Symbolic analysis of network security policies using rewrite systems

Tony Bourdier, Horatiu Cirstea

Symbolic analysis of network security policies using rewrite systems

Tony Bourdier
INRIA Grand Est – LORIA – Nancy University
615 rue du Jardin Botanique
54600 Villers-lès-Nancy (France)
tony.bourdier@loria.fr

Horatiu Cirstea
INRIA Grand Est – LORIA – Nancy University
615 rue du Jardin Botanique
54600 Villers-lès-Nancy (France)
horatiu.cirstea@loria.fr

ABSTRACT

First designed to enable private networks to be opened up to the outside world in a secure way, the growing complexity of organizations make firewalls indispensable to control information flow within a company. The central role they hold in the security of the organization information make their management a critical task and that is why for years many works have focused on checking and analysing firewalls. The composition of firewalls, taking into account routing rules, has nevertheless often been neglected. In this paper, we propose to specify all components of a firewall, *i.e.* filtering and translation rules, as a rewrite system. We show that such specifications allow us to handle usual problems such as comparison, structural analysis and query analysis. We also propose a formal way to describe the composition of firewalls (including routing) in order to build a whole network security policy. The properties of the obtained rewrite system are strongly related to the properties of the specified networks and thus, classical theoretical and practical tools can be used to obtain relevant security properties of the security policies.

Categories and Subject Descriptors

D.4.6 [**Operating systems**]: Security and Protection; F.1.1 [**Computation by abstract devices**]: Models of Computation; F.4.2 [**Mathematical logic and formal languages**]: Grammars and Other Rewriting Systems

General Terms

Security, Theory, Verification

Keywords

Security policies, firewalls, rewrite systems, tree automata

1. INTRODUCTION

Security constitutes a crucial concern in modern information systems. Several aspects are involved, such as user authentication (establishing and verifying users' identity), cryptology (changing secrets into unintelligible messages and back to the original secrets after transmission) and security policies (preventing illicit or forbidden accesses from users to information).

Due to the increasing complexity of organizations, network security policies are rarely defined as a single firewall. Most of the time, they are made up of numerous firewalls whose composition depends on the network topology and on the routing rules. That is why it is often difficult to ensure that the composition of different local security policies (*i.e.* firewalls) expresses the intended security policy. Their formal specification is thus crucial and for several years now the importance of using formal methods to specify security policies has been generally accepted. For example, to achieve high levels of certification (EAL[1] 5, 6, 7), it is necessary to provide a formal specification enabling to obtain mechanized formal proofs, to carry out techniques for test generation, or to perform static analyses ensuring required properties.

Many methods and tools have been developed for analysing and testing firewall policies. These methods are broken down into two different categories: the active methods and the passive methods. The former consist in sending packets to the network and to make a diagnosis according to the received packets. The main advantage of these methods is that they require no abstract representation of firewalls and thus no error can be introduced between the specification and the implementation. However, such methods have the major drawback of consuming bandwidth, interfering with the traffic and being non exhaustive. That is why we focused on passive methods, that is methods which send no packet and make an offline analysis. Two main categories of passive analysis are investigated in the literature: structural analysis and query analysis. Structural analysis examines the relationships that rules have with other rules within a firewall configuration or across multiple firewalls. A misconfiguration (or conflict) occurs when several rules match the same packet or when a rule can be removed without changing the behavior of the firewall. Query analysis provides a way to ask questions of the form "Which computers in the private network can receive packets from 212.12.30.25 ?". It then consists in defining a language to describe a firewall query

[1]Evaluation Assurance Level

and a way to compute its solutions. Some works [13, 1, 6, 21, 12, 28, 2] deal with structural analysis and focus on defining, detecting and discussing misconfigurations while others [23, 17, 30] concentrate on query analysis. Some of these works abstract firewall filtering rules as one or two-dimensional ranges of IP, which does not allow to take completely advantage of the obtained results. Others assume that packets are not modified during their network traversal and then do not support network translation address capabilities. Moreover, they often focus on policies based on a single firewall or do not take into account the network topology. Some of the approaches can handle these various aspects (a detailed comparison between the different techniques could be found in [31]) but the routing aspects and especially the issues related to their combination with firewall policies are not deeply investigated. More generally, several rewrite based frameworks have already been proposed for specifying and analysing security policies. In particular, [26] introduces a narrowing based method for querying rule based policies and illustrates the proposed technique by the analysis of a standalone firewall policy. This approach can be seen as a first step toward a dedicated and more complete method for rewrite based analysis of firewall security policies.

In this paper, we introduce a new framework, based on rewrite systems and automata, for specifying and analysing firewall security policies. First, we show that this approach is particularly well adapted for the efficient verification of standalone firewall rules. The particular specification we propose here is not only natural since quite close to real world firewall rules but allows also an efficient implementation. We briefly explain how this approach can be used to perform the various types of analyses described in the literature. Second, we extend this approach to take into account the network topology of the network secured by firewall security policies. We show that there is a strong relationship between relevant properties of the secured network and the properties of the rewrite systems used to specify them. We focus here on completeness, *i.e.* the ability to take a decision for any (routed) packet, and consistency, *i.e.* the coherence of the final decisions independently of the routing protocol. We consider this approach particularly interesting since the correspondence with the properties of the rewrite systems used for the specifications allowed us to use theoretical and automatic tools to perform these analyses.

The paper is structured as follows. In Section 2 we present notions and notations we use throughout this paper. Section 3 is devoted to the presentation of the rewrite-based framework for specifying and analysing (standalone) firewalls. In Section 4 we address the composition of firewalls with respect to topologies and routing rules. Finally, the last section concludes with some perspectives for further work.

2. TECHNICAL PRELIMINARIES

We suppose the reader is familiar with notions related to term algebra (terms, substitutions, positions, ...), rewriting systems (reduction relation, confluence, termination,...) [4] and tree automata [9]. In this section we recall some basic notions, present the notations we use throughout this paper and introduce the notion of constrained rewrite systems.

Term algebra. We consider in this paper many-sorted signatures of the form $(\mathcal{F}, \mathcal{S})$ consisting of a set of sorts $\mathcal{S}$ and a set of function symbols $\mathcal{F}$. Symbols of $\mathcal{F}$ are denoted by bold characters $\mathbf{f}, \mathbf{g}, \ldots$ and their profiles are denoted as follows $\mathbf{f} : \mathsf{s}_1 \times \ldots \times \mathsf{s}_n \twoheadrightarrow \mathsf{s}$ where $\mathsf{s}_1, \ldots, \mathsf{s}$ are sorts of $\mathcal{S}$ and n is the arity of $\mathbf{f}$. The set of terms of sort s built out of symbols from $\mathcal{F}$ and of sorted variables from a set $\mathcal{X}$ is denoted by $\mathcal{T}^{\mathsf{s}}_{\mathcal{X}}$ and the set of ground terms of sort s is denoted by $\mathcal{T}^{\mathsf{s}}$. For any $t \in \mathcal{T}_{\mathcal{X}} = \cup_{\mathsf{s}\in\mathcal{S}}\mathcal{T}^{\mathsf{s}}_{\mathcal{X}}$, $\mathcal{V}ar(t)$ denotes the variables occurring in t. If any variable of t occurs only once in t, then t is said linear. A position within t is a sequence ω of integers describing the path from the root of t (seen as a finite labeled tree) to the root of the subterm at that position, denoted by $t_{|\omega}$. We use ε for the empty sequence. $|\,\omega\,|$ is the length of the position. $\mathcal{P}os(t)$ denotes the set of positions of t. $t(\omega)$ is the symbol of t at position ω and $t\,[s]_\omega$ the term t with the subterm at position ω replaced by s. A substitution σ is a mapping from $\mathcal{X}$ to $\mathcal{T}_{\mathcal{X}}$ which is the identity except over a finite set of variables (its domain) and which is extended to an endomorphism of $\mathcal{T}_{\mathcal{X}}$. A substitution is said ground if all the variables of its domain are mapped to ground terms. A term t matches a term t' iff $\sigma(t') = t$ for some substitution σ. Two terms t and t' are unifiable iff $\sigma(t') = \sigma(t)$ from some substitution σ.

Tree automata. A *tree automaton* is a triple $A = (Q, Q_F, \Delta)$ where Q is a finite set of symbols called *states* disjoint from $\mathcal{F}$, $Q_F \subseteq Q$ is the set of *final states* and Δ is a finite set of transitions of the form $\mathbf{f}(q_1, \ldots, q_n) \to_\Delta q$ where $q_1, \ldots, q_n, q \in Q$ and n is the arity of $\mathbf{f}$. $\to_\Delta$ is extended to $\to^*_\Delta$ as follows: if $\forall i, t_i \to^*_\Delta q_i$ and $\mathbf{f}(q_1, \ldots, q_n) \to_\Delta q$, then $\mathbf{f}(t_1, \ldots, t_n) \to^*_\Delta q$. The language recognized by $A = (Q, Q_F, \Delta)$ is $\mathcal{L}(A) = \{t \in \mathcal{T} \mid \exists q \in Q_F, t \to^*_\Delta q\}$. A set (or a language) of terms recognized by a tree automaton is said *regular*. A relation R is regular if there exists an automaton recognizing $\{\tilde{t} \mid t \in R\}$ where for any $t = (t_1, \ldots, t_n)$ and $\omega \in \cup_i \mathcal{P}os(t_i)$, $\tilde{t}(\omega) = (t_1[\omega), \ldots, t_n[\omega))$ with $t_i[\omega) = t_i(\omega)$ if $\omega \in \mathcal{P}os(t_i)$ and the special symbol Λ otherwise. Boolean operations, Cartesian product, projection and cylindrification preserve regularity. We say that a set or a relation is effectively regular iff it is regular and we can compute an automaton which recognizes it.

Rewrite systems. A rewrite rule is a pair of terms $l \to r$. The terms l and r are respectively called the *left-hand side* and *right-hand side* of the rule. A rewrite system R is a finite set of rewrite rules. Any rewrite system R induces a binary relation over terms denoted by $\to_{\mathrm{R}}$ as follows: for any terms t, t', $t \to_{\mathrm{R}} t'$ if there exist a rule $l \to r$ of R, $\omega \in \mathcal{P}os(t)$ and a substitution σ such that $t_{|\omega} = \sigma(l)$ and $t' = t\,[\sigma(r)]_\omega$. A rewrite rule is linear iff its left-hand side and right-hand side are linear. A rewrite system is linear if all its rules are linear. A *growing* rewrite system (GRS) [24] is a linear rewrite system such that for every rule $l \to r$, if $l(\omega) = r(\omega') \in \mathcal{X}$ for some positions ω, ω', then $|\,\omega\,| \leq 1$.

