Policy Algebras for Hybrid Firewalls

Hang Zhao Steven M. Bellovin
{zhao,smb}@cs.columbia.edu
Columbia University*

Technical Report CUCS-017-07

Abstract

Firewalls are a effective means of protecting a local system or network of systems from network-based security threats. In this paper, we propose a policy algebra framework for security policy enforcement in hybrid firewalls, ones that exist both in the network and on end systems. To preserve the security semantics, the policy algebras provide a formalism to compute addition, conjunction, subtraction, and summation on rule sets; it also defines the cost and risk functions associated with policy enforcement. Policy outsourcing triggers global cost minimization. We show that our framework can easily be extended to support packet filter firewall policies. Finally, we discuss special challenges and requirements for applying the policy algebra framework to MANETs.

1 Introduction

Firewalls are a effective means of protecting a local system or network of systems from network-based security threats. Conventional firewalls rely on the notion of restricted topology and control entry points to to the network. All traffic from the inside to outside, and vice versa, must pass through the firewall. Only authorized traffic, as defined by the local security policy, will be allowed to pass. Since the traditional approach made assumptions on restricted network topology and trust on every single host in the intranet, the concept of distributed firewalls [Bel99] was proposed to address the shortcomings. In the distributed firewall scheme, security policy is still centrally defined, but enforcement is pushed to the individual end hosts. These two approaches can be combined together to achieve desired functionality with lower cost. In a hybrid implementation, some hosts are behind the traditional firewalls, while others live on the outside and can be protected using distributed firewall approach; alternatively, IPSEC may be ignored entirely and instead address-dependent rules are distributed and enforced by individual end hosts.

There are many possible arrangement for security policies. For a given initial policy, we wish to find one that has the following properties:

1. It verifiably has the same security semantics as the original;
2. It minimizes total system cost;
3. It keeps the risk within acceptable bounds.

In this paper, we propose a policy algebra framework for security policy enforcement in hybrid firewalls. The advantages of having such a framework to compose security polices are many. First of all, the policy algebras provide a formalism to compute the addition, conjunction, subtraction, summation on rule sets.

Secondly, a policy algebra frame facilitates the decentralization of security rule enforcement. By applying the algebra, different rule arrangements can be proposed. The compositional framework allows specifiers of

*Research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

policies to determine if the policy in fact result in the same accesses as the old, thus ensuring that property 1 holds.

Thirdly, firewall configuration still remains a crucial task. A quantitative study of firewall configuration [Woo04], shows that corporate firewalls are often enforcing poorly written rule sets. For well-configured firewalls, a small rule set or configuration file is highly recommended. By integrating rules or subtracting the rules that have already been enforced in the local environment, the size of individual rule sets is reduced.

Finally, a policy algebra enables rule outsourcing for hybrid firewalls. Modern heterogeneous networks consist of nodes with diversified energy storage and computation capacity. A host with low computational power can outsource part of its rule set to a security policy delegate that it trusts, thus lowering costs (property 2. In addition, the cost constraints can be made subject to the acceptable risk level (property 3. For instance, the policy algebra helps the specifier to answer the following questions: What is the effective filtering that reaches host A given the fact the A's upstream neighbors have already enforced their rule sets? What is the minimal set of rules A could push to its neighbors in order to achieve the same result without local filtering?

The rest of the paper is organized as follows. Section 2 introduce the policy algebra framework in detail. Section 3 elaborates how the framework can be further extended to support packet filter firewall policies. Section 4 discusses the specific challenges faced when importing the policy algebra framework in mobile ad hoc networks (MANET). Related works are described in section 5 and section 6 concludes the paper.

2 Policy Algebra Framework

In this section, we present the policy algebra framework in detail. To make our approach applicable for general security policies in hybrid firewalls, we do not make any assumption on the policy language which states what sort of connections are permitted or prohibited.

2.1 Policies and Rules

To resolve the ambiguity in our discussion, we formally define a policy $\mathcal{P}$ as an unordered set of rules, denoted as $\mathcal{P} = \{r_1, r_2, ..., r_k\}$ and $|\mathcal{P}| = k$. How to construct the rule set for a given policy is determined by the specific policy language; the details are irrelevant for our discussion. Without loss of generality, one can view the rule set $\{r_1, r_2, ..., r_k\}$ as a detailed interpretation of a security policy $\mathcal{P}$ to be enforced in the network. An action $\mathcal{A}$ is said to be accepted by the policy if it matches any rule in the rule set. To facilitate our discussion, we define a universal policy $\mathcal{U}$ that accepts any kind of actions and also an empty policy ϕ that accepts nothing.

