Polynomial Heuristic Algorithms for Inconsistency Characterization in Firewall Rule Sets

S. Pozo, R. Ceballos, R. M. Gasca, A. J. Varela-Vaca
Department of Computer Languages and Systems, ETS Ingeniería Informática,
University of Seville, Avda. Reina Mercedes S/N, 41012 Sevilla, Spain
{sergiopozo,ceball,gasca}@us.es
www.lsi.us.es/~quivir

Abstract

Firewalls provide the first line of defence of nearly all networked institutions today. However, Firewall ACLs could have inconsistencies, allowing traffic that should be denied or vice versa. In this paper, we analyze the inconsistency characterization problem as a separate problem of the diagnosis one, and propose formal definitions in order to characterize one-to-many inconsistencies. We identify the combinatorial part of the problem that generates exponential complexities in combined diagnosis and characterization algorithms proposed by other authors. Then we propose a decomposition of the combinatorial problem in several smaller combinatorial ones, which can effectively reduce the complexity of the problem. Finally, we propose an approximate heuristic and algorithms to solve the problem in worst case polynomial time. Although many algorithms have been proposed to address this problem, all of them are combinatorial. The presented algorithms are an heuristic way to solve the problem with polynomial complexity. There are no constraints on how rule field ranges are expressed.

1. Introduction

A firewall is a network element that controls the traversal of packets across different network segments. It is a mechanism to enforce an Access Control Policy, represented as an Access Control List (ACL). An ACL is in general a list of linearly ordered (total order) condition/action rules. The *condition* part of the rule is a set of condition attributes or selectors, where $|condition|=k$ (k is the number of selectors). The *condition* set is typically composed of five elements, which correspond to five fields of a packet header [3]. In firewalls, the process of matching TCP/IP packets against rules is called filtering. A rule matches a packet when the values of each field of the header of a packet are subsets or equal to the values of its corresponding rule selector. The *action* part of the rule represents the action that should be taken when a packet matches a rule.. In firewalls, two actions are possible: *allow* or *deny* a packet. An example of a rule set is presented in Figure 1. Firewall ACLs are commonly named *rule sets*.

Firewalls have to face many problems in modern networks [7]. One of the most important ones is rule set consistency. As can be seen from the example in Figure 1, selectors of rules can partially or totally overlap (for example, the protocol selector). There is an inconsistency when two or more rules with different actions overlap. An inconsistent firewall ACL implies in general a design error, and indicates that the firewall is accepting traffic that should be denied or vice-versa. In this paper, detection is understood as the action of finding the rules that are inconsistent with other rules; identification is the action of finding the rules that cause all the inconsistencies among the detected inconsistent rules (the faulty rules), whose removal produces a consistent rule set; and characterization is understood as the action of naming the identified inconsistent rules among a pre-established taxonomy of faults.

In this paper we analyze the inconsistency characterization problem in firewall rule sets, and extend the complete formal inconsistency characterization given by Al-Shaer et al. [10] in order to characterize inconsistencies resulting from the clustering of rules, resulting in a complete one-to-many characterization. In addition, we identify the combinatorial part of the problem that causes the combinatorial explosion in combined diagnosis and characterization algorithms proposed by other authors. Then we propose a decomposition of the combinatorial

Priority/ID	Protocol	Source IP	Src Port	Destination IP	Dst Port	Action
R1	tcp	192.168.1.5	any	*.*.*.*	80	deny
R2	tcp	192.168.1.*	any	*.*.*.*	80	allow
R3	tcp	*.*.*.*	any	172.0.1.10	80	allow
R4	tcp	192.168.1.*	any	172.0.1.10	80	deny
R5	tcp	192.168.1.60	any	*.*.*.*	21	deny
R6	tcp	192.168.1.*	any	*.*.*.*	21	allow
R7	tcp	192.168.1.*	any	172.0.1.10	21	allow
R8	tcp	*.*.*.*	any	*.*.*.*	any	deny
R9	udp	192.168.1.*	any	172.0.1.10	53	allow
R10	udp	*.*.*.*	any	172.0.1.10	53	allow
R11	udp	192.168.2.*	any	172.0.2.*	any	allow
R12	udp	*.*.*.*	any	*.*.*.*	any	deny

Fig 1. Example rule set

problem in several smaller combinatorial ones. This effectively reduces the complexity of the problem. The proposed characterization process and algorithms are built on a previous heuristic diagnosis process that is worst case *$O(n^2)$* time complexity [8].

