Label-Based Access Control Policy Enforcement and Management

Wei Zhou, Vinesh H. Raja
School of Engineering
University of Warwick
Coventry CV4 7AL, UK

Christoph Meinel
Hasso-Plattner-Institute
University of Potsdam
D-14482 Potsdam, Germany

Munir Ahmad
B2B Manufacturing Centre
University of Teesside
Middlesbrough TS1 3BA, UK

Abstract

To effectively participate in modern collaborations, member organizations must be able to share specific data and functionality with collaboration partners, while ensuring their resources are safe from inappropriate access. This requires access control models, policies, and enforcement mechanisms for the shared resources. This paper specifically addresses how to reduce the information leaks caused by authorization policies used in collaborative computing environment. The basic principle is defining some labels that specify the information flow constraints, and assigning them to authorization policy components. The usages of labeled policy components must obey the information flows constraints defined by the labels in order to avoid authorization policy components being misused. This label can also improve the authorization policy administration.

1. Introduction

With the advent of the information superhighway, businesses, governments and other organizations cooperate in innovative ways. To effectively participate in modern collaborations, member organizations must be able to share specific data and functionality with collaboration partners, while ensuring their resources are safe from inappropriate access. Such collaborations may dynamically change participants and trust relationships during the life cycle. The dynamic and multi-institutional nature of these environments introduces challenging security issues that demand new technical approaches for authorization policy enforcement and management. In this paper we specifically address the following problem: how to enhance the normal access control policy enforcement and management to reduce information leaks in the collaborative computing environment.

In this paper we proposal a Label-Based Access Control Policy (LBACP) model that can be used for access control policies management and enforcement in the collaborative computing environment. The basic idea of LBACP is placing another layer on the top of normal access control policies, i.e. add labels to the policy components. Labels are used to define the information flow. For example, add a label to the "Engineer role" used in a collaborative environment, through modifying the label definition, this role can only be used in some of the partner organizations, but the access control policies do not need to be changed. Each policy component can have one or more associated labels.

The rest of this paper is organized as follow. Section 2 introduces the label model and its basic operations. Section 3 describes label assignment and information flow. Section 4 investigates label policy for access control and policy management. Section 5 compares our work to some related works. Finally, section 6 gives the conclusion and future work.

2. Label model

2.1. Label structure

In the LBACP model there are three essential elements: *contexts*, *label polices* and *labels*. Contexts are the entities whose privacy is protected by the model, label policies are the ways that principals express their privacy concerns, and labels are composed with a set of label policies. Information is owned by, updated by, and released to contexts, which are the basic elements in the LBACP model. For example, the departments and groups at a university could be modeled as contexts. A label policy has three parts: an *owner context*, a set of *import contexts* and a set of *export contexts*. The owner context is a context whose data was observed. The import contexts are a set of contexts, from which the owner context accepts information flow. The export contexts are a set of contexts, to which the owner context allows information flow. Note that the notion of context used in this approach does not refer to dynamic environment parameters.

One label may contain multiple label policies. The intuitive meaning of a label is that every policy in the label must be obeyed as data flows through the system, so

Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'06)
0-7695-2611-X/06 $20.00 © 2006 IEEE

labeled information is permitted to flow in or out only by the consensus of all the policies. The information may flow to a context only when this context is in export contexts for every policy in the label. The information may flow from a context only when this context is in import contexts for every policy in the label. Because the intersection of all of the policies is enforced, adding more policies to a label only restricts the propagation of the labeled data.

An example of an expression that denotes a label L with two label policies is the following: $L = \{o_1 : i_1, i_2 : e_1, e_2 ; o_2 : i_2, i_3 : e_2, e_3\}$, where o_1, o_2, i_1, i_2, i_3, e_1, e_2, e_3 denote contexts. Semicolon separates two policies within the label. The owners of these policies are o_1 and o_2. The import sets for the policies are $\{i_1, i_2\}$ and $\{i_2, i_3\}$, respectively. The export sets for the policies are $\{e_1, e_2\}$ and $\{e_2, e_3\}$, respectively. The import context set of the label is $\{i_2\}$, and the export context set is $\{e_2\}$.

