International Journal of Advanced Smart Convergence Vol.7 No.4 66-74 (2018)
http://dx.doi.org/10.7236/IJASC.2018.7.4.66

IJASC 18-4-8

An extended Access Control with Uncertain Context

Woojun Kang

Dept. of Business Administration, KC University, Korea
wjkang@kcu.ac.kr

Abstract

While new information technology advances have made information access and acquisition methods much more diverse and easier, there are side effects that allow illegal access using diverse and high-performance tools. In order to cope with such threats, there are access control methods in database technology, and various studies are being conducted to extend traditional access control to cope with new computing environments. In this paper, we propose an extended access control with uncertain context-awareness. It enables appropriate security policy enforcement even if the contextual constraints specified by the security policy does not match those accompanied by access request query. We extract semantic implications from context tree, and define the argument that can quantitatively measure the semantic difference between two nodes in the context tree. It is used to semantically enforce the security policy, and to prevent the excessive authorization caused by the implication.

Keywords: *Database Security, Access Control, Context-aware, Tree Hierarchy*

1. Introduction

The remarkable development of information technology today such as ubiquitous technology, context-aware technology, and semantic technology makes it possible to distribute and share information more smoothly and freely [1,2,3]. As a result of these technological advancements, information access and acquisition methods of users are becoming more diverse and easier than ever before. But behind it, there are side effects that illegal user can invade in more diverse ways using better performance tools than before, thus posing a serious threat to system security and personal privacy.

Access control is a typical database security technology to cope with these security threats. In the traditional access control approach, there was no uncertainty in the information specified in the security policy. To investigate how to reflect or remove the uncertainty inherented in context information, and how to establish a theoretical foundation, many challenges are being undertaken by researchers and database vendors such as Oracle, IBM, and Microsoft [4,5,6].

In this paper, we propose a model that integrates uncertain context recognition and access control. In the

Manuscript Received: Sep. 23, 2018 / Revised: Sep. 28, 2018 / Accepted: Oct. 8, 2018
Corresponding Author: wjkang@kcu.ac.kr
Tel: +82-2-2600-2430, Fax: +82-2-2600-2439
Dept. of Business Administration, KC University, Korea

proposed model, semantic policy enforcement is performed based on the conceptual meaning of the context. The semantic policy enforcement means that It enables appropriate security policy enforcement even if the contextual constraints specified by the security policy does not match those accompanied by access request query.

The rest of this paper is organized as follows. Section 2 reviews some background research related to access control and context-awareness and Section 3 presents the proposed model including context hierarchical structure and the semantic context-awareness and the policy enforcement algorithm. Finally, Section 4 provides concluding remarks.

2. Related Works

Access control refers to controlling access to resources on a computer system or network. Traditional access control mechanisms can be classified into three types: mandatory access control, discretionary access control, and role-based access control [7]. The security policy in access control is a set of authorizations. An authorization is represented as a 4-tuple <*S, O, P, C*>, where subject *S* is a subject of the system, object *O* is a data, permission *P* is an access privilege to data object which is defined as a <sign, mode>, where sign={+,-}, mode={create, delete, read, write } and context *C* is an expression by Boolean operations over context predicates.

The Ubiquitous Computing Group defines the context as information on all environments, subjects, objects and conditions related to interaction between environment and users. An easy example of a context can be location, time, and temperature. The context aware infrastructure is required to include essential function to acquire and deliver situational information, and a robust formal context model to support complex reasoning, and interfaces for applications to be easily applied to various situations. The context-awareness system developed on GAIA middleware describes and processes the context based on the first-order logic [8]. The context-aware middleware is responsible for acquiring, compositing, and distributing the context information. The basic context is collected from the sensor device and composed to the context of the upper concept required by the application. The context is communicated using the above-mentioned context delivery protocol. The user searches for a necessary context item using a lookup service, which is a kind of directory service.

