Network-level access control policy analysis and transformation

Network-Level Access Control Policy Analysis and Transformation

Cataldo Basile, Alberto Cappadonia, and Antonio Lioy, *Member, IEEE*

***Abstract*—Network-level access control policies are often specified by various people (network, application, and security administrators), and this may result in conflicts or suboptimal policies. We have defined a new formal model for policy representation that is independent of the actual enforcement elements, along with a procedure that allows the easy identification and removal of inconsistencies and anomalies. Additionally, the policy can be translated to the model used by the target access control element to prepare it for actual deployment. In particular, we show that every policy can be translated into one that uses the "First Matching Rule" resolution strategy. Our policy model and optimization procedure have been implemented in a tool that experimentally demonstrates its applicability to real-life cases.**

***Index Terms*—Firewall configuration, policy anomalies, policy conflict, policy transformation, policy translation.**

I. Introduction

THE CURRENT landscape for network-level access control is quite complex as it can be implemented in several physical and logical elements: firewalls, routers, and proxies, in addition to the application servers that have the required capabilities. However, while traditionally controls were performed in a single place (border or central firewall), nowadays the defence-in-depth approach implies the use of security controls at multiple places and different levels. Therefore, configuring all these elements in a consistent way is a rather complex matter due to both technical and administrative issues. Examples of the former are the different semantics of access control rules, while the latter are typically related to the different management domains involved in the process (network, security, and application administrators). It is therefore highly desirable to have a formal way of specifying access controls and verifying their consistency across different elements and management domains. Automatic translation to the formats used for the actual enforcement would be a bonus.

The IEEE has formalized the access control model through the concept of *policy-based management* [1], where a policy is defined as "a definite goal, course or method of action to guide and determine present and future decisions." This is enforced through various elements, the most relevant ones being the policy enforcement point (PEP) and the policy decision point (PDP) that collectively intercept the access request, check which part of the policy it matches, and finally enforce the corresponding action (permit, deny, encrypt, ...). Commonly, this model is not implemented in two separate elements, but in one single block (e.g., a firewall) that acts as PEP/PDP and stores the policy locally in a manufacturer-specific format. We will therefore use hereafter the term "enforcement element" (or block or device) for the integrated PEP/PDP case.

When using policies for access control in a large multidomain networked computer system, conflicts may arise, as an access request may match more than one part of the policy. This is because these parts have been written by different administrators (based on different requirements) or simply as the result of a mistake. The PDP typically solves conflicts by applying a *resolution strategy*.

The most common strategies are First/Last Matching Rule (FMR/LMR) that respectively select the action from the first/last applicable rule in an ordered list, and Allow/Deny Takes Precedence (ATP/DTP) that enforce Allow or Deny in the case of contradicting actions simultaneously activated. Access control for Database Management System (DBMS) sometimes uses Most Specific Takes Precedence (MSTP), which enforces the action of the rule which has the most specific condition. Alternatively, the opposite strategy Least Specific Takes Precedence (LSTP) may be adopted. Automatic conflict resolution strategies are potentially very dangerous because they actually hide mistakes and conflicting requirements that, on the contrary, should be identified and explicitly addressed. Moreover, if different administrators specify policies using different strategies, their direct comparison and integration is simply not possible.

It should be noted that conflicts are only one special case of policy anomalies. Duplicate or redundant rules may exist, and they should also be identified in order to optimize the policy before enforcement in an individual element.

This paper addresses these problems by introducing a formal model for the general definition of a policy expressed as a set of rules. By using our model, administrators are no longer forced to specify rules according to the resolution strategy used by the target enforcement devices, and they may choose their favorite strategy, even creating an *ad hoc* one. Additionally, our model is able to support different rule types, thereby extending existing techniques to fields other than those typically used for packet filtering.

Policies written according to our model can be analyzed to classify, detect, and resolve anomalies. We generalized one of the most important works in firewall conflict management, that by Al-Shaer *et al.* [2]–[4]. Moreover, we have included the case of anomalies involving more than two rules, which is useful for policy optimization.

Furthermore, we introduce the definition of *morphism*, a transformation of the policy rules between different

The authors are with the Dipartimento di Automatica ed Informatica, Politecnico di Torino, Turin 10129, Italy (e-mail: cataldo.basile@polito.it; alberto.cappadonia@polito.it; antonio.lioy@polito.it).

representations while keeping the policy semantics unchanged. We will show how, inside our model, it is possible to translate between different representations of policies. As an example, we present the transformation of a generic policy into one suitable for devices using the FMR or LMR strategy. The correctness of the translation process is formally proven.

The effectiveness of our theoretical results has also been verified experimentally, as we developed a tool based on our model and used it with large policies. Results confirm that our approach is feasible and usable in practice.

The rest of the paper is organized as follows. Section II summarizes the existing works about firewall policy analysis, while Section III outlines the main contributions of this paper. The proposed policy model is described in Section IV, where a general anomaly classification is also defined. Following that, in Section V, the translation process is introduced, and Section VI presents the tool that implements the model, together with the experimental results. Section VII draws conclusions and provides hints for future works. The Appendix provides the formal proof of the proposed translation process.

This paper is an extension of [5] and [6]. The main idea was developed inside the Positif project (EC contract IST-002314) and has been improved in the PoSecCo one (EC contract IST-257129).

II. Background

The most relevant work in the field of firewall rule analysis is that of by Al-Shaer *et al.* [2], along with its extension to a distributed scenario [3] and to IPsec [4]. The work in [2] is focused on the packet filtering scenario and considers rules (corresponding to rows of a firewall table) composed by five fields (conditions): IP source and destination addresses, TCP source and destination ports, and the Protocol Type field of the IP header. Al-Shaer *et al.* introduce the concept of anomaly, defined as "the existence of two or more filtering rules that may match the same packet or the existence of a rule that can never match any packet." They consider rule pairs in ordered lists (where conflicts are solved using the FMR strategy) and identify four possible anomalies that depend on rule priorities and enforced actions. Given two rules r_1 and r_2, where r_1 is the higher priority rule, Al-Shaer *et al.* define the following.

1) *Shadowing anomaly*: r_2 is shadowed if r_1 matches all the packets that r_2 matches, r_2 will never be activated;
2) *Correlation anomaly*: r_1 and r_2 are correlated if: a) they enforce different actions; b) packets matching both rules exist; and c) packets matching only r_1 (or r_2) exist;
3) *Generalization anomaly*: r_2 is a generalization of r_1 if: a) they enforce different actions; and b) all the packets matching r_1 also match r_2, but not the contrary;
4) *Redundancy anomaly*: r_2 is redundant if r_1 matches the same packets and enforces the same action as r_2, so the removal of r_2 will not change the policy behavior.

A further anomaly is defined: the *irrelevance anomaly*. A rule is irrelevant if it does not match any packet that could pass through the firewall. It does not concern relations between rules, but rather those between a rule and the enforcing device. Moreover, they propose an algorithm to discover anomalies in rule sets and another one to identify the anomalies originated by the insertion of a new rule into an existing rule list. The resolution of anomalies is delegated to the administrator.

The approach of Al-Shaer *et al.* has three major limitations: It deals only with the packet filtering scenario, it works only if the FMR strategy is used, and it considers only anomalies in rule pairs without taking into account anomalies involving more than two rules.

Some of these limitations apply also to other works that extend the approach of Al-Shaer *et al.* or reformulate it with a different formalism. In [7], the authors extend the work in [2] to support different firewall rule formats, using a model based on mathematical relations between rule fields. The approach proposed in [8] transforms rules into bit vectors to efficiently detect and classify the anomalies presented in [2]. Furthermore, the bit vectors permit the usage of additional condition fields. In [9], the authors present a minor extension of anomalies in [2] (pre- and post-redundancy), introduce an algorithm to automatically resolve conflicts, and prove the existence of a solution. Anomalies involving more than two rules have been introduced in [6]. An algorithm to detect them only in elements using the FMR is presented in [10].

Other works deal with IPsec policies. Their analysis focuses on a single scenario and is not as general as our approach. Zao *et al.* [11] introduced the idea of combining conditions of IPsec rules belonging to different fields (e.g., source or destination ports) by using the Cartesian product. This idea was used in [12] to detect and solve IPsec conflicts when translating abstract requirements to implementable rule sets. In [13], an IPsec policy is seen as Access Control List (ACL) and Encrypt List (EL). The authors classify anomalies for ACL and EL in intra- and interdevice scenarios and propose an algorithm to classify conflicts, but they do not address the resolution phase. Moreover, a model for anomaly classification was presented also for IP Differentiated Services policies [14].

An important area of policy analysis concerns the conflicts related to higher levels of specification. Works from Sloman *et al.* model policies as objects [15] and classify inconsistencies of policy for the management of distributed systems, together with the techniques to solve them [16]–[19]. In [20] and [21], the authors propose a solution combining deontic logic with temporal operators. Conflicts are classified under four categories, and the detection is analyzed in two cases: *static conflict*, incongruence found during an offline initialization phase, and *dynamic conflicts*, unpredictable conflicts that may result from a runtime action. Despite the fact that the model is very interesting, the analysis is only partial, and a small number of conflicts remain to be resolved at runtime.

III. Improvement

Considering the packet filter scenario, our approach extends some aspects of the work of Al-Shaer *et al.* We introduce a new formalism that expresses a general formulation of a policy as a rule set. This formulation is suitable for use in different scenarios where the security policy can be represented by means of a set of rules and allows the use of *ad hoc* resolution strategies. Therefore, the model allows an easy policy specification (Section IV).

We also generalize and extend the anomaly classification in [2] and [3], to make its formulation independent of the

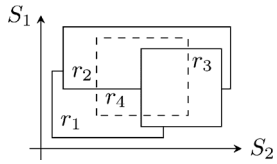

Fig. 1. Unnecessary rule example: r_4 will never be activated.

resolution strategies. In fact, our classification is valid for every resolution strategy and not only for FMR (Section IV-D). It should be noted that our work is independent of specific resolution strategies in the sense that our theory and algorithms are general and do not need to be modified for each possible resolution strategy. However, the specific resolution strategy is explicitly used for classification purposes and by translation algorithms, as will be evident in the rest of the paper.

Furthermore, we introduce two new anomaly classes that may occur when more than two rules are considered: the *general redundancy anomaly* and the *general shadowing anomaly* (Section IV-D). These are effective in finding out if a rule is unnecessary for the policy representation—for example, a rule could not be shadowed by a single rule in the rule set, but it may be shadowed by the union of two or more rules. Referring to Fig. 1, we assume that according to a given resolution strategy, rules r_1, r_2, and r_3 prevail over r_4, so all the packets that match r_4 will be matched also by another rule. Additionally, identifying and removing more anomalies permits better optimization of rule sets. This further reduction strongly depends on how the rules set is made. If an experienced administrator wrote an anomaly-free rule list, then the reduction would not be possible, but often shadowed or equivalent rules are present, according to statistical data [2]. On average, our approach allows a better reduction than existing techniques, as it removes more types of anomalies. However, this reduction cannot be easily quantified. Our model does not apply to scenarios where conflicts may be detected only at runtime (such as access control models using the separation of duties) or contexts where a decision depends on previous ones (as for the maximum number of allowed connections or flows).

