Automatic Conflict Analysis and Resolution of Traffic Filtering Policy for Firewall and Security Gateway

Simone Ferraresi∗†, Stefano Pesic∗, Livia Trazza∗, Andrea Baiocchi†

∗ELSAG S.p.A. - Via Naide, 43 - 00155 Roma, Italy - e-mail: simone.ferraresi@elsag.it, stefano.pesic@gmail.com, liviabil@tiscali.it
†INFOCOM Dept.- Univ. of Roma “La Sapienza” - Via Eudossiana, 18 - 00184 Roma, Italy - e-mail: andrea.baiocchi@uniroma1.it

Abstract—Firewalls and Security Gateways are core elements in network security infrastructure. As networks and services become more complex, managing access-list rules becomes an error-prone task. Conflicts in a policy can cause holes in security, and can often be hard to find while performing only visual or manual inspection. First, we have defined a methodology to systematically classify the severity of rule conflicts; secondly, we have proposed two different solutions to automatically resolve conflicts in a firewall. For one of them we found an algebraic proof of the existence of the solution and the convergence of the algorithm, and then we have made a software implementation to test it.

I. INTRODUCTION

A key feature of secure systems, including networked ones, is the management of security policies, from those at high level down to the platform specific implementation. Security policy defines constraints, limitations and authorization on data handling and communications. In a network environment many problems may also arise due to the inconsistency of policies implemented by firewalls and secure access gateways interconnected over an insecure network (e.g. in a S-VPN context). As distributed secure systems increase in complexity, policy configuration and maintenance tasks become increasingly prone to errors.

A single firewall system, deployed in a complex context, checks many different data flows, grants access to several networks and services and often cooperates with S-VPN systems, IDS, IPS, RAS and all the other security systems. Furthermore, during firewall lifetime the context can change frequently, generating many policy updates.

Today policy definition is often performed manually. This process is error prone and can give rise to holes in security of the entire system, which are hardly detectable before suffering an attack. For this reason a verification phase should be performed when a policy is defined and after any modification or integration.

We assume that policies are formally stated according to a well defined formal language, so that they can be reduced to an ordered list of predicates of the form $C \rightarrow A$, where $C$ is a condition and $A$ is an action. We refer to predicates implementing security policies as rules. For firewalls the condition of a filtering rule is composed of five selectors:

\[
<\text{protocol}><\text{src_ip}><\text{src_port}><\text{dst_ip}><\text{dst_port}>
\]

The action that could be performed on the packet is allow or deny. Conditions are checked on each packet flowing through the firewall, applying it exclusively the action required by the first matching rule.

In general a conflict occurs when desired effects of a policy are ambiguous or not clear. In this paper we focus our attention on automatic conflict detection and resolution on a policy implemented in a stand-alone firewall.

The paper is organized as follows. In Section II we give a synthetic overview of previous related works. Then, Section III we briefly present logical relationships between rules. In Section IV we present our formalization of severity classification for conflicts. In Section V two algorithms are defined for automatic conflict resolution. In Section VI, we give a summary of the software implementation. Finally, in Section VII, we give our conclusions and plans for future work.

II. RELATED WORK

Policy consistency has been the subject of a lot of attention from the research community. In [1] the Policy Core Information Model (PCIM) is described as an object-oriented model for representing policy information as extensions to the Common Information Model (CIM) activity within the Distributed Management Task Force (DMTF). The definition of policy and policy rule presented in PCIM and its extension shown in RFC-3198 [2] gave to the author of [3] the starting point to refine these concepts in a way useful for a formal approach. A policy can be defined in two ways, “a definite goal” and “a set of rules to administer, manage, and control access to network resources”. Furthermore the “policy is applied using a set of policy rules”. The policy rules or simply the rules can be described as “the binding of a set of actions to a set of conditions - where the conditions are evaluated to determine whether the actions are performed”. A policy conflict “occurs when the actions of two rules (that are both satisfied simultaneously) contradict each other” and it creates a problem because “the entity implementing the policy would not be able to determine which action to perform”. Furthermore, the same authors state that “the implementers of
policy systems must provide conflict detection and avoidance or resolution mechanisms to prevent this situation”.

