Filtering Postures: Local Enforcement for Global Policies *

Joshua Guttman
The MITRE Corporation
202 Burlington Road, MS A155
Bedford, MA 01730-1420
+1 617 271 2654
guttman@mitre.org

May 19, 1997

*Work supported by the National Security Agency under United States Army CECOM contract DAAB 07-96-C-E601.

Abstract

When packet filtering is used as a security mechanism, different routers may need to cooperate to enforce the desired security policy. It is difficult to ensure that they will do so correctly.

We introduce a simple language for expressing global network access control policies of a kind that filtering routers are capable of enforcing. We then introduce an algorithm that, given the network topology, will compute a set of filters for the individual routers; these filters are guaranteed to enforce the policy correctly. Since these filters may not provide optimal service, a human must sometimes alter them. A second algorithm compares a resulting set of filters to the global network access control policy to determine all policy violations, or to report that none exist.

A prototype implementation demonstrates that the algorithms are efficient enough to give quick answers to questions of realistic scale.

A short but updated version of this material appears as [9].

Acknowledgments

I am deeply indebted to my colleagues Bill Farmer, Patty Heinle, Dale Johnson, Len LaPadula, Jonathan Millen, Leonard Monk, Javier Thayer Fábrega, and Dan Vukelich.

Contents

1 Introduction **1**

2 A Motivating Example **3**
2.1 A Corporate Access Control Problem 3

3 Networks, Policies, and Postures **7**
3.1 Network Specifications 8
3.2 Network Access Control Policies 10
3.3 Filtering Postures: Localizing Policies 11

4 A Specification Language **13**
4.1 Networks and Services 13
4.2 Sets of Hosts and Sets of Services 14
4.3 Policy for the Corporate Example 16

5 Reasoning about Policies and Postures **19**
5.1 Checking a Posture 19
5.1.1 Posture Checking: Corporate Example 20
5.1.2 Posture Checking: Spoofing and Trusted Areas 21
5.2 Generating a Posture 22
5.2.1 Outbound Filtering 23
5.2.2 Inbound Filtering 24
5.2.3 Filter Generation: Corporate Example 24
5.2.4 Filter Generation: Tactical Example 25
5.3 Rectangles and Rules 32

6 Conclusion and Future Work **39**
6.1 Service Specifications 39
6.2 Router Security Testing 40

A Grammar for the Specification Language **43**

B Full Specification for Corporate Example **45**

List of References **49**

Distribution List **51**

List of Figures

1 Example Corporate Network 3
2 Tactical Communications Example 26
3 Notation for Proof of Lemma 5 34

List of Tables

1 Packet Constraints for External-to-Corporate Traffic 5
2 Packet Constraint for Periphery-to-Internal Traffic 5
3 Specifying the Corporate Network 13
4 Host Sets for the Corporate Protection Example 15
5 Auxiliary Rules for the Corporate Policy 16
6 Rules for the Periphery and External Areas 17
7 Rules for the Allied and Engineering Areas 17
8 Expanded Constraints for the Allied and Engineering Areas . . . 18
9 Sample Filtering Rules: Flowing into the Periphery 18
10 Error Report: Spoofing an Allied Source 20
11 Generated Inbound Filtering: The Allied/External Router 25
12 Generated Outbound Filtering: The Allied/Engineering Router . 26
13 Specifying the Tactical Example 27
14 Definitions for the Tactical Example 28
15 Policy for the Tactical Example 28
16 Filter Generated for Battalion/Radio2 Traffic 29
17 Error Report if Radio2 Filter Omitted 29
18 Filter Generated for HQ/Radio1 Traffic 30
19 Error Report if Radio1 Filter Omitted 30
20 Inbound Filter Generated for Traffic from Radio2 31
21 Corrected Inbound Filter for Traffic from Radio2 31
22 Timings for Filter Generation 36
23 Grammar for the Specification Language 44

1 Introduction

One network security problem—out of many—is a problem of access control: namely to ensure that if a packet such as an IP datagram travels from one portion of a network to another, then it has some legitimate business there. For instance, if the packet comes from an area that is considered untrustworthy and reaches another area that is considered in need of protection, then the packet should provide a desired service, and a service that will not damage the recipient. Conversely, if a packet comes from an area in need of protection and travels to an untrustworthy area, then we may want to ensure that the packet is providing a service that will not leak information we wish to keep confidential.

Different mechanisms may be used to solve this access control problem, possibly in combination with each other, but filtering routers are likely to play a major role if it is implemented at the network layer. In this paper we introduce a framework for stating these network access control policies and for implementing them reliably via filtering routers.

The crucial issue we will consider arises because several routers are often involved. When several different networks are involved, or when the security policy imposes different constraints as a packet traverses a succession of areas, then several routers will have to cooperate to enforce the policy. A network administrator must configure these routers—or perhaps negotiate with the network administrator at another organization in some cases—so that their composite effect is to enforce the desired access controls. It is difficult to determine by hand what division of labor will allow the routers to ensure that the constraints will be enforced, no matter what path through the network a packet might take.

This is a problem of localization. It may be relatively straightforward to define the policy for the network, but this is a global policy about what packets can get where, regardless of path. By contrast, the filtering decisions of the individual routers are based on local information: namely, what interface the packet arrived at; what interface the packet will be routed out through; and what the headers say. In this paper we describe a method that can be used to solve the problem of localization.

There are two separate ingredients in the solution. One is conceptual or logical; it amounts to deciding what filtering choices will be made at each router interface, what functional behavior each filter must exhibit. Finding a correct assignment of behaviors to the router interfaces at different locations in the network may be tricky, because one router may have to filter packets to ensure that they will not violate an access control stipulation when they eventually reach a remote portion of the network.

The second ingredient is a matter of implementation, namely encoding a filter configuration file that will correctly enforce those choices on the equipment actually available. Given the filter configuration languages of commercial routers, this is by no means trivial. However, only the first ingredient will be the subject of this paper. Other work ongoing at MITRE and elsewhere may be helpful for the second.

Because we will consider only a logical description of the filtering to be done at each router interface, we will coin a new phrase. We will speak of a "filtering posture" to mean an assignment of filter functional behavior to each router interface in a network. The filtering posture does not specify the router configuration files that will implement this functional behavior; it stipulates only the logical effects that those configuration files should achieve.

We will proceed as follows. In Section 2 we will introduce an example of the network access control localization problem. In Section 3, we will analyze what notion of security policy is appropriate here, and we introduce some basic definitions concerning security policies of this kind. Section 4 introduces a simple language that can be used to represent networks and policies. We also encode our motivating example in the language. Section 5 turns to policy enforcement; it presents two algorithms. One of these (given in Section 5.2) generates one filtering posture—which is to say, one correct solution to the localization problem for any given policy—making use of the network topology. Since this solution may not allow useful services that other secure solutions would permit, a human user may explore variants. A second algorithm (presented first, because it is simpler, in Section 5.1) takes a proposed filtering posture and determines whether it in fact enforces the global security policy faithfully. The original motivating example is examined again here, to make clear what practical insights our methods provide. In addition, a second example (Section 5.2.4) is introduced to illustrate the strengths and weaknesses of some variants. A conclusion (Section 6) summarizes and mentions some future work.

2 A Motivating Example

In this section we describe a realistic example in which a network access control policy must be enforced by a distributed collection of routers. It concerns a corporation with an allied organization requiring special network access. The problem is reasonably hard: manual attempts to localize the filtering decisions were in fact wrong. However, the security goals are reasonably easy to state.

Although in this example—and throughout the paper—we will use the vocabulary of TCP/IP, the ideas and methods we introduce are also applicable to other protocols.

2.1 A Corporate Access Control Problem

Let us suppose that a certain corporation has a network containing a peripheral subnet—a screened subnet on which its firewall is implemented—together with two groups of subnets, one serving its financial departments and the other serving its engineering departments. The corporation cooperates closely with an allied organization, which needs special access to the corporation's engineering networks.

The situation is displayed in Figure 1. We will say that a host (or single physical net) is *internal* if it lies either in the engineering area or in the financial area; it is *corporate* if either it is internal or else it lies in the periphery area.

