A Security Policy Description Language for Distributed Policy Self-management

Zengbang Ma[1, a], Yingjie Yang[1, a] and Yutong Wang[1, b]

[1]School of Information Engineering University, Zhengzhou 450001, China;

[a]184767606@qq.com, [b]wyt83780345@126.com

Keywords: policy description language; XML Schema; XML; meta policy; self-management.

Abstract. The existing security policy description languages can't support distributed policy self-management and have some problems in the aspect of describing capability, extensibility, portability and other issues. Against these problems, this paper proposes a security policy description language SPDL for distributed policy self-management. SPDL uses XML Schema to define basic elements and grammatical structure of the policy. And it uses XML to complete the policy configuration. It uses meta policy to realize distributed policy self-management. Finally, this paper gives an analysis of the characteristics and advantages of SPDL comparing with other policy languages.

Introduction

Internet Engineering Task Force (IETF), Desktop Management Task Force (DMTF) and many academic institutions and enterprise vendors believe that policy-based management is the most promising methods to solve the management problem of large distributed system, and policy description language is the basis of policy-based management.

Nowadays there have been many different policy description languages: IETF proposed a network management definition language PFDL [1], PFDL is an aggregation of a series of policy rules. These rules are organized in an IF (Condition) THEN (Action) paradigm. So it can only describe a series of operations under certain conditions, and can't support the policy triggered by events. Bell Labs proposed policy description language PDL for network management [2], PDL is a form of ECA rules which is event - condition - action, PDL is an event-based policy description language but it doesn't support the authorization policy. Ponder[3] is a declarative and entity-oriented policy language from Imperial College. There are four basic types of policies in Ponder: authorization policies, refrain policies, delegation policies and obligation policies. The former three types are targeted at defining access control policies, and obligation policies are used to support event-based policies. Ponder also provides composite policies composed of basic policies. World Wide Web (W3C) proposed P3P [4], P3P only supports the confidentiality between network communication, it doesn't support access control policies and obligations policies. HP Labs presented Rei policy description language [5], Rei is a declarative policy specification language based on deontic logic. It is concerned with obligation, permission and so on. It is designed mainly for security and privacy in dynamic and open computing environments. Its extensibility is poor and it is difficult for administrator to understand. Organization for the Advancement of Structured Information Standards OASIS ratified XACML[6] policy description language, XACML is an XML-based declarative policy description language mainly used for access control management of distributed systems, although XACML support fine-grained access control description, the language is quite redundant, and it doesn't support obligation policies. Sushil Jajodia et al. [7] proposed ASL policy description language, ASL is a first-order logic-based access control language. The authorization decisions of ASL is coded in the rule itself, attached with a role set, rather than dynamically determined by a PDP explicitly. So the decision making mechanism of ASL lacks sufficient flexibility and reusability. ASL can only support access control policy, does not support the description of the obligation policies.

Although there are so many policy description languages, they are still no standardization. The description ability, extensibility and portability of most policy language still need to be improved. Otherwise, these policy description languages require the administrator to distribute policies manually. In the management of large distributed system depending on distributed policy management it is inefficient and error prone. The amount of policy in large distributed system is huge. So it is hard for the administrator to choose proper policies to satisfy the requirements when the environment changes. The automatic policy management[8] is essential.

So we propose a security policy description language for distributed policy management SPDL aiming at simplifying the management of policy to manage policies automatically and improving the description ability, extensibility and portability of policy language.

Design of SPDL

Policy levels

In this paper, we think that the policy has three levels as show in figure 1.

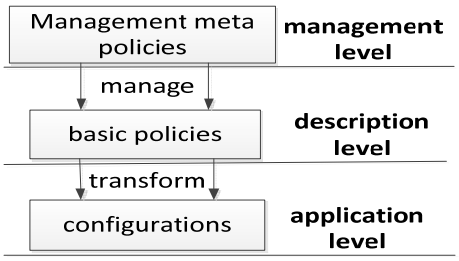

Figure 1 policy levels

The lowest level is application level which provides concrete configurations of different security devices like VPN, firewalls. It is related to concrete security device. So this paper is not concern with application level.

The middle level is description level which provides basic policies. There are so many kinds of security devices, it is not possible for administrators to know how to configure every kind of device. So the description level provides a unified policy description method to describe concrete configurations. The basic policies can transform into configurations.

