# Policy Conflict Analysis Using Tableaux for On Demand VPN Framework

Hiroaki Kamoda, Akihiro Hayakawa, Masaki Yamaoka, Shigeyuki Matsuda
NTT DATA CORPORATION, Japan.
{kamodah, hayakawaak, yamaokam, matsudasg}@nttdata.co.jp

Krysia Broda, Morris Sloman
Department of Computing, Imperial College London, UK.
{kb, mss}@doc.ic.ac.uk

## Abstract

*The medical field has a requirement for ubiquitous computing with secure and reliable access control to permit patient information to be logged as they go about their normal activities or to permit medics to remotely access patient information from various mobile devices. Healthcare involves many different people from multiple organizations - general practitioner, hospital doctor or nurse, social workers who all need different information. Defining the required authorization policies can be very complex, resulting in conflicts, which could result in information leaks with privacy implications or prevent access to information needed. In this paper we propose an approach for detecting conflicts defined in an authorization policy by using free variable tableaux. Our method enables us not only to statically detect a conflicting policy but also to give us information that would be helpful to correct the policy by using abductive inference.*

## 1. Introduction

The availability of on-body or implantable healthcare monitoring devices, mobile PDA/phones and ad-hoc network technology will result in ubiquitous computing being widely used for healthcare. Security technologies such as access control, mutual authentication and encryption are also needed to prevent access of private medical data by unauthorized people. Techniques are needed that allow doctors or emergency paramedics to securely and rapidly access appropriate patient information such as medical records or X-rays using portable devices from remote sites. An at-risk patient could have a body area network monitoring cardiac information and patient's context which is logged with a remote server. A medical ubiquitous computing infrastructure must support security to prevent information leak but should never forbid access to information that is required in an emergency. Also, to exchange confidential information efficiently over the Internet, it is required that users can establish a VPN only as needed (on demand). We are trying to establish an On Demand VPN Framework (ODV)[7] to realize these requirements for a wireless network to support ubiquitous healthcare.

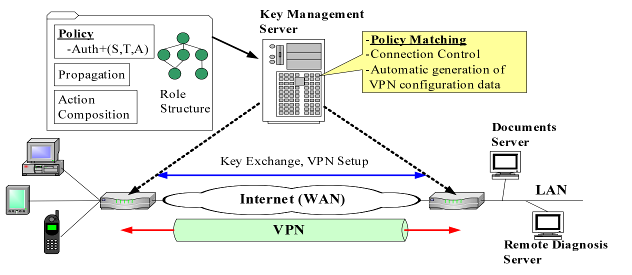


**Figure 1. On Demand VPN Framework**

In the ODV, the decision whether or not a user can connect a VPN between medical devices or institutions is based on an authorization policy which must be flexible enough to cope with changes of network topology and the complex interactions between multiple organizations involved in healthcare. Notations such as Ponder[2] and XACML[8] can be used by system administrators to define authorization policies. However the overall structure of these policies may become very complex, reflecting the complexity of the organizations and roles involved. There is an increased risk that an administrator mistakenly defines conflicting policies which results in information leak or prevents access to critical patient information in an emergency situation.

This paper proposes a new static method for detecting policy conflicts, based on free variable tableaux, which has the advantage that it can give helpful information to resolve conflicts by using abductive inference. The paper is organized as follows: Section 2 introduces the ODV; Section 3 presents an outline of conflict detection using free variable tableaux; in Section 4 we illustrate the method to detect and abduce conflicting policies through examples. In Section 5 we describe some related work, and conclusions and future work are presented in Section 6.

## 2. On Demand VPN Framework (ODV)

In this section, we describe some features of the ODV and the required policies. We also discuss the implications of policy propagation with respect to role relationships and composite actions.

### 2.1. On Demand VPN Outline

Fig.1 is a simplified version of the ODV framework model. The features of an ODV service that are relevant to this paper are summarized below.

1. When a user requests the management server to establish a VPN the user's authentication is checked and the authorization policy is checked to see whether the request should be granted or not.
2. If it is granted, then the VPN configuration information and encryption key information are distributed securely to the VPN components, automatically completing setup of a multipoint VPN.
3. The required information such as encryption keys is stored in a microchip designed to prevent illegal copying or tampering.