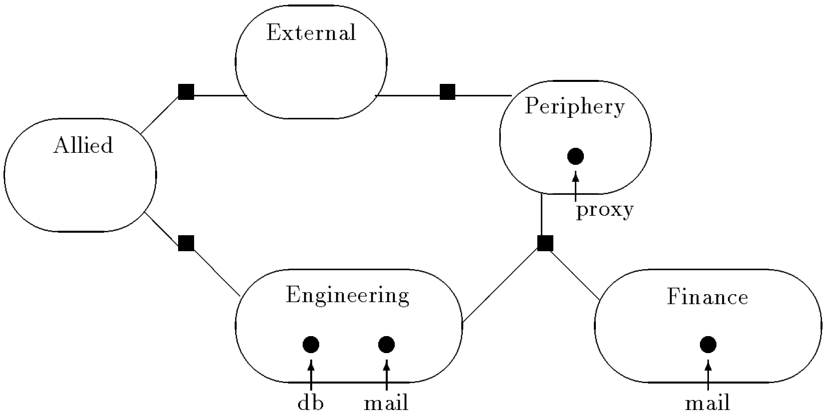

Figure 1: Example Corporate Network

The users of the network rely on it for services implemented by application protocols such as SMTP for electronic mail, FTP, HTTP, and TELNET, and also database queries to the database server shown in the engineering area. These database queries are made via remote procedure calls using UDP packets to port 2342 (let us say) on that server. We shall suppose that FTP, HTTP, and TELNET to the external area are proxied using an application level firewall on the proxy host shown in the periphery area. Thus, connections for these protocols involve either an internal host and the proxy host or else alternatively the proxy host and a host in the external area. Electronic mail (using SMTP) is not proxied; however, connections are permitted only with the mail servers in the engineering and financial areas. Database queries from the external area are not permitted.

Hosts in the allied area are permitted unproxied FTP, HTTP, and TELNET to the engineering area (but not to the financial area). In addition, they are permitted to submit database queries. We shall want to be sure that these services travel directly between the allied area and the internal networks; if they were to travel through the external area, they could be spoofed or hijacked. Connections between the allied area and the financial area are limited to electronic mail. We shall want to be sure that all packets entering the internal networks from the allied networks really originate in the allied area, as they have bypassed the controls implemented in the periphery network and its proxy host.

There may also be constraints on connections between engineering and finance. For instance, FTP, HTTP, and TELNET may be permitted only if the server is in the engineering area rather than the finance area. Possibly the engineers will attempt to discover their supervisors' salaries, or to increase their own.

How can we formalize these security goals? There appear to be three main ingredients:

1. Has the packet ever been in a particular area, for instance, the external area, in the past?

2. Has the packet now reached a particular area, for instance the engineering area?

3. What does the packet say? This in turn involves primarily three ingredients, although others (e.g., syn and ack bits) may be relevant at an implementation level:

   - The IP source field of the packet;
   - The IP destination field of the packet;
   - The service that the packet supports. This is generally disclosed by either the source port or the destination port, contained in the TCP or UDP segment in the packet.

There is a contrast between items 1 and 2—where the packet was and where it has now reached—and the information comprising item 3. The first two ingredients concern the actual path of the packet as it traverses the network, regardless of what it claims. The last concerns only what it claims, not where it has really passed. These two kinds of information can diverge in surprising ways: when routers send packets through unexpected paths, or when packet origin is spoofed, or when packets are intercepted before reaching their nominal destinations.

Our approach is to combine these ingredients so that a network access control policy permits packets to travel from one area to another only if what the packet header says indicates that it has legitimate business doing so. The relevant

Category	Source	Destination	Service
Proxied (to proxy)	external	proxy host	`ftp`, `http`, `telnet` (from server)
Email (incoming)	external	mail servers	`smtp` (to or from server)
Email (allied)	allied	mail servers	`smtp` (to or from server)

Table 1: Packet Constraints for External-to-Corporate Traffic

Category	Source	Destination	Service
Proxied (from proxy)	proxy host	internal	`ftp`, `http`, `telnet` (from server)

Table 2: Packet Constraint for Periphery-to-Internal Traffic

properties of the packet header, for a particular application, may be easily summarized; Table 1 gives those relevant for packets that might pass from the external area to a corporate network. We may now stipulate the policy for packets entering the corporation:

- If a packet p traveled from the external area to the periphery area, then one of the constraints in Table 1 holds of p.
- If a packet p traveled from the external area to the engineering or financial area, then one of the Email constraints in Table 1 holds of p.

We intend this as an exhaustive list of admissions criteria. No other packets should be permitted to enter any corporate area, if they have ever previously been in the external area.

For packets that might pass from the periphery area to the internal areas, another constraint is needed (Table 2). The policy for packets delivered from the periphery area is now:

- If a packet p traveled from the periphery area to the engineering or financial area, then either one of the Email constraints in Table 1 holds of p, or else the Proxied constraint from Table 2 holds of p.

All of the security goals we have described can be codified in this way. However, it will become tedious unless there is some language to build up related descriptions from common parts. For instance, the constraint for a packet to travel from the engineering or financial area to the periphery is so to speak the reverse of the constraint we have just given: we would interchange the roles of the source and destination, and we would interchange the orientation toward the server, so that "to server" becomes "from server" and *vice versa*. We will call this later, in the language we introduce in Section 4, "reflecting" a constraint.

3 Networks, Policies, and Postures

In this section we will develop a rigorous model for network access control, including the topology of the network, the security policies desired, and the filtering postures we want to make sure will enforce them.

The areas shown in Figure 1 may be quite large, with many hosts and single physical networks within them. The IP addresses for the areas may or may not be easy to express with a few network masks. Nevertheless, from the point of view of our network policy, the areas provide the natural building blocks, together with the distinguished hosts—for instance in Figure 1, the mail hosts, proxy host, and database server—within them. The security policy is only sensitive to whether a host is within an area, and whether it is one of the distinguished hosts.

Only at the boundaries between areas can we enforce our policy. The areas are separated by filtering routers, and the routers effectively define them as meaningful security units.

In our security model we will regard the areas as fundamental units. The purpose of a security policy will be to control what packets can travel between different pairs of areas. The filtering routers are the only enforcement mechanism we will consider. Thus, in formulating a real-world problem in this framework, if there is some packet trajectory that we would like to prevent, then we must identify the areas in such a way that the trajectory of the packet traverses more than one area.

While we may leave many routers out of a model, each of them must route packets within a single area, not between areas. If a router moves packets from one area to another, it will provide a path that may falsify our security policy. This means that they must be irrelevant to our concerns. Indeed, they may not even do any filtering.

Each area will abstract a contiguous set of physical networks, together with all the machines on them, including the routers moving packets among these networks. To be quite precise, the area will contain a set of network interfaces; if a machine (host or router) has more than one network interface, those interfaces may belong to different areas. When a host or router[1] does have interfaces on more than one area, then we will have to represent it in the model, so that we can reason about what filtering it provides. The model must identify all the devices that may move packets from one network area to another, so as to ensure that no packet can move in a way that would violate the access controls we want to impose.

We also assume that different areas contain disjoint sets of interfaces, and that no single physical network belongs to more than one area. Thus, packets can pass from one area to another only by passing between the interfaces of a

[1] We will no longer distinguish between a host that has interfaces on more than one area and a router. We will assume that any host with interfaces on more than one area is capable of filtering packets. On the other hand, some devices playing this role do not actually do IP routing. Thus, the terms "router" and "filtering router" are not ideal. Nevertheless, having noted this caveat, we will retain them.

router.

For simplicity, we will suppose here that any two areas have at most one router connecting them, and that the router has at most one interface on any area. However, this assumption is merely expository.

3.1 Network Specifications

With these assumptions, we may regard a network as a bipartite graph, where each node is either an area or a router. Each edge links a router with an area on which it has an interface. A packet trajectory is simply a path through this graph. Naturally, realistic routing tables will not send packets along all possible paths, but we will not formalize the routing constraints. Instead, we will require that a filtering posture be safe, even if the routing tables should evolve to some unexpected state. We will not assume that the routing configuration will reliably assist the filtering posture.

We will present our ideas in mathematical form using definitions and lemmas. This form is clear and compact, and it should ease the reader's burden in following cross references. We will not prove any theorems of any depth.

Definition 1 (Network specification)
A network specification $\mathcal{N}$ *is a quintuple* $\langle A, R, D, C, L\rangle$ *such that:*

- A*,* R*, and* D *are disjoint finite sets called the* areas *(or areas),* routers*, and* distinguished hosts *of* $\mathcal{N}$*;*
- $C \subseteq A \times R$ *is the* incidence relation *specifying when an area contains an interface on a router;*
- $L : D \rightarrow A$ *is the* location *function which maps each distinguished host to the area to which it belongs.*

The set $A \cup R$ *is called the set of* nodes *of* $\mathcal{N}$*. The set* $A \cup D$ *is called the set of* abstract addresses *of* $\mathcal{N}$*.* $\tilde{L}$ *is the function from abstract addresses of* $\mathcal{N}$ *to* A *such that* $\tilde{L}(a) = a$ *for* $a \in A$ *and* $\tilde{L}(d) = L(d)$ *for* $d \in D$*.*

An abstract address represents a collection of possible sources and destinations for packets. If an abstract address is a distinguished host, it will be used to represent packets addressed to (or from) that host. The same mechanism we use for distinguished hosts could also be used for distinguished subareas: sets of interfaces or addresses all belonging to one area, and playing some special role as sources or destinations in a security policy.

