Semantic Context Aware Security Policy Deployment

Stere Preda
IT TELECOM Bretagne
CS 17607, 35576
Césson-Sévigné, France
stere.preda@telecom-bretagne.eu

Frédéric Cuppens
IT TELECOM Bretagne
CS 17607, 35576
Césson-Sévigné, France
frederic.cuppens@telecom-bretagne.eu

Nora Cuppens-Boulahia
IT TELECOM Bretagne
CS 17607, 35576
Césson-Sévigné, France
nora.cuppens@telecom-bretagne.eu

Joaquin G. Alfaro
School of Computer Science
Carleton University, K1S 5B6
Ottawa, Ontario, Canada
joaquin.garcia-alfaro@acm.org

Laurent Toutain
IT TELECOM Bretagne
CS 17607, 35576
Césson-Sévigné, France
laurent.toutain@telecom-bretagne.eu

Yehia Elrakaiby
IT TELECOM Bretagne
CS 17607, 35576
Cesson-Sevigne, France
yehia.elrakaiby@telecom-bretagne.eu

ABSTRACT

The successful deployment of a security policy is closely related not only to the complexity of the security requirements but also to the capabilities/functionalities of the security devices. The complexity of the security requirements is additionally increased when contextual constraints are taken into account. Such situations appear when addressing the dynamism of some security requirements or when searching a finer granularity for the security rules. The context denotes those specific conditions in which the security requirements are to be met. (Re)deploying a contextual security policy depends on the security device functionalities: either (1) the devices include all functionalities necessary to deal with a context and the policy is consequently deployed for ensuring its automatic changes or (2) the devices do not have the right functionalities to entirely interpret a contextual requirement. We present a solution to cope with this issue: the (re)deployment of access control policies in a system that lacks the necessary functionalities to deal with contexts.

Categories and Subject Descriptors

D.4.6 [**Security and Protection**]: Access controls; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Security, Theory.

Keywords

Access Control; PEP, PDP; Network Security.

1. INTRODUCTION

An access control security policy related to an information system specifies which actions the subjects are authorized to perform. Deploying the policy means enforcing (i.e., configuring) some security devices or mechanisms in order to guarantee a system behavior as the one specified by the policy. We consider that the policy deployment process is a major concern for security officers – its complexity is associated not only with the security requirements or with the system in terms of architecture (e.g., tens of firewalls, routers) but also with the functionalities of system security devices (e.g., time-based access control lists).

Common policy deployment approaches suggest a formal expression of the security policy based on a control model or using a high level language. Additionally, there is always the hypothesis that all functionalities necessary to enforce the policy are present in the system. As long as the contextual aspect is not dealt with, such an hypothesis is acceptable: the security requirements are not dynamic and therefore all security devices may be enforced once and for all at system initialization. The unaccomplished security requirements are detected from the very beginning, (e.g., in an off-line manner). This may be the result of a deficient functionality or a missing security device (e.g., a firewall delimiting a certain subnetwork). The solution is trivial: the cost of the faulty device is evaluated against assurance requirements and the administrator may proceed to an update/upgrade. However, the policies become more and more contextual and, consequently, more complex. There is a real risk that some security requirements still remain unaccomplished: either the cost of new security functionalities is unacceptable or the security devices cannot be frequently updated with new functionalities.

In this paper we propose a solution to this problem. We propose a mechanism for (re)deploying contextual security policies over systems where security devices do not integrate all the necessary functionalities to deal with contexts. The approach is the following: the security requirements are modeled based on an extended RBAC access control model which provides means to specify contextual security requirements. We take our inspiration in the OrBAC (Organization-Based Access Control) model [1]. The input of the policy (re)deployment process is (1) the formal policy, (2) the system architecture, and (3) the device capabilities or functionalities. Then, as a result of context activation, not only enforcing the security devices but also changes of their configurations are carried out in an automatic manner. Our contribution is twofold: it enhances the OrBAC model and allows security officers to (re)deploy contextual OrBAC policies over security devices unable to interpret their contextual requirements. Moreover, it clearly benefits policy-based reaction scenarios, such as those used by intrusion detection, fault tolerance, and quality of service processes, in which a policy reconfiguration process must follow those detection mechanisms that identified a given attack or anomaly. Our approach has been implemented and evaluated. The results of our evaluation prove moreover the validity and the effectiveness of our approach.

This paper is structured as follows. Section 2 introduces the motivation of our work and some examples of context activation. Section 3 recalls the OrBAC model on which we base our approach and establishes some prerequisites necessary for our approach. Section 4 formally defines how to use the OrBAC model for the deployment of contextual policies. Section 5 presents the application of our proposal upon some sample scenarios and provides evaluation results. Section 6 provides a comparison with related work. Section 7 concludes the paper.

2. MOTIVATION

The current work is placed in the PBNM (*Policy Based Network Management*) area. The security architecture of a policy-based managed system generally integrates the following entities: the PDP (*Policy Decision Point*) and several PEPs (*Policy Enforcement Points*). We consider the PDP as the central and intelligent entity in the network. Based on the security policy data and some algorithms, the PDP decides upon the access control in the network. PDP's decisions must be unambiguous. Given the possible inconsistencies in defining a security policy (i.e., rule conflicts), the PDP may include a conflict resolution mechanism which consequently guarantees the consistency of the policy. The PEP, a security device, is the operational entity in the network which enforces PDP's decisions. Examples of PEPs are the firewalls, Intrusion Detection Systems (IDSs), and IPsec tunnels. There are two modes of interaction between PEPs and PDP both delegating the responsibility for a decision to the PDP: (1) provisioning mode where the PDP, reacting to different inputs (and PEP queries), proactively provisions the PEP and (2) outsourcing mode: when the PEP receives a query, it contacts the PDP which makes a decision, meaning that there is a "one to one correlation between PEP events and PDP decisions". Experiments show that the provisioning mode generally performs better than the outsourcing mode [17].

The decision made by the PDP generally depends on the activation of some *contexts*. The concept of *context* [10, 20] is used here within the specification of security rules which are then triggered when the context becomes active. Here are some examples of contexts:

- Temporal context. It depends on the time at which a subject is requesting for an access to the system. For example, common cisco routers with time-based ACLs (Access Control Lists) may be considered examples of PEPs able to deal with the temporal context if and only if the time intervals do not have a very fine granularity (e.g., seconds);

- Spatial context. It depends on the subject location. This may be the case of IPv6 mobility: given the risk of a Denial of Service (DoS) attack in MIPv6 (Mobile IPv6), specific security rules can be (re)deployed once new "binding updates" are detected [rfc 3775];

- Session context. It depends on the establishment of negotiation parameters, such as network ports or IP addresses. For example, on VoIP (Voice Over IP) applications, the negotiation of randomly chosen UDP ports and/or IP addresses may affect the filtering process of firewalls that are not aware of such parameters. New functionalities like those of SBC (Session Border Controler) partially solve these issues. However, often the same session policy is applied for all users and this may prove unacceptable regarding the user's requirements.

- Intrusion context. It specifies security rules to be activated as a response to an intrusion. For example, a certain IP packet with a specific payload, and related to a known attack, triggers the activation of an "intrusion context". This results in an IDS alert which, in turn, may have been defined as the activating event of a "reaction context". A reaction context may be related, for example, to the (re)deployment of certain firewall rules in order to isolate the victim of the attack;

- User-declared context. It depends on the subject objective (or purpose). For example, a certain IP traffic is not allowed unless the corresponding IP packets are encapsulated in an IPsec tunnel; therefore the IPsec tunnel parameters are part of a context definition (e.g., authentication type, encryption algorithms). See the example given in Section 5.

