for online social networks

M. Alizadeh[a], M. Amini[b,*], S.A. Javadi[a] and R. Jalili[b]

a. *Data & Network Security Lab (DNSL), Department of Computer Engineering, Sharif University of Technology, Tehran, P.O. Box 11155-9517, Iran.*
b. *Department of Computer Engineering, Sharif University of Technology, Tehran, P.O. Box 11155-9517, Iran.*

Received 10 May 2015; received in revised form 9 August 2016; accepted 27 December 2016

KEYWORDS
Security models and policies;
Access control;
Online social networks;
Description logic.

Abstract. Online Social Networks (OSNs) are very popular and users share various information in these networks. To protect these resources from unauthorized access, these frameworks must support flexible access control mechanisms. Semantic technology provides new opportunities for this purpose. This paper proposes a Prioritized Ontology-Based Access Control (POBAC) model for protecting users' information in OSNs. In POBAC, Description Logic (DL) is used for modeling of security-related information in social networks as an ontology and MKNF$^+$ rules are used for specification of system's and users' access control policies. Using MKNF$^+$, we can utilize non-monotonic inference (i.e., closed-world reasoning) in the access control procedure. Furthermore, users are able to define their access control rules, exceptions, and default policies. The potential conflict among different access control rules defined by users and the system is another problem, which is resolved in POBAC by considering priority levels for rules in a logical manner. Logical foundation of the model dedicates accuracy, expressiveness, and inference (of implicit access rules from the explicit ones) to the model and thus decreases the risk of sharing information in OSNs. The expressive power of the model is demonstrated through a case study.

1. Introduction

Online Social Networks (OSNs) are currently considered as one of the important aspects of the Web. Users in OSNs make profiles and share different types of resources such as personal information, photos, notes, and videos with others. Moreover, users can establish relationships and communicate with each other. Making these contents available to others raises some privacy and security concerns for users and imposes new security requirements for access control systems employed in these environments. Some of these new security requirements are as follows:

- In OSNs, there is no central authority in the system and users themselves determine which groups of users could be authorized to access their resources. Defining a list of authorized users for each resource is a cumbersome job for users. Therefore, an expressive policy specification language is needed to enable users to easily define their access control policy;
- OSNs are dynamic environments. Users change their relationships with others dynamically. If users like to make a list of authorized users accessing their

*. *Corresponding author. Tel.: +98 21 66166656*
E-mail addresses: malizadeh@ce.sharif.edu (M. Alizadeh); amini@sharif.edu (M. Amini); ajavadi@ce.sharif.edu (S.A. Javadi); jalili@sharif.edu (R. Jalili)

doi: 10.24200/sci.2017.4576

resources, they should update such lists continually. Thus, the access control mechanism must be dynamic.

The traditional access control models (such as discretionary, mandatory, or role-based access control) as well as the models specifically introduced for OSNs until now are not rich enough to cover all security requirements of these environments. The main problems and weaknesses of these models are as follows:

- The policy rules are coarse-grained in the existing models; however, we need to specify more fine-grained policy rules considering different types of relationships which are defined in social networks between users and their shared resources;
- In the existing models for OSNs, all people with the same relationships with a user have the same access; however, in practice, a user may differentiate between his/her colleagues in giving access to his/her personal photos;
- Most of the existing models for OSNs are not flexible and expressive enough in defining access rules by users to their resources and provide just a limited set of static rules;
- In OSNs, there are various types of resources with different properties and there are various types of relationships between users. A proper and error-prone access control model for OSNs should model and consider all types of the entities existing in these environments and their properties and relationships in access policy specification, conflict resolution, and policy inference.

The semantic technology can play an important role in a more expressive access control model for OSNs. Semantic technology encodes meanings in an abstract layer separate from data and content files, and separate from application code by describing the structure of the knowledge we have about them. With semantic technology, adding, changing, and implementing new relationships between the entities or interconnecting programs in a different way can be as simple as changing the external meta-data that these programs share. A run-time semantic data model, which is more often leveraged to represent and share knowledge in a distributed world using semantic technology, is *ontology*, which is specified by a logical language. In fact, using the ontology of entities in OSNs specifying their semantic relationships and employing an expressive logic-based language for policy specification and inference make the definition and management of access control policies much easier and reliable than traditional access control models. Description Logic (DL) is an appropriate language for knowledge representation. Wide use of DL, especially in ontology definition languages such as OWL (Ontology Web Language), makes it a suitable choice as a policy specification language at the first glance. However, DL has some limitations such as lack of support of rules and non-monotonic inference. The limitation of DL can be compensated by the other main family of knowledge representation logics called logic programming. Therefore, a hybrid logic can be an appropriate solution for use as a policy specification language.

In this paper, a Prioritized Ontology-Based Access Control (POBAC) model for OSNs is proposed. In the proposed model, DL is used to describe concepts, roles, individuals, and relationships among social network entities. As description logic does not support rules, we use MKNF$^+$ [1] as a combination of Description Logic (DL) and Answer Set Programming (ASP). Using MKNF$^+$ in the proposed model has the following advantages:

- It increases the expressive power of description logic by incorporating rules;
- As arity restriction does not exist in ASP predicates, defining access control rules is easier in this logic;
- It is possible to have some non-monotonic features such as default rule specification, closed-world reasoning, and exceptions.

POBAC is aimed at providing an expressive policy specification language which enables the users and the system in an OSN to define and infer their access control policy in an easier way without worrying about the incompatibility, coherence, and coverage of access rules in presence of complicated semantic relationships in these environments. This model substantially extends our initial model proposed in [2].

The rest of this paper is organized as follows. Section 2 reviews the related access control models proposed for OSNs. In Section 3, a brief overview of MKNF$^+$ is presented. In Section 4, a simple OSN is modeled. Formal specifications of the proposed access control model are described in Section 5. Section 6 addresses the security policy specification in the proposed model. Section 7 describes the access control procedure of the proposed model and analyzes time complexity of this procedure. A case study to demonstrate the applicability of the model is given in Section 8. Finally, Section 9 concludes the paper.

2. Related work

Boyd and Ellison [3] described some privacy and security issues in OSNs. Carminati et al. [4] proposed a semi-decentralized access control model for such environments. In their proposed model, the relation type, distance, and trust among users were considered as parameters that could be used for defining access control policies. Fong et al. [5] formalized the access control mechanisms behind Facebook. Carminati et al. [6] proposed using the semantic web tools for enforcing access control policies. Authorization, administration, and filtering policies were mentioned as different types of policies which could be defined by users in OSNs. Such policies are modeled by OWL and Semantic Web Rule Language (SWRL). Masoumzadeh and Joshi [7] considered protecting the relationships among concepts in their proposed model. They mentioned that users should have the ability to control visibility of their relationships; furthermore, their proposed model supported defining multi-authority access policies. In comparison to the aforementioned models [6,7], we propose to use MKNF$^+$ instead of SWRL for defining access policy rules. SWRL is a kind of Horn-like rules. It is a combination of OWL and the unary/binary Datalog Rule Markup Language. In MKNF$^+$, predicates with arbitrary arities can be defined. Moreover, in contrast to SWRL, which is monotonic, MKNF$^+$ enables non-monotonic reasoning by supporting negation-as-failure. Using MKNF$^+$, users can define more complex policies such as exceptions and default policy. Furthermore, in the proposed model, instead of using simple conflict resolution approaches such as denials-takes-precedence, the priority-based approach is used to resolve conflicts among access control rules. Further to users' policies, the system access policies are also considered in the model.

Various models have been proposed for conflict detection and resolution among access control rules. Bertino et al. [8] proposed to define weak and strong authorizations. In their model, strong authorizations had priority over weak authorizations. Cuppens et al. [9] proposed an Organization Based Access Control (OrBAC) which supported conflict detection and resolution. They restricted the structure of rules that users could define for preserving decidability. Javadi et al. [10] augmented non-monotonic features to OrBAC using MKNF$^+$ logic. To the best of our knowledge, no prioritized ontology based access control model has been proposed for OSNs yet.

Finding an appropriate logic that is a combination of DL and rules for policy specification and inference in OSNs with the mentioned characteristics has been a challenge in our proposed access control model. Several frameworks have been proposed for combining DLs and rules. AL-log [11] combines ALC with positive Datalog programs. DL-log [12] was proposed to extend AL-log. DL-log supports disjunctive Datalog with negation as well as binary predicates. Such frameworks use the DL-safety condition as a restriction on the integration of ontology and rules. According to the DL-safety restriction, each variable in a rule should occur in a non-DL atom in the rule's body. This restriction affects the expressive power of the framework. CARIN [13] is a framework that does not respect safe interaction between DL and rules. In fact, unrestricted interaction among DL and rules would limit the expressiveness of at least one DL or rule. Donini et al. [14] used autoepistemic operators and added negation-as-failure to ALC. In this paper, we use MKNF$^+$ as a formalism which combines DL and rules. In comparison to the proposed formalisms, to the best of our knowledge, MKNF$^+$ is the most powerful decidable formalism proposed for combination of description logic with rules [1].

