A Defeasible Description Logic Based Semantic Security Policy Conflict Detection Approach

Luokai Hu[1,2] Changchun Qiu[1] and Yuda Shi[1]

[1] *Computer School, Hubei University of Education, Wuhan, China, 430205*
[2] *State Key Lab of Software Engineering, Wuhan University, Wuhan, China, 430072*
luokaihu@gmail.com

Abstract

Policy has been widely used in field information security. Semantic policy has being widespread concerned by the academic and industry world due to its powerful ability of expression. While policy conflicts have become a key issue constraining its application. In this paper, first a Semantic Security Policy Language (SSPL) is introduced. And we propose a Defeasible Description Logic (DDL) based conflict detection and solution approach for SSPL. Finally, the experiment shows that, the method which combines with non-monotonic reasoning and semantic reasoning improves the rate of conflict detection.

Keywords: *Defeasible Description Logic, Argumentation System, Semantic Security Policy, Conflict Analysis*

1. Introduction

With the development and maturity of the Semantic Web, description and expression of policy in the semantic level has become an important research direction. Because of its own powerful ability of description, the problem of conflict detection and resolution for semantic policy are more complex. In this paper, Defeasible Description Logic (DDL)[1] Based semantic policy conflict analysis approach provides a new way of thinking to the study of semantic security policy.

2. Semantic Security Policy Language (SSPL)

The Semantic Security Policy Language (SSPL) is designed on the basis of languages such as XACML[2] and WSPL[3], in order to meet a variety of network and database security needs. SSPL is composed by three parts that is rule, policy and policy set. The syntax of SSPL rule and its DDL mapping is shown in table 1. Each ***SSPL rule*** has several consequents and may have an antecedent or not. $\rightarrow$, $\Rightarrow$ and $\rightsquigarrow$ denote strict, defeasible and defeat rule respectively.

Table 1. DDL Semantics of SSPL Rule

No.	SSPL Syntax	DDL mapping π
1	Rule::= [Pre] $\hookrightarrow$ Con	for each Apply in Con: π(Pre) $\hookrightarrow$ π(Apply)
2	$\hookrightarrow$::= $\rightarrow$ | $\Rightarrow$ | $\rightsquigarrow$	
3	Con::= Apply | Con $\cap$ Con	
4	Apply::= (attri-id attr-val afcn)	$\pi(\nu$(attr-id), ν(attr-val), afcn)
5	ν(attr-id), ν(attr-val), afcn	Table 2.

Table 2. DDL semantics of SSPL Arithmetic Operators

No.	afcn	DDL mapping π
1	type-assign	ν(attr-id)≡ν(attr-value) or ν(attr-id) ≅ ν(attr-value)
2	type-great-than	ν(attr-id)≡{ > ν(attr-value)}
3	type-great-than-or-equal	ν(attr-id)≡{ ⩾ ν(attr-value)}
4	……	……

Due to space constraints, some arithmetic operators are omitted in table 2.

SSPL Policy is composed by several SSPL rules. The policy itself has no antecedent, so the consequent of a policy is the semantic overlay by the consequents of all the rules in the policy. For a policy P with n rules, the consequent of P is denoted Con(P). The strict, defeasible and defeat rules of P are denoted P.Rs, P.Rd and P.Rdft respectively. The rule set P.R of P can be formalized as:

P.R=P.Rs ∪ P.Rd ∪ P.Rdft where, if m=#(P.Rs ∪ P.Rd) then m⩽n.

And, P.Rsd=P.Rs ∪ P.Rd, that is the rule set of non-defeat rules.

The consequent of policy P can be formalized as:

Con(P)=Con(P.r1) ⊕ Con(P.r2) ⊕ ... ⊕ Con(P.rm), P.ri∈(P.Rs ∪ P.Rd) 1⩽i⩽m.

3. DDL Based Conflict Analysis for SSPL

In this section, we use the argumentation system to analyze the SSPL policy. A typical argumentation system contains the following basic elements: an underlying logic language, the argument process and states arguments. The argumentation system of this paper uses the DDL as logical language. DDL is a non-monotonic logic. Because it cannot use monotonous argumentation, proof tree is used to record argumentation states.

Definition 1 (Literal). Literal of SSPL is defined as the description logic mapping of expression in conditions or consequents. That is, for a rule r, literal $\alpha \in \pi$(Pre(r)) or $\alpha \in \pi$(Con(r)).

Definition 2 (Proof Tree). The proof tree of Literal β is defined as a finite tree with the root β. All nodes h of the tree satisfy:

(1) If the node h has child nodes c1, c2, ..., cn, all these child nodes are the conjunctive components of rule set R[h]'s antecedent or satisfiable knowledge of knowledge base;

(2) If the node h is not the root node, then all the rules of rule set R [h] are strict rules or defeasible rules.

