Available online at www.sciencedirect.com
SCIENCE DIRECT
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 121 (2005) 117–142
www.elsevier.com/locate/entcs

# A Formal Approach for the Evaluation of Network Security Mechanisms Based on RBAC Policies

R. Laborde, B. Nasser, F. Grasset, F. Barrère, A. Benzekri [1]

*IRIT - SIERA*
*Université Paul Sabatier*
*Toulouse, France*

**Abstract**

Security policy models allow reasoning about security goals achievements. When security mechanisms are implemented, it is difficult to formally validate the security properties against the security goals especially in a network environment. To assess the implemented security properties, one should consider details regarding the network topology, the forwarding as well as filtering and transform engines. In this paper, we present a Colored Petri Net based tool which allows to describe graphically a given network topology, the network security mechanisms and the security goals required. The tool computes the different functionalities to set up the security properties and formally validates the solution using the dead state of the generated reachability graph analysis. Different security properties such as confidentiality and availability can be studied.

*Keywords:* Network Security, Security Management, Colored Petri Nets.

## 1 Introduction

The design, operation, and maintenance of network configurations constitute an important part of the security management task. Basically, the security of distributed applications is supported by a set of network security services which are implemented by means of security mechanisms.

The administrator should determine the security services to use and the security mechanisms configurations to apply. Once deployed, network security

[1] Email: {laborde, nasser, grasset, barrere, benzekri}@irit.fr


doi:10.1016/j.entcs.2004.10.011

policies often become unmanageable over time since more rules are added and there is a real difficulty in retrieving, managing and getting rid of old unnecessary rules. This fact leads to ever increasing pains in managing active security policies in network devices.

Traditional management platforms which use SNMP agents are too simple to tackle the problem complexity. For this reason, different architectures and techniques have recently appeared which increase the management agents' capacities in order to automate the management task.

In this context we find the policy based management approach which considers abstract security policies [4,12,18,20,33,34] that can be represented at different levels [25,30], ranging from business goals to device-specific configuration parameters. The process that transforms a definite goal into the corresponding configurations is called derivation process [2,24,31]. With a similar perspective the multi-agent system paradigm based approach [14] wishes the management agents to be more autonomous in order to be able to cooperate for creating strategies that fulfill to defined objectives. The terms strategies objectives employed here take after the policy abstraction level in the former approach. Finally as a third approach, the latest emerging works [36,37,38,39] proceed with this idea by the "Self-Adaptive Autonomic Computing" concept using the prefix "self" as a leitmotiv.

However, the automation sought after via the above cited approaches, is not adequate for security management yet. There is no automatic evaluation method of network security policies indeed. Access control models [5,9,29] provide a solution for the definition of security objectives.

In fact they afford a formal technique for defining what is and what is not allowed. Moreover, there are several techniques besides, associated [7,9,28,32] with each model, to guarantee that a security policy is correct.

Nevertheless, these models do not consider the associated security mechanisms or strategies. Network security management is by nature a distributed function supplied by the coordination of a variety of devices with different capabilities (PCs, routers, secure gateways, firewalls, etc). By consequence the same objective can be enforced by different compositions thus different strategies. For example, confidentiality can be implemented by filtering mechanisms or encryption mechanisms. It is then necessary to develop an automated formal evaluation technique for defining what a correct security network strategy is.

There is a variety of formal verification techniques used in the security context: theorem provers (EHDM [22], PVS [27]) and model checking/finding techniques (SMV [6], NPA [21], Alloy [13]). All formal specification languages such as Z, LOTOS or Petri Nets [16,19] were also used. Unfortunately, there

is no model associated with network security proposed to be used with these techniques. Accordingly, we propose a new formal verification tool which is specific to network security policy. It includes a model of the application security policies, the network security policies/mechanisms and the network topology.

The paper is organized as follows. In section 2, we explain what is meant by network security from our perspective. In section 3, we define our formal specification language and our formal evaluation method. In section 4, we present our tool which implements the previous concepts and automates the evaluation task. Also, we expose a small example of use. Finally, in section 6, we show our conclusions and our plans for future work.

## 2 Definition of a network security policy

Among the access control models [5,29], we have chosen the NIST RBAC model [9] because it simplifies the management tasks. Actually, the role concept allows aggregating the users' permissions and then it facilitates the users' rights modifications made by an administrator. Moreover, the hierarchies between roles represent a good tool for modelling an organization according to different points of view.

### *2.1 The NIST RBAC model*

The NIST group proposes the standardization of the RBAC model [9]. It is made up of two sub-models: the core model and the hierarchical model (Fig. 1). The core model includes five sets of basic data elements:

- A "user" is an active entity, i.e., human or intelligent agent.
- A "role" is a job function within the context of an organization with some associated semantic regarding the authority and responsibility on the user assigned to the role. We can notice that the definition is very vague.
- A "permission" is an approval to perform an operation on one or more protected objects.
- An "operation" is an executable image of a program, which upon invocation executes some function on behalf of the user.
- An "object" is an entity that contains or receives information.

Finally, a set of roles is assigned to a user, and a set of permissions is assigned to a role. A session is a mapping of one user to a set of authorized roles.

The hierarchical model adds relations for supporting role hierarchies. There exist different approaches for constructing a role hierarchy: based on privileges

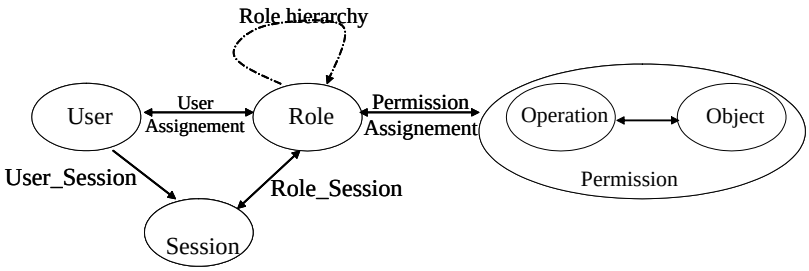


Fig. 1. The NIST RBAC Model

[26] or based on users' job functions [8,23].

### *2.2 What is the relation between an application security policy and a network security policy?*

When a user accesses a service, a set of data flow is exchanged between the device which the user launches the service and the devices supporting the service execution (Fig. 2). So, a relation between a network security policy and an application security policy can be perceived. For example, if the application security policy states that user $u_1$ can read object $o_1$- noted $(u_1, o_1, +read)$, then it implies that a corresponding data flow $flow(o_1, +read)$ between the device of user $u_1$ and the device of $o_1$ can exists on the network. Consequently, the associated network security policy must allows the data flows $flow(o_1, +read)$ between these two devices - noted $(device(u_1) \leftrightarrow device(o_1), +flow(o_1, read))$. Conversely, if the application security policy states that user $u_2$ cannot read object $o_2$ noted $(u_2, o_2, -read)$, there should not be a flow $flow(o_2, read)$ between the devices of $u_2$ and $o_2$. Therefore, the network security policy must forbid $flow(o_2, read)$ between the devices of $u_2$ and $o_2$, i.e., $(device(u_2) \leftrightarrow device(o_2), -flow(o_2, read))$. We report this information from application to network level, in order to stop these data flows and so to prevent Deny of Service or exploits/payloads based attacks.