The highest level is management level which provides management meta policies which are used to manage basic policies. Meta policy is short for management meta policy. And in the next of this paper meta policy[9] represents management meta policy. The way of using meta policies has many advantages. Administrators don't need to find out the proper policy from the large amount. They only need to know the right meta policies which are encapsulated and described detailed for certain environment. Otherwise, it is possible to realize dynamic management for distributed policy management by using of meta policies. Different meta policies meet different environments. When the environment changes, the system captures the information and decides which meta policies are appropriate. Then the basic policies managed by these meta policies will be enabled.

If we use Mi to represent meta policy and Pj to present basic policy, then the logical relationship can be expressed as Mi={Pi1,Pi2,…Pij}. The meta policy also satisfies that:

(1) $\exists M_i, M_j, stM_i \cap M_j = P_i$

That means different meta policy may manage the same basic policy.

(2) $M_i \cup M_j = \{P_{i1}, P_{i2}, \cdots, P_{im}\} \cup \{P_{j1}, P_{j2}, \cdots, P_{jn}\}$

That means the combination of different meta policies can realize different combination of basic policies.

The way to describe policies

XML proposed by the World Wide Web Consortium was a markup language used to describe structured or semi-structured data. It is designed to support data exchange between different systems. XML has many advantages: Strong expressive, well extensibility and portability. The tags of XML

come from XML Schema which offers facilities for describing the structure and constraining the contents of XML as well as provides a shared vocabulary.

SPDL uses XML Schema to define the policy structure which contains policy elements and the data type of these elements, and the policy structure is maintained in an XSD file. The specific configuration of concrete policy is done with the usage of XML, and the policy configuration is maintained in an XML file. New tags of policy type can be incorporated very easily via self-describing, extensible nature of XML.

Although it is flexible to allow administrators to customize the policy structure, it poses a problem. Different administrators may define the same policy element as a different tag when they are defining policy structure. This can inevitably lead to semantics conflict. We use the security policy ontology and formulate the standardization of security policy metadata in the way of constructing ontology knowledge base. This provides a shared vocabulary for policy elements and resolves the semantic conflict of policy elements.

The design idea of SPDL is shown in Figure 2.

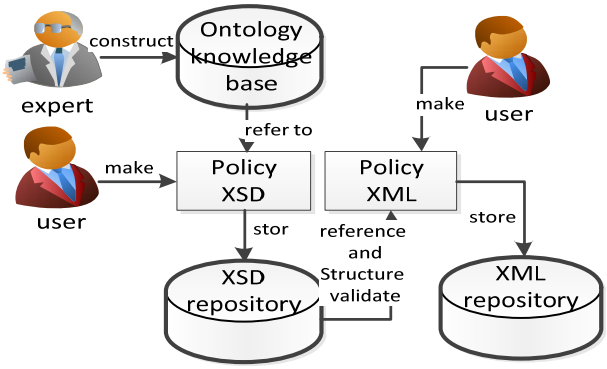

Figure 2 the design idea of SPDL

Experts of security policy develop the security policy ontology to construct the ontology knowledge base which provides the required policy elements to define policy structure. Administrators develop policy structure. The tags defined in the XSD file are from ontology knowledge base. The policy structure files were stored in the XSD repository. Administrators get policy structure from the XSD repository and develop policy configuration XML file according to the policy structure. The policy configuration XML file should be validated by the policy structure XSD file. The XML file will be stored in the XML repository unless its structure meets the policy structure defined in the XSD file.

The core issue to be solved of SPDL is how to generate valid XSD files and XML files. Although ontology knowledge base can help to solve the probable semantic conflict of policy elements when define policy structures. Standardized syntax is essential. The description method of SPDL determines that its grammar contains policy structure grammar and policy configuration grammar.

Policy types and policy elements

In SPSL, policies are divided into two main types: basic policies and meta policies. The meta policies can be seen as the policy of basic policies. They managed the basic policies. The basic policies are divided into three types: access control policy, event trigged policy and attribute configuration policy.

Access control policy constraints the operation that the subject does on the subject. It includes allowing or denying the access of subject to the target and the subject delegating authority to target. Event trigged policy refers to the subject should have to do some behavior when an event occurs. And it provides a response to the change of environment at the same time. Attribute configuration policy is used to complete the configuration of security parameters for the safety of policy entities. Unlike with Event trigged policy, it isn't triggered by event. Attribute configuration policy provides a range of security parameters for the participation of network communication.

The policy elements are the basis of forming a complete policy, different combinations between different policy elements form different types of policies.