The purpose of our paper is to provide resolution mechanism and algorithm on the specific field of security policy defined by access-list rules, thus our field of investigation is the process of selecting a resolution procedure when conflicts may arise in security policy. The most significant contribution to this subject are [4][5][6][7]. In [7] the authors only attempt to detect if firewall rules are correlated to each other, while in [4][5][6] a set of techniques and algorithms are defined to discover all of the possible policy conflicts. However, no automatic conflict solution algorithm is provided.

Original contributions of our work are the definition of a general methodology for firewall policy conflict classification and detection, and exploitation of this methodology as a basis to define automatic conflict resolution algorithms. The output of such algorithms is a corrected policy file plus possibly feedback to the security manager to fix detected conflicts that cannot be decided automatically. We mathematically prove existence of a conflict free rule list and we can also prove convergence of the proposed algorithm to such a solution.

III. MODELING OF RULES RELATION

To be able to introduce firewall policy conflict analysis and resolution it is useful to define all the relations that may tie a couple of rules. These relations, as defined in [4], require a comparison between the network fields of filtering rules independently of the rule action. First we state formally what a rule is in our context.

Firewall policies as well as IPSec gateway policies specify a filtering condition C and an action A. The condition C aims at selecting those IP packets that the corresponding action applies to. Packet filtering is based on the values of five header fields: IP destination and source addresses (dst_ip, src_ip), destination and source port numbers (dst_port, src_port) and the IP header protocol type (prot). For each of these selectors we need to specify a value or a range of values. To each predicate listed in the policy file we associate a rule R defined as a five-tuple of selector variables ranges:

\[ R = \{ S_{prot}, S_{src_ip}, S_{src_port}, S_{dst_ip}, S_{dst_port} \} \]

where \( S_{sel} = \text{ANY} \) or \( S_{sel} = [sel_{min}, sel_{max}] \) for sel \( \in \{ \text{prot}, \text{src_ip}, \text{src_port}, \text{dst_ip}, \text{dst_port} \} \), ANY being a “don’t care” flag; equivalently we could replace ANY by the entire possible range of the corresponding selector. In the following we refer to the five selector with an integer variable \( i = 1, \ldots, 5 \); we denote the \( i \)-th selector range associated to a given rule \( R \) by \( R[i] \). We also define a set operator as \( \Diamond \in \{ \subseteq, \supseteq, = \} \); also \( A \Diamond B \) means that the set \( A \) is not a superset nor a subset nor the same as the set \( B \).

We can now state formally the basic relations among rules for the conflict analysis.

**Definition 1:** Rules \( R_x \) and \( R_y \) are Completely Disjoint if every field in \( R_x \) is not a subset of, nor a superset of, nor equal to the corresponding field in \( R_y \). Formally,

\[ R_x \cap \overline{R_y} \iff \forall i : R_x[i] \nsubseteq R_y[i] \]

where \( \Diamond \in \{ \subseteq, \supseteq, = \} \) and

\[ i \in F = \{ \text{prot, src_ip, src_port, dst_ip, dst_port} \} \]

**Definition 2:** Rules \( R_x \) and \( R_y \) are Exactly Matching if every field in \( R_x \) is equal to the corresponding field in \( R_y \). Formally,

\[ R_x \cap \overline{R_y} \iff \forall i : R_x[i] = R_y[i] \]

where \( i \in F \)

**Definition 3:** Rules \( R_x \) and \( R_y \) are Inclusively Matching if they do not exactly match and every field in \( R_x \) is a subset of or equal to the corresponding field in \( R_y \). \( R_x \) is called the subset match, while \( R_y \) is called the superset match. Formally,

\[ R_x \cap \overline{R_y} \iff \forall i : R_x[i] \subseteq R_y[i] \text{ and } \exists j : R_x[j] \supseteq R_y[j] \]

where \( i, j \in F \)

**Definition 4:** Rules \( R_x \) and \( R_y \) are Partially Matching if there is at least one field in \( R_x \) that is a subset of or a superset of or equal to the corresponding field in \( R_y \) and there is at least one field in \( R_x \) that is not a subset nor a superset, nor equal to the corresponding field in \( R_y \). Formally,

\[ R_x \cap \overline{R_y} \iff \exists i, j : R_x[i] \nsubseteq R_y[i] \text{ and } R_x[j] \nsubseteq R_y[j] \]

where \( i, j \in F, i \neq j \)