Typically contexts can be represented as first-order predicates. The name of the predicates corresponds to the type of context to be described. This convention allows us uniform representation for different kinds of contexts [9]. For example, context predicates are like *LocatedIn(Bob, room209), TemperatureOf(get_room#( ), 26^{o}C)*. A predicate consists of many terms. The values of each term in a predicate are actually constrained by the domain of contexts. Some of terms in a context predicate can be functions that return some value. This logical model for context using first-order logic is so powerful as to express a rich variety of contexts. Complex context expressions can be represented by combining Boolean operations such as conjunction, disjunction and negation. The predicate model also allows both universal and existential quantification over variables. This allows us a parameterization of context for representing of a much richer set of contexts. For example, an context expression is like *LocatedIn(Bob, room209)* ^ *WhileOnDuty(Bob)*, which refers to the context “Bob is in room 209 while on his duty”.

Generally, an ontology is a description of important concepts in a domain, crucial properties of each concept and restrictions on properties such as property cardinality, property value type, domain and range of a property. The rules extracted from ontology are used to infer new contexts from existing contexts.

Inference rules used in this paper are adapted from owl:ObjectProperty, which defines the relationship between many concepts in a specific domain. Table 1 presents the inference rules extracted from the context

ontology hierarchy by defining relationship between high concepts [10]. As another similar approach, CONON, also presents a generation of inference rules with the equivalence of OWL and Description Logic. By referring each relationship among context concepts in Table 1, we realize that the instances of high-level concepts can be inferred from the instances of low-level concepts.

Table1. Inference Rule extracted from Ontology

Relationship between concepts	Inference rule
EQUIVALENCE	if $C_i \equiv C_j$ then $C_i \Rightarrow C_j$
IS-PART-OF	if $C_j \in \{C_i\}$ then $C_i \Rightarrow C_j$
IS-A	if $C_i \subset C_j$ then $C_i \Rightarrow C_j$
UNION	if $C_j = C_1 \cup C_2 \cup ... \cup C_k$
	then $C_i \Rightarrow C_j$, i = 1,...,k
INTERSECTION	if $C_i = C_1 \cap C_2 \cap ... \cap C_k$
	then $C_i \Rightarrow C_j$, j = 1,...,k

3. Uncertain Context-aware Access Control System

3.1 Context Hierarchy Structure

Hierarchies can be extracted from an ontology that describes the relationship between concepts and concepts in a standard format, or can be explicitly configured by a security administrator or an application administrator. The inference rules that can be extracted from the ontology are proposed by Qin [10]. Table 1 shows the kinds of inference rules that can be extracted from the ontology. Figure 1 shows the ontology about the hospital room and ward layout and Figure 2 shows the context hierarchy tree generated by applying the inference rule to the corresponding ontology.