An *ordered* rewrite system is a rewrite system in which rules are ordered. For an ordered rewrite system R, $\to_{\mathrm{R}}$ is defined as follows: for any terms t, t', $t \to_{\mathrm{R}} t'$ if there exists a rule $l \to r$ of R, $\omega \in \mathcal{P}os(t)$ and a substitution σ such that $t_{|\omega} = \sigma(l)$ and $t' = t\,[\sigma(r)]_\omega$ and such that there is no prior rule $l' \to r'$ such that $t_{|\omega'} = \sigma'(l')$ for some ω' and σ'.

A *constrained* rewrite system (CRS) is a rewrite system such

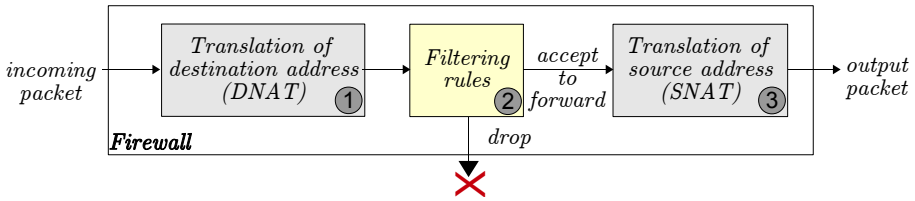


Figure 1: Firewall processing model

that for every rule $l \to r$ is associated to a set of membership constraints $x \in A$ where x is in $\mathcal{V}ar(l)$ and A is a regular tree language. If $l \to r$ is associated to $\{x_1 \in A_1, \ldots, x_n \in A_n\}$, we write $l \to r \ \| \ x_1 \in A_1, \ldots, x_n \in A_n$. The binary relation $\to_{\mathrm{R}}$ induced by a constrained rewrite system R is defined as follows: for any terms t, t', $t \to_{\mathrm{R}} t'$ iff there exists a rule $l \to r \ \| \ x_1 \in A_1, \ldots, x_n \in A_n$ of R, $\omega \in \mathcal{P}os(t)$ and a substitution σ such that $t_{|\omega} = \sigma(l)$ and $t' = t\,[\sigma(r)]_\omega$ and such that $\sigma(x_i) \in A_i$ for every i.

Given a rewrite system R, $\to_R^*$ denotes the reflexive transitive closure of the relation induced by R. For any term v, $\to_R^{-1}(v)$ denotes the set $\{u \mid u \to_R v\}$. For any set of ground terms $\mathcal{U} \subseteq \mathcal{T}$, $\to_R^{-1}(\mathcal{U})$ denotes the set $\{u \mid \exists v \in \mathcal{U}, u \to_R v\}$. A rewrite system R is *confluent* iff for any terms u, w, v, if $u \to_R^* v$ and $u \to_R^* w$, then there exists t such that $v \to_R^* t$ and $w \to_R^* t$. u is irreducible *w.r.t* R iff there is no v such that $u \to_R v$. If $u \to_R v$ and v is irreducible *w.r.t* R, then v is a *normal form* of u.

For any linear (constrained or not) rewrite system R and rule r of R, we denote by $rec(r)$ the regular set of ground terms that are reducible by r. If R is an ordered rewrite systems, we denote by $rec(r/R)$ the set of terms that are reducible by r and by no rule prior to r in R.

3. STANDALONE FIREWALL SPECIFICATION AND ANALYSIS

3.1 Short introduction to firewalls

In a network, when a host wants to transmit a message to another host, the data are encapsulated in a packet. Such a packet consists of the data that should be transmitted as well as of some additional information used to route it to the appropriate destination. The additional information, or header, mainly contains the packet's source and destination IP address, its protocol and the source and destination port. To control packet transmission between different subnetworks, it is common to deploy a network security policy based on a combination of firewalls. A firewall is an application that controls the forwarding of packets which cross it by using a combination of:

- *packet filtering*, which consists in inspecting each packet and either allowing it to continue its traversal or dropping it and
- *network address translation*, which consists in modifying network address information in packet headers.

Firewalls inspect incoming packets and accept or deny to forward them based on a list of decision rules which map the description of a set of packets to a decision. The most common criteria [10, 32] that firewalls use are the packet's source and destination address, its protocol, and, for TCP and UDP traffic, the port number. Moreover, firewalls often offer network address translation (NAT) functionality, which consists in rewriting the source (SNAT) or destination address (DNAT) into another address. The diagram in Figure 1 sums up the behavior of a firewall. At each step (1, 2 and 3), the packet is compared against a list of rules and the action (translation of destination address, drop or forward and translation of source address) corresponding to the first matched rule is performed.

EXAMPLE 1. *We give in Figure 2 a simple example of a firewall consisting of three rules: a filtering rule, a DNAT rule and an SNAT rule. We use the CIDR notation [19] to denote subnetworks*[2]*. According to the rules of this firewall, any packet of protocol tcp whose destination address is* 192.168.5.130:80 *and whose source address is* 192.168.20.1:80 *(notation address:port) is forwarded by the firewall as a packet whose source address is* 121.130.1.1:80 *and whose destination address is* 121.130.1.15:80 *whereas any packet whose destination address is* 121.130.1.30:80 *is dropped by the firewall.*

3.2 Rewrite-based specification of firewalls

In this section, we present a formalization of firewalls based on rewrite systems. The ports and IP addresses are specified as terms and the firewall rules are specified as rewrite rules. Such specifications can be easily and automatically obtained from firewall configurations defined using classical firewall configuration languages such as netfilter [32].

3.2.1 Packets and subnetworks

In our approach, packets are represented as algebraic terms. For readability reasons, we consider that firewalls inspect only addresses and ports. Other information, such as protocols, TCP flags, states, could be considered without difficulty. The selected symbolic representation of packets is based on the following signature:

$\mathbf{0}, \mathbf{1}$	:	Binary	$\to$ Binary
$\#$	:		$\to$ Binary
from	:	Binary $\times$ Binary	$\to$ SrcAddr
dest	:	Binary $\times$ Binary	$\to$ DstAddress
packet	:	SrcAddr $\times$ DstAddress	$\to$ Packet

We briefly describe in this section the meaning of the above symbols and defer until Section 3.3 the discussion about the consequences of our choices.

IPv4 as well as IPv6 addresses can be equivalently seen as sequences of bits, tuples of hexadecimal numbers or inte-

[2] $192.168.20.1/24 = [192.168.20.1, 192.168.20.255]$, $121.130.1.1/28 = [121.130.1.1, 121.130.1.15]$

Filtering:

IP address src	*IP address dest*	*Protocol*	*Port src*	*Port dst*	*Decision*
192.168.20.1/24	121.130.1.1/28	*tcp*	80	*any*	*Accept*
any	*any*	*any*	*any*	*any*	*Drop*

NAT:

Src/Dest	*Address range*	*Port range*	*New address : port*
Dest	192.168.5.128/25	*any*	121.130.1.15:80
Src	192.168.20.1/24	80	121.130.1.1:80

Figure 2: Example of a firewall

gers. We have thus numerous possibilities to describe addresses as terms. However, we must keep in mind that the packet inspection performed by firewalls strongly relies on checking whether an address belongs to some given address ranges which generally correspond to subnetwork[3] domains. A special feature of subnetwork domains is that all hosts it contains are addressed with a common, identical prefix in their IP address when this address is written as a bit vector. It is thus natural to specify subnetworks as bit sequences of variable length and to use pattern matching for testing if an address belongs to a subnetwork domain.

By representing addresses as words over $\{0,1\}$, or equivalently as terms built from monadic symbols **0** and **1** and a constant #, we obtain a representation of subnetworks by linear terms built from **0** and **1**. For example, the term $t =$ **11000000 10101000 00010100**(x) (we omit parentheses to keep readability) denotes the subnetwork 192.168.20.1/24 whereas **11000000 10101000 00010100 00000001**(#) denotes the IP address 192.168.20.1. For convenience, we will use in this paper the dot-decimal notation for addresses, the decimal notation for ports and the CIDR notation for subnetworks. When a variable occurs in the corresponding term, it will be indicated in brackets. For example t will be denoted by 192.168.20.1/24$[x]$. As we will see later on, this representation allows us to efficiently build tree automata recognizing addresses that belong to given ranges and consequently to efficiently analyse firewall behavior.

Finally, packets are terms of sort Packet. For example, the term $\mathbf{packet}\left(\begin{array}{l}\mathbf{from}(192.168.1.1, 80),\\ \mathbf{dest}(172.20.3.1, 80)\end{array}\right)$ represents a packet and $\mathbf{packet}\left(\begin{array}{l}\mathbf{from}(192.168.1.1/24[x], y),\\ \mathbf{dest}(172.20.3.1/24[x'], y')\end{array}\right)$ refers to the set of packets whose source address belongs to the subnetwork 192.168.1.1/24 and whose destination address belongs to 172.20.3.1/24.

As showed in Section 3.3, explicitly identifying the source and destination addresses (using the symbols **from** and **dest**) allows us to encode the firewall rules by appropriate tree automata that can be effectively used for firewall analysis.

3.2.2 *Firewall rules.*

To represent firewall rules, we add to the signature the following symbols:

$$\mathbf{accept}, \mathbf{drop} \quad : \quad \rightarrow \mathsf{Decision}$$

From a rewriting point of view, a filtering rule rewrites a packet into **accept** or **drop** whereas a NAT rule rewrites the source or destination address of a packet.