**Definition 2.1** The derivation relation noted $\Rightarrow^d$ is defined as

$$\forall u \in USERS, \forall o \in OBJECTS, \forall a \in ACTIONS, \\ (u, o, \pm a) \Rightarrow^d (device(u) \leftrightarrow device(o), flow(o, \pm a)). \quad (1)$$

### *2.3 Towards an "RBAC network security policy"*

Users are considered in an RBAC system by their assigned role. Consequently, the derivation relation becomes: $\forall r \in ROLES, \forall o_i \in OBJECTS, \forall op_j \in$

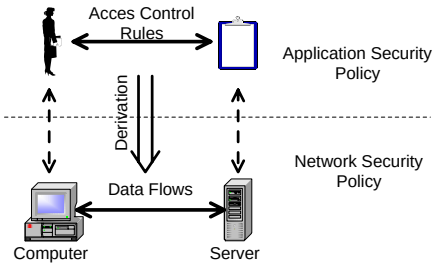


Fig. 2. Security policy derivation

$OPERATIONS, \forall u, u' \in USERS, u \neq u' \bullet (r, (op_j, o_i)) \wedge assigned(u, r) \wedge \neg assigned(u', r) \Rightarrow^{d} (device(u) \leftrightarrow device(o_i), +flow(o_i, op_j)) \wedge (device(u') \leftrightarrow device(o_i), -flow(o_i, op_j))$.

Hereafter, we consider that there is no hierarchy and that roles have disjoint privileges (if this is not the case, we may create a partition of this set): such a constraint will help us to group data flows based on the permissions assigned to one role and then identifying them by the role. Afterward, we note by the name of the role the set of flows corresponding to the permissions assigned to the role.

According to these definitions, we present our method that includes a network architecture specification language and a security mechanisms validation against an RBAC security policy process.

## 3 Network architecture model and security mechanisms analysis

In a network environment, all the applicable treatments on data flow can be brought together into four categories of functionalities:

- Mechanisms that *consume/produce data flows* such as the end-systems,
- Mechanisms that *propagate data flows* such as the supports of communication,
- Mechanisms that *transform data flows* into another one such as the security protocols,
- Mechanisms that *filter data flows* such as the firewall ones.

So, our process consists of modeling these functionalities and interactions between these functionalities. Hence, we define a graphical language with a formal semantic in order to support the network security policy verification process.

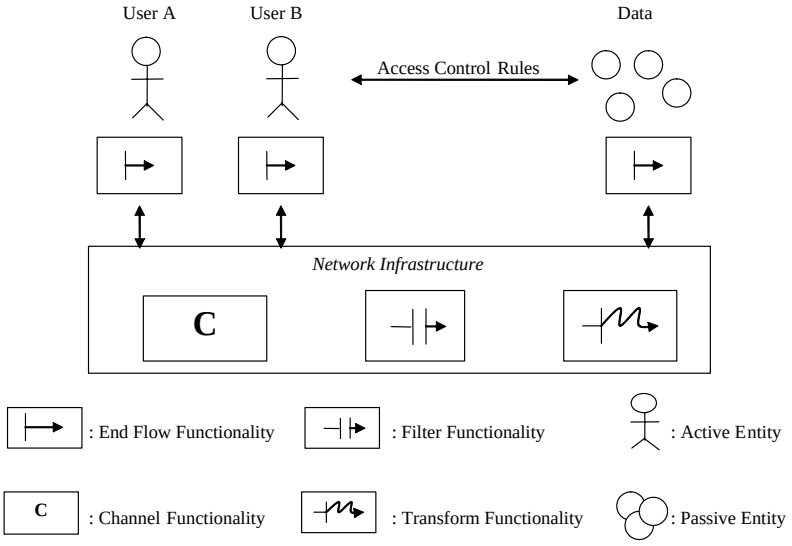


Fig. 3. The model of network topology

In our model (Fig. 3), we find a set of active entities and a set of passive entities, and a set of functionalities (end-flow, channel, transform and filter) which act on information flows. An active entity corresponds to a user in the RBAC model, and a passive entity is a set of objects in the RBAC model.

We describe the semantic of our language using the Colored Petri Nets (CPNs). CPNs [15,17] provide a framework for the construction and analysis of distributed and concurrent systems. A CPN model of a system describes the states in which the system may be and the transitions between these states.

### *3.1 Colored Petri Nets*

The states of a CPN are represented by means of places (which are drawn as ellipses or circles). Each place has an associated type (color set) determining the kind of data that the place may contain. A state of a CPN is called a marking. It consists of a number of tokens positioned (distributed) on the individual places. Each token carries a value (color), which belongs to the type of the place on which the token resides. The tokens present on a particular place are called the marking of that place. The tokens of a CPN are distinguishable from each other and hence "colored", in contrast to low level Petri nets which have "black" indistinguishable tokens. The marking of a place is, in general, a multi-set of token values. A multi-set is similar to a set, except that there may be several appearances of the same element. This means that a place may have several tokens with the same token value. For

example, $1`c_1 + +2`c_2$ means that the place contains 3 tokens, one with the value $c_1$ and two with the value $c_2$. The actions of a CPN are represented by means of transitions (which are drawn as rectangles). Transitions and places are connected by arcs. The actions of a CPN consist of occurrences of transitions. An occurrence of a transition removes tokens from places connected to incoming arcs (input places), and adds tokens to places connected to outgoing arcs (output places), thereby changing the marking (state) of the CPN. The exact number of tokens added and removed by the occurrence of a transition, and their data values are determined by the arc expressions. In addition to the arc expressions, it is possible to attach a boolean expression (with variables) to each transition. The boolean expression is called a guard. It specifies that we only accept bindings for which the boolean expression evaluates to true. A CPN has a distinguished marking - the initial marking - which is used to describe the initial state of the system. A CPN may also have one or more markings - dead markings - which cannot generate any other marking. They describe the dead states of the system.

Nevertheless, CPNs do not bring any additional power of description compared to the PNs, they just allow a compression of information. Any marked CPN can thus be associated with an isomorphic PN. The phase of transformation of a CPN into a PN is called the "unfolding". Afterwards, the analysis is performed with the CPN or the PN (temporal logic with occurrence graph, linear algebra with incidence matrix, classical PN properties).

### *3.2 Definition of the functionalities*

We model data flows with tokens. The characteristics of data flows (source, destination, service used) are represented by the tokens' colors. Each functionality is modelled by a specific CPN sub-network that acts on the tokens. Then, a CPN model of a specification is an interconnection of sub-networks. We thus define a total function that maps all specification into a sub-set of CPN. This component approach makes it possible to transform a specification into an equivalent CPN in an automatic way.

#### *3.2.1 The end-flow functionality*

An end-flow (EF) is a functionality that is specific to end-systems, i.e., data and application servers as well as workstations. It transmits applications/users flows to the network. Thus, the functionalities which produce/consume data flows are specified by CPN sub-nets that generate tokens with corresponding colors and receive tokens (Fig. 4). We consider two types of end-flow functionalities:

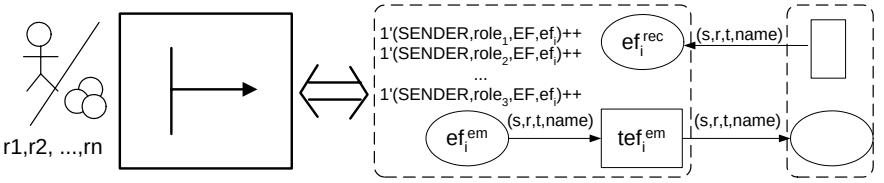


Fig. 4. The end-flow functionality CPN model

- *Active End Flow functionality* (AEF): An EF is said active if any active entity is connected to this EF.
- *Passive End Flow functionality* (PEF): An EF is said passive if any passive entity is connected to this EF.

We append a list of roles to each EF for indicating the flows that the EF can produce. The list corresponds to the set of roles assigned to the user representing the connected active entity for an AEF. In the case of a PEF, it is the set of roles assigned to the permissions that concern an object of the connected passive entity. In a CPN built from one of our specification, a token corresponds to a particular flow.

Consequently, a token is a tuple $< SENDER, ROLE, TYPE, NAME >$ that defines the color domain FLOW where:

- $SENDER \in \{AEF, PEF\}$,
- $ROLE \in ROLES$ that is the set of roles,
- $TYPE \in \{EF, TR\}$ means that the flow is transformed or not (see "The transform functionality"),
- $NAME$ is the name of the end-flow functionality.

