

Discovery of Policy Anomalies in Distributed Firewalls

Ehab S. Al-Shaer and Hazem H. Hamed

Multimedia Networking Research Laboratory
School of Computer Science, Telecommunications and Information Systems
DePaul University, Chicago, USA
Email: {ehab, hhamed}@cs.depaul.edu

Abstract—Firewalls are core elements in network security. However, managing firewall rules, particularly in multi-firewall 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 intra- and inter-firewall 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 multi-firewall environment. We also present a set of techniques and algorithms to automatically discover policy anomalies in centralized and distributed legacy 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.¹

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 defined according to 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 *intra-firewall anomalies*, where the same packet may match more than one filtering rule. Moreover, in distributed firewall environments, firewalls might also have *inter-firewall 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 intra- and inter-firewall 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”.

Although firewall security has been given strong attention in the research community, the emphasis was mostly on the filtering performance issues [6, 9, 11, 12, 24]. On the other hand, few related work [7, 11] attempt to address only one of the conflict problems which is the rule correlation in filtering policies. Other approaches [3, 10, 13, 15, 17, 25] 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 stand-alone tools such as the Firewall Policy Advisor. In addition, none of the previous work 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 legacy firewalls.

¹This research was supported in part by National Science Foundation under Grant No. DAS-0353168 and by Cisco Systems. Any opinions, findings, conclusions or recommendations stated in this material are those of the authors and do not necessarily reflect the views of the funding sources.

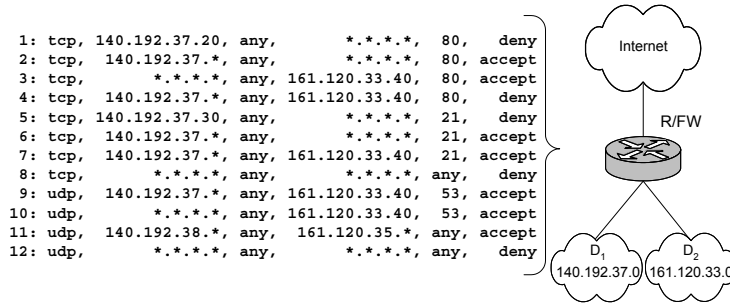


Figure 1. An example for centralized firewall setup.

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 then we describe the intra-firewall anomaly discovery algorithm. In Section V, we classify and define policy anomalies in distributed firewalls, and then we describe the inter-firewall 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, 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 range of values. 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 [4, 5]. In this paper, we assume a “deny” default policy action.

Filtering Rule Format: It is possible to use any field in IP, UDP or 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 [6, 23]. The following is the common format of packet filtering rules in a firewall policy:

```
<order><protocol><s_ip><s_port><d_ip><d_port><action>
```

An example of typical firewall rules is shown in Figure 1.

III. FIREWALL POLICY MODELLING

Modelling 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.

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 there is no other relation exists. We determine the relations based on comparing the network fields of filtering rules as follows.

Definition 1: Rules R_x and R_y are *completely disjoint* if every field in R_x is not a subset and not a superset and not equal to the corresponding field in R_y .

Formally, $R_x \mathfrak{R}_{CD} R_y$ iff

$$\forall i : R_x[i] \not\bowtie R_y[i]$$

where $\bowtie \in \{C, \supset, =\}$,

$$i \in \{\text{protocol, s_ip, s_port, d_ip, d_port}\}$$

Definition 2: Rules R_x and R_y are *exactly matched* if every field in R_x is equal to the corresponding field in R_y .

Formally, $R_x \mathfrak{R}_{EM} R_y$ iff

$$\forall i : R_x[i] = R_y[i]$$

where $i \in \{\text{protocol, s_ip, s_port, d_ip, d_port}\}$

Definition 3: Rules R_x and R_y are *inclusively matched* 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, $R_x \mathfrak{R}_{IM} R_y$ iff

