Dynamic Deployment of Context-aware Access Control Policies for Constrained Security Devices[1]

Stere Preda, Frédéric Cuppens, Nora Cuppens-Boulahia,
Joaquin Garcia-Alfaro, and Laurent Toutain

IT TELECOM Bretagne, CS 17607, 35576
Césson-Sévigné, France
E-mail: {firstName.secondName}@telecom-bretagne.eu

Abstract

Securing the access to a server, guaranteeing a certain level of protection over an encrypted communication channel, executing particular counter measures when attacks are detected are examples of security requirements. Such requirements are identified based on organizational purposes and expectations in terms of resource access and availability and also on system vulnerabilities and threats. All these requirements belong to the so-called security policy. Deploying the policy means enforcing, i.e., configuring, those security components and mechanisms so that the system behavior be finally the one specified by the policy. The deployment issue becomes more difficult as the growing organizational requirements and expectations generally leave behind the integration of new security functionalities in the information system: the information system will not always embed the necessary security functionalities for the proper deployment of contextual security requirements. To overcome this issue, our solution is based on a central entity approach which takes in charge unmanaged contextual requirements and dynamically redeploys the policy when context changes are detected by this central entity.

We also present an improvement over the OrBAC (Organization-Based Access Control) model. Up to now, a controller based on a contextual OrBAC policy is passive, in the sense that it assumes policy evaluation triggered by

[1]This is a revised and expanded version of a paper that appeared in the proceedings of the 2009 ACM Symposium on Information, Computer and Communications Security, Sydney, Australia, pages 251–261.

access requests. Therefore, it does not allow reasoning about policy state evolution when actions occur. The modifications introduced by our work overcome this limitation and provide a proactive version of the model by integrating concepts from action specification languages.

Key words: IT Security; Network Security; Authorization; OrBAC.

---

1. Introduction

The configuration of network security components, such as firewalls, intrusion detection systems, and VPN routers, must guarantee the appropriate enforcement of a security policy. This involves building up an abstract model of the security requirements expected to be enforced over the system. This model must provide to the security officer the appropriate means to deploy over the system the set of actions that every subject is authorized to perform. Therefore, the deployment of the policy over the system means enforcing (i.e., configuring) the security devices that must guarantee the proper behavior defined by the policy. Additionally, there is always the hypothesis that all the functionalities necessary to enforce a given policy are present and enabled in these devices. As long as the contextual aspect is not dealt with, such a hypothesis is acceptable: the security requirements are not dynamic and, therefore, all the security devices may be enforced once and for all at the 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 or missing functionality on the security device. The solution is trivial: the cost of the faulty device is evaluated against assurance requirements and the security officer may proceed to an update or an upgrade. However, security 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 some new security functionalities is unacceptable or the security devices cannot be frequently updated with such new functionalities.

We present a solution to this problem. We propose a mechanism for the dynamic refinement of contextual security policies over systems where security devices do not understand, or are not allowed, to deal with the semantics of contexts. Our approach works as follows. 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 refinement 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 threefold. First, it enhances the OrBAC model by integrating concepts from action specification languages, in order to allow reasoning about policy state evolution as soon as actions are detected. Second, it allows security officers to refine contextual policies over security devices that are either not capable or not allowed to interpret the contextual data. Third, it benefits policy-based mitigation scenarios, such as those used by intrusion detection, fault tolerance, and quality of service processes, in which a policy reconfiguration process must follow the detection mechanism that identified a given event.

Paper organization — Section 2 addresses our motivation. Section 3 surveys related work. Section 4 establishes the core elements of our approach. Section 5 provides the formal definition of our proposal and illustrates a concrete context-aware network scenario build upon our construction. Section 6 presents a practical validation of our approach based on existing tools. Section 7 discusses some open problems and limitations of our proposal. Section 8 concludes the paper.

2. Motivation of our Construction

Our 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. Decisions taken by the PDP 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 the decisions of the PDP. Examples of PEPs are network firewalls, Intrusion Detection Systems (IDSs), and VPN routers for the construction of 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 performs better than the outsourcing mode [22].

The decision made by the PDP generally depends on the activation of some contexts. The concept of context [10, 36] is used here within the specification of security rules which are then triggered when the context becomes active. We list next some examples of contexts. A more complete taxonomy of contexts modeling techniques can be found in [13].

- **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 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 users 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 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 and encryption algorithms).

The contextual techniques modeled in [13] are, however, passive. Indeed, a controller based on such contextual OrBAC model assumes policy evaluation triggered by access requests. It does not allow reasoning about policy state evolution when actions are observed by the system. We aim to enhance the model in order to overcome this limitation. Moreover, let us also observe that PEPs do not necessarily have to provide the functionalities to process the contextual semantics (e.g., spatial, session or intrusion contextual data). It is, hence, fair to consider situations in which the enforcement of contexts by PEPs may prove unsatisfactory for performance reasons; or situations in which some particular contexts containing sensitive data must not be handled by the PEPs. We aim to investigate how to deploy contextual policies related to the provisioning mode when, at least, a PEP is not capable, or allowed, to enforce context-aware policies.

3. Related Work

3.1. Access Control Models

The security policy of an information system may include a wide range of different requirements, such as authentication, authorization, 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. The objective of using a security policy is to represent in an abstract model the set of real-world requirements that are meant to govern the behavior of the system.

There exists in the literature several models in order to describe in a unique and unambiguous manner a security policy. However, not all the models might answer optimally the complete set of security requirements. Some models like DAC (Discretionary Access Control) or MAC (Mandatory Access Control) might require important additional administration efforts to sum up some properties that can, in turn, be satisfactorily described by more complete models, like RBAC (Role Based Access Control) [47]. A proper example is the introduction of notions such as roles and hierarchies. Although

RBAC appears as being restricted to only 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. For instance, TRBAC (Temporal Role-Based Access Control) extends the RBAC model and provides constraints, instants and periodicity characterized by time properties [8]. Similarly, GEO-RBAC extends RBAC providing the necessary mechanisms in order to deal with geographical constraints through absolute and logical representations [9].

The Organization Based Access Control model (OrBAC) natively provides means to express both static and contextual access control requirements [1]. Similarly to RBAC, which comes with an administration model ARBAC [46], OrBAC is associated with AdOrBAC [15] 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 [3, 37]) and, therefore, we choose to formalize the security policy using this model. We refer to section 4.1 for more information about the OrBAC model.

3.2. Policy Based Network Management

The IETF/DMTF settled the terminology of the PBNM architecture in [rfc 3198]. 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. The IETF excluded such a proposal because vendors provide components which behave as either a PDP or a PEP. Moreover, the policy server definition should also include the conflict resolution aspect. The 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 related literature [48]. For instance, 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 corporate network.

Several proposals suggest deploying security policies over security components but without considering contexts. In [31], the security policy is

deployed over *micro-firewalls*, i.e., firewalls assigned to each host in the network. The policy is centralized at the PDP level and can be dynamically changed as a result of an IDS alert. The authors informally compare several 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 [49]. Once a security officer detects a cooperative intrusion attempt, a response is computed. However, neither [31] nor [49] give clear indication about the response strategy or the mapping from alerts to countermeasures.

Many other approaches introduce the notions of *context* and *context-aware* applications in relation with role activation. Proposals like [18] and [19] deal with the deployment of reactive policies to neutralize security threats. In [19], the threats are modeled as contexts. Then, standard IDMEF alerts are mapped to the contexts. In [18], not only contexts but new policy instances are derived as a result of an alert. The authors discuss about context lifetime, according to the impact severity and type of an alert. The response strategy is influenced by the mapping of the alerts towards new security rules. The PDP is always the entity that manages the threat contexts, while the PEP simply includes the enforcement mechanism.

The authors in [12] bring up the notion of *context-aware* applications motivated by applications for *intelligent homes*. In these applications, services are enabled based on the location of subjects or objects. The policy formalization is RBAC-like and the main idea is the activation of roles — the environmental roles which capture the environmental states or contexts, i.e., the locations. It is worth mentioning that this idea can be easily implemented in IPv6 home-networks, in which IPv6 natively provides location information at IP level. The main difference between our approach and the one in [12] is at the access control model level. In [12], as it is in GRBAC (Generalized Role-Based Access Control Model), there is the activation of role and session, whereas in OrBAC we talk about relevant roles in organizations. In OrBAC, the subjects *moving* between organizations (e.g., IP packets flowing through a network composed of security devices — the sub-organizations) are empowered with some relevant roles which are always active. However, authorizations depend on a set of conditions regrouped under the native OrBAC notion of context which is able to gather a large diversity of conditions. A better granularity can thus be achieved, compared to activating roles. The authors also stressed the idea of the easiness in administering their frame-

work. They provide a graphical tool that facilitates the work of the security officer. This is the general trend of current GUI-based industrial applications.

