Tree automata based semantics of firewalls

Tony Bourdier

Tree automata based semantics of firewalls

Tony Bourdier
Inria Nancy & Université Henri Poincaré & Loria – Pareo Team
BP 101, 54602 Villers-lès-Nancy Cedex, France
Tel.: (+33)3.54.95.84.15
Tony.Bourdier@inria.fr

February 2011

Abstract

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). Firewalls are a core element of network security policies, that is why their analysis has drawn many attention over the past decade. In this paper, we propose a new approach for analyzing firewalls, based on tree automata techniques: we show that the semantics of any process composing a firewall (including the network address translation functionality) can be expressed as a regular set or relation and thus can be denoted by a tree automaton. We also investigate abilities opened by tree automata based representations of the semantics of firewalls.

Contents

1 Introduction and motivations

Since the late 80s, firewalls are at the heart of network security. First designed to enable private networks to be opened up to the outside in a secure way, the growing complexity of organizations make them indispensable to control information flow within a company. The central role of firewalls in the security of the organization information make their management a critical task. Moreover, it is admitted for some years the importance of using formal methods to specify security policies. 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.

Thus, for years, many methods and tools have been developed for analyzing and testing firewall policies. These methods can be 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 can be directly performed without any computation. However, such methods have the major drawback of consuming bandwidth, interfering with the traffic and being no exhaustive. That is why many works 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 examine the relationships that rules have with other rules within a firewall configuration or across multiple firewalls. These works consider that 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 `www.inria.fr`?". It then consists in defining a language to describe a firewall query and a way to compute its solutions. Some interesting work [AsH04, CCBGA06, ABR08, ASH03, BB07, GL04, CCBGA05, Liu08, ASHBH05] looked into structural analysis and others [Haz00, EZ01, LG09, MK05] looked into query analysis. Indeed, [AsH04, CCBGA06, ABR08, ASH03, BB07, GL04, CCBGA05, Liu08, ASHBH05] focus on defining, detecting and discussing misconfigurations. All of them assume that packets are not modified during their network traversal and then do not support network translation address capabilities. [Haz00, EZ01, LG09, MK05] use structures based on decision diagrams which provide a way to represent both rule sets of firewalls and solutions of some queries over firewalls. All these works are dedicated to a specific analysis.

Comparing to these works, our aim is to study a representation of the semantics of each component of a firewall based on tree automata and to build a decidable first order theory associated to the firewall. We show that issues raised by previously mentioned analyses are definable in this theory and thus obtain a generic procedure for performing these analyses.

Indeed, one of the main motivations of tree automata is the study of computational problems that can be solved using these machines. The usual approach consists in associating a logical system to a class of automata, which provides decision procedures for problems expressed as specifications in this logical system. Although

[1]Evaluation Assurance Level

one knows that decidable theories are not necessarily trivial from a computational point of view, we use here some classes of automata that enable operations over tree automata to be performed with low-complexity algorithms. Moreover, a significant work has been done during the last years in order to obtain new efficient algorithms. In particular, some libraries implementing efficient algorithms with efficient data structures have been recently developed [Len10]. A great advantage of our approach is the use of the well established algebraic frameworks of tree algebra and automata. Thus, our work takes full benefit from years of research on these domains. In particular, as we will sketch at the end of this paper, we can successfully apply from our work new techniques for model checking, called regular tree model checking, that have been developed to verify systems whose transition relation is described with a binary tree automaton [AJMd02, BHRV06]. Such techniques, following our approach, allow for example to find unwanted flows of packets from a given network security policy.

Roadmap. In Section 2, we recall basic definitions of terms, rewrite systems and tree automata. In Section 3, we show that the semantics of all firewall components are regular sets and relations. We investigate in Section 4 the possibilities opened by a tree automata based description of the semantics of firewalls. In particular, we define a first order theory in which all usual analyses are definable. We also discuss further abilities made possible by the use of tree automata. Finally, we give concluding remarks in Section 5.

2 Preliminaries

We assume that the reader is familiar with the standard notions of rewrite systems and tree automata. Comprehensive surveys can be found in [BN98] for first order terms and rewrite systems and in [CDG+08] for tree language theory. This section fixes our notations.

2.1 Term algebra and rewrite systems

A *signature* Σ consists of a finite set $\mathcal{S}_\Sigma$ whose elements are called *sorts* and an alphabet of symbols together with an application which associates to any symbol f a non empty sequence of sorts, which is denoted by $f : s_1 \times \ldots \times s_n \mapsto s$. $ar(f) = n$ is called the *arity* of f. Given a signature Σ, a sort $\kappa \in \mathcal{S}_\Sigma$ and a countable set $\mathcal{X}^s$ of *variables* for each sort s, we denote by $\mathcal{T}^{\kappa}_{\Sigma,\mathcal{X}}$ the set whose elements are called *terms sorted by* κ inductively defined as follows: for any $x \in \mathcal{X}^\kappa$, x is in $\mathcal{T}^{\kappa}_{\Sigma,\mathcal{X}}$ and for any $f : s_1 \times \ldots \times s_n \mapsto \kappa$ and $\langle t_1, \ldots, t_n \rangle \in \mathcal{T}^{s'_1}_{\Sigma,\mathcal{X}} \times \ldots \times \mathcal{T}^{s'_n}_{\Sigma,\mathcal{X}}$, with $s'_i \leq s_i$ for any i, the word $f(t_1, \ldots, t_n)$ is in $\mathcal{T}^{\kappa}_{\Sigma,\mathcal{X}}$. $\mathcal{T}_{\Sigma,\mathcal{X}}$ is the union of $\mathcal{T}^{s}_{\Sigma,\mathcal{X}}$ for every sort s. The set of variables occurring in $t \in \mathcal{T}_{\Sigma,\mathcal{X}}$ is denoted by $\mathcal{V}ar(t)$. If any variable of $\mathcal{V}ar(t)$ occurs only once in t, t is said to be *linear*. If $\mathcal{V}ar(t)$ is empty, t is called a ground term. $\mathcal{T}_\Sigma$ denotes the set of all ground terms. A *position* of a term t is a finite sequence of positive integers describing the path from the root of t to the root of the sub-term at that position. The empty sequence representing the root position is denoted by ε. $\mathcal{P}os(t)$ is called the set of positions of t. $t_{|\omega}$, resp. $t(\omega)$, denotes the subterm of t, resp. the symbol of t, at position ω. We denote by $t\,[s]_\omega$ the term

