# FAME: A Firewall Anomaly Management Environment

Hongxin Hu, Gail-Joon Ahn and Ketan Kulkarni
Arizona State University
Tempe, AZ 85287, USA
{hxhu,gahn,kakulkar}@asu.edu

## ABSTRACT

Firewalls are a widely deployed security mechanism to ensure the security of private networks in most businesses and institutions. The effectiveness of security protection provided by a firewall mainly depends on the quality of policy configured in the firewall. However, designing and managing firewall policies are often error-prone due to the complex nature of firewall configurations as well as the lack of systematic analysis mechanisms and tools. This paper represents an innovative anomaly management framework for firewalls, adopting a rule-based segmentation technique to identify policy anomalies and derive effective anomaly resolutions. In particular, we articulate a grid-based representation technique for providing an intuitive cognitive sense about policy anomaly and facilitating efficient policy anomaly management. In addition, we demonstrate the feasibility and applicability of our framework through a proof-of-concept prototype of a visualization-based firewall policy analysis tool called Firewall Anomaly Management Environment (FAME).

## Categories and Subject Descriptors

D.4.6 [**Security and Protection**]: Access controls

## General Terms

Security, Management

## Keywords

Firewall policies, anomaly management, visualization tool

## 1. INTRODUCTION

Sitting on the border between a private network and the public Internet, a firewall examines all incoming and outgoing packets based on security rules to monitor suspicious traffic and unauthorized access to Internet-based enterprises. To implement a security policy in a firewall, system administrators define a set of filtering rules that are derived from the organizational network security requirements.

Firewall policy management is a challenging task due to the complexity and interdependency of policy rules. This is further exacerbated by the continuous evolution of network environments. Therefore, effective mechanisms and tools for policy management are crucial to the success of firewalls. Recently, policy anomaly detection has received a great deal of attention [5, 6, 8, 9, 29]. Corresponding policy analysis tools, such as Firewall Policy Advisor [5, 6] and FIREMAN [29], with the goal of detecting policy anomalies have been introduced. Firewall Policy Advisor only has the capability of detecting *pairwise* anomalies in firewall rules. FIREMAN can detect anomalies among *multiple* rules by analyzing the relationships between *one* rule and the collections of packet spaces derived from all preceding rules. However, FIREMAN also has limitations in detecting anomalies [8]. For each firewall rule, FIREMAN only examines all preceding rules but ignores all subsequent rules when performing anomaly analysis. The anomaly detection procedures of FIREMAN are thus incomplete. In addition, each analysis result from FIREMAN can only show that there is a misconfiguration between *one* rule and its preceding rules, but cannot accurately indicate all rules involved in an anomaly.

On the other hand, due to the complex nature of policy anomalies, system administrators are often faced with a more challenging problem in resolving anomalies, in particular, resolving policy conflicts. An intuitive means for a system administrator to resolve policy conflicts is to remove all conflicts by modifying the conflicting rules. However, changing the conflicting rules is significantly difficult, even impossible, in practice from many aspects. First, the number of conflicts in a firewall is typically large, since a firewall policy may consist of thousands of rules, which are often logically entangled with each other. Second, policy conflicts are often very complicated. One rule may conflict with multiple other rules, and one conflict may be associated with several rules. Besides, firewall policies deployed on a network are often maintained by more than one administrator, and an enterprise firewall may contain legacy rules that are designed by different administrators. Thus, without a priori knowledge on the administrators' intentions, changing rules will affect the rules' semantics and may not resolve conflicts correctly. Furthermore, in some cases, a system administrator may intentionally introduce certain overlaps in firewall rules knowing that only the first rule is important. In reality, this is a commonly used technique to exclude specific parts from a certain action, and the proper use of this technique could result in a fewer number of rules [29]. In this case, conflicts are not an error, but intended, which would not be necessary to be changed.

Since the policy conflicts in firewalls always exist and are hard to be eliminated, a practical resolution method is to identify which rule involved in a conflict situation should take precedence when multiple conflicting rules (with different actions) can filter a par-

**Table 1: An example firewall policy.**

| Order | Rule | Protocol | Source IP | Source Port | Destination IP | Destination Port | Action |
|---|---|---|---|---|---|---|---|
| 1 | $r_1$ | UDP | 10.1.2.* | * | 172.32.1.* | 53 | deny |
| 2 | $r_2$ | UDP | 10.1.*.* | * | 172.32.1.* | 53 | deny |
| 3 | $r_3$ | TCP | 10.1.*.* | * | 192.168.*.* | 25 | allow |
| 4 | $r_4$ | TCP | 10.1.1.* | * | 192.168.1.* | 25 | deny |
| 5 | $r_5$ | * | 10.1.1.* | * | * | * | allow |

ticular network packet simultaneously. To resolve policy conflicts, a firewall typically implements a *first-match* resolution mechanism based on the order of rules. In this way, each packet processed by the firewall is mapped to the decision of the first rule that the packet matches. However, applying the *first-match* strategy to cope with policy conflicts has limitations. When a conflict occurs in a firewall, the existing first matching rule may not be a desired rule that should take precedence with respect to conflict resolution. In particular, the existing first matching rule may perform opposite action to the rule which should be considered to take precedence. This situation can cause severe network breaches such as permitting harmful packets to sneak into a private network, or dropping legal traffic which in turn could encumber the availability and utility of network services. Obviously, it is necessary to seek a way to bridge a gap between conflict detection and using first-match mechanism for resolving conflicts in firewalls.

In this paper, we represents a novel anomaly management framework for firewalls based on a rule-based segmentation technique to facilitate not only more accurate anomaly detection but also effective anomaly resolution. Moreover, the outputs of prior policy analysis tools [5, 6, 29] are mainly a list of possible anomalies, which does not give system administrators a clear view of the origination of policy anomalies. Since information visualization technique [18] enables users to explore, analyze, reason and explain abstract information by taking advantage of their visual cognition, our policy analysis tool adopts an information visualization technique to facilitate policy analysis. A grid-based visualization technique is introduced to represent outputs of policy anomaly analysis, enabling an efficient anomaly management. The implementation of our visualization-based policy analysis tool called Firewall Anomaly Management Environment (FAME) is discussed as well.

This paper is organized as follows. Section 2 overviews firewall policies and the anomalies in such policies. Section 3 presents an anomaly representation technique based on packet space. In Section 4, we articulate our policy anomaly management framework. In Section 5, we address the implementation details of FAME followed by the related work in Section 6. Section 7 concludes this paper and discusses our future directions.

## 2. PRELIMINARIES

### 2.1 Overview of Firewall Policies

A firewall policy consists of a sequence of rules that define the actions performed on packets that satisfy certain conditions. The rules are specified in the form of $\langle condition, action\rangle$. A $condition$ in a rule is composed of a set of fields to identify a certain type of packets matched by this rule. Five fields are most commonly used in a rule's $condition$: protocol type, source IP, source port, destination IP and destination port. [1] Those fields are either a single value or a finite interval of non-negative integers. An $action$ in a rule describes the corresponding action performed on the matched packets and typically takes the value "allow", which permits the packets passing through the firewall, or "deny", which leads to the packets to be blocked.

A packet matches a rule if and only if the header information of the packet satisfies all fields in the rule. Upon finding a matching rule, the corresponding decision for the packet is derived. A firewall policy with a sequence of rules typically follows a *first-match* semantic to evaluate a packet: the decision of the first matching rule is applied to the packet. If there is no matching rule that could be found in the firewall policy, a default action is performed. Most firewalls utilize "deny" as the default action, implying every packet that could not be matched by any rules will be denied.