2.2. Information flow definition

2.2.1. Information flow between label policies. If P is a label policy, then the notation $o(P)$ denotes the policy owner, the notation $i(P)$ denotes the set of import contexts, the notation $e(P)$ denotes the set of export contexts. If I and J are two label policies and C represents the set of all contexts, the information flow between them is defined as:

$$I \mapsto J \Leftrightarrow (I = J) \vee ((o(J) \in e(I) \vee e(I) = C) \wedge (o(I) \in i(J) \vee i(J) = C))$$

2.2.2. Information flow between labels. If L is a label, then $lo(L)$ denotes the union of the owner contexts, the $li(L)$ denotes the intersection of the import contexts of all the label policies, and the $le(L)$ denotes the intersection of the export contexts of all the label policies. In one label, one label policy cannot block other owners' information flow, i.e., information can freely flow among the owner contexts in one label, but it can block other owners' import and export information flow. Reason for this is because the labeled resource is owned by them together. If L_1 and L_2 are two labels, the information flow between them is defined as:

$$L_1 \mapsto L_2 \Leftrightarrow (L_1 = L_2) \vee ((le(L_1) \supseteq lo(L_2)) \wedge (lo(L_1) \subseteq li(L_2)))$$

2.2.3. Information flow channel between labels. In the real application we more concern about *information channel* between two labels. A channel is composed with *input channel* and *output channel*. Input channel describes which contexts are the source contexts of a channel. Output channel describes which contexts are the target contexts of a channel. Information can flow from any input channel context to any output channel context. If L_1 and L_2 are two labels, and the information flow direction is from L_1 to L_2, then the input channel, output channel and information channel are defined as:

$$input\_channel(L_1, L_2) = lo(L_1) \cap li(L_2)$$
$$output\_channel(L_1, L_2) = le(L_1) \cap lo(L_2)$$
$$information\_channel(L_1, L_2) = (input\_channel, output\_channel)$$

For example, B2B, VRC, WMG and IARC are four contexts, L_1 and L_2 are two labels defined as follow:

$$L_1 = \{B2B : VRC : VRC; WMG : VRC : VRC\}$$
$$L_2 = \{VRC : B2B, IARC : B2B, IARC\}$$

The information channel calculation process from L_1 to L_2 can be described as follow:

$$lo(L_1) = \{B2B, WMG\}$$
$$le(L_1) = \{B2B, WMG, VRC\}$$
$$lo(L_2) = \{VRC\}$$
$$li(L_2) = \{VRC, B2B, IARC\}$$
$$input\_channel(L_1, L_2) = \{B2B\}$$
$$output\_channel(L_1, L_2) = \{VRC\}$$
$$information\_channel(L_1, L_2) = (\{B2B\}, \{VRC\})$$

2.3. Label composing

In our model one policy component can be assigned with multiple labels. This section describes the syntax of the label composition algebra that consists of a collection of labels, a collection of operators to combine them. Label composition operators in our algebra are disjunction and conjunction, and separation. Accordingly, for disjunction and conjunction, operations on labels are interpreted as relational operators such as union and intersection. The separation simply specifies there is no any relation among labels.

Disjunction operator permits accesses that are allowed under either of its components, i.e. disjunction merges two labels by returning their union. Formally,

$$L_1 \cup L_2 = \left\{ \begin{pmatrix} O, \cup\{i(K) / K \in L_1 \vee K \in L_2, o(K) = O\}, \\ \cup\{e(K) / K \in L_1 \vee K \in L_2, o(K) = O\} \end{pmatrix} / O \in o(L_1) \vee O \in o(L_2) \right\}$$

An example of label union is show as follow:

$$L_1 = \{B2B : VRC, IMRC : VRC\}$$
$$L_2 = \{WMG : VRC : VRC, IARC\}$$
$$L_1 \cup L_2 = \{B2B : VRC, IMRC : VRC, IARC; WMG : VRC, IMRC : VRC, IARC\}$$

For instance, there is a portal manages a virtual organization that manages a dynamic collection of resources and users from different organizations. Access to this portal can be authorized by any of the administrators of different participant organizations, and such information can be organized into labels that describe the valid users' domains. The totality of the accesses through the portal to be authorized should then be the union of the statements of each organization. Intuitively, disjunction can be applied in any situation where accesses can be authorized if allowed by any of the component policies.

Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'06)
0-7695-2611-X/06 $20.00 © 2006 IEEE

Conjunction operator allows only those permitted by both components, i.e. conjunction merges two labels by returning their intersection. Formally,

$$L_1 \cap L_2 = \left\{ \begin{pmatrix} O, \bigcap \{ i(K) / K \in L_1 \vee K \in L_2, o(K) = O \}, \\ \bigcap \{ e(K) / K \in L_1 \vee K \in L_2, o(K) = O \} \end{pmatrix} / O \in o(L_1) \vee O \in o(L_2) \right\}$$

For the two labels used in the label disjunction part, the intersection of the two labels is:

$$L_1 \cap L_2 = \{B2B : VRC : VRC; WMG : VRC : VRC\}$$

Disjunction allows an access if any of the component policies allows it, whereas conjunction requires all the component policies to agree on the fact that the access should be granted. Intuitively, although disjunction enforces maximum privilege, conjunction enforces minimum privilege. For instance, consider a virtual organization in which participant organizations share certain documents (e.g., clinical folders of patients). An access to a document may be allowed only if all the authorities that have a say on the document agree on it, i.e. whether this document is permitted flow to some contexts.

Separation operator allows multiple labels can be assigned to a component, but among of them there is no any relation. At runtime, only one of them is used for a particular access control. This separation also allows system administrators manage access control policies in multiple dimensions. Formally,

$$L_1; L_2$$

For example, there is a set of rules for the access control of patients' information in a hospital, which may be assigned with label $L_{surgery}$, or $L_{medicine}$, or both of them, but separated with the separation operator. At runtime the access control engine may only use the rules labelled with $L_{surgery}$ or $L_{medicine}$ for a particular access control.

3. Label assignment and information flow

We introduced labels in order to annotate access control policy components with them. To enable fine-grained information flow restrictions, we allow label specification to access control policies at different levels, i.e. assign labels to rule elements, rules and policies. The labeled entities have to conform to the information flow constraints specified by the labels. For example, a labeled "role" in a RBAC policy can only be used in the specified contexts. Some label calculation must be followed in order to avoid information leaks. A policy component without labels means it does not care the information flow constraint.

In a labeled policy, the information flow direction is from lower level (e.g. rule element) to higher level (e.g. rule), from rule predicates to rule body. The checking process includes two steps. Firstly getting an information channel, and then checking if the labeled entities can flow through this channel. If the available channel is narrow, it means the access control is more restricted. Each label has its applicable scope, if a role is labeled in the definition part, the label will affect all the role scope. If a role is labeled in a rule, the label only affects within this rule.

To illustrate our approach, we specify authorization policy using the Flexible Authorization Framework (FAF) [1] that is a logic-based framework in specifying authorizations in the form of rules. The access control policies are modeled through rules which are expressed by access predicates. In FAF, if p is a predicate with arity n, and t_1, …, t_n are *terms* appropriate for p, then $p(t_1, \ldots, t_n)$ is an *atom*. An atom is denoted with word *literal*. For example, auth(s, o, $+a$) is a literal, s, o and a are terms. A rule can also be a predicate of another rule. An authorization rule can be defined as the form:

$$auth(s, o, \langle sign \rangle a) \leftarrow P_1 \,\&\, \ldots \,\&\, P_n$$

Where s, o, and a are elements of subject, object and action respectively, $n \geq 0$, <*sign*> is either + or -, and $P_1, \ldots, P_n$ are the predicates. In this paper, if a literal is a rule predicate, we call it rule predicate, its terms are called rule predicate elements, otherwise it is called rule, and its term is call rule element. Some authorization rules described in FAF are:

$in(Mary, Staff, ASH) \leftarrow$
$in(John, Manager, ASH) \leftarrow$
$in(file1, Document, AOH) \leftarrow$
$in(file2, Document, AOH) \leftarrow$
$cando(s, f, +read) \leftarrow in(s, Staff, ASH) \,\&\, in(f, Document, AOH)$
$cando(s, f, +write) \leftarrow in(s, Manager, ASH) \,\&\, in(f, Document, AOH)$

The first four rules consist of information about subject and object hierarchies, and the next two rules describe how accesses propagate along these hierarchies. *ASH* and *AOH* denote authorization subject and object hierarchies respectively. *ASH* consists of two roles *Staff* and *Manager*. Mary has a role of Staff, and John has a role of Manager. The object hierarchy *AOH* has one class *Document* with members of *file1* and *file2*. The two *cando* rules specify that all employees with a role Staff are allowed to read files and only managers are allowed to write these files.