If an abstract address is an area $a \in A$, then it will be used to represent packets addressed to (or from) all hosts in a that are not distinguished hosts. By this we are not referring to a broadcast mechanism. Rather, at our level of abstraction, packets addressed to different (non-distinguished) hosts within the same area are effectively the same. For the purposes of expressing policy or reasoning about how to localize policy, we may treat them identically.

We will regard a sequence σ of length n to be a finite partial function defined on the natural numbers between 0 and $n-1$ inclusive, and we will write $f(i)\downarrow$ to mean that the function f is defined at the argument i. Hence, we can define a path as follows:

Definition 2 (Path)
A path *through a network specification $\mathcal{N}$ is a finite sequence σ of nodes of $\mathcal{N}$ such that, for all $i \geq 0$, $\sigma(i+1)\downarrow$ implies*

$$\langle \sigma(i), \sigma(i+1)\rangle \in C \;\vee\; \langle \sigma(i+1), \sigma(i)\rangle \in C$$

A path is cycle free *if, for all i, j, $\sigma(i) = \sigma(j)$ implies $i = j$.*

We regard a path as indicating part of the travels of a packet; the packet may not have actually originated at $\sigma(0)$. We take this point of view because one cannot always know whether a packet has originated within an area; the mere fact that a packet was at some time in an area is easier to determine and sufficiently informative to support reasonable access control policies. If a packet is altered during its travels, we will regard the result as a new packet, traveling on a path beginning at the point of alteration.

Definition 3 (Services, Packet description, Trajectory)
Let S be some finite set of services*, and let $\mathcal{N}$ be a network specification. Then a* packet description *for S and $\mathcal{N}$ is a quadruple $\langle src, dst, s, o\rangle$, where*

- *src and dst are abstract addresses of $\mathcal{N}$ called the* source *and* destination *of the packet;*
- *$s \in S$ is the* service *of the packet; and*
- *o, the* orientation *of the packet, is one of the values* `to-server` *and* `from-server`*.*

We will sometimes refer to a service and an orientation as an oriented service*, and to the set of all packet descriptions as* All-p*.*

A trajectory *is a pair $\langle p, \sigma\rangle$ of a packet description and a path through $\mathcal{N}$.*

A packet description is thus an abstraction that covers a large collection of possible real packets. For instance, if it has an area a as its source field, it covers packets with any IP address in that area (other than the address of a distinguished host) as their source field. If the area, like `external` in our corporate example, covers a majority of the Internet, this is a very large set. However, they will all need to be treated uniformly in the policy and its localization anyway, so there is a very large benefit to abstracting them all to a single packet description.

The orientation signifies whether a service's port number occurs in the packet as the destination port number or the source port number. For some purposes it might be desirable to add other attributes to a packet description apart from its orientation o; for instance, in some cases a classification might be wanted.

The services S are generally pairs consisting of a protocol—either `tcp` or `udp`—and a port number. However, this choice mainly reflects the way that typical filtering routers work today; if filtering routers had a more flexible mechanism (for instance, to allow packets to and from the ports selected for a passive FTP connection), the same ideas and algorithms described below could be used essentially unchanged.

We will frequently assume below without special notice that some arbitrary $\mathcal{N}$ and S have been chosen.

3.2 Network Access Control Policies

As suggested in Section 2, we will regard a policy as constraining the header information on a packet, depending on what areas the packet has traversed. A policy statement is a requirement on the trajectory taken by a packet p, with the meaning:

> If p was previously in a_1 and later reaches a_2, then $\phi(p)$,

where ϕ is some set of packet descriptions. A policy will associate a policy statement of this form with each pair of distinct areas.

Not all conceivable (intuitive) policies are of this form. Some policies might concern three or even more areas. For instance, the policy, "If p was previously in a_1, then reaches a_2, and eventually enters a_3, then p should be a database query," may be desirable, because only an important service like a database query should be permitted to use the scarce bandwidth of a_2 when traveling from a_1 to a_3. Presumably, the most general notion of security policy that makes sense in this framework is simply a set of trajectories.

However, the security goals of such three-area policies can frequently be restated in two-area form in an acceptable way. In the instance given, it may suffice to ensure that any packet that was in a_1 and later reaches a_2 has the property that if its destination address is in a_3, then it is a database query.

In addition, the methods we introduce below for reasoning about security policies, and localizing them via filtering postures, would be a great deal more complex for three-area policies, not to mention arbitrary sets of trajectories.

Hence, throughout this paper, we will limit our attention to security policies expressed in two-area form.

Definition 4 (Policy Constraints and Statements, Policies, Satisfaction)
A policy constraint *is a set* ϕ *of packet descriptions* $\phi \subseteq$ *All-p. The set* $\Phi = \mathcal{P}(\textit{All-p})$ *contains all policy constraints* ϕ.

A policy statement *is a triple* $\langle a_1, a_2, \phi \rangle$*, where* $a_1 \neq a_2$.

A network access control policy *is a set of policy statements such that for every pair of distinct areas* a_1*,* a_2*, there is exactly one policy statement of the form* $\langle a_1, a_2, \phi \rangle$.

A trajectory $\langle p, \sigma \rangle$ satisfies *a policy statement* $\langle a_1, a_2, \phi \rangle$ *iff:*

$$[\sigma(0) = a_1 \wedge (\exists j \,.\, j > 0 \wedge \sigma(j) = a_2)] \;\Rightarrow\; p \in \phi$$

Since $\sigma(0)$ need not be the origin of the packet, there is no essential loss of applicability in writing $\sigma(0)$ as opposed to $\sigma(i)$ for $i < j$. Dropping the initial i entries from σ gives a σ' that we will also want to satisfy the policy.

We will sometimes regard a policy P as a function, where $P(a_1, a_2) = \phi$ iff $\langle a_1, a_2, \phi \rangle \in P$. As before, $\sigma(0)$ may not be the original source of the packet; it is merely the start of the portion of its journey that concerns us.

Some simple examples of policy constraints are:

- The empty set, signifying that no packets are permitted;
- The set s_1 of all packet descriptions $\langle src, dst, s, o\rangle$ where $s = \langle \mathtt{tcp}, 25\rangle$; i.e., the set of `smtp` packets;
- The set s_2 of all packet descriptions $\langle src, dst, s, o\rangle$ where $dst = d$ for some distinguished host $d \in D$ serving as a bastion host;
- The set $s_1 \cap s_2$, signifying that `smtp` to d is permitted.

Naturally, policies of realistic interest are more complex, and require a reasonable notation to convey effectively; we develop such a notation in Section 4.

3.3 Filtering Postures: Localizing Policies

Our goal is to implement networks that can faithfully enforce policies of the kind we have just introduced, by means of assigning filters to router interfaces. We will represent a filter by a constraint ϕ; it represents the filter that will pass a packet p just in case $p \in \phi$.

We will in fact associate *two* filters with each router interface. One examines packets as they come inbound over the interface into the router; the other examines packets as they go outbound over the interface out of the router. This corresponds reasonably well with the filtering facilities of commercially available routers or other filtering devices, for instance, Cisco routers [6]. Some other routers, such as Network Systems Corporation routers [11], provide somewhat more flexibility than this, while some (for instance, older Cisco routers) provide somewhat less.

A filtering posture is an assignment of inbound and outbound filtering constraints to each interface.

Definition 5 (Filtering posture, inbound and outbound)
A filter assignment is a partial function $f : A \times R \rightharpoonup \Phi$ such that $f(a, r)\downarrow$ iff $\langle a, r\rangle \in C$.

A filtering posture *is a pair of filter assignments $\langle$ **inb**, **outb** $\rangle$.*

Suppose that σ is a path. If $\langle \sigma(i), \sigma(i+1)\rangle \in C$, then the ith step in σ is inbound*; if $\langle \sigma(i+1), \sigma(i)\rangle \in C$, then the ith step in σ is* outbound.

For convenience, we will write σ^i for the value inbound *or* outbound *in these two cases (respectively).*

A trajectory is feasible if the packet can survive all of the filters it traverses.

Definition 6 (Feasible trajectories)
A trajectory $\langle p, \sigma \rangle$ is feasible relative to a filtering posture $\langle \mathbf{inb}, \mathbf{outb} \rangle$ if, for all i such that $\sigma(i+1)\downarrow$,

- *If σ^i is inbound, then $p \in \mathbf{inb}(\sigma(i), \sigma(i+1))$, and*
- *If σ^i is outbound, then $p \in \mathbf{outb}(\sigma(i+1), \sigma(i))$.*

A filtering posture enforces a policy if all the feasible trajectories satisfy the policy.