Table 1 shows an example of a firewall policy, which includes five firewall rules $r_1$, $r_2$, $r_3$, $r_4$ and $r_5$. Note that the symbol "*" utilized in firewall rules denotes a domain range. For instance, a single "*" appearing in the IP address field represents an IP address range from 0.0.0.0 to 255.255.255.255.

### 2.2 Anomalies in Firewall Policies

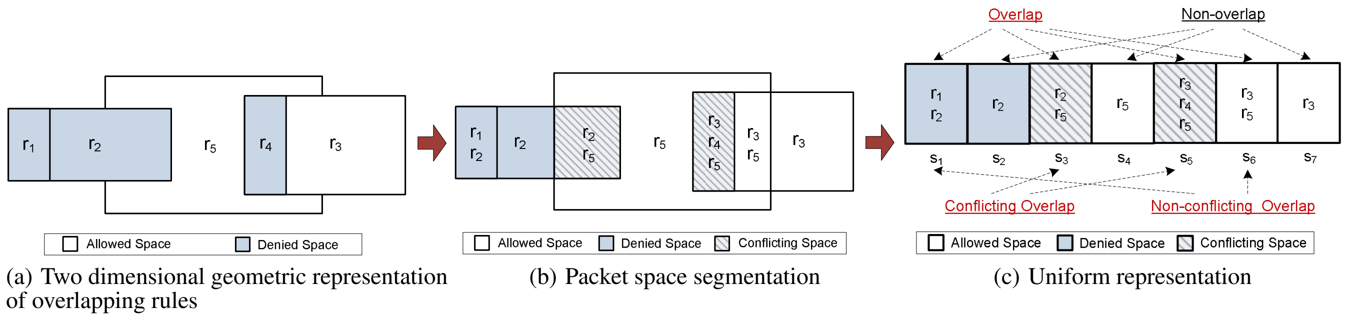
Two rules in a firewall policy may overlap, which means one packet may match both rules. Moreover, two rules in a firewall may conflict, implying that those two rules not only overlap each other but also take different actions. Policy conflicts may lead to both security problems (e.g. allowing malicious traffic) and availability problems (e.g. denying legitimate traffic), and policy redundancies will affect the performance of a firewall. A comprehensive classification of policy anomalies (misconfigurations) has been articulated by several related work [6, 29]. Following existing classification, we summarize policy anomalies as follows:

1. *Shadowing*: A rule can be shadowed by one or a set of preceding rules that match all the packets which also match the shadowed rule, while they perform a different action. In this case, all the packets that one rule intends to deny (accept) can be accepted (denied) by previous rule(s), thus the shadowed rule will never be taken effect. In Table 1, $r_4$ is shadowed by $r_3$ because $r_3$ allows every TCP packet coming from any port of 10.1.1.* to the port 25 of 192.168.1.*, which is supposed to be denied by $r_4$.

2. *Generalization*: A rule is a generalization of one or a set of previous rules if a subset of the packets matched by this rule is also matched by the preceding rule(s) but taking a different action. For example, $r_5$ is a generalization of $r_4$ in Table 1. These two rules indicate that all the packets from 10.1.1.* are allowed, except TCP packets from 10.1.1.* to the port 25 of 192.168.1.*. Note that, as we discussed earlier, generalization might not be an error.

3. *Correlation*: One rule is correlated with other rules, if a rule intersects with others but defines a different action. In this

[1]Some firewalls may occasionally utilize other fields, such as IP TOS (Type of Service) and TTL (Time to Live), which increase the dimensionality of the rule conditions but do not affect the proposed approach in this paper.



(a) Two dimensional geometric representation of overlapping rules (b) Packet space segmentation (c) Uniform representation

**Figure 1: Packet space representation derived from the example policy.**

case, the packets matched by the intersection of those rules may be permitted by one rule, but denied by others. In Table 1, $r_2$ correlates with $r_5$, and all UDP packets coming from any port of 10.1.1.* to the port 53 of 172.32.1.* match the intersection of these rules. Since $r_2$ is a preceding rule of $r_5$, every packet within the intersection of these rules is denied by $r_2$. However, if their positions are swapped, the same packets will be allowed.

4. *Redundancy*: A rule is redundant if there is another same or more general rule available that has the same effect. For example, $r_1$ is redundant with respect to $r_2$ in Table 1, since all UDP packets coming from any port of 10.1.2.* to the port 53 of 172.32.1.* matched with $r_1$ can match $r_2$ as well with the same action.

Anomaly detection algorithms and corresponding tools were also introduced by [6, 29]. However, prior work only treated a policy conflict as an inconsistent relation between *one* rule and other rules. Given a more general definition on policy conflict as shown in Definition 1, we believe that identifying policy conflicts should always consider a firewall policy as a whole piece, and precise indication of the rule set involved in a conflict is critical for effectively resolving the conflict.

DEFINITION 1. ***(Policy Conflict)****. A policy conflict pc in a firewall F is associated with a unique set of conflicting firewall rules* `cr`*=*{$r_1, ..., r_k$}*, which can derive a common network packet space. All packets within this space can match exactly the same set of firewall rules, where at least two rules have different actions:* `Allow` *and* `Deny`*.*

Similarly, we give a more general definition for rule redundancy in firewall policies as follows:

DEFINITION 2. ***(Rule Redundancy)****. A rule r in a firewall F is redundant if removing r from F fulfills that the network packet space derived from the new firewall $F'$ is equal to the network packet space defined by F. That is, F and $F'$ satisfy following equations: $S^A_F = S^A_{F'}$ and $S^D_F = S^D_{F'}$, where $S^A$ and $S^D$ denote* allowed *and* denied *network packet spaces, respectively.*

## 3. ANOMALY REPRESENTATION BASED ON PACKET SPACE

### 3.1 Packet Space Segmentation and Classification

As we discussed in Section 2.2, existing anomaly detection methods could not accurately point out the anomaly portions caused by a set of overlapping rules. In order to precisely identify policy anomalies and enable a more effective anomaly resolution, we adopt *a rule-based segmentation technique*, which can convert a list of rules into a set of disjoint network packet spaces. This technique has been recently introduced to deal with several research problems such as network traffic measurement [30], firewall testing [14] and optimization [15, 23]. Inspired by those successful applications, we adopt this technique for the purpose of firewall anomaly analysis.

To facilitate the correct interpretation of analysis results, a concise and intuitive representation method is necessary. For the purposes of brevity and understandability, we employ a two dimensional geometric representation for each packet space derived from firewall rules. Note that a firewall rule typically utilizes five fields to define the rule condition, thus a complete representation of packet space should be multi-dimensional. Figure 1(a) gives the two dimensional geometric representation of packet spaces derived from the example policy shown in Table 1. We utilize colored rectangles to denote two kinds of packet spaces: *allowed space* (white color) and *denied space* (grey color), respectively. In this example, there are two allowed spaces representing rules $r_3$ and $r_5$, and three denied spaces depicting rules $r_1$, $r_2$ and $r_4$.

Two spaces overlap when the packets matching two corresponding rules intersect. For example, $r_5$ overlaps with $r_2$, $r_3$ and $r_4$, respectively. An overlapping relation may involve multiple rules. In order to clearly represent all identical packet spaces derived from a set of overlapping rules, we adopt the rule-based segmentation technique to divide an entire packet space into a set of pairwise disjoint segments. [2] We classify the policy segments as follows: *non-overlapping* segment and *overlapping* segment, which are further divided into *conflicting overlapping* segment and *non-conflicting overlapping* segment. Each *non-overlapping* segment associates with one unique rule and each *overlapping* segment is related to a set of rules, which may conflict with each other (*conflicting overlapping* segment) or have the same action (*non-conflicting overlapping* segment). Figure 1(b) demonstrates the segments of packet spaces derived from the example policy. Since the size of segment representation does not give any specific benefits in resolving policy anomalies, we further present a uniform representation of space segments in Figure 1(c). We can notice that *seven* unique disjoint segments are generated. Three policy segments $s_2$, $s_4$ and $s_7$ are *non-overlapping* segments. Other policy segments are *overlapping* segments, including two *conflicting overlapping* segments $s_3$ and $s_5$, and two *non-conflicting overlapping* segments $s_1$ and $s_6$.