Definition 1: Policy entity refers to subject and target. Subject is the sponsor of security behavior and target is the receiver. Policy entity of SPDL is expressed in domain[10] scope expression. It can provide an index for the policy.

Domain can be regarded as way to divide policy entities into groups according to geographical boundaries and entity types for the convenience of administrator to manage. The organization of domain is similar to a file system directory. If a domain contains members of another domain, then the former is called parent domain, and the latter is called subdomain. Domain can overlap, that refers to a subdomain can belong to more than one parent domains. We can dynamically change the domain member inside of changing the policy itself by way of using domain scope expression.

Policy entities can be either individual member or all members of a domain. It may also come from a combination of different domains. We use the domain operator which is shown in Table 1 to operate the domain.

Table 1 domain operator

Operator	Explanation
A/B	Returns the child member B of domain A. B may be a subdomain as well as a single member.
$A \cup B$	The union of A and B. Returns the set contains all the members from A and B
$A \cap B$	The intersection of A and B. Returns the set contains the members not only in A but also in B.
A-B	The difference of A and B. Returns the set contains the members only in A but not B.

Different policy entity in different domain may have the same policy. The use of domain operator is error-prone when it comes to multiple domains. We define the entity which has the same policy as a certain role [11] in order to simplify the operation. Then the administrator can add or remove role members without having to change policy.

Definition 2: Policy action means the behavior sponsored by the subject to influence the target. The policy entity or security system will implement one or a series of actions after a policy is triggered. These actions are predefined by the administrator, each action has its own identity. A series of actions constitute action list, each action in the action list is associated with an action connector. Action connector includes "->" and "&". A1-> A2 means that A2 will be executed after A1 has been done. A1 & A2 means A1 and A2 will be executed concurrently.

Definition 3: Events are used to trigger the event-based policy. The event may be a simple event, it may also be composite event comes from compounding of simple events. Event was predefined by the system administrator, and each event corresponds to an event identification.

Composite event was compounded by a series of related events through event operator which is shown in Table 2.

Table 2 event operator

Operator	Explanation
E1&E2	Occurs when both E1 and E2 occur irrespective of their order
E1| E2	Occurs when either E1 or E2 occurs irrespective of their order
E1-> E2	Occurs when E1 occurs before E2
E^ n	Occurs when E occurs n times, where n is an integer value

Definition 4: Policy condition refers to the refrain of policy. Policy condition contains multiple sub-conditions, when all the sub-conditions are true the policy can be performed.

Definition 5: Attribute parameter can be seen as security attribute used to ensure the security of policy entity. An attribute configuration policy will contain multiple attributes, therefore attribute parameter element contains multiple sub-elements, these sub-elements can be password algorithm, password length, communication key parameters and other security attributes parameters. Their definition obeys the metadata standards defined by security ontology.

Definition 6: Authorization rule is used in access control policy. It has only two values: permit and deny. The former means it is allowed for the subject to do some behavior on the target while the later means it is not allowed.

Definition 7: Security level. Different environment corresponds to different security level and different security level corresponds to different basic policies.

Definition 8: Policy state. Meta policy maintain four states: initiated, enabled, disabled and deleted. The state transition is shown in figure 3. The new policy made by administrator will be stored in the policy repository in the initiated state. The initiated policies will turn to enabled state when they are distributed. The enabled policy can be disabled and the disabled policy can be re-enabled. Policy will not be removed immediately from the policy repository. These policies will be in the deleted state. The policies which are in deleted state will be removed from the policy repository until the policy repository is updated after a period of time.

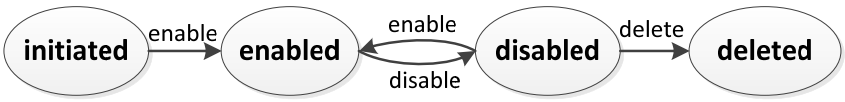

Figure 3 state transition

Syntax of SPDL

The syntax of SPDL is maintained into two parts: the policy structure and the policy configuration. Policy structure completes the definition of policy elements and organizes the elements to form different policy types. Policy configuration completes the configuration on the basis of policy structure. Many policies have the same policy structure, so the most of the structures are pre-defined. What the administrator needs to do is completing the policy configuration.

