A Formal Data Flow-Oriented Model For Distributed Network Security Conflicts Detection

Hicham El Khoury, Romain Laborde,
François Barrère, Abdelmalek Benzekri
IRIT – University Paul Sabatier
Toulouse, France
hkhoury@ul.edu.lb, Romain.Laborde@irit.fr,
Barrere.Francois@irit.fr, Abdelmalek.Benzekri@irit.fr

Maroun Chamoun
Saint Joseph University
Beirut, Lebanon
maroun.chamoun@usj.edu.lb

Abstract— **Network security is inherently a distributed function that involves the coordination of a set of devices, each device affording its specific security features. The complexity of this task resides in the number, the nature, and the interdependence of the mechanisms. Any security service can interfere with others creating a breach in the whole network security. We propose a formal data flow oriented model to detect network security conflicts. Network security services are represented by specific abstract functions that can modify the data flow. We have specified our model in hierarchical Colored Petri Nets to automate the conflicts detection analysis. This approach has been tested on various NAPT/IPsec scenarios to prove that without any a priori knowledge these conflicts can be detected.**

Keywords - network security; security conflict detection; data flow modeling; Colored Petri Nets.

I. INTRODUCTION

Network security is inherently a distributed function that involves the coordination of a set of devices, each device affording its specific security features. Any equipment (end and core devices) involved in a security solution requires a precise configuration. This configuration, determining its behavior, has a local impact on the security service provided by the equipment but also can affects the global network security. If a rule on equipment is poorly defined, the global security might be compromised (principle of the weakest link in the security chain).

The network security conflicts detection is therefore a major problem. Conflicts can be local to one security service, i.e. two rules for the same security mechanism on the same equipment may be incompatible (e.g. one filtering rule permits a data flows whereas another one on the same firewall blocks it). However, conflicts can be distributed too. In this case, the incompatibility can occur between different mechanisms on different equipment playing a role at different levels in the OSI layers (IPsec tunnels blocked by firewalls, HTTP proxy unable to filter HTTP traffic because it has been encrypted by an IPsec gateway, etc). These distributed conflicts are much harder to cope with because they expect to consider the dependencies between different equipments and/or different mechanisms.

It is therefore essential to develop tools to express and validate network security policies. One difficulty resides in the nature of network security information. How to express management information while taking into consideration constraints such as the heterogeneity of solutions, interdependencies between security mechanisms, and the sustainability of expression languages confronted to the fast evolution of technologies? The right level of abstraction should be produced, both independent of the security mechanisms and at the same time faithfully representative of the reality.

To address this problem, we have proposed a formal data flow oriented model. The first version of our model has been presented in [16] and was enhanced in [17]. In this approach, a data flow is represented as a sequence of logical elements to match physical data flow which is a sequence of bytes grouped according to the specifications of the network protocols. The security mechanisms are represented as functions handling these flows. Constraints applied to data flows and security mechanisms point out conflicts that can occur. In this article, we validate our model by specifying it in hierarchical Colored Petri Nets. This formal language is suitable for representing data flows and associated tools such as CPNtools can automate the validation task. In addition, we present how to detect distributed conflicts using our approach thought various NAPT/IPsec scenarios.

The rest of this article is organized as follows. Section 2 is dedicated to related works. Section 3 describes our modeling. Section 4 presents example of modeling. Section 5 illustrates our approach to conflict detection, on implementing IPSec and NAPT, using Colored Petri Networks. Finally, Section 6 concludes and presents our working tracks.

II. RELATED WORKS

Several works have focused on detecting misconfigurations. Their approach consists in modeling the configurations of devices and the network topology. Al-Shaer et al. [1] [5] proposed a classification of the anomalies that can appear in the configuration of one or more firewalls. Alfaro et al. [2] have improved this classification and introduced IDS. Fu et al. [3] endeavored to address the problem of inconsistency of IPSec tunnels and firewalls. Preda [7] considers firewalls, IPsec and IDS. These models describe correctly the reality. However, they are closely attached to limited set of technologies and it is difficult to adapt them to other technologies.

The other approach followed by distributed conflicts detection research considers data flow or IP datagram as the primary entity. Security technologies are then represented has functions applied on data flows. The advantage of this approach is the processing on data flow is related to the

 ISBN: 978-1-61208-186-1

abstract data flow model, not to the underlying technology. However, the strength of these solutions depends on the quality of the data flow modeling approach: not limited to one protocol or layer, but also not de-correlated from reality.

Guttman and Herzog [4] proposed an abstract model for IP datagrams by a 3-tuple $<l,k,\theta>$ where l represents the current location of the datagram, k represents the current state of treatment of the datagram and θ is the following IP header representing the history of the actions performed on the datagram. An IP header is also abstracted by a 3-tuple $<s,d,p>$ where s and d are respectively the source and destination and p represents other data. Nevertheless, the formalization is limited to some information contained in the IP header.

Laborde et al. [6] proposed a formal solution based on Colored Petri Nets for the specification coupled with the CTL logic for the analysis. Like Guttman and Herzog, this approach focuses on analyzing the data flow. A network is represented as an interconnection of generic functionality on the data flow: endings flow (terminal devices such as workstations, or servers), filtering functionality (such as firewalls or application gateways), transformation functionality (such as IPsec, NAT, etc.) and channels functionality (to represent the communication media such as WiFi, abstraction of a network). A data flow is represented by a 4-tuple < *efs, efd, r, t* > where *efs* and *efd* are source and destination end flows functionalities, *r* represents the permission used to generate this flow, and *t* represents the list of transformation applied to the data flow. This formalism allows the approach to be independent from technologies. However, the level of abstraction being too high, it does not represent explicitly what has been changed by a transformation. This level of abstraction issue is highlighted by "feasibility analysis" that was introduced in the refinement process, The goal of the feasibility analysis is to validate that something specified at the abstracted level can actually be implemented on the real device.

In a more recent article, Al Shaer et al. [15] have proposed a similar approach. They model the network as a finite state machine where each state depends on the location of IP packets $(ip_s, port_s, ip_d, port_d, location)$. However, they do not consider IP payload in their modeling. As a consequence, they had to add an extra valid bit to address the problem of IPsec encapsulation modeling. Security involving different mechanisms at different layers (TCP/IP stack or OSI model), this modeling approach is limited for describing the entire encapsulation stack.