Fine grained access control is needed in healthcare environments to distinguish between access to medical records, administrative records and applications such as remote diagnosis. Traditional VPN based access control only operates at the network layer and so our ODV supports more sophisticated application access control using authorization policies. These are defined in terms of subject and target role structures. Policies propagate up or down the role structures as explained in section 2.2.1. Authorization policies may be defined in terms of composite actions which require a number of sub-actions which can also result in conflicts if separate policies are defined for the sub-actions as explained below.

### 2.2. Features of the Policy Used in ODV

**2.2.1. Propagation Scheme** Policies in the ODV are defined by using a *role* that is a named collection of privileges [3]. Individual subjects and targets take on assigned roles. A partial order relation is defined among these roles. The

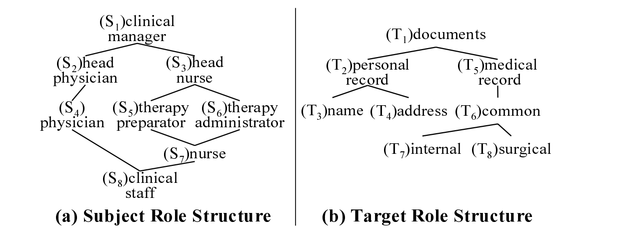


**Figure 2. Examples of Role Structures**

graph representation of the relation is called a *role structure*. In particular, the role structure corresponding to subjects is called a *subject role structure* (SRS) and that corresponding to targets is called a *target role structure* (TRS). Examples of these role structures are shown in Fig.2, where (a) is a simplification of the subject role structure presented in [10] and (b) is an example of a target role structure in the documents server as shown in Fig.1. The role structures potentially simplify policy specification by propagating policies. In general, if a certain role $r$ is allowed to perform a particular action, then roles higher than $r$ should also be allowed to perform the action. Conversely, if roles higher than $r$ are not permitted to perform an action, then $r$ should not be permitted to perform the action. For example in Fig.2(a) the privileges associated with a `physician` propagate upwards to `head physician` roles. If access is permitted to a target role $T_2$ it propagates to target roles $T_3$ and $T_4$ in Fig.2(b). However, this policy propagation can result in unforeseen conflicts. Note that the concept of the role structure is slightly different from the role hierarchies defined in the standard role based access control model [3] in that the propagation is explicitly defined by a *propagation scheme*, rather than being implicit, which gives added flexibility. The direction of the propagation may differ according to the type of policy or the type of role structures. We thus define an explicit policy propagation scheme for each role structure which is more flexible than the implicit propagation in standard role hierarchies.

The following are example policies for roles in Fig.2:

Policy r1 : $\mathtt{Auth+}(\mathtt{S_8}, \mathtt{T_5}, \mathtt{read})$
Policy r2 : $\mathtt{Auth-}(\mathtt{S_2}, \mathtt{T_5}, \mathtt{read})$

Generally, an authorization policy (`Auth+`) defines the action that a subject role is *permitted* to perform on a target role and a negative authorization policy (`Auth-`) defines the action that a subject role is *forbidden* to perform on a target role. policy r1 specifies that the subject role `clinical staff` is allowed to read documents `medical record` and policy r2 specifies that the subject role `head physician` is forbidden to read documents `medical record`. The policies r1 and r2, appear to define authorizations for different subject roles so there should be no problems. However, if these policies are compared from the viewpoint of the role structure, then a conflict occurs. In this example, as shown in Fig.2(a), `head physician` is a higher position than `clinical staff`. Therefore, we can regard that policy r2 also implicitly defines that subject roles named `physician` and `clinical staff` are not allowed to read documents named `medical record`. As a result, this causes a conflict because the subject role named `clinical staff` has opposite permissions.

**2.2.2. Action Composition Scheme** Policy may be defined in terms of many actions, not just read and write. Example policies are:

Policy r3 : $\mathtt{Auth}+(\mathtt{S}_4, \mathtt{T}, \mathtt{rm\_dgn})$
Policy r4 : $\mathtt{Auth}-(\mathtt{S}_4, \mathtt{T}, \mathtt{tv\_conf})$
Policy r5 : $\mathtt{Auth}-(\mathtt{S}_4, \mathtt{T}, \mathtt{view\_record})$

`rm_dgn`, `tv_conf` and `view_record` mean, respectively, to perform a remote diagnosis, to perform a TV conference and to view a medical record, and $T$ indicates a target application in the remote diagnosis server as shown in Fig.1 At first, comparing the three policies r3, r4 and r5, no problems are detected. However, `rm_dgn` is a *composite action*, defined as follows:

$\mathtt{rm\_dgn} = \mathtt{tv\_conf} \wedge \mathtt{view\_record}$

This specifies that two actions `tv_conf` and `view_record` are performed in the system when performing a remote diagnosis on $T$. Then r3, r4 and r5 become conflicting policies, as policy r3 specifies `physician` is allowed to perform an action `rm_dgn`, while the other two policies specify that both actions `tv_conf` and `view_record` are prohibited. In this way an action composition may also lead to policy conflicts.