3.1. Label specification for literal elements

The rule elements and rule predicate elements can all be assigned with labels. An example of rule element label assignment is as follow:

$$cando(user, f^{L_2}, +read) \leftarrow in(user^{L_1}, Staff, ASH) \,\&\, in(f, Document, AOH)$$

L_1 is a label assigned to the rule predicate element *user*. If $lo(L_1)$={"School of Engineering"}, and $li(L_1)$={ "School of Engineering", "Computer Science Department"}, it means only the users from "School of Engineering" and "Computer Science Department" are the

Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'06)
0-7695-2611-X/06 $20.00 © 2006 IEEE

valid users for this rule predicate. L_2 is a label assigned to the rule element *f*. If $lo(L_2)$={ "School of Engineering"}, and $le(L_2)$={ "School of Engineering", "Computer Science Department.Security Group"}, it means the object *f* can only be used in the "School of Engineering" and "Security Group" of Computer Science Department. These information flow constraint for this variable will automatically propagate to the whole rule scope. If there is more than one label assigned to the same variable in different literals, the information flow channel of this variable is the intersection of these labels. According to this definition, the label information flow control for the next three rules is same.

$$cando(user, f, +read) \leftarrow in(user^{L_1}, Staff, ASH) \& in(f, Document, AOH)$$
$$cando(user^{L_1}, f, +read) \leftarrow in(user, Staff, ASH) \& in(f, Document, AOH)$$
$$cando(user^{L_1}, f, +read) \leftarrow in(user^{L_1}, Staff, ASH) \& in(f, Document, AOH)$$

If an object or action labeled, for example in $cando(user, f^{L_2}, +read)$, the constraint of L_2 will automatically propagate to the action, i.e. the +*read*. If both of them are labeled, then the valid information flow is their intersection. This rule also suits the subject and object hierarchy, for example, the *Staff* and *ASH* in the above example. The information flow check among the elements in a literal is if the information can flow from the subject to other elements, for example, if the subject *s* can flow to the permission (*o*, <*sign*>*a*), or to the role(*r*, *ASH*). For a literal $cando(user^{L_1}, f^{L_2}, +read)$, the system will check if the information can flow from L_1 to L_2.

3.2. Label specification for literals

A literal can be assigned with labels, for example,

$$in^{L_2}(user^{L_1}, Staff, ASH)$$

In this case, the information flow check is if the information can flow from L_1 to L_2. A valid information flow should satisfy the checking process of

$$information\_channel(L_1, L_2) \neq \varnothing \wedge context(user) \subseteq input\_channel(L_1, L_2)$$

$context(user)$ is a function to get a user's context. If his context does belong to the information input channel, then this rule is not valid for him. There is no information flow check among the labels assigned to predicate literals. If both rule body and predicate literals are labeled, the information flow check is if the information can flow from the labels assigned to predicate literals to the labels assigned to the target literal. For example, a rule as follow:

$$cando^{L_2}(user, f, +read) \leftarrow in(user, Staff, ASH) \& in^{L_1}(f, Document, AOH)$$

The system needs to check if information can flow from L_1 to L_2. If the information flow is not permitted, it means the predicate literal is not valid, and further this rule is not a valid rule.

3.3. Label specification for rules

The whole rule can be assigned with labels. In this case, the information flow check is if the information can flow from its inside labels to the rule labels. For example,

$$\left(cando^{L_1}(user, f, +read) \leftarrow in(user, Staff, ASH) \& in(f, Document, AOH)\right)^{L_2}$$

The system needs to check if information can flow from L_1 to L_2. If the information flow is not permitted, it means this rule is not valid.

3.4. Label specification for policies

A policy can be assigned with labels. In this case the information flow check is if the information can flow form its inside labels to the policy labels. For example,

$$policy^{L_2} = \{rule_1^{L_1}, rule_2, ..., rule_n\}$$

The system needs to check if information can flow from L_1 to L_2. If the information flow is not permitted, it means this policy is not valid.

