

# Firewall Configuration based on Specifications of Access Policy and Network Environment

A. Titov, V. Zaborovsky

Saint-Petersburg State Polytechnical University, Russia  
avt@npo-rtc.ru, vlad@neva.ru

**Abstract** – *A network firewall is a widespread means to enforce a security policy, however it remains a network device. Such a duality has caused a somewhat independent way of firewall development from other security management methods. Following the way firewall vendors are focused on performance problems while management issues don't receive enough attention. Firewalls have grown to complicated computer systems, but there is no general high-level programming language for them. This problem becomes more and more urgent due to the increasing complexity of modern security policies, which must be enforced by firewalls. In this paper we 1) propose the basic idea of firewall configuration based on specifications of an access policy and a network environment; 2) describe Organization Based Access Control (ORBAC) model, which is used to specify an access policy; 3) propose a model to specify a network environment, and 4) the method of integration of an access policy with a network environment.*

**Keywords:** access control, firewall, security policy, access policy, Organization Based Access Control.

## 1 Introduction

A network firewall is a widespread means to enforce a security policy, however it remains a network device. Such a duality has caused a somewhat independent way of firewall development from other security management methods. Following the way firewall vendors are focused on performance problems while management issues don't receive enough attention.

Firewalls have grown to complicated computer systems, but there is no general high-level programming language for them. This problem becomes more and more urgent due to the increasing complexity of modern security policies, which must be enforced by firewalls. Each firewall vendor suggests various tools for convenient editing of firewall rules. Such tools are still attached to a specific firewall architecture so they are just practical solutions and rather faint and scattered attempts to solve the whole problem.

There are many works devoted to problems that are the result of poorly thought-out firewall management ([1][2][3][4]). They search for inconsistencies among single firewall filtering rules or among several distributed firewalls. Filtering rules are treated there as something entirely self-sufficient but not as just what follows from a security policy.

Some works suggest general languages to specify the configuration of a system of distributed firewalls ([3][4]). These languages are powerful enough, may suggest convenient high-level management, but such solutions continue to increase the gap between general security management and firewall management.

After all, this work is related to those ones that explicitly distinguish between security and network specificity in a firewall configuration ([5][6][7][8]). Such an approach is more difficult to implement just at the beginning while is more promising hereafter.

## 2 Basic Approach to Firewall Configuration

A firewall configuration is based on a security policy. In order to automate this process the policy needs a formalized model to specify it. It is not difficult to conclude that such a model must have a great expressiveness as there are many various issues covered by modern policies. Therefore, the effective utilization of the model on practice is disputable.

However, the complexity of firewall configuration is mainly its filtering rules, which follow from an access policy. An access policy in contrast to a security policy may just specify some requirements, e.g., “*Alex cannot talk to Bob*” without the refinement of the nature of Alex, Bob, etc. The superficiality of access policies preserves their simpleness, permits of their formalization (they are initially formalized) and effective utilization on practice.

Thus, firewall filtering rules may be automatically derived from a formalized access policy while this policy must be manually specified by an expert on the basis of an overall but still informal security policy. Of course, the last transition is still not automated, but this task may be being solved independently without references to firewalls or other tools for policy enforcement.

It would be true to say that filtering rules of firewall by themselves are a formalized expression of an access policy. A problem in this case is that the rules are a mixture of a “clear” access policy and network specificity: IP addresses, ports, etc. The manual transition from an informal security policy to such a mixture is very difficult and error-prone. Thus, the formalized description of a “clear” access policy without the specificity of the environment where it is being executed appears to be a convenient medium in the process of reliable

firewall configuration on the basis of a security policy (Fig. 1).

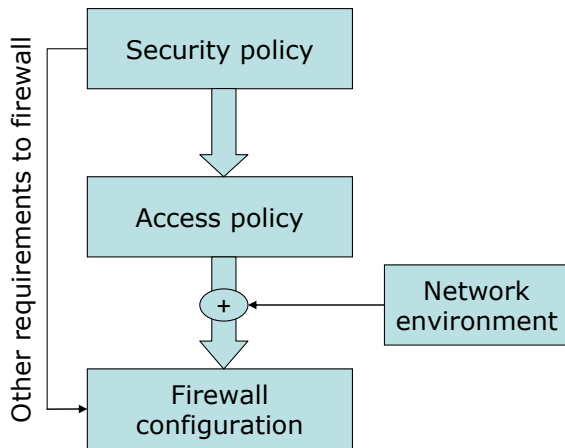


Fig. 1. Basic approach to firewall configuration

The next step to obtain final filtering rules is the integration of the “clear” access policy with a network environment specification. However, such a specification doesn’t assume to cover all the details of a network. Bluntly speaking the specification must give a description of the network that is sufficient for determining how access control may be implemented in it.