3. Introduction to MKNF$^+$

MKNF$^+$ integrates DL and ASP using the MKNF logic as a semantic infrastructure. A brief overview of its underlying components is worth presenting.

- Description Logic (DL): DL represents the knowledge of a world by defining the existing concepts in the world (its terminology) and then using the defined concepts to specify the properties of the objects and individuals existing in the world. A DL knowledge base O consists of two components [15]:
  1. Terminology Box (TBox): TBox describes the general structure of the world using concepts (classes) and roles (relations). In fact, TBox is similar to the schema of a database;
  2. Assertion Box (ABox): ABox describes the properties of particular objects in the world. ABox is similar to the data part of a database. It includes assertions such as $C(a)$ (e.g. *User(Alice)*) and $R(a, b)$ (e.g. *IsFriendOf (Alice, Bob)*) in which C and R are a concept and a relation, respectively.
- ASP: ASP is a non-monotonic rule based formalism suitable for compensating the shortcomings of DL. A literal in ASP is a formula of the form A or $\neg A$, where A is a function-free first-order atom. ASP supports two types of negations:
  1. Classical or strong negation ($\neg$): The negation is used for specifying explicit negative information. In other words, the ASP program P concludes $\neg A$, if $\neg A$ is explicitly inferred from it;
  2. Non-monotonic negation as failure (**not**): **not** A means that A can be false. In other words, A is false or it is not possible to determine its true value.

  An answer set program P is a finite set of rules of the form:

  $$B_1, \cdots, B_m, \textbf{not } B_{m+1}, \cdots, \textbf{not } B_n \rightarrow H_1 \vee \cdots \vee H_k,$$

  where B_i and H_j are literals. Comma indicates logical AND and $\vee$ indicates logical OR. Operator **not** can be used to specify the default true value for a predicate. As an example, the closed-world assumption for a predicate A can be expressed using the following rule:

$$\textbf{not } A(x) \rightarrow \neg A(x).$$

- MKNF: The logic of Minimal Knowledge and Negation as Failure (MKNF) was proposed by Lifschitz [16] to unify different non-monotonic formalisms such as default logic, auto-epistemic logic, and logic programming. MKNF extends the first-order logic by **K** and **not** operators. In fact, **K** is a non-monotonic modal operator and $\mathbf{K}\varphi$ intuitively means that φ is *known to hold*, in which φ is an MKNF formula. Moreover, **not** is negation-as-failure operator. In other words, **K** operator specifies the minimal knowledge acquired based on existing evidences about a fact. If we have no evidence (or assertions) for holding a fact like φ, we can infer the negation of $\mathbf{K}\varphi$, which is specified as $\textbf{not}\varphi$ (negation as failure)-Thus, $\neg\mathbf{K}\ \varphi = \textbf{not}\varphi$. For example, the rule (from [1]) $\mathbf{K}\mathit{seasideCity}(x) \leftarrow \mathbf{K}\mathit{portCity}(x), \textbf{not}\neg\mathit{seasideCity}(x)$ says that port cities are on the seaside by default unless there is evidence to the contrary. More details about MKNF and its formal semantics can be found in [1,16].

Each DL and ASP knowledge base can be translated into MKNF. Such possibility was used in MKNF$^+$. Each MKNF$^+$ knowledge base is a pair $K = (O, P)$, where O is a DL knowledge base and P is a program (finite set of MKNF$^+$ rules). Predicates defined in O are called DL-predicates and other predicates are called non-DL-predicates. DL-predicates are unary or binary predicates, but non-DL-predicates are not bounded. Moreover, two types of modal atoms, namely **K**-atom and **not**-atom, are defined in this formalism. The semantic of MKNF$^+$ is based on the MKNF semantic. In fact, each part of MKNF$^+$ knowledge base, namely O or P, is translated into MKNF separately (see [1] for further explanation of the MKFN$^+$ semantics). The structure of an MKNF$^+$ rule is as follows:

$$B_1, \cdots, B_n \rightarrow H_1 \vee \cdots \vee H_m.$$

where, B_i can be a non-modal predicate, a **K**-atom, or a **not**-atom, whereas H_i would be either a non-modal predicate or a **K**-atom. To preserve the decidability of MKNF$^+$, the DL-safety restriction must be applied; each variable in a rule should appear in the body of the rule in some non-DL K-atom. The main idea of the DL-safety condition is to restrict the applicability of the rules only to the individuals that are explicitly specified by name in the knowledge base.

4. Online social network model

Using semantic technology and especially OWL language can facilitate knowledge sharing, knowledge reuse, and interoperability among systems. Due to the diversity of OSNs, it is not possible to cover all information of entities existing in various OSNs in a predefined ontology. The idea of dividing the ontology into the *upper-level ontology* and the *domain-specific ontology* can be used for resolving this issue. Therefore, our proposed ontology consists of two layers:

1. *Upper-level ontology* is a high-level ontology, which represents the main general entities and their attributes. Such entities can be used in various OSNs and are required for our proposed access control model;
2. *Domain-specific ontology* is a more detailed ontology, which describes specific concepts and their relationships in an OSN, further to the main concepts. Due to the existence of a variety of OSNs, considering this ontology facilitates the extension of upper-level ontology. In contrast to the upper-level ontology, which is fixed for all OSNs, the domain-specific ontology might be defined and customized for each OSN separately.

The upper-level ontology is represented in Figure 1 and consists of the following concepts:

- *Entity:* A virtual concept by which all other concepts are subsumed;
- *Subject:* This concept models all the subjects existing in an online social network;
- *Action:* By using this concept, various actions being done by subjects in OSNs are modeled;
- *Object:* Various resources are shared in OSNs. These types of resources are defined as subconcepts of *Object* concept;
- *Priority:* Users can define various priority labels for their access rules. These priority labels are defined as individuals of the concept *Priority*. In addition to Priority, the role *HasMorePriority* is defined in the upper-level ontology for representing relations among the defined priority labels.

Further to the above concepts, the role *Owns(Subject, Object)* and the assertion *Subject(Sys)* are included in the upper-level ontology. In POBAC, the subject who owns the object can define its related access control rules. The information of the object's owner can be modeled using the role *Owns*. Also, in order to distinguish system level from user level access control rules, the individual *Sys* is defined as a member of the *Subject* concept (for more details, see Section 6.2). The rules specified by *Sys* determine the system's general access control rules.

In addition to the upper-level ontology, a simple ontology for modeling of general OSNs is proposed in Figure 1 to prove the applicability of POBAC for OSNs.