In addition, if all the rules in the proof tree are strict rules, that tree is named Strict Proof Tree; If the rules that are used to prove root node in the proof tree is strict rules or defeasible rules, then the tree is called Supportive Proof Tree.

Definition 3 (Single Argument). The single argument of a literal β is a proof tree with root β.

Definition 4 (Supportive Argument). When all the proof trees of literal β in the argument A are supportive proof tree, the argument A of literal β is called supportive argument, denoted by $A=+\Sigma\beta$.

Definition 5 (Argument). The argument of a literal β is set of all proof trees of β. In addition, if a proof tree of argument A used rule r, we should denote by $r\in A$.

Definition 6 (Argument Set). The argument set of a policy P is the set of all the argument of literal in P's conclusion, denoted by $P.\mathbb{A}$.

Definition 7 (Sub-Argument). The sub-argument of an argument A is the sub tree of A's proof tree.

Definition 8 (Leaf Set). Let t is a proof tree, the leaf set of literal β is the set of all the leaf node of t, denoted by Leaf(β).

Definition 9 (Strict Argument and Defeasible Argument). If all the proof tree of an argument A is strict proof tree, then A is called Strict Argument. While A is not a strict argument, A must be a defeasible Argument.

Definition 10 (Conflicting Literal Pair). Let (α, β) is a literal pair, if $\forall i \in \Delta T$ if $i \in \alpha\mathcal{I}$ then $i \notin \beta\mathcal{I}$, then we call ($\alpha$, β) conflicting literal pair, denoted as Conflict(α, β) or $\alpha=\|\beta$ for short.

Definition 11.(Ambiguous Argument Pair). Let A and B are the supportive argument respectively for literal βa and βb. Not consider the priority, when the literal βa and βb are conflicting literal on each other, the argument A and B are mutually ambiguous argument pair, denoted as Ambiguity (A and B).

Definition 12. (Ambiguous Argument and Ambiguous Argument Set). Let A is a supportive argument for literal βa. In the case without considering the priority, when the intersect set of description mapping π(Leaf(A)) of Leaf(A) and βa can not be satisfied by ontology interpretation domain, we call A ambiguous argument, denoted as Ambiguity(A); If argument set S contains ambiguous argument, we call S ambiguous set, denoted as Ambiguity(S).

Definition 13. (Defeasible). Let literal βa and βb satisfy $\beta a = \|\beta b$. Let A, B are two arguments, where, B is defeasible argument. If $\exists rA[\beta a]\in A$, $\exists rB[\beta b]\in B$, such that $rA \succ rB$, then we call A can defeat argument B, denoted as Defeasible(A, B). Otherwise we call A can not defeat argument B. Similarly, if $A\in$ argument set S, then we call S can defeat B, denoted as Defeasible(S, B).

Definition 14. (Conflicting Argument). If argument A and B is Ambiguity pair, and ¬Defeasible(A, B) and ¬Defeasible(B, A), and π(Leaf(A)) $\sqcap$ π(Leaf(B)) can be satisfied by ontology interpretation domain, then we call A and B is conflicting argument of each other, denoted as Conflict(A, B).

Definition 15. (Inconsistent Argument). If A is ambiguity argument and for all argument B there not exists ¬Defeasible(B, A), then we call A an inconsistent argument, denoted Inconsistency(A).

Definition 16. (Conflicting Policy). Let P be a policy, if $\exists$j such that it is unable to reason after P.D(1..j). And $\exists$R$\subset$P.Rsd such that policy (P$-$R).D is reasonable. Then we call P a conflicting policy, denoted Conflict(P).

Theorem 1. Let P be a policy, if $\exists$A, B$\in$P.$\mathbb{A}$ such that Confliction(A, B) is satisfied or $\exists$A$\in$P.$\mathbb{A}$ such that Inconsistency(A) is satisfied, then Confliction(P) is satisfied.

Proof: first, we prove Confliction(A, B) $\Rightarrow$ Confliction(P).