Definition 7 (Policy enforcement)
A filtering posture $F = \langle \mathbf{inb}, \mathbf{outb} \rangle$ enforces *a policy P iff, for all trajectories $\langle p, \sigma \rangle$, if $\langle p, \sigma \rangle$ is feasible relative to F, then $\langle p, \sigma \rangle$ satisfies every policy statement in P.*

With these definitions in hand, we now have a precise way to describe exactly what security goals a collection of filtering routers is intended to achieve.

4 A Specification Language

We will now describe a notation in which network specifications, services, and policies can be presented. Our notation has a Lisp-like syntax, because that is particularly simple for programs—especially Lisp programs—to manipulate.[2]

4.1 Networks and Services

A network specification uses two forms, one an `areas` expression, which gives the names of the areas and of the distinguished hosts located within each area, while the other, a `connectivity` expression, gives the names of the routers, together with the areas on which each router has interfaces. The specification in effect introduces a bipartite graph, in which the nodes consist of the areas and the routers. There is an edge between two nodes if one is an area and the other is a router, and the router has an interface on that area.

In the case of our corporate example (Figure 1), we have the forms presented in Table 3. Comments are introduced by semicolons, and stretch to the end of the line. In this notation, services are (currently) introduced by protocol and

```
(areas
 ;; name               distinguished hosts
 (external)
 (periphery           proxy-host)
 (engineering         eng-mail-server db-server)
 (financial           financial-mail-server)
 (allied))

(connectivity
 ;; router name        areas
 (per/ext-router      periphery external)
 (per/eng/fin-router  periphery engineering financial)
 (eng/fin-router      engineering financial)
 (eng/allied-router   engineering allied)
 (allied/ext-router   allied external))

(services
 (telnet              tcp 23)
 (ftp                 tcp 21)
 (ftp_data            tcp 20)
 ...)
```

Table 3: Specifying the Corporate Network

server port number, although as we mentioned in Section 3.1, other approaches to defining services could be easily accomodated.

[2] If the program is equipped with a graphical user interface, syntactic prettiness is irrelevant; the user need never look at the actual input format.

4.2 Sets of Hosts and Sets of Services

In this subsection we will introduce the linguistic support we need to describe packets and to express policy constraints of the kinds illustrated in Tables 1 and 2. In order to do so in a form that we will be able to process efficiently, we want simple ways to express policy constraints that allow large collections of packets.

Our choice is to use *rectangles* of packets. A rectangle is determined by two sets of hosts, representing respectively the possible source addresses and the possible destination addresses, and a "coloring" for the rectangle, representing the set of oriented services permitted for packets with sources and destinations in the rectangle. A collection of rectangles will represent a *rule*. To avoid issues about blending colors, we will always maintain rules in a form in which all of their rectangles are disjoint. We will introduce rectangles more rigorously in Section 5.3; in the current section we will introduce notation for expressing rectangles and rules, and the sets from which rectangles are built.

Any host set that makes sense for a given network specification $\mathcal{N}$ can be represented in a very uniform way. It consists of:

- All the hosts located within zero or more areas,
- Omitting zero or more distinguished hosts, and
- Including zero or more distinguished hosts.

In giving such a description we may always arrange that the distinguished hosts to be omitted belong to the areas selected, as otherwise it is redundant to omit them. Likewise, we may arrange that the distinguished hosts to be specifically included belong to areas that have not been selected, as otherwise it is redundant to include them. Finally, we may suppose that the distinguished hosts to be omitted are disjoint from those to be included. We will call a description in this form a "canonical" host set.

We present a host set in the form:

```
((areas   aname1     ...  namen)
 (without dhname1    ...  dhnamem)
 (with    dhnamem+1  ...  dhnamek))
```

where any of the keywords `areas`, `without`, and `with` may be omitted if it introduces no names.

Host sets may be combined using the boolean operations `union`, `intersection`, `difference`, and `complement`. From the point of view of implementation, these operations may be made very efficient: given a network specification $\mathcal{N}$, the set of possible areas is known, so a subset of the areas may be represented as a bit-vector. Thus, applying these operations to the areas may require only one machine instruction (as 32 areas would be unusually many). In addition, normally only a few distinguished hosts are to be omitted or included, so that these lists may be combined very quickly.

```
(defined-host-sets                     ; define some host sets
  (internal                            ; new name
   ((areas engineering                 ; two areas
           financial)))
  (corporate
   ((areas periphery                   ; three areas
           engineering financial)))
  (non-corporate (complement corporate))
  (proxy-host ((with proxy-host)))
  (mail-hosts
   ((with eng-mail-server              ; two disting. hosts
          financial-mail-server)))
  (trusted-hosts
   (union                              ; union of
    ((areas periphery))                ; area and
    (union mail-hosts                  ; host set
            ((with db-server))))))
```

Table 4: Host Sets for the Corporate Protection Example

Our notation includes a `defined-host-sets` declaration that introduces an identifier abbreviating a host set. The host set declarations for the corporate protection problem of Figure 1 are presented as an example in Table 4.

Oriented services, by contrast with host sets, are specified much more simply. Each oriented service is presented in the form `(service-name orientation)`, where the orientation is one of the two symbols `from-server` and `to-server`. A set of oriented services is presented by the symbol `all` or by a parenthesized list of oriented services. If additional attributes of services were to be added, for instance to distinguish classified and unclassified packets, it might become necessary to have better support for sets of services.

A rectangle may be specified by giving:

- A host set describing the source addresses it concerns;
- A host set describing the destination addresses it concerns;
- A list of the oriented services permitted.

A rule, which expresses a constraint on packets, is then represented as a sequence of rectangles, which in our approach are always maintained in disjoint form. Several operators make sense for rules, including `disjoin` and `conjoin`, `complement`, and `reflect`. The `reflect` operator interchanges the source host set and the destination host set of each rectangle in a rule, and it reverses the orientation of each oriented service. If a rule describes one direction of each of several kinds of conversation, then its `reflect`ion represents the other direction of the same conversations. More details about our representation of rectangles and rules, and of the operators on them, are presented in Section 5.3.

```
(defined-rules
   ;; rule name
   (source-non-corporate
     ;; source host set                dests  services
     ((non-corporate                     all  all)))
   (source-allied
     ((((areas allied))                  all  all)))
   (destination-internal
     ((all                          internal  all)))
   (destination-proxy
     ((all                        proxy-host  all)))
   (destination-mail-hosts
     ((all                        mail-hosts  all)))
   (destination-eng
     ((all                     eng-untrusted  all)))
   (db-query
     ((all               ((with db-server))  ((db-query
                                                to-server)))))
   (usual-service-requests
     ((all                               all  ((telnet to-server)
                                               (ftp to-server)
                                               (ftp_data to-server)
                                               (http to-server)))))
   (email-service
     ((all                               all  ((smtp from-server)
                                              (smtp to-server))))))
```

Table 5: Auxiliary Rules for the Corporate Policy

4.3 Policy for the Corporate Example

To illustrate the workings of the specification language, we will now present a formalization of the security policy for the corporate example of Section 2.1. We start in Table 5 by introducing some notation, using a `defined-rules` form to give names to useful rules. We want to use this notation to specify that any packet that has ever been in the `external` area and arrives in the `periphery` area must be addressed either to the proxy host (located in the `periphery` area) or else to one of the mail hosts (located in the `engineering` and `financial` areas respectively). In addition, the source address should be compatible with the packet's having arrived from outside: it should lie either in the `external` area or in the `allied` area.

Conversely, if a packet was ever in the `periphery` area, and later reaches the `external` area, then a corresponding or reflected condition should hold. The source address should be either the proxy host or else one of the mail hosts, while the destination should be outside the corporation. The operator `reflect` expresses this transformation, in which source and destination are interchanged, and the orientation of each service is reversed. The conditions for the `external`

```
(defined-rule-sets
  (external-to-periphery
   (conjoin
    source-non-corporate
    (disjoin destination-proxy destination-mail-hosts))))

(policy
 ;;previously in      reaching     rule
 (external            periphery    external-to-periphery)
 (periphery           external     (reflect external-to-periphery)))
```

Table 6: Rules for the Periphery and External Areas

```
(defined-rule-sets
  (allied-incoming-addresses
   (conjoin source-allied destination-internal))

  (allied-to-engineering
   (conjoin
    allied-incoming-addresses
    (disjoin
     db-query
     (disjoin
      (conjoin destination-mail-hosts email-service)
      (conjoin destination-eng usual-service-requests))))))

(policy
 ;;previously in      reaching     rule
 (allied              engineering  allied-to-engineering)
 (engineering         allied       (reflect allied-to-engineering)))
```

Table 7: Rules for the Allied and Engineering Areas

and `periphery` areas, in each direction, are formalized in Table 6.