The approach presented in [27] focuses on policy languages enriched with semantics using Web Ontology Languages (OWL) and RDF (Resource Description framework) [45]. It exhaustively compares the use of software engineering approaches such as KAoS, REI and SWRL. The main advantage of such approaches is that a common ontology can be spread between organizations. KAoS uses deontic-logic policies with OWL specifications, offering tools to ensure conflict resolutions. The security architecture is similar to a PDP–PEP one, with the *Guard* acting as a PDP. However, it is unclear how this approach can ensure a total policy deployment whenever the enforcement points lack functionalities. The REI specification language is prolog-like and offers no enforcement model — this is ensured by an external functionality. REI introduces the notion of policy domain, while OrBAC comes with organizations and hierarchies of sub-organizations. SWRL allows the definition of security rules in an XML syntax but still does not provide a robust specification tool and the enforcement is ensured by external functionalities. One can first observe that a semantic security policy language is very descriptive and this is not always compatible with the aim of simplifying the task of a security officer. Though these approaches are suitable for web-services, they are not optimal for IP scenarios. The complexity of IP-based scenarios is an impediment to a centralized administration with real time constraints and frequent policy updates. Nevertheless, it becomes commonly accepted that a security officer has to master formal methodologies (languages or models). Therefore, a strong requirement is a specification and enforcement model with pertaining tools to facilitate this task, as our approach does.

The proposal presented in [33] takes inspiration from the RBAC model. It takes advantage of the notion of role and uses contexts to model the conditions to be enabled for a resource to be worked with. The motivation is represented by the various requirements in securing collaborations of ad hoc coalitions. The main proposals in [33] are: (1) a context modeled with logic programming and (2) an ontology approach to specify policy and contexts. The former allows context activation represented by statements such as: *if context attributes* $\mathrm{C}_1 \ldots \mathrm{C}_n$ *then context attribute* C_m. Compared with the approach that we present in this paper, where we combine the OrBAC and the ECA (Event Condition Action) [6] formalisms, allows going further and integrate operation such as the detection of the precise moment in which a context starts or ends. Regarding the ontology approach in [33] to specify

policy and contexts, it can benefit further operations, such as organizations interoperability. This can also be achieved in our approach while, as it can be seen in [11], the combination of the OrBAC formalism and the expressivenes of general-purpose ontologies, allows a more generic manner of guaranteeing organization interoperability.

4. Core Elements of our Contruction

4.1. The OrBAC Model

The OrBAC model involves two levels of abstraction: (1) an organizational level that includes the *role*, *activity*, *view* and *context* concepts; and (2) a concrete level that includes the *subject*, *action*, *object* concepts. It uses first order logic to write access control rules in the form of permissions (*Is_permitted*), prohibitions (*Is_prohibited*), obligations (*Is_obliged*), and dispensations (*Is_dispensed*). 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)*

The previous expression stands that 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 main 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 5.2. OrBAC defines role hierarchies (as RBAC), and also views, activities and context hierarchies. In the specialization (or generalization)

hierarchy, permissions and prohibitions are inherited in a *downward* manner. These hierarchies facilitate the tasks of the security officer. They also simplify the formalization of the security policy.

In the following sections we describe our main hypotheses intended for the deployment of an OrBAC security policy P defined over a set of elementary contexts denoted by C. We assume that a set of n PEPs, PEP_i, is responsible for enforcing the policy, yet each PEP_i manages only a subset C_i of contexts. According to the OrBAC model we consider the following 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 $\mathrm{C}_1 \wedge \mathrm{C}_2$ 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).

4.2. Main Hypotheses

First, we assume that an OrBAC policy P is managed at the PDP level. The PDP represents the intelligent entity of the system. The PDP itself can manage a subset $\mathrm{C}_{\mathrm{PDP}}$ of contexts in C. For each rule SR_k in P, the set PEP_i of PEPs in charge of enforcing SR_k is known. This assumption is based either (1) on the officer's explicit indication regarding the optimal PEP_i in terms of right functionalities and right emplacement in the system; or (2) based on approaches like[42], in which 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 that 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 $\mathrm{C}_1 \wedge \mathrm{C}_2$. If an entity manages the context C_i, it is also able to manage the context $\neg\mathrm{C}_i$. We believe these hypotheses are not restrictive but fairly reflect a reality.

4.3. Methodology

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. 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 context C.

- *Case 2*: the PEP_i manages only C_2, a part of the context C. In this case, let us denote 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 context C; 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.

4.4. Limitation of the Current Formalism

An early conclusion that can be extracted from this section is that by simply relying on the OrBAC formalism we cannot specify the concrete moment at which a context C specifically starts or ends. Indeed, the classical OrBAC context definition does not include dynamic factors. An OrBAC context holds if a set of logical preconditions hold. However, the dynamism of security requirements and the evolution of environmental parameters, for instance, prove that the OrBAC formalism alone is insufficient. Otherwise, we would

be limited to the single management of static deployments where, a classical OrBAC policy would be deployed once and for all at system initialization. Therefore, an administration tool implementing the downward process of a classical OrBAC policy would prove unsatisfactory. Such administration tool should be able to implement a different policy deployment: deriving security rules correlated with some events that we call policy deployment triggering events. Thus, the deployed rules are continuously adapted to fit the security requirements in the new contexts.

To solve this limitation, we propose to enrich the OrBAC model with a new type of context definition reflecting the activation and the deactivation of a given context C — hereinafter defined as, respectively, Start(C) and End(C). To do so, we propose to combine the proposal presented in this section together with the use of the *ECA* (Event Condition Action) formalism [6]. We present this improvement in the sequel, by formalizing the definition of context activation and triggered actions.

5. Dynamic Activation of Contexts

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 deployment.

5.1. Use of ECA *Rules*

The main concepts in [6] (action, event, and active rule) are defined by using the $\mathcal{L}_{\text{active}}$ language [5]. The $\mathcal{L}_{\text{active}}$ 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. Some examples are provided next in order to show their usage.

1. *Action* definition:
   - $action(\mathrm{X})$ causes $f(\mathrm{Y})$ if $p_1(\mathrm{X}_1)$, ..., $p_n(\mathrm{X}_n)$.

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.

5.2. Dynamic Deployment

In our approach, the occurrence of an ECA event generally corresponds to the activation of a context which is handled by the OrBAC model. 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 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.

5.3. 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 ([17] 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 4.3):

- *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. 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 deployment exist and they take into account the context definition fashion, more exactly the PDP_Ctx part of context.

5.4. Deployment Optimization

In this section, we show how to reduce the number of security rules expected to be deployed over the PEPs.

The most convenient 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. The $hold_{\emptyset}$ 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. The $hold_S$ 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.