### 3.2 Grid Representation of Policy Anomaly

To enable an effective anomaly resolution, complete and accu-

[2]The detail of an algorithm for the packet space segmentation is given in [1].

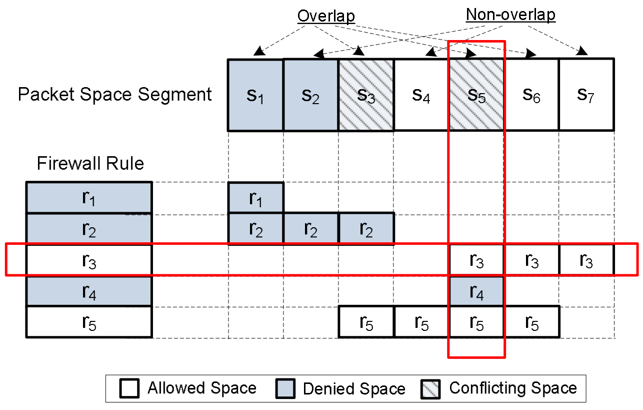
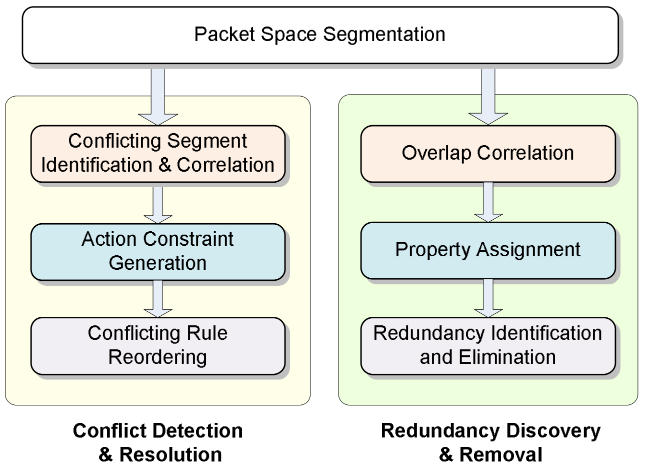


**Figure 2: Grid representation of policy anomaly.**



Packet Space Segmentation
Conflicting Segment Identification & Correlation
Action Constraint Generation
Conflicting Rule Reordering
Overlap Correlation
Property Assignment
Redundancy Identification and Elimination
Conflict Detection & Resolution
Redundancy Discovery & Removal

**Figure 3: Policy anomaly management framework.**

rate anomaly diagnosis information should be represented in an intuitive way. When a set of rules interacts, one overlapping relation may be associated with several rules. Meanwhile, one rule may overlap with multiple other rules and can be involved in a couple of overlapping relations (overlapping segments). Different kinds of segments and associated rules can be viewed in the uniform representation of anomalies (Figure 1(c)). However, it is still difficult for an administrator to figure out how many segments one rule is involved in. To address the need of a more precise anomaly representation, we additionally introduce a grid representation that is a matrix-based visualization of policy anomalies, in which space segments are displayed along the horizontal axis of the matrix, rules are shown along the vertical axis, and the intersection of a segment and a rule is a grid that displays a rule's subspace covered by the segment.

Figure 2 shows a grid representation of policy anomalies for our example policy. We can easily determine which rules are covered by a segment, and which segments are associated with a rule. For example, as shown in Figure 2, we can notice that a conflicting segment $s_5$, which points out a conflict, is related to a rule set consisting of three conflicting rules $r_3$, $r_4$ and $r_5$ (highlighted with a horizontal red rectangle), and a rule $r_3$ is involved in three segments $s_5$, $s_6$ and $s_7$ (highlighted with a vertical red rectangle). Our grid representation provides a better understanding of policy anomalies to system administrators with an overall view of related segments and rules.

## 4. ANOMALY MANAGEMENT FRAMEWORK

Our anomaly management framework is composed of two core functionalities: *conflict detection and resolution*, and *redundancy discovery and removal*, as depicted in Figure 3. Both functionalities are based on the rule-based segmentation technique. For conflict detection and resolution, conflicting segments are identified in the first step. Each conflicting segment associates with a policy conflict and a set of conflicting rules. Also, the correlation relationships among conflicting segments are identified and conflict correlation groups are derived. Policy conflicts belonging to different conflict correlation groups can be resolved separately, thus the searching space for resolving conflicts is reduced by the correlation process. The second step generates action constraints for each conflicting segment by examining the characteristics of each conflicting segment. A strategy-based method is introduced for generating action constraints. The details of action constraints are discussed in Section 4.2. The third step utilizes a reordering algorithm, which is a combination of a permutation algorithm and a greedy algorithm, to discover a near-optimal conflict resolution solution for policy conflicts. Regarding redundancy discovery and removal, segment correlation groups are first identified. Then, the process of property assignment is performed to each rule's subspaces. Consequently, redundant rules are identified and eliminated.

### 4.1 Correlation of Packet Space Segment

Technically, one rule may get involved in multiple policy anomalies. In this case, resolving one anomaly in an isolated manner may cause the unexpected impact on other anomalies. Similarly, we cannot resolve a conflict individually by only reordering conflicting rules associated with one conflict without considering possible impacts on other conflicts. On the other hand, it is also inefficient to deal with all conflicts together by reordering all conflicting rules simultaneously. Therefore, it is necessary to identify the dependency relationships among packet space segments for efficiently resolving policy anomalies.

**Algorithm 1**: Segment Correlation

```
Input: A set of segments, $C$.
Output: A set of groups for correlated segments, $G$.
1  $G.New()$;
2  foreach $s \in S$ do
3      $R \longleftarrow GetRule(s)$;
4      foreach $g \in G$ do
5          foreach $s' \in GetSegment(g)$ do
6              $R'.Append(GetRule(s'))$;
7          if $R \cap R' \neq \varnothing$ then
8              $g.Append(s)$;
9          else
10             $G.NewGroup().Append(s)$;
11 return $G$;
```

Figure 4 shows an example of segment correlation. [3] Suppose we add three new rules $r_6$, $r_7$ and $r_8$ in the example policy shown

[3]Note that for conflict resolution we only need to examine the correlation relations among conflicting segments.

in Table 1. Several rules in this firewall policy are involved in multiple anomalies. For example, $r_2$ is associated with three segments $s_1$, $s_2$ and $s_3$. Also, we can identify $r_3$, $r_5$, $r_6$ and $r_7$ are also associated with multiple segments. Assume we need to resolve the conflict related to a conflicting segment $s_3$ by reordering associated conflicting rules, $r_2$ and $r_5$. The position change of $r_2$ and $r_5$ would also affect other segments, $s_1$, $s_2$, $s_4$, $s_5$ and $s_6$. Thus, a dependency relationship among those segments can be derived. We cluster such segments with a dependency relationship as a group called *correlation group*. The pseudocode of an algorithm for identifying correlation groups is given in Algorithm 1. Applying this algorithm to our example, two correlation groups, $group1$ and $group2$, can be identified as shown in Figure 4: $group1$ contains seven segments and a rule set with five elements ($r_1$, $r_2$, $r_3$, $r_4$ and $r_5$); and $group2$ includes three segments and three associated rules, $r_6$, $r_7$ and $r_8$.