The concept of *context* is explicitly introduced in the OrBAC [1] access control model on which we base our approach. A complete taxonomy of contexts modeling techniques based on OrBAC can be found in [12]. We give in Section 3 further details about the context definition.

PEPs do not necessarily provide functionalities to implement complex contextual policies (e.g., spatial, session or intrusion contextual policies). We investigate in the sequel how to deploy contextual policies related to the provisioning mode where the PEPs lack some functionalities to enforce contextual policies. To be effective, our policy (re)deployment process requires a specific communication protocol between the PDP and the PEPs. This protocol should ensure a fast automatic update of the PEPs' configurations (cf. Section 3.4). We base our approach on the use of the Netconf protocol [19].

3. THE APPROACH

We first present in brief the chosen access control model. Then, we establish the hypothesis and definitions that introduce our approach; the final goal is to describe how a contextual policy is managed in the PDP-PEP architecture where the PEP does not include all functionalities or mechanisms necessary to manage a certain context.

3.1 The Access Control Model

The security policy of an information system may include a wide range of different requirements: authentication and access control requirements, information flow, and usage control requirements. Specifying administration and delegation policies is also a more and more important issue, especially in the context of pervasive distributed systems.

One of the most frequently used security models is the standard Role Based Access Control model (RBAC [23]). If RBAC appears as being restricted to access control requirements, several RBAC extensions confer the possibility to handle dynamic security policies through some common contexts, such as the temporal or the spatial contexts. In TRBAC [8] temporal constraints, instants and periodicity characterize the time, however a restriction is that the permissions of a role must have the same time interval. GEO-RBAC [9] deals with the geographical constraints through absolute and logical representations. GTRBAC regroups the geographical and temporal contexts.

The Organization Based Access Control model (OrBAC [1]) natively provides means to express both static and contextual access control requirements. Similarly to RBAC which comes with an administration model ARBAC, OrBAC is associated with AdOrBAC whose main feature is its self-administration: the administrative policy is specified using the same concepts of the security policy model. OrBAC is robust in terms of administration and pertaining tools (cf. MotOrBAC [2]) and therefore we choose to formalize the security policy using this model.

OrBAC, based on the *organization* concept, involves two levels of abstraction: (1) an organizational level ("role", "activity", "view" and "context" concepts); and (2) a concrete level ("subject", "action", "object"). It uses first order logic to write access control rules in the form of permissions (*Is_permitted*) and prohibitions (*Is_prohibited*). For example, a concrete permission is derived as follows:

- $\forall$ *org*, $\forall$ *s*, $\forall$ *o*, $\forall$ α, $\forall$ *r*, $\forall$ ν, $\forall$ *a*, $\forall$ *c*,
  Permission(org, r, a, ν, c) $\wedge$ *empower(org, s, r)* $\wedge$ *use(org, o, ν)* $\wedge$ *consider(org, α, a)* $\wedge$ *hold(org, s, a, o, c)* $\rightarrow$ *Is_permitted(s, α, o)*

If the organization *org* grants role r the permission to perform activity a on view ν in context c and if the role r is assigned to subject s (*empower*), the object o is used in view ν (*use*) and α is considered the action implementing activity a (*consider*), s is granted permission to perform α on o. The concepts introduced by OrBAC are the following: (1) *Activity*, regrouping *actions* having common properties; (2) *View*, several *objects* having the same properties on which the same rules are applied; and (3) *Context*, a concept defining the circumstances in which some security rules can be applied. The context allows the definition of specific security requirements directly at the OrBAC level. Subjects empowered in a certain role in an organization cannot realize an action on a given object unless specific conditions are satisfied. Hence a context denotes these specific conditions in which an access rule is activated. The dynamic management of contexts to control the deployment and the redeployment of security policies is further explained in section 4.2.

As RBAC, OrBAC defines role hierarchies, and also views, activities and context hierarchies. In the specialization (or generalization) hierarchy, permissions and prohibitions are inherited *downward*. These hierarchies facilitate the administrator's tasks and also simplify the formalization of the security policy.

In the following sections we describe our approach to deploy an OrBAC security policy P defined over a set of elementary contexts denoted by C. A set of n PEPs, PEP_i, is responsible for enforcing the policy, yet each PEP_i manages only a subset $C_i \subseteq C$ of contexts. According to the OrBAC model we consider the next SR security rule format: SR = (Decision, Role, Activity, View, Context) (i.e., OrBAC organizational level), where Decision $\in$ {Permission, Prohibition}. We call upon a simple algebra when combining elementary contexts: "$\wedge$" (a rule involving the conjunction of two contexts $c_1 \wedge c_2$ is triggered when both contexts hold), "$\vee$" (a rule is triggered when at least one context holds) and "$\neg$" (a rule is triggered when the context does not hold).

3.2 Hypotheses

The OrBAC policy P is managed at the PDP level representing the intelligent entity of the system. The PDP itself can manage a subset $C_{PDP} \subseteq C$ of contexts. For each rule $SR_k \in P$, the set of PEP_i in charge of enforcing SR_k is known. This assumption is based either on the administrator's explicit indication regarding the optimal PEP_i in terms of right functionalities and right emplacement in the system, or especially on the implementation of some algorithms at PDP level with the same purpose [22, 4, 3]: given the network architecture, the PEPs are selected based on the functionalities required by some actions and contexts and also based on the network paths the IP packets establish between a source and a destination.

Regarding the context algebra, we consider that if an entity manages contexts c_1 and c_2, it will also be able to manage the context $c_1 \wedge c_2$. If an entity manages the context c_i, it will also be able to manage the context $\neg\ c_i$. We believe these hypotheses are not restrictive but fairly reflect a reality.

3.3 Definitions and Output

If the input to our process of deployment is the policy P and the PDP-PEP architecture, the output is the concrete PEPs configuration which is usually obtained automatically. This is achieved by a set of algorithms which accomplish the so-called *provisioning* or *downward* deployment approach [13] (cf. Figure 1). The OrBAC security policy P, detached by any security device technology, is first translated into a *multi-target* expression; this represents a set of generic security rules (i.e., independent of the PEPs' technology). The second transformation results in a specific PEP configuration (e.g., package of firewall scripts). The two transformations are carried on at the PDP level. Via a specific PDP-PEP protocol, each PEP is updated with the corresponding package of rules. Globally all PEPs will reflect the OrBAC security policy. The second transformation is in fact a set of compilations that work on the *multi-target* policy, each complying with a given PEP technology. New compilations must be conceived when changes of the security device technology are planned; for example, when a software firewall like Netfilter is replaced by a hardware one like Netasq, the second compilation must be changed.

