A defeasible policy based access control approach for semantic web services composition

Luokai Hu[1,2*], Chao Liang[1], Ying Lu[2], Yan Zeng[2]

[1]*Lenovo Mobile Communication Technology Co. Ltd., Xiamen China*

[2]*Hubei Co-innovation Center of IT Service for Basic Education, Hubei University of Education, Wuhan China*

Abstract

Semantic Web services have brought great convenience to service-oriented software development. However, during the semantic Web service composition because the component Web services and licensing issues often require repeated dynamic binding, which greatly affect the efficiency of the service execution. To address this problem, we propose a defeasible policy based a ccess control approach for semantic Web service composition. Firstly, before the semantic service is bound to a component of Web services, static analysis can avoid unnecessary service binding in the semantic Web service composition and execution time. Then we give the access control enforcement process in composition and execution time. Finally, the feasibility of this method has been verified through experiments. Our approach can increase the efficiency and successful rate of semantic Web service compositio n.

Keywords: defeasible logic, access control, semantic web services composition

1 Introduction

In recent years, research in semantic Web services mainly focus on the topics of discovery, composition and execution of Web services. However, with the increasing and deepening study in semantic Web services, its security gradually became one of the key problems. If not treated appropriately, potential security risk can hamper the large scale applications of semantic Web services. Particularly, among the security issues of semantic Web service, access control is a basic and core problem, which has been attracting wide attention from academia and industry.

The autonomy member providers of composite Web service are independent entities. They have full control to their own resources [1]. Policy based access control approach for Web service enables the separation of description and implementation mechanism and also makes cross-domain integration of distributed entities secure access rules possible. Composite Web services are built on an open dynamic environment. There is no direct trust relationship among component services, the interdependence between the component services in a coordinated manner. Even if the policy of each component service has correct provision, it may still lead to conflicts among diversification regional policies [2]. It will affect quality and robustness of service composition and reduce user satisfaction. Effective realization of consistent dynamic coupling of component service access control policy for composite Web services in a multi-domain collaborative environment has been a hot issue of the research in Web service security [3].

One of the advantages of using semantic Web service is making service composition more convenient. During the semantic Web service composition, developers do not need to know every composition paths and semantic Web service will automatically bind a component Web service. However, compared to the access control for Web service composition, access control for composition of semantic Web service is more complex. Because, there are often more than one component Web services which were found in the service discovery process. In the entire service composition path, the combination of component Web services which are most likely to have the right able to be executed is an important factor affecting the efficiency of the semantic Web service composition. In this paper, before the semantic Web service binding, through the static analysis of the composition path, the path which cannot be able to be executed will be removed before the execution. It avoids repeatedly meaningless service dynamic binding during the execution period of the semantic Web service composition because of without authority.

2 Background

2.1 DEFEASIBLE LOGIC

Defeasible logic is a type of non-monotonic logic. It is proposed by Donald Nute in 1987 which is used for formalizing the defeasible reasoning [4]. The vast majority of non-monotonic logics are not linear complexity, but defeasible logic is an exception [5]. Since defeasible logic has good computational complexity and easy to implement, it has aroused wide spread concern in recent years. For example, if we know that XiaoHu is an associate Professor, it is reasonable to assume that he can teach class. However, if we later find out that XiaoHu is sick we may want to retract our previous assumption about Sam's

[*] ***Corresponding author's*** e-mail: luokaihu@gmail.com

ability to teach class.

A defeasible logic is composed of three parts: facts, rules, and superiority relationship. Facts are indisputable statements such as AssociateProfessor(XiaoHu), which states that "XiaoHu is an associate professor." Defeasible logic has three types of rules: strict rules, defeasible rules and defeater rules. Strict rules are rules in the classical sense, such as "Associate Professors are teachers" Formally: AssociatePorfessor(x)→Teacher(x).
Defeasible Rules are used to draw conclusions that may later be retracted. A defeasible rule "Associate professors typically teach class" can be formally written as: AssociatePorfessor(x)⇒teachClass(x).
Defeater Rules provide contrary evidence to defeasible rules. A defeater rules "if an associate professor is sick, it will not be able to teach class." is formally written as: sick(x)⇝¬teachClass(x).