3.5. Label policy checking process

The label policy checking process can be from top to bottom, i.e. from policy labels to literal element labels, or from bottom to top, i.e. from literal element labels to policy labels. In any checking route, the available information flow channel will become narrower, in other words, become stricter. At runtime the information flow check can be done before or after an access control rule check, adopting which way will depend on system implementation.

4. Label policy for access control and policy management

In this section we will discussion how the LBACP can be used both for access control and policy management through some examples. Compare to the system-oriented normal access control policy the label-based access control policy is more application-oriented. The label policy can be used by both system administrators and normal application users, this is because the label policies can be assigned to access control policy components at different levels, for example, assign a labels to rule elements or to rules. With the help of analyze tools, access control policies can be viewed based assigned label policies.

Consider an example of a RBAC system in an IT company. There are two departments, one is "Development department", and another is "Test department". The "Project 1" is in the developing stage,

Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'06)
0-7695-2611-X/06 $20.00 © 2006 IEEE

and needs to be assigned to "Development department"; "Project 2" is in the testing stage, and needs to be assigned to "Test department". We assume the system administrator defines a rule that allows the employees with "Software Engineer" role can access all the developing projects and testing projects, and all the employees in both departments have this role. Then he defines two labels, one permits information flowing to "Development department", and assigns it to the "Project 1"; the other permits information flowing to "Test department", and assigns it to "Project 2". At runtime, the access control engine will check the RBAC policies and the label policies, and then decides which project should be accessed by the users form which departments. When "Project 1" is finished, and needs to be tested, it can be relabeled so that only the user with "Software Engineer" in "Test Department" can access it. If it is necessary, this project can be relabeled back to "Development Department". This labeling process is obviously easier than changing the access control policies and also less error prone. The label policy can also reduce the number of role definition in a RBAC system, for example, we do not need to create a "Test engineer" role for "Test department" in this example.

Structuring introduced by labels allows administrators to work with a smaller number of policy components. For example, consider a hospital composed of three departments, namely, "Radiology", "Surgery", and "Medicine". Each of the departments is responsible for granting access to data under their (possibly overlapping) authority domains, where domains are specified by a scoping restriction, i.e. labels. With the help of label polices, the administrators can get a clear static view about the subjects, objects and actions relationships in a department. This function can also be implemented through define many special roles for different departments and assign permission to these roles, but define too many roles will bring out management problems, and these roles still need to be grouped in order to get a management view about each department. Through the roles indirectly managing the resources and users sometimes cannot easily give a clear view about which resource can be shared with other departments.

By annotating policy components with labels we can form policy component groups for the purposes of access control and policy management. Policy components may be marked with multi labels that do not have any relation (through label separation), thus allowing administrators to group policy components according to various dimensions (applications). This allows different views to a policy, and this can be exploited in policy visualization. The label policies can segregate of unrelated policy components, for example, only some rules labeled with a specified label are used for make an access control decision.

Virtual Organization (VO) is a dynamic collection of resources and users unified by a common goal and potentially spanning multiple administrative domains [2]. VO may apply some common policies about how its users access the resources assigned to the VO, but each organization will typically retain ultimate control over the policies that govern access to its resources. The dynamic and multi-institutional nature of these environments introduces challenging security issues that demand new technical approaches. Since VO resources and users are located within multiple organizations, a key problem associated with the formation and operation of distributed virtual organizations is how to manage and enforce the common and local policies. One VO many involves many participant organizations, and one organization may participate many different VOs. We will describe how these issues are addressed by our label-based access control policy. In our label-based access control policy solution, the VO will define a set of roles contractually agreed by the participant organizations. The common VO policies are RBAC policies, but do not specify any domain related information. The domain related information is described in the label policies, for example, which roles can be used in which participant organizations. For the resources providers, they can use the label policies to define which resources are provided to which VO, and which roles are permitted to access its resources.

5. Related works

There are two important related works. One is the decentralized label model developed by Myers et al. [3, 4], the other is an OASIS RBAC meta-policy approach for subdividing the administration of large-scale security environments and for enforcing information flow restrictions over policies presented by Belokosztolszki et al [5, 6].