Definition 1 (Firewall). *A firewall* $\mathfrak{f}$ *is composed of three ordered rewrite systems* $\mathrm{Pre}_\mathfrak{f}, \mathrm{Filter}_\mathfrak{f}$ *and* $\mathrm{Post}_\mathfrak{f}$ *such that:*

- *rules of* $\mathrm{Filter}_\mathfrak{f}$ *are of the form* $p \to d$ *where* p *is a linear term of sort* Packet *and* d *a (ground) term of sort* Decision*;*
- *rules of* $\mathrm{Pre}_\mathfrak{f}$ *and* $\mathrm{Post}_\mathfrak{f}$ *are, respectively, of the form:*
$$\begin{array}{rcl}\mathbf{dest}(ip, port) & \to & \mathbf{dest}(ip', port')\\ \mathbf{from}(ip, port) & \to & \mathbf{from}(ip', port')\end{array}$$
where ip, $port$ *are linear terms and* ip', $port'$ *are ground terms.*

Example 2. *The firewall described in Example 1 can be specified as follows.* $\mathrm{Filter}_\mathfrak{f}$ *is the following ordered rewrite system:*

$$\left\{\begin{array}{l}\mathbf{packet}\left(\begin{array}{l}\mathbf{from}(192.168.20.1/24[x], 80)\\ \mathbf{dest}(121.130.1.1/28[y], z)\end{array}\right) \to \mathbf{accept}\\ \mathbf{packet}(x, y) \to \mathbf{drop}\end{array}\right.$$

while $\mathrm{Pre}_\mathfrak{f}$ *and* $\mathrm{Post}_\mathfrak{f}$ *consist, respectively, of the rules* $\mathbf{dest}(162.168.5.128/25[x], y) \to \mathbf{dest}(121.130.1.15, 80)$ *and* $\mathbf{from}(192.168.20.1/24[x], 80) \to \mathbf{from}(121.130.1.1, 80)$.

Definition 2 (Semantics). *For any firewall* $\mathfrak{f}$*, its semantics is denoted by* $[\![\mathfrak{f}]\!]$ *and defined as follows:*

$$[\![\mathfrak{f}]\!] = [\![\mathfrak{f}]\!]^{\mathrm{accept}} \cup [\![\mathfrak{f}]\!]^{\mathrm{drop}}$$

with[4]

$$[\![\mathfrak{f}]\!]^{\mathrm{accept}} = \{(t,u) \in \mathcal{T}^{\mathsf{Packet}} \times \mathcal{T}^{\mathsf{Packet}} \mid \exists v \in \mathcal{T}^{\mathsf{Packet}}, t \to_{\mathrm{Pre}_\mathfrak{f};\{x\to x\}} v \to_{\mathrm{Filter}_\mathfrak{f}} \mathbf{accept} \wedge v \to_{\mathrm{Post}_\mathfrak{f};\{x\to x\}} u\}$$

$$[\![\mathfrak{f}]\!]^{\mathrm{drop}} = \{(t, \mathbf{drop}) \in \mathcal{T}^{\mathsf{Packet}} \times \mathcal{T}^{\mathsf{Decision}} \mid \exists v \in \mathcal{T}^{\mathsf{Packet}}, t \to_{\mathrm{Pre}_\mathfrak{f};\{x\to x\}} u \to_{\mathrm{Filter}_\mathfrak{f}} \mathbf{drop}\}$$

From an abstract point of view, a firewall can be seen as a partial or total function which takes as input a packet and returns either another packet (possibly the same) or **drop**.

3.3 Analysis of firewalls

In this section, we show that this rewrite based specification allows one not only to automatically check properties concerning the semantics of a firewall but also to perform structural and query analysis over firewalls.

[3]A subnetwork is a logically visible subdivision of a network characterized by an IP ranges (its domain).

[4]R; $\{x \to x\}$ is the rewrite system R in which the rule $x \to x$ has been added as the last rule.

3.3.1 Firewall semantics analysis

A firewall can be seen as a decision process which associates to an incoming packet a decision which could be either drop or another packet, and thus the following properties should be verified: *consistency*, which indicates that at most one decision is taken for a given incoming packet, *termination*, which ensures that a firewall computes a decision in a finite time and *completeness*, which means that for any incoming packet, the firewall returns a decision.

As we have already said, by construction, any firewall denotes a terminating and consistent decision process and thus a function. The completeness can be thus defined as follows:

DEFINITION 3 (COMPLETENESS). *We say that a firewall* $\mathfrak{f}$ *is* complete *iff* $[\![\mathfrak{f}]\!]$ *is a total function.*

The particular shape of the rules defining a firewall allows us to represent the semantics of a firewall as a regular relation and consequently to verify its completeness:

PROPOSITION 1. *Completeness is decidable.*

PROOF. The proof relies on the regularity of the relations involved in the definition of the semantics of a firewall. Indeed, since the left-hand sides of all the rewrite rules composing a firewall are linear and share no variable with their corresponding right-hand sides, we can easily show that $\rightarrow_{\mathrm{Pre}_\mathfrak{f}}$ and $\rightarrow_{\mathrm{Post}_\mathfrak{f}}$ are regular tree relations (some technical manipulations are needed to take into account the order). Since the identity is a regular relation, it follows that $\rightarrow_{\mathrm{Pre}_\mathfrak{f};\{x\rightarrow x\}}$ and $\rightarrow_{\mathrm{Post}_\mathfrak{f};\{x\rightarrow x\}}$ are also regular. By composition and restriction, we obtain that $[\![\mathfrak{f}]\!]^{\mathrm{accept}}$ and $[\![\mathfrak{f}]\!]^{\mathrm{drop}}$ are regular tree (functional) relations. Consequently, $[\![\mathfrak{f}]\!]$ is a regular tree (functional) relation. The completeness can be tested by checking that the first projection of $[\![\mathfrak{f}]\!]$ covers the (regular) set of all possible incoming packets. □

In the case of complete firewalls, it can be important to determine if a firewall is less or more permissive than another one. A more permissive firewall allows at least the same traffic as a less permissive one. Such an order is obviously not total.

DEFINITION 4 (ORDER). *We define a partial order over complete firewalls* $\preceq$ *as follows: for any* $\mathfrak{f}$ *and* $\mathfrak{f}'$, $\mathfrak{f} \preceq \mathfrak{f}'$ *(*$\mathfrak{f}'$ *is more permissive than* $\mathfrak{f}$*) iff* $[\![\mathfrak{f}]\!]^{\mathrm{accept}} \subseteq [\![\mathfrak{f}']\!]^{\mathrm{accept}}$. *We write* $\mathfrak{f} \approx \mathfrak{f}'$ *iff* $\mathfrak{f} \preceq \mathfrak{f}'$ *and* $\mathfrak{f}' \preceq \mathfrak{f}$.

A firewall $\mathfrak{f}'$ is thus more permissive than a firewall $\mathfrak{f}$ if it accepts all the packets $\mathfrak{f}$ accepts and if it the result of the address translation is the same for these packets. Note that $\mathfrak{f} \approx \mathfrak{f}'$ iff $[\![\mathfrak{f}]\!] = [\![\mathfrak{f}']\!]$.

For the same reasons as before, we can decide whether a firewall is more or less permissive than another one:

PROPOSITION 2. *The order relation* $\preceq$ *is decidable.*

PROOF. As we have already shown, for any firewall $[\![\mathfrak{f}]\!]$, $[\![\mathfrak{f}]\!]^{\mathrm{accept}}$ and $[\![\mathfrak{f}]\!]^{\mathrm{drop}}$ and $[\![\mathfrak{f}]\!]$ are regular relation. Consequently, the inclusion $[\![\mathfrak{f}]\!]^{\mathrm{accept}} \subseteq [\![\mathfrak{f}']\!]^{\mathrm{accept}}$ is decidable. □

Note that two firewalls may have the same semantics even if their rules are different. This is particularly interesting since it allows to simplify or optimize the rules of a firewall and check if the resulted firewall has the same semantics as before.

3.3.2 Structural analysis.

Structural analysis refers to the detection of so-called misconfigurations (or anomalies) in (the rules of) a firewall. More precisely, such misconfigurations are properties expressed as relationships between the rules of a firewall. A complete survey of misconfigurations can be found in [13, 22]. Examples of anomalies are shadowing (a rule leads to decisions contradictory to decisions of prior rules), redundancy (a rule can be removed without changing the filtering result), generalization (a rule matches a superset of the set of packets matched by a prior rule with a different decision),.... We should mention that although several approaches have been developed for the detection of the above misconfigurations, this kind of anomalies are often intentionally introduced by firewall administrators in order to obtain more compact or more efficient rule sets. Detecting them is still interesting since it can outline some potential problems.

We only discuss here our approach for detecting shadowing; the other anomalies can be treated in a similar way. Let us first recall the definition of the shadowing anomaly: we say that a firewall has *shadowing* iff it contains at least one filtering rule such that all packets it accepts (resp. drops) are dropped (resp. accepted) by a prior rule. In such a case, the concerned rule is said to be *shadowed*.

The detection of the shadowed rules, as well as of the other misconfigurations, is based on the regularity of the sets of terms associated to a given rule. More precisely, each rule r is associated to several sets: $rec(r)$, denoting the set of packets matching r; $rec(r/\mathrm{Filter}_\mathfrak{f})$, denoting the set of packets matching r that match no prior rule of $\mathrm{Filter}_\mathfrak{f}$ (*i.e.* $rec(r) \setminus \bigcup_{r'<r} rec(r')$) and $rec(r/\mathrm{Filter}_\mathfrak{f}[d])$ denoting the set of packets matching r that match no other rule of $\mathrm{Filter}_\mathfrak{f}$ associated to the decision d. Since the left-hand sides of the filtering rules are linear terms, all the sets $rec(r)$ are regular; the other sets are also regular since they can be built starting from $rec(r)$ and using operations which preserve regularity. Misconfigurations can then be detected using inclusion or emptiness tests. For example, to detect if a rule r is shadowed, it suffices to check the emptiness of $rec(r/\mathrm{Filter}_\mathfrak{f}[\mathbf{accept}])$ if the right-hand side of r is **drop** and the emptiness of $rec(r/\mathrm{Filter}_\mathfrak{f}[\mathbf{drop}])$ otherwise.

It is well-known that the complexity of the operations over tree automata is quite high in general. In our case, the complexity of the needed operations strongly depends on the representation of packets and in particular on the representation of addresses. The choice of describing addresses as words over $\{0, 1\}$ (or equivalently as terms built from the monadic symbols **0** and **1** and the constant #) was indeed made in order to obtain efficient implementations of the corresponding automata operations.

To simplify explanations, let us consider word automata; the correspondence with tree automata is straightforward. Due to the representation of address ranges, we are confronted

with n-prefix (or simply prefix) languages, *i.e.* regular languages of the form $\alpha_1.\{0,1\}^* \cup \ldots \cup \alpha_n.\{0,1\}^*$. A good property of the manipulated address ranges is that corresponding minimal and deterministic automata have no loop except at their unique final state which loops over itself for any word. The main advantages of n-prefix languages are the following:

- boolean operations preserve the prefix property,
- boolean operations can be performed in $O(n)$ (where n is the number of states of the biggest operand) over the minimal deterministic automata and
- the corresponding algorithms directly produce deterministic and minimal automata (and thus there is no need to perform any determinization).