III. A FORMAL DATA FLOW-ORIENTED MODEL

Our goal is to define a technology independent formalism to detect distributed conflicts (multi-mechanisms, multi-OSI layers). Our modeling approach is data flow oriented; the treatments of different network mechanisms are then seen as functions on flows.

It is important to consider that network security mechanisms are not applied to one single network layer only (firewalls and IPSec are both mechanisms at the IP level). Network security is multi-level (or cross-layer). For example, a VPN can be an IPsec but also L2TP, SSL or SSH. Filtering can be done at the IP level through a firewall and at the data link layer via a switch, or at the application level by a dedicated gateway.

It is also necessary to consider the influence of non "pure security labeled" devices. For example, the installation of a router into an Ethernet network composed only of switches may require the changing of MAC addresses filtering rules. Another example, NAT may have an adverse effect on the enforcement of IPSec VPNs.

A. Analysis of the problem

In concrete terms, a data flow is a contiguous set of bytes with variable size conveyed over a network. This sequence of bytes is divided into logical blocks according to encapsulation protocols. For example, a data flow corresponding to a HTTP request can be seen as < *Ethernet protocol block, IP protocol block, TCP protocol block, HTTP protocol block* >. The bytes in a logical block are not necessarily contiguous. Then, each protocol divides the block of bytes associated with fields in accordance to its description. For example, the control information of the Ethernet protocol are distributed into 14 bytes (destination MAC address, source MAC address, identifier of the encapsulated protocol) at the beginning of the frame and 4 bytes for the control field at the end.

Also, data flow is not static. A data flow evolves during its journey through the network according to the mechanisms implemented on network devices. Some mechanisms can:

- Add new blocks of bytes. For example, an IPsec gateway adds the AH header between the IP block and the UDP/TCP block when this protocol is used in transport mode,
- Remove blocks of bytes. e.g., an HTTP proxy removes the IP block when it receives a data stream,
- Modify fields. e.g., NAT changes the value of the source IP address field in the IP block or a router changes the time-to-live field,
- Authenticate fields. e.g., the AH protocol authenticates certain fields of IP and all other fields of the encapsulated protocols,
- Encrypt fields. For example, the ESP in transport mode encrypts all the fields of the encapsulated protocols,
- Etc…

In addition, the network mechanisms perform these treatments according to a subset of data flow fields they can perceive. E.g., a stateless firewall analyses only the protocol id, IP source and IP destination fields in the IP block, and port source and destination fields in the UDP/TCP block.

B. Modeling data flows

We propose a data flow model that is independent from the underlying protocols. We want this model to be able to anticipate future protocols. The difficulty is to reach the good level of abstraction between security mechanisms independence and reality closeness.

Foremost, we define our core entities:

- $\mathcal{A}$ is the set of possible attributes. An attribute $a \in \mathcal{A}$, represents a couple <*name, value*> where *name* is a field that can be found in a protocol, and *value* is its content,
- $\mathcal{P}$ is the set of protocols, i.e., the set of logical blocks. An instance of protocol $p \in \mathcal{P}$ is a couple <*protoid, attributes*> where *protoid*=<*name, id*> is the name of the protocol and an unique identifier and *attributes* is

defined on the Power-set of $\mathcal{A}$, i. e., $attributes \in \mathbb{P}(\mathcal{A})$,

- $\mathcal{E} = \mathcal{P}^{\mathbb{N}}$, is the set of finite sequences over $\mathcal{P}$. This set represents all the possible encapsulation chain of protocols, i.e. sequences of logical blocks. For reasons of notation simplicity, we use: $\text{next}(p_i) = p_{i+1}$ and $\text{rest}(p_i) = < p_{i+1}, \dots, p_n >$ for a given sequence $e = < p_1, p_2, \dots, p_i, \dots, p_n >$,
- $\mathcal{S}$ is the set of security algorithms addressing the encapsulation chain of protocols (for instance, DES, 3DES, HMAC-SHA1, etc.).

Definition 1: *Formal definition of data flows*

Based on above definitions, we present the set of data flows as: $\mathcal{F} \subseteq \mathcal{E} \times \text{AUTHN} \times \text{CONF}$ such that:

- $\mathcal{E}$ is the encapsulation chain of protocols,
- $\text{AUTHN} \subseteq (\mathcal{A} \times \mathcal{P} \times \mathcal{A} \times \mathcal{P} \times \mathcal{S})$ represents the attributes of the data flow that have been authenticated such that $(a_1, p_1, a_2, p_2, s) \in \text{AUTHN}$ indicates that attribute a_1 of protocol p_1 guarantees the integrity of attribute a_2 of protocol p_2 via the security algorithm s,
- $\text{CONF} \subseteq \text{BAG}(\mathcal{A} \times \mathcal{P} \times \mathcal{S})$ represents the attributes of the data flow that have been encrypted, such that $(a, p, s) \in \text{CONF}$ indicates that attribute a of protocol p is encrypted via the security algorithm s. We used a multi-set because an attribute can be encrypted several times by the same algorithm. A multi-set is a set of elements where an element may appear several times.

A treatment on a data flow is then seen as a particular function of $\mathcal{F}$ to $\mathcal{F}$, called *transform function* as in [6]. This function represents the capability to modify the data flows. It can symbolize encryption protocols such as IPsec where one transform function adds some security services (e.g. confidentiality) and another removes it, or the NAT where only one transform function is concerned. According to the underlying technology, each treatment considers a subset of attributes from data flow as input (e.g., for IPSec, attributes are source address, destination address and transport protocol field of the IP header as well as source port and destination port of the TCP/UDP header) and modifies the data flow. For example, when using ESP, treatment adds new protocols in the encapsulation chain of protocols and new instances in the relationship AUTHN and new attributes in multi-set CONF.

Examples:

Given flow $f = (e, \text{AUTHN}, \text{CONF}) \in \mathcal{F}, e = < p_1, p_2 >$, *where* $p_1 = ((p, 1), \{(\alpha, v1), (\beta, v2), (\gamma, v3)\})$ and $p_2 = ((p, 2), \{(\kappa, v4), (\mu, v5)\})$:

1. If $\text{AUTHN} = \{\ \}$, and $\text{CONF} = \{\ \}$, then this data flow includes two encapsulated protocols for which no field is protected,
2. If $\text{AUTHN} = \{(\kappa, p_2, \gamma, p_1, s_1), (\kappa, p_2, \mu, p_1, s_1)\}$ and $\text{CONF} = \{(\mu, p_2, s_2)\}$, then field κ in protocol p_2 authenticates fields γ and μ in protocol p_1 by algorithm s_1 and field μ in protocol p_1 is encrypted by algorithm s_2.