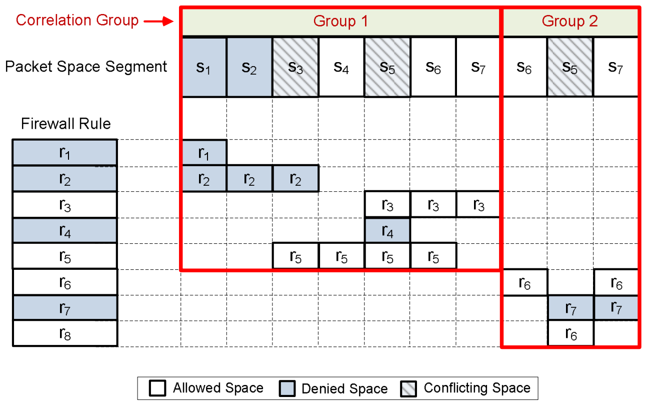


**Figure 4: Example of segment correlation.**

The major benefit of generating correlation groups for the anomaly analysis is that anomalies can be examined within each group independently, because all correlation groups are independent of each other. Especially, the searching space for reordering conflicting rules in conflict resolution can be significantly lessened and the efficiency of resolving conflicts can be greatly improved.

## 4.2 Action Constraint Generation and Rule Reordering for Conflict Resolution

Each conflicting segment indicates a policy conflict as well as a set of conflicting rules involved in the conflict. Once conflicts are identified, a possible way for a system administrator to resolve conflicts is to manually change the conflicting rules. However, as we addressed in Section 1, resolving all conflicts manually is a tedious task and even impractical due to the complicated nature of policy conflicts. Thus, a practical and effective method to resolve a policy conflict is to determine which rule should take precedence when a network packet is matched by a set of rules involved in the conflict. In order to utilize the existing first-match conflict resolution mechanism implemented in common firewalls, the rule expected to take precedence needs to be moved to the first-match rule.

Our conflict resolution mechanism introduces that an *action constraint* is assigned to each conflicting segment. Then, to resolve a conflict, we only assure that the action taken for each packet within the conflicting segment can satisfy the corresponding action constraint. A key feature of this solution is that we do not need to move a rule expected to take precedence to the first-match rule at all times. Any rule associated with the conflict on the same action (as a rule with the precedence) can be moved to the first-match rule, guaranteeing the same effect with respect to the conflict resolution. Thus, it is doable to obtain an optimal solution for conflict resolution.

DEFINITION 3. *(Action Constraint). An action constraint* `ac` *for a conflicting segment* `cs` *defines a desired action (either* `Allow` *or* `Deny`*) that the firewall policy should take when any packet in the conflicting segment comes to the firewall.*

### 4.2.1 Action Constraint Generation

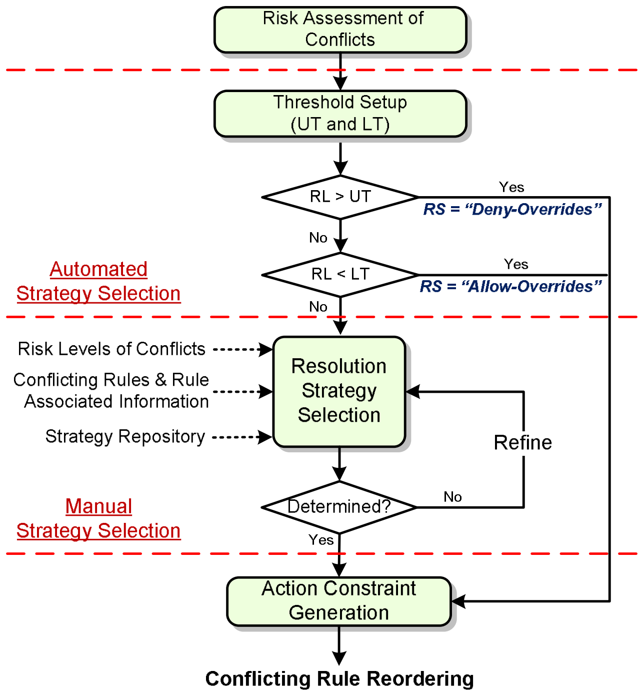

Risk Assessment of Conflicts
Threshold Setup (UT and LT)
RL > UT
Yes
RS = "Deny-Overrides"
No
Automated Strategy Selection
RL < LT
Yes
RS = "Allow-Overrides"
No
Risk Levels of Conflicts
Conflicting Rules & Rule Associated Information
Strategy Repository
Resolution Strategy Selection
Refine
Manual Strategy Selection
Determined?
No
Yes
Action Constraint Generation
Conflicting Rule Reordering

**Figure 5: Strategy-based conflict resolution.**

To generate action constraints for conflicting segments, we propose a strategy-based conflict resolution method, which generates action constraints with the help of effective resolution strategies based on the minimal interaction with system administrators. Figure 5 shows the main processes of this method, which incorporates both *automated* and *manual* strategy selections.

Once conflicts in a firewall policy are discovered and conflict correlation groups are identified, the risk assessment for conflicts is performed. The risk levels of conflicts are in turn utilized for both automated and manual strategy selections. A basic idea of automated strategy selection is that a risk level of a conflicting segment is used to directly determine the expected action taken for the network packets in the conflicting segment. If the risk level is very high, the expected action should deny packets considering the protection of network perimeters. On the contrary, if the risk level is quite low, the expected action should allow packets to pass through the firewall so that the availability and usage of network services cannot be affected. Thus, conflict resolution strategies ($RS$) can be generated automatically for *partial* conflict segments by comparing the risk levels with two thresholds, upper threshold ($UT$) and lower threshold ($LT$), which can be set by system administrators in advance. If a risk level of a conflicting segment is between the upper threshold and the lower threshold, system administrators need to examine the characteristics of each conflict, and manually select appropriate strategies for resolving the conflict, considering

**Table 2: Constraint generation from conflict resolution strategy**

| Strategy | Effect | Action Constraint |
|---|---|---|
| Deny-overrides | Deterministic | Action = "deny" |
| Allow-overrides | Deterministic | Action = "allow" |
| Specificity-overrides | Nondeterministic | Action of the most specific rule |
| High-majority-overrides | Nondeterministic | Action of the rules with greater number than the opposite rules |
| Recency-overrides | Nondeterministic | Action of the rule with the highest authority level |
| High-authority-overrides | Nondeterministic | Action of the newest rule |

network situations (e.g., risk levels of conflicts) and contexts associated with conflicting rules (e.g., priorities, creation time, authors, and so on). Thus, a fine-grained conflict resolution can be carried out with human cognition via the interaction facility with system administrators. Since some strategies may be nondeterministic and could not generate a concrete action constraint when applying to a conflict, system administrators need to adjust the strategy assignments accordingly. Once each conflicting segment has been assigned appropriate conflict resolution strategies, action constraints can be generated based on the assigned strategies. Table 2 summarizes the effect of different conflict resolution strategies utilized in our mechanism as well as the constraint generation from those strategies.