It is important to note that defeater rules cannot be used to draw conclusions, they simply prevent conclusions. An associate professor being sick is not sufficient evidence to prove that he/she cannot teach class, however we do not want to ignore to the conclusion that it indeed can teach class.

2.2 SACPL

The Semantic Access Control Policy Language (SACPL)[6,7] in which the necessary syntax elements and appropriate semantic annotation are added is designed on the basis of language such as XACML [8], in order to meet a variety of network and database security needs. SACPL is composed by three parts that is rule, policy and policy set. In the distributed computing environment, the access control method has changed from the centralized management into a distributed management approach. There has been policy markup language, such as XACML, to support description and management of distributed policies. In the semantic Web service composition, the issue of interoperability among policies is more important than ever before. Specifically, subject, object, action and attribute variables as the basic semantic element are annotated by ontology. The more detail of SACPL can refer to [6,7].

3 Static analysis of semantic web service composition path

In a semantic Web service composition path, if a semantic Web service, at current time, only one component Web service can be discovered, that component Web service is called key service, denoted as KCWS. If more than one component web service can be discovered, the set of those component Web services is called non-key service set denoted as NCWSSet. Even if the access control policy of the key service is not compatible with other policies of services in the composition path, it cannot be replaced. If the access control policy of non-key service and other service policies in the composition path are not compatible, it can be replaced. Moreover, during the static analysis of semantic Web service composition path, the selection of the candidate component web service from NCWSSet follows the order of QoS priority.

In addition, before the static analysis of semantic Web service composition path, the composition service (engine) must have trust relationship with component Web service. Otherwise, the component Web service cannot disclose its access control policy to the composition service. In that case, the service discovery procedure will not bind this component service to its semantic service. Research on the relationship of trust between services is beyond the scope of this study, we suppose that the trust relationship has established, otherwise, it is deemed not found the candidate component service. Besides, static analysis of semantic Web service composition path cannot be suited for all situations. It only can be used for the static path analysis. The definition of static path is as follows. Before the definition of static path, we give the definition of static and dynamic policy first.

Definition 3.1 (**Static Policy** and **Dynamic Policy**). Static Policy is a policy which has only static variables. Static variable is a variable which is not changed by any operations in the whole Web service composition path. That is $\forall v \in P.V$, v is static variable. Otherwise, Dynamic Policy is a policy which has at least one dynamic variable. Dynamic variable is a variable which may be changed by execution operations. That is $\exists v \in P.V$, such that v is dynamic variable. Obviously, any policy is either static or dynamic policy.

Definition 3.2 (**Static Path** and **Dynamic Path**). Static Path is a path which has only static policies or has dynamic policies such that all execution operations are "behind" the operations belongs to the dynamic policies. The term "behind" means front operations will never pass parameters to the rear operations in the Web service composition path. That is $\forall p \in \text{Path}.ACPS$, p is static policy or if $\exists p \in \text{Path}.ACPS$ such that p is dynamic policy, then $\forall eop \in \text{SP.EOP}$, $eop.out \cap p.op.in=\varnothing$ in the Web service composition path. Otherwise, if a path is not a static path, it must be a dynamic path.

From the perspective of whether the static analysis of composition path can be done, a composition path can be divided into static and dynamic path. On the other hand, from the view of whether the path can run through all the access control policy, the semantic Web service composition path *SWSC* has three types: inaccessible path, executable path and possible path. Inaccessible composition path is a static path in which there is confliction among access control policies. Executable composition path is the static path in which there is not confliction among access control policies. Possible composition path is a dynamic path in which there is not confliction among all static access control policies. This article focuses on the static analysis of executable composition path (shown as Algorithm 1). Because if a static path is not an executable composition path, it is must be an inaccessible composition path. The static analysis of possible composition path is equals to the static analysis of all the static access control policies.