According to Def. 14, if $\exists$A, B$\in$P.$\mathbb{A}$ such that Conflict(A, B) is satisfied, then ¬Defeasible(A, B) and ¬Defeasible(B, A) and $\mathbb{K}$ $\vDash$ π(Leaf(A)) $\sqcap$ π(Leaf(B)) and Ambiguity(A, B) ;

According to Def. 11, if $\exists$A, B$\in$P.$\mathbb{A}$ such that Ambiguity(A, B) is satisfied, then βa=$\|\beta$b, where, βa and βb is the literal of argument A and B respectively;

According to Def. 10, if βa=$\|\beta$b, then $\mathbb{K}$ $\nvDash$ βa $\sqcap$ βb. So when reasoning to argument A or B, it is unreasonable. If all rules of A or B were deleted, then the reasoning process can be continued to satisfy Def.16. So Confliction(A, B) $\Rightarrow$ Confliction(P) is proved.

Secondly, we prove Inconsistency(A) $\Rightarrow$ Confliction(P).

According to Def.15, if $\exists$A$\in$P.$\mathbb{A}$ such that Inconsistency(A) is satisfied, then Ambiguity(A) is satisfied and $\forall$B$\in$P.$\mathbb{A}$, there exits ¬Defeasible(B, A);

According to Def.12, if $\exists$A$\in$P.$\mathbb{A}$ such that Ambiguity(A), then A=+$\Sigma\beta$a and $\mathbb{K}$ $\nvDash$ π(Leaf(A)) $\sqcap$ βa, where βa is the literal of argument A;

According to Def.4, if $\exists$A$\in$P.$\mathbb{A}$ such that A=+$\Sigma\beta$a, then (1) $\mathbb{K}$ $\vDash$ βa or (2) $\exists$r $\in$ P.Rsd[βa] $\forall\alpha \in \pi$(Pre(r)): +$\Sigma\alpha \in$ D(1...i) or $\mathbb{K}$ $\vDash$ α;

When condition (1) is satisfied, there exists $\mathbb{K} \vDash \beta a$ and $\mathbb{K} \nvDash \pi(Leaf(A)) \sqcap \beta a$, then $\mathbb{K} \nvDash \pi(Leaf(A))$ and $+\Sigma$ is not provable. $\exists r \in A$ such that $r.Pre \in Leaf(A)$. While after all these rules r are deleted, $+\Sigma$ is provable. Def.15 is satisfied, so Confliction(P) is satisfied;

When condition (2) is satisfied, $\exists r \in P.Rsd[\beta a]$ $\forall \alpha \in \pi(Pre(r))$: $+\Sigma\alpha \in D(1...i)$ or $\mathbb{K} \vDash \alpha$, then $+\Sigma$ is not provable. While after all these rules r are deleted, $+\Sigma$ is provable. Def.16 is satisfied, so Confliction(P) is satisfied;

In summary, the theorem is proved.

The conflict of policy set between policy P1 and policy P2 should satisfy the following three conditions:

(1) If {Con(P1, P2)} is the intersection of Policy P1's consequent set {Con(P1)} and Policy P1's consequents set {Con (P1, P2)}, then {Con (P1, P2,)} $\neq \varnothing$;

(2) If A[{Con (of P1, P2),}, P1] is the argumentation set in the policy P1 with roots of all literal of {Con (P1, P2)}, and A[{Con (of P1, P2),}, P2] is the argumentation set in policy P2 with roots of all literal of {Con (P1, P2)}, then π (Leaf(A[{Con(P1, P2)}, P1,])) $\sqcap$ π(Leaf(A[{Con(P1, P2)}, P2])) can be satisfied by ontology interpretation domain.

(3) The priority of policy P1 is equal to the priority of policy P2

4. Implementation and Experiments

In order to support conflict analysis of security policy, architecture of the prototype is designed. The prototype system which is implemented to illustrate the capability of DDL based semantic security policy analysis approach includes two supporting services: a policy service and a knowledge service.

Policy Serivce. The main function of policy service is policy decision. That is, it will renders a decision after receiving a request. It also needs to store policies and solve all conflict before making decision.

Knowledge Service. In our prototype system, the knowledge service manages ontology and rules. The concepts and relations of ontology and business rules enable us to model domain knowledge. The knowledge service also contains a reasoner

In addition, some policy and knowledge management tools are also needed in this system. Based on this prototype, the experiment results for inner policy and between policies are given in table 3 and 4.

Table 3. Inner Policy Conflicts Detection

	Designed Policy			Random Policy		
Number of Concepts in ontology(Approx.)	1000	10000	50000	1000	10000	50000
Number of Policy	10			50		
Number of conflicts	10			30		
Number of conflicts reported by DDL approach	10	10	9	25	19	16
Number of conflicts reported by DL approach	5	5	4	10	10	8

Table 4. Between Policies Conflict Detection

	Designed Policy Set			Random Policy Set		
Number of Concepts in ontology(Approx.)	1000	10000	10000	1000	10000	50000
Number of Policy Set	10			50		
Number of conflicts	10			30		
Number of conflicts reported by DDL approach	9	9	8	24	22	19
Number of conflicts reported by DL approach	5	5	5	9	9	8

5. Related Work

Some exiting work in detecting conflicts of security policy is related to modality and syntax conflicts. A. Bandara from Imperial College London proposed an event calculus based access control policy analysis methods, this method is mainly used for conflict detection between the policies and optimization in 2003[4]. He also proposed a first-order logic based formal language, which increased the clear description of time in 2007. On the basis of this language he given the formal description and conflict detection method of authorization and obligation policies[5].