Our model permits the construction of semantically equivalent representations of a policy, i.e., they enforce the same actions on the same packets (Section V). Our solution makes use of an algebraic structure: the semi-lattice [22]. Through the semi-lattice, we have a representation very effective for policy manipulation because it eliminates the differences between the various resolution strategies. We call this representation *canonical form* (Section V-A). Moreover the canonical form is important because it is an intermediate policy representation that permits to translate a policy, which uses whatever resolution strategy, into one that uses another strategy (e.g., specific of a certain target device, Section V-B). By using the canonical form, we are able to prove that *every policy representable using our model can be translated to an equivalent policy that uses the* FMR *or* LMR *strategy*. From an implementation point of view, administrators could use only FMR-based devices while, at the same time, they could specify policies using more general strategies.

Finally, a minor improvement of our model, anyway useful for practical applications, is its ability to support different rule types, with the constraint that for each type of condition (e.g., source IP addresses, source port) there exists the intersection operation, which corresponds to the logical conjunction. This is not a limitation since the intersection operator is usually defined for most rule fields in currently available filtering devices. For example, we support all the filter entries of the DMTF CIM policy model [23].

IV. Rule and Policy Model

The model presented here focuses on three factors: anomaly detection, anomaly resolution, and the default action. It improves the model of rules in [5] and policies in [6].

Definition 1 (Rule Model): According to RFC-3198 [1], we model a rule in the format "if *condition* then *action*" that consists of a condition clause and an action clause. The available actions are all well known and organized into an *action set* $\mathcal{A}$.

Possible actions are Accept and Deny, i.e., $\mathcal{A} = \{A, D\}$.

Definition 2 (Condition): A *condition* s in a particular *selector* S (e.g., a field, such as the IP source address) is associated with the subset of values for which it evaluates to true.

The selector represents the set of all possible values that a field may take (e.g., the values $0 \dots 65\,535$ for TCP ports). To express that a condition concerns a selector, we write $s \subseteq S$. Conditions in different selectors cannot be linked together directly because they belong to different types and different decision spaces (e.g., source IP address, destination port). It is therefore necessary to expand the decision space. We use the Cartesian product to link different selectors together. This in turn generates the following definitions

Definition 3 (Selection Space): The *selection space* is the set $\mathfrak{S} = S_1 \times S_2 \times \cdots \times S_{\mathrm{m}}$, with m being the number of available selectors.

Definition 4 (Condition Clause): A condition clause c is a subset of a given selection space, $c = s_1 \times s_2 \times \cdots \times s_{\mathrm{m}} \subseteq S_1 \times S_2 \times \cdots \times S_{\mathrm{m}} = \mathfrak{S}$.

In our model, a condition clause is a *hyper-rectangle*. Therefore, even if $\mathfrak{S}$ represents the totality of the packets, not all the subsets of $\mathfrak{S}$ are valid condition clauses, only the hyper-rectangles created as in Definition 4. This restriction is an intrinsic constraint of the policy languages used at network level, where the rules are specified by defining a condition for each selector (see Section IV-A).

Definition 5 (Rule): Given Definitions 1–4, a rule can be expressed as $r = (c, a)$, where $c \subseteq \mathfrak{S}$ and $a \in \mathcal{A}$.

This rule model allows to ease the detection of when two or more rules can be activated simultaneously. This occurrence is identified as the nonempty intersection of rule condition clauses. Therefore, all the anomalies (see Section IV-D) can be identified by analyzing the result of the intersection of condition clauses.

In this paper, a policy is seen as a "set of rules to administer, manage, and control access to network resources" rather than "a definite goal, course or method of action" [1]. We refer to policies that satisfies the first definition as rule-based policies—that is, a policy specified by means of a set of rules R

$$R = \{r_i = (c_i, a_i)\}_i, i \in [1, n]$$

where n is the number of rules in R.

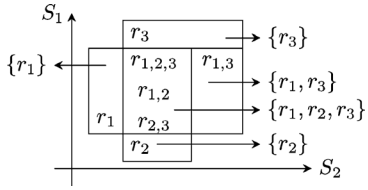

Fig. 2. $\texttt{match}_R$ function: an example.

However, this preliminary definition of policy is incomplete because it does not take into account two special cases. The first is which behavior the system must have when conflicting rules exist, and the second is what to do when no rule applies. The decision of the action to be applied in the case of conflict is abstracted by means of the *resolution strategy*

$$\mathfrak{R} : 2^R \to \mathcal{A}$$

that decides which action the system must perform.[1] Given a set of rules representing a policy, the resolution function maps all the possible groups of rules to an action $a \in \mathcal{A}$. In other words, if a packet matches more than one rule, first the matching rules are identified, then the resolution strategy chooses the action to be enforced between the matching ones.

Since every rule determines the behavior of the system within the limits of its condition clause, it is necessary to set the behavior of the system when no rules apply to a packet. This purpose is achieved by the introduction of a default action $d \in \mathcal{A}$ to enforce in this particular case.

We will now proceed to model a policy as function $p : \mathfrak{S} \to \mathcal{A}$ that connects each point of the selection space to an action taken from the action set $\mathcal{A}$ according to the rules in R. For instance, for the packet filter scenario, a point in the selection space is a packet, and the policy decides which action to apply to it. A policy p can thus be formally modeled as

$$p(x) = \begin{cases} d, & \text{if } \texttt{match}_\mathfrak{R}(x) = \emptyset \\ a_i, & \text{if } \texttt{match}_\mathfrak{R}(x) = \{r_i\} \\ \mathfrak{R}(\{r_l, r_m, \ldots\}), & \text{if } \texttt{match}_\mathfrak{R}(x) = \{r_l, r_m, \ldots\} \end{cases}$$

where $\texttt{match}_R$ is a function returning the subset $M \subseteq R$ of rules whose conditions match x, formally defined as

$$\begin{aligned} \texttt{match}_R : \mathfrak{S} &\longrightarrow 2^R \\ x &\longmapsto M = \{r_i \in R | x \in c_i\}. \end{aligned}$$

By defining $\mathfrak{R}(\emptyset) = d$ and $\mathfrak{R}(r_i) = a_i$ (see Fig. 2), the policy p may be rewritten as

$$p(x) = \mathfrak{R}(\texttt{match}_R(x)).$$

Resolution strategies may base their decision not only on intrinsic rule data (e.g., condition clause and action clause), but also on "external data" related to each rule, such as priorities, identity of the creator, and creation time. These attributes are useful to model not only simple resolution strategies, such as the FMR, but also more complex and expressive strategies created to meet the specific needs of an information system (see Section IV-B). In these cases, the resolution strategy works in two steps: First, the rules are related to the external data, then the action to be enforced is selected according to a function that operates both on the rules and the external data.

Formally, every rule r_i is extended through a function ε_E so that the rule turns into

$$\varepsilon_E(r_i) = (r_i, f_1(r_i), f_2(r_i), f_3(r_i), \ldots)$$

where $E = \{f_j : R \to X_j\}_j$ is a set of functions mapping rules to a specific set of external attributes X_j. Therefore, the resolution strategy $\mathfrak{R}$ is the composition between the extension function ε_E, and a resolution function $\mathfrak{R}_E$ that works on the rule extensions, that is $\mathfrak{R} = \mathfrak{R}_E \circ \varepsilon_E$

$$\begin{aligned} \mathfrak{R} : \quad \{r_l, r_m, \ldots\} &\overset{\varepsilon_E}{\longmapsto} \{\varepsilon_E(r_l), \varepsilon_E(r_m), \ldots\} \\ &\overset{\mathfrak{R}_E}{\longmapsto} a. \end{aligned}$$

It is clear that a policy is entirely specified by means of a set of rules, a resolution strategy (made up of an extension function and a resolution function), and a default action. Therefore, we say here that the 4-tuple $(R, \mathfrak{R}_E, E, d)$ is a representation of the policy p. This definition is independent of the resolution strategy and allows the use of a generic one. If the resolution strategy does not use external data, then the set E is empty and the policy will be represented as $(R, \mathfrak{R}, \emptyset, d)$ since $\varepsilon_\emptyset$ is the identity function in R.

We say that two policy representations $(R_1, \mathfrak{R}_{E_1}, E_1, d_1)$ and $(R_2, \mathfrak{R}_{E_2}, E_2, d_2)$ are equivalent if

$$\forall x \in \mathfrak{S}, \mathfrak{R}_1(\texttt{match}_{R_1}(x)) = \mathfrak{R}_2(\texttt{match}_{R_2}(x))$$

where $\mathfrak{R}_1 = \mathfrak{R}_{E_1} \circ \varepsilon_{E_1}$ and $\mathfrak{R}_2 = \mathfrak{R}_{E_2} \circ \varepsilon_{E_2}$. In the rest of the paper, for ease of notation, when we mention the policy p, we assume that it is represented as $(R, \mathfrak{R}_E, E, d)$ and the resolution strategy is $\mathfrak{R} = \mathfrak{R}_E \circ \varepsilon_E$. We also assume that a generic rule r has condition clause c and action clause a and that a rule r_i has condition clause c_i and action clause a_i.

A. Set Operations Between Conditions and Condition Clauses

As our approach requires many operations between clauses, some discussion about their efficiency is needed. If we represent a condition over a specific selector S as a subset, then logical operations between conditions map onto set operations. i.e., conjunction maps to intersection ($s_1 \wedge s_2 \to s_1 \cap s_2$), disjunction to union ($s_1 \vee s_2 \to s_1 \cup s_2$) and complement to set difference ($\neg s \to S \setminus s$). These operations on conditions over a single selector return a set condition, and their computational complexity is low. However, these results do not hold for condition clauses because in our model they are Cartesian products of conditions, that is, hyper-rectangles, and set operations on a Cartesian product do not necessarily generate a Cartesian product. In turn, this may create some problems. As an example, let c_1 and c_2 be two condition clauses (with m selectors) defined as follows:

$$\begin{aligned} c_1 &= s_{1,1} \times s_{1,2} \times \cdots \times s_{1,m} \\ c_2 &= s_{2,1} \times s_{2,2} \times \cdots \times s_{2,m}. \end{aligned}$$

Now consider the following operations.

[1] We use the notation of the *power set* of a set K. If K is a finite set with $|K| = k$ elements, the power set of K has $|2^K| = 2^k$ elements.

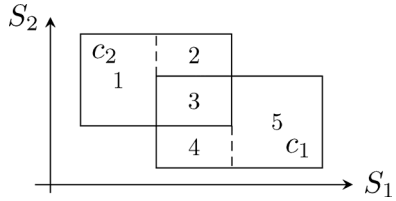

Fig. 3. Representation of set-based operations among hyper-rectangles.