In the rest of the article, we use the following notation to simplify the readability:

- we designate by p_1 the protocol element whose *protoid* equals to (p,1),
- attributes(p) for the set of attributes of a protocol p. E.g. attributes(p_1)= $\{(\alpha, v1), (\beta, v2), (\gamma, v3)\}$,
- when there is no ambiguity, we use the name of the attribute instead of the couple (name,value).

Definition 2: *Data flow integrity*

Data flow integrity indicates that no authenticated attribute has been changed. This is described in our model as follows: Let $f = (e, \text{AUTHN}, \text{CONF}) \in \mathcal{F}$, integrity of f is satisfied iff $\forall (a, a') \in \mathcal{A} \text{ x } \mathcal{A}, \forall (p_1, p_2) \in \mathcal{P} \text{ x } \mathcal{P}, \forall s \in \mathcal{S}, (a, p_1, a', p_2, s) \in \text{AUTHN} \Longrightarrow a \in \text{attributes}(p_1) \wedge a' \in \text{attributes}(p_2)$.

We do not provide any definition of data flow confidentiality in this article. Confidentiality refers to non-disclosure of sensitive information. Sensitive information in the context of data flow is a subset of the protocols fields/payload that are required to be confidential. In our attribute-based modeling, attributes within the multi-set CONF represent the encrypted information in the data flow. It can be noticed that our model faithfully represents reality. A real data flow cannot be completely encrypted (if the IP destination field is encrypted, routers won't be able to route the packet). However, particular values of specific attributes might be considered as confidential (e.g., if the knowledge that two devices are communicating over the Internet is confidential, it is important to hide their IP addresses values for example in an IPsec tunnel where only the IP addresses of the two IPsec gateways will be revealed). Thus, the set attributes required to confidential depends on external security requirements that are out of scope of this article.

IV. CASE STUDY

In this section, we present the modeling of security mechanisms to validate the expression capability of our approach. We describe examples related to IPsec [10] (namely the AH and ESP protocols) and NAPT. Although, conflicts between these technologies are well know, our intent is to explain how our approach can be used.

.

In the following examples, we use IP, TCP, UDP, AH, ESP protocols and a data flow f = (< …,ip_1,… >, AUTHN,CONF). In our formalism, they can be defined as follows:

- attributes(ip_i)={version,hlength,tos,tlength,id,flags,offset, ttl,proto,checksum,ips,ipd,options},
- attributes(tcp_i)={ports,portd,seq,ack,hlength,reserved, tcpflags,win,options,checksum},
- attributes(udp_i)={ports,portd,len,checksum},
- attributes(ah_i)={nexthdr,payloadlength,reserved,spi, seq,ad},
- attributes(esp_i)={ spi,seq,padlength,nextheader,ad}.

A. Specification of AH

AH (Authentication Header) [11] is designed to ensure integrity and authenticity of IP datagrams without data encryption. The Authentication Data (AD) field guarantees the integrity of the datagram. The AH protocol has two modes: transport and tunnel.

1) In the **transport mode**, AH is inserted after the IP header and before next layer (Fig. 1).and protects the entire IP packet except mutable fields (i.e. the fields DSCP, ECN, Flags, Offset, TTL, Header Checksum).

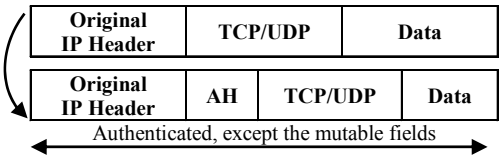

Figure 1. IP datagram before and after applying AH in transport mode

Definition 3: *Specification of AH in the transport mode*

The application of AH in transport mode on the flow *f* (as defined above) is the transformation function $tf_{AH}^{transport} : \mathcal{F} \rightarrow \mathcal{F}$, generating the flow $tf_{AH}^{transport}$=*f'*=(<…,ip_1,ah, p…>, AUTHN', CONF') where:

- $attributes(ip_1') = attributes(ip_1) \backslash \{(proto, x)\} \cup \{(proto, 51)\}$ where 51 is the value of AH protocol,
- $AUTHN' =$ $AUTHN \cup$ $\cup_{\forall x \in attributes(ip_1') \backslash \{DSCP,ECN,\ldots,checksum\}} \{(ad, ah, x, ip_1', s)\} \cup$ $\cup_{\forall y \in attributes(ah) \backslash \{ad\}} \{(ad, ah, y, ah, s)\} \cup$ $\cup_{\forall p \in rest(ah) | \forall z \in attributes(p)} \{(ad, ah, z, p, s)\}$ It indicates that integrity of all immutable fields of the IP protocol and every fields of all protocols, which are encapsulated by AH, is guaranteed by using security algorithm *s*.

2) In the **tunnel mode**, the inner IP header carries the ultimate IP source and destination addresses, while an outer IP header contains the addresses of the IPsec peers (Fig. 2) and protects the entire inner IP packet, including the entire inner IP header. The position of AH in mode tunnel, relative to the outer IP header, is the same as for AH in the transport mode. In fact, in AH Tunnel mode the entire original IP header and data becomes the "payload" for the new packet. The new IP header is protected exactly the same as the IP header in Transport mode.

