Conflict Classification and Analysis of Distributed Firewall Policies

Ehab Al-Shaer, Hazem Hamed, Raouf Boutaba, and Masum Hasan

Abstract—Firewalls are core elements in network security. However, managing firewall rules, particularly, in multifirewall enterprise networks, has become a complex and error-prone task. Firewall filtering rules have to be written, ordered, and distributed carefully in order to avoid firewall policy anomalies that might cause network vulnerability. Therefore, inserting or modifying filtering rules in any firewall requires thorough intrafirewall and interfirewall analysis to determine the proper rule placement and ordering in the firewalls. In this paper, we identify all anomalies that could exist in a single- or multifirewall environment. We also present a set of techniques and algorithms to automatically discover policy anomalies in centralized and distributed firewalls. These techniques are implemented in a software tool called the "Firewall Policy Advisor" that simplifies the management of filtering rules and maintains the security of next-generation firewalls.

Index Terms—Firewall, packet filter, policy analysis, policy conflict, policy management, security management.

I. INTRODUCTION

WITH THE global Internet connection, network security has gained significant attention in research and industrial communities. Due to the increasing threat of network attacks, firewalls have become important elements not only in enterprise networks but also in small-size and home networks. Firewalls have been the frontier defense for secure networks against attacks and unauthorized traffic by filtering out unwanted network traffic coming from or going to the secured network. The filtering decision is based on a set of ordered filtering rules written based on predefined security policy requirements.

Although deployment of firewall technology is an important step toward securing our networks, the complexity of managing firewall policies might limit the effectiveness of firewall security. In a single firewall environment, the local firewall policy may include *intrafirewall anomalies*, where the same packet may match more than one filtering rule. Moreover, in distributed firewall environments, firewalls might also have *interfirewall*

anomalies when individual firewalls in the same path perform different filtering actions on the same traffic. Therefore, the administrator must give special attention not only to all rule relations in the same firewall in order to determine the correct rule order, but also to all relations between rules in different firewalls in order to determine the proper rule placement in the proper firewall. As the number of filtering rules increases, the difficulty of adding a new rule or modifying an existing one significantly increases. It is very likely, in this case, to introduce conflicting rules such as one general rule shadowing another specific rule, or correlated rules whose relative ordering determines different actions for the same packet. In addition, a typical large-scale enterprise network might involve hundreds of rules that might be written by different administrators in various times. This significantly increases the potential of anomaly occurrence in the firewall policy, jeopardizing the security of the protected network.

Therefore, the effectiveness of firewall security is dependent on providing policy management techniques and tools that network administrators can use to analyze, purify, and verify the correctness of written firewall filtering rules. In this paper, we first provide a formal definition of filtering rule relations, and then identify all anomalies that might exist in any firewall policy in both centralized and distributed firewall environments. We also use a tree-based filtering representation to develop anomaly discovery algorithms for reporting any intrafirewall and interfirewall anomaly in any general network. We finally develop a rule editor to produce anomaly free firewall policies, and greatly simplify adding, removing, and modifying filtering rules. These algorithms and techniques were implemented using Java programming language in a software tool called the "Firewall Policy Advisor." In our previous work [1], [3], we discussed intrafirewall conflict analysis, however, in this paper we mainly focus on the discovery and resolution of interfirewall anomalies.

Although firewall security has been given strong attention in the research community, the emphasis was mostly on the filtering performance issues [15], [27], [29]. On the other hand, a few related works [9], [14] attempt to address only one of the conflict problems which is the rule correlation in filtering policies. Other approaches [4], [13], [16] propose using a high-level policy language to define and analyze firewall policies, and then map this language to filtering rules. Although using such high-level languages might avoid rule anomalies, they are not practical for the most widely used firewalls that contain low-level filtering rules. It is simply because redefining already existing policies using high-level languages require far more effort than just analyzing existing rules using standalone tools such as the Firewall Policy Advisor. In addition, none of the previous work

```
                  Source              Destination
      Protocol  Address    Port     Address      Port  Action

   1: tcp,  140.192.37.20, any,       *.*.*.*,    80,    deny
   2: tcp,  140.192.37.*,  any,       *.*.*.*,    80,  accept
   3: tcp,        *.*.*.*, any, 161.120.33.40,    80,  accept
   4: tcp,  140.192.37.*,  any, 161.120.33.40,    80,    deny
   5: tcp,  140.192.37.30, any,       *.*.*.*,    21,    deny
   6: tcp,  140.192.37.*,  any,       *.*.*.*,    21,  accept
   7: tcp,  140.192.37.*,  any, 161.120.33.40,    21,  accept
   8: tcp,        *.*.*.*, any,       *.*.*.*,   any,    deny
   9: udp,  140.192.37.*,  any, 161.120.33.40,    53,  accept
  10: udp,        *.*.*.*, any, 161.120.33.40,    53,  accept
  11: udp,  140.192.38.*,  any,  161.120.35.*,   any,  accept
  12: udp,        *.*.*.*, any,       *.*.*.*,   any,    deny
```
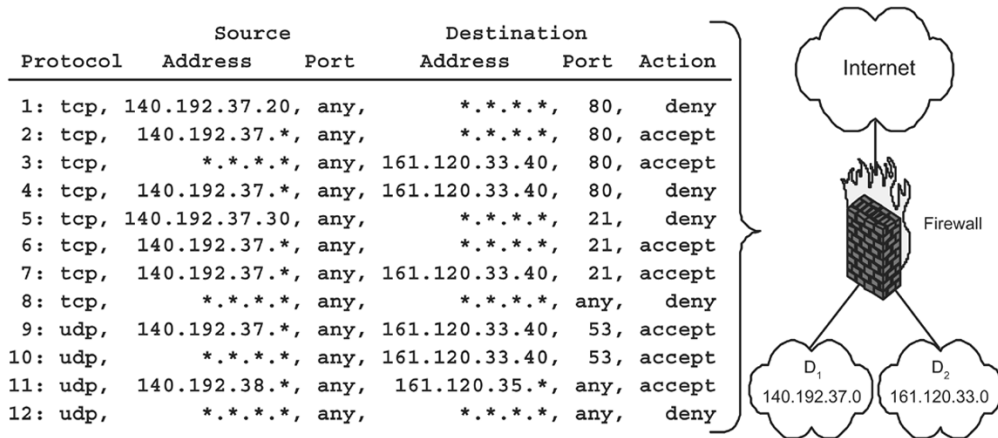
Fig. 1. Example for centralized firewall setup.

has a significant attempt to address anomalies in distributed firewalls. Therefore, we consider our work a significant progress in the area as it offers a novel and comprehensive framework to automate anomaly discovery and rule editing in both centralized and distributed firewalls.

This paper is organized as follows. In Section II, we give an introduction to firewall operation. In Section III, we present our formalization of filtering rule relations. In Section IV, we classify and define policy anomalies in centralized firewalls, and we describe the intrafirewall anomaly discovery algorithm. In Section V, we classify and define policy anomalies in distributed firewalls, and then we describe the interfirewall anomaly discovery algorithm. In Section VI, we describe the techniques for anomaly free rule editing. In Section VII, we show the implementation and evaluation of the Firewall Policy Advisor. In Section VIII, we give a summary of related work. Finally, in Section IX, we show our conclusions and our plans for future work.

II. FIREWALL BACKGROUND

A firewall is a network element that controls the traversal of packets across the boundaries of a secured network based on a specific security policy. A firewall security policy is a list of ordered filtering rules that define the actions performed on packets that satisfy specific conditions. A rule is composed of set of filtering fields (also called network fields) such as protocol type, source IP address, destination IP address, and source port, and destination port, as well as an action field. The filtering fields of a rule represent the possible values of the corresponding fields in actual network traffic that matches this rule. Each network field could be a single value or a range of values. In our work, we require that any field value in one rule cannot partially overlap with a corresponding field value in another rule. Filtering actions are either to *accept*, which permits the packet into or from the secure network, or to *deny*, which causes the packet to be blocked. The packet is permitted or blocked by a specific rule if the packet header information matches all the network fields of this rule. Otherwise, the following rule is examined and the process is repeated until a matching rule is found or the default policy action is performed [6], [7]. In this paper, we assume a "deny" default policy action.

Filtering rule format: It is possible to use any field in Internet protocol (IP), user datagram protocol (UDP), or transmission control protocol (TCP) headers in the rule filtering part, however, practical experience shows that the most commonly used matching fields are: protocol type, source IP address, source port, destination IP address, and destination port [8], [28]. The following is the common format of packet filtering rules in a firewall policy, as shown in the policy example in Fig. 1.

III. MODELING AND REPRESENTATION OF FIREWALL POLICIES

Modeling of firewall rule relations is necessary for analyzing the firewall policy and designing management techniques such as anomaly discovery and policy editing. In this section, we formally describe our model of firewall rule relations, then we describe a tree-based representation for firewall policies.

A. Formalization of Firewall Rule Relations

To be able to build a useful model for filtering rules, we need to determine all the relations that may relate packet filters. In this section, we define all the possible relations that may exist between filtering rules, and we show that no other relation exists. We determine these relations based on comparing the network fields of filtering rules, independent of the rule actions. In the following sections, we consider these relations as well as rule actions in our study of firewall rule conflicts.

Definition 1: Rules R_x and R_y are *completely disjoint* if every field in R_x is not a subset nor a superset nor equal to the corresponding field in R_y. Formally, $R_x \Re_{CD} R_y$ iff

$$\forall i : R_x[i] \not\bowtie R_y[i] \quad \text{where} \bowtie \in \{\subset, \supset, =\}$$
$$\text{and } i \in \{\text{protocol, s\_ip, s\_port, d\_ip, d\_port}\}.$$

Definition 2: Rules R_x and R_y are *exactly matching* if every field in R_x is equal to the corresponding field in R_y. Formally, $R_x \Re_{EM} R_y$ iff

$$\forall i : R_x[i] = R_y[i]$$
$$\text{where } i \in \{\text{protocol, s\_ip, s\_port, d\_ip, d\_port}\}.$$

Definition 3: Rules R_x and R_y are *inclusively matching* if they do not exactly match and if every field in R_x is a subset or equal to the corresponding field in R_y. R_x is called the

subset match, while R_y is called the *superset match*. Formally, $R_x \Re_{IM} R_y$ iff

$$\forall i : R_x[i] \subseteq R_y[i] \text{ and } \exists j \text{ such that} : R_x[j] \neq R_y[j]$$
$$\text{where } i, j \in \{\text{protocol, s\_ip, s\_port, d\_ip, d\_port}\}.$$

For example, in Fig. 1, Rule 1 inclusively matches Rule 2. Rule 1 is the subset match of the relation, while Rule 2 is the superset match.

Definition 4: Rules R_x and R_y are *partially disjoint* (or *partially matching*) if there is at least one field in R_x that is a subset or a superset or equal to the corresponding field in R_y, and there is at least one field in R_x that is not a subset and not a superset and not equal to the corresponding field in R_y. Formally, $R_x \Re_{PD} R_y$ iff

$$\exists i, j \text{ such that} : R_x[i] \bowtie R_y[i] \text{ and } R_x[j] \not\bowtie R_y[j]$$
$$\text{where } \bowtie \in \{\subset, \supset, =\},$$
$$\text{and } i, j \in \{\text{protocol, s\_ip, s\_port, d\_ip, d\_port}\},$$
$$i \neq sj.$$

For example, Rule 2 and Rule 6 in Fig. 1 are partially disjoint (or partially matching).