2.2 The Policy Algebra

Intuitively, a policy algebra could be represented as rule set operations. Compound policies can be obtained by combining policy rule sets through the algebra operators. We formally define the following algebra operators over rule sets:

- *Addition* $(+)$ merges two policy rule sets by returning their set unions provided that there is no conflict among the rules involved, denoted as $\mathcal{P}_1 + \mathcal{P}_2 = \mathcal{P}_3$, where $|\mathcal{P}_3| \leq |\mathcal{P}_1| + |\mathcal{P}_2|$. If the combination of two rule sets arises conflict, it must be resolved using a decorrelation algorithm; we discuss that further in the following section. The property of the *Addition* $(+)$ operation ensures that any action $\mathcal{A}$ accepted by $\mathcal{P}_1$ or $\mathcal{P}_2$ will be accepted by $\mathcal{P}_3$ as well.

- *Conjunction* $(\cap)$ merges two policy rule sets by returning their intersection, denoted as $\mathcal{P}_1 \cap \mathcal{P}_2 = \mathcal{P}_3$, where $|\mathcal{P}_3| \leq |\mathcal{P}_1|$ and $|\mathcal{P}_3| \leq |\mathcal{P}_2|$. Any action $\mathcal{A}$ accepted by $\mathcal{P}_3$ should be accepted by $\mathcal{P}_1$ and $\mathcal{P}_2$ as well.

- *Subtraction* $(-)$ reduces a policy rule set by eliminating all the rules in the second policy, denoted as $\mathcal{P}_1 - \mathcal{P}_2 = \mathcal{P}_3$, where $|\mathcal{P}_3| \leq |\mathcal{P}_1|$. If a given action $\mathcal{A}$ is accepted by $\mathcal{P}_1$ but not $\mathcal{P}_3$, then it must be accepted by $\mathcal{P}_2$.

- *Summation* ($\sum$) is defined as a list of *Addition* (+) operations, where $\sum_{i=1}^{n} \mathcal{P}_i = \mathcal{P}_1 + \mathcal{P}_2 + ... + \mathcal{P}_n = \mathcal{P}$ and $|\mathcal{P}| \leq |\mathcal{P}_1| + |\mathcal{P}_2| + ... + |\mathcal{P}_n|$. Conflict resolution is necessary if conflicts arise during summation. Intuitively, an action $\mathcal{A}$ accepted by any policy rule set $\mathcal{P}_i$ will be accepted by $\mathcal{P}$.

The above policy algebra operations are interpreted as set operations. For instance, *Addition*$(+)$ operation keeps only distinct rules appearing in either $\mathcal{P}_1$ or $\mathcal{P}_2$. *Conjunction* $(\cap)$ returns the common rules appearing in both $\mathcal{P}_1$ and $\mathcal{P}_2$. *Subtraction* $(-)$ eliminates the rules existing in $\mathcal{P}_2$ from $\mathcal{P}_1$. Intuitively, one can view them as *external* operations since they do not modify any single policy rule; only the set operations are applied over the policy rule set. On the other hand, it is worth pointing out that the composition of individual rules may result in fewer rules to be enforced in the network. For instance, the composition of rule r_1 and rule r_2 produces a single rule r_3 that maintains the same access control effect that r_1 and r_2 together provide. This observation inspires us to define another set of algebra operators over individual rules. We refer them as *internal* operations since individual policy rules are modified during the operation. The most significant benefit of the *internal* operations is to reduce the size of rule set. As we discussed in the previous section, reduced rule set can improve the network traffic throughput. Thus we define the following algebra operators over individual rules:

- *Addition* $(+_r)$ merges two security policy rules by returning the set union of them, denoted as $r_1 +_r r_2 = r$, where r represents a single policy rule if r_1 and r_2 can be composed into one rule, otherwise $r = \{r_1, r_2\}$. Note that *Addition* $(+_r)$ is non-commutative if the two rules are correlated. That is, the policy effect of applying r_1 followed by r_2 is not necessarily the same as applying r_2 first.

- *Conjunction* $(\cap_r)$ merges two security policy rules by returning their intersection, denoted as $r_1 \cap_r r_2 = r$, where $r = \phi$ if the two policy rules share no security effect in common. Any action $\mathcal{A}$ matches r should also match r_1 and r_2 as well.

- *Subtraction* $(-_r)$ reduces a policy rule set by eliminating the secure effect in the second policy rule, denoted as $r_1 - r_2 = r$. If a given action $\mathcal{A}$ matches r_1 but not r_3, then it must match rule r_2.

- *Summation* $(\sum_r)$ is defined as a list of *Addition* $(+_r)$ operations, where $\sum_{r\ i=1}^{n} r_i = r_1 +_r r_2 +_r ... +_r r_n = r$. Intuitively, an action $\mathcal{A}$ matches any of the policy rule r_i will also match rule r. In the worst case, none pair of rules can be composed using *Addition* $(+_r)$ operation, then *Summation* $(\sum_r)$ simply returns the set union of the rules and we have $r = \{r_1, r_2, ..., r_n\}$.