The paper is structured as follows. In section 2 related works are presented and differences with our proposal are emphasized. Section 3 presents the analysis of the characterization problem and identifies the combinatorial part of it. In section 4 the characterization process with algorithms are proposed, explaining how the problem can be reduced to several smaller combinatorial ones. We conclude in section 5.

2. Related Works

The closest works to ours are related with consistency diagnosis in general network filters. In the most recent work, Baboescu et al. [12] provides algorithms to diagnose inconsistencies in router filters that are 40 times faster than *$O(n^2)$* ones for the general case of *k* selectors per rule. Although its algorithmic complexity is not given, it improves other previous works [13, 14]. However, they preprocess the rule set and convert selector ranges to prefixes, and then apply the algorithms. This imposes the implicit assumption that a range can only express a single interval, which is true [8]. However, the range to prefix conversion technique could need to split a range in several prefixes [15] and thus the final number of rules could increase over the original rule set. Thus, results are given over the preprocessed rule set, which could be bigger and different from the original one.

Other researchers have complemented the diagnosis process with a characterization of the faults with an established taxonomy [10]. As the following proposals treat the problem as a whole and the characterization algorithms are applied directly to the full rule set, the resulting worst case time complexity will be exponential in all cases (recall that characterization is NP, as it is going to be explained bellow). One of the most important advances was made by Al-Shaer et al. [4], where authors define an inconsistency model for firewall ACLs. They give a combined algorithm to diagnose and characterize the inconsistencies between pairs of rules. In addition, they use rule decorrelation techniques [2] as a pre-process in order to decompose the ACL in a new, bigger, one with no overlapping rules. This new rule set is different from the initial one, and the user is the responsible of mapping the rules of this rule set to the original one. Their model and corresponding algorithms can only diagnose and characterize inconsistencies between pairs of rules. Although the proposed characterization algorithm proposed by Al-Shaer is polynomial, the decorrelation pre-process imposes a worst case exponential time and space complexity for the full process.

A modification to their algorithms was provided by García-Alfaro et al. [5], where they integrate the decorrelation and characterization algorithms of Al-Shaer, and generate a decorrelated and consistent rule set. Thus, due to the use of the same decorrelation techniques, this proposal also has worst case exponential complexity. The resulting ACL is also bigger and different from the original one. However, García-Alfaro et al. provide a characterization technique with multiple rules.

Ordered Binary Decision Diagrams (OBDDs) have been used in Fireman [9], where authors provide a diagnosis and characterization technique with multiple rules. A very important improvement over previous proposals is that they do not need to decorrelate the ACL, and thus, results are given over the original one. Note that the complexity of OBDD algorithms depends on the optimal ordering of its nodes, which is a NP-Complete problem [6]. This results in a worst case

O(2^n) time complexity with the number of rules, as other proposals.

The combination of diagnosis and characterization in only one stage results in exponential algorithms that are applied to a big problem (the full rule set). Although optimal diagnosis and characterization are worst-case combinatorial problems [8], diagnosis can be used to split the characterization problem in several smaller ones [8]. Then, optimal or heuristic algorithms can be applied to these smaller problems.

The main difference of these works with ours is that, previous to algorithm design, we have done an analysis of the consistency diagnosis and characterization problem in firewall rule sets. As a result, we proposed to divide consistency management in two sequential processes [8]: detection and identification (diagnosis) of inconsistent rules, and characterization of the diagnosis. We extend Al-Shaer inconsistency taxonomy [10] to characterize inconsistencies resulting from the clustering of rules, resulting in a complete one-to-many characterization. The analysis of the characterization problem enabled us to identify and isolate the combinatorial parts of it and improve the algorithmic complexity of the full process. An optimal characterization algorithm must analyze all possible solutions in order to find the optimal one. Since there is a trade off between optimally solving the problem in exponential time or using an approximation to the optimum, we propose a polynomial heuristic and algorithms that implement it that solve the characterization problem in worst case polynomial time.

The presented algorithms are capable of handling full ranges in rule selectors without doing rule decorrelation, range to prefix conversion, or any other pre-process. Thus, results are given over the original, unmodified, rule set. However, our process does not cope with redundancies, because redundancy is not a consistency problem (it does not change the semantics of the rule set).