$$\forall i : R_x[i] \subseteq R_y[i]$$

and $\exists j$ such that $R_x[j] \neq R_y[j]$

where $i, j \in \{\text{protocol, s_ip, s_port, d_ip, d_port}\}$

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

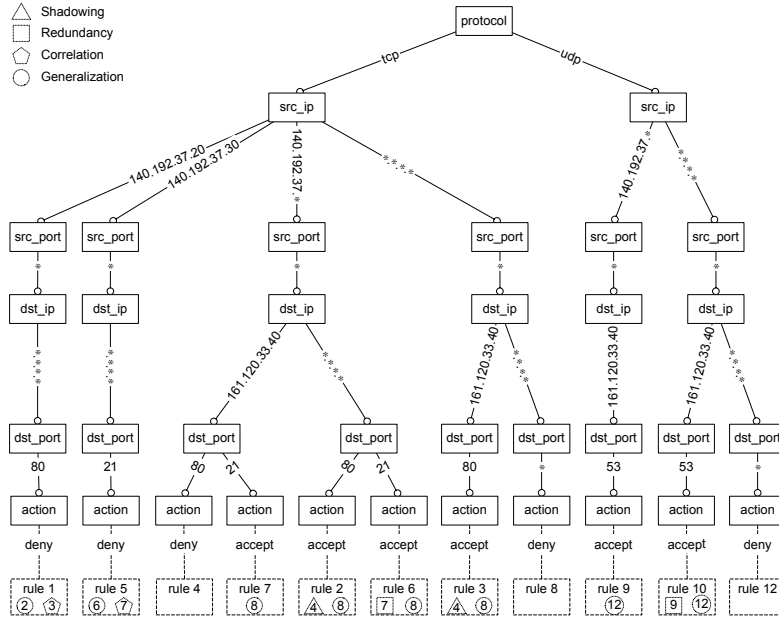


Figure 2. The policy tree for the firewall policy in Figure 1.

Definition 4: Rules R_x and R_y are *partially disjoint* (or *partially matched*) 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, $R_x \mathfrak{R}_{PD} R_y$ iff

$$\begin{aligned} &\exists i, j \text{ such that } R_x[i] \bowtie R_y[i] \text{ and } R_x[j] \not\bowtie R_y[j] \\ &\text{where } \bowtie \in \{ \subset, \supset, = \}, \\ &i, j \in \{ \text{protocol, s_ip, s_port, d_ip, d_port} \}, i \neq j \end{aligned}$$

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

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, $R_x \mathfrak{R}_C R_y$ iff

$$\begin{aligned} &\forall i : R_x[i] \bowtie R_y[i] \text{ and} \\ &\exists j, k \text{ such that } : R_x[j] \subset R_y[j] \text{ and } R_x[k] \supset R_y[k] \\ &\text{where } \bowtie \in \{ \subset, \supset, = \}, \\ &j, k \in \{ \text{protocol, s_ip, s_port, d_ip, d_port} \}, j \neq k \end{aligned}$$

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

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. The complete proofs for the theorems are presented in [2].

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

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

B. Firewall Policy Representation

We represent the firewall policy by a single-rooted tree called the *policy tree* [1]. 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. The root node of the policy tree represents the protocol field, the leaf nodes represent the action field, and intermediate nodes represent other 5-tuple filter fields in order. 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.

Figure 2 illustrates the policy tree model of the filtering policy given in Figure 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.

IV. INTRA-FIREWALL ANOMALY DISCOVERY

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 inter-related. 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 intra-firewall 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.

A. Intra-Firewall Anomaly Classification

Here, we describe and then formally define the possible intra-firewall policy 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:

$$R_x[\text{order}] < R_y[\text{order}], R_x \mathfrak{R}_{EM} R_y, R_x[\text{action}] \neq R_y[\text{action}]$$

$$R_x[\text{order}] < R_y[\text{order}], R_y \mathfrak{R}_{IM} R_x, R_x[\text{action}] \neq R_y[\text{action}]$$

For example, Rule 4 in shadowed by Rule 3 in Figure 1. Shadowing is a critical error in the policy, as the shadowed rule never takes effect. This might cause an accepted traffic to be blocked or a denied traffic to be permitted. Therefore, as a general guideline, if there is an inclusive or exact match relationship between two rules, the superset (or general) rule should come after the subset (or specific) rule. It is important to discover shadowed rules and alert the administrator to correct this error by reordering or removing these rules.