## 2.3. Issues of the ODV

As described in Section 2.2.1 and 2.2.2, conflicting policies can result from propagation and action composition, which cannot be detected by simply comparing authorization policies. We call this type of conflict *implicit conflict*. The problem is that the more complex the role structure and the action composition become, the more difficult it becomes to detect an implicit conflict. The information exchanged in the medical applications usually contains very sensitive data. Information leak caused by an incorrect policy should never be allowed. In a medical emergency, prevention of access to information resulting from an undetected conflict can have life-threatening consequences. This means runtime conflict detection methods are not suitable. Therefore, we need to develop a method to analyze a conflict statically before starting a system and also provide information to resolve the conflict. In the rest of this paper we present our approach, which is based on free variable tableaux, to satisfy these demands.

## 3. Free Variable Tableaux

In this section we describe an outline of the conflict detection method based on *free variable tableaux*[4].

### 3.1. Outline of the Approach

It is possible to enumerate all policies derived implicitly by propagation and action composition schemes and then detect an implicit conflict by comparing original policies and derived policies. However this would be computation intensive and it is hard to identify the original policies that cause a conflict. The Free Variable Tableaux (FVT) method allows faster detection of a conflict and also infers the cause of the conflict. It is a sound and complete theorem prover that can be used to show inconsistency and upon which can be built abductive reasoning. Moreover, it has optimized implementations.

The following two steps are needed to detect a conflict using FVT: i) each policy is translated into a logical sentence; ii) the FVT method is applied to these sentences to detect any possible conflicts, by detecting inconsistency, and to abduce the information that shows the cause of the conflict. In other words, all we have to do is to define the following translation mapping $\zeta$ from policies to logical sentences, such that conflicting policies become inconsistent sentences in logic.

$$\begin{array}{rccc} \zeta : & \mathcal{P} & \rightarrow & \mathcal{L} \\ & \Uparrow & & \Uparrow \\ & r & \mapsto & \zeta(r) \end{array}$$

where $\mathcal{P}$ is a set of policies and $\mathcal{L}$ is a set of sentences. Once policies have been translated into logic, a conflicting policy is detected in the same way in spite of the differences of the original policy descriptions. This means that our approach can easily be applied to various kinds of policy definition languages.

### 3.2. Formalization of Policies

In this section we show the mapping $\zeta$ for some policies. In addition to the authorization policy described in Section 2, an obligation policy [2] can be defined in the ODV. An obligation policy (`Obli+`) defines the action that a subject role *must* perform on a target role when an event occurs. A negative obligation policy (`Obli-`) defines the action that a subject role *must not* perform on a target role when an event occurs. The mapping $\zeta$ for the `Auth±` and `Obli±` is defined as follows.

$$\boxed{\begin{array}{rl} \zeta(\mathtt{Auth+}(S_1,\ T_1,\ A_1)) & := \forall x(E_x \rightarrow P(S_1, T_1, A_1)) \\ \zeta(\mathtt{Auth-}(S_1,\ T_1,\ A_1)) & := \forall x(E_x \rightarrow \neg P(S_1, T_1, A_1)) \\ \zeta(\mathtt{Obli+}(E_1, S_1, T_1, A_1)) & := E_1 \rightarrow O(S_1, T_1, A_1) \\ \zeta(\mathtt{Obli-}(E_1, S_1, T_1, A_1)) & := E_1 \rightarrow F(S_1, T_1, A_1) \end{array}}$$