To the best of our knowledge, this is the first time that the characterization problem has been divided in several smaller ones, and a polynomial heuristic algorithm has been proposed to solve it. It is also the first time Al-Shaer's formal definitions have been extended to support a complete one-to-many characterization. A Java tool called Fast Firewall ACL AnalysiS Toolkit V2 (FFaaST V2) has been implemented and is available upon request.

3. Analysis of the Inconsistency Characterization Problem

Real life rule sets can be decomposed in two different subsets of rules (Figure 2 presents an example of some subsets of Figure 1 example). The first one is a set of consistent rules (Definition 3.1). The other one is formed by subsets of inconsistent rules, called ICIRs [8], with bold rules as ICIR roots (Definition 3.2). We shall now formalize a firewall ACL.

- Let *RS* be a firewall ACL consisting of *n* rules, $RS = \{R_1, ...R_n\}$
- Let $R =< H, Action >, H \in \mathbb{N}^5$ be a rule, where $Action = \{allow, deny\}$ is its action
- Let $R_j[k], 1 \le j \le n, k \in \{protocol, src\_ip, src\_prt, dst\_ip, dst\_prt\}$ be a selector of a rule R_j
- Let '<' and '>' be operators which define the priority of the rules, where $R_i < R_j$ means that then R_i has greater priority than R_j and its action will be taken first, and vice versa

Definition 3.1. Inconsistency. Two rules $R_i, R_j \in RS$ are *inconsistent* if and only if the intersection of *each of all* of its selectors *R[k]* is not empty, and they have different actions, *independently of their priorities*. The inconsistency between two rules expresses the *possibility* of an undesirable effect in the *semantics* of the rule set. The semantics of the rule set changes if an inconsistent rule is removed.

$$Inconsistent(R_i, R_j, RS), 1 \le i, j \le n, i \ne j$$

$$\Leftrightarrow R_i[k] \cap R_j[k] \ne \varnothing \wedge R_i[Action] \ne R_j[Action],$$

$$\forall k \in \{protocol, src\_ip, src\_prt, dst\_ip, dst\_prt\}$$

Definition 3.2. Independent Cluster of Inconsistent Rules, ICIR. An ICIR represents a cluster or collection of inconsistent rules as a tree. The root of the ICIR is the rule which has the greatest number of inconsistencies with other rules of the same cluster. By definition, the action of the ICIR root is the opposite of the actions of all of its children. Children rules are consistent between them.

Let $CV = \{R_1, ...R_n\}$ be a set of rules, then

$$ICIR(root, CV) \Leftrightarrow \forall R_i \in CV \bullet Inconsistent(root, R_i) \wedge$$

$$\forall R_i, R_j \in CV, i \ne j \bullet \neg Inconsistent(R_i, R_j)$$

Definition 3.3. Diagnosis Set, DS. It is the set of rules that could be directly removed from the rule set in order to get a consistent one. It is formed by the root of all ICIRs.

Let $ICIRS = \{ICIR_1, ..., ICIR_m\}$ be the set of all ICIR of a given RS,

then $DS = \{ICIR_1(root), ..., ICIR_m(root)\}$

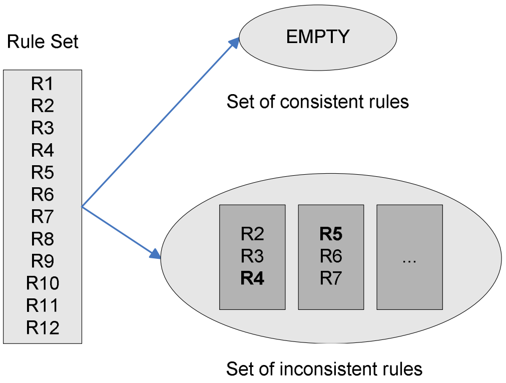

Fig. 1. Decomposition of a rule set

Clustering rules is a necessary process in order to obtain complete and correct results for one to many characterization if there is a taxonomy of faults. There exist cases where doing no clustering could return incomplete and/or incorrect results. Figure 3(a) presents a three rule example where R_z is shadowed by $R_x \cup R_y$. However, if no clustering is done, characterization would return that R_z is a generalization of R_x, and that $\{R_y, R_z\}$ are correlated, which is not a correct result. In addition, clustering of rules with the same inconsistency is also very important in order to *abbreviate* results with no loss of information. The example of Figure 3(b) presents a generalization inconsistency. In this example, R_z is a generalization of $R_x \cup R_y$. With no clustering, two generalization inconsistencies would be returned, gaining no information over the clusterized form.