Except for the PDP-PEP communication protocol (cf. Section 5), the applicability of the *downward* approach is already demonstrated ([13, 22]) and several security technolo-

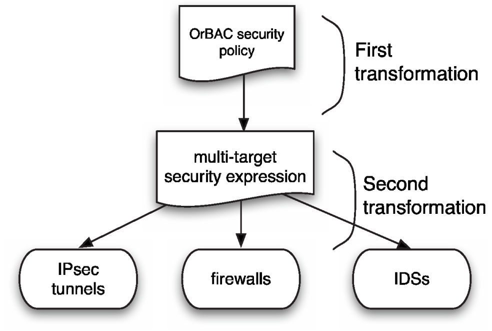

Figure 1: Downward approach

gies were addressed: Netfilter, Netasq, SNORT, etc. In [22] the assumption is that the PEP functionalities are sufficient to respond to the security requirements: all requirements including the contextual ones we were defining may be handled by the PEPs. However, the increasing complexity of the security requirements as a consequence of using contexts imposes some changes on the algorithms of the first transformation. We have to consider that there may be contexts which are not handled anymore by the PEPs' functionalities.

Let us consider SR = (Decision, Role, Activity, View, C) a security rule of the policy P (SR $\in$ P) and SR$'$ = (Decision, Role, Activity, View, C$'$). We call SR$'$ the SR *contextual version* over the context C$'$.

Let PEP$_i$ be an enforcement point able to manage only the context C$'$ and SR be the security rule PEP$_i$ must enforce. We consider that SR$'$ can be deployed and thus enforced by PEP$_i$.

3.4 Methodology

The final aim is to deploy the SR rule; one of the following situations appears:

- *Case 1*: the PEP$_i$ manages the entire C context. The rule SR is directly deployed over PEP$_i$ and the PDP does not manage SR anymore. The deployment is called *static*. Otherwise, the deployment is *dynamic* and consequently, the PDP has to manage a part of the C context.
- *Case 2*: the PEP$_i$ manages only C$_2$, a part of the C context. We note this case as C$_1$ = C $\setminus$ C$_2$. The deployment is *dynamic* and the PDP manages C$_1$: the PDP must deploy SR$'$, the SR contextual version over C$_2$ on the PEP$_i$ when C$_1$ becomes active. What we understand by PDP managing C$_1$ is, for example, that PDP can deploy the SR contextual version over C$_1$ on another PEP$_j$ in such a manner that the two SR contextual versions are equivalent to a single SR deployed on a well placed PEP which includes all functionalities required by the C context; or the PDP is just *sensitive* to the activation of C$_1$. If so, once C$_1$ is deactivated, the PDP must be able to retrieve the deployed SR$'$ from PEP$_i$.
- *Case 3*: the PEP$_i$ manages only C$_2$ and the PDP cannot manage C$_1$. A possibility would be to search for a part of the C$_1$ context, manageable by the PDP and then to deploy the SR contextual version over C$_2$ on PEP$_i$. It is clear that the initial security objective is only partially satisfied. Nevertheless this may prove unacceptable regarding, for example, initial assurance requirements; that is why we choose to stop the SR deployment in this case.

In the following sections we consider the policy is deployable as described above (cases 1 and 2). We deal with the formalization of contexts, their activation and the actions they trigger.

4. FORMALIZATION AND DEPLOYMENT

Baral et al. introduced in [6] a framework for describing active databases and their evolution through events and the actions they cause. We base our approach on the same definition of ECA (*event-condition-action*). After recalling these concepts we show how they are adapted to the formal description of context activation. Finally, we obtain a complete management of the policy (re)deployment.

4.1 The *ECA* Rules

The main concepts in [6] (action, event and active rule) are defined using the L_{active} language [5]; the L_{active} vocabulary includes the following atoms: A - actions, F - fluents (i.e., data which can change their values), E - events and R - rule names. The authors demonstrate that the separation of *event definition* from the *active rule* allows the specification of more complex events. We resume these definitions and we capture only the notions we can use in our policy deployment. Simple examples are provided in order to show their usage.

1. *Action* definition:

   - *action*(X) causes f(Y) if p_1(X$_1$), ..., p_n(X$_n$).

   This corresponds to the causality principle; *action*(X) is an action, f(Y) is the effect and p_1(X$_1$), ..., p_n(X$_n$) are the preconditions of the action (X, Y, X$_1$, ..., X$_n$ are variables). In any state in which p_1(X$_1$), ..., p_n(X$_n$) are true, the execution of the action *action*(X) determines f(Y) be true in the next state.

   The following two examples use intuitive predicates:

   (a) *enter*(Subject, Room)
      - causes *location*(Subject, Room), *nb_people*(Room, N+1)
      - if *nb_people*(Room, N).

   (b) *tick_clock*
      - causes *time*(Global_clock, Time+1)
      - if *time*(Global_clock, Time).

2. *Event* definition:

   - *event*(X) after *action*(Y) if q_1(Z$_1$), ..., q_n(Z$_n$).

   The event X occurs after the execution of the action *action*(Y) if all conditions q_1(Z$_1$), ..., q_n(Z$_n$) are satisfied. *Event*(X) may activate a rule (*active-rule*, see bellow) if certain conditions are satisfied in the current state and consequently *event*(X) is said to be *consumed* (i.e., it does not persist to a future state). Otherwise, for example, the occurrence of the event *alarm_event* may be carried on indefinitely if no rule is activated (e.g., the call of a guardian and the deactivation of the alarm).

(a) *timer_elapsed*(Timer)
  - after *tick_clock*
  - if *delay*(Timer, 0).

(b) *alarm_event*(Room)
  - after *enter*(Subject, Room)
  - if *time*(Global_clock, Time),*Time*>23:00.

3. *Active ECA rule* definition:

   - *ECA_rule_name*: *event*(X) initiates a_1(Y$_1$), ... , a_m(Y$_m$) if r_1(Z$_1$), ... , r_n(Z$_n$).

   The occurrence of *event*(X) triggers the execution of the action sequence a_1(Y$_1$), ..., a_m(Y$_m$) if the conditions r_1(Z$_1$), ..., r_n(Z$_n$) are true. The authors in [6] also use [] and {} to respectively denote a *non interruptible* and an *interruptible* sequence of actions. In our approach we deal only with non interruptible actions, such as:

   (a) *call_guardian*: *alarm_event*(Room)
     - initiates *close*(Room), *call*(Guardian)
     - if *empower*(Guardian,guardian).

   The actions *close*(Room) and *call*(Guardian) are consecutively executed if Guardian was empowered as guardian.

4.2 Application to Policy Deployment

In our approach, the occurrence of an ECA event generally corresponds to the activation of a context which is handled by the OrBAC formalism (cf. Section 3.1). We consider the security rules may involve two types of contexts:

1. *State based context*: it corresponds to the "classical" OrBAC context, modeled with derivation rules and defined as follows:

   - *hold(Org, S, A, O, Ctx)* :- p$_1$(Y$_1$), ..., p$_n$(Y$_n$);

   this means that the context Ctx holds (is active) in organization Org for subject S, action A and object O if a sequence of conditions denoted by the predicates p$_1$(Y$_1$), ..., p$_n$(Y$_n$) is true.

2. *Event based context*: corresponds to the ECA event definition and therefore takes into account the dynamic aspect of a security policy:

   - *hold(Org, S, A, O, Ctx)* after *action*(X) if p$_1$(Y$_1$), ..., p$_n$(Y$_n$);

Two "event based" contexts, *Start*(C) and *End*(C), are associated with each context C of type "state based". They are related to the activation and the deactivation of C. We also take into account the following contexts: (1) the context that persists forever after *Start*(C) and therefore no *End*(C) is associated with it and (2) the context for which *start*(C) is activated by the *init* action of the system (i.e., when the system is initialized). The following examples clarify the *Start*(C) and *End*(C) notions (given that we deal with a same *organisation*, the *Org* attribute will be omitted in the next *hold* predicates).