Note that not any two rules can be composed using the above *internal* algebra operations. For example, a policy rule on secure authentication obviously cannot be composed with another policy rule on file access control. Furthermore, the operands of the *internal* algebra operators must share the same syntax and semantics defined by the specific policy language.

2.3 The Network Model

We consider a heterogeneous network consisting of $\mathcal{N}$ nodes. Each host has its own characteristics, such as computational power. In a hybrid firewall implementation, whether a host is inside the traditional boundary, outside it, in parallel with it, or even integrated with it is irrelevant to our discussion on policy algebras.

Depending on the security goals to be achieved, each host u has its own set of policies to be enforced denoted as S_u; $|S_u|$ represents the number of distinct policies required by host u. Since the security policies are computed in a central unit before shipping out to destination hosts, we adopt the notation $\mathcal{P}_{(i,j)}$ to describe the jth policy for node n_i and use rule set $\mathcal{P}_i^*$ to denote the filtering effect host n_i has already observed in the local environment. We define the following equation:

$$\sum_{j=1}^{|S_{n_i}|} \mathcal{P}_{(i,j)} - \mathcal{P}_i^* = \sum_{j=1}^{|S'_{n_i}|} \mathcal{P}'_{(i,j)} \qquad (1)$$

where the *Summation* ($\sum$) operation aggregates all the policies required for node n_i and the *Subtraction* $(-)$ operation gives the remaining set of policies to be enforced after considering the local filtering effect.

Moreover, we have

$$\sum_{i=1}^{N}\sum_{j=1}^{|S_{n_i}|}\mathcal{P}_{(i,j)} - \sum_{i=1}^{N}\mathcal{P}_i^* = \sum_{i=1}^{N}\sum_{j=1}^{|S'_{n_i}|}\mathcal{P}'_{(i,j)} \tag{2}$$

Hence, the summation $\sum_{i=1}^{N}\sum_{j=1}^{|S'_{n_i}|}\mathcal{P}'_{(i,j)}$ describes the policy rule sets to be enforced for all the hosts in the entire network.

2.4 Cost and Risk Functions

Policy enforcement in firewalls is costly. It requires additional CPU cycles to perform rule checking; these tasks thus consume the host's computational energy and perhaps battery power. One of the goals for firewall configuration is to minimize the cost for rule enforcement while achieving the desired access control. Thus, we define a cost function associated with each policy enforcement denoted as $\mathcal{C}(\mathcal{P}_{(i,j)})$. Policy outsourcing enables a single policy $\mathcal{P}_{(i,j)}$ to be enforced at different locations in the network with, consequently, different costs. Intuitively, local host n_i does not experience $\mathcal{C}(\mathcal{P}_{(i,j)})$ if it decides to outsource this policy to some delegate in the network without any local enforcement. However the term $\mathcal{C}(\mathcal{P}_{(i,j)})$ is still included in the global cost computation.

Policy outsourcing in firewalls is risky. A host becomes vulnerable if its security policy delegates have been compromised by attackers. Different risk levels can be experienced if the host chooses to outsource its security policies to varying locations in the network. Consider a concrete example: if host A chooses to outsource rules to one of its direct neighbors, then host A is vulnerable if this neighbor node gets compromised. If, however, host A decides to ship out its rules to a upstream node that is a few hops away, then host A becomes vulnerable if any node along the path from A to its delegate is compromised. In another example, a host might be willing to split the risk by choosing a few rule delegates to avoid single point failure. Besides compromised hosts, compromised links also introduce risks in policy enforcement and delegation. Hence we use $\mathcal{R}(\mathcal{P}_{(i,j)})$ to represent the risk associated with this particular enforcement for the jth policy for node n_i. Intuitively, $\mathcal{R}(\mathcal{P}_{(i,j)})$ experiences the lowest risk level if host n_i chooses the implement $\mathcal{P}_{(i,j)}$ by itself.

We need to consider both the cost and the risk for policy enforcement in hybrid firewall configurations. Thus we compute the tuple $< cost, risk > = < \mathcal{C}(\mathcal{P}_{(i,j)}), \mathcal{R}(\mathcal{P}_{(i,j)}) >$ associated with the enforcement of policy $\mathcal{P}_{(i,j)}$.

2.5 Global Minimization

Policy outsourcing, as described in the previous section, determines that a single policy $\mathcal{P}_{(i,j)}$ can be enforced at different locations in the network with varying $< \mathcal{C}(\mathcal{P}_{(i,j)}), \mathcal{R}(\mathcal{P}_{(i,j)}) >$ to achieve the same access control effects. The advantages of security policy outsourcing are many.

- It is an effective way to reduce the size of rule sets; it thus improves the network throughput by performing less rule checking.
- It balances the utilization of computational power in the whole network. Hosts with low energy can push part or all of their security polices to their upstream neighbors, which are more capable of expensive computations.
- It can also conserve energy for packet transmission by reducing the traffic in the network. Packets blocked by security policies in upstream nodes will never appear in the downstream path.
- Finally and most importantly, policy outsourcing enables global minimization of the costs and risks associated with rule enforcement in firewall configurations.