t with the subterm at position ω replaced by s. We call *substitution* any mapping from $\mathcal{X}$ to $\mathcal{T}_{\Sigma,\mathcal{X}}$ which is the identity except over a finite set of variables $\mathcal{D}om(\sigma)$ called *domain* of σ extended to an endomorphism of $\mathcal{T}_{\Sigma,\mathcal{X}}$. σ is often denoted by $\{x \mapsto \sigma(x) \mid x \in \mathcal{D}om(\sigma)\}$. If for any $x \in \mathcal{D}om(\sigma), \sigma(x) \in \mathcal{T}_{\Sigma}$, σ is said to be ground. For any ground substitution σ, $\sigma(t)$ is called a *ground instantiation* of t. A *rewrite rule* (over Σ) is a pair $(lhs, rhs) \in \mathcal{T}_{\Sigma,\mathcal{X}} \times \mathcal{T}_{\Sigma,\mathcal{X}}$ such that $\mathcal{V}ar(lhs) \subseteq \mathcal{V}ar(rhs)$ and a *rewrite system* is a set of rewrite rules $\mathcal{R}$ inducing a *rewriting relation* over $\mathcal{T}_{\Sigma}$, denoted by $\rightarrow_{\mathcal{R}}$ and such that $t \rightarrow_{\mathcal{R}} t'$ iff there exist $(l, r) \in \mathcal{R}$, $\omega \in \mathcal{P}os(t)$ and a ground substitution σ such that $t_{|\omega} = \sigma(l)$ and $t' = t\left[\sigma(r)\right]_{\omega}$. Finally, we denote by $\xrightarrow{*}_{\mathcal{R}}$ the reflexive transitive closure of $\rightarrow_{\mathcal{R}}$.

2.2 Tree automata

We call *n-ary tree automaton* any quadruple $\mathcal{A} = \langle\, \Sigma, Q, F, \Delta \,\rangle$ such that Σ is an alphabet of function symbols, Q is a finite set of *states*, F is a subset of Q whose elements are called *final states* and Δ is a relation over $\mathcal{T}_{\Sigma^n[Q]} \times Q$ whose elements are called *transitions* where Λ is a new symbol and $\Sigma^n\left[Q\right]$ consisting of the unique sort *conf* and the alphabet $(\Sigma \cup \{\Lambda\})^n \setminus \{\langle\, \Lambda, \ldots, \Lambda \,\rangle\} \cup Q$ such that $\langle\, f_1, \ldots, f_n \,\rangle \in (\Sigma \cup \{\Lambda\})^n$ is of sort $conf \times \ldots \times conf \mapsto conf$ with $ar(f_1, \ldots, f_n) = \max_{i \in [1,n]}(ar(f_i) \mid f_i \neq \Lambda)$ and any $q \in Q$ is a constant of sort $conf$. An element of $\mathcal{T}_{\Sigma^n[Q]}$ is called a *configuration*. A transition $lhs \rightarrow rhs$ of Δ is *normalized* iff for any $\omega \neq \varepsilon$, $lhs(\omega) \in Q$. An automaton whose transitions are normalized is said normalized. A tree automaton is said *deterministic* iff all its transitions have a different left-hand side. Without loss of generality, we can consider that all automata are normalized and deterministic. The rewriting relation induced by Δ over $\mathcal{T}_{\Sigma^n[Q]}$ is denoted by $\rightarrow_{\mathbb{A}}$ and the language recognized by $\mathbb{A}$ is $\mathcal{L}(\mathbb{A}) = \{\langle\, t_1, \ldots, t_n \,\rangle \in \mathcal{T}_{\Sigma} \mid \exists q_f \in F, t_1 \otimes \ldots \otimes t_n \xrightarrow{*}_{\mathbb{A}} q_f\}$ where $t = t_1 \otimes \ldots \otimes t_n$ is the configuration such that: $\forall \omega \in \bigcup_{i=1}^{n} \mathcal{P}os(t_i)$, $t(\omega) = \langle\, t_1[\omega], \ldots, t_n[\omega] \,\rangle$ where $u[\omega] = u(\omega)$ if $\omega \in \mathcal{P}os(t)$ and Λ otherwise. A set E of n-tuples of terms (or equivalently n-ary relation) is said *regular* iff there exists an n-ary tree automaton $\mathbb{A}$ such that $E = \mathcal{L}(\mathbb{A})$. Moreover, we say that a set (or a relation) is *effectively regular* iff it is regular and we can compute the automaton which recognizes it.

The table depicted in Figure 1 recalls usual automata and operations over tree automata together with their semantics. We recall that the membership, the emptiness, the finiteness, the equivalence and the inclusion problems are decidable for tree automata.

3 Firewall semantics

In this section, we propose a definition of firewall semantics using tree automata. First, we informally explain the behavior of firewalls. Next, we describe the language from which we will describe firewall semantics. Finally, we show in the last part of this section that we can automatically compute, from usual specifications of firewalls, tree automata describing the behavior of each of their components and how to combine them to obtain an automaton corresponding to a firewall.