Algorithm 1: Static Analysis of Executable Composition Path

```
PCPSet getExecutablePath(SWSCP swscp)// This algorithm will return all the executable path of a semantic Web service composition path.
Input: Semantic Web Service Composition Path(SWSCP) swscp// The path is a static path.
Output: Web Service Composition Path Set(PCPSet) pcps //possible Web service composition path set
Parameters:
int n; //the number of Semantic Web services in the independent composition path
Component Web Service(CWS) kcws; //key component Web services of semantic service path
Component Web Service Set(CWSSet) kcwss, ncwsset;
Linked List of Component Web Service Sets(CWSSetList) ncwssl;
PolicySet ps;
Linked List of PolicySet psl;

// All candidate component Web services have trust relationship with composition service.
(1) Run the service discovery procedure to find all candidate component Web services;
(2) Add m key component Web services into key component Web service set kcwss;
    kcwss = {$kcws_i$|$1 \leqslant i \leqslant m$}
(3) Add non-key component Web services into each non-key component Web service set (NCWSSet) respectively. Add k non-key component Web service sets into ncwssl;
    ncwssl = {$ncwsset_i$|$1 \leqslant i \leqslant k$} such that n== m+k
(4) Add all rules of composition service as a policy into ps;
(5) Add all rules of $kcws_i$ in kcwss as a policy into ps;
(6) if(confliction(ps)) return pcps=null;
    //that is convert each policy p in $pst_1$ to a policy set
(7) Make the policy set $pst_1$ of $ncwsset_1$ to a Linked List of PolicySet psltemp;
(8) for (i=2; i<=k; i++)
        psltemp = GetNonConflictPolicySetLink(psltemp, $pst_i$);//$pst_i$ is the policy set of $ncwsset_i$
(9) psltemp = GetNonConflictPolicySetLink(psltemp, ps);
(10) Get each non-conflict policy set ncps from psltemp;
(11) Get each Web service set wss of ncps;
(12) Convert each wss to a Web Service Composition Path wscp;
(13) Add all wscp into pcps;
(14)Add all key component Web services into each executable Web service composition path of pcps;
(15) return pcps;
```

The algorithm 2 will get policy set link without confliction from a policy set.

Algorithm 2: Get Non-Conflict Policy Set Link.

```
PolicySetLink getNonConflictPolicySetLink(PolicySetLink psli, PolicySet ps)
Input: PolicySetLink psli; PolicySet ps; //A policy set link is a linked list of policy sets.
Output: PolicySetLink pslo; //the Policy Set Link without confliction
(1) foreach(psl[i] in psli)
(2)   foreach($p_i$ in ps)
(3)     add $p_i$ into psl[i];
(4)     if(!confliction(psl[i])) add psl[i] into pslo;
(5) return pslo;
```

4 Static analysis of access control policy set

In an access control policy set *PS* for semantic Web service composition, all rules $P_i.r$ of policy P_i ($P_i \in PS$, $1 \leq i \leq |PS|$) can be divided into two kinds of rule: permit and deny which can be denoted as $P_i.r_p$ and $P_i.r_d$ respectively. We can construct a new rule set *R* using defeasible logic, where the consequent $con(P_i.r_p)$ of each $P_i.r_p$ can be rewritten as "PERMIT" while each condition(or precondition) $pre(P_i.r_p)$ remains unchanged. Similarly, the consequent $con(P_i.r_d)$ of each $P_i.r_d$ can be rewritten as "DENY" while each $pre(P_i.r_d)$ remains unchanged.