As said before, the sets of addresses of a given subnetwork are 1-prefix. It follows that $rec(r)$, $rec(r/\text{Filter}_\text{f})$, ..., are prefix languages. Consequently, misconfigurations can be efficiently detected using our approach.

3.3.3 Query analysis

Another kind of analysis proposed by some of the firewall verification approaches [29, 30] is query analysis. This kind of analysis provides a way to assist firewall administrators in understanding the behavior of a firewall by computing the result of user-defined queries such as "Which hosts in the subnetwork 192.168.1.1/22 can receive packets from a host in the subnetwork 172.20.1.1/24 ?". We have previously shown that the semantics of a firewall is a regular relation. Thus, any query expressed as a first order formula built from:

- variables, ground terms or terms whose head is the symbol **packet** and whose subterms are variables or ground terms;
- membership constraints *w.r.t.* to one of the relations defined in Definition 2 and
- membership constraints *w.r.t.* to a linear term (which means being a ground instance of)

can be rewritten into a tree automaton recognizing the set of solutions of the query, that is values of free variables making the formula true.

4. NETWORK SECURITY POLICIES

A network security policy is generally deployed by using several firewalls. If each of the firewalls present in the network has the expected properties checked, for example, by performing the analyses presented in Section 3.3, this is not necessarily the case for their composition. Indeed, firewall composition introduces a new security element which could disturb the behavior of standalone firewalls after connecting them together: routing. Although routing rules can generate major security faults, these rules are often neglected in network security policy analyses. For example, they can lead to loops in the paths followed by packets and consequently to congestions and even to service denial. Moreover, the combination between the routing and the firewall rules can lead to packets handled differently depending on the route they follow and thus, to ambiguous network security policies.

Indeed, the effects induced by the interaction between the routers and the firewalls make the semantics of the global network security policy hard to understand, particularly in large networks with a complex topology. We propose in this section to go a step further in network security policies analysis and extend the approach proposed in the previous section to take into account the network topology and the routing rules.

4.1 Policy specifications

In order to analyse a network security policy, one should first specify the network topology. More precisely, one must specify the subnetworks, the location of the security hosts (*i.e.* the nodes in which firewalls, as logical entities, are deployed) and the connectivity between the subnetworks and the security hosts. A subnetwork can be seen as a logical unit consisting of a set of network hosts which can mutually communicate without going through a security host. It often corresponds to a particular section of an organization and it is usually represented by a symbolic name and by an IP address range (called domain). Security hosts are interconnection nodes in the network which can be connected to subnetworks (a security host connected to a subnetwork is its gateway[5]) and to other security hosts. The picture in Figure 3 gives an example of a topology. For readability reasons, we consider in this figure and in the subsequent related examples that IP addresses are built from only one octet (that is 8 bits) and that the domain of Internet is 129/2.

In order to formally specify such a topology we add to the signature given in Section 3.2 a sort Net representing subnetwork names and a sort SH representing security hosts names.

Definition 5 (Network Topology). *A network topology τ is given by:*

- *a finite set* $\mathbf{net_1}, \ldots, \mathbf{net_n} : \rightarrow \mathsf{Net}$ *of subnetwork names;*
- *a finite set* $\mathbf{s_1}, \ldots, \mathbf{s_m} : \rightarrow \mathsf{SH}$ *of security host names;*

together with

- *a total map* $\Delta : \mathcal{T}^{\mathsf{Net}} \rightarrow \mathcal{T}_{\mathcal{X}}^{\mathsf{Binary}}$ *which associates any subnetwork name* $\mathbf{net_i}$ *with a linear term of* $\mathcal{T}_{\mathcal{X}}^{\mathsf{Binary}}$ *called its domain;*
- *a total map* $\mathcal{GW} : \mathcal{T}^{\mathsf{Net}} \rightarrow \mathcal{T}^{\mathsf{SH}}$ *which associates any subnetwork name* $\mathbf{net_i}$ *with a security host called its gateway such that for any security host* $\mathbf{s}$ *and any distinct subnetwork names* $\mathbf{net_1}, \mathbf{net_2} \in \mathcal{GW}^{-1}(\mathbf{s})$*, the terms* $\Delta(\mathbf{net_1})$ *and* $\Delta(\mathbf{net_2})$ *are not unifiable (that is to say, address ranges of subnetworks connected to the same security host must be disjoint);*
- *a relation* $\mathcal{CON}$ *over* $\mathcal{T}^{\mathsf{SH}} \times \mathcal{T}^{\mathsf{SH}}$ *describing the connection between the security hosts.*

Example 3. *The topology depicted in Figure 3 is formally defined by:*

- $\mathbf{printers}, \mathbf{servers}, \mathbf{secretariat}, \mathbf{dpt\_info} : \rightarrow \mathsf{Net}$*;*

[5]Note that without loss of generality, we can consider that any subnetwork has only one gateway. If one wants to describe a topology in which a subnetwork is connected to several security hosts, it suffices to add an intermediate security host.

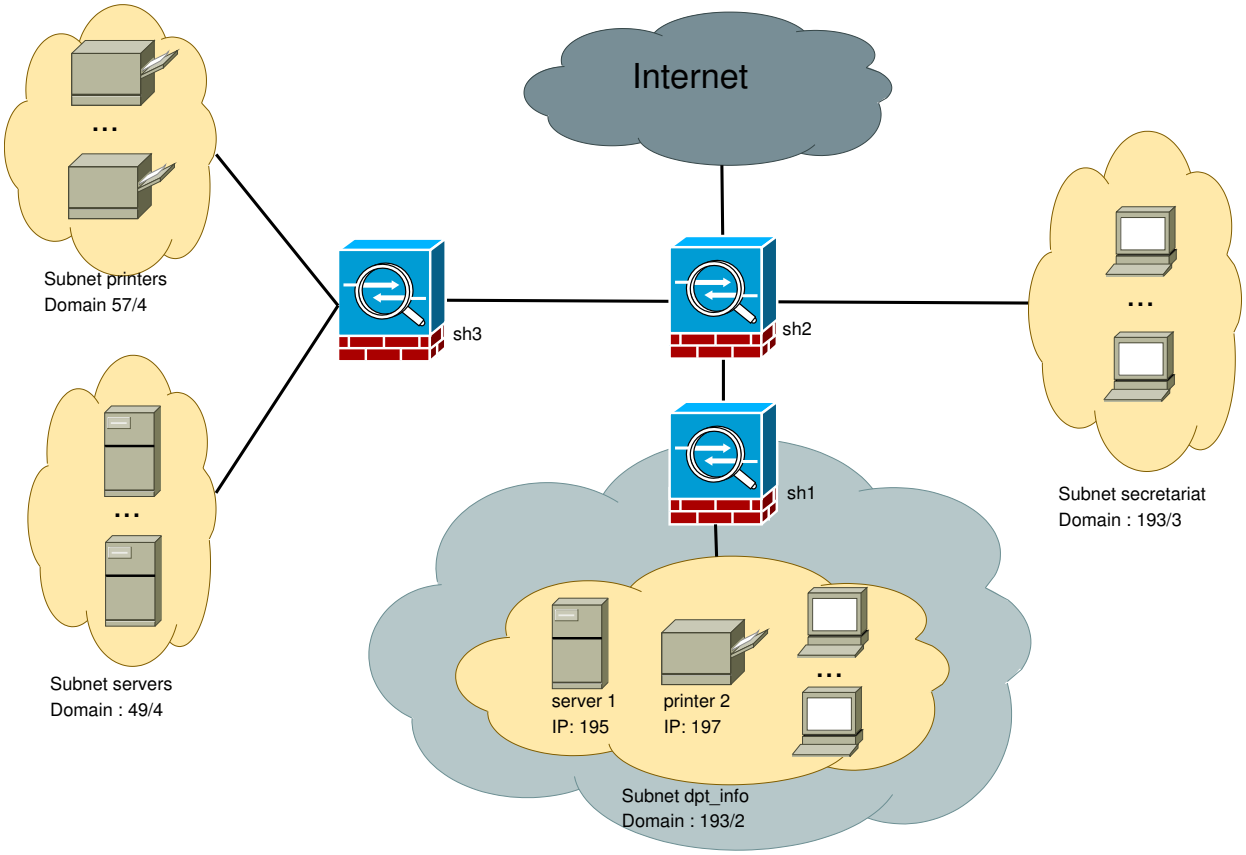


Figure 3: Example of network topology

- $\mathbf{sh_1}, \mathbf{sh_2}, \mathbf{sh_3} : \twoheadrightarrow \mathsf{SH}$;
- $\Delta = \left\{ \begin{array}{rcl} \textbf{printers} & \mapsto & 57/4[x] \\ \textbf{servers} & \mapsto & 49/4[x] \\ \textbf{secretariat} & \mapsto & 193/3[x] \\ \textbf{dpt\_info} & \mapsto & 193/2[x] \end{array} \right\}$;
- $\mathcal{GW} = \left\{ \begin{array}{rcl} \textbf{printers} & \mapsto & \mathbf{sh_3} \\ \textbf{servers} & \mapsto & \mathbf{sh_3} \\ \textbf{secretariat} & \mapsto & \mathbf{sh_2} \\ \textbf{dpt\_info} & \mapsto & \mathbf{sh_1} \end{array} \right\}$;
- $\mathcal{CON} = \{(\mathbf{sh_1}, \mathbf{sh_2}), (\mathbf{sh_2}, \mathbf{sh_1}), (\mathbf{sh_2}, \mathbf{sh_3}), (\mathbf{sh_3}, \mathbf{sh_2})\}$.

Given a network topology, defining a network security policy consists in defining for each security host a set of NAT and filtering rules (a firewall). The firewalls associated with the security hosts describe a set of "local" policies which are combined by the topology.