Notation	Language recognized by the automaton
$\mathbb{A} \oplus \mathbb{A}'$	$\mathcal{L}(\mathbb{A}) \oplus \mathcal{L}(\mathbb{A}')$ where $\oplus$ is $\cap$, $\cup$, or $\times$
$\overline{\mathbb{A}}$	$(\mathcal{T}_\Sigma)^n \setminus \mathcal{L}(\mathbb{A})$
Ω_κ	$\mathcal{T}_\Sigma^\kappa$
$Id^n(\mathbb{A})$	n-tuples $\langle t, \ldots, t \rangle$ for $t \in \mathcal{L}(\mathbb{A})$
$rec(t)$	ground instantiations of t (t linear)
$\sqcup_i(\mathbb{A})$	$(n{+}1)$-tuples $\langle t_1, \ldots, t_{i-1}, t, t_i, \ldots, t_n \rangle$ *s.t.* $\langle t_1, \ldots, t_n \rangle \in \mathcal{L}(\mathbb{A})$
$\sqcap_i(\mathbb{A})$	$(n$-1$)$-tuples $\langle t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n \rangle$ *s.t.* $\exists t \in \mathcal{T}_\Sigma$: $\langle t_1, \ldots, t_{i-1}, t, t_{i+1}, \ldots, t_n \rangle \in \mathcal{L}(\mathbb{A})$
$\sqcap_{i/t}(\mathbb{A})$	$(n$-1$)$-tuples $\langle t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n \rangle$ *s.t.* $\langle t_1, \ldots, t_{i-1}, t, t_{i+1}, \ldots, t_n \rangle \in \mathcal{L}(\mathbb{A})$
$\partial^k_{\langle f_1, \ldots, f_n \rangle}(\mathbb{A})$	n-tuples $\langle t_1, \ldots, t_n \rangle$ *s.t.* $\exists \langle f_1(\ldots, x_1^{k-1}, t_1, x_1^{k+1}, \ldots), \ldots, f_n(\ldots, x_n^{k-1}, t_n, x_n^{k+1}, \ldots)\rangle \in \mathcal{L}(\mathbb{A})$

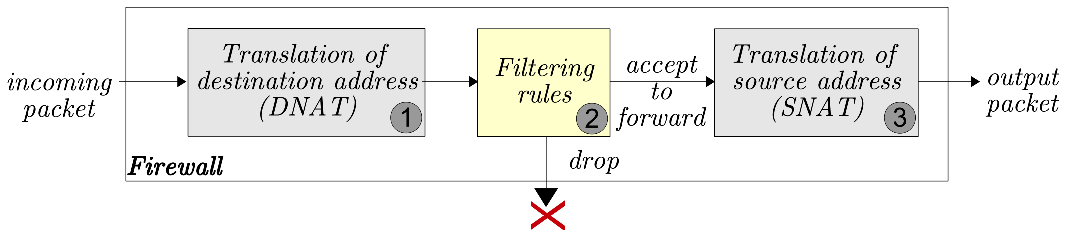
Figure 1: Operations effectively preserving regularity

3.1 Processing model

In a network, when a host wants to transmit a message to another host, data message are encapsulated in a packet. A packet consists of the data that should be transmitted as well as some additional information, called header, used to route it to the appropriate destination. To control packet transmission between different subnetworks[2], 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 upon a list of decision rules. These rules map the description of a set of packets to a decision. The most often used criteria [CF02, Rus02] 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 following diagram sums up the behavior of a firewall:


incoming packet
Translation of destination address (DNAT) 1
Filtering rules 2
accept to forward
Translation of source address (SNAT) 3
output packet
Firewall
drop

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

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. The following figure gives a simple example of firewall:

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	*any*	121.130.1.1

We use the CIDR notation [FL06] to denote subnetworks[3]. Any packet of protocol *tcp* whose source is 192.168.20.1:80 and destination 192.168.5.130:80 (notation address:port) is forwarded by the firewall as a packet whose source is 121.130.1.1:80 and whose destination is 121.130.1.15:80 whereas any packet whose destination is 121.130.1.30:80 is dropped by the firewall.

3.2 Vocabulary for formal reasoning

In order to give a formal semantics to each of firewall components, we need to define the vocabulary from which we will describe the objects of our study (IP, packets, ...). As the introduction of this paper lets it suppose, we base our work on a description of each entities which composes a firewall as a term. In other words, we use terms to represent all these entities. In what follows, we will talk about symbolic representation of these entities.

For readability reasons, we consider in what follows that packets are only described by addresses and ports. Other information, such as protocols, tcp flags, states, could be considered without difficulty. The selected symbolic representation of entities is based on the following signature:

$$\begin{array}{lrl}
0,1 & : \qquad \mathsf{Binary} & \to \mathsf{Binary} \\
\# & : \qquad & \to \mathsf{Binary} \\
ip & : \qquad \mathsf{Binary} & \to \mathsf{IP} \\
port & : \qquad \mathsf{Binary} & \to \mathsf{Port} \\
from & : \qquad \mathsf{IP} \times \mathsf{Port} & \to \mathsf{SrcAddress} \\
dest & : \qquad \mathsf{IP} \times \mathsf{Port} & \to \mathsf{DstAddress} \\
packet & : \mathsf{SrcAddress} \times \mathsf{DstAddress} & \to \mathsf{Packet}
\end{array}$$

As we consider in this paper only one signature, we will denote by sort the set of ground terms of sort sort (abuse of notation).