**Definition 5:** Rules \( R_x \) and \( R_y \) are Correlated if some fields in \( R_x \) are subsets of or equal to the corresponding fields in \( R_y \) and the rest of the fields in \( R_x \) are supersets of the corresponding fields in \( R_y \). Formally,

\[ R_x \cap \overline{R_y} \iff \forall i : R_x[i] \supseteq R_y[i] \text{ and } \exists i, j : R_x[i] \supseteq R_y[i] \text{ and } R_x[j] \supseteq R_y[j] \]

where \( i, j \in F, i \neq j \)

IV. CONFLICT ANALYSIS

To deal with the resolution of traffic filter policy conflicts we have to elaborate a new classification of them which is more selective than the one presented in [4]. This has been done by introducing the concept of conflict severity.

**Definition 6:** The severity of a conflict is defined as the rank of correlation between the presence of the conflict in the policy and the erroneous behaviour of the respective device. A device behaviour is considered erroneous when it does not correspond to the aim of the security manager. To univocally identify the aim of security manager it has been necessary to formulate a working hypothesis on his behaviour.

**Working Hypothesis.** The security manager inserts a rule in the policy because he wants to apply it to at least one packet.

Firewall policies are formalized into rules and rules are listed in an ordered file; for each packet the list is scanned and the
first matching rule is found (if any); the corresponding action is applied to the packet. Rules further down in the file are ignored for that packet. Let \( \omega \) be a positive integer denoting the position of a rule in this list: lower values of \( \omega \) mean higher priority. The position of rule \( R \) is denoted by \( \omega(R) \).

We now introduce the classification of conflicts according to severity levels:

**Exact Match:** A rule is in exact match with another one when the two rules are equal in all selectors independently of the value assumed in the action field. These two rules match the same traffic, on which, however, only the action of the rule with higher priority is performed. Formally,

\[
R_y \subset R_x \Leftrightarrow \omega(R_x) < \omega(R_y), R_y \mathbin{\mathcal{R}}_{EM} R_y
\]

The Exact Match conflict is the most severe since it is impossible to understand which one of rules \( R_x \) and \( R_y \) the security manager wanted to be performed. This is a major problem both in case the actions of the two rules are different and in case they are the same, since the rules in between them might act on packets matched by the condition of rules \( R_x \) and \( R_y \), so that it can make a critical difference to eliminate either \( R_x \) or \( R_y \).

**Shadowing:** A rule is shadowed when a previous rule, with different action, matches all the packets that this rule matches so that the shadowed rule will never be activated. Formally,

\[
R_y \subset C_{SH} R_x \Leftrightarrow \omega(R_x) < \omega(R_y), R_y \mathbin{\mathcal{R}}_{IM} R_x, A[R_x] \neq A[R_y],
\]

where \( A[R] = \text{Action corresponding to rule } R \).

This kind of conflict is very severe because the inactivity of the rule \( R_y \) represents a violation of the aim of the security manager, consequently performing an erroneous behavior of the device.

**Post Redundancy:** A rule is in post redundancy when a previous rule, with same action, matches all the packets that this rule matches so that the redundant rule will never be activated. Formally,

\[
R_y \subset C_{PO} R_x \Leftrightarrow \omega(R_x) < \omega(R_y), R_y \mathbin{\mathcal{R}}_{IM} R_x, A[R_x] = A[R_y].
\]

Since the actions are the same, an erroneous behavior of the device does not necessarily occur. Yet, this kind of conflict is severe because the inactivity of the rule \( R_y \) represents a violation of the aim of the security manager, according to our working hypothesis.

**Correlation:** Two rules are correlated if they have different filtering actions, and the first rule matches the same packets that the second rule matches and vice versa. Formally,

\[
R_x \subset C_{CO} R_y \Leftrightarrow R_x \mathbin{\mathcal{R}}_{C} R_y, A[R_x] \neq A[R_y]
\]

In this situation a violation of the aim of the security manager does not occur because both the rules are active, but is not possible to decide if the behavior of the device on the traffic matching both rules is erroneous or not, because it depends on the relative order of the two rules.