Definition 5: Rules R_x and R_y are *correlated* if some fields in R_x are subsets or equal to the corresponding fields in R_y, and the rest of the fields in R_x are supersets of the corresponding fields in R_y. Formally, $R_x \Re_C R_y$ iff

$$\forall i : R_x[i] \bowtie R_y[i]$$
$$\text{and } \exists j, k \text{ such that} : R_x[j] \subset R_y[j] \text{ and } R_x[k] \supset R_y[k]$$
$$\text{where } \bowtie \in \{\subset, \supset, =\},$$
$$\text{and } i, j, k \in \{\text{protocol, s\_ip, s\_port, d\_ip, d\_port}\},$$
$$j \neq k.$$

For example, Rule 1 and Rule 3 in Fig. 1 are correlated.

We define $\Re$ to be the universal set of rule relations as follows:

$$\Re = \{R_{CD}, R_{PD}, R_{EM}, R_{IM}, R_C\}.$$

The following theorems show that these relations are distinct, i.e., only one relation can relate R_x and R_y, and complete, i.e., there is no other relation between R_x and R_y could exist.

Theorem 1: Any two k-tuple filters in a firewall policy are related by one and only one of the defined relations.

Proof: Intuitively, we can show that the intersection between any two relations in $\Re$ is an empty set. In [1], we prove by contradiction that there are no two rules R_x and R_y such that $R_x \Re_1 R_y$ and $R_x \Re_2 R_y$ and $\Re_1 \neq \Re_2$.

Theorem 2: The union of these relations represents the universal set of relations between any two k-tuple filters in a firewall policy.

Proof: In [1], we first prove that the relation between any two two-tuple filters, R_x and R_y, must be in $\Re$. Next, we prove that adding one more field to any two filters related with one of the defined relations will produce two filters, R'_x and R'_y, that are also related by one of these relations. Based on these two results, we use induction to prove that any two rules with k-tuple filters must be related by one of the rule relations defined in this section.

Handling ranges in filtering fields: Network fields in a firewall rule could have a singular value or range of values. The range of values can be specified either using a prefix regular expression $(140.192.37.*)$ or an interval $(0$–$1023)$. Using range intervals might cause partial overlapping between field values in different rules. However, in our analysis, this is not a valid relation between fields because fields are related only by $\{\subset, \supset$, or $=\}$. Therefore, in order to avoid this overlapping, we break down these rules into several equivalent rules such that each rule has a prefix expression [26] or a subrange of values [23] that does not partially overlap with other rules in the policy. Similar rule processing techniques are also described in [12] and [22].

For example, in the following rules the destination port range of rule R_x partially overlaps with the range in R_y:

$$R_x : \texttt{tcp}, 140.192.37.*, \texttt{any}, *.*.*.*, 0\text{–}45, \texttt{deny}$$
$$R_y : \texttt{tcp}, 140.192.37.*, \texttt{any}, *.*.*.*, 23\text{–}80, \texttt{accept}.$$

To resolve this overlap, R_x is expanded into two equivalent rules R_{x_1} and R_{x_2}. As shown below, the expansion guarantees that the destination port ranges of the rules do not partially overlap, yet the policy semantics is preserved

$$R_{x_1} : \texttt{tcp}, 140.192.37.*, \texttt{any}, *.*.*.*, 0\text{–}22, \texttt{deny}$$
$$R_{x_2} : \texttt{tcp}, 140.192.37.*, \texttt{any}, *.*.*.*, 23\text{–}45, \texttt{deny}$$
$$R_y : \texttt{tcp}, 140.192.37.*, \texttt{any}, *.*.*.*, 23\text{–}80, \texttt{accept}.$$

B. Firewall Policy Representation

We represent the firewall policy by a single-rooted tree called the *policy tree* [1], [3]. The tree model provides a simple representation of the filtering rules and at the same time allows for easy discovery of relations and anomalies among these rules. Each node in a policy tree represents a network field, and each branch at this node represents a possible value of the associated field. Every tree path starting at the root and ending at a leaf represents a rule in the policy and *vice versa*. Rules that have the same field value at a specific node will share the same branch representing that value.

Fig. 2 illustrates the policy tree model of the filtering policy given in Fig. 1. Notice that every rule should have an action leaf in the tree. The dotted box below each leaf indicates the rule represented by that branch in addition to other rules that are in anomaly with it as described later in the following section. The tree shows that Rules 1 and 5 each has a separate source address branch as they have different field values, whereas Rules 2, 4, 6, and 7 share the same source address branch as they all have the same field value. Also, notice that Rule 8 has a separate branch and also appears on other rule branches of which it is a superset, while Rule 4 has a separate branch and also appears on other rule branches of which it is a subset. Rule 11 does not appear in the policy tree due to the irrelevance anomaly described in Section IV.
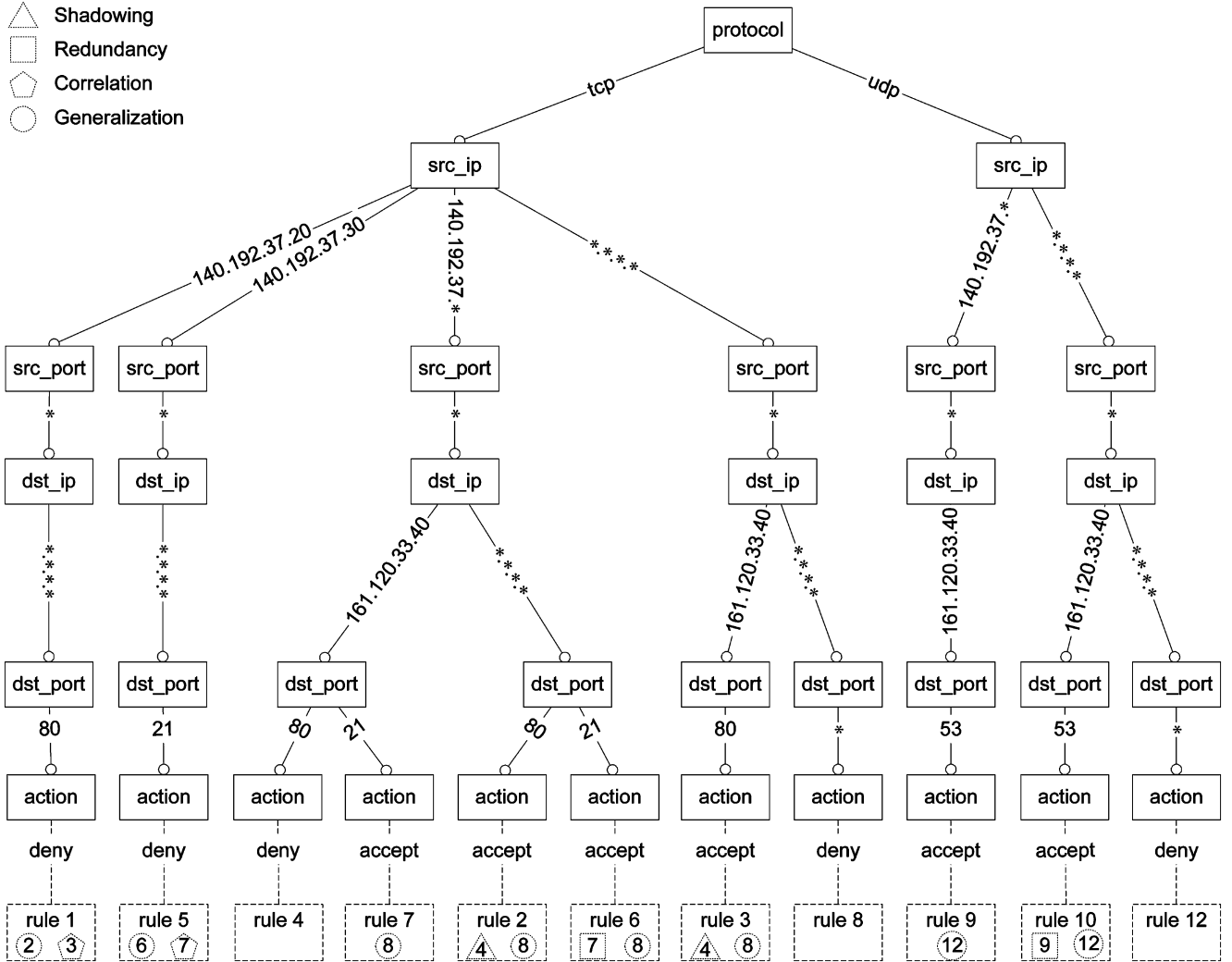
Fig. 2.   Policy tree for the firewall policy in Fig. 1.

IV. ANALYSIS AND DISCOVERY OF INTRAFIREWALL ANOMALIES

The ordering of filtering rules in a centralized firewall policy is very crucial in determining the filtering policy within this firewall. This is because the packet filtering process is performed by sequentially matching the packet against filtering rules until a match is found. If filtering rules are disjoint, the ordering of the rules is insignificant. However, it is very common to have filtering rules that are interrelated. In this case, if the relative rule ordering is not carefully assigned, some rules may be always screened by other rules producing an incorrect policy. Moreover, when the policy contains a large number of filtering rules, the possibility of writing conflicting or redundant rules is relatively high.

An intrafirewall policy anomaly is defined as the existence of two or more filtering rules that may match the same packet or the existence of a rule that can never match any packet on the network paths that cross the firewall [1]. In this section, we classify different anomalies that may exist among filtering rules in one firewall, and then describe a technique for discovering these anomalies.

1) Shadowing Anomaly: A rule is shadowed when a previous rule matches all the packets that match this rule, such that the shadowed rule will never be activated. Formally, rule R_y is shadowed by rule R_x if one of the following conditions holds:

$$R_x[\text{order}] < R_y[\text{order}], \quad R_x\Re_{EM}R_y, R_x[\text{action}] \neq R_y[\text{action}]$$
$$R_x[\text{order}] < R_y[\text{order}], \quad R_y\Re_{IM}R_x, R_x[\text{action}] \neq R_y[\text{action}].$$

For example, Rule 4 in shadowed by Rule 3 in Fig. 1. Shadowing is a critical error in the policy, as the shadowed rule never takes effect, causing an accepted traffic to be blocked or a denied traffic to be permitted.

2) Correlation Anomaly: Two rules are correlated if they have different filtering actions, and the first rule matches some packets that match the second rule and the second rule matches some packets that match the first rule. Formally, rule R_x and rule R_y have a correlation anomaly if the following condition holds:

$$R_x\Re_C R_y, \quad R_x[\text{action}] \neq R_y[\text{action}].$$

Rule 1 is in correlation with Rule 3 in Fig. 1. Correlation is considered an anomaly warning because the correlated rules imply an action that is not explicitly stated by the filtering rules.

3) Generalization Anomaly: A rule is a generalization of a preceding rule if they have different actions, and if the second rule can match all the packets that match the first rule. Formally, rule R_y is a generalization of rule R_x if the following condition holds:

$$R_x[\text{order}] < R_y[\text{order}], \quad R_x \Re_{IM} R_y, R_x[\text{action}] \neq R_y[\text{action}].$$

Rule 2 is a generalization of Rule 1 in Fig. 1. Generalization is often used to exclude a specific part of the traffic from a general filtering action, therefore, it is only considered an anomaly warning.