Given the formalism of policy algebras and cost/risk functions, we define the cost of policy enforcement for the entire network as

$$\mathcal{C}_G = \sum_{i=1}^{N}\sum_{j=1}^{|S'_{n_i}|}\mathcal{C}(\mathcal{P}'_{(i,j)}) \tag{3}$$

Similarly, the risk of policy enforcement for the entire network is represented as

$$\mathcal{R}_G = \sum_{i=1}^{N} \sum_{j=1}^{|S'_{n_i}|} \mathcal{R}(\mathcal{P}'_{(i,j)}) \tag{4}$$

Thus, global minimization on cost is achieved by looking for an optimal firewall configuration with policy outsourcing to result in the minimal value of $\mathcal{C}_G$. A similar global minimization can be performed on $\mathcal{R}_G$ as well.

Normally, one firewall configuration can help us to achieve either the minimal value of $\mathcal{C}_G$ or $\mathcal{R}_G$ but not both. Rule enforcement with lower cost may result in a higher risk level. As a concrete example, suppose host A decides to outsource policy $\mathcal{P}$ to its upstream neighbor B, which has more computational power. In this case, lower $\mathcal{C}(\mathcal{P})$ is incurred but with higher $\mathcal{R}(\mathcal{P})$ since A becomes vulnerable if any host along the path from A to B gets comprised by the attacker. Under most circumstances, we are more interested in minimizing the cost function $\mathcal{C}_G$. When such a trade off occurs during firewall configuration, it is the specifier's choice to make decisions on global cost minimization. One possible decision is to consider only the rules with risk lower than a threshold value λ_r. Thus the global cost minimization will compute $\mathcal{C}_G$, where every enforcement of $\mathcal{P}'_{(i,j)}$ satisfies the condition that $\mathcal{R}(\mathcal{P}'_{(i,j)}) \leq \lambda_r$.

3 Policy Algebra Instantiation

As discussed in the previous section, our policy algebra framework is applicable for general security policies. In this section, we elaborate how the policy algebra framework can be applied to packet filter firewalls as a concrete example. We choose packet filtering for illustration primarily because it is the original and most basic type of firewall. Packet filters are installed in many gateway routers, due to their relative simplicity and ease of implementation. They allows users to create a set of rules that either discard or accept traffic over a network connection. However, with the increasing complexity of network topology and security policy requirement, the rule set of packet filters can become extremely large and thus is hard to configure and maintain. Moreover, packet filters become the bottleneck of the network when the increased latency caused by filtering forces the interface buffer queue to drop further packets. Fortunately, with the policy algebra framework, one can perform optimization on packet filter rule sets and thus improve the throughput of the network.

3.1 Packet Filter Rules

A packet filter has a set of rules with accept or deny actions. When the packet filter receives a data packet, the filter compares the packet against a pre-configured rule set. At the first match, the packet filter either accepts or denies the packet. Filtering rules are based on information contained in a network packet: source and/or destination IP addresses, source and/or destination port numbers, IP protocol field, flags in TCP header if set, direction of inbound or outbound traffic, and which interface the packet is traversing. For the simplicity of our discussion, we consider very basic packet filer rules consisting of the most critical information: action, source IP address, source port number, destination IP address, destination port number. Simple packet filter rules look like the following:[1]

action	src	port	dest	port	comment
block	SPIGOT	*	*	*	we don't trust these people
allow	*	*	OUR-GW	25	connection to our SMTP port

These two rules specify a secure policy on our SMTP mail service. The first rule blocks any inbound traffic from SPIGOT to our internal network since we don't trust them. The second rule allows outside hosts other than SPIGOT to connect to our SMTP mail server on host OUR-GW at port 25. The rules are applied in order from top to bottom. Any packets that are not explicitly allowed by a filter rule are rejected. That is,

[1]Example taken from [CBR03].

any rule set is followed by an implicit rule saying:

action	src	port	dest	port	comment
block	*	*	*	*	default

3.2 Policy Algebra Refinement

To follow the previous notation, we describe a security policy as a set of packet filter rules, i.e, $\mathcal{P} = \{r_1, r_2, ..., r_k\}$. $\mathcal{P}$ could represent a security policy on accessing the internal SMTP mail server, handling DNS queries, or even the entire the packet filter rule set for a network host. Our previous definition on rule set represents $\mathcal{P}$ as an unordered set of rules. However, rule sets for packet filter firewall maintains strict order when rules are checked from top to bottom. Rules can be correlated to each other in the way that reversed order of rules or any subset of rules will have different filtering effect. Consider the following two rules:

action	src	port	dest	port	comment
allow	*	*	host-A	80	allows outsiders connect to host-A for web service
block	*	*	host-A	*	default