1. Temporal Context: the *morning* context, independent of *Subject*, *Action* and *Object*, is defined as follows:

   - *hold(Subject, Action, Object, morning) :- Time(Global_clock, T), 08:00≤ T≤ 12:00.*

   The *morning* context holds only if the system supplies a clock (Global_clock) which can be queried (for to assess Time(Global_clock, T)) and whose replies may be evaluated against the interval 08:00-12:00. *Morning* is a "state based" context to which two "event based" contexts are assigned: *Start(morning)* and *End(morning)*:

   - *hold(Subject, Action, Object, Start(morning))* after *tick_clock* if *Time(Global_clock, 08:00).*
   - *hold(Subject, Action, Object, End(morning))* after *tick_clock* if *Time(Global_clock, 12:00).*

2. Spatial Context (1): the following *In_room*(r) "state based" context involves the *Location*($Subject, r$) predicate; r is the identifier of an object of the type *room*. The system should provide the means necessary to evaluate the *Location*($Subject, r$) predicate.

   - *hold(Subject, Action, Object, In_room(r)) :- Location(Subject, r).*

   The following "event based" contexts, *Start(In_room(r))* and *End(In_room(r))* depend only on *Subject*; they are activated as a consequence of the actions *Enter* and *Exit* the room:

   - *hold(Subject, Action, Object, Start(In_room(r)))* after *Enter(Subject, r)* if *True.*
   - *hold(Subject, Action, Object, End(In_room(r)))* after *Exit(Subject, r)* if *True.*

3. Spatial Context (2): the *Alone_in_room(r)* context takes into account the location of a subject in the room r as well as the number of people in the same room; the system has to provide means to assess the number of people in r.

   - *hold(Subject, Action, Object, AloneIn_room(r)) :- Location(Subject, r), Nb_people(r, 1).*

   The context *Start(AloneIn_room(r))* is activated as a result of entering the room with no people or when a different subject leaves the room which initially contains only two people. The *End(AloneIn_room(r))* context follows a similar reasoning.

- *hold(Subject, Action, Object, Start(AloneIn_room(r)))* after *Enter(Subject, r)* if *Nb_people(r, 0)*.
- *hold(Subject, Action, Object, Start(AloneIn_room(r)))* after *Exit(Subject′, r)* if *Nb_people(r, 2)*, *Location(Subject,r)*, *Subject′≠Subject.*
- *hold(Subject, Action, Object, End(AloneIn_room(r)))* after *Enter(Subject′, r)* if *Nb_people(r, 1)*, *Location(Subject,r)*, *Subject′≠Subject.*
- *hold(Subject, Action, Object, End(AloneIn_room(r)))* after *Exit(Subject, r)* if *True.*

Let us now consider a composed "state based" context, $C = C_1 \wedge C_2$. The corresponding "event based" *Start*(C) and *End*(C) contexts are evaluated using the elementary contexts C_1 and C_2:

- $Start(C) = (Start(C_1) \wedge C_2) \vee (C_1 \wedge Start(C_2))$.
- $End(C) = (End(C_1) \wedge C_2) \vee (C_1 \wedge End(C_2))$.

If $C = \neg C_1$ and *init* is the context related to the system initialization, then $Start(C) = End(C_1) \vee (init \wedge \neg C_1)$, respectively $End(C) = Start(C_1)$.

The notions presented in this section along with the context algebra are used in the deployment of a contextual policy. *Start*(C) and *End*(C), corresponding to the activation and deactivation of a relevant context C (i.e., C is used in SR rules), trigger in addition the deployment and respectively the retrieval of certain SR rules. The management is at the PDP level.

4.3 Deployment Management

The use of ECA active rules represents the principle of our contextual policy deployment in the PDP-PEP architecture. As already described, an ECA involves a "triggering event" which corresponds here either to *Start*(C) or to *End*(C) and a sequence of actions; there are only two actions that need to be defined in order to meet all deployment requirements:

- *activate*(PEP, Security_Rule)
- *deactivate*(PEP, Security_Rule)

The first concerns the deployment of the SR Security_Rule over the PEP enforcement points, the second represents its retrieval. We recall, SR = (Decision, Role, Activity, View, Context); for space limitation reasons, "Decision" will represent a Permission and will be omitted in what follows ([14] shows how a security policy containing both permissions and prohibitions may be rewritten into an equivalent one containing only permissions). For each security rule SR(Role, Activity, View, Context) the following rule is derivable (cf. Section 3.4):

- *SR_version*(PEP, Role, Activity, View, PEP_Ctx, PDP_Ctx).

In the *SR_version* predicate, the first attribute specifies the PEP on which the SR contextual version over the context PEP_Ctx is deployed; PEP_Ctx is the part of context managed by the PEP and PDP_Ctx is the one managed by PDP (cf. Section 3.3). The context management is based at the PDP level and the deployment of an SR contextual version is triggered in the most general case as follows:

- *activate_rule:*
  - *hold(Subject, Action, Object, Start(PDP_Ctx))*
    - initiates *activate(PEP, SR(Subject, Action, Object, PEP_Ctx))*
    - if *SR_version(PEP, Role, Activity, View, PEP_Ctx, PDP_Ctx), Empower(Subject, Role), Consider(Action, Activity), Use(Object, View).*

This expression respects the ECA definition and uses predicates belonging to the OrBAC formalism. The PDP_Ctx activation (*Start(PDP_Ctx)*) triggers the deployment of the SR contextual rule. The PEP enforces a concrete rule (i.e., involving concrete entities in the form of subjects, actions and objects), hence the deployed rule complies with the specific format of the OrBAC concrete level (*SR(Subject, Action, Object, PEP_Ctx)*). Given that the initial policy is an OrBAC organizational policy (i.e., SR(Role, Activity, View, Context)) some conditions are classic OrBAC conditions: *Empower(Subject, Role)*, *Consider(Action, Activity)*, *Use(Object, View)*.

The retrieval of a security rule from the PEP is derived in a similar way and is first conditioned by the deactivation of the PDP_Ctx:

- *deactivate_rule:*
  - *hold(Subject, Action, Object, End(PDP_Ctx))*
    - initiates *deactivate(PEP, SR(Subject, Action, Object, PEP_Ctx))*
    - if *SR_version(PEP, Role, Activity, View, PEP_Ctx, PDP_Ctx), Empower(Subject, Role), Consider(Action, Activity), Use(Object, View).*

Different PDP_Ctx contexts may be related to a specific subject, action, and object. Hence, the above active rules will be applied to every instantiation of subjects, actions and objects. This may lead to deploy a large set of concrete security rules over each PEP. If so, the security policy does not take full advantage of the OrBAC organizational expression (*SR(Role, Activity, View, Context)*). Solutions to improve the policy (re)deployment exist and they take into account the context definition fashion, more exactly the PDP_Ctx part of context.

4.4 Deployment optimization

In this section we show how to reduce the number of security rules to deploy onto the PEPs.