- The *intersection* returns a condition clause, obtained by intersecting single conditions in each selector, such as rectangle 3 in Fig. 3

$$c_\cap = c_1 \cap c_2 = s_{1,1} \cap s_{2,1} \times s_{1,2} \cap s_{2,2} \times \cdots \times s_{1,m} \cap s_{2,m}.$$

- The *union* $c_\cup = c_1 \cup c_2$ is not in general a condition clause, and its representation with hyper-rectangles may require up to $m+1$ condition clauses—for instance, a possible but not unique representation may use rectangles 1 and 2 and the entire condition clause c_1 in Fig. 3.
- The *set difference* $c' = c_1 \setminus c_2$ is not in general a condition clause, and its representation may require up to m condition clauses—e.g., a possible but not unique representation may use rectangles 1 and 2 in Fig. 3.

Intersection aside, the other operations may therefore generate a much higher number of rules than that of the operands, and this number increases with the selector number. We have therefore developed an approach mainly based on intersection operations (Section V-A). Moreover, for those cases where the use of another operation (namely the union) is unavoidable, we have defined an algorithm that uses it only for simple cases (Section V-C), thereby limiting the computational complexity of our approach.

B. Examples of Resolution Strategies

Let us first consider the existing FMR resolution strategy. In this case, the priority is an external data associated to each rule for conflict resolution purposes. Formally, we model the assignment of priority to rules via a function π mapping each rule to an integer: The smaller the integer, the higher the priority. Priorities are normally implicit as FMR typically uses an ordered list, with high-priority rules listed first and the constraint that two rules cannot be assigned the same priority.

A policy p defined using the rule set R and the FMR strategy can be formally expressed as $(R, \texttt{FMR}, \{\pi\}, D)$, where D is the "Deny" action and

$$\texttt{FMR}: \quad \{(r_l, \pi(r_l)), (r_m, \pi(r_m)), \ldots\} \quad \mapsto \quad a_k$$

where a_k is the action of the rule r_k with the maximum priority, that is, the minimum value of π

$$\pi(r_k) = \min_{r \in \{r_l, r_m, \ldots\}} \{\pi(r)\}.$$

Likewise we can model a policy that uses the *Last Matching Rule* strategy as $(R, \texttt{LMR}, \{\pi\}, D)$ that is analogous to FMR, but it selects the action from the r_k with the minimum priority

$$\pi(r_k) = \max_{r \in \{r_l, r_m, \ldots\}} \{\pi(r)\}.$$

Priorities in FMR and LMR are the simplest way to define the precedence between rules, but this could be a limitation.

As another example, let us consider the resolution process in a scenario different from packet filtering—for instance, the access control conflicts at a target in IBM Lotus Domino [24]. In this environment, rules are definded by two selectors: the *scope* and the *subject*. The resolution strategy consists of maximum three steps.

1) *Scope specificity*: Select rules with the highest scope specificity (according to: *all* < *this container* and *all descendants* < *this container*). If one rule has been selected, then enforce its action, else go to 2
2) *Subject specificity*: Select, among rules with the same scope, the rules with the highest subject specificity (according to: *default* < *wildcard* < *group* < *self* < *individual user/server*). If just one rule has been selected, then enforce its action, else go to 3
3) *Combined rules*: "Deny access takes precedence over Allow access" (i.e., DTP), that is, enforce Deny if at least one action clause of the selected rules is Deny.

In this case, the precedence between rules is complex and involves different external information, therefore anomaly detection would need an *ad hoc* method. To model the Domino resolution strategy, we introduce two extension functions φ and ψ: The former relates a rule to the scope specificity, while the latter to the subject specificity. Furthermore, we use two selection functions σ_φ and σ_ψ that select, respectively, rules with the most specific scope and the most specific subject in a set of rules, that is $(K = \texttt{match}_R(x))$

$$\begin{array}{rccc} \sigma_\varphi: & 2^R & \to & 2^R \\ & K \subseteq R & \mapsto & K_\varphi = \{r_i \in K | \varphi(r_i) = \max_{x \in K} \varphi(x)\} \\ \sigma_\psi: & 2^R & \to & 2^R \\ & K \subseteq R & \mapsto & K_\psi = \{r_i \in K | \psi(r_i) = \max_{x \in K} \psi(x)\}. \end{array}$$

The Deny Takes Precedence strategy is modeled as

$$\texttt{DTP}: \quad \begin{array}{ccc} 2^R & \to & \mathcal{A} \\ \{r_l, r_m, \ldots\} & \mapsto & \begin{cases} D, & \text{if } \exists\, r_k \in \{r_l, r_m, \ldots\} \\ & \text{such that } a_k = D \\ A, & \text{otherwise.} \end{cases} \end{array}$$

Finally, the resolution strategy is $\mathfrak{R}_{\text{domino}} = \texttt{DTP} \circ \sigma_\varphi \circ \sigma_\psi$, and the policy may be described as $(R, \mathfrak{R}_{\text{domino}}, \{\varphi, \psi\}, D)$

Another scenario where our model may be helpful is in the composition of independently specified rule sets. For instance, consider the case of a company where there are several network managers, each one responsible for a specific portion of the network, and a single security administrator, in charge of the global security of the network and the actual configuration of the central firewall. Each network manager specifies the policy as a rule set for his subnetwork according to his preferred resolution strategy. We name these rule sets $M_1, M_2, \ldots$ and $\mathbb{M} = \bigcup_i M_i$. The security administrator is in charge of defining the policy for the sensitive part of the network, and the interarea services. For ease of specification, the administrator may write distinct rule sets for different security needs, e.g., the server policy, the DMZ policy. These rule sets are divided into high-priority rule sets ($H_1, H_2, \ldots$ and $\mathbb{H} = \bigcup_i H_i$) and low-priority rule sets ($L_1, L_2, \ldots$ and $\mathbb{L} = \bigcup_i L_i$), each one with its own resolution

strategy. The security administrator wants to implement the following global resolution strategy.

- First select the matching rules from the high-level priority rule sets; if none, then take those from the managers' rule sets; if none, then take the ones from the low-priority rule set;
- Divide selected rules according to the rule set to which they belong and use its own resolution strategy to derive an action from each rule set;
- If actions are contradictory, Deny takes precedence.

This resolution strategy can be modeled by resorting to these functions.

- $\mu : 2^R \to 2^R$ that selects the rules from the highest priority rule sets, formally

$$\mu : K \subseteq R \mapsto \begin{cases} K \cap \mathbb{H}, & \text{if } K \cap \mathbb{H} \neq \emptyset \\ K \cap \mathbb{M}, & \text{if } K \cap \mathbb{M} \neq \emptyset \wedge K \cap \mathbb{H} = \emptyset \\ K \cap \mathbb{L}, & \text{if } K \cap \mathbb{M} = \emptyset \wedge K \cap \mathbb{H} = \emptyset. \end{cases}$$

- $\{\phi_{H_1}, \phi_{H_2}, \ldots\} : 2^R \to \mathcal{A}$ is a family of functions that selects the rules belonging to a given rule set H_i and uses the internal resolution strategy to derive an action, that is

$$\phi_{H_i} : K \subseteq R \mapsto \begin{cases} \mathfrak{R}_{H_i}(K \cap H_i), & \text{if } K \cap H_i \neq \emptyset \\ \emptyset, & \text{otherwise.} \end{cases}$$

- In the same way, we define $\{\phi_{M_1}, \phi_{M_2}, \ldots\}$ and $\{\phi_{L_1}, \phi_{L_2}, \ldots\}$.
- $\Phi : 2^R \to 2^{\mathcal{A}}$ performs the union of the results of the ϕ functions, formally

$$\Phi : K \subseteq R \mapsto \bigcup \phi_{H_i} \cup \bigcup \phi_{M_i} \cup \bigcup \phi_{L_i}$$

- $\delta : 2^{\mathcal{A}} \to \mathcal{A}$ returns D if a least one action is Deny, A otherwise.

The resolution function is then $\mathfrak{R}_c : 2^R \to \mathcal{A} = \delta \circ \Phi \circ \mu$. Every policy defined according to $\mathfrak{R}_c$ may be analyzed using the techniques in Section IV-D and translated to the firewall configuration using the morphisms as in Section V without the need for *ad hoc* algorithms.

C. Defining Custom Resolution Strategies

Defining new resolution strategies is a sensitive task. All the examples presented in the previous section use ordered sets, and this is not by chance: A well-formed resolution strategy is always designed so that a partial order is defined over the data used for conflict resolution. For instance, `DTP` uses actions and imposes $D > A$, while `FMR` uses integers as priorities (see [25] for further details). A simple test to understand if a resolution strategy is sound is to consider rules having the same (generic) condition clause and see if the strategy is able to find the action to apply, then repeating the test for rules having the same external data, one field at a time.

D. Anomaly Classification

The formalism in Section IV allows the anomalies defined in [2] for `FMR` to be redefined in a more general way, so that they can be detected independently of the used resolution strategy and easily applied to different contexts. We first introduce an abstract definition of a policy containing anomalies, and then we show how to detect anomalies in rule sets. We distinguish two cases.

- *Conflict anomaly:* A policy p is *conflictual* if there exists a rule $r = (c, a) \in R$ and $\exists x \in c$ such that $p(x) \neq a$.
- *Suboptimality:* A policy p is *suboptimal* if it contains a *hidden rule* $r \in R$ such that its removal from the rule set does not affect the policy function; in other words, p is suboptimal if there exists $r \in R$ such that

$$(R \setminus \{r\}, \mathfrak{R}_E, E, d) \text{ is equivalent to } (R, \mathfrak{R}_E, E, d).$$

Note that these anomalies do not exclude each other. For instance, a policy may be conflicting and suboptimal at the same time. In fact, a hidden rule may also be conflicting. Redundancy anomaly in [2] is an example of suboptimality. The generalization and correlation anomaly are examples of conflict anomalies, while the shadowing anomaly is simultaneously a suboptimality and conflict anomaly.

These two types of anomaly highlight different policy specification issues: Suboptimal policies contain rules that do not give any contribution to the policy function, yet affect the performance of the PEP/PDP, while conflicting rules provide evidence of errors in policy specification and need to be carefully examined by the administrators.

Once anomalies are defined in theory, it is necessary to identify them in practice. In particular, we are interested in detecting anomalies in rule sets independently from the used resolution strategy. An analysis of rule pairs is sufficient to manage the conflict anomaly. In fact, if two rules are activated simultaneously but they enforce different actions, at the end, one action will be overridden. On the other hand, suboptimality cannot be treated correctly by considering rule pairs only. The effective contribution of a rule to the policy depends on all the other rules in the rule set. In fact, the condition clause of a hidden rule may not be completely covered by the condition clause of a single rule.

Conflicting rules can be classified according to the following.