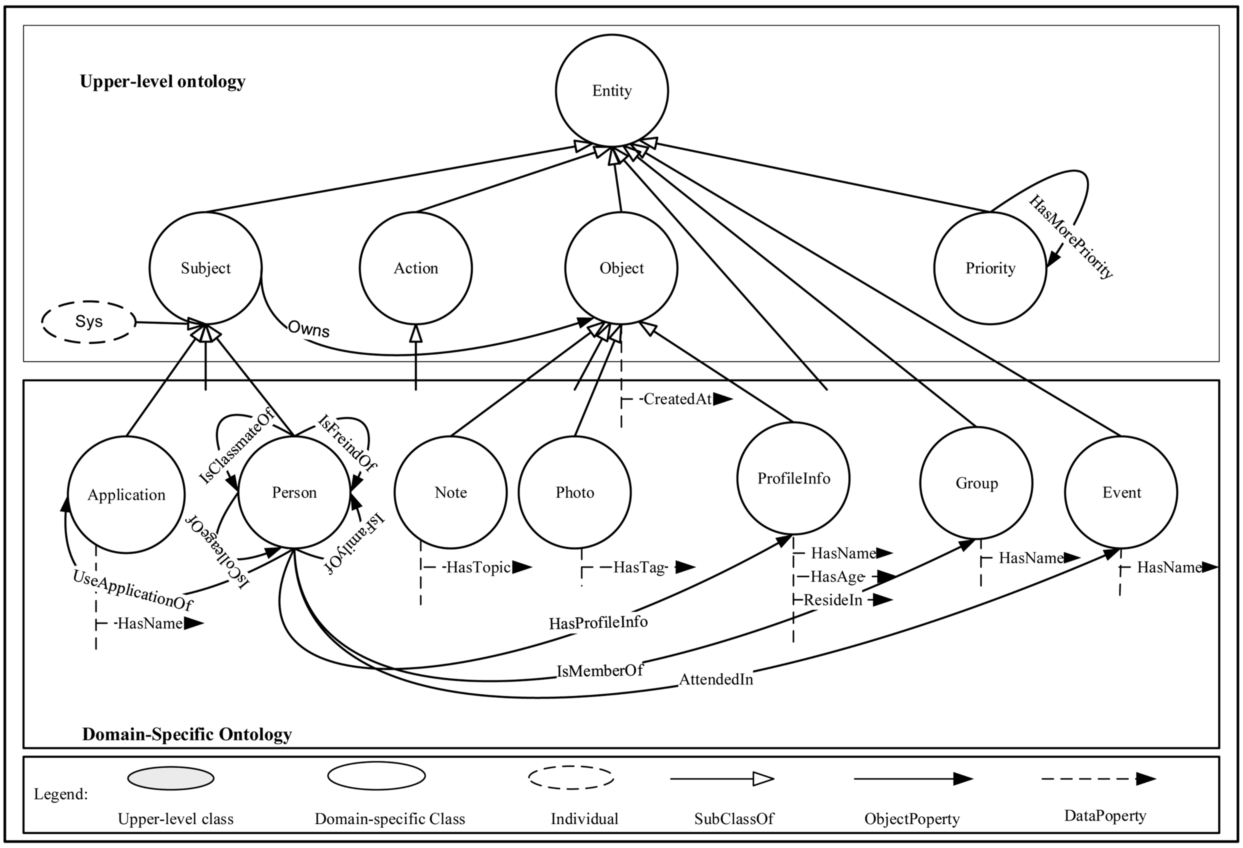

Figure 1. Upper-level and a simple ontology for online social networks.

The ontology can be easily extended to support other OSNs' requirements.

Various concepts such as *Note* are defined as subconcepts of *Object* in the ontology. Each object might have different properties. For example, the topic property for a note is defined using the object property *HasTopic*. Various applications can be created by developers using the APIs provided by OSNs. Users should be able to control which part of their information may be accessed by such applications. To consider the applications, subjects are categorized as *Person* and *Application*. Various types of groups and events are defined by users in OSNs. Furthermore, various relationships can be defined among users and other entities. For example, IsFriendOf is defined between two users or relations such as HasProfileInfo, IsMemberOf, and AttendIn and it can be defined between a person and individuals of the types ProfileInfo, Group, and Event, respectively. Hence, employing such an ontology enables the model to consider different types of relationships between users and infer appropriate access policies without wondering about the dynamic nature of such relationship instances in OSNs. Note that in POBAC, Description Logic (DL) is used for specifying the concepts and their relationships (as terminological box–TBox–in DL knowledge-base), and the instances of the defined concepts and relationships (as some assertions in assertion box–ABox–of DL knowledge-base), which provides inference capability for the model.

Due to the use of DL for ontology definition and MKNF$^+$ predicates and rules for security policy specification, two types of predicates are defined, namely, description logic predicates (which are represented by upper case names in the paper) and non-DL-predicates (which are represented by lower case names in the paper). Moreover, all individuals are defined as members of the non-DL-predicate e. For example, if Alice shares a new note called $Note_1$ with her friends, Note ($Note_1$) and e ($Note_1$) are added to the knowledge base. By doing so and using the predicate e in the body of access control rules, the DL-safety restriction (which is introduced in Section 3) is satisfied in them.

5. Formal specifications of POBAC

Before introducing the proposed policy specification language, formal definition of the proposed access control model is presented. Figure 2 represents the overall structure of the model and all of the model components, which are defined formally and precisely in the rest of this section.

Definition 5.1. *Prioritized Ontology-Based Access*

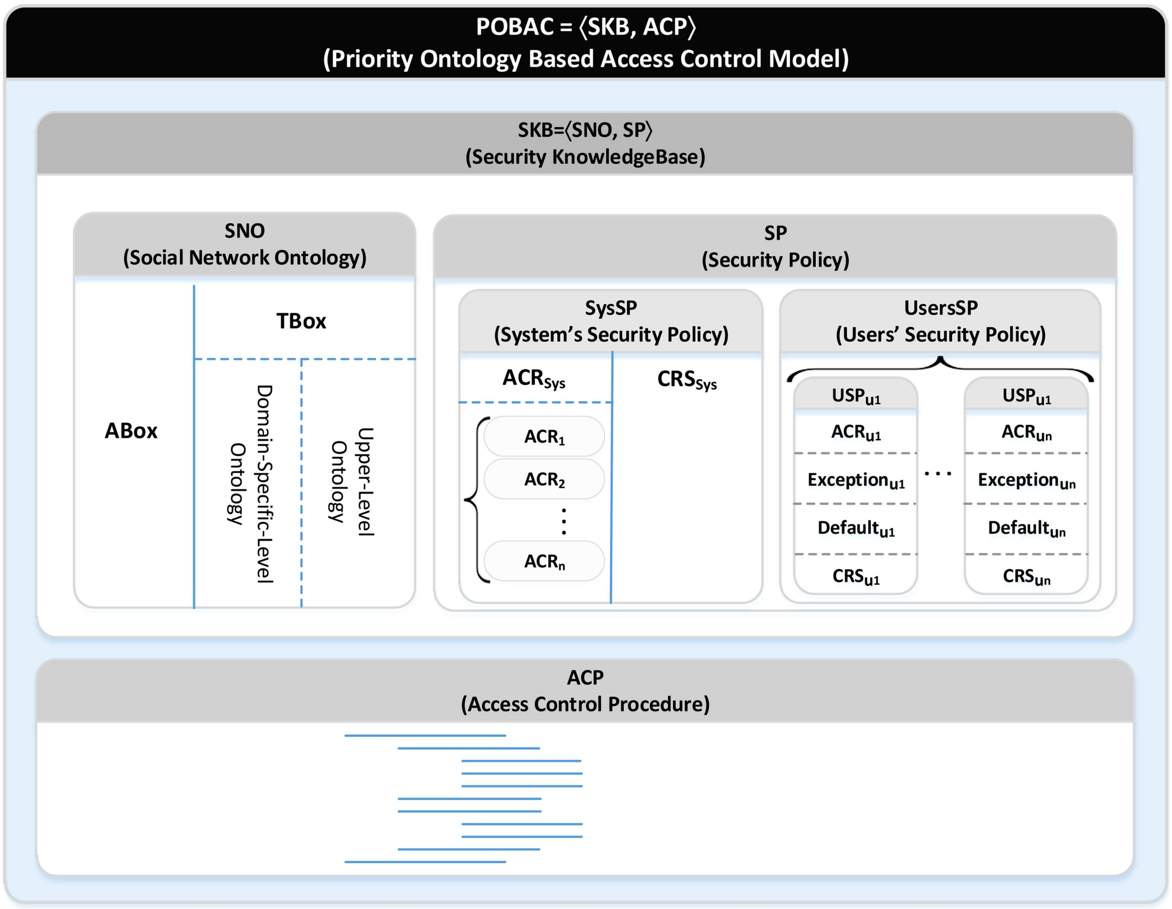

Figure 2. Overall structure of POBAC model.

Control (POBAC) model: The POBAC model is defined as a binary $\langle SKB, ACP\rangle$, where:

- $SKB = \langle SNO, SP\rangle$ is a security knowledge base. SKB includes the Social Network Ontology (SNO) as well as Security Policy (SP). SNO and SP are formally defined in Definitions 5.2 and 5.4, respectively;
- ACP is an access control procedure. According to this procedure, the system decides to whether grant or deny an access request. ACP steps are defined in Section 7.1.

In POBAC, each OSN with its defined ontology has its own security knowledge base. The knowledge base contains a social network ontology, which is defined in DL, and the OSN's security policy, which is defined in the logic program. In fact, SKB is an MKNF$^+$ knowledge base. In what follows, these two elements of security knowledge base are defined more formally.

Definition 5.2. *Social Network Ontology (SNO):* SNO is defined as a binary $\langle TBox, ABox\rangle$, where TBox is the terminological box and ABox is the assertional box in MKNF$^+$:

- TBox includes, at least, the concepts and the relationships which are shown in the upper-level ontology in Figure 1. In addition, TBox includes other concepts and relationships defined in the domain-specific ontology of OSN;
- ABox is the set of assertions about individuals existing in OSN and includes *Subject(Sys)*.

In OSN, the provider and the users must be able to define their own access control rules. For this purpose, access control rules structure and security policy are defined formally in the rest.