Risk (security) levels are determined based on the vulnerability assessment of the protected network. We have recently seen a number of attempts for qualitatively measuring risks in a network [13, 16, 25, 26]. In our work, we adopt the Common Vulnerability Scoring System (CVSS) [22] as an underlying security metrics for risk evaluation. Two major factors, *exploitability of vulnerability* (reflecting the likelihood of exploitation) and *severity of vulnerability* (representing the potential damage of exploitation), are utilized to evaluate the risk level of a network system. Beside those two factors, another important factor in determining the criticality of an identified security problem is *asset importance value*. Normally, system administrators place a higher priority on defending critical servers than non-critical PCs. Similarly, some machines are more valuable than others. We use *asset importance value* to represent a service's inherent value to network attackers or system administrators. Since the CVSS *base score* can cover both *exploitability of vulnerability* and *severity of vulnerability* factors, we incorporate the CVSS *base score* and *asset importance value* to compute the risk value for each vulnerability. To calculate the risk level (*RL*) of each conflicting segment, we accumulate all risk values of the vulnerabilities covered by a conflicting segment. In practice, system administrators may mainly concern about the security risk of each vulnerability in their network. In this case, an *average* risk value needs to be calculated as the risk level of a conflicting segment. In order to accommodate both requirements for risk evaluation of a conflicting segment. We introduce a generic equation for the risk level calculation as follows:

$$RL(cs) = \frac{\sum_{v \in V(cs)}(CVSS(v) \times IV(s))}{\alpha \times |V(cs)|} \tag{1}$$

Where, $V(cs)$ is a function to return all vulnerabilities that are contained in a conflicting segment $cs$; $CVSS(v)$ is a function to return the CVSS *base score* of vulnerability $v$; and $IV(s)$ is a function to return the importance value (range from 0.0 to 1.0) of service $s$. Also, we incorporate a coefficient factor $\alpha$ ($\frac{1}{|V(cs)|} \leq \alpha \leq 1$) that allows system administrators to express their preferences in choosing *average* or *overall* risk value to measure the risk of each conflicting segment.

### 4.2.2 *Rule Reordering*

A naive way to find an optimal solution is to exhaustively search all permutations of correlated conflicting rules. We then compute a resolving score for each permutation by counting how many action constraints can be satisfied, and select the permutation with the maximum resolving score as the best solution for a conflict resolution. However, a key limitation of using the permutation algorithm is its computational complexity which is $O(n!)$. Even though the search space can be significantly reduced by applying our correlation scheme, the number of correlated conflicting rules may still be large, leading to the permutation algorithm unapplicable. That is, the permutation algorithm can help us identify an optimal solution for a conflict resolution, but it is inefficient for the firewall policies with a larger number of conflicting rules within some conflict correlation groups. As an approximation algorithm, the greedy conflict resolution algorithm is thus more efficient in resolving conflicts, but can only find a near-optimal solution. The detail of our greedy algorithm is omitted in this paper due to the space limitation but can be found in [1].

In order to achieve the objective of resolving conflicts effectively and efficiently, our conflict resolution mechanism adopts a combination algorithm incorporating features from both permutation and greedy algorithms. A threshold $\mathcal{N}$ for selecting a suitable rule reordering algorithm to resolve a conflict can be predefined in the combination algorithm. When the number of conflicting rules is less than $\mathcal{N}$, the permutation algorithm is utilized for resolving conflicts. Otherwise, the greedy algorithm is applied to resolve conflicts.

## 4.3 Property Assignment for Redundancy Removal

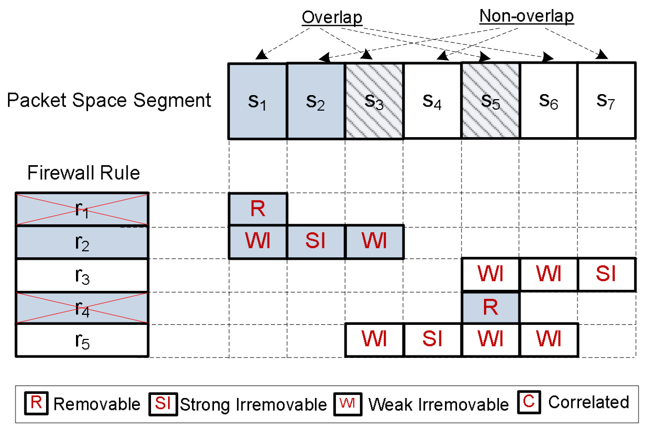


**Figure 6: Example of property assignment for redundancy removal.**

In this step, every rule subspace covered by a policy segment is assigned with a property. Four property values, *removable* (R), *strong irremovable* (SI), *weak irremovable* (WI) and *correlated* (C), are defined to reflect different characteristics of each rule subspace. *Removable* property is used to indicate that a rule subspace is removable. In other words, removing such a rule subspace does not make any impact on the original packet space of an associated policy. *Strong irremovable* property means that a rule subspace cannot be removed because the effect of corresponding policy segment can be decided only by this rule. *Weak irremovable* property is assigned to a rule subspace when any subspace belonging to the same rule has *strong irremovable* property. That means a rule subspace becomes irremovable due to the reason that other portions of this rule cannot be removed. *Correlated* property is assigned to multiple rule subspaces covered by a policy segment, if the effect of this policy segment can be determined by any of these rules. We next introduce three processes to perform the property assignments to all of rule subspaces within the segments of a firewall policy, considering different categories of policy segments discussed in Section 3.1.

**Process1:** *Property assignment for the rule subspace covered by a non-overlapping segment.* A non-overlapping segment contains only one rule subspace. Thus, this rule subspace is assigned with *strong irremovable* property. Other rule subspaces associated with the same rule are assigned with *weak irremovable* property, except for the rule subspaces that already have *strong irremovable* property.

**Process2:** *Property assignment for rule subspaces covered by a conflicting segment.* The first rule subspace covered by the conflicting segment is assigned with *strong irremovable* property. Other rule subspaces in the same segment are assigned with *removable* property. Meanwhile, other rule subspaces associated with the same rule are assigned with *weak irremovable* property except for the rule subspaces with *strong irremovable* property.

**Process3:** *Property assignment for rule subspaces covered by a non-conflicting overlapping segment.* If any rule subspace has been assigned with *weak irremovable* property, other rule subspaces without *any irremovable* property are assigned with *removable* property. Otherwise, all subspaces within the segment are assigned with *correlated* property.

The pseudocode of the algorithm for property assignment is shown in Algorithm 2. Figure 6 shows the result of applying our property assignment algorithm, which performs three property assignment processes in sequence, to the example presented in Figure 2. We can easily identify that $r_1$ and $r_4$ are *removable* rules, where all subspaces are with *removable* property. [4]

**Algorithm 2**: Property Assignment of Rule Subspaces for Redundancy Discovery

```
Input: A set of authorization space segment $S$ of a policy $p$.
Output: A set of rules $R$ with assigned property.
1  /* Process 1: handling non-overlapping segment */
2  foreach $s \in GetNonOverlapSegment(S)$ do
3      $sp \longleftarrow GetSegSubspace(s)$;
4      $sp.Property \longleftarrow$ SI ;
5      AssginWI($FirstSubspace(sp)$);
6  /* Process 2: handling conflicting overlapping segment */
7  foreach $s \in GetConflictSegment(S)$ do
8      $FirstSubspace(s).Property \longleftarrow$ SI ;
9      AssginWI($FirstSubspace(s)$);
10     foreach $sp' \in GetNonFirstSubspace(s)$ do
11         $sp'.Property \longleftarrow$ R ;
12 /* Process 3: handling non-conflicting overlapping segment */
13 foreach $s \in GetNonConflictOverlapSegment(S)$ do
14     $SP \longleftarrow GetSegSubspace(s)$;
15     if $AnyWI(SP) =$ true then
16         foreach $sp' \in WithoutWI(SP)$ do
17             $sp.Property \longleftarrow$ R ;
18     else
19         foreach $sp \in SP$ do
20             $sp.Property \longleftarrow$ C ;
21 /* Assigning week irremovable property */
22 AssignWI($sp$);
23 $SP \longleftarrow GetRuleSubspace(GetRule(sp))$;
24 foreach $sp' \in SP$ do
25     if $sp' \neq sp$ and $sp'.Property \neq SI$ then
26         $sp'.Property \longleftarrow$ WI ;
```

## 5. FIREWALL ANOMALY MANAGEMENT ENVIRONMENT: FAME

Our framework is realized as a proof-of-concept prototype called FAME. Figure 7 shows a high level architecture of FAME with two levels. The upper level is the visualization layer, which visualizes the results of policy anomaly analysis to system administrators. Two visualization interfaces, *policy conflict viewer* and *policy redundancy viewer*, are designed to manage policy conflicts and redundancies, respectively. The lower level of the architecture provides underlying functionalities addressed in our policy anomaly management framework and relevant resources including rule information, strategy repository, network asset information, and vulnerability information. In this section, we first discuss a few issues related to the implementation of our management framework. Then, we articulate the features of our visualization interfaces.