5.5. Applying our Construction to a Concrete Network System

In order to illustrate our proposal, let us consider the case of a corporation with a security policy including contextual requirements. The architecture in Figure 1 depicts several sub-networks guarded by firewalls (two of them including IPsec functionalities):

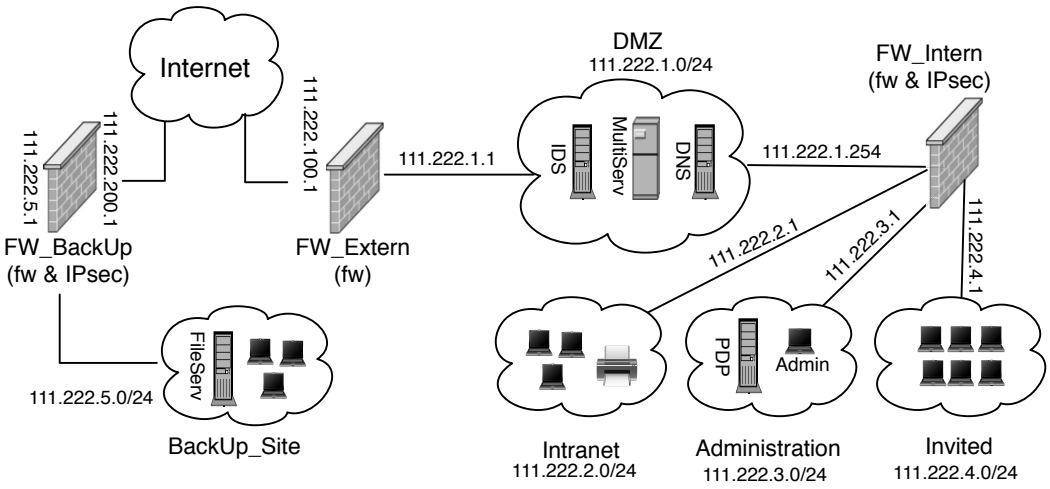

Figure 1: Network architecture of our example

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

- 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 *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

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. $\wedge$ in-intra)*
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.))*

channel with the *BackUP_Site*, i.e., an IPsec channel is required);

- 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 Tables 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 subnetwork 111.222.0.0/16 are empowered in. *R_Internet* corresponds to $0/0 \setminus 111.222.0.0/16$ (meaning that we remove from the whole internet range [0.0.0.0, 255.255.255.255] the subnet [111.222.0.0, 111.222.255.255])). *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 security officer 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 our proposed construction. 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 deployment of the related security rules. Table 4 resumes the definition of these contexts and the significant ECA rules which are automatically generated.

Table 4: Context definition and ECA rules

(1) 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. Function *path(s, o)* returns the list of PEPs ([PEP_i]) on the IPsec tunnel path between *s* and *o*.

Start / End context	**ECA rule generation**
$hold_{so}$*(Intra, BackUp_Site, Start(PDP_prot))* **after** *boot(PEP_1, PEP_n)* **if** *start_path(PEP_1,Intra,BackUp_Site),* *end_path(PEP_n,Intra,BackUp_Site).* $hold_{so}$*(Intra, BackUp_Site, End(PDP_prot))* **after** *halt(PEP_1, PEP_n)* **if** *start_path(PEP_1,Intra,BackUp_Site),* *end_path(PEP_n,Intra,BackUp_Site).*	$hold_{so}$*(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).*

(2) working-hours (w.h.) — temporal context; managed either by PEPs with a *temporal* functionality or by the PDP (we consider the latter case).

Start / End context	**ECA rule generation**
$hold_{\emptyset}$*(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.).*

(3) in-intra — spatial context managed by the PDP.

Start / End context	**ECA rule generation**
$hold_{\emptyset}$*(Start(in-intra))* **after** *enter(person,* *Intra_zone)* **if** *Nb_persons(Intra_zone, 0).* $hold_{\emptyset}$*(End(in-intra))* **after** *exit(person,* *Intra_zone)* **if** *Nb_persons(Intra_zone, 1).*	$hold_{\emptyset}$*(Start(in-intra))* **initiates** *activate(PEPi,SR(R_Intra,TCP,R_Internet))* **if** *SR_version(PEP_i,R_Intra,TCP,* *R_Internet, in-intra).*

(4) 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.

Start / End context	**ECA rule generation**
$hold_{so}$*(S, mailServ, Start(m.l.f.i))* **after** *fails(S, auth, mailServ)* **if** *Nb_fails(S, 2).*	$hold_{so}$*(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).*

(5) 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 is raised; *s.f.* deactivation after a few minutes (e.g., eight minutes).

Start / End context	**ECA rule generation**
$hold_{o}$*(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).*

6. Practical Validation of our Approach

In the previous sections, we have seen the formalization of our proposal. In this section, we present a prototype setup based on two available open source frameworks that we use to test the practical viability of our proposal. The first framework, MotOrBAC [37, 3], allows the definition and validation of an OrBAC-based network access control policy. It also provides the means to process the downward transformation of the formal policy into the concrete configuration language of a PEP (e.g., a NetFilter-based firewall). The second framework, EnSuite [21], provides the specific communication protocol between the PDP and the PEP of our tests.

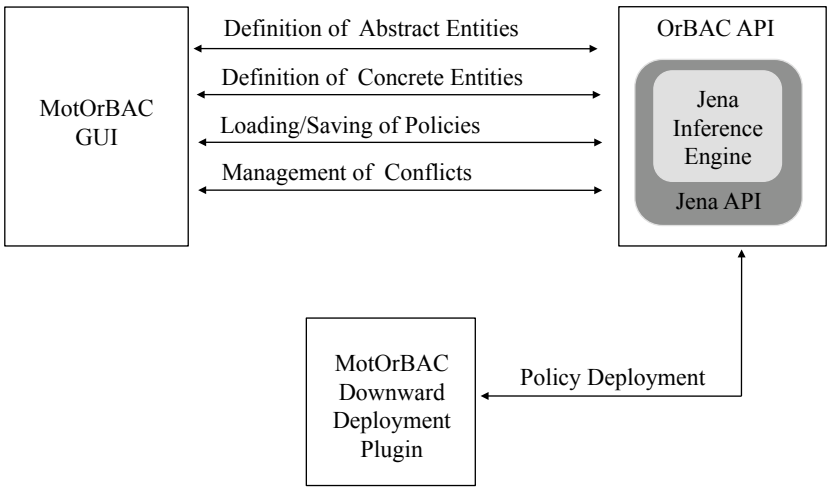

Figure 2: The MotOrBAC architecture

Figure 2 illustrates the MotOrBAC framework architecture. The OrBAC API (Application Programming Interface) is used to manage the policies displayed in the MotOrBAC GUI (Graphical User Interface). The API uses the Jena Java library [32] to represent and infer knowledge based on RDF (Resource Description framework) graphs [45]. Jena provides several inference engines. The specific reasoner used by the OrBAC API is the generic rule reasoner, a general purpose inference engine that supports rule-based inference over RDF graphs. Such a reasoner supports both forward and backward inference chaining, as well as combinations of both strategies. The generic

rule reasoner uses the associations between abstract and concrete entities and infers the concrete policy. The transformation from abstract concepts towards concrete policies is actually used in order to execute the downward transformation depicted in Figure 3. This process uses a two-step transformation. In the first transformation, the abstract OrBAC policy is refined into a generic set of security rules. These rules are still independent of any specific PEP technology. The second transformation results in a specific PEP configuration (e.g., NetFilter-based firewall rules). A detailed description about the two-step transformation process depicted in Figure 3 can be found in [16, 42]. The complete OrBAC API, as well as the MotOrBAC tool and the two-step transformation plugin described in the paper has been developed by our research team and are all available on-line [37].

Figures 4—7 illustrate the main functions of the MotOrBAC GUI used in our tests. These functions can be used by a security officer in order to define both the abstract OrBAC policy and the concrete network entities. Figure 4 depicts the abstract entities of a policy. Figure 5 depicts the concrete entities associated with the same policy. Also, by using the policiy editor of the MotOrBAC GUI, the security officer can assign permissions, prohibitions, and obligations to the abstract entities, as well as define contextual data. Once all the definitions are completed, the officer can now proceed with the downward transformation plugin to derive the complete set of NetFilter-based

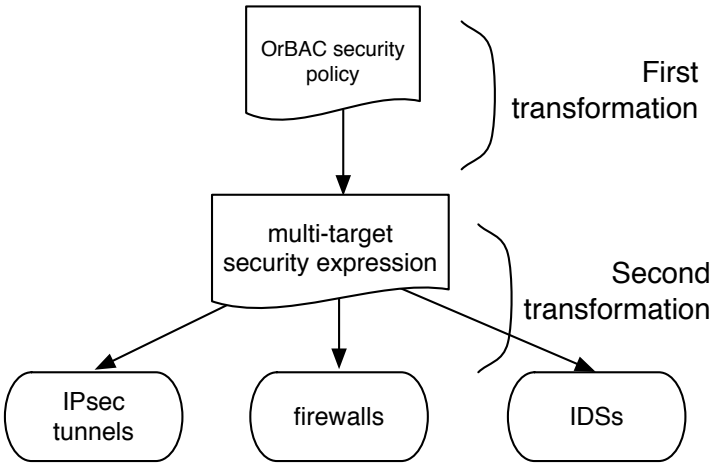

Figure 3: Downward transformation of an OrBAC security policy

firewall rules (cf. Figure 6). The downward transformation plugin does only infer rules associated with active contexts. This process can be verified by accessing the MotOrBAC simulation feature that analyzes the state of every context (cf. Figure 7). The following sketch sample shows a concrete package of rules derived from the abstract policy depicted in the previous examples:

```
iptables -N intra_to_net

iptables -A FORWARD -s 111.222.2.0/24 -p tcp -j intra_to_net

iptables -A intra_to_net –src 111.222.2.1 -j RETURN
iptables -A intra_to_net –src 111.222.2.54 -j RETURN
iptables -A intra_to_net -m iprange –dst-range \
                              111.222.0.0-111.222.255.255 -j RETURN
iptables -A intra_to_net -j ACCEPT
```