**Pre Redundancy:** A rule is in pre redundancy with a previous rule if these have the same actions and if the second rule can match all the packets that the first rule matches. Between the two rules there should not be rules that are in relation with the redundant rule. Formally,

\[
\begin{align*}
R_y \subset C_{PR} R_x \Leftrightarrow & \omega(R_x) < \omega(R_y), R_y \mathbin{\mathcal{R}}_{IM} R_y, A[R_x] = A[R_y] \\
& \not\exists R_z : \omega(R_x) < \omega(R_z) < \omega(R_y) \\
& R_x \mathbin{\mathcal{R}}_{EM} \{ R_y, \mathbin{\mathcal{R}}_{C} \} R_z, A[R_x] \neq A[R_z]
\end{align*}
\]

This kind of conflict can be seen just as an anomaly, not a severe conflict, because both rules will process some traffic.

**Generalization:** A rule is a generalization of a previous rule if these have different actions and if the second rule can match all the packets that the first rule matches. Formally,

\[
\begin{align*}
R_y \subset C_{GE} R_x \Leftrightarrow & \omega(R_x) < \omega(R_y), R_y \mathbin{\mathcal{R}}_{IM} R_y, A[R_x] \neq A[R_y] \\
& \exists R_z \mathbin{\mathcal{R}}_{IM} R_x
\end{align*}
\]

This kind of conflict is also not severe because both rules will process some traffic performing different actions. This case has to be labelled as a normal and desirable situation.

**No Conflict:** When two rules do not fall under any of the previous categories there is a state of No conflict.

To perform the comparison with the selectors of the rules necessary for the detection of conflicts the state diagram presented in [4] has been changed. The modified state machine transition diagram is shown in Fig. 1, with some simplification for space reasons.

![State transition diagram for conflict analysis](image)

**V. Conflict Resolution**

Firewall policy has to be declared incorrect if a progressive rules analysis points out at least one of the severe conflicts as Exact Match, Shadowing and Post Redundancy. Therefore it can be judged correct if the analysis discovers that the policy does not have any severe conflicts. Conflict resolution makes changes in firewall policy so that it becomes free from the
severe conflicts, without losing any of the security services required by security manager.

Two resolution algorithms have been formalized.

A. All Disjoint Algorithm

The purpose of this algorithm is to obtain a policy without any conflicts, not even anomalies. This objective could be reached by implementing a series of changes to the set of rules according to the criteria specified below. The result of this process is a set of completely disjoint rules.

This method arises from the consideration that only one action can be performed on an incoming packet in the device. Priority is just an artificial concept elaborated to help the administrator during the writing of the policy and it is, in reality, the source of the conflicts we are analyzing.

The algorithm is composed of two phases. During the first phase the rules are modified. In the second phase there is post processing that eliminates all the rules with the same action as the device’s default. Changes are performed when a conflict is detected between the analysed rule and another one previously inserted. Thus, when a new rule is inserted we are certain that the rules previously inserted are already disjoint and it does not present any conflict, not even anomaly.

Below we will illustrate the resolutions corresponding to each conflict category:

- **Exact Match:** When an exact match conflict occurs it is not possible to make an autonomous decision. We will then have to ask the security manager who will decide which action and priority he wants to apply on the traffic matched by the two rules. The other rule will then be deleted. In case of opposite actions, the conflict resolution process could be automated by selecting the most restrictive action.

- **Shadowing:** In this situation the two considered rules, despite being in the wrong order, bring to light the different aims of the security manager for the two flows of traffic. The common traffic will be deleted from the most general rule. Becoming disjoint, both rules will be maintained.

- **Post Redundancy:** When this conflict is detected it is possible to delete the redundant rule given that its action has already been performed by the implementation of the previous rule.

- **Correlation:** As with the exact match conflict, here it is not possible to decide what the aim of the security manager was about the common traffic. We can ask the user or maintain the rule with the more restrictive action. Once the action is chosen, the traffic flow matched by the corresponding rule is deleted from the other rule.