4) Redundancy Anomaly: A redundant rule performs the same action on the same packets as another rule such that if the redundant rule is removed, the security policy will not be affected. Formally, rule R_y is redundant to rule R_x if one of the following conditions holds:

$$R_x[\text{order}] < R_y[\text{order}], \quad R_x \Re_{EM} R_y, R_x[\text{action}] = R_y[\text{action}]$$
$$R_x[\text{order}] < R_y[\text{order}], \quad R_y \Re_{IM} R_x, R_x[\text{action}] = R_y[\text{action}]$$

whereas rule R_x is redundant to rule R_y if the following condition holds:

$$R_x[\text{order}] < R_y[\text{order}], \quad R_x \Re_{IM} R_y, R_x[\text{action}] = R_y[\text{action}]$$
$$\text{and} \quad \nexists R_z \text{ where } R_x[\text{order}] < R_z[\text{order}] < R_y[\text{order}],$$
$$R_x \Re_{IM,C} R_z, R_x[\text{action}] \neq R_z[\text{action}].$$

Referring to Fig. 1, Rule 7 is redundant to Rule 6, and Rule 9 is redundant to Rule 10. Although redundancy is sometimes preferred, we consider it an error in the firewall policy because a redundant rule adds unnecessary overhead to the filtering process [21].

5) Irrelevance Anomaly: A filtering rule in a firewall is irrelevant if this rule cannot match any traffic that might flow through this firewall. This exists when both the source address and the destination address fields of the rule do not match any domain reachable through this firewall. In other words, the path between the source and destination addresses of this rule does not pass through the firewall. Thus, this rule has no effect on the filtering outcome of this firewall. Formally, rule R_x in firewall F is irrelevant if

$$F \notin \{n : n \text{ is a node on a path from } R_x[\text{src}] \text{ to } R_x[\text{dst}]\}.$$

Referring to Fig. 1, Rule 11 is irrelevant. Irrelevance is considered an anomaly because it adds unnecessary overhead to the filtering process and it does not contribute to the policy semantics.

In our previous work [1], we presented the details of the intrafirewall anomaly discovery algorithm. Before we perform the policy analysis, we preprocess the policy to resolve any partial overlapping among the rules as discussed in Section III-A. Intrafirewall anomaly discovery proceeds by determining if any two rules coincide in their policy tree paths. If the path of a rule coincides with the path of another rule, there is a potential anomaly that can be determined based on the anomaly definitions specified in this section. If rule paths do not coincide, then these rules are disjoint and they have no anomalies. Applying
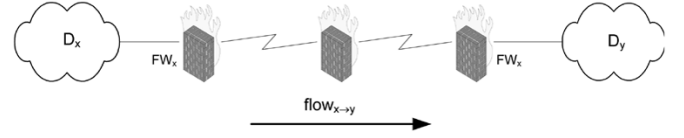

Fig. 3. Cascaded firewalls isolating domains D_x and D_y.

the algorithm on the rules in Fig. 1, the discovered anomalies are marked in the dotted shapes at the bottom of the policy tree in Fig. 2.

V. ANALYSIS AND DISCOVERY OF INTERFIREWALL ANOMALIES

It is very common to have multiple firewalls installed in the same enterprise network. This has many network administration advantages. It gives local control for each domain according to the domain security requirements and applications. For example, some domains might demand to block RTSP traffic or multicast traffic, however, other domains in the same network might request to receive the same traffic. Multifirewall installation also provides interdomain security, and protection from internally generated traffic. Moreover, end-users might use firewalls in their personal workstations for other reasons. However, because of the decentralized nature inherent to the security policy in distributed firewalls, the potential of anomalies between firewalls significantly increases. Even if every firewall policy in the network does not contain rule anomalies described in Section IV, there could be anomalies between policies of different firewalls. For example, an upstream firewall might block a traffic that is permitted by a downstream firewall or *vice versa*. In the first case, this anomaly is called interfirewall "shadowing," which is similar in principle to rule shadowing discussed in the intrafirewall anomaly analysis. In the other case, the resulted anomaly is called "spurious traffic" because it allows unwanted traffic to cross portions of the network and increases the network vulnerability to denial of service attack. In this section, we first define the anomalies that may exist in a distributed firewall environment, and then we identify with examples different types of interfirewall anomalies and we describe a technique to discover these anomalies.

A. Interfirewall Anomaly Definition

In general, an interfirewall anomaly may exist if any two firewalls on a network path take different filtering actions on the same traffic. We first illustrate the simple case of multiple cascaded firewalls isolating two network subdomains, where the firewalls are installed at the routing points in the network [2].

Referring to Fig. 3, we assume a traffic stream flowing from subdomain D_x to subdomain D_y across multiple cascaded firewalls installed on the network path between the two subdomains. At any point on this path in the direction of flow, a preceding firewall is called an *upstream firewall*, whereas a following firewall is called a *downstream firewall*. The closest firewall to the flow source subdomain (FW_x) is called the *most upstream firewall*, while the closest firewall to the flow destination subdomain (FW_y) is called the *most downstream firewall*.

```
1: tcp,  161.120.*.* : any,       *.*.*.* :  80, accept
2: tcp,  140.192.*.* : any,       *.*.*.* :  25, accept
3: tcp,       *.*.*.* : any, 140.192.*.* :  25, accept
4: tcp,  140.192.*.* : any, 161.120.*.* :  80, deny
5: tcp, 161.120.33.* : any, 140.192.37.1 : 23, deny
6: tcp,  161.120.*.* : any, 140.192.*.* :  22, deny
7: tcp,  161.120.*.* : any, 140.192.*.* : any, accept
8: tcp,  140.192.*.* : any, 161.120.*.* : any, accept
9: tcp,       *.*.*.* : any,       *.*.*.* : any, deny
```

```
1: tcp, 161.120.*.* : any  140.192.*.* :  80, accept
2: tcp, 140.192.*.* : any, 161.120.*.* :  80, accept
3: tcp, 161.120.*.* : any, 140.192.22.5 : 21, accept
4: tcp, 161.120.33.* : any 140.192.37.* : 23, deny
5: tcp, 161.120.*.* : any, 140.192.*.* :  23, accept
6: tcp, 161.120.24.* : any, 140.192.37.3 : 25, deny
7: tcp, 161.120.24.* : any, 140.192.*.* :  25, accept
8: tcp,       *.*.*.* : any,       *.*.*.* : any, deny
```

```
1: tcp, 161.120.*.* : any, 140.192.*.* :  80, accept
2: tcp, 161.120.*.* : any, 140.192.22.5 : 21,  deny
3: tcp, 161.120.*.* : any, 140.192.*.* :  21, accept
4: tcp, 140.192.*.* : any, 161.120.33.* : 23, accept
5: tcp, 161.120.33.* : any, 140.192.*.* :  23, accept
6: tcp, 161.120.24.* : any, 140.192.37.3 : 25,  deny
7: tcp, 161.120.24.* : any, 140.192.22.5 : 25,  deny
8: tcp, 161.120.24.* : any, 140.192.37.* : 25, accept
9: tcp,       *.*.*.* : any,       *.*.*.* : any, deny
```
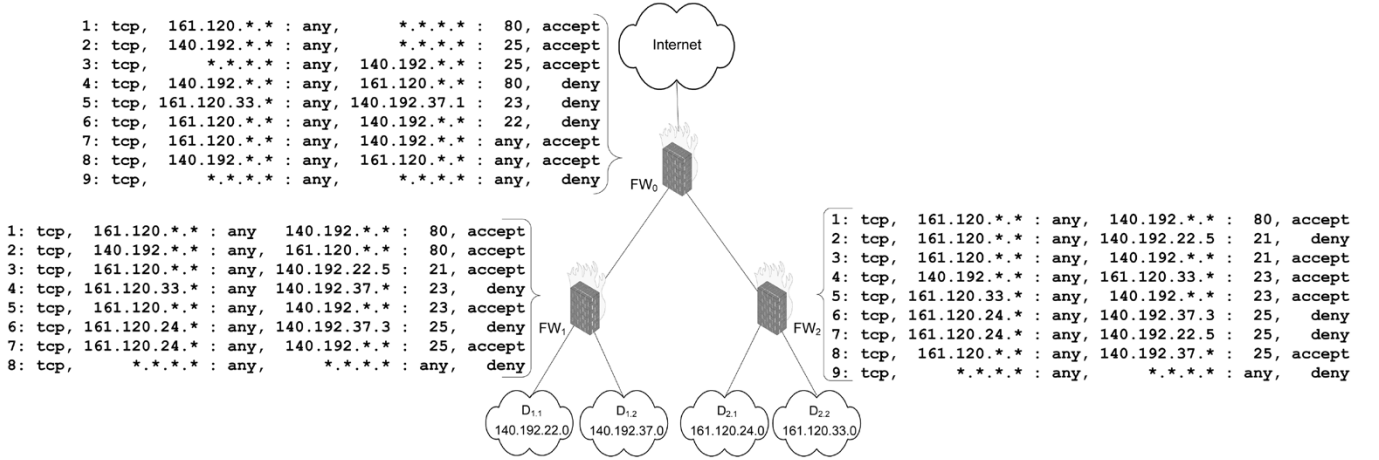
Fig. 4. Example for a hierarchical distributed firewall setup.

Using the above network model, we can say that for any traffic flowing from subdomain D_x to subdomain D_y an anomaly exists if one of the following conditions holds.

1) The most downstream firewall accepts a traffic that is blocked by any of the upstream firewalls.
2) The most upstream firewall permits a traffic that is blocked by any of the downstream firewalls.
3) A downstream firewall denies a traffic that is already blocked by the most upstream firewall.

On the other hand, all upstream firewalls should permit any traffic that is permitted by the most downstream firewall in order that the flow can reach the destination.

B. Interfirewall Anomaly Classification

In this section, we classify anomalies in multifirewall environments. Our classification rules are based on the basic case of cascaded firewalls illustrated in Fig. 3, assuming the network traffic is flowing from domain D_x to domain D_y. Rule R_u belongs to the policy of the most upstream firewall FW_x, while rule R_d belongs to the policy of the most downstream firewall FW_y. We assume that no intrafirewall shadowing or redundancy exists in any individual firewall. As illustrated in Section IV, this implies that every "deny" rule should be followed by a more general "accept" rule, and the default action of unspecified traffic is "deny." The implied rule is resembled by R'_u the in the upstream firewall, and R'_d in the downstream firewall.

1) Shadowing Anomaly: A shadowing anomaly occurs if an upstream firewall blocks the network traffic accepted by a downstream firewall. Formally, rule R_d is shadowed by rule R_u if one of the following conditions holds:

$$R_d \Re_{EM} R_u, \quad R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{accept} \quad (1)$$
$$R_d \Re_{IM} R_u, \quad R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{accept} \quad (2)$$
$$R_u \Re_{IM} R_d, \quad R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{accept} \quad (3)$$
$$R_u \Re_{IM} R_d, \quad R_u[\text{action}] = \text{accept} R_d[\text{action}] = \text{accept}$$
$$\text{and } \exists R'_u, \quad R_u \Re_{IM,C} R'_u, R'_u[\text{action}] = \text{deny}. \quad (4)$$

Intuitively, in cases (1) and (2), the upstream firewall completely blocks the traffic permitted by the downstream firewall.