As one more illustration, we characterize the policy for packets that were ever in the allied area and reach the engineering area. In this case, we want to ensure that the source address of the packet is in `allied` and its destination is `internal`, i.e. either in `engineering` or in `financial`, as we do not want the corporate network to be used as a pass-through. In addition, we will require the packets either to be database queries (addressed to the database server), or else email addressed to the mail hosts, or else the usual service requests addressed to other engineering hosts. For packets that were in the engineering area and reach the allied area, we want the corresponding reflected condition. These constraints are formalized in Table 7. The remainder of the specification is equally straightforward. The full specification is contained in Appendix B.

When this specification is parsed by our prototype implementation, it re-

```
(((areas allied))                                          ; srcs
 ((with db-server))                                        ; dsts
 ((db-query to-server)))

(((areas allied))                                          ; srcs
 mail-hosts                                                ; dsts
 ((smtp from-server) (smtp to-server)))

(((areas allied))                                          ; srcs
 ((areas engineering)                                      ; dsts
  (without eng-mail-server db-server))
 ((http to-server) (ftp_data to-server)
  (ftp to-server) (telnet to-server)))
```

Table 8: Expanded Constraints for the Allied and Engineering Areas

solves all of the definitions to their ultimate meanings. Thus, by inspecting the policy clauses that the tool generates, one can check that each rule has the intended meaning. For instance, if a packet was in `allied` and has reached `engineering`, then the packet must satisfy one of the three constraints given in Table 8. These constraints stipulate that the packet must either be a database query directed to the server, an email packet directed to a mail host, or a request for one of the protocols HTTP, FTP, FTP_DATA, or TELNET, directed to one of the ordinary (undistinguished) engineering hosts. In all cases, the source address must be in `allied`.

The input specification language also permits a user to specify filtering rules for particular router interfaces. Each router has an inbound filter and an outbound filter for each area on which it has an interface, and the behavior of each filter is specified by a rule. A filter specification gives the name of the router, the direction, the area in which the interface lies, and a rule specification. Two of the filters for the router between `external` and `periphery` are shown in Table 9. In practice, it is not necessary to write router interface filtering specifications directly; the prototype tool generates a collection, and a few small edits suffice to tailor their functionality.

```
(interface-filtering-specs
 (per/ext-router inbound       ; from
  external
  ((((areas external)) all all)))

 (per/ext-router outbound      ; to
  periphery
  (disjoin destination-proxy destination-mail-hosts)))
```

Table 9: Sample Filtering Rules: Flowing into the Periphery

5 Reasoning about Policies and Postures

The ideas introduced in Section 4 suggest a number of algorithms, which exploit the boolean operations on constraints in combination with the graph structure of the underlying network specification $\mathcal{N}$. These algorithms may be used to check a putative filtering posture (Section 5.1), or to generate a filtering posture that will enforce a given policy (Section 5.2). Examples are explored in Sections 5.2.3–5.2.4. In Section 5.3, we develop the ideas of rectangle and rule more formally, and introduce the operations on them that were the crucial ingredients in developing an efficient implementation for the algorithms contained in this section.

All of these algorithms depend on the notion of the feasibility set of a path.

Definition 8 (Feasibility set)
Let σ be a path, and let F be a filtering posture. The feasibility set of σ relative to F is the set of all packet descriptions p such that the trajectory $\langle p, \sigma\rangle$ is feasible relative to F. We write the feasibility set of σ as $\mathcal{F}_F(\sigma)$, or as $\mathcal{F}(\sigma)$ when the filtering posture F is understood.

We represent feasibility sets by rules. Naturally, we can compute the feasibility set of a path σ by iteration on the length of σ.

Lemma 1 *Let ℓ be the length of σ. If $\ell \leq 1$, then $\mathcal{F}(\sigma) =$ All-p. If $\ell > 1$ and $\sigma = \sigma_0{}^\frown[a, \rho]$, so that $\sigma^{\ell-2}$ is the inbound step from area $a \in A$ to router $\rho \in R$, then*

$$\mathcal{F}(\sigma) = \mathcal{F}(\sigma_0{}^\frown[a]) \cap \boldsymbol{inb}(\langle a, \rho\rangle)$$

If $\ell > 1$ and $\sigma = \sigma_0{}^\frown[\rho, a]$, so that $\sigma^{\ell-2}$ is the outbound step from router $\rho \in R$ to area $a \in A$, then

$$\mathcal{F}(\sigma) = \mathcal{F}(\sigma_0{}^\frown[r]) \cap \boldsymbol{outb}(\langle a, \rho\rangle)$$

We use this idea in both of the following sections.

5.1 Checking a Posture

To check that a posture enforces a policy P, it is enough to ensure that the feasibility set for any path between two areas is included in the policy constraint for the areas.

Lemma 2 (Enforcement by feasibility sets)
A filtering posture $\langle \boldsymbol{inb}, \boldsymbol{outb}\rangle$ enforces a policy P iff for all σ, if $\ell = length(\sigma)$, $\sigma(0) \in A$, and $\sigma(\ell) \in A$, then $\mathcal{F}(\sigma) \subseteq P(\sigma(0), \sigma(\ell))$.

Algorithmically, it is enough to check this property for cycle free paths (Definition 2). If σ is a path from a_0 to a_1, then there is a subpath of σ which is a cycle free path from a_0 to a_1. Moreover, if σ_0 is a subpath of σ_1, then $\mathcal{F}(\sigma_1) \subseteq \mathcal{F}(\sigma_0)$. Thus, we may implement Lemma 2 using a depth-first traversal of $\mathcal{N}$.

```
Violations found in passing
from: external
to:   engineering
along path through: <allied>
Violations:
((((areas allied))                                  ; srcs
  ((areas engineering)
   (without eng-mail-server db-server))               ; dsts
  ((http from_server)
   (http to_server) (ftp_data from_server) (ftp_data to_server)
   (ftp from_server) (ftp to_server) (telnet from_server)
   (telnet to_server)))

 (((areas allied))                                  ; srcs
  ((with db-server))                                  ; dsts
  ((db-query to_server)))

 (((areas allied))                                  ; srcs
  ((with eng-mail-server))                            ; dsts
  ((smtp from_server) (smtp to_server))))
```

Table 10: Error Report: Spoofing an Allied Source

5.1.1 Posture Checking: Corporate Example

Using this method, we can discover that to enforce the corporate policy, we must filter packets passing from the `external` area to the `allied` area. One might have supposed[3] to the contrary that no filtering would be needed for packets passing through the router between them, as there is no policy statement constraining traffic between them. However, a prototype implementation of the algorithm suggested by Lemma 2 detected a violation. Output of the prototype, presented in Table 10, indicates that packets may travel from `external` to `engineering` by way of `allied`, contrary to policy, if they purport to have their source in `allied`, and if they select a destination and service that would have been permissible had the packet really originated in `allied`. Given that one wants packets with the descriptions given in Table 10 to enter `engineering` if they have originated within `allied`, one must prevent these packets from entering `allied` from `external`. There are naturally two ways to do so. One could reject them because they have destination addresses in other areas, or because their arrival from `external` is fishy.

In the case of the firewall familair to the author, the first approach was taken. The router at the point of entry from the Internet to the allied organization refuses to pass inbound packets with corporate IP addresses as their destination. This is not implemented in the device's filtering configuration, in fact, but in its routing configuration, which is a mechanism for controlling what routing

[3]As the author did at one time, when he was doubtless very naïve.

tables are permissible. The device supports static no-route assertions, which stipulate that there is no route to certain addresses. These assertions are static in the sense that the device will not update its routing tables no matter what information reaches it via routing protocols such as `egp` or `ospf`. This has the advantage that when packets intended for the corporation reach the router, an `icmp` packet is returned advising the previous router to update its routing tables. This is preferable to having the traffic silently disappear, which is what would happen were the filtering configuration to eliminate the misguided packets.

5.1.2 Posture Checking: Spoofing and Trusted Areas

In some cases, the routing postures required by Definition 7 and Lemma 2 are too pessimistic, too restrictive. The definition of feasibility sets takes account of the possibility that any packet could traverse any area, and this may be false. Some areas may be trusted not to spoof packets. For instance, in the corporate example, we may trust all areas except `external` not to spoof packets. Thus, if a packet with description p originates in area a, and a is one of `allied`, `periphery`, `engineering`, and `financial`, then we may assume that p is of the form $\langle src, dst, s, o\rangle$ where src is either a or some distinguished host d with location $L(d) = a$. In Definition 1, we introduced the notation $\tilde{L}(src) = a$ to combine these two possibilities. Moreover, the security policy may entail that some kinds of packets will never enter an area from any area in which they might have originated. For instance, the corporate security policy entails that no packet of the form $\langle \texttt{external}, dst, \texttt{ftp}, o\rangle$ will enter the area `financial` if it has ever previously been in the area `external`, and that is the only place it could actually originate. We may formalize this idea by the *expected contents* of an area.