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:

$$R_x \mathfrak{R}_C R_y, R_x[\text{action}] \neq R_y[\text{action}]$$

Rule 1 is in correlation with Rule 3 in Figure 1. The two rules with this ordering imply that all HTTP traffic that is coming from 140.192.37.20 and going to 161.120.33.40 is denied. However, if their order is reversed, the same traffic will be accepted. Correlation is considered an anomaly warning because the correlated rules imply an action that is

not explicitly stated by the filtering rules. Therefore, in order to resolve this conflict, we point out the correlation between the rules and prompt the user to choose the proper order that complies with the security policy requirements.

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

$$R_x[\text{order}] < R_y[\text{order}], R_x \mathfrak{R}_{IM} R_y, R_x[\text{action}] \neq R_y[\text{action}]$$

Rule 2 is a generalization of Rule 1 in Figure 1. These two rules imply that all HTTP traffic that is coming from the address 140.192.37.* will be accepted, except the traffic coming from 140.192.37.20. Generalization is often used to exclude a specific part of the traffic from a general filtering action. It is considered only an anomaly warning because the specific rule makes an exception of the general rule. This might cause an accepted traffic to be blocked or a denied traffic to be permitted, and thus it is important to highlight its action to the administrator for confirmation.

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:

$$R_x[\text{order}] < R_y[\text{order}], R_x \mathfrak{R}_{EM} R_y, R_x[\text{action}] = R_y[\text{action}]$$

$$R_x[\text{order}] < R_y[\text{order}], R_y \mathfrak{R}_{IM} R_x, R_x[\text{action}] = R_y[\text{action}]$$

Whereas rule R_x is redundant to rule R_y if:

$$R_x[\text{order}] < R_y[\text{order}], R_x \mathfrak{R}_{IM} R_y, R_x[\text{action}] = R_y[\text{action}]$$

and $\nexists R_z$ where $R_x[\text{order}] < R_z[\text{order}] < R_y[\text{order}]$,

$$R_x \{ \mathfrak{R}_{IM}, \mathfrak{R}_C \} R_z, R_x[\text{action}] \neq R_z[\text{action}]$$

Referring to Figure 1, Rule 7 is redundant to Rule 6, and Rule 9 is redundant to Rule 10. Redundancy is considered an error in the firewall policy because a redundant rule adds to the size of the filtering rule list, and therefore increases the search time and space requirements of the packet filtering process [18]. In general, to avoid redundant rules, a superset rule following a subset rule should have an opposite filtering action. It is important to discover redundant rules so that the administrator may modify the filtering action or remove the rules from the policy.

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:

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

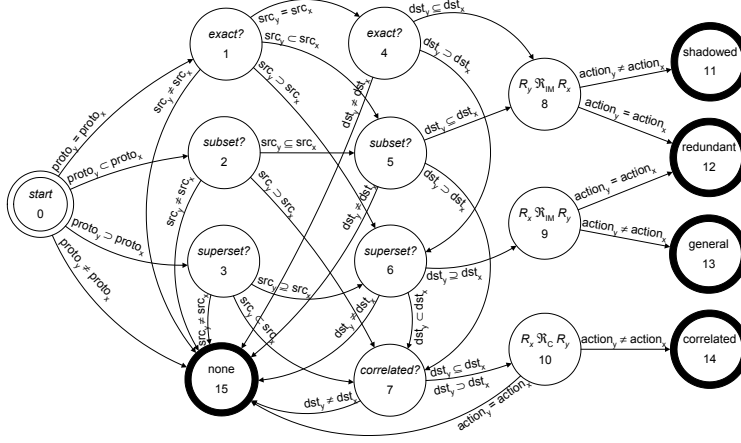


Figure 3. State diagram for detecting intra-firewall anomalies for rules R_x and R_y , where R_y comes after R_x .

Referring to Figure 1, Rule 11 is irrelevant because the traffic that goes between the source (140.192.38.*) and the destination (161.120.35.*) does not pass through this firewall.

Irrelevance is considered an anomaly because it adds unnecessary overhead to the filtering process and it does not contribute to the policy semantics. It is well-understood that keeping the filtering rule table as small as possible helps in improving the overall firewall performance [18, 23]. Thus, discovering irrelevant rules is an important function for the network security administrator.