Rules $(2/FW_2, 3/FW_1)$, and Rules $(8/FW_1, 4/FW_2)$ in Fig. 4 are examples of cases (1) and (2), respectively. In cases (3) and (4), the upstream firewall partially blocks the traffic permitted by the downstream firewall. Rules $(7/FW_2, 7/FW_1)$, and Rules $(5/FW_2, 5/FW_1)$ in Fig. 4 are examples of cases (3) and (4), respectively.

2) Spuriousness Anomaly: A spuriousness anomaly occurs if an upstream firewall permits the network traffic denied by a downstream firewall. Formally, rule R_u allows spurious traffic to rule R_d if one of the following conditions holds:

$$R_u \Re_{EM} R_d, \quad R_u[\text{action}] = \text{accept}, R_d[\text{action}] = \text{deny} \quad (5)$$
$$R_u \Re_{IM} R_d, \quad R_u[\text{action}] = \text{accept}, R_d[\text{action}] = \text{deny} \quad (6)$$
$$R_d \Re_{IM} R_u, \quad R_u[\text{action}] = \text{accept}, R_d[\text{action}] = \text{deny} \quad (7)$$
$$R_d \Re_{IM} R_u, \quad R_u[\text{action}] = \text{accept}, R_d[\text{action}] = \text{accept}$$
$$\text{and } \exists R'_d, R_d \Re_{IM,C} R'_d, R'_d[\text{action}] = \text{deny} \quad (8)$$
$$R_u \Re_{IM} R_d, \quad R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{deny}$$
$$\text{and } \exists R'_u, \Re_{IM,C} R'_u, R'_u[\text{action}] = \text{accept}$$
$$\text{and } R_d \Re_{EM,IM} R'_u, \text{ or } R'_u \Re_{IM} R_d. \quad (9)$$

In cases (5) and (6), the rule R_u in the upstream firewall permits unwanted traffic because it is completely blocked by R_d in the downstream firewall. Examples of these cases are Rules $(2/FW_1, 4/FW_0)$, and Rules $(2/FW_1, 9/FW_2)$ in Fig. 4, respectively. In cases (7) and (8), part of the traffic allowed by rule R_u in upstream firewall is undesired spurious traffic since it is blocked by rule R_d in the downstream firewall. Examples of these cases are also found in Rules $(5/FW_2, 4/FW_1)$, and $(3/FW_2, 3/FW_1)$ in Fig. 4, respectively. Case (9) is not as obvious as the previous cases and it needs further analysis. Since we assume there is no intrafirewall redundancy in the upstream firewall, the fact that R_u has a "deny" action implies that there exists a superset rule in the upstream firewall that follows R_u and accepts some traffic blocked by R_d. This occurs when the implied "accept" rule in the upstream firewall is an exact, superset or subset match (but not correlated) of R_d. Rules $(5/FW_0, 4/FW_1)$ in Fig. 4 are an example of this case.

3) Redundancy Anomaly: A redundancy anomaly occurs if a downstream firewall denies the network traffic already blocked by an upstream firewall. Formally, rule R_d is redundant to rule

R_u if, on every path to which R_u and R_d are relevant, one of the following conditions holds:

$$R_d \Re_{EM} R_u, R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{deny} \quad (10)$$
$$R_d \Re_{IM} R_u, R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{deny}. \quad (11)$$

In both of these cases, the deny action in the downstream firewall is unnecessary because all the traffic denied by R_d is already blocked by R_u in the upstream firewall. In Fig. 4, Rules (6/FW_2, 6/FW_1), and Rules (9/FW_2, 6/FW_0) are examples of cases (10) and (11), respectively.

4) Correlation Anomaly: A correlation anomaly occurs as a result of having two correlated rules in the upstream and downstream firewalls. We defined correlated rules in Section III-A. Intrafirewall correlated rules have an anomaly only if these rules have different filtering actions. However, correlated rules having any action are always a source of anomaly in distributed firewalls because of the implied rule resulting from the conjunction of the correlated rules. This creates not only ambiguity in the interfirewall policy, but also spurious, and shadowing anomalies. Formally, the correlation anomaly for rules R_u and R_d occurs if one of the following conditions holds:

$$R_u \Re_C R_d, R_u[\text{action}] = \text{accept}, R_d[\text{action}] = \text{accept}$$
$$\text{and } \exists R'_u, R_u \Re_{IM,C} R'_u, R'_u[\text{action}] = \text{deny}$$
$$\text{and } \exists R'_d, R_d \Re_{IM,C} R'_d, R'_d[\text{action}] = \text{deny} \quad (12)$$
$$R_u \Re_C R_d, R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{deny}$$
$$\text{and } \exists R'_u, R_u \Re_{IM,C} R'_u, R'_u[\text{action}] = \text{accept}$$
$$\text{and } \exists R'_d, R_d \Re_{IM,C} R'_d, R'_d[\text{action}] = \text{accept} \quad (13)$$
$$R_u \Re_C R_d, R_u[\text{action}] = \text{accept}, R_d[\text{action}] = \text{deny} \quad (14)$$
$$R_u \Re_C R_d, R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{accept}. \quad (15)$$

An example for case (12) is

$$R_u : \texttt{tcp}, 140.192.*.*, \texttt{any}, 161.120.33.*, 80, \texttt{accept}$$
$$R_d : \texttt{tcp}, 140.192.37.*, \texttt{any}, 161.120.*.*, 80, \texttt{accept}.$$

In this example, effectively, the correlative conjunction of these two rules implies that only the traffic coming from 140.192.37.* and destined to 161.120.33.* will be accepted as indicated in the following implied rule:

$$R_i : R_i : \texttt{tcp}, 140.192.37.*, \texttt{any}, 161.120.33.*, 80, \texttt{accept}.$$

This means that other traffic destined to 161.120.*.* will be shadowed at the upstream firewall, while spurious traffic originating from 140.192.*.* will reach the downstream firewall.

For case (13), the example is

$$R_u : \texttt{tcp}, 140.192.*.*, \texttt{any}, 161.120.33.*, 80, \texttt{deny}$$
$$R_d : \texttt{tcp}, 140.192.37.*, \texttt{any}, 161.120.*.*, 80, \texttt{deny}.$$

In this case, the resulting action at the downstream firewall will deny the traffic coming from 140.192.37.* and destined to 161.120.33.*. The implied filtering rule R_i will be

$$R_i : \texttt{tcp}, 140.192.37.*, \texttt{any}, 161.120.33.*, 80, \texttt{deny}.$$

This means that other traffic originating from 140.192.*.* will be shadowed at the upstream firewall, while spurious traffic destined to 161.120.*.* may reach the downstream firewall. A possible resolution for cases (12) and (13) is to replace each of the correlated rules with the implied filtering rule R_i.

The example for case (14) is

$$R_u : \texttt{tcp}, 140.192.*.*, \texttt{any}, 161.120.33.*, 80, \texttt{accept}$$
$$R_d : \texttt{tcp}, 140.192.37.*, \texttt{any}, 161.120.*.*, 80, \texttt{deny}.$$

This example shows that the resulting filtering action at the upstream firewall permits the traffic that is coming from 140.192.37.* and destined to 161.120.33.*. However, the same traffic is blocked at the downstream firewall, resulting in spurious traffic flow. To resolve this anomaly, an extra rule R_i should be added in the upstream firewall prior to R_u such that it blocks the spurious traffic as follows:

$$R_i : \texttt{tcp}, 140.192.37.*, \texttt{any}, 161.120.33.*, 80, \texttt{deny}.$$

As for case (15), the example is

$$R_u : \texttt{tcp}, 140.192.*.*, \texttt{any}, 161.120.33.*, 80, \texttt{deny}$$
$$R_d : \texttt{tcp}, 140.192.37.*, \texttt{any}, 161.120.*.*, 80, \texttt{accept}.$$

This example shows a different situation where the resulting filtering action at the upstream firewall will block the traffic that is coming from 140.192.37.* and destined to 161.120.33.*. However, because this traffic is accepted at the downstream firewall, R_d is shadowed by R_u. To resolve this anomaly, an extra rule R_i should be added in the upstream firewall before R_u to avoid the shadowing anomaly as follows:

$$R_i : \texttt{tcp}, 140.192.37.*, \texttt{any}, 161.120.33.*, 80, \texttt{accept}.$$

In the following theorem, we show that the anomaly cases we presented above are covering all the possible interfirewall anomalies.

Theorem 3: The set of conditions presented above represent all possible rule anomalies that might exist between the policies of any two firewalls.

Proof: Based on the rule relations defined in Section III-A, there are exactly six possible relations between any two rules in different firewalls, namely: completely disjoint, partially disjoint, completely matching, subset matching, superset matching, or correlated. For each relation, there are four different possible combinations considering the action type (i.e., accept and deny) and the firewall location relative to the traffic flow (i.e., upstream and downstream). Therefore, there are 24 possible combinations of rule relations.
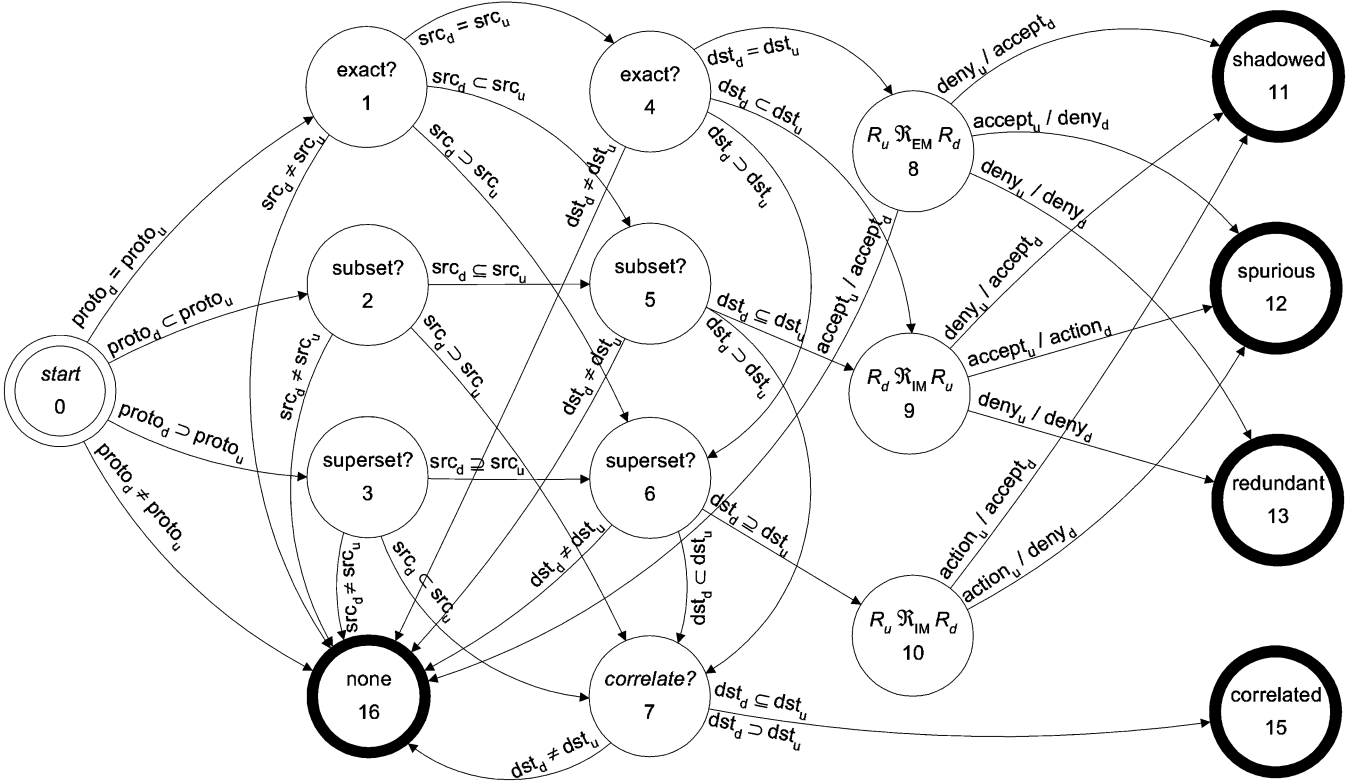
Fig. 5. State diagram for interfirewall anomaly discovery for rules R_u and R_d, where R_u belongs to the upstream firewall and R_d belongs to the downstream firewall.