- *Correlation anomaly:* Rules r_i and r_j are correlated if

$$c_i \cap c_j \neq \emptyset \ \wedge \ a_i \neq a_j.$$

- *Generalization anomaly:* A particular case of correlation anomaly, a rule r_i is a generalization of a rule r_j if

$$c_i \subseteq c_j \ \wedge \ a_i \neq a_j \ \wedge \ \mathfrak{R}(\{r_i, r_j\}) = a_i.$$

Although the presence of correlated rules may be detected by analyzing only rule pairs, the whole rule set must be considered when evaluating the action that they actually enforce. For instance, in Fig. 2, rules r_1 and r_2 are correlated, but the action to enforce within their intersection also depends on r_3. We will deal with this issue in Section V-A.

Sometimes it is possible to find cases of suboptimality involving only two rules (such as redundant and shadowed rules in [2]), but in general this does not cover all possible cases. For instance, in the case of Fig. 4(b) and (c), the rule r_4 is hidden. Additionally, the case in Fig. 4(d) is not detectable by using existing methods: r_4 is redundant even if it has the highest priority.

Our model identifies two cases of suboptimality that may occur when more than two rules are involved: the *general*

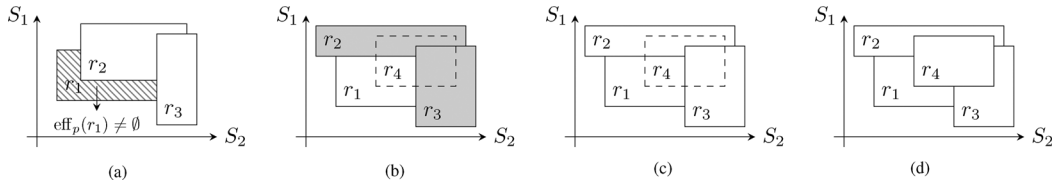

Fig. 4. Graphical representation of eff_p and general anomalies. (a) eff_p function. (b) General shadowing anomaly. (c) General redundancy anomaly. (d) General redundancy anomaly. In (a), only the gray part of r_1 contributes to the policy. In (b)–(d), r_4 is unnecessary and can be removed from the rule set.

redundancy anomaly and the *general shadowing anomaly*. Let us start with the example in Fig. 4(a): If three rules r_1, r_2, and r_3 intersect each other and the r_2 and r_3 prevail over r_1 (according to a given resolution strategy), then the part of c_1 that contributes to the policy is only $c_1 \setminus (c_2 \cup c_3)$. Considering this example, in general terms, we need to "subtract" all the boxes where the rule is unnecessary for the policy representation from the condition clause. By rephrasing the definition of suboptimality, a rule r is hidden if for all $x \in c$

$$\begin{aligned}\mathfrak{R}(\text{match}_R(x)) &= \mathfrak{R}(\text{match}_{R\setminus\{r\}}(x)) \\ &= \mathfrak{R}(\text{match}_R(x) \setminus \{r\}).\end{aligned}$$

This formula states that a rule is unnecessary if the decision of the action to apply may be taken without considering it, for every point of its condition clause. To this purpose, introduce the $\text{eff}_p(r)$ function that returns the portion of the rule that "effectively" contributes to the policy p, formally

$$\begin{array}{rccl}\text{eff}_p: & R & \rightarrow & 2^{\mathfrak{S}} \\ & r & \mapsto & x \in c \text{ such that } \mathfrak{R}(\text{match}_R(x)) \neq \\ & & & \neq \mathfrak{R}(\text{match}_R(x) \setminus \{r\}).\end{array}$$

If $\text{eff}_p(r) = \emptyset$, we can infer that r is unnecessary to p. The explicit calculation of this function does not require all the points in c to be taken into consideration. It is enough to consider the different hyper-rectangles originated by the intersection of rules, whose condition clause has nonempty intersection with c. For instance, in Fig. 4(a), the regions that concern rule r_1 are $c_1 \cap c_3$, $c_1 \cap c_2$ and $c_1 \setminus (c_1 \cap c_2 \cup c_3)$. By using eff_p, the general anomalies are formally defined as the following.

- *General redundancy anomaly:* A rule $r = (c, a)$ is redundant if

$$\text{eff}_p(r) = \emptyset \ \wedge \ \forall x \in c,\, p(x) = a.$$

- *General shadowing anomaly:* A rule $r = (c, a)$ is shadowed if

$$\text{eff}_p(r) = \emptyset \ \wedge \ \exists x \in c,\, p(x) \neq a.$$

A special case of the *general shadowing anomaly* is the *total shadowing anomaly*, which is a rule covered by rules that all enforce an action that is different from the total-shadowed one. Formally, a rule $r = (c, a)$ is total-shadowed if

$$\text{eff}_p(r) = \emptyset \ \wedge \ \forall x \in c, p(x) \neq a.$$

The classification presented here is general and covers all the possible cases of anomaly that can arise in rule sets. In fact, it includes all the pairwise anomalies in [2] plus the case of general anomalies. Identification of general anomalies relies on eff_p, and thus it presents the problem of set union and difference calculation. Thus, to evaluate their contribution to the policy, it is necessary to resort to alternative methods, such as the canonical form for policy representation (Section V-A).

V. The Semantics-Preserving Morphism

For the purpose of policy manipulation, it is useful to introduce the concept of *semantics-preserving policy morphism* or simply *morphism*: a transformation of the policy representation that keeps the policy unchanged. This is very important when administrators specify a policy with high-level techniques, but following that, rules must be translated into a format that real enforcement elements can accept. Anomalies can be analyzed independently of the resolution strategy used by the administrators (by using the techniques in Section IV-D), and the remaining step is just to translate the policy to a format that uses the resolution strategy available at the enforcement element.

Our aim is to identify an approach to change the resolution strategy of a policy while leaving its semantics intact. In other words, given $(R, \mathfrak{R}_E, E, d)$, the objective is to find an equivalent policy $(R', \mathfrak{R}'_{E'}, E', d')$. The target resolution strategy $\mathfrak{R}' = \mathfrak{R}'_{E'} \circ \varepsilon_{E'}$ will use external data from the sets of attributes $\{X_1, X_2, \ldots\}$. Thus, a procedure that implements a morphism must generate: 1) the rule set R'; and 2) the assignments of rules in R' to the external data through the functions in $E' = \{f_1 : R' \rightarrow X_1, f_2 : R' \rightarrow X_2, \ldots\}$. For example, given a filtering policy $(R, \mathfrak{R}_E, E, d)$ expressed via a generic resolution strategy, an administrator may be interested in enforcing it by using a device that only provides `FMR` (e.g., a packet filter). Thus, a morphism is needed to find the rules R_{FMR} to enforce and their priorities $\pi : R_{\text{FMR}} \rightarrow \mathbb{N}$.

In general, a morphism is needed for each pair of resolution strategies. The cost of defining new conflict resolution methods would become too high if in the end a translator were needed for every target resolution strategy. In the next sections, we will present an alternative approach to the translation process. The approach is based on the idea of splitting the process into two steps: The former translates any policy to the same intermediate representation, which is in turn used by the latter step to create device-specific rules. We shall name the intermediate representation *canonical form*. Section V-A demonstrates that a morphism exists from any policy (expressed by means of whatever resolution strategy) to its canonical form and presents a procedure that allows the canonical form to be constructed from every policy. The second step requires the definition of a transformation process from the canonical form to any target resolution strategy. While the high-level resolution strategies are bounded only by the imagination and the needs of policy editors, the number of target resolution strategies is low, as it corresponds to

those strategies implemented by actual enforcement elements. Section V-B, shows the translation of a policy in canonical form into one using the FMR or LMR.

In the next section, only morphisms between policies using the same default action are presented. Morphisms that also change the default action—from $(R, \mathfrak{R}_E, E, d)$ to $(R', \mathfrak{R}'_{E'}, E', d')$—may be treated in the same way by adding a dummy rule into the original rule set, which is valid for the whole selection space $r_d = (\mathfrak{S}, d)$ and has the "lowest possible priority" (e.g., "Deny all" or "Allow all"), that is

$$\begin{array}{ccc} (R, \mathfrak{R}_E, E, d) & \longrightarrow & (R', \mathfrak{R}'_{E'}, E', d') \\ & \searrow \quad \nearrow & \\ & (R \cup \{r_d\}, \mathfrak{R}_E, E, d') & \end{array}$$

A. Set-Based Policy Representation: The Canonical Form

The canonical form is a policy representation relying only on set operations to solve conflicts and is very helpful for policy manipulation. Before defining it, we need to introduce some elements of the set theory and an algebraic structure, the semi-lattice [22], with some of its mathematical properties.

Definition 6 (Partially Ordered Set): A partially ordered set is a system consisting of a set S and a relation "$\leq$" such that $\forall a, b, c \in S$ the relation "$\leq$" is *reflexive* $(a \leq a)$, *antisymmetric* $(a \leq b \wedge b \leq a \Rightarrow a = b)$, and *transitive* $(a \leq b \wedge b \leq c \Rightarrow a \leq c)$.

Definition 7 (Upper Bound): Given a partially ordered set S, an element $u \in S$ is an *upper bound* for $X \subseteq S$ if $a \leq u$ for all $a \in X$.

Definition 8 (Least Upper Bound): An element $u \in S$ is a *least upper bound* (lub) of $X \subseteq S$ if it is an upper bound of X and, for all upper bounds v of X, the relation $u \leq v$ holds.

Definition 9 (Finite Semi-Lattice): A finite semi-lattice is a finite and partially ordered set $(S, \leq)$, and for each pair of elements $x, y \in S$, there exists the lub of the set $\{x, y\}$, and it belongs to S. In the following, we will use the term "semi-lattice" to refer to a finite semi-lattice.

We are now in the position to introduce a binary composition induced by the resolution strategy into the rule set, such that the result: 1) applies where two rules overlap (i.e., at the intersection of their condition clauses); and 2) enforces the action resulting by the application of the resolution strategy.