Definition 5.3. *Access Control Rule (ACR):* an ACR is defined as an MKNF$^+$ rule as follows:

$$B_1, \cdots, B_n \rightarrow H,$$

where, B_i can be a **K**-atom or a **not**-atom whereas H would be a **K**-atom. Furthermore, the rule must satisfy the DL-safety restriction. Different types of required rules in our proposed model are defined in the next section.

Definition 5.4. *Security Policy (SP):* A security policy consists of system and users security policy. SP

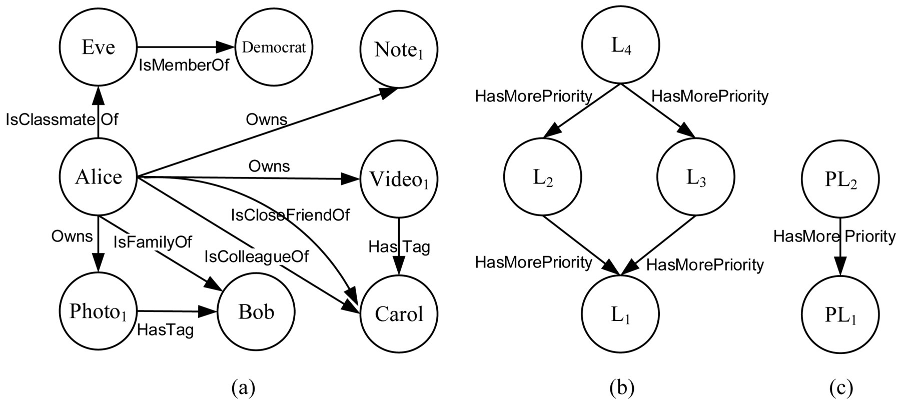

Figure 3. (a) An instance of OSN. (b) Priority labels defined by Alice. (c) Priority labels defined for enforcing denial and positive takes precedence strategy.

is modeled as a binary tuple $\langle SysSP, UsersSP\rangle$ where:

- System Security Policy (SysSP): SysSP is a binary $\langle ACR_{Sys}, CRS_{Sys}\rangle$ where:
  - $\text{ACR}_{Sys} = \{r|r$ is an ACR defined by the system$\}$;
  - CRS_{Sys} is a conflict resolution strategy used for resolving conflicts among access control rules defined by the system.
- Users Security Policy (UsersSP): UsersSP is a set of users' security policies. UsersSP is formally defined in Definition 5.5.

Definition 5.5. *User Security Policy (UsersSP):* In an OSN, a user must be able to define the security policy for his/her resources such as notes, photos, and videos. UsersSP is a set of policies defined by users. The security policy defined by each user such as u_i is modeled as a four-tuple $\langle ACR_{u_i}, Exception_{u_i}, Default_{u_i}, CRS_{u_i}\rangle$ containing the following elements:

- ACR_{u_i} is a set of access control rules defined by u_i;
- $Exception_{u_i}$ is a set of concrete access control exceptions defined by u_i;
- $Default_{u_i}$ is a default policy selected by u_i. It is possible that for a request neither permission nor prohibition would be inferred from the SKB. In this case, if open policy is selected, the requested access will be granted. Otherwise, it will be denied;
- CRS_{u_i} is a conflict resolution strategy used for resolving conflicts among access control rules defined by the user.

As POBAC supports the specification of both permissions and prohibitions; the occurrence of conflicts among the derived authorizations is possible. Thus, approaches for resolving these conflicts should be taken into account. More details of the proposed conflict resolution approach are presented in the next section.

6. Access control policy specification

In POBAC, security policy is divided into two parts, namely, users and system security policy. All of these policies accompanied by other required facts for access control are specified in the system using the MKNF$^+$ rules and predicates. The following sections describe the users and system security policy definitions respectively. Furthermore, the proposed approaches to resolve the possible conflicts among various rules are described. All the examples given in this section are based on the OSN graph and priority labels, which are represented in Figure 3 and described using DL and non-DL-predicates in Table 1.

6.1. User access control policy

In the POBAC model, by using basic access control rules, exceptions, and default policy, users determine which subjects are authorized to do a specific action on an object. For enforcing users' and the system's policies, several rules are defined. A rule can be divided into an antecedent and a consequent. If conditions in the antecedent of a rule are satisfied by the knowledge base, predicates in the consequent of the rule will be added to the knowledge base.

In MKNF$^+$, unlike DL-predicates, non-DL-predicates do not have any arity restriction.

Hence, we use non-DL-predicates for defining predicates with arbitrary arity in our access control rules. Variables, which start with a lower case letter, and individuals, which start with an upper case letter, can be used as arguments of these predicates. Before describing the structure of users' and system's policies, priory labels, which are used in the definition of the basic access control rules, will be described.

6.1.1. Priority labels

Users can define various priority labels and assign them to their access control rules. By using these priorities, possible conflicts among access control rules are resolved. Two approaches can be supposed for

Table 1. Describing the OSN represented in Figure 3 with DL and non-DL-predicates.

DL	**TBox**		Entity $\equiv$ Subject $\sqcup$ Action $\sqcup$ object $\sqcup$ Priority; Subject $\equiv$ Application $\sqcup$ Person; object $\equiv$ Photo $\sqcup$ Note $\sqcup$ Video
	ABox	**OSN**	Person(Alice), Person(Bob), Person(Carol), Person(Eve), Note(Note$_1$), Video(Video$_1$), Group(Democrat), Photo(Photo$_1$), Photo(FamilyPhoto$_1$), Owns(Alice,Photo$_1$), Owns(Alice,Note$_1$), Owns(Alice,Video$_1$), IsColleagueOf(Alice,Carol), IsFriendOf(Alice,Carol), IsMemberOf(Eve,Democrat), IsClassmateOf(Alice,Eve), IsFamilyOf(Alice,Bob), HasTag(FamilyPhoto$_1$,Eve), HasTag(Photo$_1$,Bob), Priority(PL_1), Priority(PL_2), HasMorePriority(PL_2,PL_1), Action(Read), $\cdots$
		Alice	Priority(L_1), Priority(L_2), Priority(L_3), Priority(L_4), HasMorePriority(L_4,L_2), HasMorePriority(L_4,L_3), HasMorePriority(L_2,L_1),HasMorePriority(L_3,L_1)
Non-DL	e(Alice), e(Bob), e(Carol), e(Eve), e(Photo$_1$), e(FamilyPhoto$_1$), $\cdots$		

defining priority labels for rules:

1. The OSN provider can define a set of priority labels as well as their inter-relationships. Hence, all of its users must use the same set of priority labels in this case;
2. Each user can define his/her set of priority labels and their inter-relationships.

To provide more flexibility, we suppose that priority labels can be defined by each user. Using the *HasMorePriority* predicate, users can define the relationship between two priority labels. *HasMorePriority*(x,y) determines that the priority level of label x is higher than the priority level of label y. Relationships defined for priority labels are directive, irreflexive, transitive, and acyclic. To enforce transitivity relations among priority labels defined by a user, Rule (1) in the following is defined. Defining cyclic relationships among priority labels usually occurs when a user makes a mistake in the definition of priority labels and should be eliminated. Such conflicts are supposed to be identified when a new relationship between two priority labels is being inserted into the knowledge base. To detect such anomaly in knowledge base, a propositional symbol *error* and an integrity constraint represented in Rule (2) are defined. According to this rule, the *error* is inferred from knowledge base provided that the relationships defined among the priority labels create a cycle. In this case, the defined relationship is rejected and a warning message is shown to the user. These rules satisfy DL-safety restriction because p_1, p_2, and p_3, which are our variables, occur in e, which is a non-DL-predicate in the antecedent of the rules:

$$\mathbf{K}\ e(p_1),\mathbf{K}\ e(p_2),\mathbf{K}\ e(p_3),\ \mathbf{K}\ HasMorePriority(p_1,p_2),\ \mathbf{K}\ HasMorePriority(p_2,p_3) \rightarrow \mathbf{K}\ HasMorePriority(p_1,p_3), \tag{1}$$

$$\mathbf{K}\ e(p_1),\mathbf{K}\ e(p_2),\mathbf{K}\ HasMorePriority(p_1,p_2),\ \mathbf{K}\ HasMorePriority(p_2,p_1) \rightarrow \mathbf{K}\ error. \tag{2}$$

6.1.2. Basic access control rule

These rules are defined by owners to determine which users are authorized to access resources shared in OSN. On the head of basic access control rules, *permit* and *prohibit* appear (see Table 2). An authority who grants a permission, a user who takes the permission, an action requested on the resource, the requested resource, and priority are parameters of *permit* and *prohibit* predicates. Various actions, namely, create, delete, read, and share are considered as the actions that subjects can do when they are in OSNs. Such actions are added to the knowledge base as individuals of the *Action* concept. In basic access control rules, various parameters such as types and properties of subjects and objects are used for defining the rules. For specifying the priority of a rule, one of the priority labels should be assigned to each rule. For example, suppose the scenario that Alice does not tend to let her colleagues to access her videos and she assigns the

Table 2. Predicates defined for enforcing access control policies.