3.1. Characterization taxonomy of one to many inconsistencies

A complete one to one inconsistency characterization was given by Al-Shaer [10]. Our definitions extend Al-Shaer work in order to characterize the diagnosis of an arbitrary number of rules with the same action versus one other. Our approach is also complete (as it is an extension of Al-Shaer work) based on the relationships that can be established between the selectors of rules: equality, subset and superset. To be as realistic as possible, it is considered that each selector is a set of elements whose content can be expressed using the common syntax of the most used firewall languages, which have been previously analyzed in [1]. The syntax analysis has been omitted due to space constraints, but the result is presented in Figure 3. This figure represents, for each of the five typical selectors, the common syntax supported by IPTables, Cisco PIX, Checkpoint FW-1, BSD PF, BSD IPFW and BSD IPFilter. Note that all selectors except protocol permit the representation of one element, a continuous range of elements, or a wildcard (representing all possible elements of the set). The content of each selector is also bounded by the constraints imposed by the corresponding field of the TCP/IP header. Note that, although expressing ranges is possible for all selectors, ranges must be continuous. IP address ranges are expressed in CIDR form.

$$R_x : \{port \in [10-50]\} \Rightarrow \{allow\}$$
$$R_y : \{port \in [40-90]\} \Rightarrow \{allow\}$$
$$R_z : \{port \in [10-80]\} \Rightarrow \{deny\}$$

(a)

$$R_x : \{port \in [10-39]\} \Rightarrow \{allow\}$$
$$R_y : \{port \in [40-60]\} \Rightarrow \{allow\}$$
$$R_z : \{port \in [0-65535]\} \Rightarrow \{deny\}$$

(b)

Fig. 2. Rule clustering examples

Selector	Common Syntax	Comments
Source and Destination IP Address	- IP - Block* - Wildcard	* A block is a continuous range expressed in CIDR
Protocol	- Number - Wildcard	
Source and Destination Ports	- Number - Range: [p1,p2]* - Wildcard	* The range must be continuous

Fig. 3. Common syntax for most used firewall languages

- **Shadow.** A rule R_y is shadowed by another rule R_x, with $R_x > R_y$, if all of its selectors to or supersets of the selectors of R_y, and R_x and R_y have different actions.

$$\exists R_x, R_y \in RS \bullet R_x > R_y \bullet Shadow(R_y) \Leftrightarrow$$

$$\forall k \bullet R_y[k] \subset R_x[k] \wedge R_x[Action] \neq R_y[Action]$$

$$k \in \{protocol, src\_ip, src\_prt, dst\_ip, dst\_prt\}$$

Shadow

$$\exists R_x, R_y \in RS \bullet R_x > R_y \bullet ExactShadow(R_y) \Leftrightarrow$$

$$\forall k \bullet R_y[k] = R_x[k] \wedge R_x[Action] \neq R_y[Action]$$

$$k \in \{protocol, src\_ip, src\_prt, dst\_ip, dst\_prt\}$$

Exact shadow

This definition can be extended to support a cluster of rules with the same action in R_x or R_y (but not in both). If R_x is a cluster of rules and R_y is a rule, then R_y is shadowed by R_x. Similarly, if R_x is a rule, and R_y is a cluster, then R_y are shadowed by R_x. It is only possible to form a cluster of rules if they can form a continuous range in all of its selectors. Cluster forming is shown in next section.

- **Generalization.** It is the inverse of shadow respect to the priority. A rule R_y is a generalization of R_x, with R_x>R_y, if all of the selectors of R_x are subsets of the selectors of R_y, and both rules have different actions. R_x is usually considered an exception and not an error. Again, clusters can be formed.

$$\exists R_x, R_y \in RS \bullet R_x > R_y \bullet Generalization(R_y) \Leftrightarrow$$

$$\forall k \bullet R_y \supset R_x \wedge R_x[Action] \neq R_y[Action]$$

$$k \in \{protocol, src\_ip, src\_prt, dst\_ip, dst\_prt\}$$

- **Correlation.** Two rules R_x and R_y are correlated if they have different actions, and selectors of R_x intersect with the corresponding selectors of R_y, but R_x and R_y do not have a shadow, exact shadow or generalization relation. Correlation is independent of rule priority. This definition can also be extended to clusters of rules.