Formally, given a policy p, we write the composition "$\cdot$" as

$$\begin{array}{rccc} \cdot: & R \times R & \rightarrow & 2^{\mathfrak{S}} \times \mathcal{A} \\ & r_{ij} = r_i \cdot r_j & \mapsto & (c_i \cap c_j, \mathfrak{R}\{r_i, r_j\}). \end{array}$$

We can extend the composition to more than two rules

$$\begin{aligned} r_{i,j,k} &= r_i \cdot r_j \cdot r_k \cdot \ldots \\ &= (c_i \cap c_j \cap c_k \cap \ldots, \mathfrak{R}(\{r_i, r_j, r_k, \ldots\}). \end{aligned}$$

The closure $\overline{R}$ with respect to the composition is the set of all the possible compositions of rules in R

$$\forall\, r \in R \Rightarrow r \in \bar{R} \ \wedge \ \forall\, r_1, r_2 \in \bar{R} \Rightarrow r_1 \cdot r_2 \in \bar{R}.$$

Every element in the closure $\bar{R}$ can be univocally written as a composition of rules in R [26]. $\bar{R}$ may contain rules with the same condition clause that do not contribute to the policy. Fig. 2 presents a policy with rule set $R = \{r_1, r_2, r_3\}$, where $r_{1,2}$, $r_{2,3}$, and $r_{1,2,3}$ have the same condition clause. In this case, match_R will never return $\{r_1, r_2\}$ or $\{r_2, r_3\}$. Indeed, if a packet x is in $c_1 \cap c_2 = c_2 \cap c_3 = c_1 \cap c_2 \cap c_3$, then $\mathrm{match}_R(x) = \{r_1, r_2, r_3\}$. By grouping rules that share the same condition clause, we build a new rule set $R^* \subseteq \bar{R}$ defined according to the following algorithm: If a group contains only one rule, that rule will be included in R^*; if a group contains more than one rule, then only the composition of all the rules in the group will be included in R^*. In Fig. 2, the groups of rules with the same condition clause in $\bar{R}$ are $\{\{r_1\}, \{r_2\}, \{r_3\}, \{r_{1,3}\}, \{r_{1,2}, r_{2,3}, r_{1,2,3}\}\}$. Therefore, R^* will include $\{r_1, r_2, r_3, r_{1,3}, r_{1,2,3}\}$, but not $r_{1,2}$, $r_{2,3}$.

The following properties hold.

- Distinct rules in R^* have distinct condition clauses, that is, $r_i \neq r_j \iff c_i \neq c_j$.
- Given $r_1, r_2 \in R^*$, there exists one and only one rule in R^* whose condition clause is the intersection of the condition clauses of r_1 and r_2—that is, there exists a rule such that its condition clause is $c_1 \cap c_2$ (*closure with respect to the intersection*).

We define in R^* the order relation "$\leq$"

$$r_1, r_2 \in R^*, r_1 \leq r_2 \iff c_1 \supseteq c_2.$$

$(R^*, \leq)$ is thus a semi-lattice since it is a partially ordered set with respect to $\leq$, and the least upper bound of $\{r_1, r_2\}$, which is the rule with condition clause $c_1 \cap c_2$, is contained in R^* due to the closure.

Our next step is to introduce CAN, the resolution strategy in R^*, that takes the action from the least upper bound rule

$$\begin{array}{rl} \mathrm{CAN}: & 2^{R^*} \rightarrow \mathcal{A} \\ & \{r_l, r_m, \ldots\} \mapsto a \quad \text{such that } r = (c, a) \\ & \qquad\qquad = \underset{R^*}{\mathrm{lub}}\{r_l, r_m, \ldots\}. \end{array}$$

The policy $(R^*, \mathrm{CAN}, \emptyset, d)$ is the *canonical form* of $(R, \mathfrak{R}_E, E, d)$. In Appendix-A, we prove that $(R^*, \mathrm{CAN}, \emptyset, d)$ is equivalent to $(R, \mathfrak{R}_E, E, d)$ (Theorem 1). Since the construction of R^* presented before is possible for every policy, we conclude that *there exists a morphism from every policy (expressed by means of whatever resolution strategy) to its canonical form*. Therefore, the canonical form is preferable because it is semantically equivalent to the original policy and permits easy processing. With respect to the original policy, the canonical form has a larger number of rules, but offers the advantage that all the possible cases of conflicting rules have been precomputed with the resolution strategy $\mathfrak{R}$. Thus, any algorithm working on CAN need not to consider the original strategy $\mathfrak{R}$ because the differences between resolution strategies have been eliminated. Furthermore, CAN is a simple resolution function, as it needs only to identify the least upper bound rule, and lub computation is independent of the original resolution strategy. Additionally, the canonical form has the advantage of connecting different worlds. In fact, thanks to the canonical form, *problems concerning the policy may be solved by resorting to techniques from algebra and graph theory*.

The following section illustrates how the semi-lattice representation of the canonical form can be used to translate a policy

into canonical form to a policy that is suitable for implementation in real devices, thus permitting the use of any type of *ad hoc* resolution strategy in the specification phase.

B. FMR and LMR Policy Morphisms

This section introduces a morphism to transform a policy in canonical form into one using FMR. The same algorithm also works for LMR by simply reversing the priorities.

As previously seen, if $(R^*, \mathrm{CAN}, \emptyset, d)$ is the canonical form of a policy p, the FMR-morphism must provide a set of rules R_{FMR}, and the priority assignment $\pi : R_{\mathrm{FMR}} \to \mathbb{N}$.

The FMR-morphism will use the *cover graph*, a common mapping of partially ordered sets to graphs.

Definition 10 (Cover Graph): Given two distinct elements $a, b \in S$, b *is a cover of* a (written $a \prec b$) if and only if $a \leq b$ and no element u such that $u \neq a$ and $u \neq b$ satisfies $a \leq u \leq b$. The *cover graph* of a partially ordered set $(S, \leq)$ is a directed acyclic graph whose vertices are the elements in S and edges are given by the cover relation, i.e., there is an edge between vertices a and b if and only if $a \prec b$ holds.

Therefore, every policy is representable as a cover graph through its canonical form. That is, given a policy p, we will indicate as $G(p)$ the cover graph associated to $(R^*, \leq)$. As a consequence of being a semi-lattice, $(R^*, \leq)$ contains a top element T that is greater than all the others. Formally

$$\begin{aligned} T &= \operatorname*{lub}_{r \in R^*} \{r\} = r_1 \cdot r_2 \cdot \ldots \cdot r_n \\ &= (c_1 \cap c_2 \cap \cdots \cap c_n, \mathfrak{R}\{r_1, r_2, \ldots, r_n\}) \end{aligned}$$

where n is the number of rules in R. Note that R^* will contain a top element T even if $c_1 \cap c_2 \cap \cdots \cap c_n = \emptyset$. The graph representation is necessary to find an optimized procedure that selects the rules to be included in R_{FMR} and to assign them priorities. To understand how to assign priorities, let us consider two distinct conflicting rules r_x, r_y in R^* such that $r_x \leq r_y$. This implies that $\mathrm{CAN}\{r_x, r_y\} = a_y$; that is, CAN selects the action from r_y. Therefore, an equivalent policy using FMR, which chooses the action from the rule having the greatest priority, must assign a higher priority to r_y. In general, greatest rules according to the order in R^* will have greatest priority, for instance, in case of least upper bounds

$$\pi(\operatorname*{lub}_{R^*}\{r_1, r_2\}) < \pi(r_1) \wedge \pi(\operatorname*{lub}_{R^*}\{r_1, r_2\}) < \pi(r_2).$$

The proposed algorithm relies on a modified breadth-first backward traversal in $G(p)$ to analyze and discard all the rules that are unnecessary for an FMR representation. The rules to discard are selected on the base that the condition clause of a rule in the semi-lattice is strictly included in the condition clause of its parents and ancestors. However, some caution is needed. Let us introduce in $G(p)$ the concept of *maximal domain* $\mathcal{M}(r)$ associated to a rule r, named the base rule.

Definition 11 (Maximal Domain): Given the base rule $r \in G(p)$, the maximal domain $\mathcal{M}(r)$ is the set built according to the following recursive procedure.

1) (*base clause*) $r \in \mathcal{M}(r)$;
2) (*inductive clause*) if $r'' \in G(p)$ and $r' \in \mathcal{M}(r)$ are such that $r'' \prec r'$, $a_{r''} = a_r$ and all the rules in every path between r'' and r enforce the same action a_r, then $r'' \in \mathcal{M}(r)$.

The inductive clause states that an ancestor can substitute r only if it enforces the same action and all its paths to r include only rules with the same action can substitute r. These ancestors are suitable because they are never overridden by covering rules that enforce a different action. A maximal domain is therefore a subgraph of $G(p)$. The proposed algorithm starts from the top element of $G(p)$ and continues building the maximal domains in the portion of $G(p)$ not yet included in some maximal domain. Finally, it outputs a sequence of maximal domains $\sigma = (M_1, M_2, \ldots, M_x)$. Fig. 5(a) shows a geometric representation of a policy whose canonical form semi-lattice is presented in Fig. 5(b) with maximal domains represented using Venn diagrams. The algorithm starts from the top element $r_{1,2,3,4}$ and builds the maximal domain $M_1 = \mathcal{M}(r_{1,2,3,4}) = \{r_{1,2,3}, r_{1,2,4}, r_{1,2}, r_{1,3}, r_{1,4}, r_3\}$. The rule r_4 has not been included as it enforces a different action. The next base rule is r_4, the only rule not yet processed, and its maximal domain includes only r_4, i.e., $M_2 = \mathcal{M}(r_4) = \{r_4\}$. Considering the case in Fig. 5(d) and (e), the algorithm starts in $r_{1,2,3,4}$ and builds the maximal domain $M_1 = \mathcal{M}(r_{1,2,3,4}) = \{r_{1,2,3}, r_{1,2,4}, r_{1,2}, r_{1,3}, r_3\}$. The rules $r_{1,4}$ and r_4 have not been included as they enforce a different action, and r_1 is not in M_1 because the path $(r_1, r_{1,4}, r_{1,2,4}, r_{1,2,3,4})$ includes $r_{1,4}$. The next base rule is $r_{1,4}$, therefore $M_2 = \mathcal{M}(r_{1,4}) = \{r_4, r_{1,4}\}$. Finally, $M_3 = \mathcal{M}(r_1) = \{r_1\}$. We name reduced maximal domain the subset $M^* \subseteq M$ that contains the rules whose ancestors, according to the order defined in R^*, are not in M. The rules in M^* are the *minimum* ones. The important property of rules in a reduced maximal domain is that all the condition clauses of other rules in M are subsets of at least one of them. For instance, in Fig. 5(b), the rules in the reduced maximal domains (represented with a thick border) are $M_1^* = \{r_1, r_3\}$, and $M_2^* = \{r_4\}$, while in Fig. 5(e) are $M_1^* = \{r_3, r_{1,2}\}$, $M_2^* = \{r_4\}$, and $M_3^* = \{r_1\}$. The rule set R_{FMR} will be generated as the union of all the rules in the reduced maximal domains obtained from σ

$$R_{\mathtt{FMR}} = M_1^* \cup M_2^* \cup \cdots \cup M_x^*.$$

Priorities are then assigned according to the order in σ. In practice, we have that $\forall r_i \in M_i^*$ and $\forall r_j \in M_j^*$, $i < j$ implies that $\pi(r_i) < \pi(r_j)$ (see Appendix B). Since all rules in a reduced maximal domain enforce the same action, the relative priorities are not important. In Fig. 5(e), the assignments $\pi(r_3) = 1$ and viceversa are both valid. Tables 5(c) and (f) show the equivalent FMR policy obtained from the example policies.