Because XACML is a typical access control language policy of Web services, the static analysis access control policy set will take XACML as an example. The main idea of this access control policy set static confliction detection algorithm is to first deal with each policy in policy set, to convert them to defeasible logic rules. Secondly, the confliction can be detected for all these defeasible logic rules. Reference [9] described a conflict detection method of defeasible logic rules. Typically, XACML has four types of policy combination algorithms. They are Deny-overrides, Permit-overrides, First-applicable and Only-one-applicable. The mapping of XACML to defeasible logic is discussed in these four cases. In the Deny-overrides algorithm, if any rule evaluates to Deny, then the final decision is also Deny. So if *r* is a deny rule, then *r* is a strict rule. If *r* is a permit rule, then *r* is a defeasible rule. In the Permit-overrides algorithm, if any rule evaluates to Permit, then the final decision is also Permit. So if *r* is a permit rule, then *r* is a strict rule. If *r* is a deny rule, then *r* is a defeasible rule. In the First-applicable algorithm, the effect of the first rule that applies is the decision of the policy. The rules must be evaluated in the order that they are listed. In the Only-one-applicable algorithm, if more than one rule is applicable, return Indeterminate. Otherwise return the access decision of the applicable rule.

Algorithm 3: Static Analysis of Conflict Policy Set.

```
Boolean confliction(PolicySet ps)
Input: PolicySet ps; //all the
Output: Boolean conflict;

(1) foreach(Policy p in ps)
(2)   if(p.combineAlg equals Deny-overrides)
(3)     foreach(Rule r in p)
(4)       if (con(r) equals Deny) construct r as a strict rule;//note all r s.t. con(r) equals Deny as rd
(5)       else construct r as a defeasible rule;//note all r s.t. con(r) equals Permit as rp
(6)       if(pre(rd) $\sqcap$ pre(rp)) $\neq \emptyset$) construct defeat rule: pre(rd) $\sqcap$ pre(rp) $\rightsquigarrow \neg$Permit
(7)   if(p.combineAlg equals Permit-overrides)
(8)     foreach(r in p)
(9)       if (con(r) equals Permit) construct r as a defeasible rule;
(10)      else construct r as a defeasible rule;
(11)      if(pre(rd) $\cap$ pre(rp)) $\neq \emptyset$) construct defeat rule: pre(rd) $\cap$ pre(rp) $\rightsquigarrow \neg$Deny
(12)  if(p.combineAlg equals First-applicable)
(13)      if(con(r) equals Deny) construct $r_1$ as strict rule;
(14)      else construct $r_1$ as defeasible rule;
(15)      for(i=1;i<n;i++)
(16)        if(con($r_{i+1}$) equals Deny) construct strict rule: pre($r_{i+1}$) / $\bigcup$pre($r_i$) $\longrightarrow$ Deny;
(17)        else construct strict rule: pre($r_{i+1}$) / $\bigcup$pre($r_i$) $\Rightarrow$ Permit;
(18)  if(p.combineAlg equals Only-one-applicable)
(19)    if (p.num equals 1)
(19)      if(con($r_i$) equals Deny) construct $r_i$ as a strict rule;
```

```
(20)      else construct $r_i$ as defeasible rule;
(21) Add all the rules above into a Policy po;
(22) conflict = checkConfliction(po);// Calls the functions of Reference [9].
(23) return conflict;
```

5 The access control enforcement in execution time

If a client requires an access to a composite semantic Web service, the composition execution engine needs to evaluate access control policy of composite Semantic Web Services "before execution" according to the valid access attribute and select a Web service composition path to bind corresponding component Web service to each semantic Web service.

And during the execution of the composite Web service, the composition execution engine also needs to dynamically evaluate the request on the operation according to "execution time" attributes. If access is denied, looking for other Web services to rebind is needed until no other candidate Web services or access is successful.