B. Intra-Firewall Anomaly Discovery Algorithm

The state diagram in Figure 3 illustrates intra-firewall anomaly discovery states for any two rules, R_x and R_y , where the two rules are in the same firewall and R_y follows R_x . For simplicity, the address and port fields are integrated in one field for both the source and destination. This reduces the number of states and simplifies the explanation of the diagram.

Initially no relationship is assumed. Each field in R_y is compared to the corresponding field in R_x starting with the protocol, then source address and port, and finally destination address and port. The relationship between the two rules is determined based on the result of subsequent comparisons. If every field of R_y is a subset or equal to the corresponding field in R_x and both rules have the same action, R_y is redundant to R_x , while if the actions are different, R_y is shadowed by R_x . If every field of R_y is a superset or equal to the corresponding field in R_x and both rules have the same action, R_x is potentially redundant to R_y , while if the actions are different, R_y is a generalization of R_x . If some fields of R_x are subsets or equal to the corresponding fields in R_y , and some fields of R_x are supersets to the corresponding fields in R_y , and their actions are different, then R_x is in correlation with R_y . Identifying irrelevant rules requires the knowledge of the network connectivity. Discovering this intra-firewall anomaly is discussed later in Section V-C along with the inter-firewall anomaly discovery algorithm. If none of the preceding cases occur, then the two rules do not involve any anomalies.

The basic idea for discovering anomalies is to determine 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 intra-firewall anomaly definitions in Section IV-A. If rule paths do not coincide, then these rules are disjoint and they have no anomalies. The detailed description of the intra-firewall anomaly discovery algorithm is available in [2]. Applying the algorithm on the rules in Figure 1, the discovered anomalies are marked in the dotted boxes at the bottom of the policy tree in Figure 2. Shadowed rules are marked with a triangle, redundant rules with a square, correlated rules with a pentagon and generalization rules with a circle.

V. INTER-FIREWALL ANOMALY DISCOVERY

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. Multi-firewall installation also provides inter-domain 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-A, 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 inter-firewall “shadowing” which similar in principle to rule shadowing discussed in the intra-firewall 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 inter-firewall anomalies and we describe a technique to discover these anomalies.

A. Inter-Firewall Anomaly Definition

In general, an inter-firewall 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 sub-domains where the firewalls are installed at the routing points in the network.

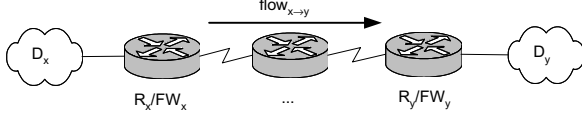


Figure 4. Cascaded firewalls isolating domains D_x and D_y .

Referring to Figure 4, we assume a traffic stream flowing from sub-domain D_x to sub-domain D_y across multiple cascaded firewalls installed on the network path between the two sub-domains. 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 sub-domain (FW_x) is called the *most-upstream firewall*, while The closest firewall to the flow destination sub-domain (FW_y) is called the *most-downstream firewall*.

Using the above network model, we can say that for any traffic flowing from sub-domain D_x to sub-domain 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. Inter-Firewall Anomaly Classification

In this section, we classify anomalies in multi-firewall environments. Our classification rules are based on the basic case of cascaded firewalls illustrated in Figure 4, 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 intra-firewall shadowing or redundancy exists in any individual firewall. As illustrated in Section IV-A, this implies that every “deny” rule should be followed by a more general “accept” rule, and the default action of unspecified traffic is “deny”.