Type	Predicates
Basic authorization	*permit*(Subject, Subject, Action, Object, Priority) *prohibit*(Subject, Subject, Action, Object, Priority)
Priority enforcement	*h-permit*(Subject, Subject, Action, Object, Priority) *h-prohibit*(Subject, Subject, Action, Object, Priority)
Basic rules conflict resolution	*b-permit*(Subject, Subject, Action, Object) *b-prohibit*(Subject, Subject, Action, Object)
Exceptions	*e-permit*(Subject, Subject, Action, Object) *e-prohibit*(Subject, Subject, Action, Object)
Final authorization result	*fd-permit*(Subject, Action, Object) *fd-prohibit*(Subject, Action, Object)
Integrity constraint	error

priority label L_2 to this policy. Alice defines this policy by using the user interface designed for defining users' privacy preferences. Then, according to the received information from Alice, Rule (3) is generated by the system and is added to Alice's rule set:

$$\mathbf{K}\ e(rsc), \mathbf{K}\ e(sbj), \mathbf{K}\ Video(rsc), \mathbf{K}\ IsColleagueOf(Alice, sbj) \rightarrow \mathbf{K}\ prohibit(Alice, sbj, READ, rsc, L_2). \quad (3)$$

6.1.3. Conflict resolution strategy

Access control rules in some conditions might be in conflict with each other. Conflict between two rules occurs when one of these rules prohibits while the other one permits an action on a resource. Regarding that various properties can be used in the body of the rules, authorities can define complex access control rules. Therefore, it is possible that two rules might not conflict with each other in all states or conflict might occur just in special conditions. For resolving the conflict among rules, the following predicates are defined (as represented in Table 2):

1. *h-permit* and *h-prohibit*: Users can define permission or prohibitions with various priority labels. To represent that a permission (prohibition) with either higher or incomparable priority level exists in the knowledge base, the *h-permit* (*h-prohibit*) predicate is defined;
2. *b-permit* and *b-prohibit*: After resolving conflicts among basic access control rules, for a request, *b-permit* or *b-prohibit* predicates might be inferred. In fact, these predicates are considered as the output of conflict resolution among the rules defined by each user.

Various priority labels might be defined in the system. To show whether permissions or prohibitions with the higher priority labels exist in the knowledge base or not, propagation rules are defined. Two scenarios are conceivable for comparison of the priority labels assigned to the contradictory access control rules:

1. The two priority labels are comparable and one of them has higher priority. In this case, Rule (4) and Rule (5) can be used to demonstrate the existence of the privilege with the higher priority label in the knowledge base.

$$\mathbf{K}\ e(p_2), \mathbf{K}\ permit(au, sbj, act, rsc, p_1), \mathbf{K}\ HasMorePriority(p_1, p_2) \rightarrow \mathbf{K}\ h\text{-}permit(au, sbj, act, rsc, p_2), \quad (4)$$

$$\mathbf{K}\ e(p_2), \mathbf{K}\ prohibit(au, sbj, act, rsc, p_1)\ \mathbf{K}\ HasMorePriority(p_1, p_2) \rightarrow \mathbf{K}\ h\text{-}prohibit(au, sbj, act, rsc, p_2). \quad (5)$$

2. The two priority labels are the same. In this case, a user may attribute greater priority to either permission or prohibition. If a denial-takes-precedence policy is chosen, Rule (6) can be used for giving higher priority to the negative privileges than to positive privileges. Otherwise, Rule (7) can be used:

$$\begin{aligned}&\mathbf{K}\ \textit{permit}(au, sbj, act, rsc, p),\\ &\quad \mathbf{K}\ \textit{prohibit}(au, sbj, act, rsc, p),\\ &\quad \mathbf{not}\ \textit{h-prohibit}(au, sbj, act, rsc, p),\\ &\quad \mathbf{not}\ \textit{h-permit}(au, sbj, act, rsc, p)\\ &\quad \rightarrow \mathbf{K}\ \textit{h-prohibit}(au, sbj, act, rsc, p), \qquad (6)\end{aligned}$$

$$\begin{aligned}&\mathbf{K}\ \textit{permit}(au, sbj, act, rsc, p),\\ &\quad \mathbf{K}\ \textit{prohibit}(au, sbj, act, rsc, p),\\ &\quad \mathbf{not}\ \textit{h-prohibit}(au, sbj, act, rsc, p),\\ &\quad \mathbf{not}\ \textit{h-permit}(au, sbj, act, rsc, p)\\ &\quad \rightarrow \mathbf{K}\ \textit{h-permit}(au, sbj, act, rsc, p). \qquad (7)\end{aligned}$$

3. The two priority labels are not comparable. Similar to the previous case, a user may attribute greater priority to either permission or prohibition. If a denial-takes-precedence policy is chosen, Rule (8) can be used for giving higher priority to the negative privileges than to the positive privileges. Otherwise, Rule (9) can be used:

$$\begin{aligned}&\mathbf{K}\ \textit{permit}(au, sbj, act, rsc, p1),\\ &\quad \mathbf{K}\ \textit{prohibit}(au, sbj, act, rsc, p2),\\ &\quad \mathbf{not}\ \textit{h-prohibit}(au, sbj, act, rsc, p1),\\ &\quad \mathbf{not}\ \textit{h-permit}(au, sbj, act, rsc, p2),\\ &\quad \mathbf{not}\ HasMorePriority(p_1, p_2),\\ &\quad \mathbf{not}\ HasMorePriority(p_2, p_1)\\ &\rightarrow \mathbf{K}\ \textit{h-prohibit}(au, sbj, act, rsc, p1), \qquad (8)\end{aligned}$$

$$\begin{aligned}&\mathbf{K}\ \textit{permit}(au, sbj, act, rsc, p1),\\ &\quad \mathbf{K}\ \textit{prohibit}(au, sbj, act, rsc, p2),\\ &\quad \mathbf{not}\ \textit{h-prohibit}(au, sbj, act, rsc, p1),\\ &\quad \mathbf{not}\ \textit{h-permit}(au, sbj, act, rsc, p2),\\ &\quad \mathbf{not}\ HasMorePriority(p_1, p_2),\\ &\quad \mathbf{not}\ HasMorePriority(p_2, p_1)\\ &\quad \rightarrow \mathbf{K}\ \textit{h-permit}(au, sbj, act, rsc, p2). \qquad (9)\end{aligned}$$

If permit (prohibit) is inferred for a specific priority label and *h*-prohibit (*h*-permit) is not inferred for that label, then *b*-permit (*b*-prohibit) will be inferred. Therefore, Rules (10) and (11) can be used for conflict resolution among basic access control rules with various priority labels:

$$\begin{aligned}&\mathbf{K}\ \textit{permit}(au, sbj, act, rsc, p),\\ &\quad \mathbf{not}\ \textit{h-prohibit}(au, sbj, act, rsc, p)\\ &\quad \rightarrow \textit{b-permit}(au, sbj, act, rsc), \qquad (10)\end{aligned}$$

$$\begin{aligned}&\mathbf{K}\ \textit{prohibit}(au, sbj, act, rsc, p),\\ &\quad \mathbf{not}\ \textit{h-permit}(au, sbj, act, rsc, p)\\ &\quad \rightarrow \textit{b-prohibit}(au, sbj, act, rsc). \qquad (11)\end{aligned}$$

6.1.4. Exceptions

Using attributes of subjects and objects (by their types and properties) in the definition of access control rules, instead of using their identities, eases the manageability of access control policies. However, in some cases, we need more precise policies. For illustration, users can join various groups in an OSN. Moreover, being a member of a group might be used as a parameter in definition of an access control rule. As members of these groups are not determined by the user, it is probable that a user may not tend to share a specific object to all members of a group. This type of policy is called exception policy and it can be considered as a white and black list, which is defined for protecting users' resources.