$$\exists R_x, R_y \in RS \bullet Correlation(R_x R_y) \Leftrightarrow$$

$$\forall k \bullet R_x[k] \cap R_y[k] \wedge R_x[Action] \neq R_y[Action] \wedge$$

$$\neg(R_x[k] \subseteq R_y[k]) \wedge \neg(R_x[k] \supset R_y[k])$$

$$k \in \{protocol, src\_ip, src\_prt, dst\_ip, dst\_prt\}$$

- **Redundancy.** A rule R_x is redundant to another rule R_y, with R_x>R_y, if all of its selectors are subsets or equal to the selectors of R_x, they have the same action, and if there is no rule between R_x and R_y which is correlated or subset of R_x. Redundancy of R_y respect to R_x is symmetrical. Redundancy is not really an inconsistency, since if all redundant rules are removed, the semantic of the rule set does not change.

$$\exists R_x, R_y \in RS \bullet R_x > R_y \bullet Redundant(R_x) \Leftrightarrow$$

$$\forall k \bullet R_x[k] \subseteq R_y[k] \wedge R_x[Action] = R_y[Action] \wedge$$

$$\neg \exists R_z \in RS, R_x > R_z > R_y \bullet$$

$$Correlation(R_x, R_z) \vee Generalization(R_z)$$

$$k \in \{protocol, src\_ip, src\_prt, dst\_ip, dst\_prt\}$$

$$\exists R_x, R_y \in RS \bullet R_x > R_y \bullet Redundant(R_y) \Leftrightarrow$$

$$\forall k \bullet R_y[k] \subset R_x[k] \vee \forall k \bullet R_y[k] = R_x[k] \wedge$$

$$R_x[Action] = R_y[Action]$$

$$k \in \{protocol, src\_ip, src\_prt, dst\_ip, dst\_prt\}$$

4. Inconsistency Characterization Process

The characterization process explained in this section takes as input the inconsistency diagnosis as ICIRs [8]. The proposed characterization process is divided in two sequential stages. First, the children of each ICIR are grouped in different clusters. Second, clusters of each ICIR are characterized against its ICIR root.

In the diagnosis process, the rule set of Figure 1 is transformed into several ICIRs. The characterization process is also divided in two sequential stages. At the first stage, children of each ICIR are joined in different clusters. At the second stage, each of these clusters are characterized against the ICIR root.

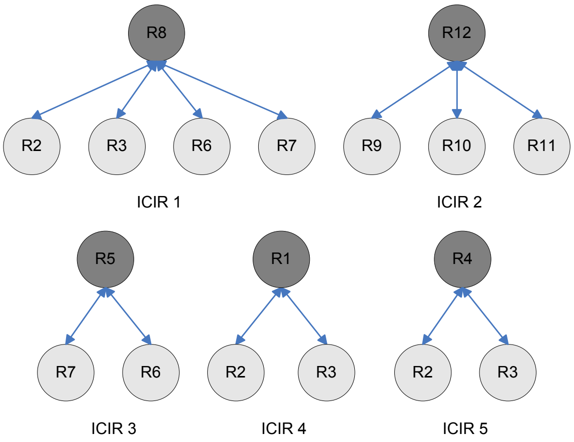

Figure 4. Uncharacterized diagnoses (result of the consistency based diagnosis process)

4.1. Stage 1. Cluster construction (rule join)

At the first stage, for each ICIR, children are joined in different clusters in order to abbreviate the returned characterization for that conflicting rule (ICIR root). These clusters are formed by rules that are subsets, supersets, equal, or form a continuous space for all selectors. The clusters represent the rules that share the same inconsistency with their ICIR root.

By definition, characterization depends on rule priority since the characterization is based on set operations. Rules that come before or after root generate different kinds of inconsistencies with it. For example, if root is shadowed by a cluster of rules that precede it, then rules that go after root cannot participate in the same inconsistency since, by definition, rules that cause a shadowing inconsistency must precede root. For this reason, rules that generate a generalization conflict must be in another cluster. However, it could be possible that root causes a shadowing inconsistency with a cluster of rules that go after it. In this case, we say that the rules in the cluster are shadowed by root. The same is applicable for generalization, as it is the inverse of shadow respect to rule priorities.

In conclusion, due to the priority dependency of characterization definitions, it is possible to simplify the problem even more, dividing ICIR rules in two lists: rules that come before and rules that go after root. Then, clustering is done independently for each of these two lists. Division process is in *O(c)* with the number of children.

Before presenting rule joining algorithm, it is necessary an analysis of the conditions under which joining of rules can be done.