Decentralized label model is a model of information flow control that protects private data while allowing the applications to share data. In their approach the label model is decentralized: it allows cooperative computation by mutually distrusting principals, without mediation by highly trusted agents. The decentralized label model permits programs using it to be checked statically to avoid information leaks. This model also presents a new language JFlow that is an extension to the Java programming language. Variable declarations in JFlow programs are annotated with labels that allow the static checker to check programs for information leaks efficiently, in a manner similar to type checking.

There are two major differences between the decentralized label model and LBACP. The first difference is the label structure. In decentralized label

model the system does not care where the information from, so the label only includes the owner and reader information. Whereas in LBACP it is important to check the information sources, so the label structure includes the information sources specification. The second difference is in decentralized label model labels are assigned to variables of Java programs in order to protect the confidentiality and integrity of information manipulated by the computing system. In LBACP labels are assigned to the access control policy components in order to confine their usages, it specifies high level access control strategy and program languages independent.

The OASIS RBAC meta-policy is an approach to control information flow in and out of the OASIS RBAC system. This approach introduces the concept of "contexts" to group and classify policy components according to various aspects. These contexts are applied to control information flows between system entities. With the help of information flow relation administrators can restrict the use of policy components alongside components belonging to certain other groups, and organize access control policies into a hierarchical, multidimensional structure.

There are two major differences between OASIS RBAC meta-policy and LBACP. In OASIS RBAC meta-policy there is no real label structure, it simply assigns the context names to the policy components, this directly assignment makes it difficult to provide flexible policy management in multidimensional way as they hope. Whereas in LBACP, there is a clear label structure specifying the owner, import and export contexts information, and one component can be assigned with multiple labels. This design provides a real flexible way for policy management in different levels and different ways. The second difference is the OASIS RBAC meta-policies are not directly used for access control but describe and restrict policies at policy specification time. On the contrary, the LBACP is tightly connects to the access control policies, it must be checked before or after access control policies checking process. The LBACP specifically address the issue of information flow relations among the entities in access control policies. It can be seen layered on the top of the normal access control policies. The major benefit of this layer structure is it easy to deal with the situation of dynamic change relationships in the collaboration computing environment.

6. Conclusion and future work

With the LBACP we can get three major benefits. The first is in the collaboration environment adding new partners can be done through redefining the label information flow constraints instead of changing access control policies. This reduces the error prone, and suits the requirements of modern dynamic collaboration activities. The second is through defining information flow restrictions it can avoid a potential information leak made by access control policy definition errors. People only need to manage the information flow with other organizations. The third is it can improve the access control policy administration through policy components group function. Its layer structure also makes it easy to integrate with exist access control systems. The future work is mainly related to investigating the hierarchical label policy specification.

7. Acknowledgement

The authors would like to thank the One North East RDA for their financial support for B2B Manufacturing Centre. This centre is in collaboration between the Warwick Manufacturing Group at the University of Warwick and the University of Teesside. http://www.b2b-mc.co.uk.

8. References

[1] S. Jajodia, P. Samarati, M. L. Sapino, V. S. Subrahmanian, Flexible support for multiple access control policies, ACM Transactions on Database Systems (TODS), Volume 26, Issue 2 (June 2001) pp. 214 – 260.

[2] L. Pearlman, C. Kesselman, V. Welch, I. Foster, S. Tuecke, The Community Authorization Service: Status and Future, http://www.globus.org/security/CAS/GT3/.

[3] A. C. Myers and B. Liskov, A decentralized model for information flow control, ACM Symposium on Operating Systems Principles, in Proceedings of the sixteenth ACM symposium on Operating systems principles table of contents, Saint Malo, France, 1997, pp. 129 - 142.

[4] A. C. Myers and B. Liskov, Complete, safe information flow with decentralized labels, in Proceedings of IEEE Symposium on Security and Privacy, Oakland, CA, USA, May 1998, pp. 186–197.

[5] A. Belokosztolszki, D. M. Eyers, and K. Moody, Policy contexts: Controlling information flow in parameterised RBAC, in Proceedings of Policy 2003: IEEE Fourth International Workshop on Policies for Distributed Systems and Networks, 2003, pp. 99–110.

[6] A. Belokosztolszki, K. Moody,D. M. Eyers, A formal model for hierarchical policy contexts, in Proceedings of Policy 2004: IEEE Fifth International Workshop on Policies for Distributed Systems and Networks, 2004, pp. 127-136.