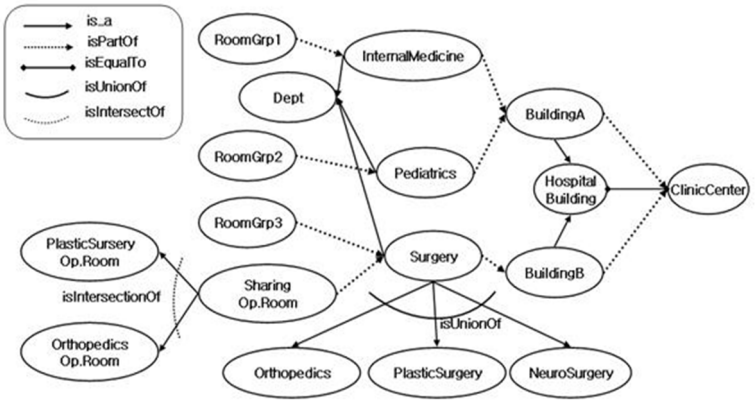

Figure 1. Hospital Ontology

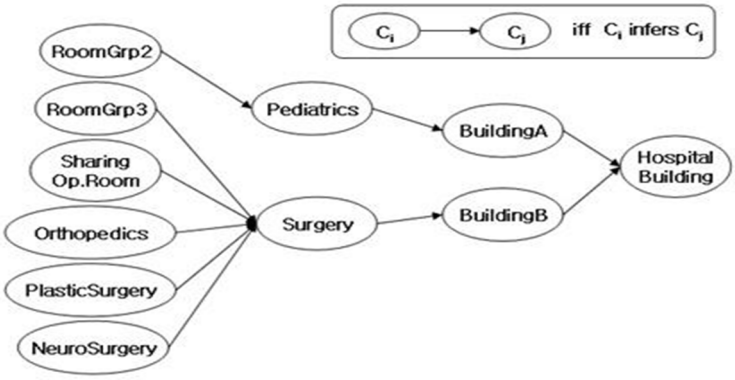

Figure 2. Context Hierarchy Tree

3.2 Semantic Difference Measurement

To quantitatively measure the semantic difference between two concepts on the hierarchical tree, we define the measurement factor, SD. The SD is the basis for setting system-defined thresholds and limits the implications to manage confidentiality and availability trade-offs, which are security requirements. The measurement factor SD is based on entropy E and uses modified form $2^{E(C)}$ for ratio computation. C is an instance (terminal node) or type (non-end node) on the hierarchical tree, and E (C) is the entropy of C. If C is an instance, then E(C) = 0 and thus $2^{E(C)} = 1$. If C is a type, then $2^{E(C)}$ means the computation cost to check whether a particular instance belongs to the type C. Here, a particular instance is a terminal node of the path to which type C belongs. In the hierarchical tree, belonging to the same path means that there is a semantic relation.

Definition 1. (Semantically Related) In a tree structure, a path is defined as a set of all nodes included in a connection from a root note node to a leaf node of a tree. On the context hierarchical tree CHT, if two context nodes C_i and C_j belong to the same path, then C_i and C_j are semantically related.

Definition 2. (Semantic Gap) On the CHT, if C_i and C_j are semantically associated and $i \neq j$, there is a semantic gap between C_i and C_j.

Definition 3. (Factor SG measuring Semantic Gap) The SG is defined as $\mathrm{SG}(\mathrm{Ci}, \mathrm{Cj}) = \frac{2^{E(Ci)}}{2^{E(Cj)}}$, where the Ci and Cj are semantically related on the CHT and Ci is an ancestor of Cj. SG is always greater than or equal to 1 (SG≥1)

Example 1. In Figure 2, two context type, surgery and RoomGrp3 is semantically related with each other on the CHT. Assuming that in surgery, there are twenty impatient rooms, and that in RoomGrp3, there are five impatient rooms. The semantic difference between surgery and RoomGrp3 is measured using SG factor.

$$\mathrm{E(Surgery)} = -\sum_{x \in Surgery} P(x)\ \log_2 P(x) = 20 \times \frac{\log_2 20}{20} = \log_2 20, \qquad 2^{E(Surgery)} = 20$$

$\mathrm{E(RoomGrp3)} = -\sum_{x \in RoomGrp3} P(x)\ \log_2 P(x) = 5 \times \frac{\log_2 5}{5} = \log_2 5, \quad 2^{E(RoomGrp3)} = 5$

$\mathrm{SD(Surgery, RoomGrp3)} = \frac{2^{E(Surgery)}}{2^{E(Surgery)}} = \frac{20}{5} = 4.$

3.3 Semantic Context-aware Access Control Model

In the context-aware access control system, the authority of S, O, and P, which are the basic elements of the access control model, is checked at the time of access request. When this check passes, the context conditions are checked as a constraint conditions. In other words, if the access query is < S', O', P', C' >, the context aware access control system first performs a basic access check to determine if < S, O, P > matched with < S', O', P' > is specified in the security policy. If there is a syntactically correct policy rule, the context constraint is checked again, otherwise the access is denied.

In the existing model, the context constraint check evaluates whether the context specified in the policy is matched with the context accompanied by the query, and if both context are syntactically identical, the final access is granted. However, even though the two contexts are semantically related with, the existing model does not take this into consideration. To solve this problem, we propose a method to create a hierarchical structure for context and to infer the relation between context concepts in hierarchical structure.