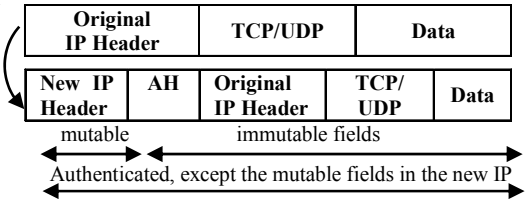

Figure 2. IP datagram before and after applying AH in tunnel mode

Definition 4: *Specification of AH in the tunnel mode*

The application of AH in the tunnel mode on the flow *f* (as defined above) is the transformation function: $tf_{AH}^{tunnel} : \mathcal{F} \rightarrow \mathcal{F}$, generating flow $tf_{AH}^{tunnel}(f) = f' = (< \ldots, ip_2, ah, ip_1, p, \ldots >, AUTHN', CONF)$ where:

- $attributes(ip_2) = \{(ips, sourcegateway), (ipd, destinationgateway), (proto, 51), \ldots\}$,
- $AUTHN' =$ $AUTHN \cup$ $\cup_{\forall x \in attributes(ip_2) \backslash \{DSCP,ECN,\ldots,checksum\}} \{(ad, ah, x, ip_2, s)\} \cup$ $\cup_{\forall y \in attributes(ah) \backslash \{ad\}} \{(ad, ah, y, ah, s)\} \cup$ $\cup_{\forall p \in rest(ah) | \forall z \in attributes(p)} \{(ad, ah, z, p, s)\}$ This indicates that integrity of all immutable fields of the new IP header, and every fields of all protocols, which are encapsulated by AH (including the original IP header), is guaranteed by using security algorithm *s*.

B. Specification of ESP

ESP protocol (Encapsulating Security Payload) [12] provides authentication and encryption of data carried in IP datagram. Like AH protocol, the Authentication Data (AD) field guarantees the integrity of the datagram and ESP has two modes: transport and tunnel.

1) In the **transport mode**, the ESP bounds the transport and data layers (Fig. 3). ESP authenticates the data transported in the IP datagram but not the IP header. In addition, it encrypts the data protocol transport layer.

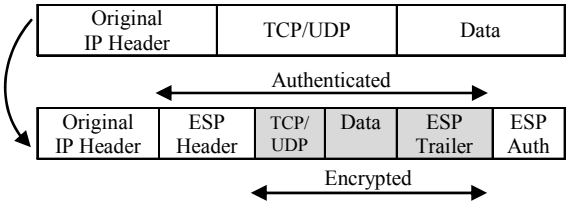

Figure 3. IP datagram before and after applying ESP in transport mode

Definition 5: *Modeling of ESP in transport mode*

Our model considers protocols as logical blocks. So, we do not differentiate between the header and the tail of ESP.

The application of ESP in the transport mode on the flow *f* (as defined above) is the transformation function $tf_{ESP}^{transport} : \mathcal{F} \rightarrow \mathcal{F}$, generating flow $tf_{ESP}^{transport} = f' = (< \ldots, ip_1', esp, p, \ldots >, AUTHN', CONF')$ where:

- $attributes(ip_1') = attributes(ip_1) \backslash \{(proto, x)\} \cup \{(proto, 50)\}$ where 50 is the value of ESP protocol.
- $AUTHN' =$ $AUTHN \cup$ $\cup_{\forall x \in attributes(esp)\{ad\}} \{(ad, esp, x, esp, s_1)\} \cup$ $\cup_{\forall p \in rest(esp) | \forall y \in attributes(p)} \{(ad, esp, y, p, s_1)\}$ indicating integrity of every fields of all protocols, which are encapsulated by ESP, is guaranteed by using security algorithm s_1,
- $CONF' =$ $CONF \cup \cup_{\forall x \in attributes(esp) \backslash \{spi,seq,ad\}} \{(x, esp, s_2)\} \cup$ $\cup_{\forall p \in rest(esp) | \forall y \in attributes(p)} \{(y, p, s_2)\}$. This indicates that every fields of all protocols, which are encapsulated by ESP, are encrypted using security algorithm s_2,

2) In the **tunnel mode**, the inner IP header carries the ultimate IP source and destination addresses, while an outer IP header contains the addresses of the IPsec

gateways (Fig.4) and protects the entire inner IP packet, including the entire inner IP header. The position of ESP in the tunnel mode, relative to the outer IP header, is the same as for ESP in transport mode. The integrity of the datagram is checked against the field Authentication Data (AD). In fact, in ESP Tunnel mode the entire original IP header and data becomes the "payload" for the new packet. The new IP header is not protected.

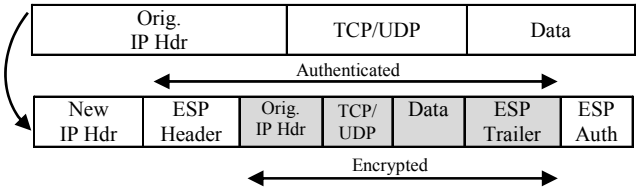

Figure 4. IP datagram before and after applying ESP in tunnel mode

Definition 6: *Modeling of ESP in the tunnel mode*

The application of ESP in tunnel mode on a flow f (as defined above) is the transformation function $tf_{ESP}^{tunnel}: \mathcal{F} \rightarrow \mathcal{F}$, generating flow $tf_{ESP}^{tunnel}(f) = f' = (< \ldots, \mathrm{ip}_2, \mathrm{esp}, \mathrm{ip}_1, \mathrm{p}, \ldots >, \mathrm{AUTHN}', \mathrm{CONF}')$ where:

- $\mathrm{attributes}(\mathrm{ip}_2) = \{(\mathrm{ips}, \mathrm{sourcegateway}), (\mathrm{ipd}, \mathrm{destinationgateway}), (\mathrm{proto}, 50), \ldots\}$,
- $\mathrm{AUTHN}' = \mathrm{AUTHN} \cup \bigcup_{\forall x \in \mathrm{attributes(esp)} \setminus \{\mathrm{ad}\}} \{(\mathrm{ad}, \mathrm{esp}, \mathrm{x}, \mathrm{esp}, \mathrm{s}_1)\} \cup \bigcup_{\forall p \in \mathrm{rest(esp)} \mid \forall y \in \mathrm{attributes(p)}} \{(\mathrm{ad}, \mathrm{esp}, \mathrm{y}, \mathrm{p}, \mathrm{s}_1)\}$ indicating that integrity of every fields of all protocols, which are encapsulated by ESP, is guaranteed by using security algorithm s_1,
- $\mathrm{CONF}' = \mathrm{CONF} \cup \bigcup_{\forall x \in \mathrm{attributes(esp)} \setminus \{\mathrm{spi,seq,ad}\}} \{(\mathrm{x}, \mathrm{esp}, \mathrm{s}_2)\} \cup \bigcup_{\forall p \in \mathrm{rest(esp)} \mid \forall y \in \mathrm{attributes(p)}} \{(\mathrm{y}, \mathrm{p}, \mathrm{s}_2)\}$. This indicates that every fields of all protocols encapsulated by ESP are encrypted using security algorithm s_2.