As represented in Table 2, *e-permit* and *e-prohibit* predicates can be used by users for defining exceptions. All the input parameters of these predicates are individuals and no variables are used as parameters of these predicates. In fact, although access control rules are defined based on the attributes of users and resources, the exception rules are just allowed to be defined based on the identities of users and resources in this model. For instance, suppose the case that Alice does not tend to let Eve to see $Note_1$. This policy is defined as:

$$\textit{e-prohibit}(Alice, Eve, READ, Note_1). \qquad (12)$$

To prevent occurring conflicts among exceptions, a new exception is inserted into the knowledge base provided that it does not have any conflicts with other defined exceptions. An integrity rule shown in Rule (13) is defined for this purpose. In other words, by adding a new exception, no *error* should be inferred from the knowledge base. Otherwise, the defined exception causes of such conflicts will be eliminated and a warning message is shown to the user:

$$\begin{aligned}&\mathbf{K}\ \textit{e-permit}(au, sbj, act, rsc),\\ &\quad \mathbf{K}\ \textit{e-prohibit}(au, sbj, act, rsc) \rightarrow \mathbf{K}\ \textit{error}. \qquad (13)\end{aligned}$$

6.1.5. Default policy

After conflict resolution among various access control rules, *fd-permit* and *fd-prohibit* predicates (as represented in Table 2) will be inferred from the knowledge base. In some situations, it is possible to infer neither permissions nor prohibitions for a specific request from the defined access control rules. In this situation, based on the open or close policy which is defined by users, the system decides whether the requested access should be granted or denied. Corresponding rules for enforcing open and close policies are represented in Rule (14) and Rule (15), respectively:

$$\textbf{not}\ \textit{fd-prohibit}(sbj, act, rsc) \rightarrow \mathbf{K}\ \textit{fd-permit}(sbj, act, rsc), \quad (14)$$

$$\textbf{not}\ \textit{fd-permit}(sbj, act, rsc) \rightarrow \mathbf{K}\ \textit{fd-prohibit}(sbj, act, rsc). \quad (15)$$

6.2. System access control policy

Each OSN provider might have specific access control rules for its framework. To distinguish system level from user level access control rules, the individual *Sys* is defined as a member of the *Subject* concept. As an example, the system might define the policy that owners and users tagged on a resource are authorized to access it. Rules (16) and (17) are defined for enforcing such policies. In these rules, the priority label PL_1 is considered for the last parameter of permit:

$$\mathbf{K}\ e(sbj), \mathbf{K}\ e(rsc), \mathbf{K}\ Person(sbj), \mathbf{K}\ Object(rsc), Owns(sbj, rsc) \rightarrow \mathbf{K}\ permit(Sys, sbj, READ, rsc, PL_1), \quad (16)$$

$$\mathbf{K}\ e(sbj), \mathbf{K}\ e(rsc), \mathbf{K}\ Person(sbj), \mathbf{K}\ Object(rsc), HasTag(rsc, sbj) \rightarrow \mathbf{K}\ permit(Sys, sbj, READ, rsc, PL_1). \quad (17)$$

User level rules are usually more complex than system level rules. In the system level, simpler conflict resolution strategies can be used. In the proposed model, denial or positive takes precedence are considered as two possible approaches for resolving conflicts among the access control rules defined in the system level. As shown in Figure 3(c), PL_2 has higher priority than PL_1. To enforce *denial takes precedence* strategy, PL_2 should be assigned to the negative rules and PL_1 should be assigned to positive ones. To have *positive takes precedence* strategy, the inverse approach should be followed.

7. Access control procedure and time complexity analysis

To enforce access control policy rules specified in this model, an access control procedure should be introduced. The time overhead of employing an access control system (developed based on the proposed access control model) depends on the time complexity of the procedure proposed for this purpose. In the following, after introducing the proposed access control procedure in this model, time complexity of this procedure is analyzed.

7.1. Access control procedure

As represented in Figure 4, various priorities are considered for each rule type. In fact, the system level rules have higher priority than the other ones. Among the user level rules, exceptions have higher priority than the basic access control rules and both of these rules have higher priority than the default policy. For enforcing these priorities, a set of rules are defined:

1. If the rules defined at the system level permit (prohibit) a request, the request will be granted (denied):

$$\mathbf{K}\ \textit{b-permit}(Sys, sbj, act, rsc)$$

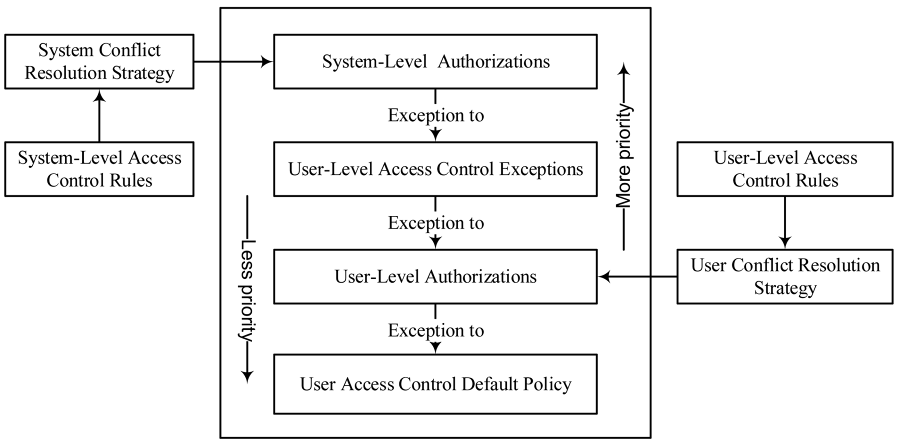
System Conflict Resolution Strategy
System-Level Access Control Rules
System-Level Authorizations
Exception to
User-Level Access Control Exceptions
Exception to
User-Level Authorizations
Exception to
User Access Control Default Policy
More priority
Less priority
User-Level Access Control Rules
User Conflict Resolution Strategy

Figure 4. Comparison among the priority of various access control rules.

$$\rightarrow \mathbf{K}\ \textit{fd-permit}(sbj, act, rsc), \tag{18}$$

$$\mathbf{K}\ \textit{b-prohibit}(Sys, sbj, act, rsc)$$

$$\rightarrow \mathbf{K}\ \textit{fd-prohibit}(sbj, act, rsc). \tag{19}$$

2. If the rules defined at the system level neither permit nor prohibit the request, the decision is made based on the exception rules (if the related ones exist):

$$\begin{aligned}&\textbf{not}\ \textit{b-permit}(Sys, sbj, act, rsc),\\ &\quad \textbf{not}\ \textit{b-prohibit}(Sys, sbj, act, rsc),\\ &\quad \mathbf{K}\ \textit{e-permit}(au, sbj, act, rsc)\\ &\quad \rightarrow \mathbf{K}\ \textit{fd-permit}(sbj, act, rsc),\end{aligned} \tag{20}$$

$$\begin{aligned}&\textbf{not}\ \textit{b-permit}(Sys, sbj, act, rsc),\\ &\quad \textbf{not}\ \textit{b-prohibit}(Sys, sbj, act, rsc),\\ &\quad \mathbf{K}\ \textit{e-prohibit}(au, sbj, act, rsc)\\ &\quad \rightarrow \mathbf{K}\ \textit{fd-prohibit}(sbj, act, rsc).\end{aligned} \tag{21}$$

3. If the rules defined in the system level and the exceptions neither permit nor prohibit the request, the decision is made based on the user level basic access control rules.

$$\begin{aligned}&\textbf{not}\ \textit{b-permit}(Sys, sbj, act, rsc),\\ &\quad \textbf{not}\ \textit{b-prohibit}(Sys, sbj, act, rsc),\\ &\quad \textbf{not}\ \textit{e-permit}(au, sbj, act, rsc),\\ &\quad \textbf{not}\ \textit{e-prohibit}(au, sbj, act, rsc),\\ &\quad \mathbf{K}\ Person(au),\\ &\quad \mathbf{K}\ \textit{b-prohibit}(au, sbj, act, rsc)\\ &\quad \rightarrow \mathbf{K}\ \textit{fd-prohibit}(sbj, act, rsc),\end{aligned} \tag{22}$$

$$\begin{aligned}&\textbf{not}\ \textit{b-permit}(Sys, sbj, act, rsc),\\ &\quad \textbf{not}\ \textit{b-prohibit}(Sys, sbj, act, rsc),\\ &\quad \textbf{not}\ \textit{e-permit}(au, sbj, act, rsc),\\ &\quad \textbf{not}\ \textit{e-prohibit}(au, sbj, act, rsc),\\ &\quad \mathbf{K}\ Person(au),\\ &\quad \mathbf{K}\ \textit{b-permit}(au, sbj, act, rsc)\\ &\quad \rightarrow \mathbf{K}\ \textit{fd-permit}(sbj, act, rsc).\end{aligned} \tag{23}$$

4. As the last step, if the aforementioned rules neither permit nor prohibit the request, the decision is made based on the default policy. Rule (14) or Rule (15) is added to the knowledge base if a user defines open or close policy, respectively.