We can then simply enumerate all possible relations between any two filtering rules in two different firewalls. In this section, we identified 15 of these relations/combinations, and the remaining 9 are listed below

$$R_u\Re_{EM}R_d, \quad R_u[\text{action}] = \text{accept}, R_d[\text{action}] = \text{accept} \quad (16)$$
$$R_u\Re_{CD}R_d, \quad R_u[\text{action}] \in \{\text{accept}, \text{deny}\},$$
$$R_d[\text{action}] \in \{\text{accept}, \text{deny}\} \quad (17\text{--}20)$$
$$R_u\Re_{PD}R_d, \quad R_u[\text{action}] \in \{\text{accept}, \text{deny}\},$$
$$R_d[\text{action}] \in \{\text{accept}, \text{deny}\}. \quad (21\text{--}24)$$

By studying each individual case, we can show that none of the above nine cases causes any anomaly. Case (16) resembles the necessary condition to permit some traffic between the two firewalls and thereby cannot be considered as an anomaly. Cases (17)–(24) represent partially or completely disjoint rules, which in either case do not cause any interfirewall anomaly because the two rules operate on completely different traffic flows. Therefore, none of the cases (16)–(24) could cause any rule anomaly between any two firewalls. Notice that the fact that there is an extra implicit rule R'_u or R'_d as a required condition in cases (4), (8), (9), (12), and (13) does not increase the number of possible relations/combinations. This is because if the extra rule (R'_u or R'_d) does not hold, this implies that rule R_u and rule R_d become redundant for the cases (4), (8), (9), (12) and (13), and the cases (8), (12), and (13), respectively. In this case, R_u and R_d do not contribute to the anomaly analysis and, thus, the dissatisfaction of this implicit condition does not impact this proof.

In conclusion, since cases (1)–(24) identify all possible combinations between any two interfirewall rules, the set of con-

ditions stated in this section represent a complete set of rule anomalies between any two firewalls. Without loss of generality, this result can be applied on any number of rules in distributed firewalls.

C. Interfirewall Anomaly Discovery Algorithm

This algorithm finds the rule relations described in Section V-B and discovers the anomalies between filtering rules in two or more connected firewalls. In Fig. 5, we show the state diagram of the interfirewall anomaly discovery algorithm. The figure shows the anomaly discovery for any two rules, R_u and R_d, where R_u is a rule in the upstream firewall policy, and R_d is a rule in the downstream firewall policy. For simplicity, the address and port fields are integrated in one field for both the source and destination. At the start state, we assume no relationship between the two rules. Each field in R_d is compared with the corresponding field in R_u starting with the protocol, then source and destination addresses and ports. Based on these comparisons, the relation between the two rules is determined, as well as the anomaly if it exists. For example, if R_u is found to inclusively match R_d (State 10), then R_d is partially shadowed if its action is "accept" (State 11), or R_u is spurious if the action of R_d is "deny" (State 12).

Since more than two firewalls may exist between subdomains in an enterprise network, the interfirewall anomaly discovery process should be performed on all firewalls in the path connecting any two subdomains in the network. For example, in Fig. 4, interfirewall anomaly analysis is performed on (FW_1, FW_0) for all traffic that goes between $D_{1.2}$ and the Internet, on (FW_2, FW_0) for all traffic that goes between $D_{2.2}$ and the

Internet, and on (FW_1, FW_0, FW_2) for all traffic that goes between $D_{1.2}$ and $D_{2.2}$. Intuitively, interfirewall anomaly discovery is achieved by building the aggregate policy tree presented in Section III-B for all the firewalls isolating every two subdomains in the network. We start the analysis by determining the list of network paths between every two subdomains in the network. For each path, we determine all the firewalls in the traffic flow. Then, for every firewall in the path, we first run the intrafirewall anomaly discovery algorithm (Section IV) to ensure that every individual firewall is free from intrafirewall anomalies. Next, we build the policy tree of the most upstream firewall, and then add into this tree the rules of all the consecutive firewalls in the path as described later in this section. During this process, only the rules that apply to this path (have relevant source and destination addresses) are selected and marked. At the end of the routine, and as a result of applying the algorithm on all the network paths, the rules that potentially create anomalies are reported. In addition, any rule left unmarked is reported as an irrelevant rule anomaly (Section IV) because it does not apply to any path in the network. Algorithm 1 implements the interfirewall anomaly discovery state diagram shown in Fig. 5. The algorithm can be divided in two phases: the state transition phase (lines 2–24), which represents the transition states in the state diagram, and the state termination phase (lines 25–51), which represents the termination states.

Algorithm 1 Interfirewall anomaly discovery

```
1:  for each field in rule.fields
2:    if field ≠ ACTION then { find
      transition states }
3:     relation ← UNDETERMINED
4:     if branch = field then
5:       if relation = UNDETERMINED then
6:        relation ← EXACT
7:       end if
8:     else if field ⊃ branch then
9:       if relation ∈ { SUBSET, CORRELATED }
         then
10:         relation ← CORRELATED
11:      else if relation ≠ DISJOINT then
12:         relation ← SUPERSET
13:      end if
14:     else if field ⊂ branch then
15:       if relation ∈ { SUPERSET, CORRELATED }
         then
16:         relation ← CORRELATED
17:      else if relation ≠ DISJOINT then
18:         relation ← SUBSET
19:      end if
20:     else
21:      relation ← DISJOINT
22:     end if
23:     field ← field.next
24:     branch ← branch.next
25:    else { find termination state }
26:     if relation = EXACT then
27:       if field = accept and branch = deny
         then
28:        anomaly = SHADOWING
29:       else if field = deny and branch =
         accept then
30:        anomaly = SPURIOUSNESS
31:       else if field = deny and branch =
         deny then
32:        anomaly = REDUNDANCY
33:       end if
34:     else if relation = SUBSET then
35:       if field = accept and branch =
         deny then
36:        anomaly = SHADOWING
37:       else if branch = accept then
38:        anomaly = SPURIOUSNESS
39:       else if field = deny and branch =
         deny then
40:        anomaly = REDUNDANCY
41:       end if
42:     else if relation = SUPERSET then
43:       if field = deny then
44:        anomaly = SPURIOUSNESS
45:       else if field = accept then
46:        anomaly = SHADOWING
47:       end if
48:     else if relation = CORRELATED then
49:      anomaly = CORRELATION
50:     end if
51:    end if
52:  end for
```

After building the policy tree for the most upstream firewall, the transition routine is invoked upon inserting every rule from subsequent downstream firewalls in the aggregate policy tree. Based on the field values, the current rule is inserted in the policy tree by matching the fields of previously inserted rules in the tree, which belong to the preceding upstream firewalls. If the rule is disjoint or correlated, it is inserted into a new branch. If the rule is a superset match, it is inserted into the branches of all the subset rules. Otherwise, the rule is inserted in the first branch of a rule that is an exact or superset match. The termination routine determines the anomaly based on the discovered relation and the actions of the currently inserted rule with the existing rule in the policy tree as described in Fig. 5. If an anomaly is discovered, the involved rules are marked accordingly and the anomaly is reported.

As an example, we apply the interfirewall anomaly discovery algorithm on the example network in Fig. 4. We start by identifying the participating subdomains in the network given the network topology and routing tables. The domains in the figure are $D_{1.1}$, $D_{1.2}$, $D_{2.1}$, and $D_{2.2}$ in addition to the global Internet domain. The Internet domain is basically any address that does not belong to one of the network subdomains. Afterwards, we identify all the possible directed paths between any two subdomains in the network and determine the firewalls that control the traffic on that path, and we run the algorithm on each one of these paths. According to the figure, the algorithm analyzes 20 distinct paths for interfirewall anomalies and produces the anomalies indicated in Section V-B.

Analyzing general network topologies: Although we use a hierarchical network topology example for interfirewall anomaly discovery, this analysis can be performed on any network topology since a static routing topology can be represented as a tree [5]. In case the network topology contains loops, the network topology can be expanded into multiple hierarchical subnetworks (i.e., routing trees) based on the routing configuration of the network nodes. In this case, by combining the anomaly analysis results for the different subnetworks, all possible interfirewall anomalies can be discovered. In case dynamic routing is used, similarly, each routing/delivery tree has to be considered separately for the analysis and the combination of all the resulting reports gives the total anomalies.

Performance optimization of interfirewall anomaly discovery: In complex hierarchical networks, there is a high degree of overlapping between the paths connecting different network subdomains. Since the basic interfirewall analysis technique is performed on every path in the network, the same policy tree might be reconstructed a number of times due to the overlapping of network paths. This occurs when a network segment is used in more than one path in the analysis. For example, in the network in Fig. 4, the segment $R_1 - R_0$ is common to all the paths between the subdomains of D_1 and D_2 and, therefore, the policy tree involving firewalls FW_1 and FW_0 is repeatedly constructed for the analysis of each path between these subdomains. To avoid this computational redundancy and optimize the processing time, we store the policy trees of the paths that are constructed during our analysis so that they can be used in the analysis of other overlapping paths. Unlike the basic technique, this approach requires that every rule in the firewalls on this path is inserted in the constructed policy tree even if this rule does not relate to the currently analyzed path. This is required in order to guarantee that the reused policy tree includes all rules needed to analyze other new paths and subdomains. This optimization produces considerable reduction in the processing time at the expense of requiring more memory space to store intermediate policy trees.

VI. RULE EDITING IN DISTRIBUTED FIREWALL POLICIES

Firewall policies are often written by different network administrators and occasionally updated (by inserting, modifying, or removing rules) to accommodate new security requirements and network topology changes. Editing an enterprise security policy can be far more difficult than creating a new one. A new filtering rule may not apply to every network subdomain, therefore, this rule should be properly located in the correct firewalls to avoid blocking or permitting the wrong traffic. Moreover, as rules in a local firewall policy are ordered, a new rule must be inserted in a particular order to avoid creating intrafirewall anomalies. The same applies if the rule is modified or removed. In this section, we present firewall policy editing techniques that simplify the rule editing task significantly, and avoid introducing anomalies due to policy updates. The policy editor helps the user to determine the correct firewalls at which a new rule should be located avoiding interfirewall anomalies, and helps to determine the proper order for the rule within these firewalls avoiding intrafirewall anomalies. Using the policy editor, administrators

need no prior analysis of the firewall rules in order to insert, modify, or remove a rule.