The following are definitions for the proposed method. We define the relationship between ancestor, descendants, and lineage on the tree of the context hierarchy tree.

Definition 4. (Context Hierarchy Tree) A context is information about all environments, subjects, objects and conditions related to interaction between an application system and a user. The context set C is organized into a tree-like hierarchy. This is called the context hierarchy tree CHT. Each node on the CHT represents context and the edge represents the order of the upper and lower concepts between the two contexts. If two node C_i and C_j are in the CHT and there is a downward path from C_i to C_j, then C_i is an ancestor of C_j, or C_j is a descendant of C_i.

Definition 5. (Ancestry, Descendants, Lineage) When CS is the set of contexts on the CHT, For any context set C $\subseteq$ CS, ANC(C) is a set of the ancestor nodes of each node belonging to C, and C itself is included. DSC(C) is a set of descendant nodes of each node belonging to C, and C itself is included. LNG(C) is a set of all the ancestors and descendants of each node belonging to C. That is, LNG(C) = ANC(C) $\cup$ DSC(C).

It can be seen that access requests should be allowed when the context specified in the security policy explicitly or implicitly infer the context accompanied by the access request query. That is, the policy context should include not only context explicitly included by the security policy but also context implicitly implied by the context hierarchy. In defining the policy context, one consideration is that it should support the prohibition rules for flexible policy description and easy policy management. However, if the prohibition rules are allowed, there will be a policy conflict with the rules. The proposed method defines a Negative Policy-specific Context (NPC) and a Positive Policy-specific Context (PPC) so that the prohibition rule and the permission rule can be supported for the policy context. In case of a policy conflict between the prohibition and the permission, it is solved by applying the prohibition-priority policy. In other words, when there is a policy conflict, the NPC overrides the PPC.

In the hierarchical tree, the parent node contains the concept of a child node. Therefore, if a permission rule is applied to a specific node, the implication is performed downward. The implication of applying the

permission rule is a simple meaning, but the implication when applying the prohibition rule has two meanings. First, if the access is prohibited in the context of a super-concept, such as when applying the permission rule, it means that the access should also be prohibited in the sub-concepts included in this concept. Second, applying the contrapositive proposition in the formal logic, if access is prohibited in the context of sub-concepts, it should also be prohibited in the context of super-concepts. Therefore, when applying the prohibition rule to a concept in a hierarchical structure, implications should be made upward and downward, both directions. In this study, we propose a method to enable efficient policy enforcement reflecting the meaning of the context based on the basic concept proposed by Byun [11].

Definition 6. (Query Context, Policy Context, Implicated Policy Context) The context described in the rule of the security policy is called a policy-specific context, PC. The context accompanied by the access request query is called the query-specific context, QC. Let C be a set of contexts on the CHT, the policy context PC is defined as 2-tuple <PPC, NPC>. Here, PPC $\subset$ C is a set of contexts to which the permission rule applies, and NPC $\subseteq$ C is a set of contexts to which the prohibition rule applies. The set of all policy context, including implications, is called the implicit policy context IPC = DSC (PPC) - LNG (NPC).

Example 2. Assuming PC = <BuildingB, SharingOp.Room> in CHT in Figure 3, DSC(PPC) = DSC(Building B) = {Building B, Surgery, RoomGrp3, Orthopedics, SharingOp.Room,Room301, ... , Room305, RoomS01, … , RoomS05, Room105, ... , Room 110}, LNG (NPC) = LNG (Sharing Op. Room) = {Hospital Building, Building B, Surgery, SharingOp.Room, Room105, ... , Room110}, Therefore, IPC = {RoomGrp3, Orthopedics, Room301, ... , Room305, RoomS01, ... , Rooms05}.

At the time of access request, firstly, the authority of <S', O', P'> which are the basic elements of the access control model, is checked. When this check passes, the context conditions are checked as a constraint conditions. The results of the context constraint test are determined by the relationship between the PC and the QC. When QC is included in the IPC, access is allowed. In this case, it is said that the query context complies with the policy context.