Figure 5 shows our proposed architecture for POBAC. An authority (owner) can use Policy Administration Point (PAP) to define his/her Access Control Rules (ACR), Conflict Resolution Strategy (CRS), default policy, and security labels. The steps which are necessary to be taken into account when a new priority label is defined by $user_i$ are as follows:

1. TBox, $user_i$'s priority labels and their relationships existing in ABox, and the integrity constraint rule (Rule (2)) are retrieved and are given to the PDP;
2. If *error* is not inferred by PDP, the defined priority label will be added to the knowledge base. Otherwise, a warning message is shown to the user and the defined priority label is discarded.

An access request is modeled in a triple ⟨*subject*, *action*, *object*⟩ form. Application interface is used for sending users access requests to the Policy Enforcement Point (PEP). According to SNO, as well as user and system level policies, the Policy Decision Point (PDP) decides whether the request must be granted or denied. Policy decisions made by PDP are enforced by PEP. After receiving an access request, the following steps are taken by PDP:

1. A query is sent to the knowledge base to discover the owner of the object;
2. The set of access control rules defined by the authority, the rules corresponding to the selected conflict resolution strategy and the default policy, the priority labels defined by the authority, and the OSN information (concepts, roles, and individuals) are retrieved from the knowledge base and are given to the PDP;
3. If the *fd-permit(subject, action, object)* predicate is inferred from the knowledge base, the request is granted. Otherwise, it is denied.

7.2. Time complexity analysis

The complexity of the access control procedure in our proposed model is completely dependent on time complexity of reasoning in MKNF$^+$ logic. Motik et al. [1] proved that the complexity of reasoning in MKNF$^+$ depends on the complexity of reasoning in its underpinning Description Logic (DL). If we take $\mathcal{SHOIN}$ as a widely used DL into account, the complexity of reasoning in this logic would be NExpTime-complete.

Although the time complexity of access control procedure in this model is NExpTime-complete, it

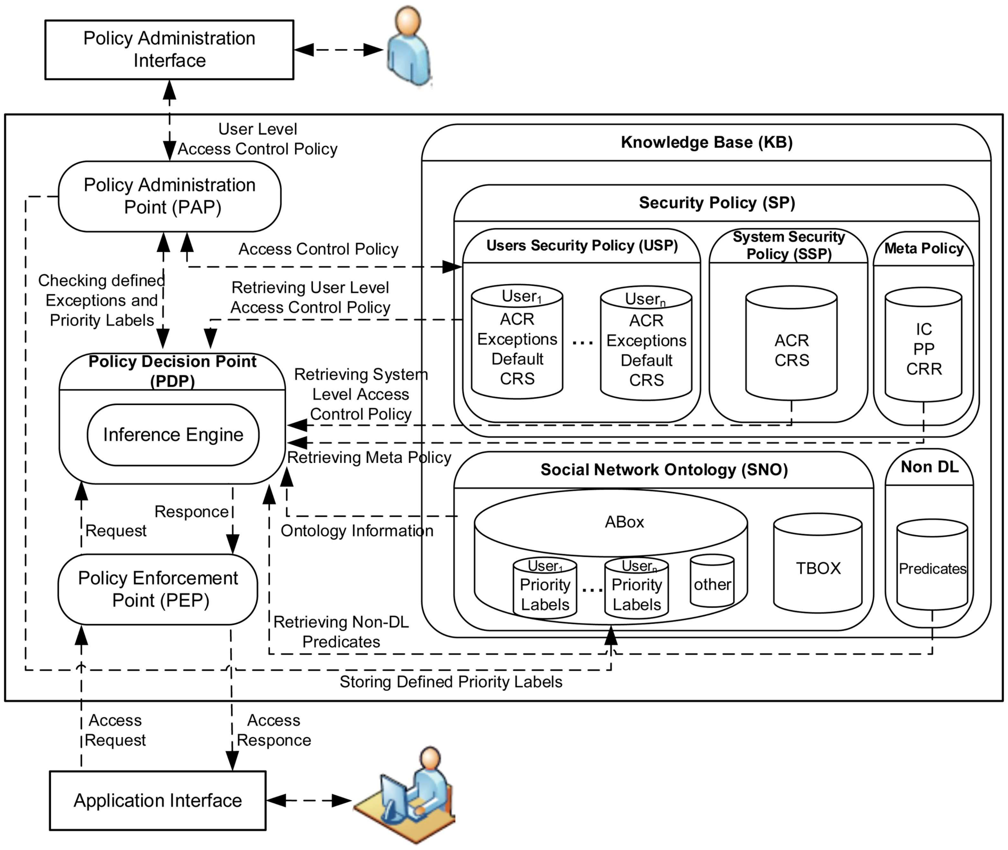
Policy Administration Interface
User Level Access Control Policy
Policy Administration Point (PAP)
Access Control Policy
Checking defined Exceptions and Priority Labels
Retrieving User Level Access Control Policy
Policy Decision Point (PDP)
Inference Engine
Retrieving System Level Access Control Policy
Retrieving Meta Policy
Knowledge Base (KB)
Security Policy (SP)
Users Security Policy (USP)
System Security Policy (SSP)
Meta Policy
User1 ACR Exceptions Default CRS
Usern ACR Exceptions Default CRS
ACR CRS
IC PP CRR
Social Network Ontology (SNO)
ABox
User1 Priority Labels
Usern Priority Labels
other
TBOX
Non DL
Predicates
Responce
Request
Ontology Information
Policy Enforcement Point (PEP)
Retrieving Non-DL Predicates
Storing Defined Priority Labels
Access Request
Access Responce
Application Interface

Figure 5. The architecture of proposed access control model for online social networks.

should be noted that for each access request, we just need to consider the rules of the user owning the requested resource as well as the general rules determined by the system administrator (named *Sys*) for all resources. Since the size of the part of the knowledge base considered for reasoning about a requested access is very small, such a complexity does not have significant overhead in access control procedure in practice.

8. Case study

To represent how our proposed access control model is applicable to OSNs, a case study is provided in the following. Suppose the social graph and the security levels represented in Figure 3. In this scenario, the system and user level rules are as follows:

1. System level: The system lets owners and people tagged in an object to see it (Rules (16) and (17)). In this scenario, as no prohibition is defined by the system, no conflict occurs among the rules;
2. User level: Alice defines four priority levels, which are represented in Figure 3(b). Moreover, she considers denial-takes-precedence strategy for conflict resolution among the access control rules. The list of access control rules defined by Alice is as follows:

(a) Exception: Alice prohibits Eve to access $Note_1$. Using Rule (12), this policy is enforced in the OSN;

(b) Basic Access Control Rules: Alice defines the following access control rules:

i. Alice has tendency to share her notes with people who are the members of Democrat group (Rule (24)):

$$\mathbf{K}\ e(sbj), \mathbf{K}\ e(rsc), \mathbf{K}e\ (g), \mathbf{K}Group(g),$$
$$\mathbf{K}\ IsMemberOf(sbj,g), \mathbf{K}\ Note(rsc)$$
$$\rightarrow \mathbf{K}\ prohibit(Alice, sbj, READ, rsc, L_1). \quad (24)$$

ii. Due to privacy concerns, Alice does not tend to let her colleagues to see her videos (Rule (3));

iii. Alice does not let her colleagues to see her photos and she assigns the priority label L_1 to this rule (Rule (25)).

$$\mathbf{K}\ e(sbj), \mathbf{K}\ e(rsc), \mathbf{K}\ Person(sbj),$$
$$\mathbf{K}\ IsColleagueOf(Alice, sbj),$$

$$\mathbf{K}\ Photo(rsc)$$

$$\rightarrow \mathbf{K}\ prohibit(Alice, sbj, READ, rsc, L_1). \tag{25}$$

iv. Alice lets her close friends to see her photos and she assigns the priority label L_2 to this rule:

$$\mathbf{K}\ e(sbj), \mathbf{K}\ e(rsc), \mathbf{K}\ Person(sbj),$$

$$\mathbf{not}\ IsCloseFriendOf(Alice, sbj),$$

$$\mathbf{K}\ Photo(rsc), HasTag(rsc, Bob)$$

$$\rightarrow \mathbf{K}\ permit(Alice, sbj, READ, rsc, L_2). \tag{26}$$

v. Alice does not let people out of her family to see her photos in which at least one member of the family is tagged. She assigns priority label L_4 to this rule (Rule (27)):

$$\mathbf{K}\ e(sbj), \mathbf{K}\ e(rsc), \mathbf{K}\ Person(sbj),$$

$$\mathbf{not}\ IsFamilyOf(Alice, sbj),$$

$$\mathbf{K}\ Photo(rsc), \mathbf{K}\ HasTag(rsc, per),$$

$$\mathbf{K}\ IsFamilyOf(Alice, per)$$

$$\rightarrow \mathbf{K}\ prohibit(Alice, sbj, READ, rsc, L_4). \tag{27}$$

(c) Default Rules: Alice defines close policy as her default policy.