There may also be a customizable, security concerned part of a firewall configuration apart from filtering rules: utilized cryptographic algorithms, sizes of keys, etc. This part may still be manually specified by the expert (the arrow “other requirements to firewall” on Fig. 1). The part of firewall configuration that is not mixed with security and can be defined independently (performance parameters, etc.) is not considered here.

### 3 Access Policy Specification

An access policy in contrast to an overall security policy may just specify some requirements, e.g., “Alex cannot talk to Bob” without the refinement of the nature of Alex, Bob... and the reason of the requirements.

There is a common structure of access policy requirements, which uses the notions of *subject*, *action* and *object*. Thus, the informally described requirement “Alex cannot talk to Bob” can be formally represented as the combination of the subject “Alex”, the action “talk”, the object “Bob” and the decision “prohibition”. The base four is also may be completed by a *context*, which specifies various additional requirements restricting the cases of rule’s application, e.g.: time, previous actions of the subject, attributes’ values of the subject or object, etc.

However, access rules which are based on the notions of subject, action and object are not sufficient alone to implement complex real-world policies. As a result, new approaches have been developed. One of them, Role Based Access Control (RBAC), uses the notion of *role*. A role replaces a subject in access rules and it’s more invariant.

Identical roles may be used in multiple information systems while subjects are specific to a particular system. As an example, remember the roles of a system administrator and unprivileged user that are commonly used while configuring various systems. Administrator-subjects (persons) may be being added or removed while an administrator-role and its rules are not changing.

However, every role must be *associated* with some subjects as only rules with subjects can be finally enforced. During policy specification roles must be created firstly, then access rules must be specified with references to these roles, then the roles must be associated with subjects.

The ORBAC model [9][10] expands the traditional model of Role Based Access Control. It brings in the new notions of *activity*, *view* and *abstract context*. An activity is to replace an action, i.e., its meaning is analogous to the meaning of a role for a subject. A view is to replace an object. “Entertainment resources” can be an example of view, and “read” or “write” can be examples of an activity. Thus, the notions of role, activity, view and abstract context finally make up an abstract level of an access policy. ORBAC allows the specification of access rules only on the abstract level using abstract notions. They are called *abstract rules*. For instance, the abstract rule “user is prohibited to read entertainment resources”, where “user” – role, “read” – activity, “entertainment resources” – view. The rules for subjects, actions and objects are called *concrete access rules*.

Fig. 2 shows a firewall configuration process based on ORBAC model.

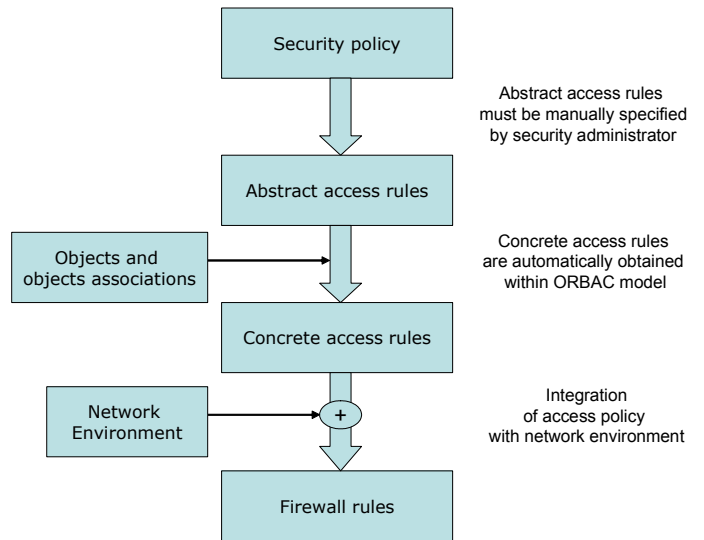


Fig. 2. Firewall configuration based on ORBAC

The specification of an access policy is divided into two convenient stages: the specification of abstract rules using abstract notions and the association of system’s objects (subjects, actions and contexts are also assumed) with the abstract notions. Thereafter, concrete rules may be obtained automatically.