The filtering effect of the two rules indicate that only web traffic connected to port 80 on host-A is allowed whereas any other connection to host-A will be blocked. If we reverse the order of the two rules, no web traffic can reach host-A. On the other hand, if we enforce the second rule (a subset of rules) at a upstream neighbor of host-A, host-A will not receive any web traffic either. This ordering property introduce extra difficulty for *external* algebra operations since the order of packet filter rules may alter the semantics; even worse, the filtering effect cannot be maintained. The solution is to perform *decorrelation* on packet filter rules to remove any ambiguity. We describe each packet filter rule as $r = [action,\ IP_{src},\ Port_{src},\ IP_{dest},\ Port_{dest}]$, where the action is either allow or block in packet filter firewall rules. IP_{src} and IP_{dest} may represent one single IP address or a particular subnet of interest. Similarly, $Port_{src}$ and $Port_{dest}$ can also refer to a specific port number or a set of port numbers. We also have the default rule represented as $r^* = [block,\ *,\ *,\ *,\ *]$, which simply blocks any incoming or outgoing traffic. Given the detailed syntax and semantics of packet filter rules, two rules are considered to be correlated if there is a non-nil intersection between the values of each of their attributes (except for the $Action$ field). The goal of a decorrelation algorithm is to remove any existing correlations in the rule set. Once the rules are decorrelated, there is no longer any ordering requirement on the filtering rules, since only one rule will match any requested communication. [SC00] describes a decorrelation algorithm in detail. The previous example after decorrelation looks like this:

action	src	port	dest	port	comment
allow	*	*	host-A	80	allows outsiders connect to host-A for web service
block	*	*	not host-A	*	default

External operations like *Addition* $(+)$, *Conjunction* $(\cap)$, *Subtraction* $(-)$ and *Summation* $(\sum)$ for policy algebra are similar for most policy languages since they are interpreted as basic set operations. Here we want to focus our discussion on *internal* operations over individual packet filter rules since this part of the policy algebra framework is determined by the specific semantics of a policy language. Given $r_1 = [action_1,\ IP_{src_1},\ Port_{src_1},\ IP_{dest_1},\ Port_{dest_1}]$ and $r_2 = [action_2,\ IP_{src_2},\ Port_{src_2},\ IP_{dest_2},\ Port_{dest_2}]$, we then refine the *internal* algebra operators as following:

- *Addition* $(+_r)$ We have the following five cases:
  1. If $action_1 = action_2$, $Port_{src_1} = Port_{src_2}$, $IP_{dest_1} = IP_{dest_2}$ and $Port_{dest_1} = Port_{dest_2}$, then $r_1 +_r r_2 = [action_1,\ IP_{src_1} \cup IP_{src_2},\ Port_{src_1},\ IP_{dest_1},\ Port_{dest_1}]$;
  2. If $action_1 = action_2$, $IP_{src_1} = IP_{src_2}$, $IP_{dest_1} = IP_{dest_2}$ and $Port_{dest_1} = Port_{dest_2}$, then $r_1 +_r r_2 = [action_1,\ IP_{src_1},\ Port_{src_1} \cup Port_{src_2},\ IP_{dest_1},\ Port_{dest_1}]$;
  3. If $action_1 = action_2$, $IP_{src_1} = IP_{src_2}$, $Port_{src_1} = Port_{src_2}$ and $Port_{dest_1} = Port_{dest_2}$, then $r_1 +_r r_2 = [action_1,\ IP_{src_1},\ Port_{src_1},\ IP_{dest_1} \cup IP_{dest_2},\ Port_{dest_1}]$;

4. If $action_1 = action_2$, $IP_{src_1} = IP_{src_2}$, $Port_{src_1} = Port_{src_2}$ and $IP_{dest_1} = IP_{dest_2}$, then $r_1 +_r r_2 = [action_1,\ IP_{src_1},\ Port_{src_1},\ IP_{dest_1},\ Port_{dest_1} \cup Port_{dest_2}]$;
5. Otherwise, $r_1 +_r r_2 = \{r_1, r_2\}$.