### 5.1 Implementation of Anomaly Management Framework in FAME

FAME was implemented in Java. Based on our policy anomaly management framework, it consists of six components: segmentation module, correlation module, risk assessment module, action constraint generation module, rule reordering module, and property assignment module. The segmentation module takes firewall policies as an input and identifies the packet space segments by partitioning the packet space into disjoint subspaces. FAME utilizes Ordered Binary Decision Diagrams (BDDs) [5] to represent firewall rules and perform various set operations, such as unions ($\cup$), intersections ($\cap$), and set differences ($\setminus$), required by the segmentation algorithm. A BDD library called BuDDy [2] is employed by FAME. Once the segmentation of packet space is identified, FAME further identifies different kinds of segments and corresponding correlation groups. In risk assessment module, Nessus [3] is utilized as a *vulnerability scanner* to identify the vulnerabilities within a conflicting segment. Network address space of each conflicting

[4]Note that we may need to further examine rules, which contain some subspaces with *correlated* property, through a correlation break mechanism.

[5]BDD has been demonstrated as an efficient data structure to deal with a variety of network configuration analysis [7, 29].

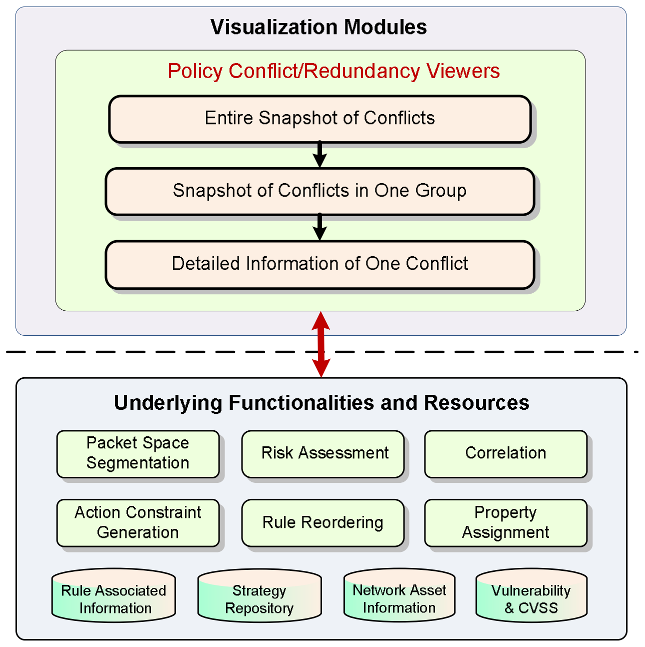

Visualization Modules
Policy Conflict/Redundancy Viewers
Entire Snapshot of Conflicts
Snapshot of Conflicts in One Group
Detailed Information of One Conflict
Underlying Functionalities and Resources
Packet Space Segmentation
Risk Assessment
Correlation
Action Constraint Generation
Rule Reordering
Property Assignment
Rule Associated Information
Strategy Repository
Network Asset Information
Vulnerability & CVSS

**Figure 7: Architecture of FAME.**

segment is fed into Nessus to get the vulnerability information of a given address space. Nessus produces the vulnerability information in a "nbe" format. The risk assessment module utilizes tissynbe script [4] to parse the Nessus results and store the vulnerability information to a *vulnerability database*. A *risk calculator* retrieves vulnerability information, such as CVSS base score and asset importance value, to calculate the risk level of each conflicting segment. The action constraint generation module takes conflicting segments as an input and generates action constraints for each conflicting segment. Action constraints are generated based on strategies assigned to each conflicting segment. The rule reordering module takes conflict correlation groups and action constraints of conflicting segments as inputs and generates optimal or near-optimal conflict resolution for policy conflicts using a combined reordering algorithm in our framework. The property assignment module takes segment correlation groups as inputs and automatically assigns corresponding properties to each rule subspace covered by policy segments. The assigned properties are in turn utilized to identify redundant rules.

## 5.2 Visualization Interfaces of FAME

FAME provides two policy viewers to visualize the outputs of policy conflict analysis and policy redundancy analysis. Each viewer offers two kinds of visualization interfaces: one interface shows an entire snapshot of all anomalies; another interface shows a partial snapshot only containing anomalies within one correlation group.

Figure 8 depicts interfaces of FAME conflict viewer. The grid representation shows accurately how a set of rules interacts with each other. FAME conflict viewer has the ability to show an overview of the entire conflicts as well as portions of the policy conflicts, that need to be examined in depth for conflict resolution, based on correlation groups. As illustrated in Figure 8 (a), all conflicting segments and conflict correlation groups are displayed along the horizontal axis at the top of the interface. All conflicting rules are shown along the vertical axis at the left of the interface. Each grid cell represents a rule's subspace. In our interface, the icons for conflicting segments indicate four different states with respect to conflicting resolution. One icon represents a conflicting segment with the state of strategy-unassigned. Two other icons indicate conflicting segments with the state of strategy-assigned with "Allow" action constraint and strategy-assigned with "Deny" action constraint, respectively. The fourth icon indicates a conflicting segment with the state of conflict-unresolved. In addition, this interface allows an administrator to set the risk level thresholds for automatically assigning strategies.

Clicking on a group name box of the interface in Figure 8 (a), another window as shown in Figure 8 (b) is displayed with the targeted conflicts that an administrator needs to examine and resolve. In this interface, the number of visible entities is reduced to display conflicting segments in only one correlation group and a list of conflicting rules associated with this group. This significantly eliminates administrators' workloads in resolving conflicts by highlighting conflicts within a group. For resolution strategy selection, the administrator needs to further examine rule information for selecting suitable strategies for each conflicting segment. When the administrator clicks the icon of a conflicting segment, the detailed information related to the conflict is displayed in a window as shown in Figure 8 (c). [6]