4.1.1. Firewall language syntax. In general, clustering is possible if all rule selectors permit multiple values, ranges and/or wildcards in their syntax. Fortunately, firewall languages support ranges and/or wildcards in all selectors, but only continuous ranges (Figure 3). This enables the clustering of rules for all selectors.

4.1.2. ICIR Structure. In addition, an ICIR must comply with:

1. It must have at least two children. In other case, there are no rules to be joined.
2. At least one selector of ICIR root must be a range of values or a wildcard. The joined rules in a cluster must form a continuous range (with or without

Algorithm 1. Initialization

```
Func initialization(in Rule: root, List of Rule: children)
Alg
  if root.DstPort().isRangeOrWildcard() AND
    children.size()>1 {
      sortAscendingByDestinationPort(children)
      if root.Priority()>children.last().Priority() OR
        root.Priority()<children.first().Priority() {
          clusterize(root, children)
      }
      else {
        List before = Rules with priority > root
        List after = Rules with priority < root
        clusterize(root, before)
        clusterize(root, after)
      }
  else {
    doPairwiseCharacterization(root, children)
  }
End Alg


Func doPairwiseCharacterization(in Rule: root, List of
Rule: children)
Alg
  for each i=1..children.size()
    doClassification(root, children.get(i))
End Alg
```

Algorithm 2. Cluster construction

```
Func clusterize(in Rule: root, List of Rule: children)
Var
  Rule cluster
Alg
  cluster = children.first()
  for each i=2..children.size() {
    if isClusterizable(cluster, children.get(i) AND
      i<children.size()) {
        cluster.joinWith(children.get(i))
    }
    else if isClusterizable(cluster, children.get(i) AND
          i==children.size()) {
            cluster.joinWith(children.get(i))
            doClassification(root, cluster)
    }
    else { // Not clusterizable
      doClassification(root, cluster)
      // re-initializes for a new cluster
      cluster=children.get(i)
    }
  }
End Alg
```

overlapping) and must be subset, superset or equal the corresponding root selector.

3. For root selectors that do not have multiple values, rules in the cluster must have the same value as root, or at least one of them must be a wildcard.

4.1.3. Polynomial heuristic. Traditionally, the diagnosis and characterization of firewall rule sets have been solved in only one stage, resulting in a worst case *$O(2^n)$* time complexity process with the number of *rules* in the rule set. Separating diagnosis from characterization has produced the effect of dividing the combinatorial part of the problem in several much smaller ones, which effectively reduces the computational complexity. Since ICIRs represent independent clusters of inconsistencies, they can also be characterized independently, effectively reducing the problem complexity: the combinatorial problem have been reduced from the entire rule set to several smaller ICIRs. However, there is still a trade off between optimally solving the problem in exponential time, or using an approximation to the optimum. In order to show the feasibility of this approach, in this paper we propose a worst case polynomial heuristic. The heuristic is used when clustering ICIR children. It only takes into account one selector for rule clustering, and does not try to check all possible unions between all selectors. To be restrictive, a selector with a narrow domain should be chosen, because it generally guarantees a good approximation to the optimum. In real rule sets, one of the selectors with the narrower domains is destination port. Although this entirely depends on the particular rule set, destination port is usually expressed as a unique value in the vast majority of real rule sets. Other heuristics could be considered.

Then, for each ICIR, their children are clusterized in several groups by destination port forming a continuous range. This task can be done in linear time if children are ordered by destination port. The first cluster is formed with the first children. Then the next children should be added only if its destination port selector can form a continuous range with the cluster, and if the rest of selectors are equal, subset, superset or wildcard. If it cannot be joined, then the cluster is closed and a new one is formed with that child and the process begins again until there are no more children.

Note that although characterization definitions are complete, the algorithms are not, since they are an approximation.

4.1.4. Cluster construction algorithm. The first part of the process checks ICIR root structure and prepare children for clustering (Algorithm 1). Then, it identifies the rules that can be united with others in each ICIR (Algorithm 2).

Algorithm 1 takes as input the ICIR root and children, and first checks if the ICIR has a valid structure for clustering, as explained in a previous subsection. Then it sorts children by destination port in ascending order, as a preparation for the heuristic. Then, the algorithm checks if root is the last or first rule or is in between. If root is in between it divides children in two lists: rules that come before and rules that go after root, as also was explained before.