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:

$$R_d \mathfrak{R}_{EM} R_u, R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{accept} \quad (1)$$

$$R_d \mathfrak{R}_{IM} R_u, R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{accept} \quad (2)$$

$$R_u \mathfrak{R}_{IM} R_d, R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{accept} \quad (3)$$

$$R_u \mathfrak{R}_{IM} R_d, R_u[\text{action}] = \text{accept}, R_d[\text{action}] = \text{accept} \quad (4)$$

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 Figure 5 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 Figure 5 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:

$$R_u \mathfrak{R}_{EM} R_d, R_u[\text{action}] = \text{accept}, R_d[\text{action}] = \text{deny} \quad (5)$$

$$R_u \mathfrak{R}_{IM} R_d, R_u[\text{action}] = \text{accept}, R_d[\text{action}] = \text{deny} \quad (6)$$

$$R_d \mathfrak{R}_{IM} R_u, R_u[\text{action}] = \text{accept}, R_d[\text{action}] = \text{deny} \quad (7)$$

$$R_d \mathfrak{R}_{IM} R_u, R_u[\text{action}] = \text{accept}, R_d[\text{action}] = \text{accept} \quad (8)$$

$$R_u \mathfrak{R}_{IM} R_d, R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{deny} \quad (9)$$

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 Figure 5 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 Figure 5 respectively. Case (9) is not as obvious as the previous cases and it needs further analysis. Since we assume there is no intra-firewall 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 not in correlation with R_d . Rules (5/ FW_0 , 4/ FW_1) in Figure 5 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:

$$R_d \mathfrak{R}_{EM} R_u, R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{deny} \quad (10)$$

$$R_d \mathfrak{R}_{IM} R_u, R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{deny} \quad (11)$$

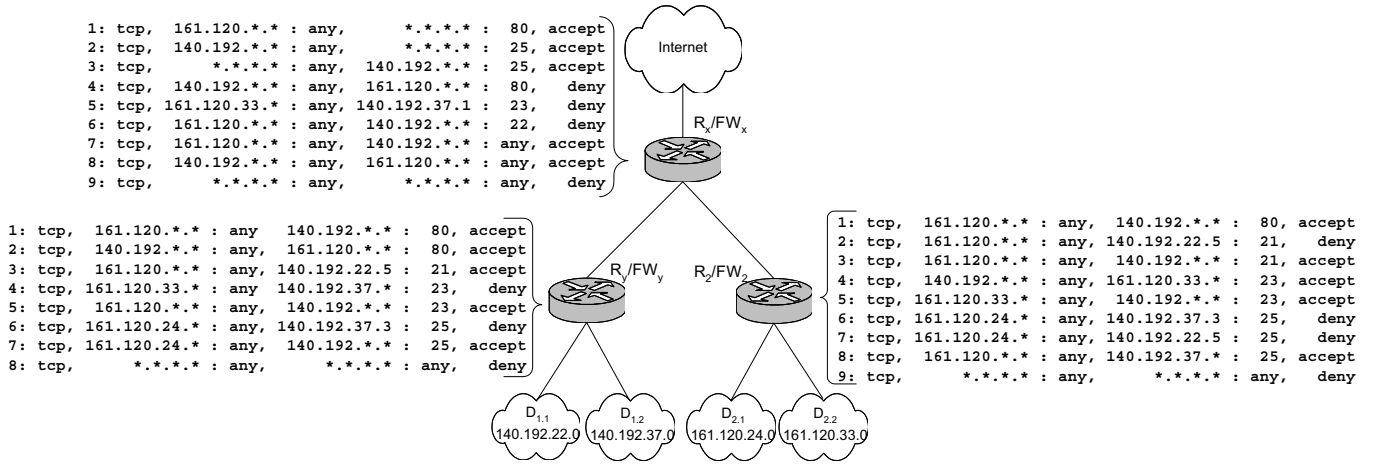


Figure 5. An example for a hierarchical distributed firewall setup.

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 Figure 5, 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. Intra-firewall 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 inter-firewall 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:

$$R_u \mathcal{R}_C R_d, R_u[\text{action}] = \text{accept}, R_d[\text{action}] = \text{accept} \quad (12)$$

$$R_u \mathcal{R}_C R_d, R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{deny} \quad (13)$$

$$R_u \mathcal{R}_C R_d, R_u[\text{action}] = \text{accept}, R_d[\text{action}] = \text{deny} \quad (14)$$

$$R_u \mathcal{R}_C R_d, R_u[\text{action}] = \text{deny}, R_d[\text{action}] = \text{accept} \quad (15)$$