A. Rule Placement and Insertion Algorithm

The process of inserting a new rule in the global security policy is performed in two steps. The first step is to identify the firewalls in which this rule should be placed. This is needed in order to apply the filtering rule only on the relevant subdomains without creating any interfirewall anomalies. The second step is to determine the proper order of the rule in each firewall such that no intrafirewall anomaly is created.

Algorithm 2 is used to locate the firewalls where a rule should be inserted. As a first step to insert a rule, we identify all the possible paths that go from the source address to the destination address of the rule (lines 1–13). If any of the source or destination addresses is an external address (Internet address), then we find the path to/from the closest firewall to the Internet. Second, the rule is inserted in all firewalls in the identified paths if the rule action is "accept." Otherwise, if the rule action is "deny," the rule is inserted only in the most upstream firewalls(s) relative to the source(s) (lines 14–24). As an example, the following rules are inserted in the policy shown in Fig. 4:

$R_1 : \mathtt{icmp}, *.*.*.*, \mathtt{any}, 140.192.*.*, \mathtt{any}, \ \mathtt{deny}$

$R_2 : \mathtt{icmp}, 140.192.*.*, \mathtt{any}, 161.120.*.*, \mathtt{any}, \mathtt{accept}.$

Algorithm 2 Rule placement and removal in distributed firewalls
Input: *rule*, *topology*, *edit_action*
1: **for each** *domain* ∈ *topology* **do**
2:   **if** *domain* ⊂ *rule.src_ip* or *domain* ⊃ *rule.src_ip* **then**
3:     Append(*src_domains*, *domain*)
4:   **else if** *domain* ⊂ *rule.dst_ip* or *domain* ⊃ *rule.dst_ip* **then**
5:     Append(*dst_domains*, *domain*)
6:   **end if**
7: **end for**
8: **for each** *src_domain* ∈ *src_domains* **do**
9:   **for each** *dst_domain* ∈ *dst_domains* **do**
10:     *path* ← {$f_1, f_2, \ldots, f_i$: f_i is a firewall on the path from *src_domain* to *dst_domain*}
11:     Append(*src-dst_paths*, *path*)
12:   **end for**
13: **end for**
14: **for each** *path* ∈ *src-dst_paths* **do**
15:   **for each** *firewall* ∈ *path* **do**
16:     **if** *edit_action* = insert **then**
17:       Invoke Algorithm 3 to insert *rule* in *firewall*
18:     **else if** *edit_action* = remove **then**
19:       Remove *rule* from *firewall*
20:     **end if**
21:     **if** *rule.action* = deny **then**
22:       **break**
23:     **end if**
24:   **end for**
25: **end for**

R_1 is installed in firewalls FW_0 and FW_2 because they are the most upstream firewalls on the paths from the Internet and domain D_2 (161.120. $*$.$*$) to domain D_1 (140.192. $*$.$*$), respectively. R_2 is installed in firewalls FW_0, FW_1, and FW_2 as they all exist on the path from the domain D_1 (140.192. $*$.$*$) to domain D_2 (161.120. $*$.$*$).

In the second step, the order of the new rule in the local firewall policy is determined based on its relation with other existing rules. In general, a new rule should be inserted before any rule that is a superset match, and after any rule that is a subset match of this rule. Algorithm 3 uses the local policy tree to keep track of the correct ordering of the new rule, and discover any potential anomalies. We start by searching for the correct rule position in the policy tree by comparing the fields of the new rule with the corresponding tree branch values. If the field value is a subset of the branch, then the order of the new rule so far is smaller than the minimum order of all the rules in this branch (line 11). If the field value is a superset of the branch, the order of the new rule so far is greater than the maximum order of all the rules in this branch (line 16). On the other hand, if the rule is disjoint, then it can be given any order in the policy. Similarly, the tree browsing continues, matching the next fields in the rule as long as a field value is an exact match or a subset match of a branch (lines 8, 12, 21). A new branch is created for the new rule any time a disjoint or superset match is found (line 23). When the action field is reached, the rule is inserted and assigned an order within the maximum and minimum range determined in the browsing phase. If the new rule is redundant because it is an exact match or a subset match and it has the same action of an existing rule, the policy editor rejects it and prompts the user with an appropriate message (lines 27–35).

Algorithm 3 Rule insertion in a single firewall policy
Input: $rule$, $tree$
Output: min_order, max_order, $anomaly$
1: $min\_order, max\_order \leftarrow$ UNDERTERMINED
2: $node \leftarrow tree.root$
3: **for each** $field \in rule.fields$ **do**
4: **if** $field \neq$ ACTION **then**
5:   $target \leftarrow$ nil
6:   **for each** $branch \in node.branches$ **do**
7:    **if** $branch = field$ **then**
8:     $target \leftarrow branch$
9:    **else if** $field \subset branch$ **then**
10:     **if** $max\_order >$ MinOrder($branch$) **then**
11:      $max\_order \leftarrow$ MinOrder($branch$)-1
12:      $target \leftarrow branch$
13:     **end if**
14:    **else if** $field \supset branch$ **then**
15:     **if** $min\_order <$ MaxOrder($branch$) **then**
16:      $min\_order \leftarrow$ MaxOrder($branch$)+1
17:     **end if**
18:    **end if**
19:   **end for**
20:   **if** $target \neq$ nil **then** { browse target branch }
21:    $node \leftarrow target.next$
22:   **else** { create new branch }
23:    $node \leftarrow$ NewBranch($node$, $field$)
24:   **end if**
25: **else if** $target \neq$ nil **then** { and action field reached }
26:   $anomaly \leftarrow$ NOANOMALY
27:    **if** $min\_order =$ UNDERTERMINED and $max\_order =$ UNDERTERMINED **then**
28:     **if** $field = branch$ **then** { similar actions }
29:      $anomaly \leftarrow$ REDUNDANCY
30:     **else** { different actions }
31:      $anomaly \leftarrow$ SHADOWING
32:     **end if**
33:    **else if** $max\_order \neq$ UNDERTERMINED and $field = branch$ **then** { similar actions }
34:     $anomaly \leftarrow$ REDUNDANCY
35:    **end if**
36: **end if**
37: **end for**

After inserting the rule in the appropriate firewalls, the interfirewall anomaly discovery algorithm in Section V-C is activated to verify that no intrafirewall or interfirewall anomalies are introduced in the distributed security policy, and to identify any correlation or generalization anomalies the new rule might have created.

B. Rule Removal Algorithm

In distributed firewall environments, removing a rule from a specific firewall may result in creating an interfirewall anomaly. For example, if a "deny" rule is removed from the upstream firewall, this may result in spurious traffic flowing downstream, but if an "accept" rule is removed from the upstream firewall, the relevant traffic may be blocked and all the related (exact, subset, or superset) downstream rules will be shadowed.

When the user decides to remove a rule from a certain firewall, the first step is to identify all the source and destination subdomains that will be impacted by removing this rule. Again, we use Algorithm 2 to locate the firewalls where a rule should be removed. In the second step, we remove the rule from the firewall policy as follows. If the rule is an "accept" rule, then we remove it from all the firewalls in all paths from source to destination subdomains. Otherwise, shadowing or spuriousness anomaly is created if the rule is removed only from the upstream or the downstream firewalls, respectively. If the rule is a "deny" rule, then we just remove it from the local firewall. In this case, we alert the administrator that removing the rule will result in allowing some extra traffic to flow through the firewall to/from other domains.

Modifying a rule in a firewall policy is also a critical operation. However, a modified rule can be easily verified and inserted based on the rule removal and insertion techniques described above. It is important to note that the policy editor only alerts the administrator in case rule modification results in an interfirewall anomaly. If confirmed, the update is applied in spite of the anomaly existence. This approach is practically appropriate since the administrator might have legitimate reasons for

Experience	Shadowing	Redundancy	Correlation	Irrelevance
Expert	0%	5%	3%	0%
Intermediate	1%	9%	3%	0%
Beginner	4%	12%	9%	2%

Fig. 6. Average percentage of discovered anomalies in a man-written centralized firewall policy.

Experience	Shadowing	Spuriousness	Redundancy	Correlation
Expert	1%	7%	9%	1%
Intermediate	3%	10%	11%	2%
Beginner	6%	14%	17%	2%

Fig. 7. Average percentage of discovered anomalies in a man-written distributed firewall policy.

this update, like temporarily blocking undesired traffic (i.e., creating shadowing anomaly) or performing live testing on domain firewalls (i.e., creating spurious anomaly).

VII. FIREWALL POLICY ADVISOR: IMPLEMENTATION AND EVALUATION

We implemented the techniques and algorithms described in Sections V and IV in a software tool called the "Firewall Policy Advisor" or FPA.[1] The tool implements the interfirewall and intrafirewall anomaly discovery algorithms, as well as the distributed firewall policy editor. The FPA was developed using Java programming language and it includes a graphical user interface. In this section, we present our evaluation study of the usability and the performance of the anomaly discovery techniques described in this paper.

To assess the practical value of our techniques, we first used the FPA tool to analyze real firewall rules in our university network, as well as in some local industrial networks in the area. In many cases, the FPA has shown to be effective by discovering many firewall anomalies that were not discovered by human visual inspection. We then attempted to quantitatively evaluate the practical usability of the FPA by conducting a set of experiments that consider the level of network administrator expertise and the frequency of occurrence of each anomaly type. In this experiment, we created two firewall policy exercises and asked 12 network administrators with varying level of expertise in the field to complete each exercise. The exercises include writing filtering rules in centralized and distributed firewalls based on a given security policy requirements. We then used the FPA tool to analyze the rules in the answer of each one and calculated the ratio of each anomaly relative to total number of rules. The average total number of rules was 40 in the centralized firewall, and 90 in the distributed firewall for a network having only three firewalls. The results of this experiment are shown in Figs. 6 and 7 for the centralized and distributed firewall exercises, respectively.

These results show clearly that the margin of error that can be done even by an expert administrator is quite significant (about 8% for centralized one and 18% for the distributed one). This figure is even much higher for an intermediate and beginner administrators (about 13% and 27% for centralized firewall and 26% and 39% for the distributed firewalls, respectively). Another interesting observation is the high percentage of redundant as well as spurious rules in all experience levels.

In the last phase of our evaluation study, we conducted a number of experiments to measure the performance and

the scalability of firewall anomaly discovery under different filtering policies and network topologies. Our experiments were performed on a Pentium PIII 400 MHz processor with 128 MByte of RAM.

To study the performance of the intrafirewall anomaly discovery algorithm, we produced four sets of firewall rules. The first set includes rules that are different in the destination address only, and the second set includes rules that have distinct source addresses. These two sets resemble a realistic combination of practical firewall rules, and represent the best case scenario because they require the minimum policy-tree navigation for analyzing each rule. In the third set, each rule is a superset match of the preceding rule. This set represents the worst case scenario because each rule requires complete policy-tree navigation in order to analyze the entire rule set. The fourth set includes rules that are randomly selected from the three previous sets in order to represent the average case scenario. We used the FPA tool to run the intrafirewall policy analysis algorithm on each set using various sizes of rule sets (10–90 rules). In each case, we measured the processing time needed to produce the policy analysis report. The results we obtained are shown in Fig. 8(a). Set 1 shows the least processing time because all the rules are aggregated in one branch in the policy tree, which makes the number of field matches and node creations minimal. Set 2 has a slightly higher processing time, since each rule in the set has a distinct branch at a higher level in the policy tree. This requires more time to create new tree nodes for inserting each rule in the tree. Set 3 is expected to have the highest processing time since every rule in the set must be matched with all rules in the policy tree. Set 4 shows a moderate (average) processing time and represents the most practical scenario as it combines many different cases. Even in the worst case scenario (Set 3), the processing time looks very reasonable; approximately 20–240 ms for 10–90 rules. In addition, the processing time increases about 2.1–2.8 ms per rule, which is considered insignificant overhead even if hundreds of rules exist in a firewall.