Definition 9 (Expected contents; enforcement with limited spoofing) *Suppose that a set of areas $A_0 \subseteq A$ are distinguished as trusted not to spoof packets, and suppose that a policy P is chosen.*

Then $\mathcal{O}(a)$, the expected original contents *of the area a, equals*

- *All packet descriptions p, if $a \notin A_0$;*
- *$\{\langle x, d, s, o\rangle \mid \tilde{L}(x) = a\}$, otherwise.*

$\mathcal{C}(a)$, the expected contents *of a, equals*

$$\mathcal{O}(a) \cup (\bigcup_{a' \neq a} (\mathcal{O}(a') \cap P(a', a)))$$

A filtering posture $F = \langle \textbf{\textit{inb}}, \textbf{\textit{outb}}\rangle$ enforces P with no spoofing from A_0 *iff, for all trajectories $\langle p, \sigma\rangle$, if $p \in \mathcal{C}(\sigma(0))$ and $\langle p, \sigma\rangle$ is feasible relative to F, then $\langle p, \sigma\rangle$ satisfies every policy statement in P.*

We may now introduce a variant $\mathcal{F}'$ of the notion of feasibility set, in which the base case for an area r is $\mathcal{C}(r)$ rather than *All-p*. The smaller value for the base case reflects our knowledge that some areas will never contain certain kinds of packets. Details are straightforward. We have also implemented an algorithm to check whether a filtering posture enforces a policy with no spoofing from a given set of areas.

5.2 Generating a Posture

Although it is useful—and conceptually straightforward—to check the correctness of a given posture, it is more informative to create a satisfactory posture from a policy to be enforced.

It is also a more open-ended problem. There are essentially different solutions to the problem, which is to say different ways to assign filtering behavior, possibly to different routers or to different interfaces of the same router, such that the net result is secure in either case.

A concrete example was mentioned in Section 5.1. In the corporate example, some filtering must be done between when packets pass into the `allied` area from the `external` area. However, that filtering could check either the source field, to detect spoofed messages, or else destination fields, to avoid pass-through packets.

In other examples, which are easy to construct, one may make different decisions about which router rejects a packet. For instance, one approach would be to discard packets early on a path, if no extension of the path that would reach the packet's destination would be compatible with the global policy. Alternatively, one could clip each path later, immediately before it would enter an area incompatible with the global policy.

Although there are a number of different ways to construct filtering postures, those we will consider have a form in common. One starts with a weak filtering posture—perhaps the vacuous posture that does not discard any packet at any interface—and considers a collection of strengthenings that are needed to prevent particular violations of the global policy. Thus, one needs a partial ordering on filtering postures. We use the obvious pointwise condition.

Definition 10 (Weaker filtering posture)
A filtering posture $F = \langle \mathbf{inb}, \mathbf{outb} \rangle$ *is* weaker *than* $F' = \langle \mathbf{inb}', \mathbf{outb}' \rangle$ *iff, whenever* $\langle a, r \rangle \in C$, $\mathbf{inb}'(a, r) \subseteq \mathbf{inb}(a, r)$ *and* $\mathbf{outb}'(a, r) \subseteq \mathbf{outb}(a, r)$. *We will write* $F \preceq F'$ *in this case.*

Because of the properties of $\subseteq$, the "weaker" relation $\preceq$ determines a lattice using pointwise meet $\prod$ and join $\sum$ operations. The meet takes the intersection of the packet descriptions permitted at any interface, while the join takes the union.

Policy enforcement is preserved under strengthening. If $F \preceq F'$, then $\mathcal{F}_{F'}(\sigma) \subseteq \mathcal{F}_F(\sigma)$; so, if $\langle a_1, a_2, \phi \rangle$ is a policy statement that F enforces, then so does F'. Thus, from a security point of view, an algorithm is free to start off with a given posture, construct various strengthenings (possibly independently) path by path or policy statement by policy statement, and take the meet of the resulting postures. By contrast, from the point of view of ensuring that certain services are provided through the network, it may be important to find some cannier strategy that ensures that filtering is done only in preferred places.

5.2.1 Outbound Filtering

Let us start with the simplest approach to these ideas. We will use the notation $f[x_0 \mapsto y_0]$ for function update; i.e. $f[x_0 \mapsto y_0]$ is the function which agrees with f for all arguments other than x_0, for which it yields the value y_0.

Lemma 3 (Posture generation, I) *Let P be a policy and $F = \langle \mathbf{inb}, \mathbf{outb} \rangle$ be a filtering posture; let σ be a path with length $\ell > 1$. Suppose $\sigma(0) \in A$ and $\sigma(\ell - 1) = a \in A$, so $\sigma(\ell - 2) = \rho \in R$.*

Let $\phi_\sigma = \mathcal{F}_F(\sigma) \setminus P(\sigma(0), a)$. Let $F_\sigma =$

$$\langle \mathbf{inb},\ \mathbf{outb}[\langle a, \rho \rangle \mapsto (\mathbf{outb}(\langle a, \rho \rangle) \setminus \phi_\sigma)] \rangle$$

Then $\prod_\sigma (F_\sigma)$ enforces P.

As ϕ_σ is the set of policy violations permitted by F along σ, F_σ does not permit any policy violations on that path. Therefore, the meet over all paths cannot allow any policy violations.

However, this method for generating a posture has several shortcomings. First, it does not say what F to start with. Second, it does not use the inbound filters. Third, it may do redundant filtering. For instance, if σ_0 is a proper initial subpath of σ_1, and $\phi = \phi_{\sigma_0} \cap \phi_{\sigma_1} \neq \emptyset$, then this common subset ϕ of violations will be combined into F twice, once at the end of σ_0 and once at the end of σ_1. But this may be unnecessary: Any packet that traverses σ_1 has necessarily survived all of the filters along σ_0. We may resolve this third shortcoming easily. Recall that we use $[\,]$ for the null sequence, $[x, \ldots, y]$ for the sequence containing the terms shown, and $\sigma ^\frown \tau$ for concatenation. Let P, F, σ, and ℓ be as in Lemma 3.

Lemma 4 (Posture generation, II) *Define ϕ'_σ and F'_σ recursively:*

1. *If $\ell \leq 1$, then $\phi'_\sigma = \emptyset$ and $F'_\sigma = F$.*
2. *$\phi'_{\sigma ^\frown [r,a]} = \mathcal{F}_{F'_\sigma}(\sigma ^\frown [r, a]) \setminus P(\sigma(0), a)$.*
3. *$F'_{\sigma ^\frown [r,a]} =$*

   $$\langle \mathbf{inb}', (\mathbf{outb}'[\langle a, r \rangle \mapsto (\mathbf{outb}'(\langle a, r \rangle) \setminus \phi'_{\sigma ^\frown [r,a]})]) \rangle$$

   where $F'_\sigma = \langle \mathbf{inb}', \mathbf{outb}' \rangle$.

Then $\prod_\sigma (F'_\sigma)$ enforces P.

Although this lemma does not describe a depth-first traversal of the network, it nevertheless justifies one. For, the conclusion still holds if, instead of using F'_σ for certain σ, we use any stronger posture. Indeed, in a depth-first traversal, in each step forward we strengthen the posture as described in Clause 3. When we backtrack to σ, having explored $\sigma^\frown[r,a]$, we continue with the stricter posture we have constructed rather than returning to the posture F'_σ. Thus, when the algorithm completes there is no need to take a meet; the last posture computed is stricter than all the preceding ones, so it is itself the meet. This is, not surprisingly, what the implemented algorithm does, as it performs the strengthenings by side-effecting data structures that hold the filters for the individual router interfaces.

5.2.2 Inbound Filtering

The approach of Lemma 4 uses the outbound filters in a reasonable way, but not the inbound filters. We would like also to put the inbound filters to effective use. We propose to devote them to protection against spoofing. This is a natural use to make of inbound filters, as the inbound filters know which interface the packet has arrived through, which the outbound filter does not. Many human-constructed firewalls use inbound filters for this purpose. The inbound filter is uniquely suited to determining whether a packet which claims to be from a particular source has any business arriving from the area in question.

The currently implemented version is based on a simple idea. Areas are normally relatively homogeneous units, and packets should not normally take a detour through other areas in order to reach a router to which they are directly connected. Thus, if a packet $p = \langle src, dst, s, o\rangle$ reaches a router ρ through its interface to area a, and ρ also has an interface on area $a' \neq a$, then src should be neither a' nor any distinguished host d such that $L(d) = a'$; i.e. $\tilde{L}(src) \neq a'$. For if p had really originated where it claims to have originated, then p should have reached ρ through its interface on a'. This depends only on the structure of the network specification $\mathcal{N}$.