An ORBAC policy may be (and should be) specified using a common language, e.g., XACML (eXtensible Access Control

Markup Language) [11]. In this case the language maintains the generality of policy's specification while ORBAC provides additional notions for convenient editing.

## 4 Network Environment Specification

First, it should be said that the assumed here network specification is not one that covers all the details of a network. It may be called an interface to a complete elaborate specification of the network. The interface gives a sufficient representation of the network as an environment where an access policy may be enforced.

The basic notion that is used to specify a network environment is a *data channel*. The most common interpretation of this notion might be that a channel is a distinguishable part of interaction between two initially independent subsystems of the network. The whole interaction within the network must be represented as the set of channels. However, if there is no need to refine upon subsystem's internal processes then the interaction within it may be represented by a single aggregated channel.

There may be an infinite set of channels as various protocols may be used for data exchange. Each protocol provides its own way of distinguishing separate channels in subsystem interconnection, e.g., by some addresses. If the systems are local subnetworks then, e.g., IP protocol permits of the division of the whole interaction between them into channels specified by pairs of IP addresses of hosts from both subnetworks (is called here *network channel*), e.g., "*IP traffic between the host with IP = 1.1.1.1 and the host with IP = 2.2.2.2*".

A network channel is rather an obvious example. A more sophisticated one is an *application channel* that corresponds to defined requests and responses transmitted at application layer. Such a channel may be specified by various parameters, e.g., URL in HTTP request, user's name, file's name, or the pair of user's name and file's name. Requests and responses matching the parameters belong to the channel. For example: "*FTP request from user "alex" to read the file "report.doc" and response to it*".

Interaction may also be distinguished by the time it happens. Thus, the description of a channel may include time intervals. This example makes it obvious that the notion of channel used here should not be equated to that one traditionally implied. Channel's specification may usually resemble the part of a filtering rule that defines the conditions when the rule must be applied.

The main property of every channel in the context of information security is its state: *open* or *close*. An open state means that any interaction associated with the channel can be successfully performed if it happens. A close state, otherwise, means that such an interaction is completely blocked somehow. *Network state* here is the set of states of all its channels.

It's obvious that an implementation of an open/close channel is based on an embedded firewall and may correspond to some filtering rule(s). However, we don't

always may say exactly whether there is a single firewall or a group. Therewith, a firewall alone cannot be responsible for the correct implementation of an open/close state. It's the responsibility of the whole network environment to support the secure separation of channels, opening one and closing another.

The difference between a filtering rule and a corresponding opened/closed channel is that the first tries to determine whether a transmitted block of data belongs to the channel (that may be difficult to reliably implement), whereas the second doesn't refine on this procedure. Thus, for the present we use the open/close states of channels and abstract from its specific implementation via filtering by firewall(s), thereby avoiding the problem of correct identification of data blocks. All such problems are assumed to be solved somehow within network environment.

### Relations between channels

An important relation that may exist between channels is that one channel can be an *underlying channel* of another *underlaid channel*. An underlaid channel cannot work properly (be opened) if its underlying one doesn't work properly (is closed). Physical channels underlay all over channels (may be indirectly), e.g., a network channel cannot work if there is no opened underlying physical channel. An application channel must be underlaid by a network channel as application requests may only be transmitted using a network connection.

A channel may have several underlying channels, and there are two base *compositions* of them: serial ( $\wedge$ ) or parallel ( $\vee$ ). In a serial composition the underlaid channel can be opened only if all the underlying channels are opened. In a parallel composition the underlaid channel can be opened if at least one underlying channel is opened. There can be used more complex compositions that are combinations of the two base ones. Fig. 3 shows an example where the network channel is underlaid by three physical channels (PC) using the complex composition:  $(PC1 \wedge PC2) \vee PC3$ . PC1 and PC2 may be successive parts of the same physical link, PC3 is an alternative physical link.

If a network application runs on several hosts (servers) then its application channels are underlaid by several network channels (for each host). A client may choose one from the hosts to establish an underlying network connection and gain the full access to the application. A parallel composition must be used in this case. If there is a router between a client and the server then the application channels are underlaid by two network channels (in front of and behind the router). Both network channels must be opened to gain an access to the application. A serial composition must be used in the case.

Physical channels compose the lowest *layer* of network environment (physical layer). The channels that are directly underlaid by physical ones (e.g., network channels) compose a higher layer (e.g., network layer), etc. However, the mentioned notion of layer is not necessary. It's used here for explanation only as may be associated with the notion of layer commonly used by network specialists.