For evaluating the performance of the interfirewall anomaly discovery algorithm, we conducted two different experiments. In the first experiment, we applied the discovery algorithm on a set of firewalls that exist on one network path. The rules used in each firewall are similar to Set 2 rules in the previous experiment. We varied the number of rules in each firewall, and the number of firewalls on the path. The results in Fig. 8(b) show that the processing time of the algorithm applied on multiple firewalls is very close to the performance of intrafirewall analysis applied on a single firewall containing an equivalent number of rules. For example, it takes 100 ms to analyze four firewalls each containing 20 rules. This is almost equal to the
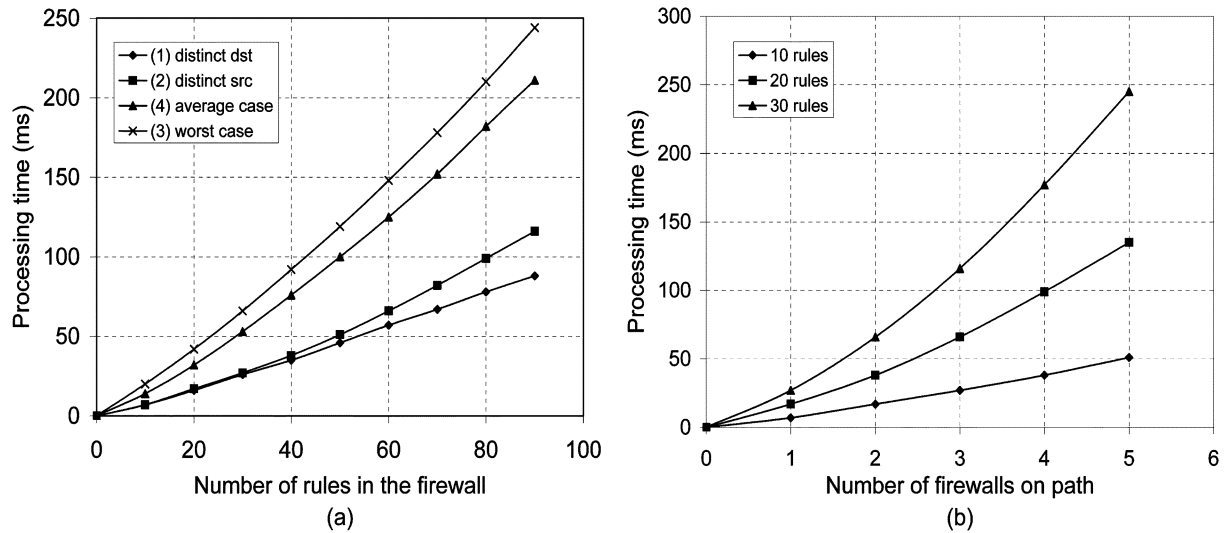
Fig. 8.   Processing time for (a) intrafirewall anomaly discovery, (b) interfirewall anomaly discovery of one path.
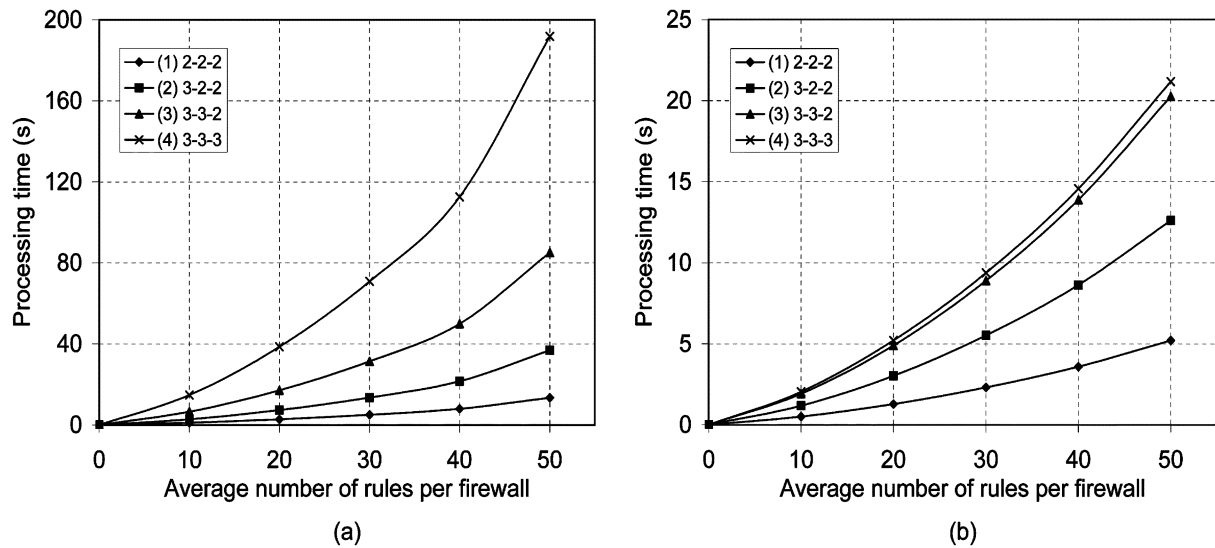

Fig. 9.   Interfirewall processing time for (a) basic technique and (b) optimized technique.

time required to perform intrafirewall analysis on a single firewall having 80 rules. These results indicate that the complexity of the interfirewall analysis algorithm is dependent on the total number of rules in all firewalls on the analyzed path rather than the number of firewalls being analyzed.

In the second experiment, we applied the discovery algorithm on a complex network of distributed firewalls. We used a balanced three-level hierarchical network topology connected to the Internet via the root node. Each nonleaf node in the network has filtering capability. We created four networks with different branching degrees at each level in the hierarchy starting at the root node: (1) 2-2-2, (2) 3-2-2, (3) 3-3-2, and (4) 3-3-3. For example, the root node in Network 2 has 3 branches, whereas every node on levels two and three has two branches. For each network, we installed a random set of filtering rules in each firewall. The generated topology information and the firewall setup of each network are used as inputs for our experiment. We then used the FPA to run the interfirewall policy analysis algorithm

on each network with a different number of rules (10–50 rules) for each firewall. We measured, in each case, the processing time required to produce the final policy analysis report. The results are shown in Fig. 9(a). We noticed that for small and midsize networks (such as Network 1 that has 8 subdomains and Network 2 that has 12 subdomains), the processing time ranges from 3 to 40 s. However, in case of large networks (such as Networks 3 and 4 that have 18 and 27 subdomains, respectively), the firewall anomaly discovery requires much higher processing time ranging from 11 to 180 s depending on the rule complexity. The increase in the processing time as the network size increases is due to the fact that the complexity of our algorithm is dependant on the total number of paths between subdomains in the network. Fig. 9(b) shows the results when running the same experiment using the tree-construction optimization discussed in Section V-C. By applying this optimization, the processing time for the different network topologies is reduced by 60%–90% when each firewall has 50 rules. The results reflect significant

improvement over the basic technique especially for large network topologies (such as Networks 3 and 4). The high degree of path overlapping in these networks enables the optimization technique to significantly reduce the number of policy trees constructed during the analysis.

VIII. RELATED WORK

A significant amount of work has been reported in the area of firewall and policy-based security management. In this section, we focus our study on the related work that intersects with our work in three areas: packet filter modeling, conflict discovery and rule analysis, and distributed firewall policy management.

Several models have been proposed for filtering rules. Ordered binary decision diagram is used as a model for optimizing packet classification in [15]. Another model using tuple space is developed in [27], which combines a set of filters in one tuple stored in a hash table. The model in [29] uses bucket filters indexed by search trees. Multidimensional binary tries are also used to model filters [22]. In [9], a geometric model is used to represent two-tuple filtering rules. Because these models were designed particularly to optimize packet classification in high-speed networks, we found them too complex to use for firewall policy analysis. Interval diagrams are used in [12] to compact firewall rules. However, it requires preprocessing of firewall rules to resolve any rule overlap and, therefore, it cannot be used for our anomaly analysis. We can confirm from experience that the tree-based model we use is simple and powerful enough for this purpose.

Research in policy conflict analysis has been actively growing for many years. However, most of the work in this area addresses general management policies rather than firewall-specific policies. For example, authors in [18] classify possible policy conflicts in role-based management frameworks, and develop techniques to discover them. A policy conflict scheme for IP-Security (IPSec) is presented in [11]. Although this work is very useful as a general background, it is not directly applicable in firewall anomaly discovery. On the other hand, few research projects address the conflict problem in filtering rules. Both [9] and [14] provide algorithms for detecting and resolving conflicts among general packet filters. However, they only detect what we defined as correlation anomaly because it causes ambiguity in packet classifiers. Other research work goes one step forward by offering query-based tools for firewall policy analysis. In [20] and [30], the authors developed a firewall analysis tool to perform customized queries on a set of filtering rules and extract the related rules in the policy. A similar approach using expert systems is presented in [10]. All these tools can help in manual verification of the correctness of firewall policies. However, they require high user expertise in order to write the proper queries to identify different types of firewall policy problems.

In the field of distributed firewalls, current research mainly focuses on the management of distributed firewall policies. The first generation of global policy management technology is presented in [13], which proposes a global policy definition language along with algorithms for verifying the policy and generating filtering rules. In [4], the authors adopted a better approach by using a modular architecture that separates the security policy and the underlying network topology to allow for flexible modification of the network topology without the need to update the security policy. Similar work has been done in [16] with a procedural policy definition language, and in [19] with an object-oriented policy definition language. In terms of distributed firewall policy enforcement, a novel architecture is proposed in [17], where the authors suggest using a trust management system to enforce a centralized security policy at individual network endpoints based on access rights granted to users or hosts. We found that none of the published work in this area addressed the problem of discovering anomalies in distributed firewall environments.

In conclusion, we could not find any published research work that uses low-level filtering rules to perform a complete anomaly analysis and guided editing of centralized and distributed firewall policies.

IX. CONCLUSION AND FUTURE WORK

Firewall security, like any other technology, requires proper management in order to provide proper security services. Thus, just having firewalls on the network boundaries or between subdomains may not necessarily make the network any secure. One reason of this is the complexity of managing firewall rules and the resulting network vulnerability due to rule anomalies. The Firewall Policy Advisor presented in this paper provides a number of techniques for purifying and protecting the firewall policy from rule anomalies. The administrator may use the firewall policy advisor to manage firewall policies without prior analysis of filtering rules. In this paper, we formally defined a number of firewall policy anomalies in both centralized and distributed firewalls, and we proved that these are the only conflicts that could exist in firewall policies. We then presented a set of algorithms to detect rule anomalies within a single firewall (intrafirewall anomalies), and between interconnected firewalls (interfirewall anomalies) in the network. When an anomaly is detected, users are prompted with proper corrective actions. We intentionally made the tool not to automatically correct the discovered anomaly but rather alarm the user because we believe that the administrator should have the final call on policy changes. Finally, we presented a user-friendly Java-based implementation of Firewall Policy Advisor.