Regarding the communication protocol between a PDP and one or more PEPs, several communication protocols can address this issue. For example, the Simple Network Management Network (SNMP) protocol [rfc 1157], the Common Open Policy Server (COPS) protocol [rfc 2748], and the NetConf protocol [rfc 4741] are, actually, solutions that are easy to find implemented in most of todays' support tools. They appear, moreover, in the related literature in order to implement similar operations as the deployment method that we discussed in our work. The SNMP protocol is one of the most widely used protocols by network management tools. This protocol relies on a traditional agent-manager paradigm, in which the agent is basically a program running over the equipments of a system; and the manager is a program centralized by a platform that controls and collects the execution results of each agent over the system. Most solutions of well known vendors like Cisco or Check Point base the synchronization and management of their tools on SNMP or other variants. In contrast, the COPS and NetConf protocols are essentially of a query-response nature. They have also been reported in recent literature aiming at exchanging policy information between servers and clients. COPS have been used in [26], for example, to dynamically distribute IPsec-based VPN policies. The main limitation reported in [22] seems to be

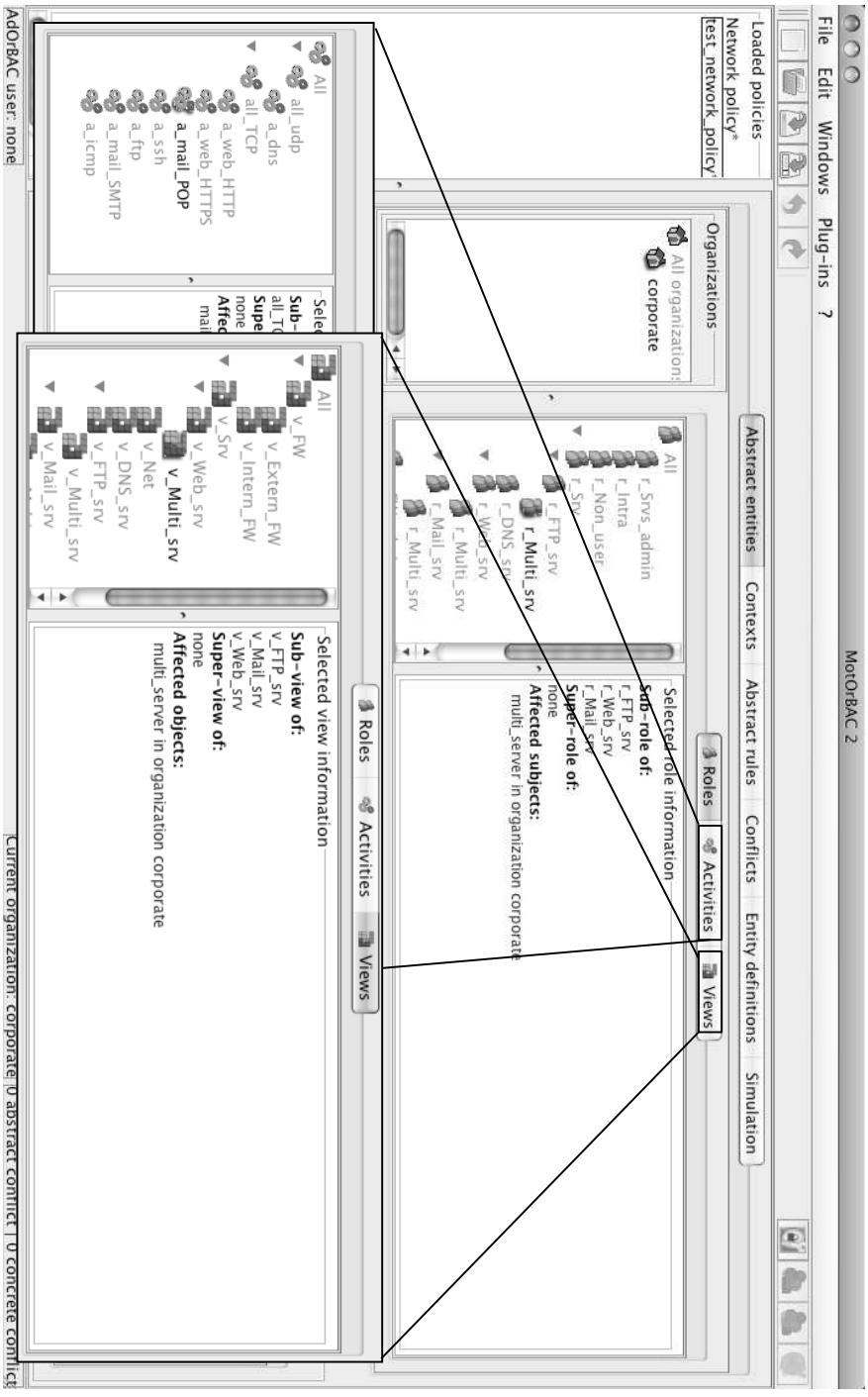

Figure 4: Definition of the abstract entities of an OrBAC network policy using the MotOrBAC GUI