Definition 6 (Network security policy). *A network security policy $\wp$ over a network topology τ is a total mapping which associates any security host* **sh** *of τ with a firewall $\wp(\mathbf{sh})$.*

Example 4. *Let τ be the topology defined in Example 3. We define a network security policy $\wp$ over τ as follows:*

- *the aim of the security host* $\mathbf{sh_1}$ *is to hide the private IP address space of the subnetwork* **dpt_info** *and its machines except for the printer 2 and the server 1 which serves as a proxy for other machines of* **dpt_info**. *To achieve this goal,* $\mathbf{sh_1}$ *is associated to a firewall* $\wp(\mathbf{sh_1})$ *containing the NAT rules:*

$$\left\{ \begin{array}{rcl} \textbf{dest}(19, 515) & \rightarrow & \textbf{dest}(197, 515) \\ \textbf{dest}(17, 8080) & \rightarrow & \textbf{dest}(195, 32) \end{array} \right.$$
$$\left\{ \begin{array}{rcl} \textbf{from}(197, 515) & \rightarrow & \textbf{from}(19, 515) \\ \textbf{from}(195, 32) & \rightarrow & \textbf{from}(17, 8080) \end{array} \right.$$

and the following filtering rules:

$$\left\{ \begin{array}{rcl} \textbf{packet}(\textbf{from}(197, 515), x) & \rightarrow & \textbf{accept} \\ \textbf{packet}(\textbf{from}(195, 32), x) & \rightarrow & \textbf{accept} \\ \textbf{packet}(x, \textbf{dest}(197, 515)) & \rightarrow & \textbf{accept} \\ \textbf{packet}(x, \textbf{dest}(195, 32)) & \rightarrow & \textbf{accept} \\ \textbf{packet}(x, y) & \rightarrow & \textbf{drop} \end{array} \right.$$

The only machines accessible to the outside world are 197 (printer 2) masqueraded as 19 on the port 515 (the one associated to the Line Printer Daemon) and 195 (server 1) masqueraded as 17 on the port 32 masqueraded as 8080.

- $\mathbf{sh_2}$ *is associated to a firewall* $\wp(\mathbf{sh_2})$ *containing the following filtering rules:*

$$\left\{ \begin{array}{rcl} (i) \quad \textbf{packet}\left(\begin{array}{l} \textbf{from}(193/3[x], x') \\ \textbf{dest}(129/2[y], y') \end{array} \right) & \rightarrow & \textbf{drop} \\ (ii) \quad \textbf{packet}(\textbf{from}(129/2[x], x'), y) & \rightarrow & \textbf{drop} \\ (iii) \quad \textbf{packet}(x, y) & \rightarrow & \textbf{accept} \end{array} \right.$$

meaning that: (i) the hosts in the secretariat cannot initiate an Internet communication; (ii) no communication can be initiated by a host outside the organization (i.e. from Internet); (iii) all the other packets are

transmitted by the security host (as they are).

- *The third security host* $\mathbf{sh_3}$ *focuses on traffic sent from or received by the subnetworks* **printers** *and* **servers**. *It is associated to a firewall* $\mathfrak{f}_3 = \wp(\mathbf{sh_3})$ *containing the following filtering rules:*

$$\left\{\begin{array}{rrcl} (iv) & \mathbf{packet}\left(\begin{array}{l}\mathbf{from}(193/3[x], x')\\ \mathbf{dest}(57/4[x], x')\end{array}\right) & \rightarrow & \mathbf{accept}\\ (v) & \mathbf{packet}(\mathbf{from}(97/3[x], x'), y') & \rightarrow & \mathbf{accept}\\ (vi) & \mathbf{packet}(\mathbf{from}(17/4[x], x'), y') & \rightarrow & \mathbf{accept}\\ (vii) & \mathbf{packet}(x, y) & \rightarrow & \mathbf{drop}\end{array}\right.$$

meaning that (iv) *hosts from* **secretariat** *can reach the printers but not the servers;* (v) *the security host does not block traffic sent from* $97/3 = 57/4 \cup 49/4$ *(that is from subnetworks* **printers** *and* **servers**$)$*;* (vi) *packets from* **dpt_info** *(seen by* $\mathbf{sh_3}$ *as a subnetwork whose domain is* $17/4$*) are allowed and* (vii) *any other traffic is forbidden.*

To entirely characterize the network traffic under a given security policy, one must define a routing strategy to determine the paths that packets must follow to reach their destinations.

Definition 7 (Routing strategy). *Given a network topology* τ*, a routing strategy is a map* $\zeta : \mathcal{T}^{\mathsf{SH}} \twoheadrightarrow \mathcal{T}^{\mathsf{Binary}}_{\mathcal{X}} \twoheadrightarrow \mathcal{T}^{\mathsf{SH}}$ *such that:*

- (i) *for any* $(\mathbf{sh}, t_1, \mathbf{sh'_1})$ *and* $(\mathbf{sh}, t_2, \mathbf{sh'_2}) \in \zeta$*, there is no* σ *such that* $\sigma(t_1) = \sigma(t_2)$ *and*
- (ii) *for any* $(\mathbf{sh}, t, \mathbf{sh'}) \in \zeta$*, there are no* $\mathbf{net} \in \mathcal{GW}^{-1}(\mathbf{sh})$ *and substitution* σ *such that* $\sigma(\Delta(\mathbf{net})) = \sigma(t)$.

A routing strategy ζ associates to any security host a routing map which indicates the next security host to which a given packet must be forwarded depending on its destination. We impose (i) a deterministic routing, *i.e.* all security hosts route any incoming packet to at most another one security host and (ii) that packets whose destination is a directly reachable subnetwork must be delivered (and then must not be routed to another security host). Moreover, the aim of a routing strategy is to transmit any packet from (security) host to (security) host until it reaches one which is able to take a definitive decision (*i.e.* deliver the message to its final destination or drop it). According to this point of view, a routing strategy must satisfy certain conditions. First, it must create no loop when it transmits a request (packet). Second, it must eventually transmit any packet to a (security) host which is able to take a decision if one can be taken (if no security host can take a decision, the packet is lost or, in a network jargon, the packet is non-routable). Thus, we say that a routing strategy ζ is *sound w.r.t.* a topology τ if it satisfies the conditions mentioned above. More formally, ζ is sound *w.r.t.* $\tau = (\mathsf{Net}, \mathsf{SH}, \Delta, \mathcal{GW}, \mathcal{CON})$ iff for any $t \in \bigcup_{\mathbf{net} \in \mathcal{T}^{\mathsf{Net}}} rec(\Delta(\mathbf{net}))$ and $\mathbf{sh} \in \mathcal{T}^{\mathsf{SH}}$:

- there exist a finite sequence $(\mathbf{sh_i}, t_i, \mathbf{sh_{i+1}})_{i=1\ldots n}$ of tuples from ζ and a sequence of substitutions $(\sigma_i)_{i=1\ldots n+1}$ such that $\mathbf{sh_1} = \mathbf{sh}$ and for all $1 \leq i \leq n$: $(\mathbf{sh_i}, \mathbf{sh_{i+1}}) \in \mathcal{CON}$, $t = \sigma_i(t_i)$ and $t = \sigma_{n+1}(\Delta(\mathbf{net}))$ for some $\mathbf{net}$ such that $\mathcal{GW}(\mathbf{net}) = \mathbf{sh_{n+1}}$ and
- there exist no sequence $(\mathbf{sh_i}, t_i, \mathbf{sh_{i+1}})_{i=1\ldots n}$ and no $(\sigma_i)_{i=1\ldots n}$ such that $\mathbf{sh_1} = \mathbf{sh_{n+1}} = \mathbf{sh}$ and for all $1 \leq i \leq n$: $(\mathbf{sh_i}, \mathbf{sh_{i+1}}) \in \mathcal{CON}$ and $t = \sigma_i(t_i)$.

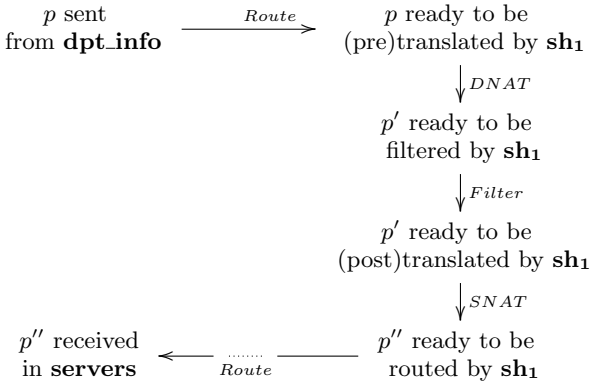
In what follows all routing strategies are considered to be sound *w.r.t.* the topology for which they are designed.

Example 5. *Following the previous example, we define the routing strategy below:*

- $\zeta(\mathbf{sh_1}) = \left\{\begin{array}{rcl} 125/1[x] & \mapsto & \mathbf{sh_2} \\ 32/3[x] & \mapsto & \mathbf{sh_2} \end{array}\right.; \left.\begin{array}{rcl} 64/2[x] & \mapsto & \mathbf{sh_2} \\ 1/4[x] & \mapsto & \mathbf{sh_2} \end{array}\right.$
  $\zeta(\mathbf{sh_3}) = \left\{\begin{array}{rcl} 128/1[x] & \mapsto & \mathbf{sh_2} \\ 64/3[x] & \mapsto & \mathbf{sh_2} \end{array}\right.; \left.\begin{array}{rcl} 1/2[x] & \mapsto & \mathbf{sh_2} \end{array}\right.$
  specifying that any packet sent to an address which is not directly reachable from $\mathbf{sh_1}$ *(resp.* $\mathbf{sh_3}$*) is routed to* $\mathbf{sh_2}$ *and*
- $\zeta(\mathbf{sh_2}) = \left\{\begin{array}{rcl} 97/3[x] & \mapsto & \mathbf{sh_3} \end{array}\right.; \left.\begin{array}{rcl} 17/4[x] & \mapsto & \mathbf{sh_1} \end{array}\right.$

The questions we are interested in are the same as for single firewalls. Which packets reach their final destination? Which ones are dropped? However, the analysis becomes more complicated in this case. Indeed, comparing to the case of a single firewall where the decision of accepting or dropping a packet is taken locally, in the case of a network topology, there is potentially an important number of intermediate steps between the moment a packet is sent and the moment it is received at its final destination (or dropped).

In other terms, a packet can be in different states: sent, received, about to be filtered by a given security host, ... The following figure represents an example of successive states in which a packet p can be:



To analyse the traffic under a network security policy, our approach consists in labeling packets to indicate their current state and describing their state evolution as a rewriting process. To represent states, we introduce the following symbols:

sent	: Net × SrcAddr × DestAddr	↠ State
received	: Net × SrcAddr × DestAddr	↠ State
pre	: SH × SrcAddr × DestAddr	↠ State
post	: SH × SrcAddr × DestAddr	↠ State
filter	: SH × SrcAddr × DestAddr	↠ State
route	: SH × SrcAddr × DestAddr	↠ State

To conform with the classical idea that a security policy evaluates access requests to decisions, we use, for any network topology, the following vocabulary:

- network access requests refer to ground terms of the

form $\mathbf{sent}(\mathbf{net}, \mathbf{from}(t_1, t_2), \mathbf{dest}(t_3, t_4))$ for some $\mathbf{net} \in \mathcal{T}^{\mathsf{Net}}$ and ground terms t_1, t_2, t_3 and t_4 such that $t_1 = \sigma(t)$ for $t = \Delta(\mathbf{net})$ and some ground substitution σ and
- decisions refer to either **drop** or ground terms of head **received**.