Using Firewall Policy Advisor was shown to be very effective for firewalls in real-life networks. In regards to usability, the tool was able to discover filtering anomalies in rules written by expert network administrators. In regards to performance, although the policy analysis algorithms are parabolically dependant on the number of rules in the firewall policy, our experiments show that the average processing time in intrafirewall and interfirewall anomaly discovery is very reasonable for practical applications. Using our Java implementation of the anomaly discovery algorithms, our results indicate that it, in the worst case, it takes 10–240 ms of processing time to analyze a security policy of 10–90 rules in a single firewall. However, in a considerably large network (27 subdomains with 13 firewalls), it

takes 20–180 s to analyze the filtering rules of all firewalls in the network.

We believe that there is much more to do in firewall policy management area. Our future research plan includes online automatic discovery and recovery of anomalies created as a result of the rule editing, generation of test traffic for policy verification, dynamic rule placement based on firewall performance, and translation of low-level filtering rules into high-level textual description and *vice versa*.

REFERENCES

[1] E. Al-Shaer and H. Hamed, "Modeling and management of firewall policies," *IEEE Trans. Netw. Serv. Manage.*, vol. 1-1, Apr. 2004.

[2] ——, "Discovery of policy anomalies in distributed firewalls," in *Proc. IEEE INFOCOM*, Mar. 2004, pp. 2605–2616.

[3] ——, "Firewall policy advisor for anomaly detection and rule editing," in *Proc. IEEE/IFIP Integr. Manage. Conf. (IM'2003)*, Mar. 2003, pp. 17–30.

[4] Y. Bartal, A. Mayer, K. Nissim, and A. Wool, "Firmato: A novel firewall management toolkit," in *Proc. IEEE Symp. Security Privacy*, May 1999, pp. 17–31.

[5] T. Bu, N. Duffield, F. L. Presti, and D. Towsley, "Network tomography on general topologies," in *Proc. ACM SIGMETRICS'02 Conf.*, Jun. 2002, pp. 21–30.

[6] D. Chapman and E. Zwicky, *Building Internet Firewalls*, 2nd ed. : Orielly & Associates, 2000.

[7] W. Cheswick and S. Belovin, *Firewalls and Internet Security*. Reading, MA: Addison-Wesley, 1995.

[8] S. Cobb, *ICSA Firewall Policy Guide V2.0*, ser. NCSA Security White Paper Series, 1997.

[9] D. Eppstein and S. Muthukrishnan, "Internet packet filter management and rectangle geometry," in *Proc. 12th Annu. ACM-SIAM Symp. Discrete Algorithms*, Jan. 2001, pp. 827–835.

[10] P. Eronen and J. Zitting, "An expert system for analyzing firewall rules," in *Proc. 6th Nordic Workshop Secure IT-Syst.*, Nov. 2001, pp. 100–107.

[11] Z. Fu, F. Wu, H. Huang, K. Loh, F. Gong, I. Baldine, and C. Xu, "IPsec/VPN security policy: Correctness, conflict detection and resolution," in *Proc. Policy Workshop*, Jan. 2001, pp. 39–56.

[12] M. Gouda and X. Liu, "Firewall design: Consistency, completeness, and compactness," in *Proc. 24th IEEE Int. Conf. Distrib. Comput. Syst.*, Mar. 2004, pp. 320–327.

[13] J. Guttman, "Filtering posture: Local enforcement for global policies," in *Proc. IEEE Symp. Security Privacy*, May 1997.

[14] B. Hari, S. Suri, and G. Parulkar, "Detecting and resolving packet filter conflicts," in *Proc. IEEE INFOCOM*, Mar. 2000, pp. 1203–1212.

[15] S. Hazelhusrt, "Algorithms for analyzing firewall and router access lists," Dept. Comput. Scie., Univ. Witwatersrand, South Africa, Tech. Rep. TR-WitsCS-1999, 1999.

[16] S. Hinrichs, "Policy-based management: Bridging the gap," in *Proc. 15th Annu. Comput. Security Applicat. Conf.*, Dec. 1999, pp. 209–218.

[17] S. Ioannidis, A. Keromytis, S. Bellovin, and J. Smith, "Implementing a distributed firewall," in *Proc. 7th ACM Conf. Comput. Commun. Security*, Nov. 2000, pp. 190–199.

[18] E. Lupu and M. Sloman, "Conflict analysis for management policies," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage.*, May 1997, pp. 430–443.

[19] I. Luck, C. Schafer, and H. Krumm, "Model-based tool assistance for packet-filter design," in *Proc. Workshop Policies Distrib. Syst. Netw.*, Jan. 2001, pp. 120–136.

[20] A. Mayer, A. Wool, and E. Ziskind, "Fang: A firewall analysis engine," in *Proc. 2000 IEEE Symp. Security Privacy*, May 2000, pp. 177–187.

[21] R. Panko, *Corporate Computer and Network Security*. Englewood Cliffs, NJ: Prentice-Hall, 2003.

[22] L. Qiu, G. Varghese, and S. Suri, "Fast firewall implementations for software and hardware-based routers," in *Proc. 9th Int. Conf. Netw. Protocols*, Nov. 2001, pp. 344–345.

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

[24] R. Smith and S. Bhattachayra, "Firewall placement in a large network topology," in *Proc. 6th Workshop Future Trends Distrib. Comput.*, Oct. 1997, pp. 40–45.

[25] ——, "A protocol and simulation for distributed communicating firewalls," in *Proc. 23rd IEEE Int. Comput. Softw. Applicat. Conf.*, Oct. 1999, pp. 74–79.

[26] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvoge, "Fast and scalable layer four switching," in *Proc. ACM SIGCOMM Conf.*, Sep. 1998, pp. 191–202.

[27] V. Srinivasan, S. Suri, and G. Varghese, "Packet classification using tuple space search," *Comput. ACM SIGCOMM Commun. Rev.*, pp. 135–146, Oct. 1999.

[28] J. Wack, K. Cutler, and J. Pole, "Guidelines on firewalls and firewall policy," NIST Recommendations, SP 800-41, 2002.

[29] T. Woo, "A modular approach to packet classification: Algorithms and results," in *Proc. IEEE INFOCOM*, Mar. 2000, pp. 1213–1222.

[30] A. Wool, "Architecting the lumeta firewall analyzer," in *Proc. 10th USENIX Security Symp.*, Aug. 2001, pp. 381–420.

Ehab Al-Shaer is an Associate Professor and the Director of the Multimedia Networking Research Laboratory (MNLAB) in the School of Computer Science, Telecommunications and Information System, DePaul University, Chicago, IL. He also was a Co-Editor of number of books including *Management of Multimedia on the Internet* published by Lecture Notes in Computer Science and *Monitoring Internet Traffic and Services* (Piscataway, NJ: IEEE). His primary research areas are network security, Internet monitoring, fault management in overlay networks, and multimedia transport protocols. He has more than 50 refereed publications in premier journals and conferences in his area.

Prof. Al-Shaer was awarded the Best Paper Award at the Integrated Management Conference (IM03) and NASA fellowship in 1997. He was the Conference Program Co-Chair for the Fourth IEEE/IFIP International Conference on Management of Multimedia Networks and Services (MMNS) in October 2001 and the TPC Co-Chair for IEEE End-to-End Monitoring Techniques and Services (E2EMON) in 2003, 2004, and 2005, respectively. He was Guest Editor for number of journals including the *Journal of High Speed Networking* and the *Journal of Computer Communications*. He also served as conference invited speaker, panelist, tutorial presenter, poster chair, workshops chair, and program committee member for many major IEEE and ACM conferences including INFOCOM, ICNP, IM, NOMS, ICDCS, CCNC, GLOBECOM, ICC, MMNS DSOM, and E2EMON.

Hazem Hamed received the B.Sc. degree in computer and systems engineering and the M.Sc. degree in computer engineering from Ain-Shams University, Cairo, Egypt, in 1994 and 1999, respectively. He is currently working towards the Ph.D. degree in the School of Computer Science, Telecommunications and Information Systems, DePaul University, Chicago, IL.

He is also working as a Teaching and Research Assistant in the Computer Science Department, DePaul University. His main research is directed to the management of network security policies including firewall and IPSec policies. His other research interests include network security, differentiated services, and reliable multicasting.

Mr. Hamed was awarded the Best Paper Award at the Integrated Management Conference (IM 2003).

Raouf Boutaba is an Associate Professor in the School of Computer Science, University of Waterloo, Waterloo, ON, Canada. Before that he was with the Department of Electrical and Computer Engineering of the University of Toronto. Before joining academia, he founded and was the Director of the Telecommunications and Distributed Systems Division, Computer Science Research Institute of Montreal (CRIM). He has published more than 150 papers in refereed journals and conference proceedings. He conducts research in the areas of network and distributed systems management and resource management in multimedia wired and wireless networks.

Dr. Boutaba is the recipient of the Premier's Research Excellence Award, the NORTEL Networks Research Excellence Award, and several Best Paper Awards. He is a Fellow of the Faculty of Mathematics, University of Waterloo, and a Distinguished Lecturer of the IEEE Computer Society. He is the Chairman of the Working Group on Networks and Distributed Systems of the International Federation for Information Processing (IFIP), the Vice Chair of the IEEE Communications Society Technical Committee on Information Infrastructure, and the Director of the Standards Board of the IEEE Communications Society. He is the Founder and acting Editor-in-Chief of the *IEEE eTransactions on Network and Service Management*, on the Advisory Editorial Board of the *Journal of Network and Systems Management*, on the Editorial Board of the *KIKS/IEEE Journal of Communications and Networks*, on the Editorial Board of the *Journal of Computer Networks*, and the *Journal of Computer Communications*. He has also served as a Guest Editor of several special issues of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (JSAC), the *Journal of Computer Networks*, the *Journal of Computer Communications*, and the *Journal of Network and System Management*. He acted as the Program Chair for the IFIP Networking Conference and the IEEE Consumer Communications and Networking Conference (CCNC), and a Program Co-Chair for the IEEE/IFIP Network Operation and Management Symposium (NOMS), the IFIP/IEEE Conference on Management of Multimedia Networks and Services (MMNS), the IEEE Feature Interaction Workshop, the IEEE Autonomic Computing and Communications (ACC), and two IEEE International Conferences on Communications (ICC) Symposia.

Masum Hasan is a Senior Technical Leader at Cisco Systems, San Jose, CA. Prior to joining Cisco Systems, he was a Principal Investigator at Bell Laboratories Research, and a Research Associate at the University of Toronto, Toronto, ON, Canada. He has worked in industry and academia in Bangladesh and Canada. He has presented tutorials and participated as invited speaker and on invited panels on network management and control plane issues of QoS, IP VPN, traffic engineering, GMPLS, optical/ASON networks, and grid networking. He has published extensively in a number of areas of computer science discipline, including network management, active, temporal, and text database systems, computer languages and environments, distributed and parallel programming systems, grid networking, and structured data visualization. His current R&D focus is on control and management issues of autonomic grid networking, [G]MPLS and optical networks, network QoS, VPN, traffic engineering, and configuration.

Mr. Hassan is on the committees of a number of IFIP and IEEE international conferences, including IM, NOMS, IPOM, MMNS, and AGNM. He is on the Editorial Board of the *Journal of Network and Systems Management*. He also participates and contributes to international standard bodies, such as Global Grid Forum (GGF), ITU-T, Optical Internetworking Forum (OIF), and IETF.