Let us describe the meaning of the above symbols. IP addresses are represented as terms of sort Binary describing the inverted binary representation of the address. For example, the IP address 192.168.1.1 is symbolically denoted by the following term $ip(1(0(0(0(0(0(0(0\ (1(0(0(0(0(0(0(0\ (0(0(0(1(0(1(0(1\ (0(0(0(0(0(0(1(1(\#)\cdots)$ (knowing that the integer 192 has 11000000 for binary representation and 168 has 10101000). We proceed in the same way for ports (with the symbol port instead of ip). Finally, packets are terms of sort Packet. For convenience, we will use in this paper the dot-decimal notation for addresses and the decimal notation for ports. For example:

$packet(from(ip(192.168.1.1), port(80)), dest(ip(172.20.3.1), port(80)))$ has to be understood as the term $packet(from(ip(t_s), port(t_p)), dest(ip(t_d), port(t_p)))$ where

$$\left\{\begin{array}{l} t_s = 1(0(0(0(0(0(0(0\ (1(0(0(0(0(0(0(0\ (0(0(0(1(0(1(0(1\ (0(0(0(0(0(0(1(1(\#)\cdots) \\ t_d = 1(0(0(0(0(0(0(0\ (1(1(0(0(0(0(0(0\ (0(0(1(0(1(0(0(0\ (0(0(1(1(0(1(0(1(\#)\cdots) \\ \text{and } t_p = 0(0(0(0(1(0(1(0(0(0(0(0(0(0(0(0(\#)\cdots) \end{array}\right.$$

Now, let us recall that a subnetwork is a logically visible subdivision of an IP network. Subnetworks are characterized by the partition of IP addresses into two parts: a "network prefix" and a "host number". More precisely, defining a subnetwork consists in giving a number $n < max$ of bits (where max is 32 for IPv4 and 128 for IPv6) together with a sequence of n bits (characterizing the network prefix). The remaining $max - n$ bits identify the host within the subnetwork. For example, the subnetwork (CIDR notation [FL06]) 192.168.5.64/26 corresponds to the range of IP addresses whose binary representation begins with the 26 first bits of the binary representation of 192.168.5.64. Note that for convenience, we suppose that any term of sort IP represents an IP address. A term of sort IP corresponds to an IPv4 address (resp. IPv6) iff it contains exactly 32 (resp. 128) symbols 0 or 1.

Proposition 2. The set of symbolic representations of IP addresses which belong to a given subnetwork is a regular set.

Proof. Let be a subnetwork characterized by a prefix $b_1, \dots, b_n$. The minimal deterministic automaton recognizing the set of IP addresses which belong to this subnetwork is given by:

$$\left\{\begin{array}{ccc} ip(q_n) & \rightarrow & q_F \\ 0(q_n) & \rightarrow & q_n \\ 1(q_n) & \rightarrow & q_n \end{array}\right\} \cup \{b_i(q_{i-1}) \rightarrow q_i \mid i = 1, \dots, n\} \cup \{\# \rightarrow q_0\}$$

Note that if we denote by $E[n]$ the subset of terms of E of length n, for any boolean operation $\oplus$, $E[n] \oplus F[n] = (E \oplus F)[n]$. Thus, there is no problem to consider automata which recognize IP addresses of an arbitrary length. $\square$

3.3 Tree automata based semantics

In what follows, we consider that a firewall $\mathbf{f}$ is given by three sets $\textsc{filter}(\mathbf{f})$, $\textsc{dnat}(\mathbf{f})$ and $\textsc{snat}(\mathbf{f})$ respectively containing filtering rules, prerouting (or DNAT) rules and postrouting (or SNAT) rules as well as an order relation $<_\mathbf{f}$ over these rules.

3.3.1 Filtering rules

Let us start by giving a semantics of filtering rules. If we denote by Packet the set of all packets, then any filtering rule $r \in \textsc{filter}(\mathbf{f})$ is associated to a subset of Packet which corresponds to the packets validating all conditions specified by r. For any packet $p \in \mathsf{Packet}$ and filtering rule r, we write $r \ll_{\textsc{filter}} p$ to indicate that the packet p matches the rule r.

Proposition 3. For any filtering rule r, the set $\ll_{\text{FILTER}} (r) = \{p \in \mathsf{Packet} \mid r \ll_{\text{FILTER}} p\}$ is effectively regular.

In addition, since filtering rules of firewalls are evaluated depending on the order $<_{\mathbf{f}}$ (only the first matched rule is applied), then the set of packets that effectively match a rule depends on the set of prior rules. For any firewall $\mathbf{f}$ and filtering rule r occurring in $\mathbf{f}$, we define the relation $\ll^{\mathbf{f}}_{\text{FILTER}}$ as follows: $r \ll^{\mathbf{f}}_{\text{FILTER}} p$ iff $r \ll_{\text{FILTER}} p$ and there is no prior rule $r' <_{\mathbf{f}} r$ in $\text{FILTER}(\mathbf{f})$ such that $r' \ll_{\text{FILTER}} p$.

Proposition 4. For any firewall $\mathbf{f}$ and filtering rule $r \in \text{FILTER}(\mathbf{f})$, the set $\ll^{\mathbf{f}}_{\text{FILTER}} (r) = \{p \in \mathsf{Packet} \mid r \ll^{\mathbf{f}}_{\text{FILTER}} p\}$ is effectively regular.

We saw that the goal of a filtering rule is to associate some set of packets with a decision (ACCEPT or DROP). Thus, we can see any filtering rule r as a partial function $\overset{r}{\longmapsto}_{\text{FILTER}}$ which associates to any packet p such that $r \ll_{\text{FILTER}} p$ either ACCEPT or DROP. Moreover, as previously, we define a partial function $\overset{r}{\longmapsto}{}^{\mathbf{f}}_{\text{FILTER}}$ which maps any packet p such that $r \ll^{\mathbf{f}}_{\text{FILTER}} p$ to the decision (ACCEPT or DROP) associated to r.

Proposition 5. For any firewall $\mathbf{f}$ and filtering rule $r \in \text{FILTER}(\mathbf{f})$, the partial functions $\overset{r}{\longmapsto}_{\text{FILTER}}$ and $\overset{r}{\longmapsto}{}^{\mathbf{f}}_{\text{FILTER}}$ are effectively regular (as relations).