We specify the producer ability with a place ($ef_i^{em}$) that initially contains all data flow tokens that the end-flow functionality can send (that is $\bigcup_{assigned\_ef(R,ef_i)} 1' < SENDER, ROLE, EF, NAME >$) and transition ($tef_i$) to connect it to another functionality. Its consumer capability is represented by one place ($ef_i^{rec}$) that stores the received tokens.

### *3.2.2 The channel functionality*

The channel functionality models the physical network. It receives the flow on an interface and retransmits it to all the connected entities. This functionality may be viewed as a broadcast channel. When a flow is oriented, it is not only received by the addressed destinations but also by all of the systems connected to this channel. The functionalities which propagate data flows are specified

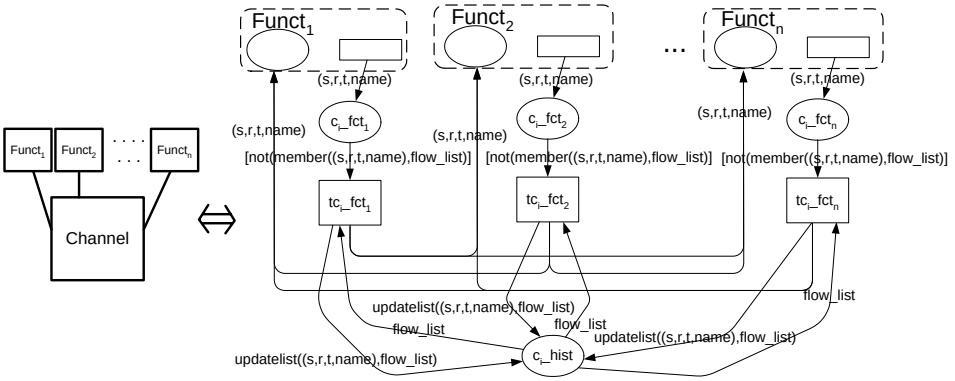


Fig. 5. The channel functionality CPN model

by CPN sub-nets that receive a token from a functionality and send replica to all the other connected functionalities.

So, channel sub-networks are composed of a set of couples (place, transition) for each connected functionality (Fig. 5). Transitions are connected to all other functionalities. For instance, $tc_i fct_1$ is connected to $funct_2$, $funct_3$, , $funct_n$. We also add a place ($c_i\_hist$) which contains the list of all the tokens that have passed through the channel. It is connected to each transition to ensure us that a token can pass once and only once through a channel functionality.

### *3.2.3 The transform functionality*

The transform functionality receives a data flow on one of its two interfaces, and according to transformation rules, it sends via the other interface this data flow or a transformation of it. The BNF definition of the syntax of transform functionalities configuration is as follows:

<TransformConfiguration> ::=
[<interface> "→"<interface><rule>]
[<interface> "←" <interface><rule>]
<rule> :: = [<name> "="] <roles_list>
<roles_list> ::= <role> | <role> "," <roles_list>

Consequently, transform CPN sub-nets change the color of some tokens according to the transformation rules. We set up the functions on the post-arcs transitions to change the color of the token ($Transf\_funct_1$ and $Transf\_funct_2$ in Fig. 6). Moreover, if a functionality can transform a flow, it should be able to recover the original flow (see section 7.1.1). We also add two places ($hist\_tf_i\_fct_1\_fct_2$ and $hist\_tf_i\_fct_2\_fct_1$) to save traces of all the flows that

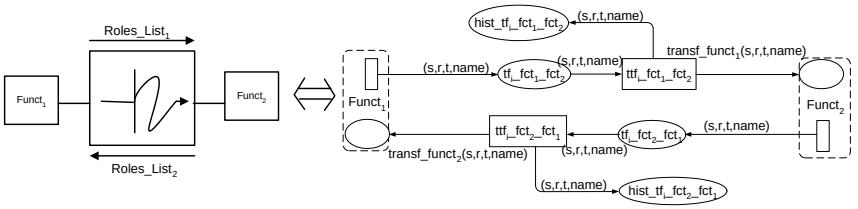


Fig. 6. The transform functionality CPN model

have passed through this functionality.

### *3.2.4 The filter functionality*

The filter functionality stops or forwards a data flow. We find this functionality in firewalls, Application Level Gateways or filtering routers. But we restrict it to only connect two functionalities. The filtering rules explicitly express the permitted flows between its two interfaces. If they are preceded by "EF" then they arrive untransformed from an end-flow functionality, else if they are preceded by "TR" then they have been modified by a transform functionality. The BNF definition of the syntax of filter functionalities configuration is as follows:

<FilterConfiguration> ::=
[ <interface> "→" <interface> <rules>] [";"]
[ <interface> "←" <interface> <rules>]
<rules> ::=
<name> "=" ["EF" <flow_list>] ["TR" <flow_list>]
<flow_list> ::= <flow> | <flow> "," <flow_list>
<flow> ::= "(" <EF_type> "," <role> ")"
<EF_type> ::= "AEF" | "PEF"

Consequently, the filter CPN sub-nets stops or not some tokens according to their color and the filtering rules (Fig.7). We represent the filtering rules by restricting the colors permitted by the transitions with guards (see section 7.1.2). Then, a token with a color that is not in the guard of a transition cannot be fired. The transition $tf_{i_{fct_1 fct_2}}$ (resp. $tf_{i_{fct_2 fct_1}}$) is used to filter data flows coming from $funct_1$ (resp. $funct_2$) to $funct_2$ (resp. $funct_1$). In addition, we add two places ($hist\_f_i\_fct_1\_fct_2$ and $hist\_f_i\_fct_2\_fct_1$) to save all the flows that have passed through this functionality.

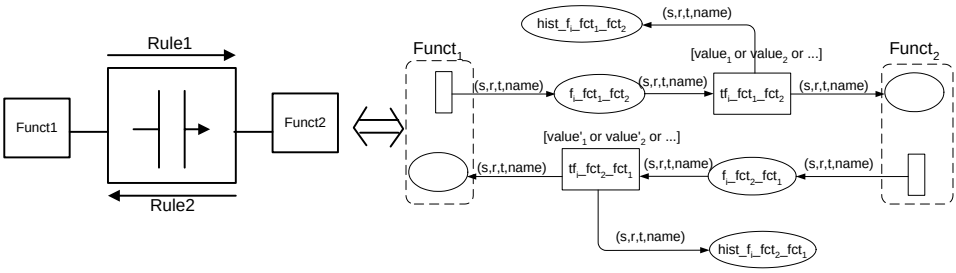


Fig. 7. The filter functionality CPN model

### *3.3 Security analysis*

We use the model checking technique to determine if a specification satisfies the security properties. Nevertheless, this technique is sensitive to the combinatorial explosion problem. Thus, we expose two theorems that allow us to limit our analysis of the CPN to only two states of the reachability graph.

**Theorem 3.1** *There is one and only one dead state in the reachability graph of a CPN produced by any specification.*

**Proof.** see section 7.2.1. □

**Theorem 3.2** *The analysis of the initial and dead states is necessary and sufficient.*

**Proof.** see section 7.2.2. □

The theorem 3.2 guarantees that we only have to study the initial and dead states in the reachability graph. The first theorem ensures us obtaining the dead state by simulation. Consequently, there is no combinatorial explosion problem for the dead state analysis and then big size specifications can be studied.

We now present the security properties that the initial and dead state must satisfy.