There is also some representative research work on semantic conflict detection. The KAoS adopt policy priority for the conflict resolution of semantic policy[6]. Rei combined the method of meta-policy and priority for conflict detection and resolution of semantic policy[7]. In 2005, W. Nejdl from the University of Hannover proposed two types of policy that is Mandatory, Policy and default policy and related semantic policy analysis method[8]. K. Verlaenen proposed a rule based semantic policy analysis approach[9]. In 2006 and 2007, V. Kolovski were put forward non-monotonic logic based policy analysis methods which used default logic and defeasible logic for non-monotonic reasoning[10].

Different from their work, defeasible description logic is used for conflict analysis of semantic security policy. Non-monotonic reasoning and semantic reasoning are both used for policy analysis in this paper.

6. Conclusions

In this paper, we present a conflict analysis method based on the semantics of DDL and its argumentation system for security policy. Proved by the experiment, the DDL is suitable for the formalization modeling and conflict analysis for the semantics security policy. The rate of the conflict detection has improved. This study enriches the applied research semantic Web technology in the field of application security. This paper has some theoretical significance in the study of conflict detection and resolution based on the non-monotonic logic. Our research result has a great value for the field of access control in the Internet environment, information system integration, security of e-commerce and e-government application.

Acknowledgements

This work was supported by the Program for Innovative Research Team in Hubei University of Education, the Hubei Science and Technology Research Program Projects for Outstanding Young Talent Grant (No.Q20113001), the Key Scientific Research Project of Hubei Provincial Department of Education (No. D20103004).

References

[1] G. Governotori, "Defeasible description logic", In: Antoniou, G., Boley, H. (eds.) Rules and Rule Markup Languages for the Semantic Web, LNCS vol.3323, pp. 98-112. Springer, Heidelberg **(2004)**.
[2] OASIS, Extensible Access Control Markup Language (XACML) Version 2.0 **(2005)**, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.
[3] A. Anderson, "Predicates for Boolean Web Service Policy Languages. Proceedings of the 14th International World Wide Web Conference, **(2005)** Chiba, Japan.
[5] A. Bandara, E.Lupu and A. Russo, "Using Event Calculus to Formalize Policy Specification and Analysis", Proceedings of the 4th International Workshop on Policies for Distributed Systems and Networks (POLICY 03), **(2003)** Lake Como, Italy.
[6] A. Bandara, S. Calo, J. Lobo, E. Lupu and A. Russo, "Toward a Formal Characterization of Policy Specification & Analysis. Proceedings of Annual Conference of ITA, **(2007)** Maryland, US.
[7] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni and J. Lott, "KAOS Policy and Domain Services: Toward a Description-Logic Approach to Policy Representation, Deconfliction, and Enforcement", Proceedings of the 4th International Workshop on Policies for Distributed Systems and Networks, **(2003)** Lake Como, Italy.
[8] L. Kagal, T. Berners-Lee, D. Connolly and D. Weitzner, "Using Semantic WEB Technologies for Policy Management on the WEB", Proceedings of 21st National Conference on Artificial Intelligent, **(2006)** Boston, MA, US.
[9] W. Nejdl, D. Olmedilla, M. Winslett and C. Zhang, "Ontology-Based Policy Specification and Management", Proceedings of Second European Sematic Web Conference, **(2005)** Crete, Greece.
[10] K. Verlaenen, B. De Win and W. Joosen, "Policy analysis using a hybrid semantic reasoning engine", Proceedings of Eighth IEEE International Workshop on Policies for Distributed Systems and Networks, **(2007)** Bologna, Italy.
[11] V. Kolovski, J. Hendler and B. Parsia, "Analyzing WEB Access Control Policies", Proceedings of the 16th International World Wide Web Conference, **(2007)** Banff, AB, Canada.

Authors

Hu Luokai

Hu Luokai received the MS degree from Wuhan University, China, in 2006. He is currently pursuing the PhD degree in State Key Lab of Software Engineering at Wuhan University under the guidance of Dr. Ying Shi. His current research interests include security of service oriented architecture and semantic Web.