In the above translations, the predicate $P$ can be read as "subject role $S_1$ is permitted to carry out action $A_1$ on target role $T_1$" and predicate $O$ as "subject role $S_1$ must carry out action $A_1$ on target role $T_1$" and predicate $F$ as "must not carry out". The atom $E_x$ says that event $E_x$ occurs. Then, for example, the second and third translations can be read, respectively, as "for any event $E_x$, $S_1$ is forbidden to carry out $A_1$ on $T_1$" and "if event $E_1$ occurs then $S_1$ must carry out action $A_1$ on target role $T_1$". There needs to be two axioms that relate $P$, $O$ and $F$ *i.e.* an obligation policy requires an authorization policy to permit the action and it contradicts a negative obligation policy:

Ax1 :$\forall s, t, a(O(s,t,a) \rightarrow P(s,t,a))$

Ax2 :$\forall s, t, a(\neg(O(s,t,a) \wedge F(s,t,a)))$

Ax1 is used to detect conflicts involving both authorization and obligation policies together.

## 4. Conflict Detection

In this section the definition of $\zeta$ is presented by using some example policies for the ODV. We also show that our approach can detect a conflict and abduce the cause.

### 4.1. Conflict Caused by Propagation

We show that policy r1 and r2 described in Section 2.2.1 become a conflict due to the propagation scheme. Policies r1 and r2 are translated into the following sentences by using the definitions described in Section 3.

$\zeta(\text{Policy r1}) = \forall x(E_x \rightarrow P(S_8, T_5, A_1))$

$\zeta(\text{Policy r2}) = \forall x(E_x \rightarrow \neg P(S_2, T_5, A_1))$

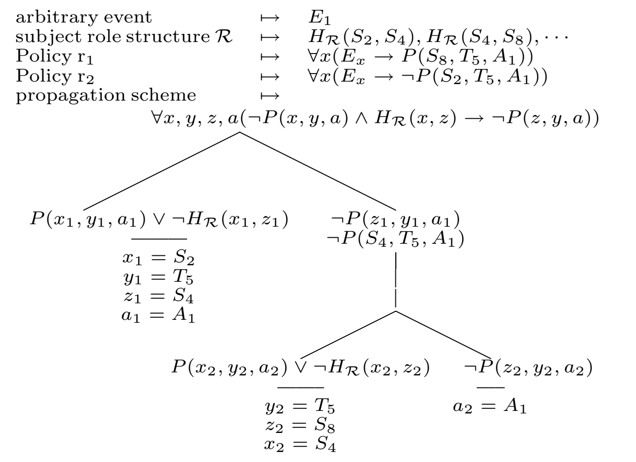
where $A_1$ stands for `read`.

A propagation scheme defines how an authorization policy propagates in accordance with the partial order. In this case, `Auth+` policy propagates upwards and `Auth-` policy propagates downwards. These propagation schemes can both be translated into:

$\forall x, y, z, a(\neg P(x,y,a) \wedge H_{\mathcal{R}}(x,z) \rightarrow \neg P(z,y,a))$

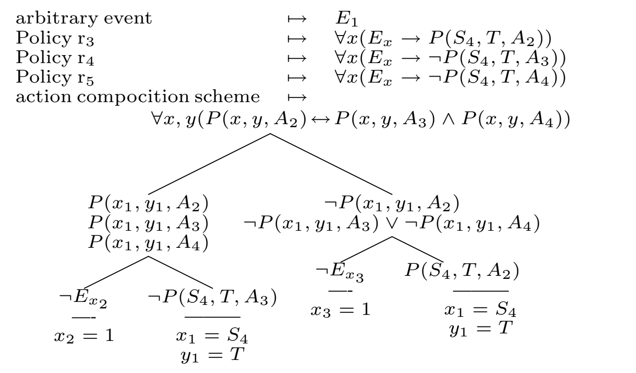
where $H_{\mathcal{R}}(i,j)$ is a predicate such that $i \in \mathcal{R}$ is a parent node of $j \in \mathcal{R}$, and $\mathcal{R}$ stands for the subject role structure described in Fig.2(a).