We use the following notation:

- $FUNCT$, the set of functionalities,
- $FILTER$, the set of filter functionalities,
- $ACTIVE$, the set of active end-flow functionalities,
- $PASSIVE$, the set of passive end-flow functionalities,

- $ROLES$, the set of roles,
- $SENDER = \{AEF, PEF\}$,
- $Connected \subseteq FUNCT \times FUNCT$, the relation that defines direct connection between functionalities,
- $Assigned : (ACTIVE \cup PASSIVE) \rightarrow 2^{ROLE}$, the relation that defines the set of roles assigned to an end-flow functionality,
- $COLOR$, the set of colors in the CPN,
- $PLACE$, the set of places in the CPN,
- $Tokens : PLACE \rightarrow Bag(COLOR)$, where Bag(COLOR) is the set of multiset over COLOR. It provides the set of colored tokens present in a place and a state.

For simplifying writing properties, we use the special character "_" for indicating that one of the possible values is a member of the variable type. The expression $state \models property$ denotes that the state in the CPN reachability graph satisfies the property - $s_i$ is the initial state and $s_f$ is the dead sate. Now, we define the security properties.

**Definition 3.3** *Property of confidentiality*
Basically, the property of confidentiality protects the data from unauthorized disclosure. Thus, in our model, it prohibits an end-flow functionality from receiving at any time an untransformed data flow from any unassigned role.

$$\forall ef \in ACTIVE, \forall r \in ROLES, r \notin Assigned(ef) \Rightarrow s_f \models < \_, r, EF, \_ > \notin Tokens(ef^{rec}) \quad (2)$$

**Definition 3.4** *Property of integrity*
Classically, the property of integrity prohibits non authorized entities from any creation, modification or destruction of objects. Then, in our model, this property implies that an end-flow functionality can only generate data flows through its assigned roles.

$$\forall ef \in ACTIVE \cup PASSIVE, \forall r \in ROLES, r \notin Assigned(ef) \Rightarrow s_i \models < \_, r, \_, ef > \notin Tokens(ef^{rec}) \quad (3)$$

**Definition 3.5** *Property of availability*
This property stipulates that all the granted services must be available to all the authorized entities. In the network environment, the data flows corresponding to these services, must be able to travel between both devices. Consequently, translating it in our model results in: all active (resp. passive) end-flow functionalities must be able to consume all the data flows with an

assigned role sent by every passive (resp. active) end-flow functionality.
Let $ACTIVE_r = \{ef_a \in ACTIVE | r \in Assigned(ef_a)\}$
$PASSIVE_r = \{ef_p \in PASSIVE | r \in Assigned(ef_p)\}$

$$\forall r \in ROLES, \forall ef_a \in ACTIVE_r, \forall ef_p \in PASSIVE_r,$$
$$s_f \models < PEF, r, EF, ef_p > \in Tokens(ef_a^{rec}) \wedge < AEF, r, EF, ef_a > \in Tokens(ef_p^{rec}) \quad (4)$$

As we intend to address devices configurations, we complete these classical security properties with new ones.

**Definition 3.6** *Property of partitioning*
With this propertiy, we wish to limit the propagation of data flows as much as possible. It declares that a data flow can only pass a filter functionality that is situated between the data flow source and a possible correct destination.
Let $ACTIVE_r = \{ef_a \in ACTIVE | r \in Assigned(ef_a)\}$
$PASSIVE_r = \{ef_p \in PASSIVE | r \in Assigned(ef_p)\}$

$$\begin{aligned} \forall f \in FILTER, \forall fct_1, fct_2 \in FUNCT, \forall r \in ROLES, \\ Connected(f, fct_1) \wedge Connected(f, fct_2) \\ \wedge (s_f \models < AEF, r, \_, \_ > \in Tokens(hist\_f\_fct_1 fct_2) \Rightarrow \\ \exists ef_a \in ACTIVE_r \wedge fct_2 \in Path(f, ef_a)) \\ \wedge (s_f \models < PEF, r, \_, \_ > \in Tokens(hist\_f\_fct_1 fct_2) \Rightarrow \\ \exists ef_p \in PASSIVE_r \wedge fct_2 \in Path(f, ef_p)) \end{aligned} \quad (5)$$

The two following constraints aim to suppress implemented filtering or transform rules that are not used in a usual context.

**Definition 3.7** *Non productive filtering rule*
Let $f$, be a filter functionality connected to the functionalities $fct_1$ and $fct_2$. We say that the filtering rule FRL, which lets the data flow $< s, r, t, ef >$ pass, from $fct_1$ to $fct_2$, is non productive if this flow never tries to pass through the filter functionality.

Let the rule FRL $= fct_1 \rightarrow fct_2$ t (s, r) where $fct_1, fct_2 \in FUNCT$,
$t \in \{EF, TR\}, s \in AEF, PEF, r \in ROLES$ then FRL is non productive iff

$$s_f \models < s, r, t, \_ > \notin Tokens(hist\_f\_fct_1 fct_2) \quad (6)$$

**Definition 3.8** *Non productive transform rule*
Let $tf$, be a transform functionality connected to the functionalities $fct_1$ and $fct_2$. We say that the transform rule TRL, that transforms the data flows with the role $r$ from $fct_1$ to $fct_2$ is non productive if any flow with the role

$r$ pass through the transform functionality in the direction $fct_1$-$fct_2$ at any time.

$$\begin{aligned}&\text{Let TRL} = fct_1 \rightarrow fct_2 \text{ r where } fct_1, fct_2 \in FUNCT, r \in ROLE\\ &\text{then TRL is non productive iff } s_f \models < \_, r, EF, \_ > \notin Tokens(hist\_f\_fct_1 fct_2)\\ &\qquad\qquad \vee < \_, r, TR, \_ > \notin Tokens(hist\_f\_fct_2 fct_1) \quad (7)\end{aligned}$$

## 4 A network security policy evaluation example

In this example, we consider a traditional case of an enterprise network infrastructure. It is composed of a private network and a DMZ. The whole is connected by an edge router. In the private network, an *App_Server* server is installed and a *FTP* server in the DMZ (Fig.8). The application level security policy is a non hierarchical RBAC policy. It defines two user groups: the group *VPNmembers* and the group *Others*. This organization is only based on the granted privileges. The *App_Server* server is dedicated only to services used by the *VPNmembers* group. The *FTP_Server* has two directories: /confidential and /pub. The directory *confidential* contains data only accessible by the *VPNmembers* users group. Data of the *pub* directory is accessible by everyone. User1, User2, User3 and User4 belong to VPNmembers and Others groups. User5 is only member of the Others group.

The application level security policy can be expressed as:

Permissions(VPNmembers) = {(+all_access,FTP_Server/confidential),
(+all_access, App_Server)}
Permissions(Others) = {(+all_access, FTP_Server/pub)}

Fig.8 shows also the network topology specification and the network level security policy implemented in our language. We have also appended the name used in the CPN specification to each functionality. First of all, we point out that our approach does not take into account devices as entities, but is based on the treatments carried out on data flows. The Private Network, the DMZ and the Internet interconnection infrastructures are specified thanks to channel functionalities because we use their transmission functionality. This approach of specification with large granularity only considers the minimum set of functionalities provided by these infrastructures: their interconnection capability.