The most convenient (re)deployment case corresponds to the definition of PDP_Ctx independently of subjects, actions and objects (e.g, the temporal context). The SR rule will automatically be deployed as soon as PDP_Ctx is active; the ECA activate rule involves a minimum set of predicates:

- *activate_rule:* $hold_{\emptyset}$*(Start(PDP_Ctx))*
  - initiates *activate(PEP, SR(Role, Activity, View, PEP_Ctx))*

– if *SR_version(PEP, Role, Activity, View, PEP_Ctx, PDP_Ctx).*

The ECA deactivate rule is similarly defined. "$hold_\emptyset$" indicates that a context independent of subject, action and object is active. Another case arises when the PDP_Ctx context is dependent only on subject (e.g., the "*In_room(r)*" context which is activated when a subject enters in room r and deactivated when this subject exits). Before deploying the rule, the subject must be instantiated. "$hold_S$" indicates that the activation of context does not depend on actions or objects but on subject:

- *activate_rule:* $hold_S$*(Subject, Start(PDP_Ctx))*
  - initiates *activate(PEP, SR(Subject, Activity, View, PEP_Ctx))*
  - if *SR_version(PEP, Role, Activity, View, PEP_Ctx, PDP_Ctx), Empower(Subject, Role).*

A corresponding deactivation rule using "$hold_S$" is similarly defined. Here, the PDP_Ctx activation may actually rely on subjects (S), actions (A), objects (O) and/or combinations; hence, there are eight possible context definition cases according to: $hold_{SAO}$, $hold_\emptyset$, $hold_S$, $hold_A$, $hold_O$, $hold_{SA}$, $hold_{SO}$, $hold_{AO}$, the first three being detailed in this section.

5. APPLICATION OF OUR APPROACH

In order to illustrate our proposal we consider the case of a corporation with a security policy including contextual requirements. The architecture in Figure 2 depicts several sub-networks guarded by firewalls (two of them including IPsec functionalities):

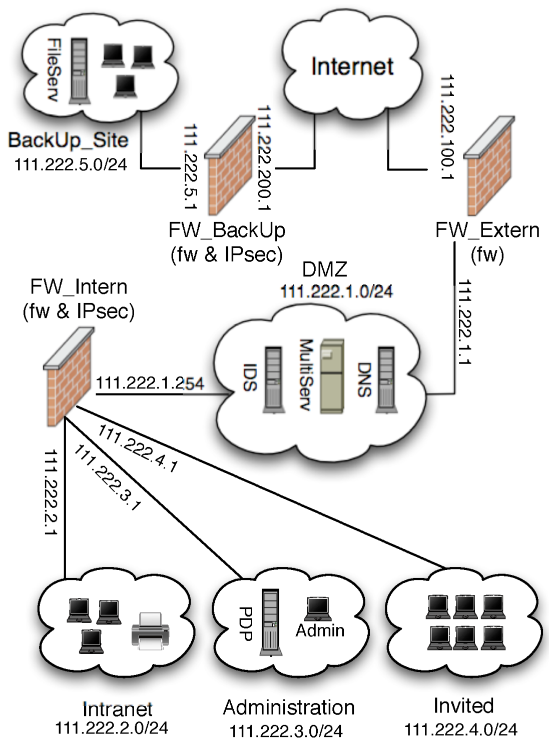

Figure 2: Topology example.

Table 1: Default Requirements

The email services are accessible from the Intra zone
The email server activates the smtp service to the Internet
The DNS service is accessible from all zones
The icmp traffic is allowed in the Corporate (Corp) zones
The PDP activates the Netconf service with the PEPs
The Administration may access the PEPs via ssh

Table 2: Contextual Requirements

All Intra – BackUP_Site TCP traffic is protected
During w.h., TCP traffic is allowed from Intra to Internet
The pop, imap and smtp are accessible from the Invited zone These services are blocked after 3 failed logins
The IDS detects the syn-flooding attacks and alerts the PDP
The web server is public if no syn-flooding (s.f.) is detected

Table 3: The OrBAC policy

R1	*Permisssion(R_Intra, rw_mail, multi-serv, default)*
R2	*Permisssion(R_MultiServ, w_mail, Internet, default)*
R3	*Permisssion(R_Internet, w_mail, multi-serv, default)*
R4	*Permisssion(R_Corporate, dns, dnsServ, default)*
R5	*Permisssion(R_Internet, dns, dnsServ, default)*
R6	*Permisssion(R_DNS, dns, Internet, default)*
R7	*Permisssion(R_Corporate, icmp, Corp, default)*
R8	*Permisssion(R_Corporate, icmp, Internet, default)*
R9	*Permisssion(R_PDP, netconf, PEP, default)*
R10	*Permisssion(R_Admin, ssh, PEP, default)*
R11	*Permisssion(R_Admin, TCP, Internet, default)*
R12	*Permisssion(R_PEP, ssh, Admin, default)*
R13	*Permisssion(R_Intra, TCP, BkUp, protected)*
R14	*Permisssion(R_Intra, TCP, Internet, w.h.)*
R15	*Permisssion(R_Invited, rw_mail, multi-serv, !(m.l.f.i.))*
R16	*Permisssion(R_IDS, alert, PDP, syn-flooding)*
R17	*Permisssion(R_Internet, WEB, multi-serv, !(s.f.))*

- the *DMZ* zone (111.222.1.0/24) includes a multi-server: web (webServ) and email (mailServ) server; the email server is accessible via imap and pop from the *Intra* zone or via a web-mail service from the *Internet*. The corporation network brings in a *DNS* server and the DMZ zone also contains a signature-based IDS;
- the *Administration* zone (111.222.3.0/24) comes with the administration tools: an *Admin* PC which may access all PEPs (firewalls and IDS) via ssh; the PDP equipment uses the data format in [22] to store and process the OrBAC corporation policy. We focused on Netconf [rfc 4741] as the communication PDP-PEP protocol;
- the *Intranet* (Intra) zone (111.222.2.0/24) is considered the corporation working area. The Intra equipments may access all TCP Internet services during the working hours (w.h., 08:00–20:00) and use a protected channel with the *BackUP_Site*, i.e., an IPsec channel is required);