- **Pre Redundancy:** As in the case of Post Redundancy we can delete the redundant rule.

- **Generalization:** The common traffic will be deleted from the most general rule.

- **No Conflict:** To obtain a truly disjoint set of rules, we also have to identify the cases in which two rules are correlated but the correlation conflict does not occur because these two rules have the same action. In this situation we have to delete the common traffic from one of the two rules. In any other case of no conflict no action has to be performed.

The All Disjoint algorithm is the most effective in suppressing conflicts, as it removes any inter-dependence among all rules in the policy configuration file. However, in doing so it may split a single original rule into a large number of new rules, to separate different ranges of selector variables. This destroys completely the original file and can give rise to an impractical number of rules in the configuration file. For these reason we consider next a different algorithm that solves the most severe conflicts by reordering existing rules.

B. Inclusive Match Ordered Algorithm

This Inclusive Match Ordered (IMO) algorithm ensures that, through the elimination of some rules, and modifications of the priority value of the remaining rules, the list of all firewall policy converges to a final state without severe conflicts.

Let $A$ be any set consisting of $n$ distinct rules, subset of set $B$ which includes all the rules. $B$’s elements can be assumed distinct provided to have taken care of all the rules in Exact Match relation as described in the previous algorithm.

Any pair of rules arising a severe conflict other than Exact Match, i.e. Shadowing or Post Redundancy, are in $R_{IM}$ relation, so that the most specific rule has a higher priority than the other rule. Thus, the research of a severe-conflict-free policy list state can be reduced to a problem of partial ordering of the rules related by $R_{IM}$.

**Algorithm description:** The IMO algorithm is composed of four steps:

1) Delete Exact Match conflicts asking the user on the processing he wants to apply to the examined traffic (or by a conservative philosophy automatically deleting the less restrictive rule).

2) Scan the policy list and solve any found conflict $R_xCSHR_y$ or $R_xCPOR_y$ by moving $R_y$ immediately before $R_x$ which was hiding it.

3) Repeat step 2) until reaching the state of absence of Shadowing and PostRedundancy conflicts.

4) Compare the output of the previous two steps with the initial policy list and notify the security manager with the possible presence of Correlation conflicts so that the respective rule order is modified.

The existence of a conflict-free state and the convergence of the research algorithm on this final state within a finite number of steps, are both proved next.

**Solution Existence Proof:** First, we define a relation Exact/Inclusive Match, denoted as $R_{E/IM}$, on the set $A$ of all the policy rules, as follows:

$$R_xR_{E/IM}R_y \iff \forall i: R_x[i] \subseteq R_y[i]$$

where $i \in F = \{\text{prot, src_ip, src_port, dst_ip, dst_port}\}$. 
The relation $\mathcal{R}_{E/IM}$ is made up of the join of $\mathcal{R}_{EM}$ and $\mathcal{R}_{IM}$. As a matter of fact, if the relations are verified with equal signs for all the selectors, $\mathcal{R}_{E/IM}$ is referable to $\mathcal{R}_{EM}$, instead if $\exists k$ such that $R_x[k] \subseteq R_y[k]$ where $k \in F$, the relation $\mathcal{R}_{E/IM}$ is referable to $\mathcal{R}_{IM}$. Since subset $B$ only contains distinct rules, $\mathcal{R}_{E/IM}$ on $B$ coincides with $\mathcal{R}_{IM}$.

Relation $\mathcal{R}_{E/IM}$ defines a subset $\mathcal{R}_{E/IM}$ of the cartesian product $A \times A$. Our aim is to demonstrate that the relation $\mathcal{R}_{E/IM}$ is a relation of (partial) order, on the set $A$, that is:

$$
\begin{align*}
R_x R_{E/IM} R_y \text{ and } R_y R_{E/IM} R_z & \Rightarrow R_x R_{E/IM} R_z & (1) \\
R_x R_{E/IM} R_y \text{ and } R_y R_{E/IM} R_z & \Rightarrow R_x R_{E/IM} R_y & (2)
\end{align*}
$$

Proof of (1)

Regarding the protocol and port selectors, relation $\subseteq$ is equivalent to the set theory’s inclusion relation, so this can be proved by the transitivity of $\subseteq$ operator for the set theory.