We denote by $\mathcal{R}equest$ the set of network access requests and by $\mathcal{D}ecision$ the set of decisions.

As for single firewalls, any network security policy $\wp$ is associated to a function which takes as input a packet together with the subnetwork it is sent from and returns drop (if the packet is dropped during its transmission) or the corresponding delivered packet together with the recipient subnetwork.

Definition 8 (Semantics of a security policy *w.r.t.* a routing strategy). *Given a network security policy $\wp$ over a topology τ and a routing strategy ζ sound w.r.t. τ, we call* semantics *of $\wp$ w.r.t. ζ and we denote by $[\![\wp]\!]_\zeta$ the (partial or total) function associating any request $r \in \mathcal{R}equest$ with a decision $d \in \mathcal{D}ecision$, when it exists, such that $r \xrightarrow{*}_{\mathrm{Flow}_{\wp,\zeta}} d$ with $\mathrm{Flow}_{\wp,\zeta}$ the term rewrite system defined in Figure 4.*

The rewrite system $\mathrm{Flow}_{\wp,\zeta}$ can be built for any network security policy $\wp$ and routing strategy ζ and computes the successive states of a given packet during its traversal of the network. Note that, by construction and due to the conditions imposed on ζ, $\mathrm{Flow}_{\wp,\zeta}$ is deterministic in the sense that for any ground term t, there exists at most one t' such that $t \rightarrow_{\mathrm{Flow}_{\wp,\zeta}} t'$.

To make clear the role of each rule scheme, the figure is divided into five categories corresponding to the functionality the corresponding rules specify. The rule scheme (B) expresses that two hosts from the same subnetwork can communicate without going through a security host. (R_1) expresses that a packet which is sent from a given subnetwork *net* toward another subnetwork must pass through the gateway of *net*. (R_2) describes the packet forwarding from a security host to the one selected by the routing rules. (R_3) indicates that if no routing rule applies and if the packet destination belongs to a subnetwork connected to the current security host, then the packet must be delivered to its recipient. $(DNAT_1)$ and $(SNAT_1)$ describe the address translation process when the packet matches a NAT rule whereas $(DNAT_2)$ and $(SNAT_2)$ express that a packet that does not match any NAT rule should only change its state. Finally, (F_{drop}) (resp. (F_{accept})) specifies that a packet is dropped or forwarded according to the filtering rules of the current security host.

4.2 Policy properties

In this section, we discuss some crucial policy properties and see how the rewrite-based encoding we proposed allows us to reason about these properties.

As mentioned previously, completeness is the ability to take a decision for every network access request. In the case of a single firewall, for verifying this property it is sufficient to check that the set of filtering rules covers all possible packets. However, the case of a network security policy is more complicated. Indeed, incompleteness is obtained due an incomplete firewall or due to the fact that a packet is never neither dropped nor delivered to its final destination (in the latter case, we say that the packet is lost).

Definition 9 (Completeness). *A network security policy $\wp$ over a network topology τ is* complete *w.r.t. a routing strategy ζ, or simply ζ-*complete*, if for any $r \in \mathcal{R}equest$, there exists $d \in \mathcal{D}ecision$ such that $[\![\wp]\!]_\zeta(r) = d$.*

Notice that a network security policy can be complete even if the firewalls it is made of are not complete and conversely, it can be incomplete even if all the firewalls it contains are complete.

Proposition 3. *For any network security policy $\wp$ and routing strategy ζ, the ζ-completeness of $\wp$ is decidable.*

More precisely, we have the following property:

Proposition 4. *Given a network security policy $\wp$ over a network topology τ and a routing strategy ζ, the sets:*

(i) $(\rightarrow^*_{\mathrm{Flow}_{\wp,\zeta}})^{-1}(\mathbf{drop})$ *and*
(ii) $(\rightarrow^*_{\mathrm{Flow}_{\wp,\zeta}})^{-1}(\mathcal{D}ecision \setminus \mathbf{drop})$

are effectively regular.

Proof. The proofs of Propositions 3 and 4 are based on the fact that $\mathrm{Flow}_{\wp,\zeta}$ is a constrained growing rewrite system. The method [24] used for computing a tree automaton which recognizes the set $(\rightarrow^*_{\mathrm{R}})^{-1}(L)$ for any regular tree language L and growing rewrite system R can be extended to constrained growing rewrite systems (the proof is given in Appendix A). Since $\{\mathbf{drop}\}$ and $\mathcal{D}ecision \setminus \{\mathbf{drop}\}$ are regular sets, we can build the tree automata which recognize $(\rightarrow^*_{\mathrm{Flow}_{\wp,\zeta}})^{-1}(\mathbf{drop})$ and $(\rightarrow^*_{\mathrm{Flow}_{\wp,\zeta}})^{-1}(\mathcal{D}ecision \setminus \mathbf{drop})$. The completeness of a security policy can be verified by checking that the union of these two automata covers $\mathcal{R}equest$. □

Even if a routing strategy is sound, the presence of address translation rules may cause routing loops. Indeed, some packets can be caught into a loop involving two or more NAT steps which are contradictory. For example, a security host s may translate a packet p into p' and route it to s' which may translate p' into p before routing it to s. Routing loops may significantly impact the quality of traffic and may even create a denial of service. We say that a routing strategy is *safe* under a security policy if the policy creates no routing loop. It is easy to see that the size of rewritten terms is not increased during the rewriting process by $\mathrm{Flow}_{\wp,\zeta}$ and thus only loops can lead to the non-termination of the rewrite system. Consequently, we obtain:

Proposition 5 (Safe routing). *Given a security policy $\wp$ over a topology τ together with a routing strategy ζ, ζ is* safe *under $\wp$ iff $\mathrm{Flow}_{\wp,\zeta}$ is terminating.*

We have considered so far only security policies *w.r.t.* static routing strategies. However, in practice, only small networks use manually configured routing. Most of the time networks have a too complex or a rapidly changing topology which

Broadcast

(B) $\mathbf{sent}(\mathbf{net}, x, y) \rightarrow \mathbf{received}(\mathbf{net}, x, y) \parallel y \in rec(\mathbf{dest}(\Delta(net), z))$
for any $\mathbf{net} \in \mathsf{Net}$

Route

(R_1) $\mathbf{sent}(\mathbf{net}, x, y) \rightarrow \mathbf{pre}(\mathbf{s}, x, y) \parallel y \in \overline{rec(\mathbf{dest}(\Delta(net), z))}$
for any $\mathbf{net} \in \mathsf{Net}$, $s = \mathcal{GW}(\mathbf{net})$

(R_2) $\mathbf{route}(\mathbf{s}, x, y) \rightarrow \mathbf{pre}(\mathbf{s}', x, y) \parallel y \in rec(\mathbf{dest}(t, z))$
for any $(s, t, s') \in \zeta$

(R_3) $\mathbf{route}(\mathbf{s}, x, y) \rightarrow \mathbf{received}(\mathbf{net}, x, y) \parallel y \in rec(\mathbf{dest}(\Delta(net), z))$
for any $s \in \mathsf{SH}$, $\mathbf{net} \in \mathsf{Net}$, *such that* $\mathcal{GW}(\mathbf{net}) = s$

Destination NAT

$(DNAT_1)$ $\mathbf{pre}(\mathbf{s}, x, y) \rightarrow \mathbf{filter}(\mathbf{s}, x, r) \parallel y \in rec(l \rightarrow r/\mathrm{Pre}_\mathsf{f})$
for any $s \in \mathsf{SH}$, $\mathsf{f} = \wp(\mathbf{s})$, $l \rightarrow r \in \mathrm{Pre}_\mathsf{f}$

$(DNAT_2)$ $\mathbf{pre}(\mathbf{s}, x, y) \rightarrow \mathbf{filter}(\mathbf{s}, x, y) \parallel y \in \overline{\bigcup_{r \in \mathrm{Pre}_\mathsf{f}} rec(r)}$
for any $s \in \mathsf{SH}$, $\mathsf{f} = \wp(\mathbf{s})$

Filter

(F_{drop}) $\mathbf{filter}(\mathbf{s}, x, y) \rightarrow \mathbf{drop} \parallel \mathbf{packet}(x, y) \in \bigcup_{l \rightarrow \mathbf{drop} \in \mathrm{Filter}_\mathsf{f}} rec(l \rightarrow \mathbf{drop}/\mathrm{Filter}_\mathsf{f})$
for any $s \in \mathsf{SH}$, $\mathsf{f} = \wp(\mathbf{s})$

(F_{accept}) $\mathbf{filter}(\mathbf{s}, x, y) \rightarrow \mathbf{post}(\mathbf{s}, x, y) \parallel \mathbf{packet}(x, y) \in \bigcup_{l \rightarrow \mathbf{accept} \in \mathrm{Filter}_\mathsf{f}} rec(l \rightarrow \mathbf{accept}/\mathrm{Filter}_\mathsf{f})$
for any $s \in \mathsf{SH}$, $\mathsf{f} = \wp(\mathbf{s})$

Source NAT

$(SNAT_1)$ $\mathbf{post}(\mathbf{s}, x, y) \rightarrow \mathbf{route}(\mathbf{s}, r, y) \parallel x \in rec(l \rightarrow r/\mathrm{Post}_\mathsf{f})$
for any $s \in \mathsf{SH}$, $\mathsf{f} = \wp(\mathbf{s})$, $l \rightarrow r \in \mathrm{Post}_\mathsf{f}$

$(SNAT_2)$ $\mathbf{post}(\mathbf{s}, x, y) \rightarrow \mathbf{route}(\mathbf{s}, x, y) \parallel x \in \overline{\bigcup_{r \in \mathrm{Post}_\mathsf{f}} rec(r)}$
for any $s \in \mathsf{SH}$, $\mathsf{f} = \wp(\mathbf{s})$

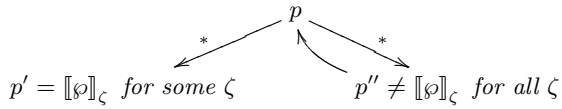
Figure 4: Rewrite system $\mathrm{Flow}_{\wp,\zeta}$ computing the traffic under a network security policy $\wp$ *w.r.t.* ζ

makes the definition of a static routing strategy unfeasible. Moreover, with the growing complexity of networks, the notions of performance and efficiency of the routing process appear and thus, the routes should be continuously calculated to take into account the situation of the traffic and to find the best paths, that is the shortest, the most reliable or those balancing best the network load. That is why adaptive routing, or dynamic routing, is widely used.