Existing policy description language (such as XACML) can only express access control policies in the syntax layer, in order to meet the needs of access control for semantic Web service, we propose SACPL (Semantic Access Control Policy Language)[6, 7] as a policy description language. Based on XACML, SACPL Language adds semantic annotation and corresponding auxiliary mechanisms. This article uses SACPL as access control policy description language for composition semantic Web services. The realization project of SACPL can be rapidly achieved through the extension of XACML from open source projects (such as Sun's XACML). With the limitation of space, more details about SACPL could refer to [6, 7]. Since SACPL is an extension language of XACML, some terms of XACML is integrated to describe the architecture of access control for semantic Web services composition in Figure 1. The interaction among modules of this architecture could be described in detail.

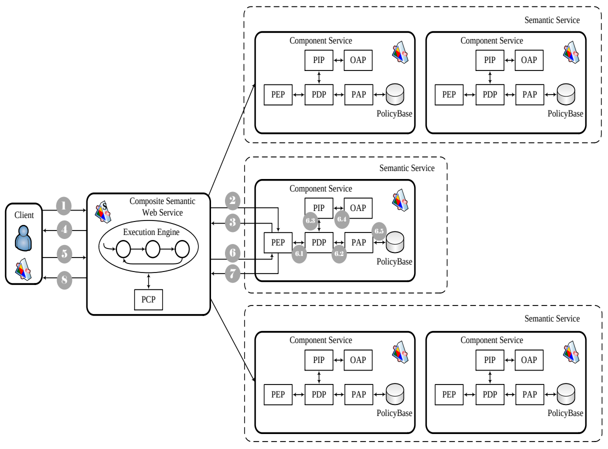

FIGURE 1 Enforcement process of access control for semantic web service composition

1) The Subject *S* wishes to visit a Composite Semantic Web service (object) *CSWS*. *S* sends a SOAP message with a key K_s to *CSWS*. The SOAP message is received by *CSWS* and decided by PDP of *CSWS* using its own access control policy written with SACPL. If the result is Permit, then the SOAP is forwarded to the Web service WS_i which is a Web service in the composite path. Otherwise the visit will be denied.

2) The SOAP message is intercepted by the security agent PEP_i of WS_i.

3) PEP_i issues an inquiry to the *CSWS* to ask about all known attributes of *S*.

4) *CSWS* forwards the inquiry to the authentication center AC_s of *S* to ask about all known attributes of *S*.

5) Authentication center *ACs* first confirms the validity of Key K_s. If the answer is yes, then AC_s sends a request to the Policy Information Point PIP_s to query attribute, otherwise refusing the request. PIP_s queries all the known attributes A_s of Subject *S* from the attribute database. PIP_s sends A_s as input to the semantic reasoning module to derive more valuable attribute information A'_s of subject *S*. PIP_s returns the query results (A_s+A'_s) to the authentication center AC_s. AC_s sends the query results to the context handler CH_s. The query results will be encapsulated into the format of SACPL Request by context handler CH_s and sent to *CSWS*.

6) *CSWS* forwards the query results (A_s+A'_s) to PEP_i. (6.1) PEP_i sends the SACPL Request to the policy decision point PDP_i of WS_i. (6.2) PDP_i parses the SACPL Request and queries the related policy (written in SACPL) with the *Target* token as an index from the policy administration point PAP_i. (6.3) PDP_i sends an attribute query request to policy information point PIP_i. PIP_i converts (A_s+A'_s) to A_i which can be understood by domain of WS_i with the help of domain ontology of WS_i. PIP_i queries all the attributes Aws_i of Web service WS_i to which S wants to access from Attribute Database. (6.4) PIP_i gathers all environment attributes A_e. PIP_i returns all the attribute information (A_i+Aws_i+A_e) to policy decision point PDP_i. (6.5) PDP_i generates an access decision which will be sent to the context handler CH_i. CH_i encapsulates the access control decision into the SACPL Response format and sends it to security agent PEP_i.

7) According to the *result* in SACPL response, security agent PEP_i could perform the corresponding *Permit* or *Deny* action for this visit. If the decision is *Permit*, request operation will run and PEP_i will return the operation result to the *CSWS*. CSWS will call the next semantic Web service in its composition path. Otherwise, PEP_i will return access *Deny* to the *CSWS*. In that case, composition engine will choose another component Web service in the candidate service list until the list is empty. If no component Web service can be access return *Deny* to the *CSWS*.