For the IP address selectors, instead, the check is more complicated. We are given the hypotheses:

$$
\begin{align*}
R_x[ip] \subseteq R_y[ip] & \Leftrightarrow \left\{ \begin{array}{l}
mask_x \geq mask_y \quad (3) \\
Q_{xy} \oplus Q_{yy} = 0 \quad (4)
\end{array} \right. \\
R_y[ip] \subseteq R_z[ip] & \Leftrightarrow \left\{ \begin{array}{l}
mask_y \geq mask_z \quad (5) \\
Q_{yz} \oplus Q_{zz} = 0 \quad (6)
\end{array} \right.
\end{align*}
$$

where $\oplus$ is the bitwise XOR logical operator and $Q_{ij} = ip_i \& MS(mask_j)$, $\&$ being the bitwise AND logical operator and $MS(x)$ is the 32 bit mask having its $x$ most significant bits set to 1 and the remaining $32 - x$ bits set to 0.

We have to prove the following:

$$
R_x[ip] \subseteq R_z[ip] \Leftrightarrow \left\{ \begin{array}{l}
mask_x \geq mask_z \quad (7) \\
Q_{xz} \oplus Q_{zz} = 0 \quad (8)
\end{array} \right.
$$

Equation (3) and (5) imply $mask_x \geq mask_y \geq mask_z$, hence eq. (7) follows.

Equation (8) says that the IP addresses of rules $R_x$ and $R_z$ must have the same most significant $mask_z$ bits. Indeed, eqs. (5)-(6) imply the most significant $mask_z$ bits of $R_y[ip]$ and $R_z[ip]$ are the same; eqs. (3)-(4) show that the most significant $mask_z$ bits of $R_x[ip]$ and $R_y[ip]$ are the same. Since $mask_y \geq mask_z$, eq. (8) is proved.

Proof of (2)

For protocol and port selectors we can demonstrate (2) by the antisymmetric property of the set theory’s operator $\subseteq$.

For the IP address selector, instead, given the hypotheses:

$$
\begin{align*}
R_y[ip] \subseteq R_x[ip] & \Leftrightarrow \left\{ \begin{array}{l}
mask_x \geq mask_y \quad (9) \\
Q_{xy} \oplus Q_{yy} = 0 \quad (10)
\end{array} \right. \\
R_y[ip] \subseteq R_z[ip] & \Leftrightarrow \left\{ \begin{array}{l}
mask_y \geq mask_z \quad (11) \\
Q_{yz} \oplus Q_{zz} = 0 \quad (12)
\end{array} \right.
\end{align*}
$$

we have to prove that:

$$
R_x[ip] = R_y[ip] \quad (13)
$$

The unique case where both (9) and (11) are true is when $mask_x = mask_y$. Then, eqs. (10) and (12) imply that the most significant $mask_x = mask_y$ bits of IP addresses in $R_x$ and $R_y$ are the same. Therefore, $R_x[ip]$ and $R_y[ip]$ denote the same set of IP addresses and eq. (13) is verified, q.e.d.

Since $\mathcal{R}_{IM}$ on $B$ is an order relation, we can define an ordering function $\Omega$:

$$
\Omega : B \rightarrow \{1, \ldots, n\} \quad s.t. \quad R_x \mathcal{R}_{IM} R_y \Rightarrow \omega(R_x) < \omega(R_y)
$$

This function $\Omega$ binds the position to the rules depending on the order relation $\mathcal{R}_{IM}$. The application of $\Omega$ on the set $B$ assures that a final state is reached in which there is no Shadowing or PostRedundancy conflict.

Algorithm convergence proof: To show the convergence of the IMO algorithm we define a Disorder Set $D$ of the initial configuration, that contains all the couples of rules $(R_i, R_j) \in B \times B$, so that:

$$
R_i \mathcal{R}_{IM} R_j \quad \text{and} \quad \omega(R_j) < \omega(R_i)
$$

The cardinality $|D|$ of the disorder set is named Disorder Index $d$. In the final state the configuration has to have a Disorder Index equal to zero. We also define a basic step of the algorithm as the processing necessary to reduce $d$ by at least one unit. Therefore this basic step is equal to the step 2) of the IMO algorithm.