Finally we can associate any firewall $\mathbf{f}$ with a function which maps any packet $p \in \mathsf{Packet}$ to a decision. This function is, by definition:

$$\longmapsto^{\mathbf{f}}_{\text{FILTER}} = \bigcup_{r \in \text{FILTER}(\mathbf{f})} \overset{r}{\longmapsto}{}^{\mathbf{f}}_{\text{FILTER}}$$

(where the functions $\overset{r}{\longmapsto}{}^{\mathbf{f}}_{\text{FILTER}}$ are seen as relations with disjoint domains). We obtain the following proposition:

Proposition 6. For any firewall $\mathbf{f}$, $\longmapsto^{\mathbf{f}}_{\text{FILTER}}$ is effectively regular.

3.3.2 Translation rules

The objective of the network address translation rules is to transform any packet into another packet (possibly identical). So, every rule r of $\text{DNAT}(\mathbf{f})$ can be considered as a partial function $\overset{r}{\longmapsto}_{\text{DNAT}}$ from Packet to Packet.

A similar reasoning leads us to define for every rule $r \in \text{DNAT}(\mathbf{f})$ the relations $r \ll_{\text{DNAT}} p$ and $r \ll^{\mathbf{f}}_{\text{DNAT}} p$ respectively meaning that p matches the rule r and that p matches r but does not match any prior rule $r' <_{\mathbf{f}} r$.

Proposition 7. For any DNAT rule r, $\overset{r}{\longmapsto}_{\text{DNAT}}$ is effectively regular.

Proof. Let r be a DNAT rule. r is characterized by a source subnetwork net together with a target IP address add as well as possibly a source port $sport$ together with a target port $tport$. If no port is specified, then the translation does not alter the destination port of the packet. Let us build the automaton recognizing the functional relation $\overset{r}{\longmapsto}_{\text{DNAT}}$. We denote by:

- $\mathbb{A}[net]$ the tree automaton which recognizes the set of IP addresses which belong to net;
- $\mathbb{A}[add]$ the automaton recognizing the IP address add;
- $\mathbb{A}[sport]$ and $\mathbb{A}[tport]$ the automata recognizing, if specified, $sport$ and $tport$.

We build the automaton $\mathbb{A}_{\text{DNAT}}[r]$ as the automaton containing the following rules:

- $\langle\, packet, packet\,\rangle\,(q_{fr}, q_{dst}) \to q_F$ (q_F is the unique final state);
- rules of the automaton $Id^2(\Omega_{\mathsf{SrcAddress}})$ whose final state is renamed into q_{fr};
- $\langle\, dest, dest\,\rangle\,(q_{ip}, q_{port}) \to q_{dst}$;
- rules of the automaton $Id^2(\Omega_{\mathsf{Port}})$ whose final state is renamed into q_{port} if $sport$ and $tport$ are not specified or rules of $\mathbb{A}[sport] \times \mathbb{A}[tport]$ whose final state is renamed into q_{port} otherwise;
- rules of the automaton $\mathbb{A}[net] \times \mathbb{A}[add]$ whose final state is renamed into q_{ip};

□

Like filtering rules, any DNAT rule r has a different semantics in the context of the set of DNAT rules of a firewall $\mathbf{f}$. We denote by $\overset{r}{\longmapsto}{}^{\mathbf{f}}_{\text{DNAT}}$ the corresponding semantics which also is a function from Packet to Packet.

Proposition 8. For any firewall $\mathbf{f}$ and filtering rule $r \in \text{DNAT}(\mathbf{f})$, the partial function $\overset{r}{\longmapsto}{}^{\mathbf{f}}_{\text{DNAT}}$ is effectively regular (as a relation).

When a packet matches no DNAT rule, then it is not translated by DNAT. In other terms, the semantics of the set of rules $\text{DNAT}(\mathbf{f})$ is a total function $\overset{r}{\longmapsto}_{\text{DNAT}}$ from Packet to Packet which is the identity except over the union of domains of $\overset{r}{\longmapsto}_{\text{DNAT}}$ for $r \in \text{DNAT}(\mathbf{f})$. Formally, we have:

$$\longmapsto^{\mathbf{f}}_{\text{DNAT}} = \left(\bigcup_{r \in \text{DNAT}(\mathbf{f})} \overset{r}{\longmapsto}{}^{\mathbf{f}}_{\text{DNAT}} \right) \cup id_{|\mathsf{Packet} \setminus \left(\bigcup_{r \in \text{DNAT}(\mathbf{f})} \ll_{\text{DNAT}}(r)\right)}$$

Proposition 9. For any firewall $\mathbf{f}$, $\longmapsto^{\mathbf{f}}_{\text{DNAT}}$ is effectively regular.

All the notations and results presented in this part are naturally extended to SNAT.

3.3.3 Firewall semantics

We saw in Section 3.1 that when a packet p goes through a firewall $\mathbf{f}$, the following steps occur:

- p is rewritten into p' by the DNAT rules,
- p' is either accepted or dropped,
- if p' is accepted, then p' is rewritten into p'' by the SNAT rules.

In other words, any firewall $\mathbf{f}$ describes a function which associates to a packet $p \in \mathsf{Packet}$ either another packet or DROP. We denote by $\xmapsto{\mathbf{f}}$ this function.

Proposition 10. For any firewall $\mathbf{f}$, $\xmapsto{\mathbf{f}}$ is effectively regular.