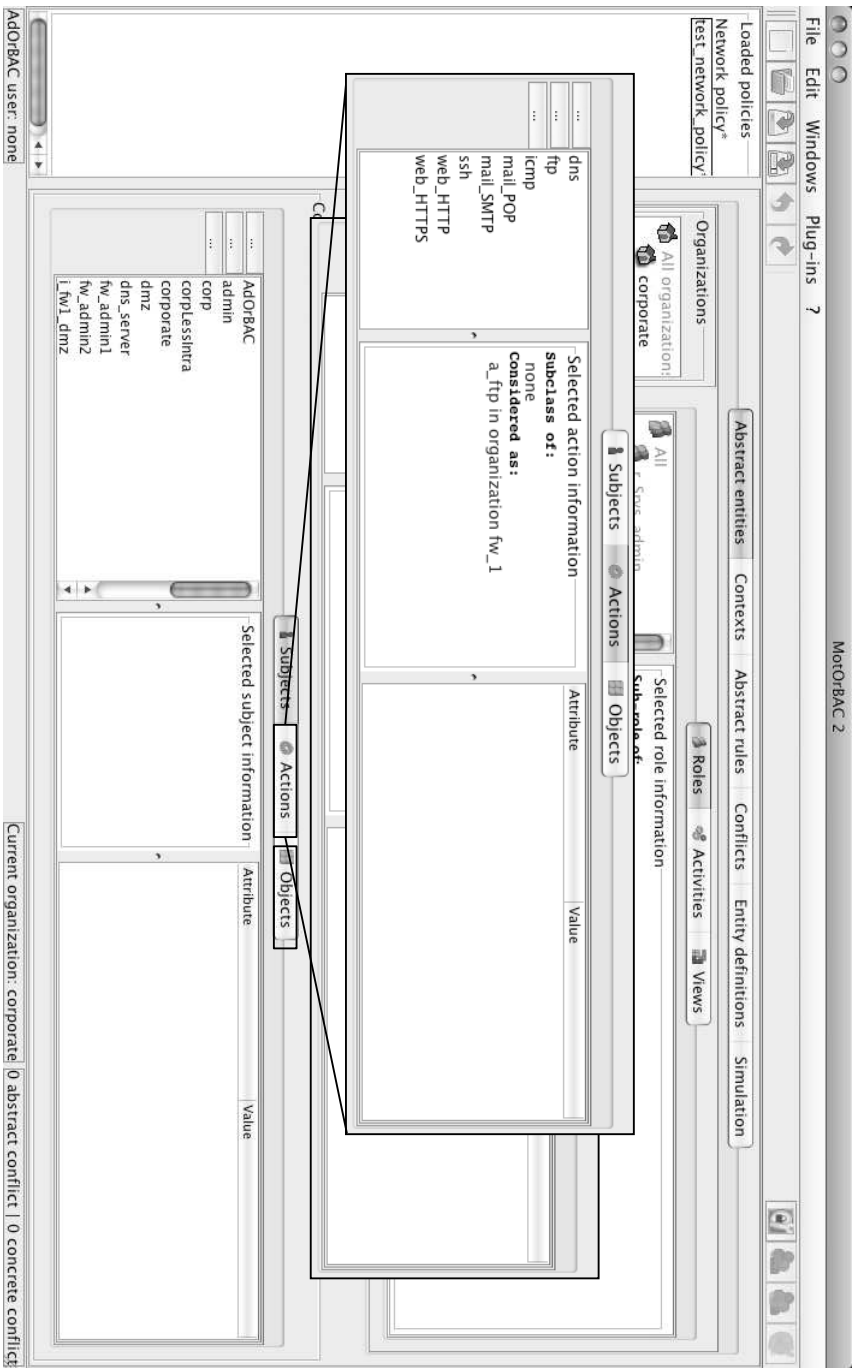

Figure 5: Definition of the concrete entities of an OrBAC network policy using the MotOrBAC GUI