Table 4: Context Definition and ECA rules

context	Start / End context	ECA rule generation
protected (user declared): managed by PEPs IPsec-enabled (PEP_ctx) and by the PDP (PDP_prot) which updates firewalls (*default* context) on the tunnel path; *path(s, o)* returns the list of PEPs ([PEP$_i$]) on the IPsec tunnel path between *s* and *o*.	$hold_{SO}$*(Intra, BackUp_Site, Start(PDP_prot))* after *boot(*PEP_1*,* PEP_n*)* if [PEP$_i$]←*path(Intra, BackUp_Site)*. $hold_{SO}$*(Intra, BackUp_Site, End(PDP_prot))* after *halt(*PEP_1*,* PEP_n*)* if [PEP$_i$]←*path(Intra, BackUp_Site)*, i=$\overline{(1,n)}$.	$hold_{SO}$*(Intra, BackUp_Site, Start(PDP_prot))* initiates *activate(*PEP_i*, SR(R_PEP, ipsec, PEP))*, *activate(*$PEP_{1,n}$*, SR(Intra, TCP, BkUp, PEP_ctx))* if *SR_version(*$PEP_{1,n}$*, R_Intra, TCP, BkUp, PEP_ctx, PDP_prot)*, *Empower(Intra, R_Intra), Use(BackUp_Site, BkUp)*, *Empower(*PEP_i*, R_PEP), Use(*PEP_i*, PEP)*.
working-hours (w.h.): temporal context; managed either by PEPs with a *temporal* functionality or by the PDP (we consider the latter case).	$hold_{\emptyset}$*(Start(w.h.))* after *tick_clock* if *Time(Global_clock, 08:00)*. $hold_{\emptyset}$*(End(w.h.))* after *tick_clock* if *Time(Global_clock, 20:00)*.	$hold_{\emptyset}$*(Start(w.h.))* initiates *activate(PEPi, SR(R_Intra, TCP, R_Internet))* if *SR_version(*PEP_i*, R_Intra, TCP, R_Internet, w.h.)*.
mail-login-fail-invited (m.l.f.i.): three failed logins in the email server from *Invited* result in blocking the email services for the traffic causing the failures (End(C) irrelevant); managed by the PDP.	$hold_{SO}$*(S, mailServ, Start(m.l.f.i))* after *fails(S, auth, mailServ)* if *Nb_fails(S, 3)*.	$hold_{SO}$*(S, mailServ, Start(m.l.f.i))* initiates *deactivate(*PEP_i*, SR(S, rw_mail, mailServ))* if *SR_version(*PEP_i*, R_Invited, rw_mail, multi-serv, !(m.l.f.i))*, *Empower(S, R_Invited), Use(mailServ, multi-serv)*.
syn-flooding (s.f.): dependent on subject and managed by the PDP, *s.f.* is activated after a "syn-flooding" IDS alert on the web server; *s.f.* deactivates after a few minutes (e.g., 8 minutes).	$hold_{O}$*(webServ, Start(s.f.))* after *alert(IDS, s.f.-attack-webServ)*. *alert(IDS, s.f.-attack-webServ)* causes *set(Timer.s.f., 8mn)*. $hold_{O}$*(webServ, End(s.f.))* after *tick_clock* if *delay(Timer.s.f, 0)*.	$hold_{O}$*(webServ, Start(s.f.))* initiates *deactivate(*PEP_i*, SR(R_Internet, WEB, webServ))* if *SR_version(*PEP_i*, R_Internet, WEB, multi-serv, !(s.f.)), Use(webServ, multi-serv)*. $hold_{O}$*(webServ, End(s.f.))* initiates *activate(*PEP_i*, SR(R_Internet, WEB, webServ))* if *SR_version(*PEP_i*, R_Internet, WEB, multi-serv, !(s.f.)), Use(webServ, multi-serv)*.

- the *Invited* zone (111.222.4.0/24) is a wireless network. All new "invited" equipments dynamically obtain an IP address. The security policy also specifies a default imap/pop access to the multi-server as long as the *mail-login-failed-invited* context is not activated (see below);
- the *BackUP_Site* (BkUp) zone (111.222.5.0/24) is a geographically different sub-network including the back-up file server of the corporation.

The corporation security policy is informally presented in Table 1 and 2. Table 1 introduces the *default* requirements (i.e., they have to be satisfied in any conditions). However, ensuring a protected channel Intra – BackUP_Site is considered a contextual requirement (cf. R13 in Table 3). In what follows we resume the OrBAC concepts related to the above architecture and we insist on the definition of contexts:

- *roles*: *R_Administration*, *R_Intra*, *R_Invited*, *R_DMZ*, *R_MultiServ*, *R_BackUP_Site* and *R_DNS* (these correspond respectively to the aforementioned zones); the role *R_Corporate* is the one that all entities in the sub-network 111.222.0.0/16 are empowered in. *R_Internet* corresponds to 0/0$\setminus$111.222.0.0/16. *R_Admin* is the role of the "Admin" subject. *R_PEP* is inherited by all security devices including the firewalls and the IDS. The PDP equipment has the role *R_PDP*;
- *activities* (abstraction of network services): *WEB* (http and https), *TCP* (all tcp services), *dns* (either intra or inter zone dns transfers), *ipsec* (isakmp, ESP and/or AH traffic); the *netconf* and *ssh* activities are related to respectively the Netconf and ssh protocols. The *r_mail* (*read email*) activity has two sub-activities: *r_pop* and *r_imap*. The *w_email* (*write mail*) corresponds to the smtp service; moreover, in the specialization hierarchy, *rw_mail* is a more specialized activity than *r_mail* and *w_mail* and consequently inherits them. The *icmp* activity includes all *icmp* traffics.

The *view* definitions follow the same reasoning; the administrator chooses either entities or zones on which the above activities are realized. E.g., the *Internet* view includes the same Internet zone. The *PEP* view includes the security devices on which either *netconf* or *ssh* activities are performed. All interesting views are depicted in Table 3 which resumes the OrBAC security policy.

The process of deriving concrete PEPs configurations is automatically realized at the PDP level following the methodology of Sections 3 and 4.3. According to the requirements of Table 2, several contexts must be defined (the *default* context which is always true is trivially interpreted by the PEPs with firewall functionalities). Based on the active ECA rule generations at the PDP level, the activation of these contexts triggers the (re)deployment of the related security rules. Table 4 resumes the definition of these contexts and the significant ECA rules which are automatically generated.

We recall that the (re)deployment process requires a PDP-PEP specific protocol. To validate our approach, we use the Netconf protocol based on the YencaP implementation [27]. YencaP lacks of security modules but designing new ones is possible. YencaP comes with enough tools which facilitate this task. However, since the original implementation was thought as a GUI tool where the PDP-PEP communication has to be assisted by the administrator, more important efforts are necessary if the module is designed as a stand-alone one (i.e., the PDP-PEP communication is triggered by the activation of contexts). We proved the feasibility of such security modules via a provisioning module which we use to remotely configure (i.e., update) the policies of firewalls, e.g. Netfilter.

Motivated by the latency that our approach may impose over real case scenarios, we evaluated the communication delay of our implementation during the (re)deployment of an incremental set of real temporal filtering policies towards Netfilter-based PEPs. Although PBNM systems are typically deployed on local intranets and/or local area networks (LANs), they may also involve the use of WANs, the Internet, or any other kinds of networks. Solutions such as YencaP, Shibboleth [24], and Permis [21] are just few examples of systems that can allow WAN scenarios based on the deployment of PDP-PEP through the Internet. That is why we assumed the following two scenarios during our evaluation: a best case scenario that comprises the deployment of a Policy Based Network Management (PBNM) system on a LAN; and a worst case scenario where the system is deployed on a wide area network (WAN). We show in the sequel the results of our practical evaluation via YencaP.

5.1 Evaluation and Results

We describe in this section a practical evaluation of two different experiments: (1) evaluation of the communication latency during the (re)deployment of temporal filtering policies from a YencaP-based PDP towards Netfilter-based PEPs; and (2) evaluation of bandwidth degradation due to the overhead imposed by setting equivalent rules on Netfilter-based PEPs.