C. Modeling of NA(P)T

NA(P)T (Network address and port translation) [8] transforms the IP source address of an IP datagram to allow the communication between an equipment with private IP address with another one connected to the Internet.

The operation of this system can be summarized as follows:

1. NAPT generates dynamically a source port,
2. NAPT records the association (old IP source address, old source port, new IP address, new source port),
3. NAPT modifies the fields source port and checksum fields of the UDP / TCP,
4. NAPT modifies the source IP address and the checksum in the IP header.

Definition 7: *Basic NAPT*

Consequently, we can represent the NAPT processing system by the transformation function tf_{NAPT}^{B} as follows:

- Pre-condition 1: the protocol following IP header should be either tcp or udp.
  $\forall f = (< \ldots \mathrm{ip}, \mathrm{next(ip)} \ldots >, \mathrm{AUTHN}, \mathrm{CONF})$, with: $\{\mathrm{ports}, \mathrm{portd}, \mathrm{checksum}\} \in \mathrm{attributes(next(ip))}$.
- Pre-condition 2: NAPT must be able to read the source IP address and the source port:
  $\forall f = (< \ldots, \mathrm{ip}, \mathrm{next(ip)} \ldots >, \mathrm{AUTHN}, \mathrm{CONF})$,
  $\nexists s \mid (\mathrm{ips}, \mathrm{ip}, s) \in \mathrm{CONF} \vee (\mathrm{ports}, \mathrm{next(ip)}, s) \in \mathrm{CONF}$.

NAPT transforms the following fields: source IP address, checksum of IP, the source port, and the checksum of transport protocol. Therefore, tf_{NAPT}^{B} transforms a flow $\mathrm{f} = (< \ldots, \mathrm{ip}, \mathrm{next(ip)} \ldots >, \mathrm{AUTHN}, \mathrm{CONF})$ into a data flow $\mathrm{f}' = (< \ldots, \mathrm{ip}', \mathrm{next(ip)}' \ldots >, \mathrm{AUTHN}, \mathrm{CONF})$ such that:

- $\mathrm{attributes}(\mathrm{ip}') = \mathrm{attributes}(\mathrm{ip}) \setminus \{(\mathrm{ips}, \mathrm{value}), (\mathrm{checksum}, \mathrm{value})\} \cup \{(\mathrm{ips}', \mathrm{new\_value}), (\mathrm{checksum}', \mathrm{new\_value})\}$,
- $\mathrm{attributes}(\mathrm{next(ip)}') = \mathrm{attributes}(\mathrm{next(ip)}) \setminus \{(\mathrm{ports}, \mathrm{value}), (\mathrm{checksum}, \mathrm{value})\} \cup \{(\mathrm{ports}', \mathrm{new\_value}), (\mathrm{checksum}', \mathrm{new\_value})\}$

This treatment considers that protocol TCP or UDP follows immediately the IP header, which leads to the problem of the evolution of the arrangement of protocols encapsulation. We therefore propose an advanced version of NAPT processing that is not limited by this assumption. This version, called advanced NAPT, is able to search the TCP or UDP protocol deeper in the data flow.

Definition 8: *Advanced NAPT*

The transformation function tf_{NAPT}^{adv} represents an advanced NAPT which is defined as:

- Pre-condition 1: the IP protocol encapsulates directly or indirectly the TCP or UDP protocol: e.g.,
  $\forall f = (< \ldots, \mathrm{ip}, \ldots, \mathrm{p}, \ldots >, \mathrm{AUTHN}, \mathrm{CONF})$,
  $\exists p \in \mathrm{rest(ip)} \mid \{\mathrm{ports}, \mathrm{portd}, \mathrm{checksum}\} \in \mathrm{attributes(p)}$. thus p is a transport protocol.
- Pre-condition 2: NAPT must be able to read the IP source address and the source port: e.g.,
  $\forall f = (< \ldots, \mathrm{ip}, \ldots, \mathrm{p}, \ldots >, \mathrm{AUTHN}, \mathrm{CONF})$,
  $\nexists s$ such that $(\mathrm{ips}, \mathrm{ip}, s) \in \mathrm{CONF} \vee (\mathrm{ports}, \mathrm{p}, s) \in \mathrm{CONF}$

NAPT transforms the fields IP source address, IP checksum and the source port and checksum of the transport protocol. Therefore, tf_{NAPT}^{adv} transforms a flow $f = (< \ldots, \mathrm{ip}, \ldots \mathrm{p} \ldots >, \mathrm{AUTHN}, \mathrm{CONF})$ into a flow $f' = (< \ldots, \mathrm{ip}', \ldots \mathrm{p}' \ldots >, \mathrm{AUTHN}, \mathrm{CONF})$ such that:

- $\mathrm{attributes}(\mathrm{ip}') = \mathrm{attributes}(\mathrm{ip}) \setminus \{(\mathrm{ips}, \mathrm{value}), (\mathrm{checksum}, \mathrm{value})\} \cup \{(\mathrm{ips}', \mathrm{new\_value}), (\mathrm{checksum}', \mathrm{new\_value})\}$,
- $\mathrm{attributes}(\mathrm{p}') = \mathrm{attributes}(\mathrm{p}') \setminus \{(\mathrm{ports}, \mathrm{value}), (\mathrm{checksum}, \mathrm{value})\} \cup \{(\mathrm{ports}', \mathrm{new\_value}), (\mathrm{checksum}', \mathrm{new\_value})\}$

V. CONFLICT ANALYSIS IN COLORED PETRI NETS (CPN)

In this section, we demonstrate that it is possible to detect conflicts between security mechanisms without a priori knowledge. The problems between IPsec and NAPT are well known [9] but they are not trivial without using the human expertise. Our goal here is to detect these conflicts by applying our formalization only. Our approach, being independent of the underlying technologies, can handle conflicts between other technologies.

 ISBN: 978-1-61208-186-1

In order to automate the conflict detection task, we have specified our formalism in colored Petri nets; this formal language being adapted to data flows oriented approach [6] and featured with tools (CPN tools [14]) to validate our formal methodology.