The result of analyzing policies r1 and r2 using the FVT method is shown in Fig.3. To simplify the diagram some details are omitted. Since a conflict only happens if an event occurs, we assume an arbitrary event $E_1$ occurs. The tableau is developed by analyzing the data, starting from the premise that the data is not conflicting and deriving contradictions in each branch of the analysis yielding the conclusion that the initial premise was false. Each time a universal sentence is analyzed new variables are used, which receive values in order to obtain a contradiction in a branch. For example, in the first branch, if the variables $\{x_1,\ y_1,\ z_1,\ a_1\}$ are given the values $\{S_2,\ T_5,\ S_4,\ A_1\}$, then the branch



arbitrary event $\mapsto$ $E_1$
subject role structure $\mathcal{R}$ $\mapsto$ $H_{\mathcal{R}}(S_2,S_4), H_{\mathcal{R}}(S_4,S_8), \cdots$
Policy $r_1$ $\mapsto$ $\forall x(E_x \rightarrow P(S_8,T_5,A_1))$
Policy $r_2$ $\mapsto$ $\forall x(E_x \rightarrow \neg P(S_2,T_5,A_1))$
propagation scheme $\mapsto$
$\forall x,y,z,a(\neg P(x,y,a) \wedge H_{\mathcal{R}}(x,z) \rightarrow \neg P(z,y,a))$
$P(x_1,y_1,a_1) \vee \neg H_{\mathcal{R}}(x_1,z_1)$
$\neg P(z_1,y_1,a_1)$
$\neg P(S_4,T_5,A_1)$
$x_1 = S_2$
$y_1 = T_5$
$z_1 = S_4$
$a_1 = A_1$
$P(x_2,y_2,a_2) \vee \neg H_{\mathcal{R}}(x_2,z_2)$
$\neg P(z_2,y_2,a_2)$
$y_2 = T_5$
$z_2 = S_8$
$x_2 = S_4$
$a_2 = A_1$

**Figure 3. Conflict Caused by Propagation**



arbitrary event $\mapsto$ $E_1$
Policy $r_3$ $\mapsto$ $\forall x(E_x \rightarrow P(S_4,T,A_2))$
Policy $r_4$ $\mapsto$ $\forall x(E_x \rightarrow \neg P(S_4,T,A_3))$
Policy $r_5$ $\mapsto$ $\forall x(E_x \rightarrow \neg P(S_4,T,A_4))$
action compocition scheme $\mapsto$
$\forall x,y(P(x,y,A_2) \leftrightarrow P(x,y,A_3) \wedge P(x,y,A_4))$
$P(x_1,y_1,A_2)$
$P(x_1,y_1,A_3)$
$P(x_1,y_1,A_4)$
$\neg P(x_1,y_1,A_2)$
$\neg P(x_1,y_1,A_3) \vee \neg P(x_1,y_1,A_4)$
$\neg E_{x_2}$
$x_2 = 1$
$\neg P(S_4,T,A_3)$
$x_1 = S_4$
$y_1 = T$
$\neg E_{x_3}$
$x_3 = 1$
$P(S_4,T,A_2)$
$x_1 = S_4$
$y_1 = T$

**Figure 4. Conflict Caused by Action Cmp.**

contradicts with the assumption $H_{\mathcal{R}}(S_2,S_4)$ and policy r2. From the tableau we deduce that these policies conflict with each other and that the conflict is caused by the propagation $\{S_2, S_4, S_8\}$.

### 4.2. Conflict Caused by Action Composition

Next we show that policies r3, r4 and r5 described in Section 2.2.2 become conflicting due to an action composition scheme. These policies are translated by using the definitions described in Section 3:

$\zeta(\text{Policy r3}) = \forall x(E_x \rightarrow P(S_4, T, A_2))$

$\zeta(\text{Policy r4}) = \forall x(E_x \rightarrow \neg P(S_4, T, A_3))$

$\zeta(\text{Policy r5}) = \forall x(E_x \rightarrow \neg P(S_4, T, A_4))$

where $A_2$, $A_3$, $A_4$ stand for `rm_dgn`, `tv_conf` and `view_record` respectively.

An action composition scheme defines the relationship among actions occurring in an ODV system. The syntax of the action composition is defined from $n$ actions $A_1, \cdots, A_n$ by $A_1 = \Gamma(A_2, \cdots, A_n)$, where $\Gamma$ is a propo-

sitional sentence using $A_2, \cdots, A_n$. The mapping $\zeta$ for the action composition policy is defined as follows.
$\zeta(A_1 = \Gamma(A_2, \cdots, A_n))$
$:= \forall x, y(P(x, y, A_1) \leftrightarrow \Gamma(P(x, y, A_2), \cdots, P(x, y, A_n)))$

For example, the following action composition scheme described in Section 2,

`rm_dgn` $=$ `tv_conf` $\wedge$ `view_record`

is translated as follows.