Finally, if clustering is possible, it calls Algorithm 2, and if not, it calls directly the inconsistency characterization (Algorithm 3). All operations of Algorithm 1 run in constant time except *sortAscending()*, which is in *O(clogc)* where *c* is the number of children, list copy operations and *doPairWiseCharacterization()* which are in *O(c)*. By the sum rule, time complexity of this algorithm is in worst case *O(clogc)* with the number of children. These algorithms must be run for each ICIR.

Algorithm 2 also takes as input ICIR root and children. This algorithm implement the heuristic as it has been described in the previous section. All operations inside the loop run in constant time and the loop is run for each child, *c*. Algorithm 3 is in *O(c).*

4.2. Stage 2. Inconsistency characterization

As clusterization has been done in the first stage of the process, the inconsistency characterization results a very easy task. Characterization follows directly the extended definitions proposed in an earlier section. Algorithm 3 takes as input ICIR root and the clusters of that ICIR. Then, it checks each type of inconsistency using the equality, subset and superset operations. Note that characterization is different depending on the relative priority of the ICIR root (if it is the first or last rule). All operations of Algorithm 3 run in constant time. Result is returned as a text string.

Algorithm 3. Inconsistency Characterization

```
Func doClassification(in Rule: root, Rule: cluster; out
String: conflictType)
Alg
  if (cluster == root)
    conflictType = “Root is exactly shadowed by union”
  else if (root.getPriority()>cluster.getLastRulePriority()){
    else if (superset(cluster, root))
      conflictType = “Root is shadowed by cluster”
    else if (subset(cluster, root))
      conflictType = “Root is generalization of cluster”
    else
      conflictType = “Root and cluster are correlated”
  }
  else { // Root is first rule
    if (superset(cluster, root)
      conflictType = “Cluster is generalization of root”
    else if (subset(cluster, root)
      conflictType = “Cluster is shadowed by root”
    else
      conflictType = “Root and cluster are correlated”
  }
  return conflictType
End Alg
```

The combined worst case complexity of the three algorithms using the proposed heuristic is, by the sum rule, the maximum of the complexities of the three algorithms, *T(c) = O(clogc)+O(c)+k = O(clogc).* These algorithms must be run for each ICIR and thus, the final time complexity is in *O(h*clogc)*, where *h* is the cardinality of the *Diagnosis Set* (or the number of ICIRs) and *c* is the number of children of each ICIR. Note that the combinatorial part of the inconsistency characterization problem is only the clusterization (where the heuristic has been used), and not the characterization itself.

4.3. Example

Algorithms take as input the ICIRs presented in Figure 4 in no particular order. Suppose that Algorithm 1 receives ICIR1 as input. As destination port of R8 is a wildcard, children are ordered by destination port. Then, as root is the rule with less priority of the ICIR, the algorithm directly calls *clusterize()* (Algorithm 2). Algorithm 2 receives the ICIR root and ordered children. It creates the first cluster with the first child *(R6).* Then it tries to join the next child, *R7,* to the cluster and, as it complies to the restrictions (*R2* and *R7* destination ports are equal), it joins it. It repeats the check with *R2* and, as *R2* destination port is not a subset, superset, wildcard or can form a continuous range with the cluster destination port (21), then the cluster is closed and characterization is called. Then, a second cluster is created with *R2.* Algorithm tries the join of R3 with the cluster (currently formed only by *R2*), and it joins it (*R2* and *R3* destination ports are equal). Since there are no more children, the second cluster is closed and characterization is called. For both clusters *{R6,R7}* and *{R2,R3}* the characterization algorithm returns that *R8* is a generalization of both of them, since all their selectors are subsets of *R8.*

The process applied to the other ICIRs is similar and can be easily followed. Optimal solution is 7 inconsistencies, but the proposed algorithms returns 10:

- ICIR1. Rule *R8* is a generalization of *{R2,R3}* and of *{R6,R7}*
- ICIR2. Rule *R12* is a generalization of *{R9,R10}* and of {*R11}*
- ICIR3. *R6* is a generalization of R5, and *R7* is in correlation with R5
- ICIR4. *R2* is a generalization of R1, and *R3* is in correlation with R1
- ICIR5. *R2* and *R3* are shadow of R4

5. Conclusions and Future Works

In this paper, we have analyzed the inconsistency characterization problem in firewall rule sets. We have proposed complete and formal inconsistency characterization definitions for clusters of rules (a one-to-many characterization).