A. Introduction to CPN

Colored Petri Net (CPN) is a formal specification language consisting of a set of tokens whose type is represented by a color, a set of transitions, and a set of places with a domain (which defines the types of tokens that can be stored in that place) and a set of arcs connecting places and transitions. It allows creating formal models of systems.

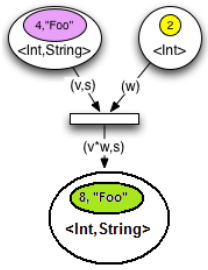

Figure 5. Example of CPN specification

The state of a system (Fig. 5) is represented by the distribution of tokens in places. It changes when a transition is fired. A Boolean expression called guard may be associated with a transition to set the conditions required to fire the transition. If the tokens contained in places connected by incoming arcs in the transition satisfy the guard, then they are removed from these places and new tokens are created in the places connected to outgoing arcs of the transition.

Our choice of Colored Petri Nets formalism [13] to address the design of modeling data flow is motivated by the following reasons: Colored Petri Nets are well-known for their graphical and analytical capabilities for the specification and verification of concurrent, asynchronous, distributed, parallel and nondeterministic systems. Various features contribute to such a success include graphical nature, the simplicity of the model and the firm mathematical foundation. It also provides modularity in design.

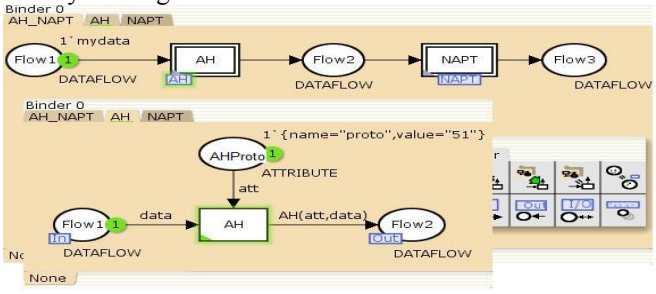

Figure 6. Navigating through marking menus

In addition to colors, it is possible to create hierarchical descriptions, i.e., structure a large description as a set of smaller pieces by using the facilities within CPN Tools through well-defined interfaces and relationships to each other. This is similar to the use of modules in a programming language. Conceptually, CPNs with substitution transitions are CPNs with multiple layers of detail. It enhances the readability of the CPN specification. Figure 6 shows an example of navigation from a super-page to a subpage. The substitution transition AH (with double line in the CPN at the top of figure 6) is actually a black box view of a more detailed CPN (at the bottom) that specifies AH in transport mode.

B. Net structure and declaration

We simulate and validate our CPN model with "CPN Tools" [14]. The CPN development environment uses an extension of ML language to formally specify colors of tokens, guards at transitions, and functions on arcs. Fig. 7 presents the definition data flow in CPN-ML.

```
// Definition of attributes
color ATTRIBUTE = record name:STRING * value:STRING;
// Definition of protocol identification
color PROTOCOLID = record name :STRING * id :INT;
// Definition of the list of attributes
Color ATTList = list ATTRIBUTE;
// Definition of protocols
color PROTOCOL = record protoid:PROTOCOLID*value:ATTList;
// Definition of encapsulation chain of protocols
Color ENCAPSULATION = list PROTOCOL;
// Definition of security algorithm
color SECALGO = with DES | 3DES | HMAC | …;
// Definition of authentication elements
color AUTHN = product ATTRIBUTE*PROTOCOL*ATTRIBUTE
*PROTOCOL*SECALGO;
// Definition of the list of authentication elements
color AUTHNLIST = list AUTHN;
// Definition of confidentiality elements
color CONF = product ATTRIBUTE*PROTOCOL*SECALGO ;
// Definition of the list of confidentiality elements
color CONFLIST = list CONF;
// Definition of data flows
Color DATAFLOW = product ENCAPSULATION*AUTHNLIST
*CONFLIST;
```

Figure 7. Definition in CPN-ML of data flows

C. Scenarios

We choose three scenarios to validate our model. The first one is AH in the transport mode transformed data flow through NAPT to show the use of the authentication "AUTHN". The second one is ESP flow in the transport mode transformed data flow through NAPT to show the use of both authentication "AUTHN" and confidentiality "CONF". Finally, AH in the tunnel mode data flow through NAPT presents tunneling.

1) Scenario 1: AH flow in transport mode with NAPT

In this first scenario, we study the interaction between a mechanism that implements AH in the transport mode and the NAPT mechanism (Fig. 8). Our study consists in analyzing, for a given data flow $f = (< ip_1, tcp >, \{\}, \{\})$, the transformation chain $tf_{NAPT} \circ tf_{AH}^{transport}$.

Based on definition 3, data flow f is transformed to $f' = tf_{AH}^{transport}(f) = (< ip'_1, ah, tcp >, AUTHN, \{\})$ where:

- $attributes(ip'_1) = attributes(ip_1) \backslash \{(proto, 4)\} \cup \{(proto, 51)\}$,
- AUTHN = $\cup_{\forall x \in attributes(ip'_1) \backslash \{DSCP, ECN, \ldots, \text{checksum}\}} \{ (ad, ah, x, ip'_1, s) \} \cup$

$\cup_{\forall y \in attributes(ah)\backslash\{ad\}}\{(ad, ah, y, ah, s)\} \cup$
$\cup_{\forall z \in attributes(tcp)}\{(ad, ah, z, tcp, s)\}.$

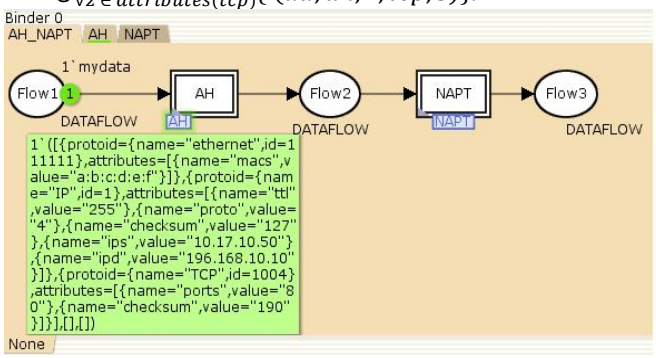

Figure 8. AH in the transport mode through NAPT

Then, data flow f' can't be transformed by *basic NAPT* (definition 7), because f' does not verify pre-condition 1; i.e. the protocol encapsulated directly by IP is AH. As a result, the token representing the data flow if blocked in place Flow2.