An example for case (12) is

$$R_u : \text{tcp}, 140.192.*.* : \text{any}, 161.120.33.* : 80, \text{accept}$$

$$R_d : \text{tcp}, 140.192.37.* : \text{any}, 161.120.*.* : 80, \text{accept}$$

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

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

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

$$R_u : \text{tcp}, 140.192.*.* : \text{any}, 161.120.33.* : 80, \text{deny}$$

$$R_d : \text{tcp}, 140.192.37.* : \text{any}, 161.120.*.* : 80, \text{deny}$$

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

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

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

$$R_u : \text{tcp}, 140.192.*.* : \text{any}, 161.120.33.* : 80, \text{accept}$$

$$R_d : \text{tcp}, 140.192.37.* : \text{any}, 161.120.*.* : 80, \text{deny}$$

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

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

As for case (15), the example is

$$R_u : \text{tcp}, 140.192.*.* : \text{any}, 161.120.33.* : 80, \text{deny}$$

$$R_d : \text{tcp}, 140.192.37.* : \text{any}, 161.120.*.* : 80, \text{accept}$$

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

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

VI. FIREWALL POLICY EDITING

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 sub-domain, 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 intra-firewall 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 avoids 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 inter-firewall anomalies, and helps to determine the proper order for the rule within these firewalls avoiding intra-firewall anomalies, and provides visual aids for users to track and verify policy changes. Using the policy editor, administrators require no prior knowledge or understating of the firewall policy in order to insert, modify or remove a rule.

A. Rule Insertion

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 deployed. This is needed in order to apply the filtering rule only on the relevant sub-domains without creating any inter-firewall anomalies. The second step is to determine the proper order of the rule in each firewall so that no intra-firewall anomaly is created.

In the first step, we first identify all the possible paths that go from the source address to the destination address of the rule. 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, the rule is inserted only in the most-upstream firewalls(s) relative to the source(s). As an example, the following two rules are inserted in the security policy shown in Figure 5:

R_1 : icmp, *.*.*.*,any, 140.192.*.*,any, deny

R_2 : icmp, 140.192.*.*,any, 161.120.*.*,any, accept

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. On the other hand, R_2 is installed in firewalls FW_0 , FW_1 and FW_2 because 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. The local policy tree is used 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. 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. On the other hand, if the rule is disjoint, then it can given any order in the policy. Similarly, the tree browsing continues evaluating the next fields in the rule recursively as long as the field value is an exact match or a subset match of a branch. 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. A new branch is created for the new rule any time a disjoint or superset match is found. 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.

After inserting the rule in the appropriate firewalls, the inter-firewall anomaly discovery algorithm in Section V-C is activated to verify that no intra-firewall or inter-firewall 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

In distributed firewall environments, removing a rule from a specific firewall may result in creating an inter-firewall 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 sub-domains that will be impacted by removing this rule. We use the same technique described in rule insertion process to determine the network path between every source-destination domain pair relevant to this rule. 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 the firewalls in all paths from source to destination. Otherwise, shadowing and/or spuriousness anomaly is created if the rule is removed from the upstream and/or the downstream firewalls respectively. However, if the rule is a “deny” rule, then we just remove it from the local firewall and alert the administrator of the extra traffic that will be permitted as a result of removing this rule. After removing the rule, we run the inter-firewall anomaly analysis in order to highlight any potential anomalies that might be introduced in the modified policy.

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.

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

Figure 7. The 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%

Figure 9. The average percentage of discovered anomalies in a man-written distributed firewall policy.

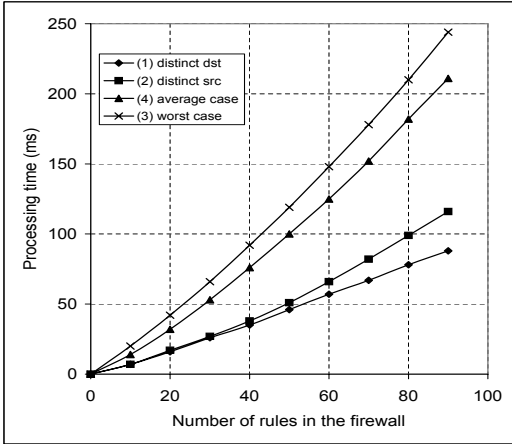


Figure 8. Processing time for intra-firewall anomaly discovery.