Among the policy rules defined by the system and Alice, several conflicts occur:

1. According to Rule (3) defined by Alice, Carol's request to access $Video_1$ is rejected. However, according to Rule (17), which is defined in the system level, Carol is authorized to see $Video_1$. In this context, as the rules defined in the system level have higher priority than the rules defined in the user level, Carol's request to access $Video_1$ is authorized;
2. Rule (24) permits Eve to see $Note_1$. However, according to Rule (12), Eve is prohibited to see it. Since exceptions have higher priority than basic rules, Eve's request to access $Note_1$ will be denied;
3. Conflicts among Rules (25)-(27) occur if a user such as Carol, who has both the close friend and colleague relationships with Alice, sends a request to access $Photo_1$. In this context, regarding that Rule (27) has higher priority than Rules (25) and (26), Carol's request to access $Photo_1$ is denied.

9. Conclusion

OSNs are vastly used by different users with different flavors and relationships. In OSNs, we desire to have two types of access policies, namely, system's policies and users' personal policies. But, due to the existing various and complicated relationships between the users and the variety of their shared resources, they may not understand the effects and consequences of their defined security policies and checking the leakage of permissions to unintended users might be error-prone. In this paper, we proposed a Prioritized Ontology Based Access Control (POBAC) model for OSNs in which a non-monotonic logic, named MKNF$^+$, was leveraged for policy specification and inference. MKNF$^+$ is an integration of description logic and Answer Set Programming (ASP) rules; whereas DL is used in POBAC for modeling of OSNs' entities and their relationships (in two layers). Logical rules are employed for definition and non-monotonic inference of access control policies. Non-monotonic nature of the employed logical framework in the proposed model enables users to easily define their access control policies, exceptions, and default policy to protect their resources in OSNs. In the proposed model, incompatible and conflicting policies could be detected and their conflict could be resolved using different strategies, which are defined by a set of labels placed in the vertex of a non-cyclic, irreflexive, and transitive priority graph. Inference of access policies based on the hierarchies of entities (defined in the two-layered ontology) and possibility of determining default policies in this model guarantee the coverage and coherence of the defined policies for all types of resources.

Considering the mentioned properties of the proposed model, the following advantages could be enumerated for the proposed model in comparison to the traditional access control models [17-19] employed in OSNs:

- *Model liberality:* The proposed model is a general model, which could be specialized for each OSN by customizing the domain-specific ontology (specified as the lower layer ontology);
- *Ease of administration:* Using ontology of subjects, objects, and actions, a security officer can easily specify his/her policy rules in different levels of abstraction. The non-monotonic features of the employed logic enable defining exceptions and default access policies for easier definition and management of access policies.
- *Policy inference:* In the proposed model, the ontology of engaging entities is specified by description logic and policy rules are specified by MKNF$^+$ rules. Thus, we can infer implicit policy rules (from the explicit defined ones) based on the hierarchies

and semantic relationships defined between different entities in the ontology in presence of exceptions and default access policy;

- *More expressiveness:* The proposed access control policy language is more expressive than the one presented in the traditional DAC and RBAC models. For example, in this model, we can easily specify different complicated contextual conditions, which are impossible to define in traditional models.

References

1. Motik, B. and Rosati, R. "Reconciling description logics and rules", *Journal of the ACM (JACM)*, **57**(5), pp. 93-154 (2008).
2. Alizadeh, M., Javadi, S.A., Amini, M. and Jalili, R. "Policy specification and enforcement in online social networks using MKNF^{+}", In *Proceedings of the 9th International ISC Conference on Information Security and Cryptology (ISCISC'12)*, Tabriz, Iran, pp. 48-53 (2012).
3. Boyd, D.M. and Ellison, N.B. "Social network sites: definition, history, and scholarship", *Journal of Computer-Mediated Communication*, **13**(1), pp. 210-230 (2008).
4. Carminati, B., Ferrari, E. and Perego, A. "Enforcing access control in web-based social networks", *ACM Transactions on Information and System Security*, **13**(1), pp. 1-38 (2009).
5. Fong, P.W.L., Anwar, M. and Zhao, Z. "A privacy preservation model for facebook-style social network systems", In *Proceedings of the 14th European Conference on Research in Computer Security (ESORICS'09)*, Berlin, Heidelberg: Springer-Verlag, pp. 303-320 (2009).
6. Carminati, B., Ferrari, E., Heatherly, R., Kantarcioglu, M. and Thuraisingham, B., *A Semantic Web Based Framework for Social Network Access Control*, New York, NY, USA, pp. 177-186 (2009).
7. Masoumzadeh, A. and Joshi, J. "Ontology-based access control for social network systems", *International Journal of Information Privacy, Security and Integrity*, **1**(1), pp. 59-78 (2011).
8. Bertino, E., Jajodia, S., and Samarati, P. "Supporting multiple access control policies in database systems", In *Proceedings of the IEEE Symposium on Security and Privacy (SP'96)*, Washington, DC, USA, pp. 94-107 (1996).
9. Cuppens, F., Cuppens-Boulahia, N. and Miege, A. "Inheritance hierarchies in the orBAC model and application in a network environment", In *Proceedings of the Foundations of Computer Security (FCS'04)*, pp. 41-60 (2004).
10. Javadi, S.A., Amini, M. and Jalili, R. "Non-monotonocity in OrBAC through default and exception policy rules", in *Proceedings of the 9th International ISC Conference on Information Security and Cryptology (ISCISC'12)*, Tabriz, Iran, pp. 87-94 (2012).
11. Donini, F., Lenzerini, M., Nardi, D. and Schaerf, A. "Al-log: Integrating datalog and description logics", *Journal of Intelligent Information Systems*, **10**(3), pp. 227-252 (1998).
12. Rosati, R. "DL+log: Tight integration of description logics and disjunctive datalog", In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning*, pp. 68-78 (2006).
13. Levy, A. and Rousset, M. "CARIN: A representation language combining horn rules and description logics", In *Proceedings of the European Conference on Artificial Intelligence (ECAI'96)*, pp. 323-327 (1996).
14. Donini, F., Nardi, D. and Rosati, R. "Description logics of minimal knowledge and negation as failure", *ACM Transactions on Computational Logic (TOCL)*, **3**(2), pp. 177-225 (2002).
15. Baader, F., Calvanese, D., McGuinness, D., Nardi, D. and Patel-Schneider, P., *The Description Logic Handbook: Theory, Implementation and Applications*, Cambridge University Press (2003).
16. Lifschitz, V. "Nonmonotonic databases and epistemic queries", In *Proceedings of the 12th International Conference on Artificial Intelligence*, pp. 381-386 (1991).
17. Harrison, M.A., Ruzzo, W.L. and Ullman, J.D. "Protection in operating systems", *Communications of the ACM*, **19**(8), pp. 461-471 (1976).
18. Bell, D.E. and La Padula, L.J., *Secure Computer System: Unified Exposition and Multics Interpretation*, Mitre Corporation, Bedford, MA, Technical Report, ESD-TR-75-306 (1976).
19. Ferraiolo, D. and Kuhn, R. "Role-based access control", In *Proceedings of the 15th NIST-NCSC National Computer Security Conference*, Baltimore, MD, pp. 554-563 (1992).

Biographies

Mahdi Alizadeh obtained his BS degree in Computer Engineering from Amirkabir University of Technology in 2009 and his MS degree in Information Technology from Sharif University of Technology in 2011. He started his PhD at Eindhoven University of Technology in 2014. His research interests include privacy, access control, business process analysis, and auditing.

Morteza Amini received his PhD degree in Software Engineering (Information Security field) from Sharif University of Technology. He is currently an Assistant Professor in the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran. His research interests include database security, access control, intrusion detection, and formal methods in information security.

Seyyed Ahmad Javadi received his MS degree in

Software Engineering from Sharif University of Technology, Tehran, Iran. He is currently a PhD candidate in the Computer Science Department, Stony Brook University, USA, NY. His research focuses on cloud computing and application performance analysis.

Rasool Jalili received his PhD in Computer Science from The University of Sydney, Australia, in 1995. He then jointed the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran. He is now an Associate Professor, doing research in the areas of distributed computing and information security in his network security laboratory at Sharif Data & Network Security Lab.