The analysis of the characterization problem enabled us to identify and isolate the combinatorial part of it. Since there is a trade off between optimally solving the problem in exponential time or using an approximation to the optimum, we have proposed a polynomial heuristic and algorithms that solve the characterization problem in worst case polynomial time. Algorithms are capable of handling full ranges in rule selectors without doing rule decorrelation, range to prefix conversion, or any other pre-process. Results are given over the original, unmodified, rule set.

Anyway, we showed that the combinatorial problems to be solved are very small due to the decomposition made in the diagnosis process, which has effectively reduced a worst case $O(2^n)$ problem in several $O(2^c)$ ones, with *n>>c*. For that reason we expect that an optimal characterization algorithm is going to improve the other reviewed algorithms. A Java tool called Fast Firewall ACL AnalysiS Toolkit V2 (FFaaST V2) with the full diagnosis process is available upon request. No performance analysis has been done due to the fact that, to the best of our knowledge, this is the first approximated solution to this problem. A comparison with an optimal one does not make sense from the performance point of view.

However, our approach has some limitations that give us opportunities for improvement in future works. The most important one is the design of optimal algorithms, and compare its performance with the reviewed algorithms. In addition, since the diagnosis process does not cope with redundancies, the characterization algorithms cannot characterize it. The support of redundancy is another goal.

6. References

[1] S. Pozo, R. Ceballos, R.M. Gasca. "AFPL, An Abstract Language Model for Firewall ACLs". 8th International Conference on Computational Science and Its Applications (ICCSA). Perugia, Italy. Springer-Verlag, 2008.

[2] S. Luis, M. Condell. "Security policy protocol." IETF Internet Draft IPSPSPP-01, 2002.

[3] David E. Taylor. Survey and taxonomy of packet classification techniques. ACM Computing Surveys, Vol. 37, No. 3, 2005. Pages 238 – 275.

[4] E. Al-Shaer, Hazem H. Hamed. Modeling and Management of Firewall Policies". IEEE eTransactions on Network and Service Management (eTNSM) Vol.1, No.1, 2004.

[5] J. García-Alfaro, N. Boulahia-Cuppens, F. Cuppens, Complete Analysis of Configuration Rules to Guarantee Reliable Network Security Policies, Springer-Verlag International Journal of Information Security (Online) (2007) 1615-5262.

[6] B. Bollig, I. Wegener. "Improving the Variable Ordering of OBDDs is NP-Complete". IEEE Transactions on Computers, Vol.45 No.9, September 1996.

[7] A. Wool. A quantitative study of firewall configuration errors. *IEEE Computer*, 37(6):62-67, 2004.

[8] S. Pozo, R. Ceballos, R.M. Gasca. "Improving Computational Complexity of the Inconsistency Characterization Problem in Firewall Rule Sets". International Conference on Security and Cryptography (SECRYPT). Porto, Portugal. INSTICC Press, 2008.

[9] L. Yuan, J. Mai, Z. Su, H. Chen, C. Chuah, P. Mohapatra. FIREMAN: A Toolkit for FIREwall Modelling and ANalysis. IEEE Symposium on Security and Privacy (S&P'06). Oakland, CA, USA. May 2006.

[10] H. Hamed, E. Al-Shaer. "Taxonomy of Conflicts in Network Security Policies." IEEE Communications Magazine Vol.44, No.3, 2006.

[11] E. Al-Shaer, H. Hamed. "TR04-11. Design and Implementation of Firewall Policy Advisor Tool". Multimedia Networking Research Laboratory. School of Computer Science, DePaul University, USA.

[12] F. Baboescu, G. Varguese. "Fast and Scalable Conflict Detection for Packet Classifiers." Elsevier Computers Networks (42-6) (2003) 717-735.

[13] B. Hari, S. Suri, G. Parulkar. "Detecting and Resolving Packet Filter Conflicts." Proceedings of IEEE INFOCOM, March 2000.

[14] D. Eppstein, S. Muthukrishnan. "Internet Packet Filter Management and Rectangle Geometry." Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), January 2001.

[15] V. Srinivasan, G. Varguese, S. Suri, M. Waldvogel. "Fast and Scalable Layer Four Switching." Proceedings of the ACM SIGCOMM conference on Applications, Technologies, Architectures and Protocols for Computer Communication, Vancouver, British Columbia, Canada, ACM Press, 1998.

Acknowledgements

This work has been partially funded by Spanish *Ministry of Science and Education project* under grant DPI2006-15476-C02-01, and by FEDER (under ERDF Program). Many thanks to the anonymous reviewers for their useful comments.