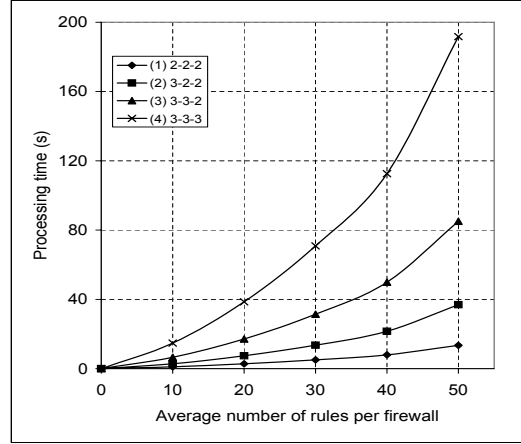


Figure 10. Processing time for inter-firewall anomaly discovery.

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 FPA². The tool implements the inter-firewall and intra-firewall 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 twelve 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 Figures 7 and 9 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 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 intra-firewall 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

²<http://www.mnlab.cti.depaul.edu/mnlab/projects/FPA>

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 intra-firewall 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 Figure 8. 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 inter-firewall anomaly discovery algorithm, we conducted a similar experiment on a network of distributed firewalls. We used a balanced three-level hierarchical network topology connected to the Internet via the root node. Each non-leaf 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 2 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 inter-firewall 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 Figure 10. We noticed that for small and mid-size networks (such as Network 1 that has 8 sub-domains and Network 2 that has 12 sub-domains), the processing time ranges from 3 to 40 seconds. However, in case of large networks (such as Networks 3 and Network 4 that have 18 and 27 sub-domains respectively), the firewall anomaly discovery requires much higher processing time ranging from 11 to 180 seconds 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 sub-domains in the network.

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 modelling, 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 [12]. Another model using tuple space is developed in [22], which combines a set of filters in one tuple stored in a hash table. The model in [24] uses bucket filters indexed by search trees. Multi-dimensional binary tries are also used to model filters [19]. In [7] a geometric model is used to represent 2-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. 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 [15] classify possible policy conflicts in role-based management frameworks, and develop techniques to discover them. A policy conflict scheme for IPSec is presented in [9]. 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 [7] and [11] 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 go one step forward by offering query-based tools for firewall policy analysis. In [17] and [25], 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. In [8], an expert system is used for verifying the functionality of filtering rules by performing queries. All these tools help in manually verifying the correctness of the firewall policy, however, they require high user expertise to write the proper queries to identify different 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 [10], which proposes a global policy definition language along with algorithms for verifying the policy and generating filtering rules. However, in [3] 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 [13] with a procedural policy definition language, and in [16] with an object-oriented policy definition language. In terms of distributed firewall policy enforcement, a novel architecture is proposed in [14] 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. CONCLUSIONS 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 sub-domains 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 legacy 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 (intra-firewall anomalies), and between inter-connected firewalls (inter-firewall 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 intra- and inter-firewall 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 sub-domains with 13 firewalls), it takes 20-180 seconds 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 implementation optimization of intra- and inter-firewall anomaly discovery, online automatic discovery and recovery of anomalies created as a result of the rule editing, rule placement based on firewall performance, self-configurable firewalls, translating low-level filtering rules into high-level textual description and vice versa.