- *Conjunction* $(\cap_r)$ if $action_1 = action_2$, then we have $r_1 \cap_r r_2 = [action_1,\ IP_{src_1} \cap IP_{src_2},\ Port_{src_1} \cap Port_{src_2},\ IP_{dest_1} \cap IP_{dest_2},\ Port_{dest_1} \cap Port_{dest_2}]$.
- *Subtraction* $(-_r)$ similar to the *Addition* $(+_r)$ operation, we also have the following five cases:
  1. If $action_1 = action_2$, $Port_{src_1} = Port_{src_2}$, $IP_{dest_1} = IP_{dest_2}$ and $Port_{dest_1} = Port_{dest_2}$, then $r_1 -_r r_2 = [action_1,\ IP_{src_1} \backslash IP_{src_2},\ Port_{src_1},\ IP_{dest_1},\ Port_{dest_1}]$;
  2. If $action_1 = action_2$, $IP_{src_1} = IP_{src_2}$, $IP_{dest_1} = IP_{dest_2}$ and $Port_{dest_1} = Port_{dest_2}$, then $r_1 -_r r_2 = [action_1,\ IP_{src_1},\ Port_{src_1} \backslash Port_{src_2},\ IP_{dest_1},\ Port_{dest_1}]$;
  3. If $action_1 = action_2$, $IP_{src_1} = IP_{src_2}$, $Port_{src_1} = Port_{src_2}$ and $Port_{dest_1} = Port_{dest_2}$, then $r_1 -_r r_2 = [action_1,\ IP_{src_1},\ Port_{src_1},\ IP_{dest_1} \backslash IP_{dest_2},\ Port_{dest_1}]$;
  4. If $action_1 = action_2$, $IP_{src_1} = IP_{src_2}$, $Port_{src_1} = Port_{src_2}$ and $IP_{dest_1} = IP_{dest_2}$, then $r_1 -_r r_2 = [action_1,\ IP_{src_1},\ Port_{src_1},\ IP_{dest_1},\ Port_{dest_1} \backslash Port_{dest_2}]$;
  5. Otherwise, $r_1 -_r r_2 = r_1$.
- *Summation* $(\sum_r)$ is defined as a list of *Addition* $(+_r)$ operations, where $\sum_{r\ i=1}^{n} r_i = r_1 +_r r_2 +_r ... +_r r_n = r$.

As a concrete example, consider the following set of packet filter rules:

rule	action	src	port	dest	port
r_1	allow	10.0.0.0/16	*	Mail-server	25
r_2	allow	10.0.0.0/8	*	Mail-server	25
r_3	block	SPIGOT	*	DMZ, Mail-server	25
r_4	block	SPIGOT	*	DMZ	25

Now we show how the *internal* algebra operators can be applied:

rule	action	src	port	dest	port
$r_1 +_r r_2$	allow	10.0.0.0/8	*	Mail-server	25
$r_3 \cap_r r_4$	block	SPIGOT	*	DMZ	25
$r_3 -_r r_4$	block	SPIGOT	*	Mail-server	25

3.3 Cost and Risk Functions

In reality, the size of rule set limits how well a packet filter can achieve its goal. A rule set is a linear list of individual rules, which are evaluated from top to bottom for a given packet. Packets have to pass through the filtering device, adding some amount of latency between the time a packet is received and the time it is forwarded. Any device can only process a finite amount of packets per second. When packets arrive at a higher rate than the device can forward them, packets are lost. The input processor passes packets to packet filter in a sequential manner. An obvious choice for the cost function associated with each packet filter firewall policy $\mathcal{P}$ is the number of rules in the rule set. As the packet filter rule set gets larger, it takes more time and computing resources to perform the filtering task. Although several rule set optimization mechanisms have been proposed, it remains the bottleneck for packet filter performance.

As discussed in the policy algebra framework, outsourcing packet filter rules to upstream neighbors can be risky. A benefit from rule outsourcing is that it conserves communication channel bandwidth by eliminating blocked traffic at upstream nodes. Thus, unauthorized packets will not even reach the target host. This also saves battery power on the transmitting node. Similarly, if the same packet filter rule is required by several downstream hosts, a composed rule can be enforced at the upstream node to maintain the same filtering effect. However, once this upstream node 9s compromised, the protected downstream hosts become vulnerable.

Intuitively, the number of hops from the enforcement delegate to the protected host can be used as the risk function.

3.4 Conflict Resolution

Multiple, correlated packet filter rules can cause conflicts when mapping packets to filters, as discussed in [HSP00]. In reality, we assume that with careful configuration by network administrator, such conflict will not occur within the local rule set of a network host. Unfortunately, security policy outsourcing may bring this problem back. Consider the following two rules:

rule	action	src	port	dest	port
r_a	block	10.0.0.0/8	*	*	25
r_b	allow	10.0.0.0/8	*	host-B	25

Rule r_a is required by upstream host-A such that any traffic connecting on port 25 from 10.0.0.0/8 is blocked. However, rule r_b enforced at downstream host-B allows 10.0.0.0/8 to connect to its mail service. If host-B wishes to outsource r_b to its upstream neighbor host-A, a conflict arises. Moreover, depending on the order of rules to be checked, the union of these two rules produces different filtering effects. If rule r_a is placed in front of rule r_b, then legitimate web traffic from 10.0.0.0/8 can never reach host-B; if rule r_b is checked first, then only web traffic from 10.0.0.0/8 destined for host-B is allowed. Obviously, the first approach marks the more restrictive rule with higher precedence, while the second approach may be desired if host-B is configured to handle web traffic from 10.0.0.0/8. One feasible solution is to apply decorrelation algorithms to the set of correlated rules. Thus the two rules become:

rule	action	src	port	dest	port
r_a	block	10.0.0.0/8	*	not host-B	25
r_b	allow	10.0.0.0/8	*	host-B	25

No matter what order these two rules are placed at host-A, it results in the same filtering effect as the second approach described above. Another possible solution is to simply disallow rule outsourcing if conflict can be detected. It is the network administrator's decision to resolve conflicts when applying the policy algebra framework. Our algebra cannot prevent conflicts; it can, however, reveal their existence.