8) The operation result or the access *Deny* returned to the client *S*.

6 Performance evaluation

To validate the effectiveness and evaluate the performance of the defeasible logic based semantic Web service composition access control approach, we design a set of experiments which is set up in a simulation system to simulate the semantic Web service composition path.

The simulation system includes 300 concrete Web services. We use IBM Rational Application Developer [10] to simulate Web services. Firstly, 12 graduate students designed different access control policy for 240 Web services respectively using XACML. Each Web service access control policy has 3-10 rules. The other 60 Web services do not have access control policy and can be freely accessed. Then, five semantic Web service composition paths were designed, each path containing 5-20 semantic Web services. Thus the generating composition Web services were designed five access control policy by the author using SACPL. Each policy has rules ranging from 5-10. Next, these five semantic Web service composition paths will be executed in both situations with and without the use of static analysis (situation 1 and situation 2) and get the number of its dynamic binding services, composition and execution time and other experimental data. Static analysis time itself will also be measured. The experiment result is shown as Figure 2 and Figure 3.

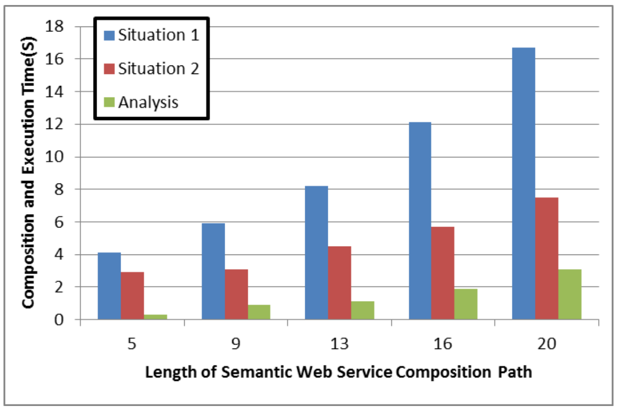

FIGURE 2 Composition and execution time

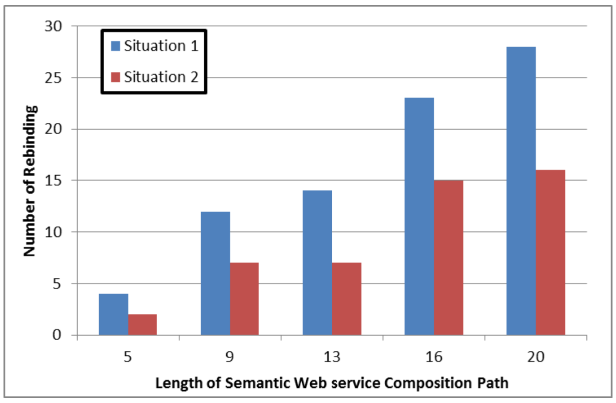

FIGURE 3 Number of rebinding

7 Related work

Currently, some research work focused on the access control for single Web service. But there has not been a good solution for the problem of access control for Web service composition especially semantic Web service composition. Access control for the semantic Web service composition problem has gradually become one of the hot research fields among semantic Web service security [11]. The attribute-based access control (ABAC) has been introduced to access control Web services [12, 13]. It realized the access control for a single Web service, but did not consider the access control for Web service composition. C. Ardagna et al. consider the credential based access control with the abstraction of complex concepts such as set disjunction/conjunction and so on, into a single concept in policy specification [14]. M. Srivatsa et al proposed the access control system for service composition which meets the security needs of a dynamic web service environment [15]. But it did not involve its application in the semantic Web service composition Access Control. E. Bertino introduced role-based access control (RBAc) model to BPEL (Web Service Business Process Execution Language) and proposed access control architecture known as RBAC-WS-BPEL [16]. But there are limitations in terms of dynamics and control granularity of access control since it is based on RBAC model. F. Satoh studied the combination method of security policy using predicate logic and proposed auto security policy generation mechanisms for service composition [17]. But it did not solve the problem of policy conflict in service composition path. F. Paci et al. presented the controlled dissemination of policy information to users in conversational web service [18].