The policy structure syntax of access control policy is show in figure 4.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3school.com.cn"
xmlns="http://www.w3school.com.cn"
elementFormDefault="qualified">
<xs:element name="accesscontrol">
<xs:complexType>
<xs:sequence>
<xs:element name="PolicyId" type="xs:string"/>
<xs:element name="Subject" type="Scope_Type"/ >
<xs:element name="Target" type="Scope_Type"/ >
<xs:complexType name="Scope_Type" >
<xs:attribute name="Scope">
 <xs:simpleType>
 <xs:restriction base="xs:string">
 <xs:pattern value="Domain|Role">
 </xs:restriction>
 </xs:simpleType>
</xs:attribute >
</xs:complexType>
<xs:element name="Action" type="xs:string"/>
<xs:element name="AuthRule" type="xs:string"/>
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="Permit"/>
<xs:enumeration value ="Deny"/>
</xs: restriction >
</xs:simpleType>
<xs:element name="Condition">
<xs:complexType>
<xs:sequence>
<xs:element name=" Condition1" type="xs:string"/>
<xs:element name=" Condition2" type="xs:string"/>
……
<xs:any minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:any minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Figure 4 policy structure syntax of access control policy

Access control policy contains five main elements: subject, target, action, authorization rule and condition. Each element has its own tag which is unique in the restriction of ontology knowledge base. In some policies the element condition is not essential. Some policies' conditions are totally different while some policies have some common condition elements. SPDL provides a way to simplify this. It uses an XSD file to define the common elements and another XSD file to define the different element.

The XML file can refers to this two XSD file. In this way the XSD file which defines the common elements can be reused for many times.

The configuration syntax of the policy structure show in figure 4 is shown in figure 5. The policy configuration completes that gives each element defined in the policy structure concrete value.

```
<?xml version="1.0"?>
<accesscontrolpolicy
xmlns="http://www.w3school.com.cn"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3school.com.cn *.xsd">
<PolicyId>policyid</PolicyId>
<Subject Scope= Domain/Role> Domain_Express /Role_Express </Subject >
<Target Scope= Domain/Role>Domain_Express /Role_Express</Target >
<Action> Action_List</Action>
<AuthRule>Permit/Deny</AuthRule>
<Condition>
<Condition1> Condition_Express1 </Condition1>
<Condition2> Condition_Express2 </Condition2>
……
</Condition>
</accesscontrolpolicy>
```

Figure 5 policy configuration syntax of access control policy

The syntax of event trigged policy and attribute configuration policy is the same with access control policy. No matter syntax of policy structure or syntax of policy configuration. The only difference is the policy elements are different. The policy structure syntax of event trigged policy completes the definition of subject, target, action, event and condition. The condition element is optional. The policy structure syntax of attribute configuration policy completes the definition of subject and attribute parameter. The attribute parameter element may have many sub-elements. After the structure is well defined the policy configuration can be completed in the restriction of policy structure.

Meta policy can be expressed as: MetaPolicy=<PolicyId, TargetPolicy, Security_Level, State, Description>. “MetaPolicy” represents meta policy. “PolicyId” is the identification of meta policy. TargetPolicy =($Policy_1$, … , $Policy_n$). It refers to the basic policies managed by meta policy under different external environment. “Security_Level” indicates the security level and “State” indicates the state of the basic policies maintained by the meta policy. The meta policy changes along with the changes of the environment. Then the state of basic policies maintained by the proper meta policies will be changed from disabled state to enabled state. Description refers to the detailed information of the effect of basic policies managed by this meta policy. These identifications introduced above are the elements of meta policy. Policy structure syntax of meta policy completes the define of these elements and policy configuration syntax completes the concrete value of these elements.

Analysis of SPDL

Weili Han et al.[1] put forward six characteristics which can be regarded as policy description language evaluation criteria. They are ECA rules, XML, index, role-based access control RBAC, obligations and formal description. ECA rule refers to support event - condition - action formats. XML means whether the representation of this language is encoded in the XML format. Index refers to whether there exists a special item for policy engine to retrieve the required policies more efficiently. Obligation means that once the event occurs the policy language can trigger tasks that must be performed. If the policy language lacks the description of obligation, it is not expressive enough. Formalization refers to whether this policy description language is directly in logical rule’s form or can be converted into formal description. This feature determines the policy description Language can do automated reasoning and conflict detection. In our opinion, these characteristics are essential and it should help to simplify the management of policy and provide the self-management.

Table 3 shows the compare between SPDL and other policy languages.