Algorithm 1 formally presents the FMR-morphism. It first empties the processed structure, used to store the rules that have already been included in some maximal domain (line 1). The algorithm starts from the top vertex T to calculate the maximal domain $M_1 = \mathcal{M}(T)$. If T has an empty condition clause, it will never match a packet, therefore the algorithm selects all its parents and starts calculating the maximal domains. This is done by initializing the ready structure that contains the rules in R^* for which it is allowed to calculate the maximal domain

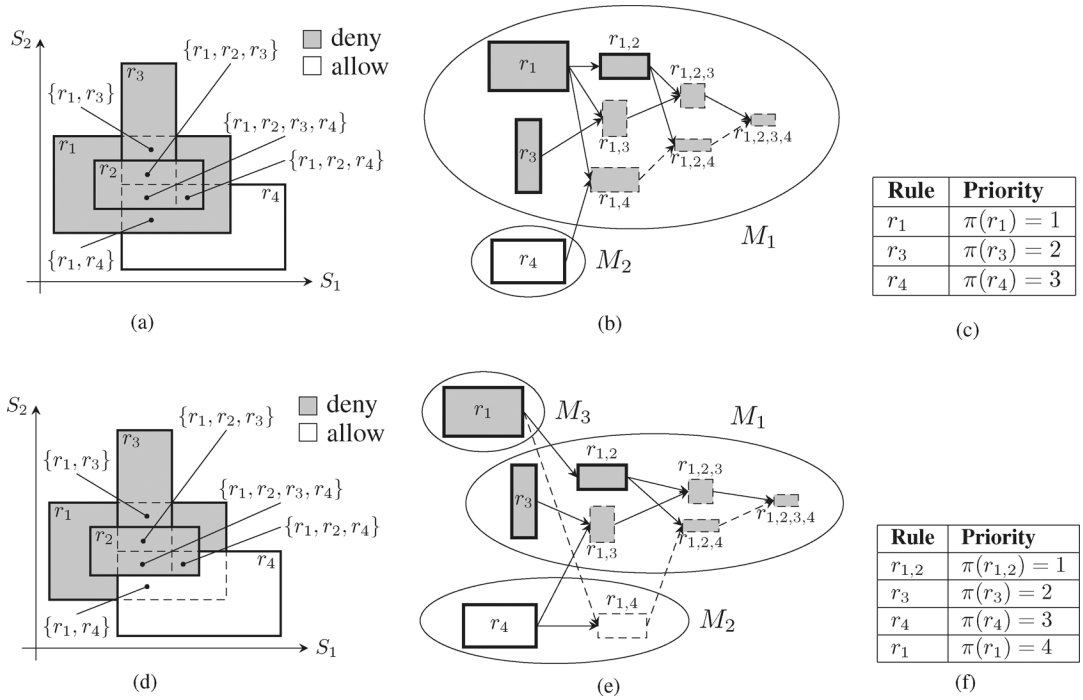

Fig. 5. Use of the proposed FMR-morphism to two sample policies. (a) Example policy 1: graphical representation and results of match function. (b) Example policy 1: the cover graph. (c) Example policy 1: exported rules. (d) Example policy 2: graphical representation and results of match function. (e) Example policy 2: the cover graph. (f) Example policy 2: exported rules.

Algorithm 1: The FMR-morphism

```
1:  processed ← ∅
2:  if the condition clause of T is not empty then
3:      ready ← T
4:  else
5:      ready ← parent Of(T)
6:  end if
7:  priority ← 1
8:  while ready ≠ ∅ do
9:      r ← selectFrom(ready)
10:     M ← maxdom(r)
11:     M* ← reduce(M)
12:     R_FMR ← M*
13:     for all x ∈ M* do
14:         π ←put (x, priority)
15:         priority ← priority + 1
16:     end for
17:     processed ←add M
18:     update(ready, processed)
19: end while
20: return R_FMR and π
```

(lines 2–6). The main cycle iterates over the elements in ready (line 8). The next rule r to process is selected using `selectFrom()` (line 9). This can be a random choice as all the rules in ready are selectable since the algorithm guarantees that the composition of rules in ready have been already included in a previously calculated maximal domain.

The function `maxdom()` (line 10) implements the recursive function presented in Definition 11. It calculates and returns the maximal domain M associated to r. The `reduce()` function (line 11) returns the reduced maximal domain M^* by walking in the subgraph M of $G(p)$ to identify rules whose ancestors are not in M. Rules in M^* are then included in the target rule set $R_{\texttt{FMR}}$ (line 12). All rules in the current M^* are associated to priorities using π, a hash map that associates rules to integers (line 14). The next priority to assign is managed using the variable priority (line 7 and line 15). Afterwards, both processed (line 17) and ready must be updated resorting to the `update()` function that selects all the rules whose covers are all in processed (line 18). The algorithm terminates when all the rules in R^* have been included in processed.

The LMR-morphism uses the FMR-morphism to calculate the $R_{\texttt{FMR}}$ and π. Then, it is necessary to reassign the priorities. If N is the cardinality of $R_{\texttt{FMR}}$, the new assignment is

for all $r \in R_{\texttt{FMR}}$ **do**
  $\pi(r) \leftarrow N - \pi(r) + 1$
end for

Afterwards, π maps the relations such that $(R_{\texttt{FMR}}, \texttt{LMR}, \{\pi\}, d)$ is equivalent to the initial canonical form.

In Appendix-B, we prove the correctness of the FMR-morphism (Theorem 2), and we sketch also the proof of termination of the Algorithm 1. Thus, we can state that *a morphism from a policy expressed in canonical form to a policy described using the FMR or LMR strategy exists*. Moreover, since every policy can be represented in canonical form, we can infer that every policy defined using our model can be represented through another policy using the FMR or LMR strategy.

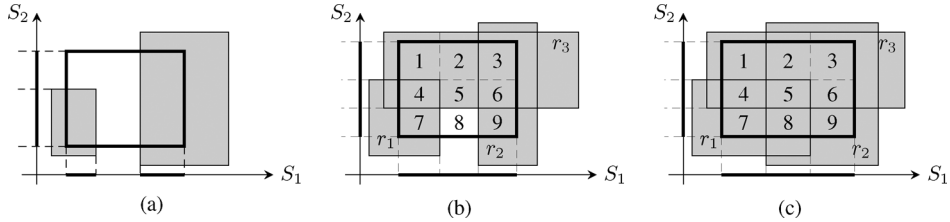

Fig. 6. Graphical representation of the techniques resorting to the canonical form to evaluate the function eff_p. (a) Simple verification: $\mathrm{eff}_p(r) \neq \emptyset$. S_1 and S_2 are not fully covered. (b) Full verification: $\mathrm{eff}_p(r) \neq \emptyset$, the rule is necessary because of box number 8. (c) Full verification:$\mathrm{eff}_p(r) = \emptyset$.

C. Using the Canonical Form to Detect the General Anomalies

The semi-lattice and its representation as a cover graph enable calculation of eff_p without resorting to set union. Using the semi-lattice, $\mathrm{eff}_p(r)$ can be evaluated only considering the covers of r, formally, if $r' = (c', a')$

$$\mathrm{eff}_p(r) = \emptyset \iff \bigcup_{r \prec r'} c' \cap c = c.$$

The verification of this condition is done in two steps: the simple and the full verification. The *simple verification* consists in calculating the union of the conditions of covering rules in each selector. If at least one union is not equal to the corresponding condition in r, then we deduce that $\mathrm{eff}_p(r) \neq \emptyset$ [see Fig. 6(a)]; otherwise we use the full verification.

The *full verification* consists in enumerating all the boxes obtained from the intersection of the condition clause of r with the condition clauses of its covers. For each box, we verify the sub-optimality property, that is, $\mathfrak{R}(\mathrm{match}_R(x)) = \mathfrak{R}(\mathrm{match}_R(x) \setminus \{r\})$ [see Fig. 6(c) and (b)]. To this purpose, it is enough to take all the compositions of r, that is the sub-semi-lattice $\{r' \in R^* \mid r < r'\}$, whose elements can be written as $r \cdot (r_l \cdot r_m \cdot \ldots)$, and to check whether $\mathfrak{R}(\{r\} \cup \{r_l, r_m, \ldots\}) = \mathfrak{R}(\{r_l, r_m, \ldots\})$ is satisfied or not. If for at least one composition the previous property is not satisfied, then r is necessary; otherwise, r is hidden. The type of general anomaly (redundant or shadow) depends on the action clause of the rules in the sub-semi-lattice as shown in Section IV-D.

VI. Implementation

The policy model, anomaly detection, and translation process have been implemented in an extensible and modular tool whose purpose is to guide administrators in analyzing, validating, and exporting rule-based policies. The tool is written in Java, and its architecture is shown in Fig. 7. The Policy Model module permits the definition of generic policies expressed according to our model. It includes the classes that represent policy, rules, actions, condition clauses and selectors, and the logic for performing set-based operations on condition clauses. Additionally, it provides the resolution mechanisms and the means to define and associate external data.

The Format Translator module loads the policy and instantiates the model. By using a modular architecture, the policy can be specified using an implementable language or an *ad hoc* syntax, thanks to specific parsers able to instantiate the policy model. Additionally, policies can be exported to different languages for direct use by enforcement devices.

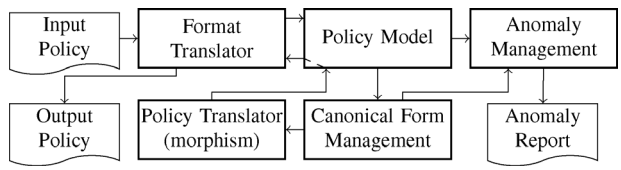

Fig. 7. Tool's architecture.

The Canonical Form Management module translates the policy into canonical form and generates the associated semi-lattice. Both the policy and semi-lattice of its canonical form are used by the anomaly detection and classification functionalities provided by the Anomaly Management module. The anomalies are reported to a file. The Policy Translator transforms the policy according to the desired target policy parameters, that is, the default action, resolution strategy, and necessary external data.

The tool already includes the most used resolution strategies and the ones presented in this paper. A custom resolution strategy can be written by implementing the following interface:

```
public interface ResolutionStrategy ⟨T extends Rule⟩ {
    public T composeRules(T r1, T r2);
    public Action composeActions(T r1, T r2);
    public T composeRules(Collection ⟨T⟩ rules);
    public Action composeActions(Collection ⟨T⟩ rules);
    public ResolutionStrategy ⟨T⟩ cloneResolutionStrategy();
}
```

The design and testing of the tool required careful planning since, theoretically, the worst case for the number of rules of the canonical form can be exponential in the number of rules (due to the cardinality of the closure $\overline{R}$). Thus, the focus was on the design of optimized data structures for fast rule lookup and composition. However, while the worst case is independent of the actual rule sets, real cases are strongly dependent on the way rules are specified. Statistical considerations and the analysis of existing rule sets confirmed that practical cases are very far from the worst-case scenario that occurs if all the rules intersect with each other. Moreover, in the worst case, many compositions would have the same condition clause and would be removed from the canonical form. Therefore, further studies are needed to evaluate the maximum number of rules in a canonical form where rules with the same condition clause are eliminated.