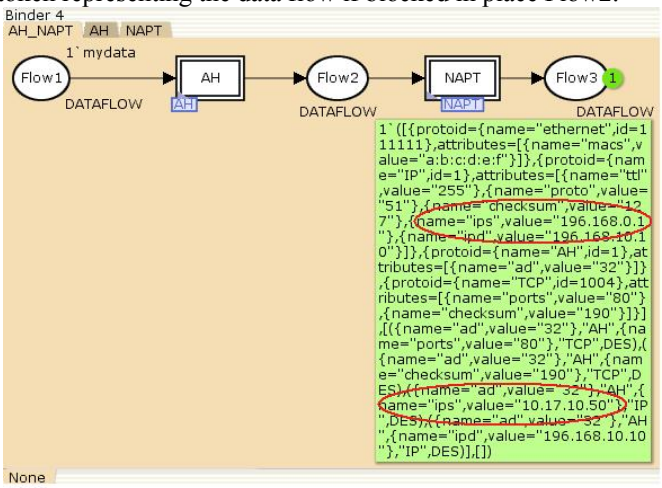

Figure 9. Conflict detection: AH in the transport mode through Advanced NAPT

While using the *advanced NAPT* (definition 8), we get the following data flow:

$f'' = tf^{adv}_{NAPT} \circ tf_{AH}(f) = (< ip'_1, ah, tcp' >, AUTHN, \{\})$, where:

- $attributes(ip'_1) = attributes(ip_1)\backslash\{(ips, old), (checksum, old)\} \cup \{(ips, new), (checksum, new)\}$,
- $attributes(tcp') = attributes(tcp)\{(ports, old), (checksum, old)\} \cup \{(ports, new), (checksum, new)\}$.

In this case, integrity of f'' (definition 2) is violated (Fig. 9) because:

- $(ad, ah, (ips, old), ip'_1, s) \in$ AUTHN and $(ips, old) \notin ip'_1$,
- $(ad, ah, (checksum, old), ip'_1, s) \in$ AUTHN and $(checksum, old) \notin ip'_1$,
- $(ad, ah, (ports, old), tcp', s) \in$ AUTHN and $(ports, old) \notin tcp'$,
- $(ad, ah, (checksum, old), tcp', s) \in$ AUTHN and $(checksum, old) \notin tcp'$.

2) Scenario 2: a flow ESP in the transport mode with NAPT

In our second scenario, we study the interaction between a mechanism that implements ESP in the transport mode and the NAPT mechanism. Our study consists in analyzing, for a given data flow $f = (< ip_1, tcp >, \{\}, \{\})$, the transformation chain $tf_{NAPT} \circ tf_{ESP}(f)$.

Based on definition 5, data flow f is transformed to $f' = tf^{transport}_{ESP}(f) = (< ip'_1, esp, tcp >, \text{AUTHN}, \text{CONF})$ where:

- $attributes(ip'_1) = attributes(ip_1)\backslash\{(proto, 4)\} \cup \{(proto, 50)\}$,
- AUTHN = $\cup_{\forall x \in attributes(esp)\backslash\{ad\}}\{(ad, esp, x, esp, s_1)\} \cup \cup_{\forall y \in attributes(tcp)}\{(ad, esp, y, tcp, s_1)\}$,
- CONF = $\cup_{\forall x \in attributes(esp)\backslash\{spi,seq,ad\}}\{(x, esp, s_2)\} \cup \cup_{\forall y \in attributes(tcp)}\{(y, tcp, s_2)\}$.

Using transformation function *basic NAPT* (definition 7), there is a conflict because f' does not verify pre-condition 1; i.e. the protocol encapsulated directly after IP is ESP. As a consequence, the token representing the data flow is blocked in place Flow2. On the other hand, using the *advanced NAPT* (definition 8), pre-condition 2 is not satisfied because $(ports, tcp, s_2) \in$ CONF, i.e. the source port of TCP is encrypted and therefore incomprehensible for NAPT (Fig. 10).

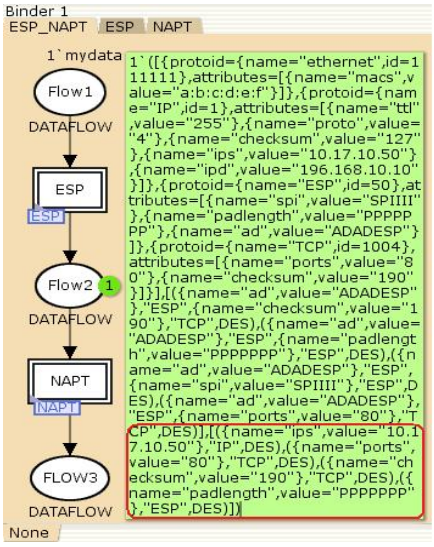

Figure 10. Conflict detection: ESP in the transport mode through NAPT

3) Scenario 3: a flow AH in tunnel mode with NAPT

In our third scenario, we study the interaction between a mechanism that implements AH in the tunnel mode and the NAPT mechanism. Our study analyzes, for a given data flow $f = (< ip_1, tcp >, \{\}, \{\})$, the transformation chain $tf_{NAPT} \circ tf^{tunnel}_{AH}(f)$.

Basing on definition 4, data flow $f' = tf^{tunnel}_{AH}(f) = (< ip_2, ah, ip_1, tcp >, AUTHN, \{\})$ where:

- $attributes(ip_2) =$

$\{(ips, sourcegateway), (ipd, destinationgateway), (proto, 51), \dots\}$,

- AUTHN =
  $\bigcup_{\forall x \in \text{attributes}(ip_2)\setminus\{\text{ttl,tos,flags,...}\}} \{(ad, ah, x, ip_2, s)\} \cup$
  $\bigcup_{\forall y \in attributes(ah)\setminus\{ad\}} \{(ad, ah, y, ah, s)\} \cup$
  $\bigcup_{\forall z \in attributes(ip_1)} \{(ad, ah, z, ip_1, s)\} \cup$
  $\bigcup_{\forall t \in attributes(tcp)} \{(ad, ah, t, tcp, s)\}$.