In this context, it is crucial that any packet has the same sort independently of the selected routing path it follows; it should be either delivered to the same recipient or dropped in all the cases. In other terms, routing should be a strategy for achieving a goal and should not alter the goal. We call consistency the property ensuring that the semantics of a security policy is the same for any (sound) routing strategy.

Definition 10 (Consistency). *A network security policy $\wp$ over a network topology τ is* consistent *iff for any routing strategies ζ and ζ' sound w.r.t. τ, $[\![\wp]\!]_\zeta = [\![\wp]\!]_{\zeta'}$.*

We express the consistency of a policy as a property over a new rewrite system. Given a network security policy $\wp$, we define $\mathrm{Flow}_\wp$ as the left-linear rewrite system describing all possible packet traversal scenarios *w.r.t.* any sound routing strategy, *i.e.* which rewrites any request r into $d \in \mathcal{D}ecision$ iff there exists a strategy ζ such that $[\![\wp]\!]_\zeta(r) = d$ and which rewrites it into a normal form which is not a decision if there exists no ζ such that $r \in \mathscr{D}([\![\wp]\!]_\zeta)$. The way the rewrite system $\mathrm{Flow}_\wp$ is build starting from the rules in Figure 4 is detailed in Appendix B. Roughly speaking, it chooses non-deterministically all the possible routes in order to reach all the possible decisions. If we have explored a path which does not lead to a decision, then the term backtracks:



To ensure the termination of the exploration precess, packets are labeled with the set of branches they have already explored. Thus, we obtain a rewrite systems which rewrites any request into all the possible decisions according to any sound routing strategies. More precisely, we obtain a system verifying the following proposition.

Proposition 6. *Let $\wp$ be a network security policy over a network topology τ. $\mathrm{Flow}_\wp$ is terminating iff any sound routing strategy is safe under $\wp$. In a such case, $\wp$ is consistent iff $\mathrm{Flow}_\wp$ is ground confluent.*

PROOF. One can easily show that $r \xrightarrow{*}_{\mathrm{Flow}_\wp} d \in \mathcal{D}ecision$ iff there is a strategy ζ such that $[\![\wp]\!]_\zeta(r) = d$. Starting from that, the equivalence between the consistency of $\wp$ and the ground confluence of $\mathrm{Flow}_\wp$ stands iff for any routing strategies ζ and ζ', $\mathscr{D}([\![\wp]\!]_\zeta) = \mathscr{D}([\![\wp]\!]_{\zeta'})$. Since we consider only sound strategies, when a request cannot be solved, only two cases can occur: either the request involves a non-routable address or the strategy is not safe. In the former case, the request cannot be solved for any sound strategy. Thus, if all strategies are safe, they have all the same domain. □

EXAMPLE 6. *Let us consider the policy described in Example 4. We have automatically prove (using A3PAT (CiME) [11] and AProVE [20]) that* $\mathrm{Flow}_\wp$ *is terminating and thus that any sound routing strategy is safe under* $\wp$. *We also attempted a paper proof for the consistency of* $\wp$ *but the property is not verified in this case. The following rewriting derivations prove that* $\wp$ *is not consistent.*

$$\begin{array}{c}
\mathbf{sent}(\mathbf{secretariat}, \mathbf{from}(\mathit{195}, \mathit{32}), \mathbf{dest}(\mathit{49}, \mathit{25})) \\
\downarrow^* \\
\mathbf{route}(\mathbf{sh_2} :: \#, \mathbf{from}(\mathit{195}, \mathit{32}), \mathbf{dest}(\mathit{49}, \mathit{25})) \\
\downarrow^* \\
\mathbf{route}(\mathbf{sh_3} :: \mathbf{sh_2} :: \#, \mathbf{from}(\mathit{195}, \mathit{32}), \mathbf{dest}(\mathit{49}, \mathit{25})) \\
\downarrow^* \\
\mathbf{drop} \\
\downarrow \\
\mathbf{pre}(\mathbf{sh_1} :: \mathbf{sh_2} :: \#, \mathbf{from}(\mathit{195}, \mathit{32}), \mathbf{dest}(\mathit{49}, \mathit{25})) \\
\downarrow^* \\
\mathbf{route}(\mathbf{sh_1} :: \#, \mathbf{from}(\mathit{17}, \mathit{8080}), \mathbf{dest}(\mathit{49}, \mathit{25})) \\
\downarrow^* \\
\mathbf{route}(\mathbf{sh_2} :: \mathbf{sh_1} :: \#, \mathbf{from}(\mathit{17}, \mathit{8080}), \mathbf{dest}(\mathit{49}, \mathit{25})) \\
\downarrow^* \\
\mathbf{route}(\mathbf{sh_3} :: \mathbf{sh_2} :: \mathbf{sh_1} :: \#, \mathbf{from}(\mathit{17}, \mathit{8080}), \mathbf{dest}(\mathit{49}, \mathit{25})) \\
\downarrow^* \\
\mathbf{received}(\mathbf{servers}, \mathbf{from}(\mathit{17}, \mathit{8080}), \mathbf{dest}(\mathit{49}, \mathit{25}))
\end{array}$$

It shows that, under some specific routing strategy, hosts from the **secretariat** *can exploit the server 1 to attack the subnetwork* **servers**.

We should mention that the specification of the network traffic as a rewrite system offers us two other possibilities non detailed in this paper. Indeed, semantic (dis)unification techniques [25] provide a way to perform query analyses similar to the ones briefly discussed for standalone firewalls and to detect covert channels by solving the following equation modulo $\mathrm{Flow}_\wp$:

$$\begin{array}{ll}
& \mathbf{sent}(n, \mathbf{from}(x, x'), \mathbf{dest}(y, y')) = \mathbf{drop} \\
\wedge & \mathbf{sent}(n, \mathbf{from}(x_1, x_2), \mathbf{dest}(y_1, y_2)) \\
& \qquad = \mathbf{received}(n', \mathbf{from}(x, x'), \mathbf{dest}(y, y'))
\end{array}$$

In order to use specific tools and techniques to check the properties of the rewrite system $\mathrm{Flow}_\wp$, one must represent it as an unconstrained rewrite system. For that, we use the fact that automata used in our constrained rewrite systems are all based on prefix-automata. Since for any prefix-based automaton A, we can compute a finite set of linear terms $\{t_1, \ldots, t_n\}$ such that $\cup_i rec(t_i) = \mathcal{L}(A)$ and (t_i, t_j) not unifiable for any $i \neq j$, we can transform any rule of our constrained rewrite systems into an equivalent finite set of rules which do not overlap. We obtain a finite left-linear rewrite system containing only trivial overlaps (between rules corresponding to the routing).

5. CONCLUSION

We have proposed in this paper an approach to describe firewalls using rewrite systems and automata. We have shown that this approach can be used to perform several kinds of analyses not only for standalone firewalls but also for routed networks of firewalls. In both cases we consider firewalls that can handle NAT rules.

We have shown that this rewrite based specification of firewalls is adapted for verifying usual properties related to the semantics of a firewall (such as completeness) and to perform firewall comparisons. We can also detect the so-called misconfigurations and perform query analyses. Moreover, all these analyses can be performed in the same formalisms and using the same tools.

The same rewrite based approach has been used to specify a network topology together with routing rules and firewall policies. The obtained rewrite system provides an executable specification of the network traffic under the security policy and, because of its form (left-linear, etc.), can be used to effectively perform completeness, reachability and consistency verifications. Some properties, like the consistency, are considered *w.r.t.* a routing policy or for any routing policy.

The modeling hypotheses assumed in this paper allow us to apply the obtained results in real cases. In particular, the automata operations needed here can be performed quite efficiently. Furthermore, one of the main advantages of using rewrite systems is the possibility to use automatic tools for verifying their properties. Indeed, termination of rewrite systems can be proved using A3PAT (CiME) [11], AProVE [20] or TTT [27] for example while confluence can be checked using ACP (Automated Confluence Prover) [3] and ground confluence with CrC Maude [14]. Moreover, reachability analysis can be performed using Autowrite [15, 16] (which implements the reachability problem for left-linear growing term rewrite system) or Timbuk [18] and efficient simulations of network traffics can be done using Tom [5], ELAN [7] or Maude [8].

We have already started the implementation of the approach presented here. The rewrite systems could be automatically generated starting from a given topology and a security policy and the preliminary results are quite promising. Once again, the particular shape of the rules we used allowed relatively quick automatic termination proofs (around 3s for 200 generated rules).

There are numerous perspectives to this work. Let us just mention that we plan to extend the approach in order to take into account statefull firewalls and to model proxy abilities. More precisely, to completely model TCP connections, we will consider sequences of packets instead of single packets. This should allow us to capture other vulnerabilities such as, for example, denial of services caused by unshared memories of firewalls. Moreover, since proxy abilities essentially con-

sist in rewriting sequences of packets, rewrite systems seem to be particularly suitable to model them.

6. REFERENCES

[1] T. Abbes, A. Bouhoula, and M. Rusinowitch. An inference system for detecting firewall filtering rules anomalies. In *ACM Symp. on Applied Computing*, pages 2122–2128. ACM, 2008.

[2] E. Al-Shaer, H. Hamed, R. Boutaba, and M. Hasan. Conflict classification and analysis of distributed firewall policies. In *IEEE Journal on Selected Areas in Communications*, volume 23, pages 2069 – 2084, 2005.

[3] T. Aoto, J. Yoshida, and Y. Toyama. Proving confluence of term rewriting systems automatically. In *Rewriting Techniques and Applications*, pages 93–102. Springer, 2009.

[4] F. Baader and T. Nipkow. *Term rewriting and all that*. C.U.Press, 1998.

[5] E. Balland, P. Brauner, R. Kopetz, P.-E. Moreau, and A. Reilles. Tom: Piggybacking rewriting on java. In *Rewriting Techniques and Applications*, LNCS, pages 36–47. Springer, 2007.

[6] A. Benelbahri and A. Bouhoula. Tuple based approach for anomalies detection within firewall filtering rules. In *IEEE Symp. on Computers and Communications*, pages 63–70. IEEE C.S., 2007.

[7] P. Borovanský, C. Kirchner, H. Kirchner, P.-E. Moreau, and C. Ringeissen. An overview of elan. *Electronic Notes in Th. Comp. Sci.*, 15:329–344, 1998.