On the contrary it is possible to refine a specification as the edge router shows it. It has obviously the interconnection functionality (the channel functionality), setting as a security gateway with filtering capabilities (the three filter functionalities) and encryption mechanisms (the transform functionality,

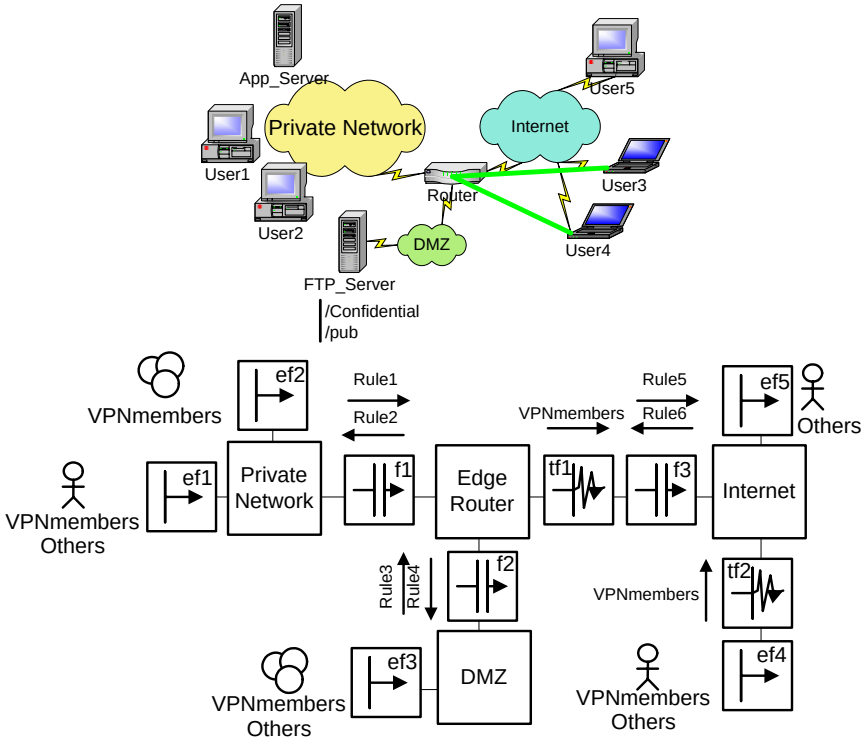


Fig. 8. Architecture and graphical sepcification of our VPN example

for example an IPsec module is installed). The modelling of routing is done by filtering rules on the filter functionalities.

The servers are specified by two PEF. The *App_Server* server has the *VPNmembers* role because only the users with the *VPNmembers* role have access rights. The PEF corresponding to the *FTP server* has the roles *Others* and *VPNmembers* because the permission (+all_access, FTP_Server/pub) is assigned to the *Others* role and (+all_access, FTP_Server/confidential) to the *VPNmembers* role.

The devices of $user_1$ and $user_2$ are represented by a single AEF ($EF_1$) because $user_1$ and $user_2$ have the same roles (*Others* and *VPNmembers*) and these AEF are connected to the same channel functionality thanks to the concept of role which reduces the overall size of the specification. In the same way, the devices of $user_3$ and $user_4$ are specified by the AEF $EF_5$. The device of $user_5$ is specified by the AEF $EF_4$. Arbitrarily adding an AEF with roles whose permissions are reduced makes it possible to define a degree of confidence (see [3] to get the complete definition of the "channel trust property") that can be granted to a channel functionality. In this example,

we do not specify the structure of the Internet network, but it is perceived as an interconnection environment where any connected user has at least the permission to access the /pub directory of the FTP_Server. This allows a great flexibility of specification according to the level of desired and/or known details.

We will now explain the security policy. The filtering rules associated with the filter functionalities of our example are:

- $Rule_1 = EF(AEF, Others), (AEF, VPNmembers)$
- $Rule_2 = EF(PEF, Others)$
- $Rule_3 = EF(PEF, Others), (PEF, VPNmembers), (AEF, VPNmembers)$
- $Rule_4 = EF(AEF, Others), (AEF, VPNmembers)$
- $Rule_5 = EF(PEF, Others), (AEF, Others)$
  $TR(PEF, VPNmembers)$
- $Rule_6 = EF(AEF, Others)$
  $TR(AEF, VPNmembers)$

Two transform functionalities are defined to secure the *VPNmembers* role data flows on the Internet channel functionality. Indeed, users connected to the Internet channel functionality with the *Others* role can never access confidential data at the *FTP_Server* or the *App_Server*.

We have used CPN/tool [40] to create the CPN (Fig. 10) associated to the specification. It shows the initial marking. Fig. 10 points out that the CPN becomes complicated to be manually built for big size specifications. So, we have developed using Java programming language a tool that automates the evaluation task. It takes as an input a specification file (Fig. 9). First, it analyzes the syntax. If the syntax is correct, it generates the equivalent CPN and checks all the properties. Finally, it produces as a result a file (Fig. 11) indicating if the properties are satisfied or not. If a property is not satisfied, the reason is explained.

In our example, the tool indicates (Fig. 11) that the property of confidentiality is satisfied and there is no non-productive transform rule. Nevertheless, the availability is not satisfied because $ef_2$ cannot receive any flow with the role *VPNmembers* from $ef_5$, $ef_1$ cannot receive any flow with the role *VPNmembers* from $ef_3$ and $ef_5$ cannot receive any flow with the role *VPNmembers* from $ef_2$. The partitioning property is not satisfied because of the rule $EF(AEF, Others)$ from $tf_1$ to Internet in the filter functionality $f_3$. And finally, the filtering rule $EF(AEF, VPNmembers)$ from *dmz* to *edge_router* in the filter functionality $f_2$ is non productive. To sum up, this specification

```
/* end-flow functionalities
definition */
<AEF>
#name = ef1
#roles = others, vpn-members;
#connection = private_network

<PEF>
#name = ef2
#roles = vpn-members;
#connection = private_network

<PEF>
#name = ef3
#roles = others, vpn-members;
#connection = dmz

<AEF>
#name = ef4
#roles = others;
#connection = internet

<AEF>
#name = ef5
#roles = others, vpn-members;
#connection = tf2

/*transform functionalities
definition */
<TRANSF>
#name = tf1
#connection1 = edge_router
#connection2 = f3
#rules_1->2 = vpn-members;
#rules_2->1 = NONE;

<TRANSF>
#name = tf2
#connection1 = ef5
#connection2 = internet
#rules_1->2 = vpn-members;
#rules_2->1 = NONE;

/* filter functionalities
defintion */
<FILTER>
#name = f1
#connection1 = private_network
#connection2 = edge_router
#rules_1->2 =
EF (AEF,others), (AEF, vpn-members);
TR NONE;
#rules_2->1 =
EF (PEF, others);
TR NONE;

<FILTER>
#name = f2
#connection1 = dmz
#connection2 = edge_router
#rules_1->2 =
EF (PEF,others), (PEF, vpn-members),
(AEF, vpn-members);
TR NONE;
#rules_2->1 =
EF (AEF, others), (AEF, vpn-
members);
TR NONE;

<FILTER>
#name = f3
#connection1 = tf1
#connection2 = internet
#rules_1->2 =
EF (PEF,others), (AEF, others);
TR (PEF, vpn-members);
#rules_2->1 =
EF (AEF, others);
TR (AEF, vpn-members);

/* channel functionalities
definition */
<CHANNEL>
#name = private_network
#connection = ef1, ef2, f1;

<CHANNEL>
#name = edge_router
#connection = f1, f2, tf1;

<CHANNEL>
#name = internet
#connection = f3, ef4, tf2;

<CHANNEL>
#name = dmz
#connection = ef3, f2;
```

Fig. 9. The specification file

is not secure.