During this step we solve the first instance so that $R_y \mathcal{R}_{IM} R_x$, with $\omega(R_x) < \omega(R_y)$. For the first occurrence of the conflict we define the case when, given a couple $(R_x, R_y) \in D$, we have:

$$
\begin{align*}
\text{if } R_x \neq R_j & \Rightarrow \omega(R_y) < \omega(R_j) \quad (\text{14}) \\
\text{if } R_y \neq R_j & \Rightarrow \omega(R_x) < \omega(R_i) \quad (\text{15})
\end{align*}
$$

\forall(R_i, R_j) \in D, s.t.(R_i, R_j) \neq (R_x, R_y)

In other words $\omega(R_y)$ should be the lowest of the position of all the rules labelled $R_j$ in the Disorder Set $D$. Since set $B$ does not contain duplicates, there could only be equality if the rules $R_y$ and $R_j$ represent the same element in $B$. In this case the value of $\omega(R_y)$ should be the lowest of all the rules labelled $R_i$ in the Disorder Set $D$.

As seen, conflict resolution requires the modification of rule $R_y$ position, such that the updated value of $\omega(R_y)$ becomes less than the updated value of $\omega(R_x)$. This change brings about $(R_x, R_y) \notin D$ decreasing the disorder index $d$ by one unit.

This position modification could potentially generate other alterations on the composition of set $D$, as it could change the respective order of $R_y$ toward a rule $R_x$, such that $\omega(R_x) < \omega(R_x)$. It should be considered that the final state puts restrictions on the partial order of the couples in $\mathcal{R}_{IM}$ relation only, so this problem could exist exclusively when the rules $R_x$ and $R_y$ are in this relation.

We want to demonstrate that any incidental alterations of set $D$ composition, caused by the presence of one or more intermediate rules $R_i$ in $\mathcal{R}_{IM}$ with $R_y$, could result only in the removal of other couples from $D$. Disorder index is therefore strictly decreasing during execution of the basic step. The problematic scenarios are the two following ones only:

1) $R_x \mathcal{R}_{IM} R_y$

For transitivity of $\mathcal{R}_{IM}$, condition $R_y \mathcal{R}_{IM} R_z$ requires
R_\z R_\xi R_\z) to be verified also. From the definition of R_\z as an intermediate rule between R_\x and R_\y it has to be \omega(R_\z) < \omega(R_\x) < \omega(R_\y), so rules R_\z and R_\x have to verify the condition (R_\x, R_\z) \in D. This situation would contrast with the hypothesis that (R_\x, R_\y) was the first conflict occurred. From the definition, in fact, it should be \omega(R_\y) < \omega(R_\x), for every couple (R_\x, R_\z) \in D, such that R_\z \neq R_\y, but for j = z, this can not be true. For these reason this scenario can never take place.

2) R_\y R_\xi R_\z

From the definition of R_\z as an intermediate rule between R_\x and R_\y it has to be \omega(R_\z) < \omega(R_\x) < \omega(R_\y), so rules R_\x and R_\z have to verify the condition (R_\x, R_\z) \in D. For this reason, moving rule R_\y before rule R_\x, would also entail that new\omega(R_\y) < new\omega(R_\x) < new\omega(R_\z), where new\omega denotes the updated position value. This involves the elimination of another couple (R_\y, R_\z) from D. Disorder index d would then be further decreased by one unit.

Since the disorder index is strictly decreasing during the execution of the basic step, the algorithm converges to the final state by a finite number of basic steps, at most equal to the Disorder Index d_0 of the initial configuration. The value d_0 can be upper bounded by:

\[ \sum_{i=2}^{n} (i-1) = \frac{(n-1) \cdot n}{2} \]

This concludes the proof. Incidentally, note that the last equality poses an upper bound on the computational complexity of IMO algorithm.

VI. SOFTWARE IMPLEMENTATION

Analysis and resolution methodologies have been implemented in a software tool developed in C#, the object oriented Microsoft programming language. The tool is named PETRA. Development phase has been realized in the now freeware environment MS Visual Studio 2005 Express Edition based on the framework .NET 2.0.