## 5 Integration of Access Policy with Network Environment

In order to represent a general method of the integration of an access policy with a network environment we call the combination (subject, action, object and context) by a *procedure*. In general, a procedure is an element of an access policy model that may be permitted (a permission of the access policy) or prohibited (a prohibition of the access policy). Thus, for a start we abstract from a specific structure of access policy requirements.

The integration is implemented as the building of a new layer over the existing layers in a network environment (Fig. 3).

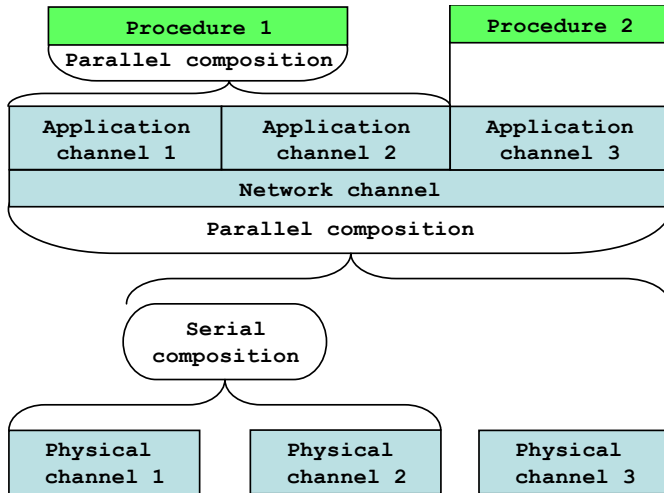


Fig. 3. Integration of access policy with network environment

A procedure is treated as a new peculiar channel underlaid by channels in the network environment. Different compositions (serial or parallel) in this case are also permitted, e.g., it may be specified that the procedure “Alex talks to Bob” is underlaid by the network channel “IP traffic between the host with IP = 1.1.1.1 and the host with IP = 2.2.2.2”; the procedure “Alex reads financial report” is underlaid by the application channel “FTP request from user “alex” to read the file “report.doc” and response to it”. On Fig. 3 procedure 2 is underlaid by application channel 3, procedure 1 is underlaid by application channels 1 and 2 using a parallel composition.

The permission of a procedure means that its underlying channels and their underlying ones (recursively) must be opened (the all of them if a serial composition is used, and at least one if a parallel composition is used, see procedure 1 on Fig. 3). The prohibition of a procedure means that all its *direct* underlying channels must be closed in the case of a parallel composition, and at least one must be closed in the case of a serial composition. A recursive closing is not necessary (e.g., physical channels may always be opened while access control at higher layers is restricted with correspondence to the access policy). The prohibition of

procedure 1 means that the application channels 1 and 2 must be closed, but the network channel may still be opened.

The simple rules above are a minimal set to guarantee the security and availability of information. However, in the case of a permission all the channels in a parallel composition should be opened if there is no explicit prohibition for someone. It guarantees higher availability. Similarly, in the case of a prohibition all the channels in a serial composition should be closed if there is no explicit permission for someone. It guarantees higher security.

The rules above can easily be used to check the conformity of network’s state (open/close states of its channels) with an access policy. They are also the basis of an algorithm producing network’s state from an access policy.

Placing a procedure over underlying channels is crucial enough. Missing a channel in a parallel composition may cause a security breach when the procedure is prohibited but the channel is opened, and it remains unnoticed. Missing a channel in a serial composition may cause a lack of availability when the procedure is permitted but the necessary channel is closed.

### 5.1 Integration of ORBAC with Network Environment

Turn back to the specific structure of a procedure composed of the items: subject, action, object and context. The placing of a procedure over underlying channels splits: every item must be placed separately. Therefore the descriptions of the channels (its identifiers) must also be split into four partial identifiers:  $ID_s$ ,  $ID_o$ ,  $ID_A$ ,  $ID_C$ .

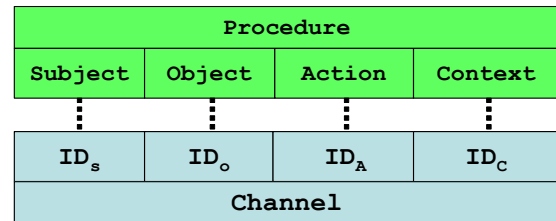


Fig. 4. Placing ORBAC procedure over a channel

If there is a single underlying channel, each item of the procedure is simply associated with the corresponding partial identifier (Fig. 4). If there are several underlying channels using a serial or parallel composition then each item of the procedure is associated with the set of corresponding partial identifiers using the same composition.