$\forall x, y(P(x, y, A_2) \leftrightarrow P(x, y, A_3) \wedge P(x, y, A_4))$

The result of analyzing these policies using FVT is shown in Fig.4. To simplify the diagram some details are omitted. We can recognize that these policies conflict with each other since all branches are contradictory.

## 5. Related Work

There are several approaches to detect and resolve conflicting policies statically. Lupu et al. [6] discusses how conflicts may occur if subject and target domains overlap, and proposes a method to resolve the conflict by policies for more specific domains having priority. This approach thus uses an implicit propagation scheme defined by the domain structure and does not deal with composite actions.

Other approaches mention conflicts that occur due to the hierarchical structure of the underlying organization and the associated propagation schemes. Several methods using precedence to resolve conflicts are proposed. For example, S. Jajodia et al.[5] propose to resolve conflicts by using default rules such as deny override. However, this approach may not result in the correct resolution; for example in a medical situation, sometimes permit override is needed in case of emergency. Therefore, conflicting rules should be detected and notified to the administrator who needs to specify a method of resolving them or use an application specific precedence policy. Some methods only perform conflict detection, such as [1] and [9], which present a logical and static method to detect some inconsistent rules, whilst [5] describes a method to detect conflicts using derivation rules. However, these approaches do not provide information about the cause of the conflict.

## 6. Conclusion and Future Work

In this paper we have presented an approach to statically detect policy conflicts using free variable tableaux by translating each access control policy into logic. As the tableaux method is sound and complete, it is guaranteed that all conflicting policies can be detected if the logical translation is correct. We have also ensured the usability of the approach by showing how the conflicting policy can be detected and the conflicting information, helpful to correct the policy, may be abduced. Moreover, it has the advantage that it can be applied to various policies written in different policy definition languages. Free variable tableaux can be easily implemented *e.g.* using Prolog, so the approach can be integrated into a policy specification toolkit to automate the analysis.

Future work will include proving the validity of logical translations and evaluating the computational complexity of the method. Our approach uses a fragment of first order logic – in particular we have not needed functors except in ground atoms and so we anticipate that the complexity would be less than NP-complete. We still have to integrate the system with the ODV to support automated analysis and we will investigate automated conflict resolution.

## 7. Acknowledgement

This research was conducted with funding from the Ministry of Internal Affairs and Communications of Japan. We would like to thank Alessandra Russo, Naranker Dulay, Emil Lupu and Arosha Bandara for their many helpful comments and suggestions.

## References

[1] L. Cholvy and F. Cuppens. Analyzing Consistency of Security Policies. In Proc. 1997 IEEE Symposium on Security and Privacy, pp. 103–112. IEEE Computer Society, 1997.

[2] N. Damianou, N. Dulay, E. Lupu, M. Sloman. The Ponder Policy Specification Language. In *Proc. Policy 2001: Workshop on Policies for Distributed Systems and Networks, Bristol, U.K.*, pp. 18–39. Springer-Verlag LNCS 1995, Jan. 2001.

[3] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security*, 4(3):224–274, August 2001.

[4] M. Fitting. *First Order Logic and Automated Theorem Proving*. Springer, second edition, 1996.

[5] S. Jajodia, P. Samarati, and V. S. Subrahmanian. A Logical Language for Expressing Authorizations. In *Proc. 1997 IEEE Symposium on Security and Privacy, Oakland, CA, USA*, pp. 31–42.

[6] E. C. Lupu and M. Sloman. Conflicts in Policy-Based Distributed Systems Management. *IEEE Trans. on Software Eng.*, 25(6):852–869, Nov. 1999.

[7] NTT DATA CORPORATION. Toward On-Demand VPN Communications. *Press Release*, September 2004. http://www.nttdata.co.jp/en/media/2004/093000.html.

[8] OASIS. eXtensible Access Control Markup Language (XACML) Version 1.1. *OASIS Standard*, July 2003.

[9] C. Ribeiro, A. Zúquete, P. Ferreira, and P. Guedes. Security Policy Consistency. *Technical Report, INESC*, June 2000.

[10] M. Wilikens, S. Feriti, A. Sanna, M. Masera. A Context-related Authorization and Access Control Method Based on RBAC. In *SACMAT '02: Proc. 7th ACM Symp. on Access Control Models and Technologies*, pp. 117–124, Dec. 2002.