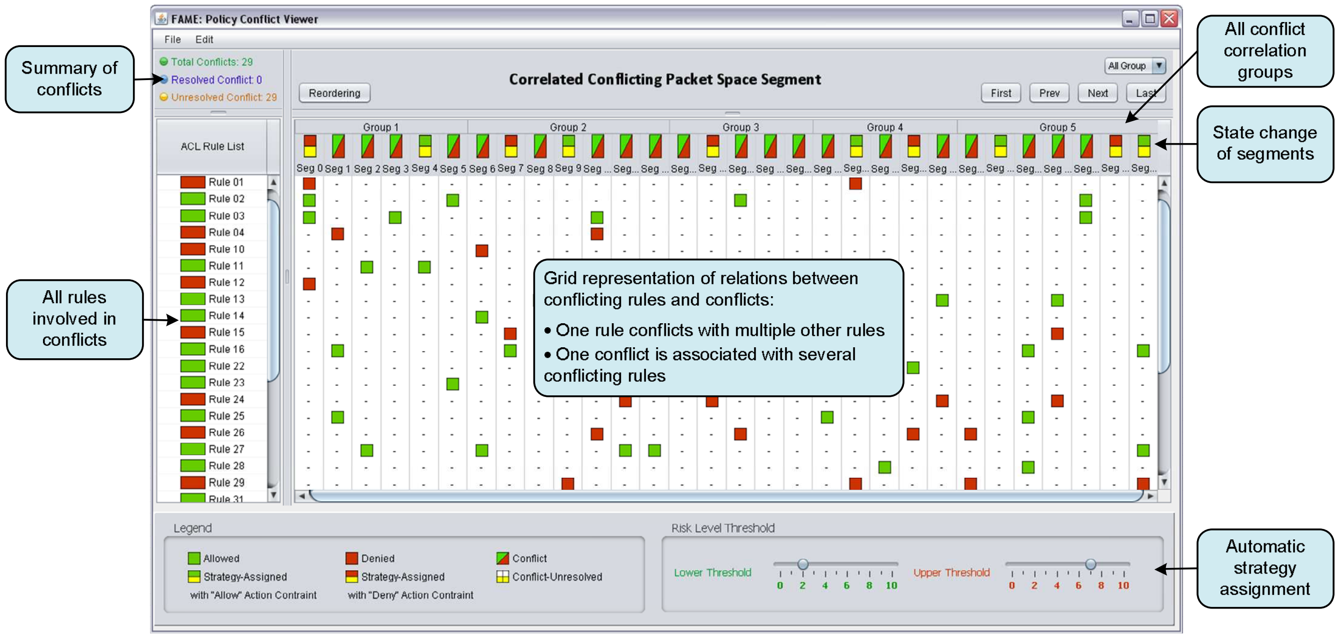
We believe that the visualization interfaces in FAME will assist administrators in viewing the outputs from policy anomaly analysis and facilitate a more effective and efficient anomaly resolution with following contributions: First, FAME helps the administrator minimize the portions of the policy that they need to examine at any given time. Second, the grid representation of policy anomalies offers an intuitive and succinct view of the interactions of overlapping rules and enables administrators to better understand policy anomalies.

## 6. RELATED WORK

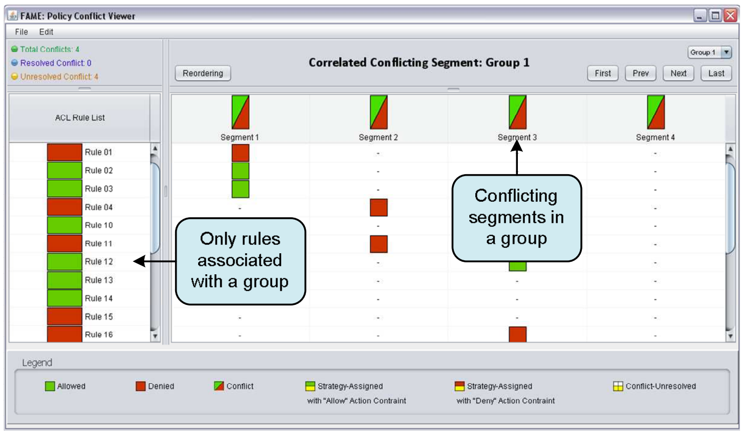
There exist a number of algorithms and tools designed to assist system administrators in managing and analyzing firewall policies [5, 6, 8, 9, 20, 29]. Lumeta [28] and Fang [21] allow user queries for the purpose of analysis and management of firewall policies. Essentially, they introduced lightweight firewall testing tools but could not provide a comprehensive examination of policy misconfigurations. Several other approaches presenting policy analysis tools with the goal of detecting policy anomalies are closely related to our work. Al-Shaer and Hamed [5, 6] designed a tool called Firewall Policy Advisor to detect pairwise anomalies in firewall rules. A corresponding policy visualization tool called PolicyVis [27] was developed as well. Yuan et al. [29] presented FIREMAN, a toolkit to check for misconfigurations in firewall policies through static analysis. As we discussed previously, our tool, FAME, overcomes the limitations of those tools by conducting a complete anomaly detection and providing more accurate anomaly diagnosis information. In particular, the key distinction of FAME is its capability to perform an effective conflict resolution, which has been ruled out in other firwall policy analysis tools.

There are several interfaces that have been developed to assist users in creating and manipulating security policies. Expandable Grid is a tool for viewing and authoring access control policies [24]. The representation in Expandable Grids is a matrix with subjects shown along the rows, resources shown along the columns, and effective accesses for the combinations of subjects and resources in the matrix cells. The SPARCLE Policy Workbench allows policy

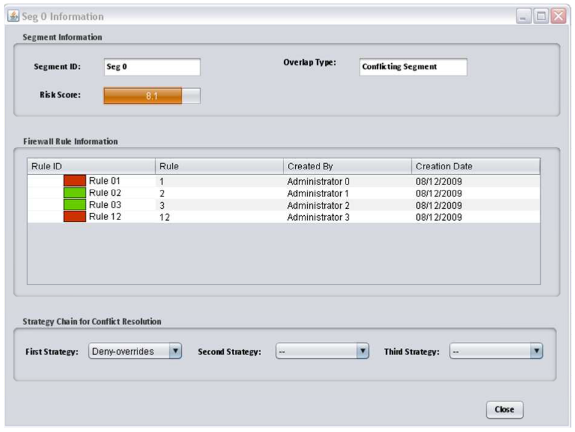
[6]Due to the space limitation, we elide the discussion of policy redundancy viewer in this paper. Our redundancy viewer was also developed in a similar fashion.


FAME: Policy Conflict Viewer
File Edit
Total Conflicts: 29
Resolved Conflict: 0
Unresolved Conflict: 29
Reordering
Correlated Conflicting Packet Space Segment
All Group
First Prev Next Last
Group 1 Group 2 Group 3 Group 4 Group 5
ACL Rule List
Legend
Allowed
Strategy-Assigned with "Allow" Action Contraint
Denied
Strategy-Assigned with "Deny" Action Contraint
Conflict
Conflict-Unresolved
Risk Level Threshold
Lower Threshold
Upper Threshold
Summary of conflicts
All rules involved in conflicts
All conflict correlation groups
State change of segments
Automatic strategy assignment
Grid representation of relations between conflicting rules and conflicts:
• One rule conflicts with multiple other rules
• One conflict is associated with several conflicting rules

(a) Conflict Viewer for all conflicts


FAME: Policy Conflict Viewer
File Edit
Total Conflicts: 4
Resolved Conflict: 0
Unresolved Conflict: 4
Reordering
Correlated Conflicting Segment: Group 1
Group 1
First Prev Next Last
ACL Rule List
Segment 1 Segment 2 Segment 3 Segment 4
Only rules associated with a grogup
Conflicting segments in a group
Legend
Allowed
Denied
Conflict
Strategy-Assigned with "Allow" Action Contraint
Strategy-Assigned with "Deny" Action Contraint
Conflict-Unresolved

(b) Conflict Viewer for conflicts in one group



Segment Information

Segment ID: Seg 0

Overlap Type: Conflicting Segment

Risk Score: 8.1

Firewall Rule Information

| Rule ID | Rule | Created By | Creation Date |
|---|---|---|---|
| 🟥 Rule 01 | 1 | Administrator 0 | 08/12/2009 |
| 🟩 Rule 02 | 2 | Administrator 1 | 08/12/2009 |
| 🟩 Rule 03 | 3 | Administrator 2 | 08/12/2009 |
| 🟥 Rule 12 | 12 | Administrator 3 | 08/12/2009 |

Strategy Chain for Conflict Resolution

First Strategy: Deny-overrides

Second Strategy: --

Third Strategy: --

(c) Information of one conflict

**Figure 8: Interface of FAME Conflict Viewer.**

authors to construct policies in a natural language interface, which are in turn translated into machine-readable policies [12]. Even though these tools are useful for authoring access control policies, they cannot effectively represent the results of policy analysis for firewalls.