Taylor and Lunteren [27]–[29] have shown how far the typical firewall scenarios are from the worst-case one.

- The maximum number of rules that a packet can match is very small, typically less than five.

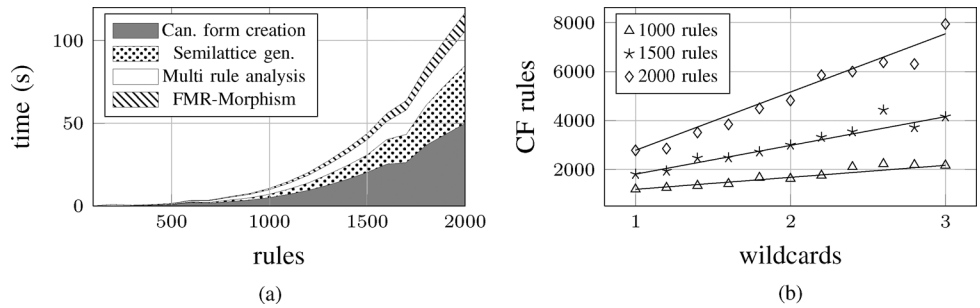

Fig. 8. Tool performance: testing results. (a) Elaboration time depending on the size of the rule set. (b) Canonical form rules depending on wildcards.

- The protocol field is restricted to a small set of values: TCP, UDP, ICMP, (E)IGRP, GRE, or any protocol. Moreover TCP, UDP, and "any" cover almost all cases.
- TCP ports are usually specified as wildcard, high range (≥ 1024), low range (< 1024), or exact matching. The value distribution depends on the field type, i.e., source ports are seldom part of the condition, and when they are, they rarely require a fixed specific value.
- The distribution of wildcards depends on the field (e.g., IP source addresses contain more wildcards than the destination ones).

This suggests that the closure will rarely contain rules composed of more than five rules. Therefore, the depth of $G(p)$ will be close to five, independently of the rule number. Additionally, Al-Shaer *et al.* analyzed man-written rule sets and noticed that, in the worst case (a beginner administrator), at most 25% of the rules present anomalies [2]. In our model, this means that no more than 25% of the rules have at least one (nonempty) composition, while 75% of the rules do not generate compositions. It should be noted that the anomaly percentage strongly decreases to 8% in the case of expert administrators. Consequently, the percentage of compositions involving three or more rules will be very limited. This information also confirms that, in real cases, the closure calculation stops after a few compositions since the majority of condition clauses do not intersect. Our experiments confirmed that the number of rules in R^* rises in the same way as a (low-degree) polynomial. The default action also impacts over the specification as it is common to write rules only for those parts of the selection space where an action different from the default one is desired. This practice increases the chance of unnecessary rules.

We tested our implementation in four steps: random generation of 5-tuple rules according to statistical information, canonical form generation, anomaly report, and translation. The tests used a standard PC (Intel Core Duo 2.33 GHz with 2 GB of RAM).

Rule sets were generated according to Taylor's statistical data, with policies ranging from 50 to 2000 rules, in steps of 50. To reduce statistical deviation, we generated 100 rule sets for each size. A resolution strategy was then selected among the available ones. The default action was fixed to Deny.

Additionally, another parameter was varied: the average number of wildcards per rule (from 1 to 3 by 0.2 on a total of five fields). The usage of wildcards affects the average number of intersecting rules, while Taylor's statistical data is preserved: The more the wildcards, the more the intersections. This increases the number of rules in the canonical form and therefore the computational time.

The tool performance was satisfactory: In the worst case, it took only 120 s to elaborate 2000 rules and to translate them using the `FMR`-morphism. Therefore, it is suitable for real cases. We noticed that the performance was independent of the resolution strategy, that is, the cost of resolution did not considerably affect the computation. There were less exported rules than the original ones. As mentioned before, this reduction is strongly dependent on the rule set. The results of these tests are presented in Fig. 8(a), which plots the average computing time, and Fig. 8(b), which shows the average number of rules in the canonical form.

VII. Conclusion and Future Work

The work presented here focuses on access control network-level policies, providing optimization and conflict resolution via set-based representation and manipulation through special procedures (the morphisms). The model and procedure have been implemented in a tool used to give an experimental demonstration of its applicability to real cases. Our ambition is to develop an integrated development environment (IDE) for defining and verifying policies, also providing a language to define the resolution strategies, which currently need to be written as Java classes.

We are confident that this work can be applied equally well to other access-control scenarios, as long as they respect some principles: 1) they are rule-based 2) their conditions and actions have a finite set of values; and 3) selectors permit the use of intersection. These principles apply to some important cases such as identity- and role-based access control at application and OS level. By removing the limitation of the condition clause being a hyper-rectangle, we aim at extending our approach to support even more complex scenarios arising at higher abstraction levels. Moreover, we aim to extend our model to support contexts where a history-based analysis is necessary to identify dynamic conflicts.

The model can also be proficiently used in other scenarios. We are currently exploring two areas: rule number optimization and analysis of distributed scenarios. The present version of the anomaly discovery tool deals with rules written by the administrators, which is in a human-readable form. This is useful for showing the anomalies in a user-friendly way, but unnecessary when translating them to machine-readable form. More compact and optimized representation of rule-based policies can be obtained if rules can be freely manipulated without links to their

original form. However, our preliminary study shows that this task is computationally heavy and thus feasible only for offline policy optimization. Finally, by considering also topological information, we aim to detect anomalies and optimize policies in distributed systems.

APPENDIX

A. Equivalence of the Canonical Form

In this section, we first prove Theorem 1 stating that the canonical form obtained using the construction in Section V-A is equivalent to the original policy.

Lemma 1: Let $r_1, r_2 \in R^*$, then the following holds:

$$\underset{R^*}{\text{lub}}\{r_1, r_2\} = r_1 \cdot r_2 = (c_1 \cap c_2, a_{1,2}).$$

Note that $a_{1,2}$ depends not only on r_1, r_2, but also on all the rules in $\overline{R}$ with the same condition clause (see Section V-A). First of all, R^* contains (by construction) a rule with the condition clause $c_1 \cap c_2$, thus $r_1 \cdot r_2 \in R^*$. Since $c_1 \cap c_2 = \underset{\mathfrak{S}}{\text{lub}}\{c_1, c_2\}$, from the definition of the order relation in R^*, we derive $r_1 \cdot r_2 = \underset{R^*}{\text{lub}}\{r_1, r_2\}$.

Lemma 2: For all x, $\text{match}_{R^*}(x)$ is an R^* sub-semi-lattice.

Let $r_1, r_2 \in \text{match}_{R^*}(x) \subseteq R^*$, according to Lemma 1, $\underset{R^*}{\text{lub}}\{r_1, r_2\} = r_1 \cdot r_2 = (c_1 \cap c_2, a_{1,2})$. Additionally, since $x \in c_1$ and $x \in c_2$, we have that $x \in c_1 \cap c_2$, thus $r_1 \cdot r_2 \in \text{match}_{R^*}(x)$. This proves the closure and the lemma and enables the usage of the lub for the rules selected by match_{R^*}.

Lemma 3: If $\text{match}_R(x) = \{r_l, r_m, \ldots, r_z\}$, then $r_{\text{lub}} = \underset{R^*}{\text{lub}}\{\text{match}_{R^*}(x)\} = r_l \cdot r_m \cdot \ldots \cdot r_z$.

This is evident because

$$r_l \cdot r_m \cdot \ldots \cdot r_z = (c_l \cap c_m \cap \cdots \cap c_z, \mathfrak{R}(\text{match}_R(x)))$$

and $c_l \cap c_m \cap \cdots \cap c_z$ is the least upper bound in $\mathfrak{S}$ of conditions $\{c_l, c_m, \ldots, c_z\}$. Additionally, by construction, any other composition in $\overline{R}$ with the condition clause $c_l \cap c_m \cap \cdots \cap c_z$, has been excluded from R^*.

Theorem 1: $(R^*, \text{CAN}, \emptyset, d)$ is equivalent to $(R, \mathfrak{R}_E, E, d)$.

Proof: It is enough to show that

$$\forall x, \mathfrak{R}(\text{match}_R(x)) = \text{CAN}(\text{match}_{R^*}(x)).$$

By combining the definitions of CAN and Lemma 3, we obtain that (α is the function returning the action clause)

$$\begin{aligned}
&\text{CAN}(\text{match}_{R^*}(x)) \\
&\quad = \alpha\left(\underset{R^*}{\text{lub}}\{\text{match}_{R^*}(x)\}\right) = \alpha(r_l \cdot r_m \cdot \cdots \cdot r_z) \\
&\quad = \alpha((c_l \cap c_m \cap \cdots \cap c_z, \mathfrak{R}(\text{match}_R(x)))) \\
&\quad = \mathfrak{R}(\text{match}_R(x))
\end{aligned}$$

and this proves Theorem 1.

B. Correctness of the FMR-Morphism

In this section, we prove Theorem 2 stating that the policy obtained by means of the FMR-morphism in Section V-B is equivalent to the canonical form from which it was derived.

Lemma 4: $\text{match}_{R_{\text{FMR}}}(x) \subseteq \text{match}_{R^*}(x)$.

This is evident because the algorithm selects a subset of the rules in R^*. Thus, by means of Lemma 2 and the existence of a top element, all the matching rules in R_{FMR}, when considered in R^*, are less than or equal to $r_{\text{lub}} = \underset{R^*}{\text{lub}}\{\text{match}_{R^*}(x)\}$.

Lemma 5: Let $M_1^* = \mathcal{M}^*(r_1)$ and $M_2^* = \mathcal{M}^*(r_2)$ be two disjoint reduced maximal domains associated to rules r_1 and r_2 such that $r_1 < r_2$. Then, for any $t_1 \in M_1^*$ and $t_2 \in M_2^*$, we have $\pi(t_1) > \pi(t_2)$.

The FMR-morphism selects M_2^* before M_1^*, and this implies that $\forall r_{1i} \in M_1^*$ and $\forall r_{2j} \in M_2^*$ their priorities are ordered as $\pi(r_{1i}) > \pi(r_{2j})$. A simple extension of Lemma 5 is the following lemma.

Lemma 6: Let $t \in R^*$. All the rules $r < t$ belong to the same maximal domain of t or maximal domains with base rule t' such that $t' < t$.

Lemma 7: Let r_1, r_2 be two rules in R^* such that $r_1 < r_2$. If both are also in R_{FMR}, then $\pi(r_1) > \pi(r_2)$.