In the semantic Web access control research, S. Agarwal pointed out that access control policy of semantic Web service composition in general was combined with all access control policies of component services produced [19]. But it just described the combination process of access control policy for service composition without detailed policy combination approach. G. Bayer et al. presented a personal file sharing architecture based on ABAC and semantic web technology [20]. They suggest a harvesting mechanism to capture the user data from the social and professional network sites, thus extends the attribute ontology with new rules and relationships. T. Chowdhury and J. Noll proposed a semantic aware role based identity management mechanism which provided secure access to enterprise content management system [21]. T. Priebe et al. extended the ABAC with semantic web technology for highly open system like the Internet [22]. Specifically, they extended XACML language with the inference engine and ontology administration point (OAP).

8 Conclusion

We have developed a defeasible policy based semantic Web service composition approach. Before service composition, all the rules in policies of composition semantic Web service, semantic Web services and component Web service will be modeled as strict rule, defeasible rule and defeat rule of defeasible logic. And then all the policies in the composition path in which the composition semantic Web service and the component Web service have trust relationship will be analyzed. After the static analysis, the impossible path will be removed. In the composition and execution time, the composition engine only checked the Web service in the list of candidate component service. In that case, the composition time will be reduced and successful rate increased. The experiment result proved such above conclusion.

Our approach can greatly reduce the potential access deny and increase composition efficiency. However, our approach only considered the access control policy of semantic Web service and component Web service, but did not include the access control of external data resource. The access deny of data resource will also reduce the successful rate of composition. Therefore, the data access control in the semantic Web service composition will be a research direction in our future work.

9 Conflict of interest

CONFLICT OF INTEREST: Financial contributions to the work being reported should be clearly acknowledged, as should any potential conflict of interest.

Acknowledgments

This work was supported by a grant from the Hubei Provincial Department of Education scientific research programs for Youth project (No. Q20133003) and the natural science foundation of Hubei Province (No. 2014CFB568).

References

[1] Demian D, Ananthanarayana S 2010 Dynamic Web Service Composition Based on Operation Flow Semantics *International Journal of Computer Applications* **26**(1) 4-14

[2] Kim K, Choi W, et al. 2008 A Collaborative Access Control Based on XACML in Pervasive Environments *Proc. of 2008 IEEE Conference on Hybrid Information Technology* 7-13

[3] Yannick C, Mohamed M 2008 Automatic Composition of Services with Security Policies *Proc. of 2008 IEEE Congress on Services* 529-38

[4] Nute D 1987 Defeasible Logic *in Handbook of Logic in Artificial Intelligence and Logic Programming. Oxford University Press* 353-95

[5] Maher M J 2001 Propositional defeasible logic has linear complexity *Theory and Practice of Logic Programming* 1(6) 691-711

[6] Hu L, Ying S, et al. 2009 Towards an Approach of Semantic Access Control for Cloud Computing *Proc. of the 1st International Conference on Could Computing, Beijing China Springer*

[7] Hu L, Ying S, et al. 2010 A Semantics Based Approach for Cross Domain Access Control *Journal of Internet Technology* **11**(2)

[8] OASIS, Extensible Access Control Markup Language (XACML), [Online]

[9] http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf

[10] Hu L, Qiu C, Shi Y 2012 A defeasible description logic based semantic security policy conflict detection approach *International Journal of Security and its applications*

[11] IBM, Rational Application Developer, [Online]

[12] http://www.ibm.com/developerworks/downloads/r/rad/?S_CMP=rnav

[13] Brown K, Capretz M 2013 ODEP-DPS: Ontology-Driven Engineering Process for the Collaborative Development of Semantic Data Providing Services *Information and Software Technology* **55**(9) 1563-79