Proof. $\xmapsto{\mathbf{f}} = \mathcal{L}\left(\begin{array}{c} \sqcap_2 \left(\sqcup_3 \left(\mathbb{A}_{\text{DNAT}}\right) \cap \sqcup_{1,3}(\mathbb{A}^{\text{ACCEPT}}_{\text{FILTER}}) \cap \sqcup_1(\mathbb{A}_{\text{SNAT}})\right) \\ \cup \sqcap_2 \left(\mathbb{A}_{\text{DNAT}} \cap \sqcup_1(\mathbb{A}^{\text{DROP}}_{\text{FILTER}})\right) \times rec(\text{DROP}) \end{array}\right)$ where:

- $\mathcal{L}(\mathbb{A}_{\text{DNAT}}) = \xmapsto{\mathbf{f}}_{\text{DNAT}}$, $\mathcal{L}(\mathbb{A}_{\text{SNAT}}) = \xmapsto{\mathbf{f}}_{\text{SNAT}}$,
- $\mathcal{L}(\mathbb{A}^{\text{ACCEPT}}_{\text{FILTER}}) = (\xmapsto{\mathbf{f}}_{\text{FILTER}})^{-1}(\text{ACCEPT})$ and $\mathcal{L}(\mathbb{A}^{\text{DROP}}_{\text{FILTER}}) = (\xmapsto{\mathbf{f}}_{\text{FILTER}})^{-1}(\text{DROP})$.

$\square$

4 Applications

Our goal is to show that all usual analyses can be expressed as operations (which preserve regularity) over sets and relations that have previously been shown regular and that our approach allows us to extend the possibilities of analyses. In this section, we first study simple properties over a firewall (completeness and comparison). We next show that our approach allows us to deal with usual analyses (structural and query analyses). Finally, we discuss one further ability enabled by our approach.

4.1 Properties

If one sees a firewall as a decision process which associates to an incoming packet a decision which can be DROP or another packet, the following properties are relevant to analyze: *consistency*, which indicates that at most one decision is taken for a given incoming packet, *termination*, which assures that a firewall computes a decision in a finite time and *completeness*, which means that for any incoming packet, the firewall returns a decision. Another important issue is to compare different firewalls. By construction, every firewall denotes a terminating and consistent decision process. Then, we deal with completeness and comparison. We first give the formal definition of completeness and next define an order to compare firewalls.

Definition 11 (Completeness). We say that a firewall $\mathbf{f}$ is *complete* iff $\xmapsto{\mathbf{f}}$ is a total function.

Proposition 12. Completeness is decidable.

It is sufficient to check that the projection over the first component of the automaton recognizing $\xrightarrow{\mathbf{f}}$ equals to Ω_{Packet} (decidable).

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

Definition 13 (Order). We define a partial order over complete firewalls $\preceq$ as follows: for any $\mathbf{f}$ and $\mathbf{f}'$, $\mathbf{f} \preceq \mathbf{f}'$ ($\mathbf{f}'$ is more permissive than $\mathbf{f}$) iff for any $p, p' \in \mathsf{Packet}$, if $p \xmapsto{\mathbf{f}} p'$ then $p \xmapsto{\mathbf{f}'} p'$. We write $\mathbf{f} \approx \mathbf{f}'$ iff $\mathbf{f} \preceq \mathbf{f}'$ and $\mathbf{f}' \preceq \mathbf{f}$. Note that $\mathbf{f} \approx \mathbf{f}'$ iff $\xmapsto{\mathbf{f}} = \xmapsto{\mathbf{f}'}$.

Proposition 14. The order relation $\preceq$ is decidable.

Proof. It is sufficient to compute the automata recognizing the restriction of $\xmapsto{\mathbf{f}}$ and $\xmapsto{\mathbf{f}'}$ to packets which are not associated to DROP and to test the inclusion between them. $\square$

4.2 Structural analysis

Structural analysis refers to the detection of misconfigurations (or anomalies) in firewalls rules. A complete survey of misconfigurations can be found in [CCBGA06, HAS06]. Examples of anomalies are shadowing, redundancy, correlation, exception,... We have shown that the sets and relations of Figure 2 are regular and that we can compute tree automata which recognize them.

(i)	$\ll_{\text{FILTER}}$	(ii)	$\ll_{\text{SNAT}}$	(iii)	$\ll_{\text{DNAT}}$
(iv)	$\ll^{\mathbf{f}}_{\text{FILTER}}$	(v)	$\ll^{\mathbf{f}}_{\text{SNAT}}$	(vi)	$\ll^{\mathbf{f}}_{\text{DNAT}}$
(vii)	$\overset{\cdot}{\longmapsto}_{\text{FILTER}}$	$(viii)$	$\overset{\cdot}{\longmapsto}_{\text{SNAT}}$	(ix)	$\overset{\cdot}{\longmapsto}_{\text{DNAT}}$
(x)	$\overset{\cdot}{\longmapsto}{}^{\mathbf{f}}_{\text{FILTER}}$	(xi)	$\overset{\cdot}{\longmapsto}{}^{\mathbf{f}}_{\text{SNAT}}$	(xii)	$\overset{\cdot}{\longmapsto}{}^{\mathbf{f}}_{\text{DNAT}}$
$(xiii)$	$\longmapsto^{\mathbf{f}}_{\text{FILTER}}$	(xiv)	$\longmapsto^{\mathbf{f}}_{\text{SNAT}}$	(xv)	$\longmapsto^{\mathbf{f}}_{\text{DNAT}}$
(xvi)	$\xmapsto{\mathbf{f}}$				

Figure 2: Sets and relations of the theory of a firewall $\mathbf{f}$

It is well known that the set of solutions of any first order formula containing only variables and ground terms (or equivalently the computation of algorithms containing only operations depicted in Figure 1) over regular sets and relations are effectively regular, *i.e.* we can compute a tree automaton which recognizes the set of solutions.