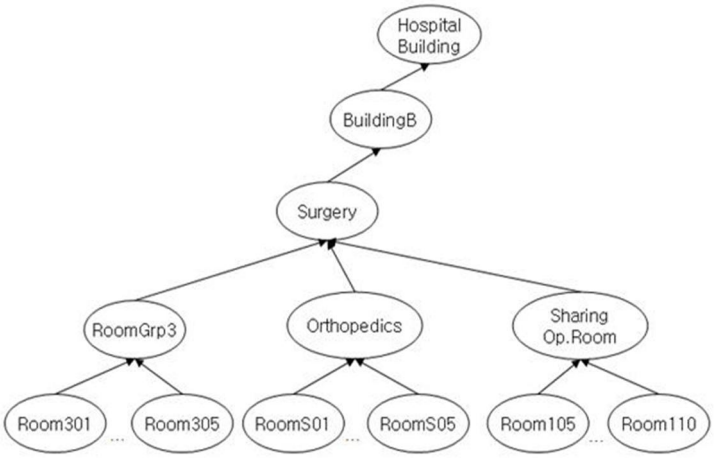

Figure 3. Context Hierarchy Tree with Instance

Definition 7. (Compliance with Policy Context with Implication) On the CHT, the PC is defined as a pair of positive policy context and negative policy context, PC = <PPC, NPC>. If the QC is an element of IPC, QC $\in$ IPC, then the query context QC complies with the implication policy context IPC.

Example 4. When PC = <BuildingB, SharingOp.Room> and QC = Surgery, the QC does not comply with the PC. This is because IPC = {RoomGrp3, Orthopedics, Room301, ... , Room305, RoomS01, ... , Rooms05} and QC is not an element of the IPC set. If the QC was RoomGrp3, QC is an element of IPC, so the QC complies with the PC.

Taking into account the security requirements of confidentiality, the implications that are performed on the hierarchical tree cause the problem of excessive authorization. To solve this problem, it is necessary to limit the degree of implication. The implication of the prohibition rule is not subject to restriction because it restricts the authority itself, but the implication of the permission rule is subject to restriction since it gives authority to all descendant nodes. The simplest way to limit implication on a hierarchical tree is to represent the degree of implication in terms of the level of the tree. In other words, when we want to find the implication of a concept, we define the degree to which the concept can be implied down from the level to which the concept belongs. Although this method is simple, it cannot express the meaning of each node quantitatively and precise control is not possible.

In the proposed method, the implication is limited by using the SG which quantitatively measures the difference between the contexts. Based on the SG, system defined threshold is set and permits implications only within the threshold range, so that authorization is quantitatively limited. Figure 4 shows an algorithm for obtaining a Limited Descendent LDSC, which is a set of nodes satisfying the restricted implication among descendant nodes.

Algorithm (Limited Descendent)

```
LDSC (C, CHT, TH)
IN:     a set C of nodes in CHT
        Context hierarchy tree CHT
        System defined threshold TH
OUT:    a set LDSC of nodes in CHT
{
        LDSC = {} ;
        FOR each c ∈ C {
                i = the level of node c on CHT ;

                // Ni , located in the same path of c, is a node at level i on CHT
                WHILE ( SG(c, Ni) < TH ) {
                        LDSC = LDSC ∪ Ni ;
                        i = i + 1 ;
                }
        }
        RETURN ( LDSC ) ;
}
```

Figure 4. Algorithm for generation of LDSC

Definition 8. (Limited Implication Policy Context) When the restriction is applied to the implication policy context, the limited implication policy context is defined as LIPC = LDSC(PPC) - LNG(NPC).