Definition 11 (Inbound filtering) *The inbound filtering posture* $\mathtt{inb}_0$ *is the function that for any arguments* a, ρ *returns*

$$\{\langle src, dst, s, o\rangle \mid \tilde{L}(src) = a \vee \neg C(\tilde{L}(src), \rho)\}$$

Thus, in practice we begin with the filtering posture that pairs this function $\mathtt{inb}_0$ with the vacuous outbound filtering posture $\mathtt{outb}_0$, by which we mean the function that permits everything because $\mathtt{outb}_0(a,\rho)$ is the set of all packet descriptions. Our implemented filter generation algorithm applies Lemma 4 (or rather, the depth-first traversal algorithm justified by Lemma 4) starting from the initial filtering posture $F = \langle \mathtt{inb}_0, \mathtt{outb}_0\rangle$.

5.2.3 Filter Generation: Corporate Example

The filter generation algorithm based on Lemma 4 and Definition 11 produces a good filtering posture in the case of the corporate example of Figure 1. The inbound filtering detects spoofed packets entering the `allied` area, so the error in the hand-coded version (Table 10) is eliminated from the start. The inbound filtering specifications for the router between `allied` and `external` are given in

```
(interface-filtering-specs
 (allied/ext-router
  inbound      ; from
  external
  ((((areas allied))                  ; srcs
    any                                 ; dsts
    ())
   (((areas financial engineering     ; srcs
             periphery external))
    any                                 ; dsts
    all)))

 (allied/ext-router
  inbound      ; from
  allied
  ((((areas external))                ; srcs
    any                                 ; dsts
    ())
   (((areas allied financial          ; srcs
             engineering periphery))
    any                                 ; dsts
    all))))
```

Table 11: Generated Inbound Filtering: The Allied/External Router

Table 11. Outbound filters are generated during a depth-first traversal of the graph, by progressive tightening to ensure that the feasibility set for any path will always be included in the set permitted by the policy. An example, for the router between `engineering` and `allied`, is shown in Table 12. In order to see a limitation of this treatment of inbound filtering, it will be informative to turn to a second example.

5.2.4 Filter Generation: Tactical Example

Our new example concerns tactical communications. We consider a hypothetical situation in which a tactical headquarters consists of two separate local area networks, connected by two different radio networks to various field units that are at either the battalion level or the company level, as illustrated in Figure 2.[4]

In this example, the main motivation for access control may be the bandwidth of the radio networks: if a packet goes out on the wrong radio network, or if it is routed circuitously over both, then some other packet cannot be delivered in time. To prevent this denial-of-service attack, which may be a serious information warfare threat, we propose the following security goals:

- No packet should ever traverse both the Radio1 network and the Radio2 network;

[4]The situation is hypothetical, but the problem is not.

```
(interface-filtering-specs
 (eng/allied-router
  outbound      ; to
  engineering
  ((((areas allied))          ; srcs
    ((with db-server))          ; dsts
    ((db-query to_server)))

   (((areas engineering))     ; srcs
    any                         ; dsts
    all)
   (((areas allied))          ; srcs
    ((areas allied financial  ; dsts
              periphery external)
     (without financial-mail-server))
    ())

   (((areas allied))          ; srcs
    eng-untrusted               ; dsts
    ((http to_server)
     (ftp_data to_server) (ftp to_server) (telnet to_server)))

   (((areas financial         ; srcs
              periphery external))
    any                         ; dsts
    ())

   (((areas allied))          ; srcs
    mail-hosts                  ; dsts
    ((smtp from_server) (smtp to_server))))))
```

Table 12: Generated Outbound Filtering: The Allied/Engineering Router

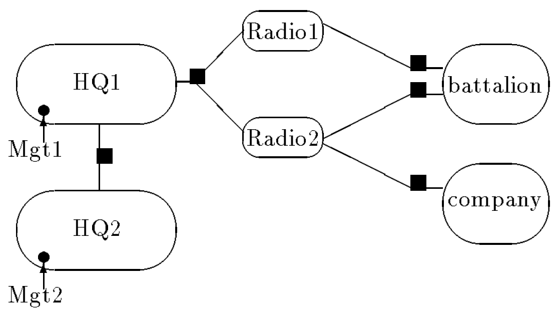

Figure 2: Tactical Communications Example

```
(areas
 ;;name             special hosts
 (hq1               mgt1)
 (hq2               mgt2)
 (radio2)
 (radio1)
 (battalion)
 (company))

(connectivity
 ;; router          areas
 (hq2/hq1           hq2     hq1)
 (hq1/radios        hq1     radio1  radio2)
 (radio1/battalion  radio1  battalion)
 (radio2/battalion  radio2  battalion)
 (radio2/company    radio2  company))
```

Table 13: Specifying the Tactical Example

- No packet that has been on the company network ever traverses the Radio1 network, or *vice versa*;
- The Radio1 network should be used (outbound from the tactical headquarters) only for packets to the battalion level, not the company level, and it should be used only for packets originating from the two special workstations named Mgt1 and Mgt2. They are used for network management within the tactical headquarters, and, over Radio2, within the battalions. The company networks do not contain networking equipment in need of remote management;
- The Radio2 network should be used (outbound from the tactical headquarters) only for packets from the tactical headquarters to the field units.

In this example we are not concerned with what service a packet supports; all services have the same status.

To model this example as a network specification is perfectly straightforward, except for one point. We represent the radio networks themselves as areas, despite the fact that they would not normally contain hosts. Nevertheless, they are areas in the relevant sense, namely that a goal of the security policy is to restrict which packets can enter them. Moreover, there may be many devices with interfaces that put packets onto these areas and receive packets from them. The specification is contained in Table 13. It is useful to introduce a few definitions, which are contained in Table 14, after which the policy may be expressed in a few lines (Table 15). The predefined rule `no-access` denotes the empty set of packet descriptions, and therefore does not permit any packets.

The prototype generates a filtering posture from this input, two aspects of

```
(defined-host-sets
 ;;name              meaning
 (mgt                ((with mgt1 mgt2)))
 (hq                 ((areas hq1 hq2)))
 (field              ((areas battalion company))))

(defined-rule-sets
 ;;name              source  dest                         services
 (mgt-source         ((mgt   all                          all)))
 (hq-source          ((hq    all                          all)))
 (battalion-dest     ((all   ((areas battalion))          all)))
 (company-dest       ((all   ((areas company))            all)))
 (field-dest         ((all   ((areas battalion company))  all))))
```

Table 14: Definitions for the Tactical Example

```
(policy
 ;; from   reaching rule-set
 (radio1   radio2   no-access)
 (radio2   radio1   no-access)

 (company  radio1   no-access)
 (radio1   company  no-access)

 (hq1      radio1   (conjoin mgt-source battalion-dest))
 (hq1      radio2   (conjoin hq-source  field-dest)))
```

Table 15: Policy for the Tactical Example

```
(interface-filtering-specs
 (radio2/battalion
  outbound      ; to
  radio2
  ((((areas company radio1 hq2 hq1))                  ; srcs
    any                                                 ; dsts
    ())
   (((areas battalion radio2))                        ; srcs
    any                                                 ; dsts
    all))))
```

Table 16: Filter Generated for Battalion/Radio2 Traffic

```
Violations found in passing
from: radio1
to:   radio2
along path through: <battalion>
Violations:
((((areas company radio1 hq2 hq1))    ; srcs
  any                                   ; dsts
  all))
```

Table 17: Error Report if Radio2 Filter Omitted

which are of interest. First, despite the fact that there is no policy statement dealing with packets that have previously been in the `battalion` area, nevertheless filtering is needed for packets leaving `battalion`. The filtering rules for the router connecting `battalion` to `radio2` is shown in Table 16. If one removes these rules, to allow instead all packets, then the algorithm of Section 5.1 discloses the error reported in Table 17. This—and one in the corresponding filter between `battalion` and `radio1`—is doubtless the most likely mistake to be seen in practice in this sort of example. It illustrates again that enforcing network access control policies require distributed cooperation to an extent one might not have expected.