## 5 Related works

Different works focus on suitable tool assistance. The approach of model based management [18,20] utilizes object-oriented models of managed system to support the derivation which is divided into three abstraction levels. The designer graphically defines the three abstraction level models and the tool guides the derivation. In addition, one of the most advanced tools [34] proposes

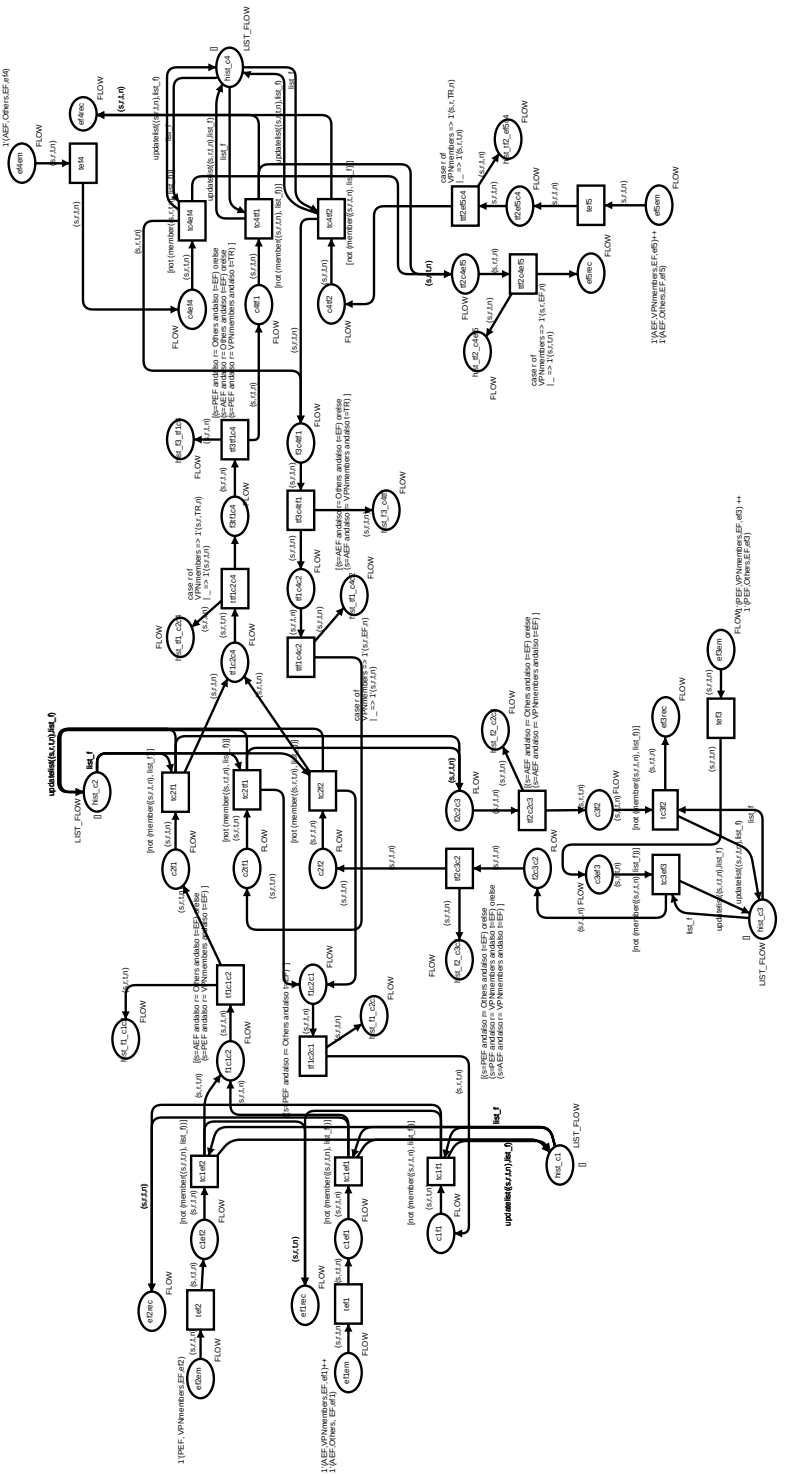

Fig. 10. The CPN model of our VPN example

```
Property of Confidentiality :
-----------------------------
ef5 : OK

ef4 : OK

ef1 : OK

=> The property of confidentiality is satisfied


Property of Availability :
--------------------------
ef5 :
    no flow with the role vpn-members from ef2

ef4 : OK

ef1 :
    no flow with the role vpn-members from ef3

ef3 : OK

ef2 :
    no flow with the role vpn-members from ef5

=> The property of availability is not satisfied


Non Productive Transform Rules :
--------------------------------
tf2 :
    rules 1 -> 2 : OK
    rules 2 -> 1 : OK

tf1 :
    rules 1 -> 2 : OK
    rules 2 -> 1 : OK

=> There is no non productive rule


Non Productive Filtering Rules :
--------------------------------
f3
    rules 1 -> 2 :  OK

    rules 2 -> 1 :  OK

f2
    rules 1 -> 2 : [ EF (AEF, vpn-members) ],
    rules 2 -> 1 :  OK

f1
    rules 1 -> 2 :  OK

    rules 2 -> 1 :  OK

=> There is one or more non productive rule


Partitioning Property :
-----------------------
f3 :
    Rule 1 -> 2 :
       [ EF (AEF ,others) ]
    Rule 2 -> 1 : OK
f2 :
    Rule 1 -> 2 : OK
    Rule 2 -> 1 : OK
f1 :
    Rule 1 -> 2 : OK
    Rule 2 -> 1 : OK

=> There is one or more partitioning problem
```

Fig. 11. The evaluation result file

to logically model network architecture without considering the specificities of the devices such as vendor or version. Then, the designer defines the network security policy (IPsec tunnels, NAT, firewall rules) which is translated into each specific device configuration language. These tools facilitate the design and the deployment of network security policies, but they do not guarantee the correctness of the security policy, i.e., the carried out decisions are relevant.

Most of the network security analysis techniques (for example [1]) only check rules conflicts. They do not consider the global security policy. The work [10,11] is really interesting because it proposes a solution that formally evaluates IPsec VPNs. It models the network on a directed bipartite graph. The nodes of the graph are areas, collections of hosts and networks which are similar in terms of security policy; and devices, which are dual homed hosts or packet filtering/IPsec routers connecting the areas and moving packets between them. Nevertheless, the users are ambiguously considered. That implies that all hosts in a given area own the same set of privileges.

## 6 Conclusion

The design of a security policy becomes increasingly difficult because of the complexity of the factors to consider. The common approach of defining different abstraction levels, up from the objectives till the devices configurations, is used to overcome the problem. Some existing tools implement this approach. Nevertheless, a formal and automatic evaluation of decisions must complete this achievement.

In this paper, we have presented a tool that realizes a formal evaluation of the network security policy. The language is quiet simple, but it owns the CPN formal analysis power. Moreover, we have demonstrated that all the defined security properties are checked at the initial and dead state. Consequently, our approach is not vulnerable to the combinatorial explosion problem and then is applicable to complex studies.

At present, we are testing our approach through different case studies to enhance our method. In addition, our future work will be focused on validating the real configurations on devices. Our tool - being independent from the security technologies implemented on the devices, confines itself to validating security mechanisms constraints. Therefore, we are working on bridging this gap thanks to the Common Information Model (CIM) [35] defined by the DMTF task force to harmonize the management systems. Hence, we could interconnect our work with management platforms.

## References

[1] Al-Shaer Ehab and Hamed Hazem, *Discovery of Policy Anomalies in Distributed Firewalls*, in IEEE INFOCOMM'04, 2004.

[2] Bandara Arosha K , Emil C Lupu, Jonathan Moffet, Alessandra Russo, *A Goal-based Approach to Policy Refinement*, in: IEEE Policy, 2004.

[3] Barrere F., A. Benzekri, F. Grasset, R. Laborde, B. Nasser, *SPIDERNet : A Security Policy Derivation tool for Networks*, in 3rd IEEE Latina America Network Operations and Management Symposium, 2003.

[4] Barrere F., A. Benzekri, F. Grasset, R. Laborde , B. Nasser, *Inter-Domains policy negociation.* in: IEEE Policy, 2003.

[5] Bishop M., "Computer Security: Art and Science", ISBN 0-201-44099-7, ed. Addison-Wesley, 2003.

[6] Burch J., E. Clarke, D. Long, K. McMillan, D. Dill, L. Hwang, *Symbolic Model Checking: 1020 states and beyond*, in Information and Computation, pp 142-170, 1992.

[7] Cholvy L., F. Cuppens, *Analysing consistency of security policy*, IEEE Symposium on Security and Privacy, 1997.

[8] Crook R., D. Ince, B. Nuseibeh, *Modeling Access Policies using Roles in Requirements Engineering*, Information and Software Technology, 2003.