3.5 Efficiency of Decorrelation Algorithm

As we discussed previously, the decorrelation algorithm is required to solve the ordering problem for packet filter rule sets and to resolve conflicts if possible. On the other hand, one of the objectives for policy outsourcing is to minimize the size of rule sets and to improve the filtering performance. Hence the efficiency of decorrelation algorithm becomes an interesting problem. Let us consider the following example:

rule	action	src	port	dest	port
r_1	allow	1.2.3.4	*	web-server	80
r_2	block	*	*	web-server	80

The filtering effect of applying rule r_1 and r_2 will block all http traffic and only allow source 1.2.3.4 to communicate with the web-server on port 80. To decorrelate this rule set, we have two options:

rule	action	src	port	dest	port
r_1	allow	1.2.3.4	*	web-server	80
r_2	block	0.0.0.0-1.2.3.3	*	web-server	80
r_3	block	1.2.3.5-255.255.255.255	*	web-server	80

or,

rule	action	src	port	dest	port
r'_1	allow	1.2.3.4	*	web-server	80
r'_2	block	not 1.2.3.4	*	web-server	80

The first approach results in more rules in the decorrelated rule set. If more check points like host 1.2.3.4 or more subnets are involved, as in many firewalls or routers, we may end up with a combinational explosion, resulting in a very inefficient system. Fortunately, the second approach, which uses negative rules, does not increase the size of rule set by as much. The decorrelation algorithm introduced in [SC00] takes this approach. Moreover, negative statements are often more natural to users, and sometimes allow a more efficient statement of the security policy. For instance, an administrator often needs to exclude a small set of machines from the normal host set for security purposes.

We claim that introducing negative expressions in the rule attributes improves the efficiency of decorrelation algorithm by generating a relatively small set of rules. However, the design issue becomes how to define and allow a simple, yet adequate, form of negative expressions in security policies [BMNW04].

4 Policy Algebra in MANET

Previous discussion on the policy algebra framework makes the general assumption of a heterogeneous network. In this section, we will describe the challenges and special requirements for applying the policy algebra framework in mobile ad-hoc networks (MANETs). Furthermore, we state several requirements such that the framework can be extended to become effective in a coalition MANET environment.

4.1 Challenges in MANET

A mobile ad-hoc network (MANET) is a self-configuring network of mobile routers and associated hosts connected by wireless links. The union of these mobile nodes forms an arbitrary network topology, changing rapidly and unpredictably with time. Such a network may operate on its own or be connected to the outside Internet. Firewall configuration in MANET faces many challenges.

- Mobile nodes join or leave the network unpredictably. Hybrid firewall implementation is a good choice for MANETs, since conventional firewalls strongly rely on a static topology with the concept of “inside” and “outside”. The policy algebra framework facilitates policy enforcement at each end host, and hence is unaffected by topology changes.

- While full-fledged desktop PCs could manage security mechanisms such as cryptography without much trouble, the same cannot be said for small, energy-limited mobile nodes. The heterogeneity property of MANETs dictates that computationally intensive tasks may not be acceptable for hosts with lower computational power. Policy outsourcing introduced in our algebra framework balances the utilization of power across the whole network, by outsourcing security policy to hosts with more capability.

- As individual nodes move around, they need to assess the authority of other nodes to act as firewalls for them. Thus, the framework plays a crucial role in security policy delegation. Intuitively, a node that is at risk of capture should not be considered as a good candidate for policy outsourcing. Less obviously, a rapidly moving nodes may be a poor choice as well, because it may drop out of the communication graph too soon.

4.2 Requirements of Policy Algebra Framework for MANET

Since the firewall configuration in a MANET faces special challenges, we discuss several requirements as an extension for our policy algebra framework to facilitate the coalition MANET environment.

Firstly, optimizing firewall policy to a changing topology requires secure knowledge of the topology, as well as communication and negotiation of security policy. Particularly, policy outsourcing in our framework requires that hosts be able to exchange security policies with neighboring nodes. Delegate selection for policy outsourcing also requires that the local host securely assess the authorization of its neighbors.

Secondly, light-weight algorithms for computing the cost and risk are strongly desired in policy algebra framework for MANETs. Small and energy-limited mobile nodes cannot afford computationally intensive operations, since the network lifetime is also a crucial metric for measuring MANET overall performance.