Firewall policy engineering and design models have also attracted a lot of attention [10, 11, 17, 19]. Firmato [10] supports policy specification with respect to the information of a global entity relationship; Bellovin et al. [11, 19] introduced a distributed firewall model that supports centralized policy specification; and Gouda et al. [17] devised a firewall decision diagram (FDD) to support consistent, complete and compact firewall policy generation. Since anomaly discovery and resolution are indispensable processes in policy design and specification, all of those work are orthogonal to our work.

## 7. CONCLUSIONS

In this paper, we have proposed a novel anomaly management framework that facilitates systematic detection and resolution of firewall policy anomalies. A rule-based segmentation technique was introduced to achieve the goal of effective and efficient anomaly analysis. In addition, we have described an implementation of our anomaly management environment called FAME, clearly demonstrating that our proposed anomaly analysis methodology is practical and useful for system administrators to enable an assurable network management.

Our future work includes usability studies to evaluate functionalities and system requirements of our policy visualization approach with subject matter experts. Also, we would explore how our anomaly management framework and visualization approach can be applied to other types of access control policies including Web access control policies such as XACML-based policies.

## Acknowledgments

This work was partially supported by the grants from National Science Foundation (NSF-IIS-0900970 and NSF-CNS-0831360) and Department of Energy (DE-SC0004308 and DE-FG02-03ER25565).

## 8. REFERENCES

[1] A Systematic Approach for Conflict Resolution in Firewall Policies. Technical Report ASU-SCIDSE-10-2, Arizona State University, Tempe, May 2010. `http://sefcom.asu.edu/confres/confres.pdf`.

[2] Buddy version 2.4. `http://sourceforge.net/projects/buddy`.

[3] TENABLE Network Security. `http://www.nessus.org/nessus`.

[4] Tissynbe.py. `http://www.tssci-security.com/projects/tissynbe_py`.

[5] E. Al-Shaer and H. Hamed. Firewall Policy Advisor for anomaly discovery and rule editing. In *Integrated Network Management, 2003. IFIP/IEEE Eighth International Symposium on*, pages 17–30, 2003.

[6] E. Al-Shaer and H. Hamed. Discovery of policy anomalies in distributed firewalls. In *IEEE INFOCOM*, volume 4, pages 2605–2616, 2004.

[7] E. Al-Shaer, W. Marrero, A. El-Atawy, and K. ElBadawi. Network Configuration in A Box: Towards End-to-End Verification of Network Reachability and Security. In *Proceedings of the 17th IEEE International Conference on Network Protocols (ICNP)*, pages 123–132, 2009.

[8] J. Alfaro, N. Boulahia-Cuppens, and F. Cuppens. Complete analysis of configuration rules to guarantee reliable network security policies. *International Journal of Information Security*, 7(2):103–122, 2008.

[9] F. Baboescu and G. Varghese. Fast and scalable conflict detection for packet classifiers. *Computer Networks*, 42(6):717–735, 2003.

[10] Y. Bartal, A. Mayer, K. Nissim, and A. Wool. Firmato: A novel firewall management toolkit. *ACM Transactions on Computer Systems (TOCS)*, 22(4):381–420, 2004.

[11] S. Bellovin. Distributed firewalls. *Journal of Login*, 24(5):37–39, 1999.

[12] C. Brodie, C. Karat, and J. Karat. An empirical study of natural language parsing of privacy policy rules using the SPARCLE policy workbench. In *Proceedings of the second symposium on Usable privacy and security*, page 19. ACM, 2006.

[13] E. Chew, M. Swanson, K. Stine, N. Bartol, A. Brown, and W. Robinson. Performance measurement guide for information security. *NIST Special Publication*, pages 800–55, 2008.

[14] A. El-Atawy, K. Ibrahim, H. Hamed, and E. Al-Shaer. Policy segmentation for intelligent firewall testing. In *1st Workshop on Secure Network Protocols (NPSec 2005)*, 2005.

[15] A. El-Atawy, T. Samak, E. Al-Shaer, and H. Li. Using online traffic statistical matching for optimizing packet filtering performance. In *IEEE INFOCOM 2007. 26th IEEE International Conference on Computer Communications*, pages 866–874, 2007.

[16] M. Frigault, L. Wang, A. Singhal, and S. Jajodia. Measuring network security using dynamic bayesian network. In *Proceedings of the 4th ACM workshop on Quality of protection*, pages 23–30. ACM, 2008.

[17] M. Gouda and X. Liu. Firewall Design: Consistency, Completeness, and Compactness. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, page 327. IEEE Computer Society, 2004.

[18] I. Herman, G. Melançon, and M. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, pages 24–43, 2000.

[19] S. Ioannidis, A. Keromytis, S. Bellovin, and J. Smith. Implementing a distributed firewall. In *Proceedings of the 7th ACM conference on Computer and communications security*, page 199. ACM, 2000.

[20] A. Liu and M. Gouda. Complete redundancy detection in firewalls. *Data and Applications Security XIX*, pages 193–206, 2005.

[21] A. Mayer, A. Wool, and E. Ziskind. Fang: A firewall analysis engine. In *IEEE Symposium on Security and Privacy*, pages 177–189. IEEE Computer Society, 2000.

[22] P. Mell, K. Scarfone, and S. Romanosky. A complete guide to the common vulnerability scoring system version 2.0. In *Published by FIRST-Forum of Incident Response and Security Teams, June*, 2007.

[23] G. Misherghi, L. Yuan, Z. Su, C.-N. Chuah, and H. Chen. A general framework for benchmarking firewall optimization techniques. *IEEE Transactions on Network and Service Management*, 5(4):227–238, Dec. 2008.

[24] R. Reeder, L. Bauer, L. Cranor, M. Reiter, K. Bacon, K. How, and H. Strong. Expandable grids for visualizing and authoring computer security policies. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1473–1482. ACM, 2008.

[25] M. Sahinoglu. Security meter: A practical decision-tree model to quantify risk. *IEEE security & privacy*, pages 18–24, 2005.

[26] R. Sawilla and X. Ou. Identifying Critical Attack Assets in Dependency Attack Gaphs. In *13th European Symposium on Research in Computer Security (ESORICS)*. Springer, 2008.

[27] T. Tran, E. Al-Shaer, and R. Boutaba. PolicyVis: firewall security policy visualization and inspection. In *Proceedings of the 21st conference on Large Installation System Administration Conference*, pages 1–16. USENIX Association, 2007.

[28] A. Wool. Architecting the lumeta firewall analyzer. In *Proceedings of the 10th conference on USENIX Security Symposium-Volume 10*, page 7. USENIX Association, 2001.

[29] L. Yuan, H. Chen, J. Mai, C. Chuah, Z. Su, P. Mohapatra, and C. Davis. Fireman: A toolkit for firewall modeling and analysis. In *2006 IEEE Symposium on Security and Privacy*, page 15, 2006.

[30] L. Yuan, C. Chuah, and P. Mohapatra. ProgME: towards programmable network measurement. *ACM SIGCOMM Computer Communication Review*, 37(4):108, 2007.