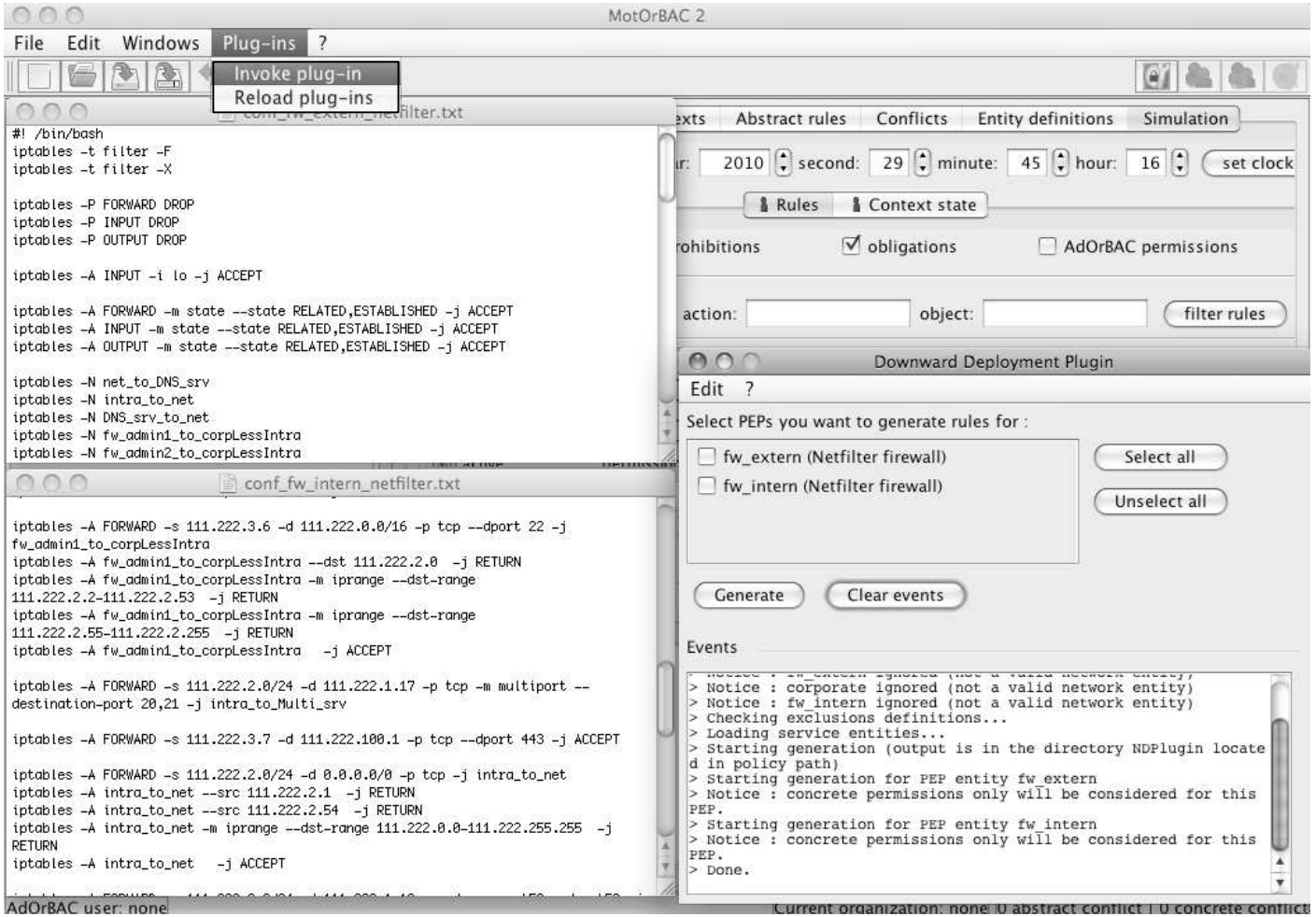

Figure 6: Downward transformation of the policy

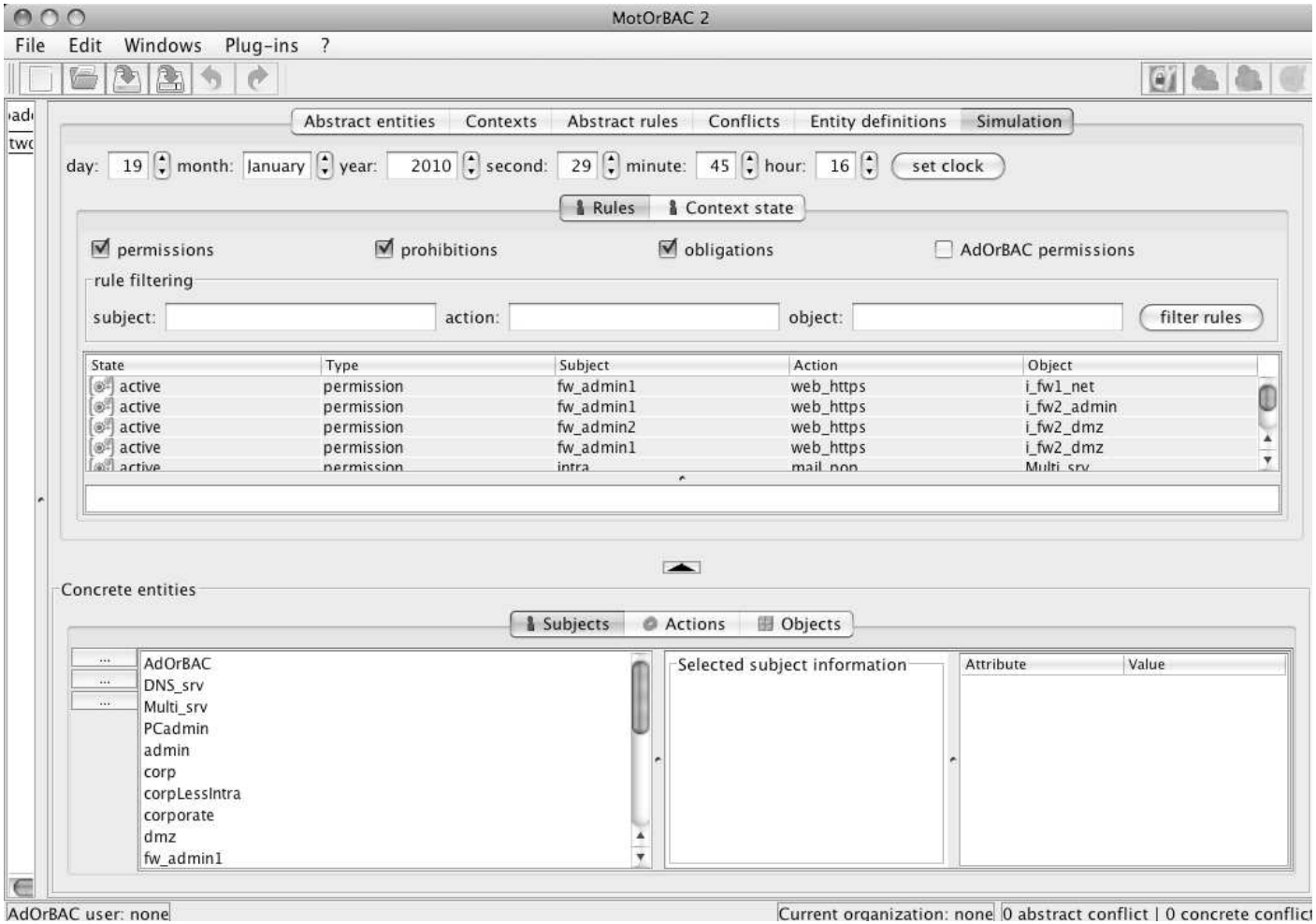

Figure 7: Verification of active contexts

the handling of XML encoding, as it is not natively handled by COPS, rather than other protocols like NetConf. For the experimental evaluation of our approach, we propose the use of the NetConf protocol to deploy and communicate the packages of rules between PDPs and PEPs. For this purpose, we use the EnSuite framework to push each corresponding PEP configuration via its NetConf protocol implementation. The EnSuite framework mainly consists of a NetConf web-based manager, a NetConf agent and a set of extension modules. All these components are implemented in Python. The two main modules are the Yencap-Manager and the Yencap-Agent. On the one hand, the Yencap-Manager offers a management application built as web-based GUI. On the second hand, the yencap-agent consists of the NetConf agent implementation. It supports the addition of new modules as well as new operations. In this sense, we implemented an OrBAC module as a new extension module for NetConf that demonstrates the feasibility of our approach via a provisioning module which we use to remotely configure (i.e., update) the policies of NetFilter-based firewall PEPs. We show the main interface of the EnSuite components prior and after the update of PEPs configurations in Figure 8.

Motivated by the latency that our approach may impose over real case scenarios, we evaluate the communication delay of deploying an incremental set of temporal filtering policies derived from the aforementioned prototype setup. We assume the following two scenarios during our evaluation: a best case scenario that comprises the deployment of the policy over a LAN (Local Area Network); and a worst case scenario where the policies are deployed on a WAN (Wide Area Network). The following two experiments are carried out: (1) evaluation of the communication latency during the deployment of the temporal filtering policies from the EnSuite PDP towards a NetFilter-based PEP; and (2) evaluation of bandwidth degradation due to the overhead imposed by setting equivalent rules on a NetFilter-based PEP. The set of rules are derived from the MotOrBAC framework and transmitted from the Yencap-Manager to the Yencap-Agent that is executed at the NetFilter-based PEP. All the rules correspond to the following template: *iptables -A FORWARD -i $Iface -p $Protocol -s $SrcNet -d $DstNet –sport $SrcPorts –dport $DstPorts -j ACCEPT*. The following table shows the size assigned to each data set depicted in Figure 9:

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

Figure 8: Main interface of the EnSuite components prior and after the configuration update of a NetFilter-based firewall PEP

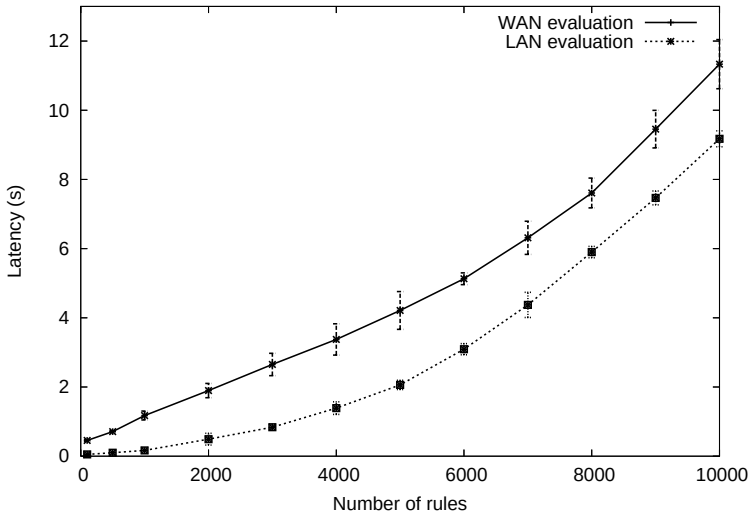

Figure 9: Communication latency during the deployment of temporal filtering policies

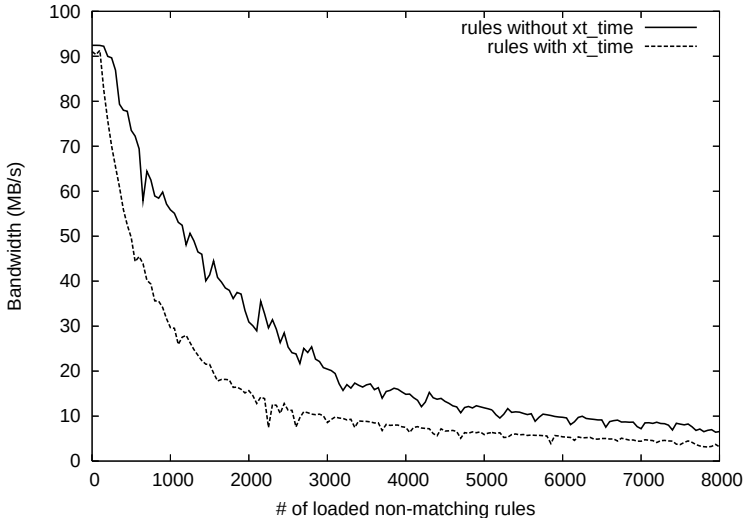

Figure 10: Bandwidth degradation due to the consecutive loading of NetFilter rules

We can appreciate by looking at the curves of Figure 9 that the average time for 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.

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 10 a practical evaluation that proves our claim.

7. Open Problems and Limitations

The Organization-Based Access Control (OrBAC) model, first presented in [1], stems from the work carried out by members of the MP6/RNRT project (funded by the French Ministry of Research). OrBAC provides a robust modeling solution to address access, network and usage control policies in traditional IT systems. OrBAC suggests a very expressive and modular formalism that enables policy administrators to make a distinction between the policy and its concrete implementation [16]. This is obtained by making an abstraction of the traditional access control entities subject, action and object. While traditional access control models focus on modeling users and roles only (e.g., the RBAC model [47]), OrBAC adds an abstract view of actions

and objects using two new concepts: activity and views. This way, in OrBAC, subjects are empowered in roles, objects are used in views and actions implement activities. The concept of organization in OrBAC provides means to better analyze interoperability and specification of hierarchies which, in turn, leads to a flexible specification of collaborative work and information flow between different organizations (e.g., companies and institutions). The OrBAC formalism and the expressiveness of general-purpose ontologies can be combined in order of guaranteeing organization interoperability and collaboration [11, 2]. Negative authorizations are also allowed in OrBAC, in order to specify complex policies. As conflicts might occur between positive and negative authorizations, OrBAC provides conflict management strategies to detect and resolve such potential conflicts [14]. OrBAC is robust in terms of administration and pertaining tools (cf. OrBAC API, AdOrBAC administration, MotOrBAC tool and available plugins [3, 15, 37]).

The applicability of the OrBAC model for policy deployment has already been proved in previous work such as [16, 42, 34, 29]. In this paper, and compared with approaches in [16, 42, 34], we added the management of contextual security semantics, and address network access control policies rather that software-based policies (as the work in [29] does). The motivation goal was to deploy contextual policies over enforcement points which do not embed the right security functionalities necessary to handle the contexts. Our methodology aims to dynamically deploy concrete policies by relying on a central entity — the policy decision point — in charge of managing the context whenever the enforcement points lack the required functionalities. In order to establish the formal framework, we extend passive controllers based on static OrBAC policies (whose evaluation is triggered by access requests) into proactive controllers that integrate concepts from action specification languages. The final goal is to ensure a complete policy deployment.