A partial identifier may have an obvious interpretation, e.g., for a network channel  $ID_s$ ,  $ID_o$  are IP addresses of the end hosts ( $ID_A$ ,  $ID_C$  are singular, “empty”). For an application channel  $ID_s$  may be user’s name,  $ID_o$  – file’s name. Thus, partial IDs are useful notions for channel description as well as subjects and objects are useful notions for procedure description. However, an access policy by itself remains to be based on procedures corresponding to some interactions within an information system, so the network environment

specification is being designed to be based on channels, which also correspond to some interactions.

The integration of an ORBAC access policy with a network environment consist in that the network environment is providing partial identifiers of its channels as something like tie points, and then subjects, objects, actions and contexts are being tied to them. Fig. 5 represents such an interpretation (without actions and contexts in order to maintain obviousness).

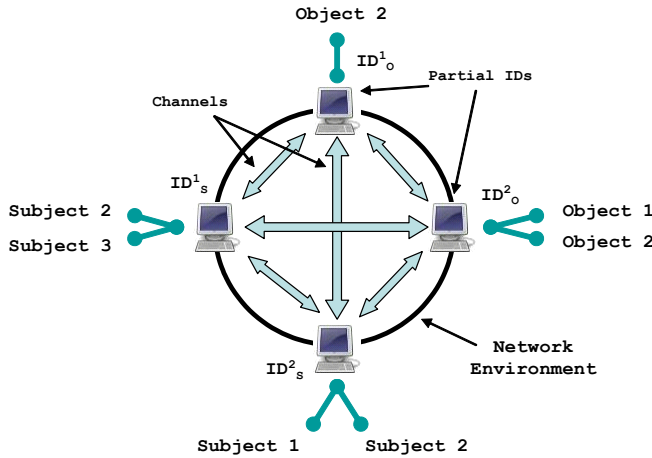


Fig. 5. Integration of ORBAC with network environment

A tie between a subject and an  $ID_s$  (e.g., a host) may correspond to the case when a person (which corresponds to the subject) is being given the password to login to the host. A tie between a subject and user's name ( $ID_s$ ) means that the person is being given the password to a particular user account of host's operating system. A tie between an object and file's name ( $ID_o$ ) means that the object's information is stored in this file.

Following Fig. 5 assume that subject 1 is permitted for access to object 1 then the channel between  $ID_s^2$  and  $ID_o^1$  must be opened. If subject 2 is permitted for access to object 2 (they are tied with several IDs so assume that a serial composition is used) then channels ( $ID_s^2 - ID_o^2$ ), ( $ID_s^1 - ID_o^1$ ), ( $ID_s^1 - ID_o^2$ ), ( $ID_s^2 - ID_o^1$ ) must be opened. If subject 1 is permitted for access to object 1 and subject 2 is prohibited from access to object 2 then it's a case when the given access policy cannot be correctly enforced in the given environment (the prioritization of the requirements might solve the problem but it's not considered here).

## 6 Following work

There are important issues not considered here, which are the subject of the following work.

### Dependent channels

The underlay relation is not only one that may exist between channels. Closing a channel may affect the correct work of another opened one (and vice versa). Such channels may correspond to intersecting filtering rules.

### Prioritization

The prioritization of conflicting permissions and prohibitions makes an access policy specification more flexible. A network environment must also support a prioritization when, e.g., the close state of one channel is specified to have more priority than the open state of another dependent channel (so the interaction associated with both channels is blocked).

### Dynamic integration with access policy

The open/close state of a channel should be only a default state. In common cases a firewall (which just opens and closes channels) decides by itself whether a channel must be actually opened (based on its default state). The decision procedure is assumed to be very fast in this case that preserves performance. In special cases the firewall must be able to request a more sophisticated access policy manager ("Policy Decision Point" following [11]). It may cause performance degradation while maintains the possibility of the enforcement of most flexible access policies. For instance, the involved requirement "subject 1 and subject 2 are permitted for access to object 1 simultaneously only" may require the firewall to dynamically request the manager during its work.

## 7 Conclusions

In this paper we have proposed an approach to firewall configuration based on the integration of initially separated descriptions of an access policy and a network environment.