The hardware profile of our first experiment consists of a PDP $\mathcal{P}_1$ which runs on an Intel Core 2 Duo 2 GHz and 512 MB of memory; a PDP $\mathcal{P}_2$ which runs on an Intel PIV 1 GHz with 512 MB of memory; and a PEP running on an Intel Pentium IV 1 GHz and 256 MB of memory. PEP filtering is based on Linux 2.6.24 and Netfilter. The first curve in Figure 3 corresponds to the deployment of rules between PDP $\mathcal{P}_1$ and PEP through a Local Area Network (LAN) setup based on an ethernet 100MB/s switch. The second curve in Figure 3 corresponds to the deployment of rules between PDP $\mathcal{P}_2$ and PEP through a Wide Area Network (WAN) setup. PDP $\mathcal{P}_2$ and PEP are located on different networks and on different countries. On average, the number of routers between PDP $\mathcal{P}_2$ and PEP is from thirteen to fifteen. Each set or rules submitted by both $\mathcal{P}_1$ and $\mathcal{P}_2$ to PEP is based on the template: *iptables -A FORWARD -i $Iface -p $Protocol -s $SrcNet -d $DstNet –sport $SrcPorts –dport $DstPorts -j ACCEPT*. The following table shows the size associated to each data set depicted in Figure 3:

# of rules	2000	4000	6000	8000	10000
Size (KB)	192	383	575	766	958

We can appreciate by looking at the curves of Figure 3 that the average time for (re)deploying from one to two thousand filtering rules is lower than one second in the LAN case scenario; and less than two seconds in the WAN case scenario. Though filtering sets of more than two thousand rules are rarely used among organizations, we considered interesting to measure the delays up to ten thousand rules which, on average, is lower than ten seconds in the LAN case scenario; and lower than twelve seconds in the WAN case scenario. We consider that these results are very positive and prove the validity of our approach.

With the objective of comparing these results with the alternative of holding temporal filtering rules directly at the PEP, we conducted a second experiment to measure the bandwidth degradation that such an approach may suppose. This alternative solution assumes the possibility that smart PEPs could manage contexts on their own. Although this option may not always be possible, i.e., management of provisional context whose activation depends on some previous actions performed by a human subject, we consider in the following the use of temporal filtering rules. Indeed, most current filtering PEPs (i.e., firewalls) can be upgraded with *temporal* functionality. Consequently, PDPs may leave those components to manage their temporal contexts.

Netfilter takes into account the temporal context via the *time* option. For example, the following Netfilter rule states that all traffic must be accepted in the interval 08:00:15 to 16:00:15: *iptables -A FORWARD -m time –timestart 08:00:15 –timestop 16:00:15 -j ACCEPT*. Nevertheless, using the *time* option can cause a degradation of the firewall resource consumptions. As negative result, the network bandwidth may decrease much faster than using equivalent rules without the *time* option. We show in Figure 4 a practical evaluation that proves our claim.

The hardware profile of the second experiment consists of the same PEP described above, i.e., an Intel PIV 1 GHz and 256 MB of memory whose filtering process is based on Linux 2.6.24 and Netfilter. Throughput was measured using netperf.

Each rule loaded into the PEP during this second experiment (associated with the accumulative set of rules repre-

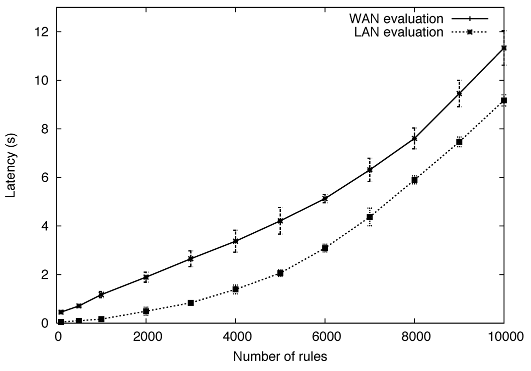

Figure 3: Communication latency during the deployment of temporal filtering policies.

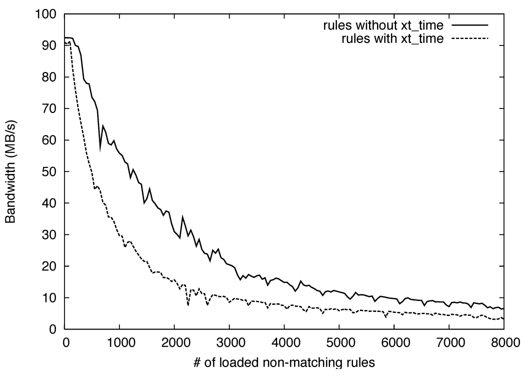

Figure 4: Bandwidth degradation due to the consecutive loading of Netfilter rules.

sented by the curves shown in Figure 4) is based on the following template: *iptables -A FORWARD -i $Iface -p $Protocol -s $SrcNet -d $DstNet –sport $SrcPorts –dport $DstPorts -j ACCEPT*. Rules associated with the second curve of Figure 4 add moreover the following three parameters: *-m time –datestart $Date1 –datestop $Date2*, where both variable *$Date1* and *$Date1* are given in ISO 8601 format, i.e., *YYYY-MM-DDTHH:MM:SS.* We notice a clear bandwidth diminishing when the firewall policy involves a great number of rules with the extra *time* parameter. In such cases the PDP should handle the *temporal* context.

6. RELATED WORK

IETF/DMTF [rfc 3198] settled the terminology of the PBNM architecture, however several notions need to be clarified. For example, regarding the policy server, it is mentioned that as the [rfc 3198] evolved, the policy server refers specifically to a PDP. Beside requesting and providing decisions in the system, the server also maintains a close interaction with the entire system and consequently the server is perceived by some authors as including also the PEP. IETF excluded such a proposal because vendors provide components which behave as either a PDP or a PEP. On the other hand, the policy server definition should also include the conflict resolution aspect. The rfc's authors give no clear indication as to "the implementers of policy system provide conflict detection and avoidance or resolution mechanisms to prevent this situation". All these concepts were addressed and slightly redefined in the literature ([25]). The notion of context is used in [rfc 3198] (i.e., "particular context") but with a different semantic than our *context* concept. The "particular context" refers to a domain policy (i.e., a given set of entities the security policy operates on) as, for example, a company's network.

Several approaches suggest deploying security policies over security components but without considering contexts (see [7] for instance). In [18], the security policy is deployed over *micro-firewalls*, firewalls assigned to each host in the network. The policy is centralized at the policy manager level and can be dynamically changed as a result of IDS alerts. [18] informally compares three technologies in terms of speed and resource consumption for implementing the micro-firewall architecture: the Mobile Agents and the CORBA Middleware implement a distributed IDS and the RMI Middleware is used to implement the policy updates. A similar work is [26]. Once a policy (sub)administrator detects a cooperative intrusion attempt, a response is computed. However, neither [18] nor [26] give clear indication about the response strategy or the mapping from alerts to countermeasures. [15] and [16] deal with deploying a security policy and with providing responses to security threats. In [16] the threats are modeled as contexts and the IDMEF alerts are mapped to contexts. In [15] not only contexts but new policy instances are derived as a result of IDMEF alerts. [15] discusses the context lifetime according to IDMEF impact severity and type. The response strategy is influenced by the mapping of IDMEF alerts to new security rules. Both [15] and [16] consider the PDP manages the threat contexts and the PEPs include all other functionalities necessary to enforce the new policy instances.

Regarding the PDP-PEP protocol, [11] uses COPS-PR to distribute an IPsec policy in an IPv6 network. [17] compares COPS-PR and Netconf in terms of bandwidth and CPU consumptions and concludes that the Netconf protocol is more suitable for network configurations. As the IETF Resource Allocation Protocol group submitted the last COPS draft in December 2004, the IETF Netconf group currently proposes new bricks to the Netconf protocol [19].