Definition 15. For any firewall $\mathbf{f}$, we define $\mathcal{T}\!h_{\mathbf{f}}$ as the first order theory based on the following syntax:

- Constants: rules of $\mathbf{f}$, packets of Packet, decisions (ACCEPT and DROP)
- Predicate symbols: those of the Figure 2 and $<_\mathbf{f}$
- Quantifiers: $\exists$ and $\forall$
- Variables: any countable set
- Connectives: $\wedge, \vee, \Rightarrow, \neg$

Proposition 16. For any firewall $\mathbf{f}$, the theory $\mathcal{T}\!h_\mathbf{f}$ is decidable. Moreover, for any formula of $\mathcal{T}\!h_\mathbf{f}$, we can compute a tree automaton recognizing the set of solutions.

A possible algorithm to compute the tree automaton corresponding to a formula of $\mathcal{T}\!h_\mathbf{f}$ can be established following the approach described in [DT90]. In addition, we can easily show that any misconfiguration can be expressed as a formula of $\mathcal{T}\!h_\mathbf{f}$. It would be tedious to consider all anomalies, that is why we only discuss about an example of misconfigurations (shadowing). Let us first recall the definition of the shadowing anomaly.

Definition 17 (Shadowing). 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.*

Proposition 18. The shadowing property can be expressed as a formula of $\mathcal{T}\!h_\mathbf{f}$.

Indeed, we can define the formula $\mathit{Shadowed}_\mathbf{f}(r)$ as follows :

$$\begin{array}{rl} & \forall p\colon p \overset{r}{\longmapsto}{}^{\mathbf{f}}_{\text{FILTER}} \text{ACCEPT} \Rightarrow (\exists r'\colon r' <_\mathbf{f} r \wedge p \overset{r'}{\longmapsto}{}^{\mathbf{f}}_{\text{FILTER}} \text{DROP}) \\ \vee & \forall p\colon p \overset{r}{\longmapsto}{}^{\mathbf{f}}_{\text{FILTER}} \text{DROP} \Rightarrow (\exists r'\colon r' <_\mathbf{f} r \wedge p \overset{r'}{\longmapsto}{}^{\mathbf{f}}_{\text{FILTER}} \text{ACCEPT}) \end{array}$$

Thereby, the real question which arises from structural analysis in our case is to know if the selected symbolic representation of packets provides efficient algorithms to perform these operations. Indeed, it is well-known that operations over tree automata have high complexity in general. We can remark that in our case the complexity of operations strongly relies on the representation of addresses. Therefore, let us focus our discussion over addresses and their representation. We made the choice of describing addresses as words over $\{0,1\}$ (or equivalently as terms built from monadic symbols 0 and 1 and a constant #). To simplify explanations, we consider word automata (the correspondence with tree automata is straightforward). A good property of manipulated ranges of addresses is that corresponding minimal and deterministic automata has no loop except at their unique final state which loops over itself for any word. Let us call n-prefix (or simply prefix) language any regular language of the form $\alpha_1.\{0,1\}^* \cup \ldots \cup \alpha_n.\{0,1\}^*$. The main advantages of such 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 bigger operand) over the minimal deterministic automata and
- the corresponding algorithms directly produce deterministic and minimal automata (which avoid to perform any determinization).

Consequently, misconfigurations can be efficiently detected.

4.3 Query analysis

Another main issue addressed about firewalls is query analysis. Query 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 ?". Such analysis only recently emerged with [MK05, LG09]. To handle the class of queries presented for example in [LG09], it is sufficient to slightly extend the formulae of $\mathcal{Th}_{\mathbf{f}}$. The main feature handled in usual queries which cannot be computed by a formula of $\mathcal{Th}_{\mathbf{f}}$ is reasoning about the different fields of packets (source or destination IP, ...). Thus, we add to $\mathcal{Th}_{\mathbf{f}}$:

- linear terms of sort Packet whose variable are of sort IP or Port
- subnetworks (as constants)
- a binary predicate $\in$ over terms of sort Packet and subnetworks

This extension of $\mathcal{Th}_{\mathbf{f}}$ preserves the Proposition 16.

Example 19. Let us consider a firewall $\mathbf{f}$ between a private network and the internet together with the following query "Who can access to `www.inria.fr` from the private network 192.168.1.1/22" ? This query can be expressed as the following formula:

$$\begin{aligned}&ip_1 \in 192.168.1.1/22 \land \exists ip_2, ip_3, ip_4, p_2, p_3:\\ &packet(from(ip_1,p_1), dest(ip_2, 80)) \xmapsto{\mathbf{f}} packet(from(ip_3,p_2), dest(138.96.146.2, 80))\\ &\land\ packet(from(138.96.146.2, 80), dest(ip_3,p_2)) \xmapsto{\mathbf{f}} packet(from(ip_4,p_3), dest(ip_1,p_1))\end{aligned}$$

where 138.96.146.2 is the IP address of `www.inria.fr`. From this formula we can compute a tree automaton recognizing the set of pairs of IP addresses and ports $\langle\, ip_1, p_1 \,\rangle$ satisfying the query.

4.4 Further abilities

Let us mention a promising example (that we can not develop due to the lack of place): regular tree model checking. Regular tree model checking is a method for formal verification of infinite-state systems whose states are encoded as trees and whose transition relation is denoted by a tree transducer (transducers are binary tree automata recognizing relations over terms having the same set of positions). Given a transducer T, regular tree model checking consists in computing the transitive closure of $\rightarrow = \mathcal{L}(T)$ and then the set of reachable states (from a regular initial set of configurations) as tree automata ([ALdR05, BHRV06]).