[9] Ferraiolo D. F., R. Sandhu, S. Gavrila, D.R. Kuhn, and R. Chandramouli, "A Proposed Standard for Role-Based Access Control", Proposed 4/4/2003 Draft NIST, 2003, URL: http://csrc.nist.gov/rbac.

[10] Guttman J., *Filtering postures : Local enforcement for global policies*, IEEE Symposium on Security and Privacy, 1997.

[11] Guttman J., A. Herzog, F. Thayer, *Authentication and confidentiality via IPsec*, 6th European Symposium in Computer Security ESORICS, 2000.

[12] Hinrichs S., *Policy Based Management : bridging the gap*, in 15th Annual Computer Security Applications Conference (ACSAC 99), 1999.

[13] Jackson D., *Alloy: a lightweight object modeling notation*, ACM Transactions on Software Engineering and Methodology (TOSEM), v.11 n.2, p.256-290, 2002

[14] Jennings N.R., S. Bussmann, *Agent based Control Systems, why are they suited to engineering complex systems?*, IEEE Control Systems Magazine, vol 23, No 3, 2003.

[15] Jensen K., *An Introduction to the Theoretical Aspects of Coloured Petri Nets*, In: J.W. de Bakker, W.-P. de Roever, G. Rozenberg (eds.): A Decade of Concurrency, Lecture Notes in Computer Science vol. 803, Springer-Verlag 1994, 230-272.

[16] Knorr Konstantin, *Multilevel Security and Information Flow in Petri Net Workflows*, in: Proceedings of the 9th International Conference on Telecommunication Systems - Modeling and Analysis, Special Session on Security Aspects of Telecommunication Systems, 2001.

[17] Kristensen L.M., S. Christensen, K. Jensen, *The Practitioner's Guide to Coloured Petri Nets*, International Journal on Software Tools for Technology Transfer, 2 (1998), Springer Verlag, 98-132.

[18] Lück I., C. Schäfer, H. Krumm, *Model-based Tool-Assistance for Packet-Filter Design* In: IEEE Policy, LNCS 1995, pp. 120-136, Springer-Verlag, 2001.

[19] Krzysztof Juszczyszyn, "Verifying Enterprise's Mandatory Access Control Policies with Coloured Petri Nets", in Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, June 2003.

[20] Lück I., S. Vögel, H. Krumm, *Model-based configuration of VPNs*, in Proc. 8th IEEE/IFIP Network Operations and Management Symposium NOMS 2002, pages 589-602, 2002.

[21] Meadows C., *The NRL Protocol Analyzer: An Overview*, in Journal of Logic Programming 26 (2), pp 113-131, February 1996.

[22] Melliar-Smith P., J. Rushby, *The Enhanced HDM system for specication and verication*, in Proc. VerkShop III, Wat-sonville, CA, Feb. 1985, pp. 41-43, published as ACM Software Engineering Notes, Vol. 10, No. 4.

[23] Moffett J. D., *Control Principle and Role Hierarchies*, 3rd ACM Workshop on Role Based Access Control, 1998.

[24] Moffet J., M. Sloman, *Policy Hierarchies for Distributed Systems Management*, IEEE Journal on Selected Areas in Communications, 11, 9, 1993.

[25] Moore B., E. Ellesson, J. Strassner, A. Westerinen, "Policy Core Information Model – Version 1 Specification", RFC 3060, February 2001.

[26] Nyanchama M., S. Osborn, *The role graph model and conflict of interest*, ACM Transactions on Information and System Security (TISSEC), vol. 2, 1999.

[27] Owre S., J. Rushby, N. Shankar, *PVS: A prototype verification system*, Lecture Notes in Computer Science, Vol. 607 (1992), Springer-Verlag.

[28] Peri R., "Specification and verification of security policies", PhD Dissertation, University of Virginia, January 1996.

[29] Samarati P., S. De Capitani di Vimercati, *Access Control: Policies, Models and Mechanisms*, Foundations of Security Analysis and Design, LNCS 2171, Springer-Verlag. 2001.

[30] Westerinen A., J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, S. Wald-busser, "Terminology for Policy-Based Management", RFC 3198, November 2001.

[31] Wies R., *Using a Classification of Management Policies for Policy Specification and Policy Transformation*, In Proc. of the 4th IFIP/IEEE Int. Symposium on Integrated Network Management,, 1995.

[32] Wijesekera D., S. Jajodia, *A propositional policy algebra for access control*, ACM Transactions on Information and System Security (TISSEC), vol 6,2003.

[33] Yavatkar R., D. Pendarakis, R. Guerin, "A Framework for Policy-based Admission Control", RFC 2753, January 2000.

[34] URL: http://www.solsoft.com

[35] URL: http://www.dmtf.org/standards/cim

[36] IBM Corporation, *An Architectural Blueprint for Autonomic Computing*, IBM white papers, April 2003.

[37] URL: http://wwws.sun.com/software/solutions/n1/index.html

[38] URL: http://www.microsoft.com/windowsserversystem/dsi/design.mspx

[39] http://h71028.www7.hp.com/enterprise/cache/6842-0-0-225-121.aspx

[40] URL: http://wiki.daimi.au.dk/cpntools/cpntools.wiki

# 7 Appendices

## *7.1 Appendix A*

We define the equivalence between the definitions of filtering and transform rules in our specification language and the generated CPN sub-net model.

### *7.1.1 Transform rules translation*

Consider:

- $TF_i$ a transform functionality,
- $funct_1$ and $funct_2$ the two functionalities connected to $TF_i$,
- $C_i$ the configuration of $TF_i$,
- $tf_i$ a transform CPN sub-network,
- $Transf\_funct_1$ the function associated to the $ttf_{fct_1 fct_2}$ post-arc,
- $Transf\_funct_2$ the function associated to the $ttf_{fct_2 fct_1}$ post-arc.

We say that $TF_i \equiv^{TF} tf_i$ iff

$$C_i = \quad funct_1 \rightarrow funct_2\ r_1, r_2, \ldots r_k$$
$$\qquad\quad funct_1 \leftarrow funct_2\ r_{k+1}, \ldots r_n$$

and
$\forall E \in \{AEF, PEF\}$,
$Transf\_funct_1(< E, r_1, EF, \_ >) =< E, r_1, TR, \_ > \wedge$
$\ldots$
$Transf\_funct_1(< E, r_k, EF, \_ >) =< E, r_k, TR, \_ > \wedge$
$Transf\_funct_1(< E, r_{k+1}, TR, \_ >) =< E, r_{k+1}, EF, \_ > \wedge$
$\ldots$
$Transf\_funct_1(< E, r_n, TR, \_ >) =< E, r_n, EF, \_ > \wedge$
$\forall < E, R, T, \_ > \in \{AEF, PEF\} \times ROLES \times \{EF, TR\} \times NAME$
$\quad \setminus \{< E, r_1, EF, \_ >, \ldots, < E, r_k, EF, \_ >, < E, r_{k+1}, TR, \_ >, \ldots,$
$\quad\quad < E, r_n, TR, \_ >\}$,
$Transf\_funct_1(< E, R, T, \_ >) =< E, R, T, \_ >$,
and
$\forall E \in \{AEF, PEF\}$,
$Transf\_funct_2(< E, r_{k+1}, EF, \_ >) =< E, r_{k+1}, TR, \_ > \wedge$
$\ldots \wedge$
$Transf\_funct_2(< E, r_n, EF, \_ >) =< E, r_n, TR, \_ > \wedge$
$Transf\_funct_2(< E, r_1, TR, \_ >) =< E, r_1, EF, \_ > \wedge$
$\ldots \wedge$
$Transf\_funct_2(< E, r_k, TR, \_ >) =< E, r_k, EF, \_ >$,
$\forall < E, R, T, \_ > \in$
$\{AEF, PEF\} \times ROLES \times \{EF, TR\} \times NAME$
$\quad \setminus \{< E, r_1, EF, \_ >, \ldots, < E, r_k, EF, \_ >, < E, r_{k+1}, TR, \_ >, \ldots,$
$\quad\quad < E, r_n, TR, \_ >\}$,
$Transf\_funct_2(< E, R, T, \_ >) =< E, R, T, \_ >$.