REFERENCES

- [1] E. Al-Shaer and H. Hamed. "Firewall Policy Advisor for Anomaly Detection and Rule Editing." *IEEE/IFIP Integrated Management Conference (IM'2003)*, March 2003.
- [2] E. Al-Shaer and H. Hamed. "Design and Implementation of Firewall Policy Advisor Tools." *DePaul CTI Technical Report, CTI-TR-02-006*, August 2002.
- [3] Y. Bartal, A. Mayer, K. Nissim and A. Wool. "Firmato: A Novel Firewall Management Toolkit." *Proceedings of 1999 IEEE Symposium on Security and Privacy*, May 1999.
- [4] D. Chapman and E. Zwicky. *Building Internet Firewalls, Second Edition*, Orielly & Associates Inc., 2000.
- [5] W. Cheswick and S. Belovin. *Firewalls and Internet Security*, Addison-Wesley, 1995.
- [6] S. Cobb. "ICSA Firewall Policy Guide v2.0." NCSA Security White Paper Series, 1997.
- [7] D. Eppstein and S. Muthukrishnan. "Internet Packet Filter Management and Rectangle Geometry." *Proceedings of 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, January 2001.
- [8] P. Eronen and J. Zitting. "An Expert System for Analyzing Firewall Rules." *Proceedings of 6th Nordic Workshop on Secure IT-Systems (NordSec 2001)*, November 2001.
- [9] Z. Fu, F. Wu, H. Huang, K. Loh, F. Gong, I. Baldine and C. Xu. "IPSec/VPN Security Policy: Correctness, Conflict Detection and Resolution." *Proceedings of Policy'2001 Workshop*, January 2001.
- [10] J. Guttman. "Filtering Posture: Local Enforcement for Global Policies." *Proceedings of 1997 IEEE Symposium on security and Privacy*, May 1997.
- [11] B. Hari, S. Suri and G. Parulkar. "Detecting and Resolving Packet Filter Conflicts." *Proceedings of IEEE INFOCOM'00*, March 2000.
- [12] S. Hazelhurst. "Algorithms for Analyzing Firewall and Router Access Lists." *Technical Report TR-WitsCS-1999*, Department of Computer Science, University of the Witwatersrand, South Africa, July 1999.
- [13] S. Hinrichs. "Policy-Based Management: Bridging the Gap." *Proceedings of 15th Annual Computer Security Applications Conference (ACSAC'99)*, December 1999.
- [14] S. Ioannidis, A. Keromytis, S. Bellovin and J. Smith. "Implementing a Distributed Firewall." *Proceedings of 7th ACM Conference on Computer and Communications Security (CCS'00)*, November 2000.
- [15] E. Lupu and M. Sloman. "Model-Based Tool Assistance for Packet-Filter Design." *Proceedings of Workshop on Policies for Distributed Systems and Networks (POLICY'2001)*, January 2001.
- [16] I. Lck. C. Schfer and H. Krumm. "Conflict Analysis for Management Policies." *Proceedings of IFIP/IEEE International Symposium on Integrated Network Management (IM'1997)*, May 1997.
- [17] A. Mayer, A. Wool and E. Ziskind. "Fang: A Firewall Analysis Engine." *Proceedings of 2000 IEEE Symposium on Security and Privacy*, May 2000.
- [18] R. Panko. *Corporate Computer and Network Security*, Prentice Hall, 2003.
- [19] L. Qiu, G. Varghese, and S. Suri. "Fast Firewall Implementations for Software and Hardware-based Routers." *Proceedings of 9th International Conference on Network Protocols (ICNP'2001)*, November 2001.
- [20] R. Smith, S. Bhattachayra. "Firewall Placement in a Large Network Topology." *Proceedings of 6th Workshop of Future Trends of Distributed Computing (FTDCS'97)*, October 1997.
- [21] R. Smith, S. Bhattachayra. "A Protocol and Simulation for Distributed Communicating Firewalls." *Proceedings of 23rd IEEE International Computer Software and Applications Conference (COMSPAC'99)*, October 1999.
- [22] V. Srinivasan, S. Suri and G. Varghese. "Packet Classification Using Tuple Space Search." *Computer ACM SIGCOMM Communication Review*, October 1999.
- [23] J. Wack, K. Cutler and J. Pole. "Guidelines on Firewalls and Firewall Policy." *NIST Recommendations, SP 800-41*, January 2002.
- [24] T. Woo. "A Modular Approach to Packet Classification: Algorithms and Results." *Proceedings of IEEE INFOCOM'00*, March 2000.
- [25] A. Wool. "Architecting the Lumeta Firewall Analyzer." *Proceedings of 10th USENIX Security Symposium*, August 2001.
- [26] "Cisco Secure Policy Manager 2.3 Data Sheet." http://www.cisco.com/warp/public/cc/pd/sqsw/sqppmn/prodlit/spmgr_ds.pdf
- [27] "Check Point Visual Policy Editor Data Sheet." http://www.checkpoint.com/products/downloads/vpe_datasheet.pdf