[8] M. Clavel et al., editors. *All About Maude - A High-Performance Logical Framework*, volume 4350 of *LNCS*. Springer, 2007.

[9] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 2008.

[10] B. Conoboy and E. Fictner. Ip filter based firewalls howto. Available on: `http://www.obfuscation.org/ipf/ipf-howto.pdf`, 2002.

[11] É. Contejean, A. Paskevich, X. Urbain, P. Courtieu, O. Pons, and J. Forest. A3PAT, an approach for certified automated termination proofs. In *ACM SIGPLAN Work. on Partial evaluation and program manipulation*, pages 63–72. ACM, 2010.

[12] F. Cuppens, N. Cuppens-Boulahia, and J. Garcia-Alfaro. Detection and removal of firewall misconfiguration. In *Intl Conf. on Communication, Network and Information Security*. ACTA Press, 2005.

[13] F. Cuppens, N. Cuppens-Boulahia, and J. Garcia Alfaro. Detection of network security component misconfiguration by rewriting and correlation. In *Joint Conf. on Security in network ARchitectures and Security of Information Systems*, 2006.

[14] F. Durán and J. Meseguer. A Church-Rosser checker tool for conditional order-sorted equational Maude specifications. pages 69–85. Springer, 2010.

[15] I. Durand. Autowrite: A tool for term rewrite systems and tree automata. *Electronic Notes in Th. Comp. Sci.*, 124(2):29–49, 2005.

[16] I. Durand. Autowrite version 4, 2011. `http://dept-info.labri.u-bordeaux.fr/~idurand/autowrite/`.

[17] P. Eronen and J. Zitting. An expert system for analyzing firewall rules. In *Nordic Work. on Secure IT Systems*, pages 100–107, 2001.

[18] G. Feuillade, T. Genet, and V. Viet Triem Tong. Reachability analysis over term rewriting systems. 33(3):341–383, 2004.

[19] V. Fuller and T. Li. *Classless Inter-Domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan*. The Internet Society, 2006. RFC 4632.

[20] J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic termination proofs in the dependency pair framework. In *Intl Joint Conf. on Automated Reasoning*, pages 281–286. Springer, 2006.

[21] M. Gouda and A. Liu. Firewall design: Consistency, completeness, and compactness. In *IEEE Intl Conf. on Distributed Computing Systems*. IEEE C.S., 2004.

[22] H. Hamed and E. Al-Shaer. Taxonomy of conflicts in network security policies. *IEEE Communications Magazine*, 44(3):134–141, 2006.

[23] S. Hazelhurst. Algorithms for analysing firewall and router access lists. Technical Report TR-WITS-CS-1999-5, University of the Witwatersrand, South Africa, 2000.

[24] F. Jacquemard. Decidable approximations of term rewriting systems. In *Rewriting Techniques and Applications*, pages 362–376. Springer, 1996.

[25] J.-P. Jouannaud and C. Kirchner. Solving equations in abstract algebras: a rule-based survey of unification. In *Computational Logic: Essays in Honor of Alan Robinson*, chapter 8, pages 257–321. The MIT-Press, 1991.

[26] C. Kirchner, H. Kirchner, and A. de Oliveira. Analysis of rewrite-based access control policies. *Electronic Notes in Th. Comp. Sci.*, 234:55–75, 2009.

[27] M. Korp, C. Sternagel, H. Zankl, and A. Middeldorp. Tyrolean termination tool 2. In R. Treinen, editor, *Rewriting Techniques and Applications*, volume 5595 of *Lecture Notes in Computer Science*, pages 295–304. Springer Berlin / Heidelberg, 2009.

[28] A. Liu. Formal verification of firewall policies. In *IEEE Intl Conf. on Communications*, pages 1494 – 1498. IEEE C.S., 2008.

[29] A. Liu, M. Gouda, H. Ma, and A. Ngu. Firewall queries. *Principles of Distributed Systems*, pages 197–212, 2005.

[30] A. X. Liu and M. G. Gouda. Firewall policy queries. *IEEE Transactions on Parallel and Distributed Systems*, 20(6):766–777, 2009.

[31] T. Nelson, C. Barratt, D. Dougherty, K. Fisler, and S. Krishnamurthi. The margrave tool for firewall analysis. In *Proceedings of the 24th international conference on Large installation system administration*, pages 1–8. USENIX Association, 2010.

[32] R. Russell. Linux 2.4 packet filtering howto. Available on: `http://www.netfilter.org/documentation`,2002.

APPENDIX

A. EXTENSION OF JACQUEMARD'S RESULT TO CGRS

Without restriction, we can consider that any CGRS is a set of rules of one of the following forms

$$x \to r[x] \quad \| \quad x \in A \tag{1}$$

$$f(x_1, \ldots, x_n) \to r[x_1, \ldots, x_n] \quad \| \quad x_1 \in A_1, \ldots, x_n \in A_n \tag{2}$$

knowing that any unconstrained variable x can be seen as a variable constrained by $x \in A$ where A is the automaton recognizing all ground terms. Thus, we consider that any variable is constrained.

Let be L a regular language recognized by A_L and R a CGRS. The automaton recognizing $(\to_{\mathrm{R}}^*)^{-1}(L)$ is built as follows:

$$\mathcal{R}each_0 = (Q, Q_0, \Delta_0) := \biguplus_{(l \to r \| C) \in \mathrm{R}} \left(\biguplus_{(x \in A) \in C} A \right) \uplus A_L$$

where the disjoint sum ($\uplus$) of two automata over the same signature is the automaton whose set of states, set of final states and set of rules are the union of corresponding sets of the two automata, provided that they are all disjoint. Then, we transform Δ_k ($k \geq 0$) into Δ_{k+1} by applying the following rules:

$$(i) \ \frac{\begin{cases} \big(f(x_1, \ldots, x_n) \to g(r_1, \ldots, r_m) \| \bigwedge_i x_i \in A_i\big) \in \mathrm{R} \\ g(q_1, \ldots, q_m) \to q \in \Delta_k \end{cases}}{f(q'_1, \ldots, q'_n) \to q \in \Delta_{k+1}}$$

with the conditions:

1. for all $1 \leq i \leq n$, q'_i is a final state of A_i
2. for all $1 \leq j \leq m$, there exists a substitution $\theta : \mathcal{X} \to Q$ such that $\theta(r_j) \xrightarrow{*}_{\Delta_k} q_j$ and for each x_i occurring in $g(r_1, \ldots, r_m)$, we have $\theta(x_i) = q'_i$.

$$(ii) \ \frac{\big(f(x_1, \ldots, x_n) \to x \| \bigwedge_i x_i \in A_i\big) \in \mathrm{R} \ ; \ q \in Q}{f(q'_1, \ldots, q'_n) \to q \in \Delta_{k+1}}$$

with the conditions:

1. $x = x_i$ for some $1 \leq i \leq n$
2. for all $1 \leq i \leq n$, if $x_i = x$ then $q'_i = q$, otherwise q'_i is a final state of A_i.

The desired automaton is $\mathcal{R}each = (Q, Q_L, \Delta)$ where Q_L is the set of final states of A_L.

B. REWRITE SYSTEM ASSOCIATED TO A SECURITY POLICY

Given a network security policy $\wp$, we define $\mathrm{Flow}_\wp$ as the rewrite system built out from the following signature:

$\bot, \top$	:		$\twoheadrightarrow$ Bool
$\langle \_, \ldots, \_ \rangle_\_$	:	$\overbrace{\mathsf{Bool} \times \ldots \times \mathsf{Bool}}^{m \text{ times}} \times \mathsf{SH}$	$\twoheadrightarrow$ Context
::	:	Context × Trace	$\twoheadrightarrow$ Trace
#	:		$\twoheadrightarrow$ Trace
sent	:	Net × SrcAddr × DestAddr	$\twoheadrightarrow$ State
received	:	Net × SrcAddr × DestAddr	$\twoheadrightarrow$ State
pre	:	Trace × SrcAddr × DestAddr	$\twoheadrightarrow$ State
post	:	Trace × SrcAddr × DestAddr	$\twoheadrightarrow$ State
filter	:	Trace × SrcAddr × DestAddr	$\twoheadrightarrow$ State
route	:	Trace × SrcAddr × DestAddr	$\twoheadrightarrow$ State

where m is the number of security hosts. We modify the rewrite system presented in Figure 4 as follows (we only show modifications concerning the rewrite part of the rules - the constraints are unchanged):

(R_1) becomes $\mathbf{sent}(\mathbf{net}, x, y) \to \mathbf{pre}(\langle \bot, \ldots, \bot \rangle_{\mathbf{s}} :: \#, x, y)$;

(R_2) becomes $\mathbf{route}(\langle x_1, \ldots, \underbrace{\bot}_{\textit{index of } \mathbf{s}'}, \ldots, x_m \rangle_{\mathbf{s}} :: q, x, y) \to$

$$\mathbf{pre}(\langle x_1, \ldots, \underbrace{\top}_{\textit{index of } \mathbf{s}'}, \ldots, \underbrace{\top}_{\textit{index of } \mathbf{s}}, \ldots, x_m \rangle_{\mathbf{s}'} :: \langle x_1, \ldots, \underbrace{\top}_{\textit{index of } \mathbf{s}'}, \ldots, x_m \rangle_{\mathbf{s}} :: q, x, y)$$

(R_3) $\mathbf{route}(\langle x_1, \ldots, x_m \rangle_{\mathbf{s}} :: q, x, y) \to \mathbf{received}(\mathbf{net}, x, y)$;

($DNAT_1$) $\mathbf{pre}(\langle x_1, \ldots, x_m \rangle_{\mathbf{s}} :: q, x, y) \to \mathbf{filter}(\langle \bot, \ldots, \bot \rangle_{\mathbf{s}} :: \#, x, r)$

and ($SNAT_1$) is built in the same way. A new rule is added: (R_{back}) $\mathbf{route}(\langle \underbrace{x_1, \ldots, x_m}_{x_i = \top \textit{ for each } (\mathbf{s}, \mathbf{s_i}) \in \mathcal{CON}} \rangle_{\mathbf{s}} :: q, x, y) \to \mathbf{route}(q, x, y)$. Roughly speaking, the term $\langle t_1, \ldots, t_n \rangle_{\mathbf{s}}$ indicates that the packet is currently treated by the security host $\mathbf{s}$ and that the packet went through $\mathbf{s_i}$ iff $t_i = \top$. If the packet is translated, then the trace is reinitialized since from the network point of view this is another packet.