### *7.1.2 Filtering rules translation*

Consider:

- $F_i$ a filter functionality,
- $funct_1$ and $funct_2$ the two functionalities connected to $F_i$,
- $C_i$ the configuration of $F_i$,
- and $f_i$ a filter CPN sub-network.

We say that $F_i \equiv^F f_i$ iff

$$
\begin{aligned}
C_i = \quad & funct_1 \quad \rightarrow funct_2 \\
& \qquad Rule_1 = EF(e_1, r_1)(e_2, r_2) \ldots (e_k, r_k) \\
& \qquad\qquad TR(e_{k+1}, r_{k+1}) \ldots (e_n, r_n); \\
& funct_1 \quad \leftarrow funct_2 \\
& \qquad Rule_2 = EF(e\prime_1, r\prime_1) \ldots (e\prime_j, r\prime_j) \\
& \qquad\qquad TR(e\prime_{j+1}, r\prime_{j+1}) \ldots (e\prime_m, r\prime_m);
\end{aligned}
$$

and

$guard(tf_{i_{fct_1 fct_2}}) = [< e_1, r_1, EF, \_ >, < e_2, r_2, EF, \_ >, \ldots, < e_k, r_k, EF, \_ >$
$, < e_{k+1}, r_{k+1}, TR, \_ >, \ldots, < e_n, r_n, TR, \_ >]$
$guard(tf_{i_{fct_2 fct_1}}) = [< e\prime_1, r\prime_1, EF, \_ >, \ldots, < e\prime_j, r\prime_j, EF, \_ >,$
$< e\prime_{j+1}, r\prime_{j+1}, TR, \_ >, \ldots, < e\prime_m, r\prime_m, TR, \_ >]$.

### *7.2 Appendix B*

We present the proofs of both theorems.

#### *7.2.1 Proof of theorem 3.1*

First, we prove that all CPN generated from any specification is K-bounded.

We use the following notation:

- $\mathcal{P}$ is the finite set of places in the CPN that have the color domain $FLOW$ (i.e., all places excluding the places $c_i\_hist$ that have the color domain $FLOW\_LIST$) ,
- $Pre_{\mathcal{P}} : \mathcal{P} \rightarrow 2^{\mathcal{P}}$, the relation that defines the set of places which have one of their post-arcs connected to the same transition as one of the pre-arcs of a place in the CPN,
- $\mathcal{P}_{EF} = \bigcup_{\forall i} \{ef_i^{em}\}$, the finite set of places $ef_i^{em}$,
- $nb\_tok : \mathcal{P} \rightarrow \mathbb{N}$, provides the number of tokens that have passed in one place,
- $< x_1, x_2, \ldots x_n >$ a structural path between $x_1$ and $x_n$ in a CPN where $\forall i > 0, x_i \in \mathcal{P}, x_i \in Pre_{\mathcal{P}}(x_{i+1})$,
- $[x_1 \triangleright x_n]$ the set of the possible structural paths between the places $x_1$ and $x_n$.

By construction, we have:

(i) $\forall p \in \mathcal{P} \backslash \mathcal{P}_{EF}, nb\_tok(p) \leq \sum_{\forall x \in Pre_{\mathcal{P}}(p)} nb\_tok(x)$

(ii) $\forall ef_i^{em} \in \mathcal{P}_{EF}, nb\_tok(ef_i^{em}) = k_i$ where $k_i$ is the the number of tokens in $ef_i^{em}$ in the initial state

(iii) $\forall i, nb\_tok(c_i\_hist) = 1$ because each $c_i\_hist$ contains an ordered list of tokens.

$$\begin{aligned} \text{So, } \forall p \in \mathcal{P} \backslash \mathcal{P}_{EF}, nb\_tok(p) &\leq \sum_{\forall x \in Pre_{\mathcal{P}}} nb\_tok(x) \\ &\leq \sum_{\forall x \in Pre_{\mathcal{P}}(p)} \sum_{\forall y \in Pre_{\mathcal{P}}(x)} nb\_tok(y) \end{aligned}$$

We can note it, $\forall p \in \mathcal{P} \backslash \mathcal{P}_{EF}, \forall y \in \mathcal{P}, < y, \ldots p >, nb\_tok(p) \leq \sum_{[y \triangleright p]} nb\_tok(y)$

By recursion, we obtain

$$\forall p \in \mathcal{P} \backslash \mathcal{P}_{EF}, \forall ef_i^{em} \in \mathcal{P}, < ef_i^{em}, \ldots p >, nb\_tok(p) \leq \sum_{[ef_i^{em} \triangleright p]} nb\_tok(ef_i^{em})$$

(i) if there is only one structural path between two places then $\sum_{[ef_i^{em} \triangleright p]} nb\_tok(ef_i^{em}) = \sum k_i$.

(ii) if there exists cycles in structural paths - e.g. $< x_2, x_3 >$ is a cycle in the path $< x_1, x_2, x_3, x_2, x_3, x_4 >$ - then there is an infinite number of possible paths between $x_1$ and $x_4$, as $< x_1, x_2, x_3, x_2, x_3, x_2, x_3, x_2, x_3, x_4 >$ because a token that can pass two time through a place can pass infinitely. However by construction, a cycle in the CPN implies that the associated specification contains a cycle too (i.e., there are different paths between two functionalities). And always by construction, there are at least two channel functionalities in a cycle. A channel functionality retransmits a token with a specific color $< a, b, c, d >$ once. Considering that each flow can only take two possible colors $< a, b, EF, d >$ and $< a, b, TR, d >$, a cycle can be covered once by a data flow. Consequently, $\sum_{[ef_i^{em} \triangleright p]} nb\_tok(ef_i^{em}) < \infty$

To sum up:

(i) $\forall p \in \mathcal{P} \backslash \mathcal{P}_{EF}, nb\_tok(p) < \infty$

(ii) $\forall ef_i^{em} \in \mathcal{P}_{EF}, nb\_tok(ef_i^{em}) = k_i$

(iii) $\forall i, nb\_tok(c_i\_hist) = 1$

So, any CPN associated to a specification is K bounded. As a consequence, the CPN is K-bounded and there a reachability graph can be computed.

Moreover, each token will be consumed by an end-flow or stopped by a filter or a channel functionality. They are also consumed by all the historic places. Then there is one or more dead state.

In addition, there is no choice (i.e., a place with different post-arcs) in the produced CPN, and tokens are arranged in order in the flow list of the $c_i\_hist$ places. Consequently, there is only one dead state.

### *7.2.2 Proof of theorem 3.2*

For this proof, we use the theorem 3.1 which demonstrates that there is only one dead state in the reachability graph.

The property of confidentiality and the definition of non productive filtering/transform rules state that a specific place must never contains a specific colored token. In each case, this place does not have any post-arc. For that reason, if this place contains at a specific state a colored token then for all future states it contains this token. As a consequence, if the place never contains a colored token then the place does not contain the token at the dead state. And if the dead state does not contain a colored token, then there is no state such that the place contains the token.

Inversely, the properties of availability and partitioning impose that there must exists a state such that a specific place with no post-arc contains a specific colored token. If such a state exists then the dead state satisfies the property. Moreover, if the place contains the token in the dead state then such a state exists.

Finally, the property of integrity states that the place $ef^{rec}$ must never contain some colored tokens accordingly to its assigned roles. This place does not have pre-arc. So, if the place contains the token in a state, then there is no past state such that the place does not contain this token. Hence, if the place does not contain the token at the initial state then the place will never contain the token. And if the place never contains the token then the place doesn't contain the token at the initial state.