The detection phase achieves the necessary comparisons using a tree data structure. This structure is built according to a recursive procedure. The policies are read from a SQL database that gives a general representation model of the network security policies. The database stores the configuration of all the network security devices; it can be populated either manually in a general case or by means of automated tools developed specifically for ELSAG networking products.

The resolution phase is performed by the implementation of the algorithm Inclusive Match Ordered Resolution. The user can choose between using a completely automatic or an interactive conflict resolution.

A testing phase has also been performed so that the component was put through a set of scenarios which allowed to verify the resolution action undertaken every time. In all the realistic scenarios we tested the component performed the modifications as expected, by taking a negligible running time.

In Figure 2 we plot the required running time of the conflict resolution software as a function of the number of rules so built that each rule is conflicting with all subsequent rules in the ordered list given as input to the software. This is a possibly artificial yet a worst case for the conflict resolution algorithm, even if it is hopefully not realistic as shown in Table I, where a hypothetic security administrator placed 8514 conflicts in only 492 rules; however in that case our solution needs only a couple of minutes to solve all the conflicts. Our experiments were performed on an Intel Centrino Core2 Duo 1800 MHz platform with 1 GByte of RAM.

<table>
<thead>
<tr>
<th>Rules</th>
<th>Conflicts</th>
<th>Res. time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>45</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>120</td>
<td>31</td>
</tr>
<tr>
<td>21</td>
<td>231</td>
<td>62</td>
</tr>
<tr>
<td>26</td>
<td>325</td>
<td>78</td>
</tr>
<tr>
<td>30</td>
<td>378</td>
<td>94</td>
</tr>
<tr>
<td>32</td>
<td>435</td>
<td>109</td>
</tr>
<tr>
<td>33</td>
<td>528</td>
<td>140</td>
</tr>
<tr>
<td>165</td>
<td>2310</td>
<td>4500</td>
</tr>
<tr>
<td>231</td>
<td>3663</td>
<td>14450</td>
</tr>
<tr>
<td>264</td>
<td>4389</td>
<td>23150</td>
</tr>
<tr>
<td>297</td>
<td>5148</td>
<td>34250</td>
</tr>
<tr>
<td>330</td>
<td>5940</td>
<td>49312</td>
</tr>
<tr>
<td>363</td>
<td>6765</td>
<td>67625</td>
</tr>
<tr>
<td>396</td>
<td>7623</td>
<td>89829</td>
</tr>
<tr>
<td>492</td>
<td>8514</td>
<td>116953</td>
</tr>
</tbody>
</table>

VII. CONCLUSIONS AND FUTURE WORKS

One of the most critical aspects of security problems is the impossibility of accurately checking a system real weaknesses. In a complex and distributed environment this problem is greatly accentuated. During the process of configuration
and implementation of the network security policies errors can occur, resulting in holes in security and, consequently, compromising the entire system functionality. These errors are often very hard to detect by performing a manual or visual inspection. For this reason, automatic management of this phase is required.

We define a framework and algorithms to alleviate this problem for generic stand-alone firewalls or security gateway, by means of automated policy conflict identification and resolution. In this way, the security manager is supplied with an automatic tool that can detect, locate and solve conflicts that may occur within a firewall policy file. Based on the previous literature on this subject we have formalized a new conflict classification founded on the severity concept. The research community described detection mechanism but, none of these studies provided a resolution phase. Thus we have proposed two automatic conflict resolution algorithms, providing a formal proof for the existence of the solution and the algorithm convergence. Finally we implemented one of them in a software tool.

Our future research plans include extending the proposed anomaly resolution techniques to handle distributed firewall policies, S-VPN map policies and their interaction.

ACKNOWLEDGMENT

The authors are grateful to Prof. Stefano Capparelli, University of Roma “La Sapienza”, for his advice on the algebraic theory formulation. The authors would also like to thank Alessio Orlandi and Emanuele Francocci, “EXCITE Security” Dept. of ELSAG S.p.A., for their help on the software testing phase.

REFERENCES