Regular tree model checking can be successfully applied to network security policies analysis. Indeed, given a network composed by firewalls separating several subnetworks, we can define a transition system whose states are terms of sort Packet in which the head symbol has been replaced by a symbol which indicates the subnet in which the packet is located. To any firewall $\mathbf{f}$ in the network is associated a part of the transition relation corresponding to the semantics of $\mathbf{f}$ such that $net_1(t) \rightarrow_{\mathbf{f}} net_2(t')$ if $\mathbf{f}$ separates the networks net_1 and net_2 and $packet(t) \overset{\mathbf{f}}{\longmapsto} packet(t')$. A host h which belongs to the subnet net can receive the packet $packet(t)$ from another host h' located in net' if $net(t)$ is a reachable state (and t contains the subterm $dest(ip, port)$ where ip is the IP address of h) from the automaton recognizing $net'(t')$ (and t' contains the subterm $dest(ip', port')$ where ip' is the IP address of h'). Given such a transition system and knowing that the transition relation is recognized by a transducer if we fix the size of IP addresses (the proof is obvious), our framework allows us to detect violations of the security policy. Indeed, contrary to query analysis which only allow to compute valid instantiations of a given path (since any query describes a path in the network), regular tree model checking provides a way to prove the existence of paths denoting unwanted (and no necessarily direct) flows of packets.

5 Conclusion

We have proposed in this paper an original way to describe firewall semantics using tree automata. We have shown that this approach allows us to perform usual analysis over firewall policies. Some advantages of our approach are the following: First, unlike most of existing works, tree automata allow us to consider the network address translation functionality. Second, the proposed approach allows us to perform structural and query analysis. Misconfigurations are automatically computed from their formal definitions. Moreover, we have shown that our approach opens new perspectives of analyses. For all these reasons, describing firewall semantics with tree automata seems to be relevant and promising. We currently work on several follows-up of the results presented in this paper. The first one concerns composition of firewalls: we try to extend the specification of firewalls to take into account the routing functionality. It would allow us to define the semantics of complex compositions of firewalls. We also want to address the problem of minimalizing the set of rules of a given firewall, *i.e.* building from a tree automata based firewall semantics an equivalent firewall having a minimum of filtering and NAT rules. Finally, we currently work on the implementation of the work presented in this paper.

Acknowledgment

The author would like to thank Jean-Christophe Bach for discussion about firewalls and Horatiu Cirstea for its improvement suggestions. This work was supported by ANR SSURF and région Lorraine.

References

[ABR08] 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.

[AJMd02] P.A. Abdulla, B. Jonsson, P. Mahata, and J. d'Orso. Regular tree model checking. In *Intl Conf. on Computer Aided Verification*, volume 2404 of *LNCS*. Springer, 2002.

[ALdR05] Parosh Aziz Abdulla, Axel Legay, Julien d'Orso, and Ahmed Rezine. Simulation-based iteration of tree transducers. In *Tools and Algorithms for the Construction and Analysis of Systems*, LNCS, pages 30–44. Springer, 2005.

[ASH03] E. Al-Shaer and H. Hamed. Firewall policy advisor for anomaly detection and rule editing. In *Intl Symp. Integrated Network Management*, volume 246 of *IFIP Conf. Proceedings*, pages 17–30. Kluwer, 2003.

[AsH04] E. Al-shaer and H. Hamed. Discovery of policy anomalies in distributed firewalls. In *IEEE Intl Conf. on Computer Communications*, pages 2605–2616. IEEE C.S., 2004.

[ASHBH05] 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.

[BB07] 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.

[BHRV06] A. Bouajjani, P. Habermehl, A. Rogalewicz, and T. Vojnar. Abstract regular tree model checking. *Electronic Notes in Th. Comp. Sci.*, 149(1):37–48, 2006.

[BN98] F. Baader and T. Nipkow. *Term rewriting and all that.* C.U.Press, 1998.

[CCBGA05] 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.

[CCBGA06] 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.

[CDG$^+$08] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: `http://gforge.inria.fr/projects/tata/`, 2008.

[CF02] B. Conoboy and E. Fictner. Ip filter based firewalls howto. Available on: `http://www.obfuscation.org/ipf/ipf-howto.pdf`, 2002.

[DT90] M. Dauchet and S. Tison. The theory of ground rewrite systems is decidable. In *Symp. on Logic in Computer Science*, pages 242–248. IEEE C.S., 1990.

[EZ01] P. Eronen and J. Zitting. An expert system for analyzing firewall rules. In *Nordic Work. on Secure IT Systems*, pages 100–107, 2001.

[FL06] V. Fuller and T. Li. *Classless Inter-Domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan.* The Internet Society, 2006. RFC 4632.

[GL04] M.G. Gouda and A.X. Liu. Firewall design: Consistency, completeness, and compactness. In *IEEE Intl Conf. on Distributed Computing Systems*. IEEE C.S., 2004.

[HAS06] H. Hamed and E. Al-Shaer. Taxonomy of conflicts in network security policies. *IEEE Communications Magazine*, 44(3):134–141, 2006.

[Haz00] S. Hazelhurst. Algorithms for analysing firewall and router access lists. Technical Report TR-WITS-CS-1999-5, University of the Witwatersrand, South Africa, 2000.

[Len10] O. Lengál. An efficient tree automata library. Master's thesis, Brno university of technology, 2010.

[LG09] Alex X. Liu and Mohamed G. Gouda. Firewall policy queries. *IEEE Transactions on Parallel and Distributed Systems*, 20(6):766–777, 2009.

[Liu08] A.X. Liu. Formal verification of firewall policies. In *IEEE Intl Conf. on Communications*, pages 1494 – 1498. IEEE C.S., 2008.

[MK05] R. Marmorstein and P. Kearns. An open source solution for testing NAT'd and nested iptables firewalls. In *Conf. on Large Installation System Administration*, pages 103–112, Berkeley, CA, USA, 2005. USENIX.

[Rus02] R. Russell. Linux 2.4 packet filtering howto. Available on: `http://www.netfilter.org/documentation`, 2002.