7. CONCLUSION

Contextual access control policies provide the means to handle complex security system requirements in a flexible and dynamic manner. However, Policy Enforcement Points (PEPs), such as firewalls, Intrusions Detection Systems (IDSs), and IPsec tunnels, do not necessarily have the required functionalities to manage the expressiveness of these policies. We presented in this paper an approach to cope with this problem. Our work enhances the Organization-Based Access Control (OrBAC) model and allows security officers to (re)deploy contextual OrBAC policies over PEPs unable to interpret their contextual requirements. The approach has been implemented and evaluated. We have presented the performance of the communication between Policy Decission Points (PDPs) and PEPs based on our approach and the use of the Netconf protocol. We compared the overhead of (re)deploying firewall rules without contextual constraints towards the use of firewall rules including those constraints. The results prove the validity and effectiveness of our approach.

The contribution of our work benefits policy-based reaction scenarios, such as those used by intrusion detection processes. In these scenarios, a policy reconfiguration process must follow those detection mechanisms that identified a given attack or anomaly. The reconfiguration process aims at providing long term reaction by fixing the security weaknesses that allowed the attacks identified during the detection process. PDPs must ensure the consistency of new configurations that are placed into, and enforced by, their associated PEPs. Similarly, our approach may also benefit fault tolerance and quality of service scenarios. In these other scenarios, an automatic (re)deployment process must equally ensure that intelligent components, i.e., PDPs, which are aware of the policy of a system as a whole, guarantee the compliance of the statements and constraints defined by network officers. At the same time, they ensure the compliance of the expected objectives, such as reduction of damage, balanced stability, and proper performances. We are currently working on extending our approach to deploy access control policies over more complex scenarios, such as those scenarios that must include policy obligations.

8. ACKNOWLEDGMENTS

The work presented in this paper is supported by a grant from The Britany Region, France, and by funding from POLUX ANR-06-SETIN-012 project.

9. REFERENCES

[1] Abou el Kalam, A., Baida, R. E., Balbiani, P., Benferhat, S., Cuppens, F., Deswarte, Y., Miège, A., Saurel, C., and Trouessin, G. Organization Based Access Control. In *IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, pp. 120–131, Lake Come, Italy, 2003.

[2] Autrel, F., Cuppens, F., Cuppens-Boulahia, N., and Coma, C. MotOrBAC 2: A security policy tool. In *3rd*

Joint Conference on Security in Network Architectures (SAR) and Security of Information Systems (SSI), Loctudy, France, Octobre 2008.

[3] Alfaro, J. G., Cuppens, F., and Cuppens-Boulahia, N. Analysis of Policy Anomalies on Distributed Network Security Setups. In *European Symposium On Research In Computer Security (Esorics)*, Lecture Notes in Computer Science, 4189, pp. 496-511, Hamburg, Germany, September, 2006.

[4] Alfaro, J. G., Boulahia-Cuppens, N., and Cuppens, F. Complete Analysis of Configuration Rules to Guarantee Reliable Network Security Policies. In *International Journal of Information Security*, 7(2):103-122, April 2008.

[5] Baral, C., and Lobo, J. Formal characterization of active databases. In *International Workshop on Logic in Databases (LID'96)*, Italy 1996.

[6] Baral, C., Lobo, J., and Trajcevski, G. Formal characterization of active databases: part II. In *5th International Conf. on Deductive and Object-Oriented Databases*, 1997.

[7] Bartal, Y., Mayer, A., Nissim, K., and Wool, A. Firmato: A novel firewall management toolkit. In *20th IEEE Symposium on Security and Privacy*, Oakland, California, 1999.

[8] Bertino, E., Bonatti, P. A. and Ferrari, E. TRBAC: A temporal role-based access control model. In *ACM TIS- SEC*, 4(3):191233, 2001.

[9] Bertino, E., Catania, B., Damiani, M. L. and Perlasca, P. Geo-rbac: a spatially aware rbac. In *10th ACM Symposium on Access control Models and Technologies*, June 2005.

[10] Cholewka, D. G. , Botha, R. A. , and Eloff, J. H. P.. A Context-sensitive Access Control Model and Prototype Implementation. In *IFIP TC 11 16th Annual Working Conference on Information Security*, Beijing, China, 2000.

[11] Clemente, F., Lopez, G., Martinez, G., and Gomez-Skarmeta, A. Deployment of a Policy-Based Management System for the Dynamic Provision of IPsec-Based VPNs in IPv6 Networks. In *2005 Symposium on Applications and the Internet Workshops*, pp.10–13, 2005.

[12] Cuppens, F. and Cuppens-Boulahia, N. Modeling contextual security policies. In *International Journal of Information Security*, 7(4): 285-305, 2008.

[13] Cuppens, F., Cuppens-Boulahia, N., Sans, T. and Miege, A. A formal approach to specify and deploy a network security policy. In *2nd Workshop on Formal Aspects in Security and Trust*, pp. 203–218, Toulouse, France, 2004.

[14] Cuppens-Boulahia, N., Cuppens, F., Abi Haidar, D. and Debar H. Negotiation of Prohibition: An Approach Based on Policy Rewriting. In *23rd International Information Security Conference (SEC 2008)*, Italy, September 2008.

[15] Debar, H., Thomas, Y., Cuppens, F. and Cuppens-Boulahia, N. Enabling Automated Threat Response through the Use of a Dynamic Security Policy. In *Journal in Computer Virology (JCV)*, 3(3):195-210, 2007.

[16] Debar, H., Thomas, Y., Cuppens, F. and Cuppens-Boulahia, N. Using contextual security policies for threat response. In *Third GI International Conference on Detection of Intrusions & Malware, and Vulnerability Assessment (DIMVA)*, Berlin, Germany, 2006.

[17] Franco, T., Lima, W., Silvestrin, G., Pereira, R. C., Almeida, M. J. B, Tarouco, L. M. R., Granville, L. Z., Beller, A., Jamhour, E., and Fonseca, M. Substituting COPS-PR: An Evaluation of NETCONF and SOAP for Policy Provisioning. In *7th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'06)*, pp. 195–204, USA, 2006.

[18] Hwang, K. and Gangadhran, M. Micro-Firewalls for Dynamic Network Security with Distributed Intrusion Detection. In *International Symp. on Network Computing and Applications*, 2001.

[19] IETF Netconf Working Group. [On-line]. Available from: `http://www.ops.ietf.org/netconf/`.

[20] McDaniel, P. On Context in Authorization Policy. In *8th ACM Symposium On Access Control Models and Technologies (SACMAT 2003)*, Como, Italy, June 2003.

[21] Permis project. [On-line]. Available from: `http://sec.cs.kent.ac.uk/permis/`.

[22] Preda, S., Cuppens, F., Cuppens-Boulahia, N., Alfaro, J. G., and Toutain, L. Reliable Process for Security Policy Deployment. In *International Conf. on Security and Cryptography*, Spain, 2007.

[23] Sandhu, R., Coyne, E. J., Feinstein, H. L., and Youman, C. E. Role-Based Access Control Models. In *IEEE Computer*, 29(2):38–47, 1996.

[24] Shibboleth system. [On-line]. Available from: `http://shibboleth.internet2.edu/`.

[25] Strassner, C.J. Policy-based Network Management, Solutions for the Next Generation. Elsevier, Morgan Kaufmann Publishers 2004. ISBN: 1-55860-859-1.

[26] Xian, F., Jin, H., Liu, K. and Han, Z. A Mobile-Agent based Distributed Dynamic μFirewall Architecture. In *9th International Conf. on Parallel and Distributed Systems*, pp. 431-436, 2002.

[27] YencaP, a Netconf agent for Linux. [On-line]. Available from: `http://ensuite.sourceforge.net/`.