Finally, policy refinement and conflict resolution remain challenging tasks in MANETs. With rapid changes in MANET topology, it is very difficult for nodes to exchange and negotiate security policies with their neighbors. Periodic consistency checking performed globally for the entire network may not be feasible under these circumstances. One possible approach is to define the MANET as small-scale clusters, to facilitate local consistency maintenance.

5 Related Work

Previous work on firewall policy analysis mainly focus on studying the firewall configuration effects from the view point of network administrators. [MWZ00] designed a implemented a firewall analysis tool to allow the administrator to easily discover and test the global firewall policy. Their tool took the network topology and configuration files as input and interacted with the user through a question-and-answer session. The main advantage of this tool is to permit study of the firewall configuration before actually deploying it in the real network. [BKLR06] proposed an approach to firewall policy specification and analysis that uses a formal framework for argumentation based preference reasoning. By allowing administrators to define network abstractions, security requirements can be specified in a declarative manner using high-level terms.

Another approach related to our work is on the composition of access control policies. Access control policies have a very clear and restricted semantics over a collection of subjects, objects and action terms. [WJ03] presented a propositional policy algebra for access control. They model policies as nondeterministic relations over a collection of subjects, objects and action terms. The goal of their policy algebra is to manipulate access control policies at propositional level; the operations of the algebra are abstracted from their specification details. Moreover, it can be used to reason about role-based access control policies combined with other forms of discretionary policies. [BdVS02] took a different approach. They formulate access control policies as a set of ground terms over an alphabet for subject object action terms. Thus, they took a set-based approach influenced by logic programming.

Both of the above approaches have their limitation in that the proposed framework or analysis tool is only for one specific policy language, i.e., firewall rules or access control policies. To the best of our knowledge, we are the first to construct a policy algebra framework for security policy enforcement in hybrid firewalls. We do not make any assumption on the policy language; thus our framework is applicable for any general security policies. Moreover, we are the first to define cost and risk functions associated with each policy enforcement. Based on the administrator's goal, global minimization function can be applied to reduce the cost of policy enforcement while still maintaining specified degree of security. We show how policy algebra operations may introduce conflicts into the network. We use decorrelation algorithms to eliminate conflicts. We also demonstrate how our policy algebra framework can be further extended to support packet filtering firewall policies.

6 Conclusions

Firewalls are a effective means of protecting a local system or network of systems from network-based security threats. In this paper, we propose a policy algebra framework for security policy enforcement in hybrid firewalls. The framework describes external and internal algebra operations over rule sets and individual rules, respectively. Cost and risk functions are associated with security policy enforcement. As a concrete example, we extend the policy algebra framework to support packet filter firewalls. At the end of this paper, we also discuss the challenges and requirement for applying policy algebra framework in mobile ad hoc networks. Our future work will be to further extend this framework to support policy-based security management in MANETs.

Acknowledgments

Avishai Wool and Sotiris Ioannidis provided many helpful comments.

References

[BdVS02] Piero Bonatti, Sabrina De Capitani di Vimercati, and Pierangela Samarati. Using argumentation logic for firewall policy specification and analysis. *ACM Transactions on Information and System Security (TISSEC)*, 5(1):1–35, February 2002.

[Bel99] Steven M. Bellovin. Distributed firewalls. *;login:, pp. 37-39*, November 1999.

[BKLR06] Arosha Bandara, Tony Kakas, Emil Lupu, and Alessandra Russo. Using argumentation logic for firewall policy specification and analysis. In *Proceedings of 17th IFIP/IEEE Distributed Systems: Operations and Management*, 2006.

[BMNW04] Yair Bartal, Alain Mayer, Kobbi Nissim, and Avishai Wool. Firmato: A novel firewall management toolkit. *ACM Transactions on Computer Systems (TOCS)*, 22(4), November 2004.

[CBR03] William Cheswick, Steven Bellovin, and Aviel Rubin. *Firewalls and Internet Security*. Addison-Wesley, second edition, February 2003.

[HSP00] Adiseshu Hari, Subhash Suri, and Guru Parulkar. Security policy protocol. Work in progress, draft-ietf-ipsp-spp-00.txt, `http://www.ietf.org/proceedings/00jul/ID/ipsp-spp-00.txt`, July 2000.

[MWZ00] Alain Mayer, Avishai Wool, and Elisha Ziskind. Fang: A firewall analysis engine. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2000.

[SC00] Luis A. Sanchez and Mathew N. Condell. Detecting and resolving packet filter conflicts. In *Proceedings of INFOCOM*, 2000.

[WJ03] Duminda Wijesekera and Sushil Jajodia. A propositional policy algebra for access control. *ACM Transactions on Information and System Security (TISSEC)*, 6(2):286–325, May 2003.

[Woo04] Avishai Wool. A quantitative study of firewall configuration errors. *IEEE Computer*, 37(6):62–67, 2004.