Table 3 comparison between SPDL and other policy description languages

policy description language	XML	index	access control	obligation	security parameter	formalization	self-management
SPDL	✓	✓	✓	✓	✓		✓
PFDL			✓				
PDL				✓			
Ponder		✓	✓	✓	✓		
P3P	✓				✓		
Rei		✓	✓	✓	✓	✓	
XACML	✓	✓	✓		✓		
ASL			✓			✓	

SPDL is based on XML, so it inherits the advantages of XML. Each type of SPDL policy has at least one policy entity, namely policy subject or policy entity. The policy entity provides an index term to help to retrieve policies. It has two main policy types: basic policy and meta policy. The basic policy can divided into three types: access control policy, event-triggered policy and attribute configuration policy. It supports the description of all these policies. Although SPDL is not formal description, it has applied ontology which can support logical reasoning. And XML can easily transform into ontology description language. So it is easy for SPDL to translate into formal reasoning. It uses meta policy to manage basic policies. This simplifies the management of policy and provides automatic management.

SPDL not only meets the design principle of policy language, but also has good performance.

SPDL is good expressive, the mark of XML document describes the structure and semantics of the document. We can clearly understand the correlation between the various policy elements. SPDL has good extensibility, it supports to custom policy tags, and the newly defined tags comply with the ontology metadata standard of ontology knowledge base. This helps to realize the expanding of policy description language. SPDL is well compatible, portable and platform-independent, policy files can be directly transferred from one system to another without any handling. The receiver can understand the meaning of the documents and data, and validate its contents. SPDL is ease to use, its syntax is simple. Its XSD file can be applied by many XML files to complete the description of concrete policy once the structure of policy is define. Tags used in the XML file can also from more than one XSD files. In this way there is no need to define a completely new policy structure if the new structure only has a little difference from the old one. The administrator only need to define the different tags in another XSD file and make sure this file is referenced when he is describing policies with XML file.

Summary

If you follow the “checklist” your paper will conform to the requirements of the publisher and facilitate a problem-free publication process.

In this paper we presented SPDL which provides a flexible, formal, and extensible language to allow the security administrators developing their own security policies with the rules in a readable and formal format. The use of meta policy help to simplify the management of policy and provide automatic management.

Although the syntax of SPDL is simple it requires the administrator to know some knowledge about XML. Next we will develop the interface used to make policies easily for the administrator. And we will study the algorithm which can realize automatically changing basic policies to adapt to the changing environment.

References

[1] Weili Han, Chang Lei. A Survey on Policy Languages in Network and Security Management [J]. Computer Networks, 2012, 56: 477-489.

[2] Damianou N, Bandara A, Sloman M, et al. A survey of policy specification approaches[J]. Department of Computing, Imperial College of Science Technology and Medicine, London, 2002, 4: 1-37.

[3] Damianou N, Dulay N, Lupu E C, et al. Ponder: A Language for Specifying Security and Management Policies for Distributed Systems. Imperial College[R]. UK, Research Report Department of Computing 2001, 2000.

[4] Cranor L, Langheinrich M, Marchiori M. A P3P preference exchange language 1.0 (APPEL1. 0)[J]. W3C working draft, 2002, 15.

[5] Mitchell-Wong J, Goh S K, Chhetri M B, et al. Framework for Open, Distributed and Self-Managed Social Platforms[M]//Pervasive Collaborative Networks. Springer US, 2008: 361-368.

[6] Rissanen E. extensible access control markup language (xacml) version 3.0[J]. 2013

[7] Duflos S, Diaz G, Gay V, et al. A comparative study of policy specification languages for secure distributed applications[M]//Management Technologies for E-Commerce and E-Business Applications. Springer Berlin Heidelberg, 2002: 157-168.

[8] Choudhary A R, Raggad B G. Digital policy management requirements and architecture [C]//Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), 2014 IEEE International Inter-Disciplinary Conference on. IEEE, 2014: 144-150.

[9] Díaz-López D, Dólera-Tormo G, Gómez-Mármol F, et al. Managing XACML systems in distributed environments through Meta-Policies[J]. Computers & Security, 2015, 48: 92-115.

[10] Sonkoly B, Czentye J, Szabo R, et al. Multi-Domain Service Orchestration Over Networks and Clouds: A Unified Approach[C]//Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication. ACM, 2015: 377-378

[11] Sehra S S, Singh J. Policy Specification in Role based Access Control on Clouds[J]. arXiv preprint arXiv:1308.5177, 2013.