We can, however, identify a few limitations of our current work. First of all, the context activation and deactivation trigger serious computations at PDP level. With a high frequency, these events may lead to situations where the PEPs are not updated in real time. Moreover, this may introduce serious vulnerabilities in the system, since an adversary can use such events to launch DoS attacks over the PDP. For example, in IPv6 networks the *binding-updates* packets could be defined as the event to activate a context of mobility, leading to the deployment of some firewall rules in the foreign network. An attacker can forge such IPv6 packets with the result of useless PDP computations and filtering rules. A solution to such problem is either (1) distributing the

intelligence of the PDP so that the events of activating (or deactivating) the contexts be correlated and that the PDP uses reliable triggering events or (2) delaying the PDP decisions. The latter idea is not totally new. E.g., certain Radius server implementations, such as Freeradius [23], use a reject_delay parameter to delay the Access-Reject messages in order to slow down DoS attacks. Certainly, the drawback of this solution is the lack of *real-time* deployments. Another limitation is identified at the choice of the PDP–PEP protocol. NetConf uses a tree representation of policies, but our approach considers a total redeployment/re-computation of the access control policy. The advantage of our solution is that the redeployed policies are free of anomalies (the detection of policy conflicts is ensured during the downward process). An improvement would be to consider the redeployment of only the *interesting* parts of the access control policy, with a positive result — a faster redeployment.

Another interesting improvement would be the refinement of policies in scenarios where PDPs and PEPs are not working together in the management of a certain context. There are few approaches in the literature that provide solutions to such a case; and they generally apply optimization methods. For instance, in [28] the authors choose the best combination of security functionalities that minimize the system's vulnerabilities. Their work is meant to select the most appropriate security technologies to address the vulnerabilities in the system. Each technology is considered to match a certain sub-set of vulnerabilities but potentially introduces new ones. A real value in the interval $[-1; 1]$ is assigned to each technology-vulnerability couple: -1 if the technology introduces a vulnerability, 1 if the vulnerability is solved or 0.5 if the vulnerability is only partially addressed (or partially created). A mapping of the targeted technologies to vulnerabilities is therefore manually defined and given that each technology has a normalized cost, the objective function is intended to maximize the vulnerability coverage while maintaining a minimum cost of the new technologies and also a minimum set of newly created (residual) vulnerabilities. The optimization problem is solved using a genetic algorithm. As we have notated, our approach is slightly different since we deal with the deployment of policies. However, our current approach can be improved with an *always best deployed* pattern for the already existing functionalities in the system: a solution consisting in deploying the policy by using either a context closely related to the unmanaged one or certain security functionalities related to the missing ones. The final result (i.e., the deployed policy) should always meet the initial assurance requirements.

We would not map vulnerabilities or security requirements to functionalities but we could consider a context functionalities mapping, the objective being to find the best set of technologies to support each requirement. Another paradigm is presented in [20]: the use of attack-tree representations to address the network policy enforcement problem. The authors in [20] propose to quantify the damage following an attack and the cost related to the implementation of new security controls; both quantifications follow the same cost model. The optimization problem, solved with a genetic algorithm, is to find a minimum security cost that corresponds to a minimum damage.

The solution proposed in [30] employs attack graphs which are obtained using the MulVal tool [40]. The attack graph is reduced to a first-order logic formula which, in conjunction with a security policy (expressed as a conjunction of privileges that must not be acquired by the attacker) and with a security usability (stating those network configurations that should never be altered whatever the attacks may be), represents an unsatisfying formula. Costs are associated with those variables describing (1) changes in the system configuration and (2) privileges acquired by an attacker. Using two SAT techniques – UnSAT core elimination [35] and MinCostSAT [50] – a satisfying solution is derived that represents minimum costs. Our approach is different from those using attack graph models: we deploy a policy based on a more abstract model than the one dealing with attacker's privileges. Our methodology is placed prior to the attack graph applications and the result of our deployment process could be jointly used with such approaches in [20, 30]; our deployment would clearly influence the attack graph instantiation.

Regarding the use of similar strategies on industrial solutions based on environmental constraints, such as Network Access Servers, WiFi Controllers, Session Border Controllers, and to our knowledge, there is no solution proposing a top-down approach leading to a formally proved deployment without anomalies. The experience shows that today's (security) administrators work with too many subjects and not enough roles, activities or views, leading thus to a large number of rules (e.g., security rules, security remote access strategies, and backup strategies). Integrating an OrBAC-based policy representation with an ECA-like top-down deployment process would considerably simplify the administration task. However, it is a fact that the current industrial applications still implement basic concept since administering a high-level model for example (in security, backup, data base policy areas) requires nevertheless solid expertise of the model.

8. Conclusion

Contextual access control policies provide the means to handle complex security system requirements in a flexible and dynamic manner. However, PEPs (Policy Enforcement Points), such as firewalls, Intrusions Detection Systems (IDSs), and VPN routers, do not necessarily have the required functionalities to manage the expressiveness of these policies. It is also fair to envision situations in which the enforcement of contexts by PEPs may prove unsatisfactory for performance reasons; or situations in which some particular contexts containing sensitive data must not be handled by the PEPs. We presented in this paper an approach to handle these scenarios.

Our proposal enhances the OrBAC (Organization-Based Access Control) model and allows security officers to dynamically deploy contextual OrBAC policies over PEPs that are not allowed to interpret contextual semantics. The approach has been implemented and evaluated. We have presented the performance of the communication between PDPs (Policy Decission Points) and PEPs based on our approach and the use of the NetConf protocol. We compared the overhead of deploying firewall rules without contextual constraints towards the use of firewall rules including those constraints. Complex scenarios will grasp the benefits of our approach and the tasks of a security officer are considerably simplified.

The contribution of our work may also benefit policy-based reaction scenarios, such as those used by intrusion detection processes [4]. 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. Fault tolerance and quality of service scenarios may also benefit from our work. In these scenarios, PDPs are aware of the policy of a system as a whole, in order to 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.

Acknowledgments

This work has been supported by the French National Research Agency (SELKIS project ANR-08-SEGI-018), the Spanish Ministry of Science (grants

TSI2007-65406-C03-03 E-AEGIS and CONSOLIDER-INGENIO 2010 CSD-2007-00004 ARES), and the European Commission (DEMONS project ICT FP7-257315). The authors thank Xavier Rimasson and Fabien Autrel for all their help on the prototype setup used in Section 6. The authors also thank the anonymous reviewers for their constructive and insightful remarks that helped in improving this paper.

References

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

[2] Abou el Kalam, A., Deswarte, Y., Baïna, A., Kaâniche, M., 2009. PolyOrBAC: A Security Framework for Critical Infrastructures. *International Journal on Critical Infrastructure Protection*, Springer, Vol. 2(4):154–169.

[3] Autrel, F., Cuppens, F., Cuppens-Boulahia, N., and Coma, C., 2008. MotOrBAC 2: A security policy tool. In *3rd Joint Conference on Security in Network Architectures (SAR) and Security of Information Systems (SSI)*, Loctudy, France, Editions Publibook, pp. 273–288.

[4] Autrel, F., Cuppens-Boulahia, N. Cuppens, F., 2009. Reaction Policy Model Based on Dynamic Organizations and Threat Context. In *24th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy (DBSec 2009)*, Canada, Lecture Notes in Computer Science, Springer, Vol. 5645, pp. 49–64.

[5] Baral, C. and Lobo, J., 1996. Formal Characterization of Active Databases. In *International Workshop on Logic in Databases (LID'96)*, Italy, Lecture Notes in Computer Science, Springer, Vol. 1154, pp. 175–195.

[6] Baral, C., Lobo, J., and Trajcevski, G., 1997. Formal Characterization of Active Databases: Part II. In *5th International Conf. on Deductive and Object-Oriented Databases*, Montreux, Switzerland, Lecture Notes in Computer Science, Springer, Vol. 1341, pp. 247–264.

[7] Bartal, Y., Mayer, A., Nissim, K., and Wool, A., 1999. Firmato: A Novel Firewall Management Toolkit. In *20th IEEE Symposium on Security and Privacy*, Oakland, CA, USA, IEEE, pp. 17–31.

[8] Bertino, E., Bonatti, P. A. and Ferrari, E., 2001. TRBAC: A Temporal Role-based Access Control Model. In *ACM Transactions on Information and System Security (TISSEC)*, ACM, 4(3):191–233.

[9] Bertino, E., Catania, B., Damiani, M. L. and Perlasca, P., 2005. GeoRBAC: a Spatially Aware RBAC. In *10th ACM Symposium on Access control Models and Technologies*, Stockholm, Sweden, ACM, pp. 29–37.