[14] Hebig R N, Meinel C, Menzel M, et al 2009 A Web Service Architecture for Decentralised Identity- and Attribute-Based Access Control *Proc. of IEEE International Conference on Web Services* 551-8

[15] Yuan E, Tong J 2005 Attribute based Access Control for Web Services *Proc. of the 2005 IEEE International Conference on Web Services, IEEE Computer Society* 561-9

[16] Ardagna C, Vimercati S, Paraboschi S, et al 2011 Expressive and Deployable Access Control in Open Web Service Applications *IEEE Trans. Services Computing* **4**(2) 96-109

[17] Srivatsa M, Ivengar A, Mikalsen A, et al 2007 An Access Control System for Web Service Compositions *Proc. of the 2007 IEEE International Conference on Web Services, IEEE Computer Society* 1-8

[18] Bertino E, Cramptou J, Paci F 2006 Access Control and Authorization constrains for WS-BPEL *Proc. of the 2006 IEEE International Conference on Web Services IEEE Computer Society* 275-84

[19] Satoh F, Tokuda T 2008 Security Policy Composition for Composite services *Proc. of the 2008 IEEE International Conference on Web Engineering IEEE Computer Society* 86-97

[20] Paci F, Mecella M, Ouzzani M, et al 2011 ACCONV - An Access Control Model for Conversational Web Services *ACM Trans. Web* **5**(3) article 13

[21] Agarwal S, Sprick B 2004 Access control for Semantic Web Services *Proc. of the 2008 IEEE International Conference on Web Services* IEEE Computer Society 770-3

[22] Bayer G, Sengupta D, Wang T, et al PrplAc: Attribute-based Access Control in PRPL for Fine Grained Information Sharing using Semantic Web Technical Report, http://senguptas.org/Documents/cs343-PrplAc.pdf

[23] Chowdhury M, Noll J 2007 Access Control and Privacy Enhancement through Role-based Identity Management *Telektronikk (Norwegian Telecommunications Journal published by Telenor AS.)* **103**(3/4) 161-70

[24] Priebe T, Dobmeier W, Schläger C, et al 2007 Supporting Attribute-based Access Control in Authorization and Authentication Infrastructures with Ontologies *Journal of Software* **2**(1) 27-38

Authors

	Hu Luokai, born in October, 1981, Wuhan P.R. China. **Current position, grades**: postdoctor of Lenovo Mobile Communication Technology Co. Ltd. Associate Professor of Hubei University of Education. **University studies**: PhD from Wuhan University, 2011, in China. **Scientific interest**: semantic computing, mobile computing. **Publications**: more than 20 papers. **Experience**: 5 years of experience in semantic Web, software engineering and mobile computing, 3 scientific research projects.
	Chao Liang, born in June, 1972, Hunan P.R. China. **Current position, grades**: senior engineer of Lenovo Mobile Communication Technology Co. Ltd. **University studies**: PhD from Chinese Academy of Sciences Institute of Computing Technology, 1999, in China. **Scientific interest**: mobile computing and communication technology. **Publications**: more than 8 papers. **Experience**: 15 years of experience in mobile computing and communication technology, 20 scientific research projects.
	Ying Lu, born in March, 1994, Wuhan P.R. China. **Current position, grades**: undergraduate students of Hubei University of Education. **University studies**: undergraduate students from Hubei University of Education, 2016, in China. **Scientific interest**: semantic computing. **Publications**: 1 paper. **Experience**: participation in 1 scientific research project.
	Yan Zeng, born in March, 1993, Wuhan P.R. China. **Current position, grades**: undergraduate students of Hubei University of Education. **University studies**: undergraduate students from Hubei University of Education, 2016, in China. **Scientific interest**: semantic computing. **Publications**: 1 paper. **Experience**: participation in 1 scientific research project.