Definition 9. (Compliance with Policy Context with Restricted Implication) On the CHT, When PC = <PPC, NPC> and the limited implication threshold is set, if QC is an element of LIPC, QC ∈ LIPC, then the QC complies with the PC.

```
Algorithm (Semantic Context Constraints Evaluation)
SCCE (QC, PC)
IN:     query-specific context QC
        policy-specific context PC = <PPC, NPC>
OUT:    PERMIT / DENY
{
        IF ( QC ∈ ( IPC = LDSC(PPC) – LNG(NPC) ) {
                RETURN (PERMIT) ;
        }
        ELSE {
                RETURN (DENY) ;
        }
}
```

Figure 5. Algorithm SCCE for Evaluation of Semantic Context Constraints

```
Algorithm (Semantic Policy Enforcement)
SPE (access query)
IN:     access query < S', O', P', C' >
OUT:    PERMIT / DENY
{
        IF( ! basic access evaluation against <S, O, P> ∈ Security Policy Set ) {
                RETURN (DENY);
        }
        ELSE IF( ! SCCE evaluation against <S, O, C> ∈ Security Policy Set ) {
                RETURN (DENY);
        }
        ELSE {
                RETURN (PERMIT);
        }
}
```

Figure 6. Algorithm for Semantic Enforcement of Access Control Policy

The final algorithm for policy enforcement in access control considering semantic context is shown in Figure 6. First, the basic access control function is executed to check whether the <S ', O', P '> accompanying the query matches the security policy <S, O, P>. If the check passes, the second step is to perform SCCE, which is a semantic constraint evaluation algorithm, showed in Figure 5, to check whether QC = <C '> complies with PC = <C> within the implication threshold range. Eventually, access control policy enforcement only allows access to queries that pass both of these checks.

4. Conclusions

While new information technology advances have made information access and acquisition methods much more diverse and easier, there are side effects that allow illegal access using diverse and high-performance tools. In this paper, we propose a semantic context-aware access control method to deal with these threats. In this method, semantic policy enforcement is enabled based on the conceptual meaning of the context even if the syntax of the context constraint specified by the security policy does not match with the syntax of the context constraint accompanied by the access request query.

The context hierarchy tree was used to formalize the implication relationship between contextual concepts. We define the SG argument to quantitatively measure the semantic difference between contextual concepts in order to prevent excessive authorization that can be caused by semantic implications. The system defined threshold is set based on the SG so that the authorization by implication is made only within a limited range. The proposed method overcomes the semantic difference between the policy syntax and the query syntax, thereby maximizing the convenience and efficiency in the management and operation of the security policy and the security system.

References

[1] Weiser, M., "Hot Topics: Ubiquitous Computing", IEEE Computer, 1993.
[2] Kumar, N., Chafle, G., "Context Sensitivity in Role-based Access Control", Operating Systems Review, Vol. 36, No. 3, IBM Journal, 2002.
[3] Wang, X.H., Xhang, D.Q., Gu, T., and Pung, H.K., "Ontology Based Context Modeling and Reasoning using OWL", in PerCom2004 Annual Conference on Pervasive computing and Communications Workshop, 2004.
[4] Rastogi et al, "Access Control over Uncertain Data", PVLDB '08, 2008.
[5] P. Balbiani, "Acces control with uncertain surveillance", International Conference on Web Intelligence, 2005.
[6] Dalvi et al, "Efficient query evaluation on probabilistic databases", VLDB J, 2007.
[7] Sandhu, R., Ferraiolo, D., and Kuhm, R., "The NIST Model for Role-Based Access Control: Towards A Unified Standard", in Proceedings of the fifth ACM workshop on Role-based access control, 2000.
[8] Ranganathan, R, Campbell R.H., "An Infrastructure for context-awareness based on first-order logic", Personal and Ubiquitous Computing, Vol. 7, Issue 6, 2003.
[9] R. Sandhu, P. Samarati., "Access control: principles and practice", IEEE Communication Magazine, vol. 32, 1994.
[10] Qin, L., Atluri, V., "Concept-level Access Control for the Semantic Web", in ACM Workshop on XML Security, 2003.
[11] Byun, J., Bertino, E., Li, N., "Purpose-based Access Control of Complex Data for Privacy Protection", SACMAT, pp102-110, 2005.