[10] Cholewka, D. G. , Botha, R. A. , and Eloff, J. H. P. 2000. A Context-sensitive Access Control Model and Prototype Implementation. In *IFIP TC 11 16th Annual Working Conference on Information Security*, Beijing, China, IFIP Conference Proceedings, Vol. 175, pp. 341–350.

[11] Coma, C., Cuppens-Boulahia, N., Cuppens, F., and Cavalli, A., 2008. Context Ontology for Secure Interoperability. In *3rd International Conference on Availability, Reliability and Security (ARES 2008)*, Barcelona, Spain, IEEE, pp. 473–486.

[12] Convington, M., Long, W., Srinivasan, S., Dev, A. K., Ahamad, M., and Abowd. G. D., 2001. Securing Context aware Applications Using Environment Roles. In *6th ACM symposium on Access control models and technologies*, Chantilly, Virginia, ACM, pp. 10–20.

[13] Cuppens, F. and Cuppens-Boulahia, N., 2008. Modeling Contextual Security Policies. In *International Journal of Information Security*, Springer, 7(4):285–305.

[14] Cuppens, F., Cuppens-Boulahia, N., and Ben-Ghorbel, B., 2007. High-level conflict management strategies in advanced access control models. In *Electronic Notes in Theoretical Computer Science (ENTCS 2007)*, Elsevier, Vol.186, pp. 3–26.

[15] Cuppens, F. and Miège, A., 2003. Administration model for OrBac In *Workshops of OTM 2003, On The Move to Meaningful Internet Systems*, Italy, Lecture Notes in Computer Science, Springer, Vol. 2889, pp. 754–768.

[16] Cuppens, F., Cuppens-Boulahia, N., Sans, T. and Miège, A., 2004. A Formal Approach to Specify and Deploy a Network Security Policy. In *2nd Workshop on Formal Aspects in Security and Trust*, Toulouse, France, IFIP Conference Proceedings, Vol. 173, pp. 203–218.

[17] Cuppens-Boulahia, N., Cuppens, F., Abi Haidar, D., and Debar H., 2008. Negotiation of Prohibition: An Approach Based on Policy Rewriting. In *23rd International Information Security Conference (SEC 2008)*, Italy, IFIP Conference Proceedings, Vol. 278, pp. 173–187.

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

[19] Debar, H., Thomas, Y., Cuppens, F. and Cuppens-Boulahia, N., 2006. Using Contextual Security Policies for Threat Response. In *Third GI International Conference on Detection of Intrusions & Malware, and Vulnerability Assessment (DIMVA)*, Berlin, Germany, Lecture Notes in Computer Science, Springer, Vol. 4064, pp. 109–128.

[20] Dewri, R., Poolsappasit, N., Ray, I., and Whitley, D., 2007. Optimal security hardening using multi-objective optimization on attack tree models of networks. In *14th ACM conference on Computer and communications security (CCS'07)*, ACM, pages 204–213.

[21] Ensuite framework. `http://ensuite.sourceforge.net/`. Accessed November 3, 2011

[22] 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. 2006. 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)*, IEEE, pp. 195–204.

[23] Freeradius project. `http://freeradius.org/`. Accessed November 3, 2011

[24] Garcia-Alfaro, J., Cuppens, F., and Cuppens-Boulahia, N., 2006. Analysis of Policy Anomalies on Distributed Network Security Setups. In *Eu-*

ropean Symposium On Research In Computer Security (Esorics), Hamburg, Germany, Lecture Notes in Computer Science, Springer, Vol. 4189, pp. 496–511.

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

[26] Garcia-Clemente, F. J., Lopez, G., Martinez, G., and Gomez-Skarmeta, A. F., 2005. 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.

[27] Garcia-Clemente, F. J., Perez, G. M., Botia-Blaya, J. A., Gomez-Skarmeta, A.F., 2005. Representing Security Policies in Web Information Systems. In *14th International WWW Conference Policy Management for the Web (PM4W)*, Chiba, Japan, IEEE, pp. 10–16.

[28] Gupta, M., Rees, J., Chaturvedi, A., and Ch, J., 2006. Matching information security vulnerabilities to organizational security profiles: a genetic algorithm approach. In *Decision Support Systems*, volume 41, pages 592–603.

[29] He, R., Lacoste, M., Pulou, J., Leneutre, J., 2010. A DSL for Specifying Autonomic Security Management Strategies. In *3rd International Workshop on Autonomous and Spontaneous Security (SETOP 2010)*, Athens, Greece, Lecture Notes in Computer Science, Springer, Vol. 6514, pp. 216–230.

[30] Homer, J. and Ou, X., 2009. SAT-solving approaches to contextaware enterprise network security management. In *IEEE Journal on Selected Areas in Communications*, 27(3):315–322.

[31] Hwang, K. and Gangadhran, M., 2001. Micro-Firewalls for Dynamic Network Security with Distributed Intrusion Detection. In *IEEE International Symposium on Network Computing and Applications*, IEEE, pp. 68–79.

[32] Jena Java Library. `http://jena.sourceforge.net/`. Accessed November 3, 2011

[33] Kagal, L., Montanari, R., Lassila, O., Toninelli, A., 2006. A Semantic Context-aware Access Control Framework for Secure Collaborations in Pervasive Computing Environments. In *5th International Semantic Web Conference (ISWC 2006)*, Athens, GA, USA, Lecture Notes in Computer Science, Springer, Vol. 4273, pp. 473–486.

[34] Li, K., Mounier, L., Groz, R., 2007. Test Generation from Security Policies Specified in Or-BAC. In *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, IEEE, Beijing, China, pp. 255–260.

[35] Mahajan, Y. S., Fu, Z., and Malik, S., 2004. Zchaff2004: An efficient SAT solver. In *7th international conference on Theory and applications of satisfiability testing (SAT 2004)*, Springer, pp. 360–375.

[36] McDaniel, P., 2003. On Context in Authorization Policy. In *8th ACM Symposium On Access Control Models and Technologies (SACMAT 2003)*, Como, Italy, ACM, pp. 80–89.

[37] MotOrBAC website. `http://motorbac.sourceforge.net/`. Accessed November 3, 2011

[38] Narayanan, A., and Shmatikov, V., 2008. Robust De-anonymization of Large Sparse Datasets. In *IEEE Symposium on Security and Privacy 2008*, Oakland, CA, USA, IEEE, pp. 111–125.

[39] NetConf working group. `http://ops.ietf.org/netconf/`. Accessed November 3, 2011

[40] Ou, X., Govindavajhala, S., and Appel, A. W., 2005. MulVAL: A logic based network security analyzer. In *In 14th USENIX Security Symposium*, Volume 14, pp. 8–23.

[41] Permis project. `http://sec.cs.kent.ac.uk/permis/`. Accessed November 3, 2011

[42] Preda, S., Cuppens, F., Cuppens-Boulahia, N., Garcia-Alfaro, J., and Toutain, L., 2007. Reliable Process for Security Policy Deployment. In *International Conference on Security and Cryptography*, Spain, pp. 5–15.

[43] Preda, S., Cuppens, F., Cuppens-Boulahia, N., and Toutain, L., 2007. Architecture-Aware Adaptive Deployment of Contextual Security Policies. In *5th International Conference on Availability, Reliability and Security (ARES 2010)*, Krakow, Poland, IEEE, pp. 87–95.

[44] Preda, S., Cuppens-Boulahia, N., Cuppens, F., Garcia-Alfaro, J., Toutain, L., 2010. Model-Driven Security Policy Deployment: Property Oriented Approach. In *Second International Symposium on Engineering Secure Software and Systems*, Pisa, Italy, Lecture Notes in Computer Science, Springer, Vol. 5965, pp. 123–139.

[45] Resource Description Framework. `http://www.w3.org/RDF/`. Accessed November 3, 2011

[46] Sandhu, R., Bhamidipati, V., and Munawer, Q., 1999. The ARBAC97 Model for Role-Based Administration of Roles. In *ACM Transactions on Information and System Security*, 2(1):105–135.

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

[48] Strassner, C. J., 2003. *Policy-based Network Management, Solutions for the Next Generation.* Morgan Kaufmann Publishers, 1rst edition.

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

[50] Xiao, Y. L., 2004. Optimization Algorithms for the Minimum-Cost Satisfiability Problem. PhD thesis, North Carolina State University.