ORBAC model is used for the formalized description of a "clear" access policy without the specificity of the environment where it is being executed. The specification of the access policy is divided into two convenient stages: the specification of abstract rules using abstract notions (roles, views, activities and abstract contexts) and the association of particular system's objects with the abstract notions. Thereafter, concrete rules (for subjects, objects, actions and contexts) may be obtained automatically.

The formalized description of a network environment is based on a *channel*. The most common interpretation of this notion might be that a channel is a distinguishable part of interaction between two initially independent subsystems of the network.

The main property of every channel is its state: *open* or *close*. An open state means that any interaction associated with the channel can be successfully performed if it happens. A close state, otherwise, means that such an interaction is completely blocked somehow. The implementation of an open/close channel is based on an embedded firewall and may correspond to some filtering rule(s).

An important relation that may exist between channels is that one channel can be an *underlying channel* of another *underlaid channel*. The underlaid channel cannot work properly (be opened) if its underlying one doesn't work properly (is closed). Physical channels underlay all over channels.

The integration of an access policy with a network environment is implemented as the building of a new layer over the existing layers in the network environment. An access policy procedure (the combination of subject, action, object and context) is treated as a channel underlaid by channels in the network environment. Thus, there are rules of correspondence between the permission/prohibition of a procedure and the open/close states of its underlying channels. The rules can be used to check the conformity of network's state (open/close states of its channels) with the access policy. They are also the basis of an algorithm producing network's state from the access policy.

Considering the specific structure of a procedure based on the items (subject, action, object and context) the placing of a procedure over underlying channels splits: every item must be placed separately. Thus, the descriptions of the channels (its identifiers) must also be split into four partial identifiers (ID). They may have an obvious interpretation, e.g., for a network channel they are IP addresses of the end hosts. For an application channel it may be user's name or file's name. The integration of an ORBAC access policy with a network environment consist in that the network environment is providing partial identifiers of its channels as something like tie points, and then subjects, objects, actions and contexts are being tied to them. A tie between a subject and an ID<sub>s</sub> (e.g., a host) may correspond to the case when a person (which corresponds to the subject) is being given the password to login to the host.

## References

- [1] W. Geng, S. Flinn, and J. DeDourek. Usable firewall configuration. In PST '05: Proceedings of the 3rd Annual Conference on Privacy, Security and Trust. 2005.
- [2] E. Al-Shaer, H. Hamed, R. Boutaba, M. Hasan. "Conflict classification and analysis of distributed firewall policies". IEEE Journal on Selected Areas in Communications (JSAC), Vol. 23 No.10, p.2069-84. 2005.
- [3] T. Hinrichs, N. Gude, M. Casado, J. Mitchell, S. Shenker. Practical Declarative Network Management. Workshop on Research in Enterprise Networks (WREN). 2009.
- [4] B. Zhang, E. Al-Shaer, R. Jagadeesan, J. Riely, C. Pitcher. Specifications of a high-level conflict-free firewall policy language for multi-domain networks. Proceedings of the 12th ACM symposium on Access control models and technologies. p.: 185 – 194. 2007.
- [5] V. Zaborovsky, A. Titov. Specialized Solutions for Improvement of Firewall Performance and Conformity to Security Policy. Proceedings of the 2009 International Conference on Security & Management. v. 2. p. 603-608. July 13-16, 2009.
- [6] A. Tongaonkar, N. Inamdar, R. Sekar. Inferring higher level policies from firewall rules. Proceedings of the 21st conference on Large Installation System Administration Conference. 2007.
- [7] Y. Bartal, A. Mayer, K. Nissim, and A. Wool. Firmato: A novel firewall management toolkit. In 20th IEEE Symposium on Security and Privacy. 1999. - p. 17–31.
- [8] K. Trevisani, R. Garcia. SPML: A Visual Approach for Modeling Firewall Configurations. International Conference on Model Driven Engineering Languages and Systems (MODELS). 2008.
- [9] A. Abou El Kalam, R. El Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel et G. Trouessin, Organization Based Access Control, IEEE 4th International Workshop on Policies for Distributed Systems and Networks (Policy 2003), Lake Como, Italy, June 4-6, 2003.
- [10] F. Cuppens, N. Cuppens-Boulahia, T. Sans and A. Miège, A formal approach to specify and deploy a network security policy, Second Workshop on Formal Aspects in Security and Trust (FAST). Toulouse, France. August, 2004.
- [11] eXtensible Access Control Markup Language. Version 3.0. OASIS. 2010.