Data flow f' cannot be transformed by *basic NAPT* (definition 7), because f' does not verify pre-condition 1; i.e. the protocol encapsulated directly after IP is AH.

While using *advanced NAPT* (definition 8), we get the following data flow: $f'' = tf_{NAPT}^{adv} \circ tf_{AH}^{tunnel}(f) = (< ip_2', ah, ip_1, tcp' >, AUTHN, \{\})$, where:

- $attributes(ip_2') = attributes(ip_2) \setminus \{(ips, old), (checksum, old)\} \cup \{(ips, new), (checksum, new)\}$,
- $attributes(tcp') = attributes(tcp) \setminus \{(ports, old), (checksum, old)\} \cup \{(ports, new), (checksum, new)\}$.

In this case, integrity of f'' (definition 2) is violated (Fig. 11) because:

- $\left(ad, ah, (ips, old), ip_2^{'}, s\right) \in$ AUTHN and $(ips, old) \notin ip_2^{'}$,
- $\left(ad, ah, (checksum, old), ip_2^{'}, s\right) \in$ AUTHN and $(checksum, old) \notin ip_2^{'}$,
- $(ad, ah, (ports, old), tcp', s) \in$ AUTHN and $(ports, old) \notin tcp'$,
- $(ad, ah, (checksum, old), tcp', s) \in$ AUTHN and $(checksum, old) \notin tcp'$.

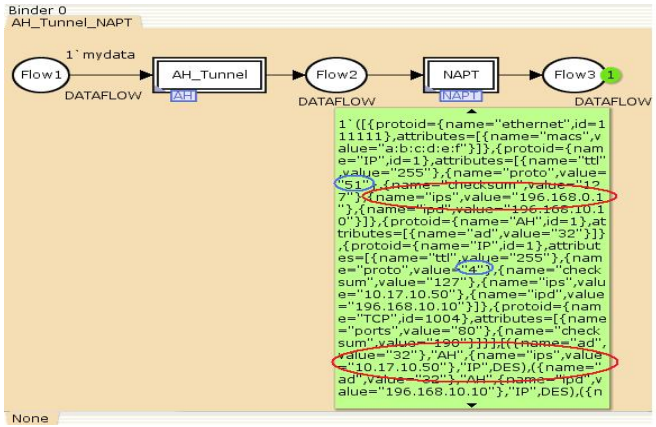

Figure 11. Conflict detection: AH in the tunnel mode through Advanced NAPT

VI. CONCLUSION

Network security requires the coordination of various heterogeneous and interdependent devices. As a consequence, multi-mechanisms and multi-levels conflicts may occur. The fast evolution of technologies imposes network security analysis to be independent from current technology. We proposed in this article a formal data flow oriented approach to analyze conflicts related to security policy deployment. This formal representation is based on an abstraction of a physical data flow consisting of blocks and the relation between each block and the underlying protocol.

By using CPN Tools, we have validated our approach based on well-known scenarios (IPsec and NAPT). This work allows us to verify the capacity of our model to express and analyze real distributed inconsistency. The results are encouraging because conflicts have been detected without requiring any a priori knowledge or experience. Indeed, we do not have to specify the semantics of the elements of a data flow. This leads us to believe that our approach can be used to detect unknown conflicts involving new security mechanisms that may occur at different levels.

Our future work will, therefore, focus on generalizing the model, especially on providing a generic model of devices' configurations built on our attribute-based approach. This will improve the reusability of hierarchical CPN specifications. Then, we will consider more realistic scenarios by looking at the future protocols available as draft specifications to prove our approach is adapted to new technologies too.

REFERENCES

[1] E. Al-Shaer and H. Hamed, "Discovery of Policy Anomalies in Distributed Firewalls", in IEEE INFOCOM, vol.4, pp. 2605-2616, 2004.

[2] J. G. Alfaro, N. B. Cuppens, F. Cuppens, "Complete analysis of configuration rules to guarantee reliable network security policies", in International Journal of Information Security,7(2), pp. 103-122, 2008.

[3] Z. Fu, F. Wu, H. Huang, K. Loh, F. Gong, I. Baldine, C. Xu, "IPSec/VPN Security Policy: Correctness, Conflict Detection and Resolution", in IEEE POLICY, pp. 39-56, 2001.

[4] J. D. Guttman and A. M. Herzog, "Rigorous automated network security management", in International Journal of Information Security, 4(3), pp. 29-48, 2005.

[5] H. Hamed and E. Al-Shaer, "Taxonomy of Conflicts in Network Security Policies", in IEEE Communications Magazine, 44(3), pp. 134-141, 2006.

[6] R. Laborde, M. Kamel, F. Barrère, and A. Benzekri, "Implementation of a Formal Security Policy Refinement Process in WBEM Architecture", in Journal of Network and Systems Management, 15(2), pp. 241-266, 2007.

[7] S. Preda, "Reliable context aware security policy deployment with applications to IPv6 environments", PhD thesis, Télécom Bretagne, 2010.

[8] P. Srisuresh and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", IETF RFC 3022, 2001.

[9] B. Aboba and W. Dixon, "IPsec-Network Address Translation (NAT) Compatibility Requirements", IETF RFC 3715, 2004.

[10] S. Kent and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, IETF, 2005.

[11] S. Kent, "IP Authentication Header (AH)", IETF RFC 4302, 2005.

[12] S. Kent , "IP Encapsulating Security Payload (ESP)", IETF RFC 4303, 2005.

[13] K. Jensen and L. M. Kristensen, "Coloured Perti Nets: Modelling and Validation of Concurrent Systems", Springer, 2009.

[14] http://cpntools.org/, <retreived: 03, 2012>

[15] Al-Shaer, E., Marrero, W., El-Atawy, A., ElBadawi, K., "Network configuration in a box: towards end-to-end verification of network reachability and security", in IEEE ICNP, pp. 123-132, 2009.

[16] I. El Khoury, R. Laborde, F. Barrère, A. Benzekri, "Towards a Formal Data Flow Oriented Model for Network Security Policies Analysis", in SAR-SSI, pp. 1-7, 2011.

[17] I. El Khoury, R. Laborde, F. Barrère, A. Benzekri, C. Maroun, "A generic data flow security model", in SAFECONFIG, pp. 1-2, 2011.

 ISBN: 978-1-61208-186-1