The proof is simple. First, since $r_1, r_2 \in R_{\texttt{FMR}}$, r_1 and r_2 do not belong to the same maximal domain because the FMR-morphism would only have selected the minimum when calculating the reduced maximal domain. Then, let $M_1^* = \mathcal{M}^*(r_1)$ and $M_2^* = \mathcal{M}^*(r_2)$. If $r_1 \in M_1$ and $r_2 \in M_2$ and $M_1 \neq M_2$, then $\pi(r_1) > \pi(r_2)$ holds due to Lemma 5.

Lemma 8: Let $\text{match}_{R^*}(x) = \{r_l, r_m, \ldots, r_x\}$ and $r_{\text{lub}} = \underset{R^*}{\text{lub}}\{\text{match}_{R^*}(x)\}$. If M_{lub} is maximal domain containing r_{lub} and M_{lub}^* the corresponding reduced maximal domain, then rules in $r \in \text{match}_{R_{\text{FMR}}}(x) \cap M_{\text{lub}}^*$ (that is, the rules in the maximal domain containing r_{lub} selected by the FMR-morphism) have higher priority than rules in $\text{match}_{R_{\text{FMR}}}(x) \setminus M_{\text{lub}}^*$ (that is, the rules selected by the FMR-morphism that are not in the maximal domain containing r_{lub}).

In other words, rules in the maximal domain containing r_{lub} selected by the FMR-morphism have higher priority than rules selected by the FMR-morphism that are not in the maximal domain containing r_{lub}. This lemma is proved by combining Lemmas 6 and 7. In fact, rules with greater priority are those in the same maximal domain as r_{lub} because every rule in $\text{match}_{R_{\text{FMR}}}(x)$ is less than r_{lub}.

Theorem 2: $(R_{\text{FMR}}, \text{FMR}, \{\pi\}, d)$ obtained by executing the FMR-morphism is equivalent to $(R^*, \text{CAN}, \emptyset, d)$.

Proof: Theorem 2 is proved if we show that

$$\begin{aligned}
&\left(\text{FMR} \circ \varepsilon_{\{\pi\}}\right)(\text{match}_{R_{\text{FMR}}}(x)) = a_{\text{lub}} \\
&\qquad = \alpha\left(\underset{R^*}{\text{lub}}\{\text{match}_{R^*}(x)\}\right) = \text{CAN}(\text{match}_{R^*}(x)).
\end{aligned}$$

If $r_{\text{lub}} \in R_{\text{FMR}}$, then Lemma 7 guarantees that all rules r in $\text{match}_{R_{\text{FMR}}}(x)$ have greater priority than r_{lub}, that is $\pi(r_{\text{lub}}) < \pi(r)$. Thus, the $\text{FMR} \circ \varepsilon_{\{\pi\}}$ strategy will select the action from r_{lub}. If $r_{\text{lub}} \notin R_{\text{FMR}}$, by Lemma 8 the rules with greater priority are the ones in the same maximal domain as r_{lub}. Due to the property that rules in the same maximal domain have the same action clause, even in this case the $\text{FMR} \circ \varepsilon_{\{\pi\}}$ strategy will select the action from r_{lub}. This completes the proof of Theorem 2.

We outline here the proof that the FMR-algorithm terminates only when all the rules in R^* have been processed. This is true if ready $\neq \emptyset$ unless all the rules in R^* have been processed. In general, after every iteration of the main cycle, processed

is a connected subgraph of $G(p)$ (including T), therefore also its complement $R^* \setminus \mathsf{processed}$ is connected. If W is the set of all the elements not in processed with at least one cover in processed and ready is the set of the elements not in processed having all the covers in processed

$$W = \{r \in R^* \mid \exists r', r \prec r' \wedge r' \in \mathsf{processed}\}$$
$$\mathsf{ready} = \{r \in R^* \mid \forall r', r \prec r' \wedge r' \in \mathsf{processed}\}.$$

It is evident that $\mathsf{ready} \subseteq W$, we must prove that $\mathsf{ready} \neq \emptyset$ unless all the rules have been processed. Since $(R^* \setminus \mathsf{processed})$ is connected, a maximal element z in W exists. Additionally, $\exists r', z \prec r' \wedge r' \in \mathsf{processed}$, but since z is maximal, all its covers are not in W, thus they are in processed. This proves that $z \in W \cap \mathsf{ready}$, thus $z \in \mathsf{ready}$.

References

[1] A. Westerinen, "Terminology for policy-based management," RFC-3198, Nov. 2001.

[2] E. Al-Shaer and H. Hamed, "Modeling and management of firewall policies," *IEEE Trans. Netw. Service Manage.*, vol. 1, no. 1, pp. 2–10, Apr. 2004.

[3] E. Al-Shaer, H. Hamed, R. Boutaba, and M. Hasan, "Conflict classification and analysis of distributed firewall policies," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 10, pp. 2069–2084, Oct. 2005.

[4] H. Hamed and E. Al-Shaer, "Taxonomy of conflicts in network security policies," *IEEE Commun. Mag.*, vol. 44, no. 3, pp. 134–141, Mar. 2006.

[5] C. Basile and A. Lioy, "Towards an algebraic approach to solve policy conflicts," in *Proc. WOLFASI*, Turku, Finland, July 2004, pp. 319–338.

[6] C. Basile, A. Cappadonia, and A. Lioy, "Geometric interpretation of policy specification," in *Proc. IEEE Policy*, New York, NY, Jun. 2008, pp. 78–81.

[7] M. Benelbahri and A. Bouhoula, "Tuple based approach for anomalies detection within firewall filtering rules," in *Proc. IEEE ISCC*, Aveiro, Portugal, Jul. 2007, pp. 63–70.

[8] S. Thanasegaran, Y. Yin, Y. Tateiwa, Y. Katayama, and N. Takahashi, "A topological approach to detect conflicts in firewall policies," in *Proc. IEEE IPDPS*, Rome, Italy, May 2009, pp. 1–7.

[9] S. Ferraresi, S. Pesic, L. Trazza, and A. Baiocchi, "Automatic conflict analysis and resolution of traffic filtering policy for firewall and security gateway," in *Proc. IEEE ICC*, Glasgow, Scotland, 2007, pp. 1304–1310.

[10] M. Rezvani and R. Aryan, "Analyzing and resolving anomalies in firewall security policies based on propositional logic," in *Proc. IEEE INMIC*, Islamabad, Pakistan, 2009, pp. 1–7.

[11] J. Zao, "Semantic model for IPSec policy interaction," Internet Draft, Mar. 2000.

[12] Z. Fu, S. F. Wu, H. Huang, K. Loh, F. Gong, I. Baldine, and C. Xu, "IPSec/VPN security policy: Correctness, conflict detection and resolution," in *Proc. IEEE Policy*, Bristol, U.K., 2001, pp. 39–56.

[13] Z. Li, X. Cui, and L. Chen, "Analysis and classification of IPSec security policy conflicts," in *Proc. FCST*, Aizu, Japan, Nov. 2006, pp. 83–88.

[14] A. K. Bandara, E. C. Lupu, A. Russo, N. Dulay, M. Sloman, P. Flegkas, M. Charalambides, and G. Pavlou, "Policy refinement for IP differentiated services quality of service management," *IEEE Trans. Netw. Service Manage.*, vol. 3, no. 2, pp. 2–13, Apr. 2006.

[15] J. D. Moffett and M. S. Sloman, "The representation of policies as system objects," in *Proc. SIGOIS*, Atlanta, GA, 1991, pp. 171–184.

[16] J. D. Moffett and M. S. Sloman, "Policy hierarchies for distributed system management," *IEEE J. Sel. Areas Commun.*, vol. 11, no. 9, pp. 1404–1414, Nov. 1993.

[17] J. D. Moffett and M. S. Sloman, "Policy conflict analysis in distributed system management," *J. Org. Comput.*, vol. 4, no. 1, pp. 1–22, 1993.

[18] E. Lupu and M. Sloman, "Conflicts in policy-based distributed system management," *IEEE Trans. Softw. Eng.*, vol. 25, no. 6, pp. 852–869, Nov. 1999.

[19] M. Sloman, "Policy driven management for distributed systems," *J. Netw. Syst. Manage.*, vol. 2, no. 4, pp. 333–360, 1994.

[20] N. Dunlop, J. Indulska, and K. A. Raymond, "Dynamic policy model for large evolving enterprises," in *Proc. EDOC*, Seattle, WA, Sep. 2001, pp. 193–197.

[21] K. A. R. N. Dunlop and J. Indulska, "A formal specification of conflicts in dynamic policy-based management system," CRC for Enterprise Distributed Systems, Univ. Queensland, Brisbane, Australia, DSTC Tech. Rep., Aug. 2001.

[22] G. Szasz, *Introduction to Lattice Theory*. New York: Academic, 1963.

[23] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen, "Policy core information model," RFC 3060, Feb. 2001.

[24] "Precedence rules used to resolve access conflicts at a target," IBM Lotus Domino and Notes Information Centre, 2011 [Online]. Available: http://publib.boulder.ibm.com/infocenter/domhelp/v8r0/index.jsp

[25] C. Basile, A. Cappadonia, and A. Lioy, "Algebraic models to detect and solve policy conflicts," in *Proc. MMM-ACNS*, St. Petersburg, Russia, 2007, pp. 242–247.

[26] G. Birkhoff, *Lattice Theory*. Providence, RI: Amer. Math. Soc., 1967.

[27] D. Taylor and J. Turner, "Scalable packet classification using distributed crossproducting of field labels," Dept. Comput. Sci. Eng., Washington Univ., Washington, DC, Tech. Rep. WUCSE-2004-38, 2004.

[28] D. Taylor, "Survey and taxonomy of packet classification techniques," *Comput. Surveys*, vol. 37, no. 3, pp. 238–275, 2005.

[29] J. van Lunteren and T. Engbersen, "Fast and scalable packet classification," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 4, pp. 560–571, May 2003.

Cataldo Basile received the M.Sc. (*summa cum laude*) and Ph.D. degrees in computer engineering from Politecnico di Torino, Turin, Italy, in 2001 and 2005, respectively.

He is currently a Research Assistant with Politecnico di Torino. His research is concerned with policy-based management of security in networked environments, policy refinement, general models for detection, resolution and reconciliation of specification conflicts, and software security.

Alberto Cappadonia received the M.Sc. degree in telematics engineering and Ph.D. degree in computer engineering from Politecnico di Torino, Turin, Italy, in 2006 and 2011, respectively.

His research interests are in network security and policy-based management systems, automatic refinement and translation of policy for devices, and general models for policy analysis and debug.

Antonio Lioy (M'89) received the M.Sc. degree in electronic engineering (*summa cum laude*) and the Ph.D. degree in computer engineering from Politecnico di Torino, Turin, Italy, in 1982 and 1987, respectively.

He is a Full Professor with Politecnico di Torino, where he leads the TORSEC research group active in information system security. His research interests include network security, public-key infrastracture (PKI), and policy-based system protection.