The second interesting point disclosed by the example is a weakness in our approach to using inbound filters to prevent spoofing, as formulated in Definition 11. The filter generated for the `hq1/radios` router (as shown in Table 18) has an unexpected asymmetry: it treats the distinguished hosts Mgt1 and Mgt2 differently, despite the fact that the policy requests that they be treated the same. This filter permits packets from Mgt1 to `battalion`, as desired. It also prohibits packets from Mgt1 to less worthy destinations, as desired. But it prohibits packets from all other sources to all destinations. Why is Mgt2 not accorded the same privilege as Mgt1? The easiest way to answer the question is edit the filter specifications to treat both network management workstations permissively. The policy-checking algorithms then report the error shown in

```
(interface-filtering-specs
 (hq1/radios
  outbound      ; to
  radio1
  ((((with mgt1))                             ; srcs
    ((areas battalion))                     ; dsts
    all)
   (((with mgt1))                             ; srcs
    ((areas company radio1 radio2 hq2 hq1))   ; dsts
    ())
   (((areas company battalion radio2 hq2 hq1)
     (without mgt1))                          ; srcs
    any                                       ; dsts
    ())
   (((areas radio1))                        ; srcs
    any                                       ; dsts
    all))))
```

Table 18: Filter Generated for HQ/Radio1 Traffic

```
Violations found in passing
from: radio2
to:   radio1
along path through router: <hq1/radios>
Violations:
((((with mgt2))         ; srcs
  ((areas battalion)) ; dsts
  all))
```

Table 19: Error Report if Radio1 Filter Omitted

Table 19. The problem is that packets may be injected into `radio2` which spoof Mgt2 as their source. These packets will then be routed onto `radio1` just as if they were genuine messages from Mgt2, as the outbound filter onto `radio1` cannot tell that they arrived through the wrong interface. Thus, the cause of the problem is the inbound filter into the router from `radio2`. The rule generated according to Definition 11 is shown in Table 20. It detects spoofed packets from `radio2` if they purport to have source addresses in `radio1` or `hq1`. These are the other areas connected to the router `hq1/radios`. By contrast, the area `hq2` is not directly connected to the router; it is connected only by way of `hq1`. Thus, Definition 11 does not detect packets purportedly from `hq2` that are spoofed on `radio2`.

With this understanding, it is easy to fix the filters specified for the tactical example, by substituting the filtering specification given in Table 21. It is also possible to improve on Definition 11.

We infer that a packet arriving at `hq1/radios` through the `radio2` inter-

```
(interface-filtering-specs
(hq1/radios
  inbound       ; from
  radio2
  ((((areas radio1 hq1))      ; srcs
    any                         ; dsts
    ())
   (((areas company battalion radio2 hq2)) ; srcs
    any                         ; dsts
    all))))
```

Table 20: Inbound Filter Generated for Traffic from Radio2

```
(interface-filtering-specs
(hq1/radios
  inbound       ; from
  radio2
  ((((areas radio1 hq1 hq2))  ; srcs
    any                         ; dsts
    ())
   (((areas company battalion radio2)) ; srcs
    any                         ; dsts
    all))))
```

Table 21: Corrected Inbound Filter for Traffic from Radio2

face and claiming to be from `hq2` must be spoofed, because we know that the only noncyclic path from `hq2` to `hq1/radios` traverses the `hq1` interface. This reasoning may easily be encoded in our framework. However, in this case, we have relied on the structure of the network alone. In the case of a specification like the corporate example (Figure 1), in which there are noncyclic paths from `engineering` (e.g.) to both interfaces of the `external/periphery` router, we would like also to use our security policy. A packet that really originated in `engineering` should not be permitted to travel any noncyclic path leading to an inbound traversal of the external interface on the router. Hence, if any packet that claims to originate in `engineering` does reach that interface, it should be discarded.

On this analysis, the second disjunct in Definition 11, $\neg C(\tilde{L}(src), \rho)$, should be strengthened. Let us say (temporarily) that a trajectory $\langle p, \sigma \rangle$ is *permissible* for a policy P if, whenever σ' is a terminal subsequence of σ, then $\langle p, \sigma' \rangle$ satisfies all of the policy statements in P.

The strengthened disjunct will allow a packet only if $\neg C(\tilde{L}(src), \rho)$ holds for its source and also there is a permissible noncyclic path σ with length ℓ such that

1. $\sigma(0) = \tilde{L}(src)$;
2. $\sigma(\ell - 1) = \rho$; and
3. $\sigma(\ell - 1) = a$, the area from which the packet is arriving.

We have also implemented this idea. It leads to acceptable filtering postures, for instance in the tactical example.

However, it has the drawback that the resulting filtering rules are more complex. At the current stage of work, the results of filter generation can be used only if they are evaluated by a human being and manually translated into configuration files for a particular brand of router. Therefore, there is a great advantage to identifying simple, intuitively well-motivated rules. Future work is likely to lead to automated translation to configuration files, and at that time this more sophisticated treatment of inbound filters may be worthwhile.

5.3 Rectangles and Rules

We use host sets and sets of services to build up rectangles.

Definition 12 (Rectangle, applicability, (dis)allowed)
If h_1 and h_2 are host sets, and S is a set of oriented services, then $r = \langle h_1, h_2, S \rangle$ is a rectangle.

If $p = \langle src, dst, s, o \rangle$, then r applies to *p iff $src \in h_1$ and $dst \in h_2$. We will call the set of p to which r applies the* domain *of r, $dom(r)$.*

If r applies to p, then r allows *p if $\langle s, o \rangle \in S$. It* disallows *$p$ if it applies to p and $\langle s, o \rangle \notin S$.*

Two rectangles r_0 and r_1 overlap *if $dom(r_0) \cap dom(r_1) \neq \emptyset$ and $dom(r_0) \neq dom(r_1)$.*

The domains of two rules $\langle h_1, h_2, S\rangle$ and $\langle h'_1, h'_2, S'\rangle$ are disjoint just in case

$$h_1 \cap h'_1 = \emptyset \ \vee \ h_2 \cap h'_2 = \emptyset$$

Rectangles may fail to overlap for either of two reasons: Either their domains are identical, or else they are disjoint.

To express policy constraints we use sets of rectangles. However, we maintain them in a form with disjoint domains, so that at most one rectangle applies to any packet description.

Definition 13 (Rule, applicability, (dis)allowed)
Suppose R is a set of rectangles. R is a rule *if, whenever r_0 and r_1 are distinct members of R, then r_0 and r_1 have disjoint domains.*

A rule R applies *to p if any $r \in R$ applies to p. R* allows *p if any $r \in R$ allows p; it disallows p if any $r \in R$ disallows p.*

Because the rectangles in a rule R have disjoint domains, it is never the case that R both allows and disallows p.

Definition 14 (Rule equivalence, compatibility)
Two rules R and R' are equivalent *if they apply to the same packets, and allow the same packets.*

Two rules R_0 and R_1 are compatible if, whenever $r_0 \in R_0$ and $r_1 \in R_1$, r_0 and r_1 do not overlap.

A packet description p is an incompatibility point *of R_0 and R_1 if there exists an $r_0 \in R_0$ and an $r_1 \in R_1$ such that $p \in dom(r_0) \cup dom(r_1)$ and r_0 overlaps r_1.*

Rules are compatible if and only if they have no incompatibility points.

Lemma 5 (Dividing rectangles)
If R_0 and R_1 are rules, then there is a rule R'_0 equivalent to R_0 and a rule R'_1 equivalent to R_1 such that R'_0 and R'_1 are compatible.

Proof. If R_0 and R_1 are already compatible, then we are done, so we may assume that they are not. Thus, there exist overlapping rectangles $\langle \eta, \delta, S\rangle \in R_0$ and $\langle \eta', \delta', S'\rangle \in R_1$. Let us define $\eta_1 = \eta \cap \eta'$, $\eta_0 = \eta \setminus \eta_1$, and $\eta_2 = \eta' \setminus \eta_1$; likewise let us define $\delta_1 = \delta \cap \delta'$, $\delta_0 = \delta \setminus \delta_1$, and $\delta_2 = \delta' \setminus \delta_1$. This notation is illustrated in Figure 3. The assumption that the rectangles overlap ensures that $\eta_1 \neq \emptyset$ and $\delta_1 \neq \emptyset$; it also ensures that $\eta_1 \times \delta_1 \neq (\eta \times \delta) \cup (\eta' \times \delta')$.

Consider the set of rectangles $X_0 =$

$$(R_0 \setminus \{\langle \eta, \delta, S\rangle\}) \cup \{\langle \eta_0, \delta, S\rangle, \langle \eta_1, \delta_0, S\rangle, \langle \eta_1, \delta_1, S\rangle\}$$

X_0 is equivalent to R_0 because it simply divides the original rectangle between three smaller rectangles, assigning the same services to each. Similarly, the set of rectangles $X_1 =$

$$(R_1 \setminus \{\langle \eta', \delta', S'\rangle\}) \cup \{\langle \eta_2, \delta', S'\rangle, \langle \eta_1, \delta_2, S'\rangle, \langle \eta_1, \delta_1, S'\rangle\}$$

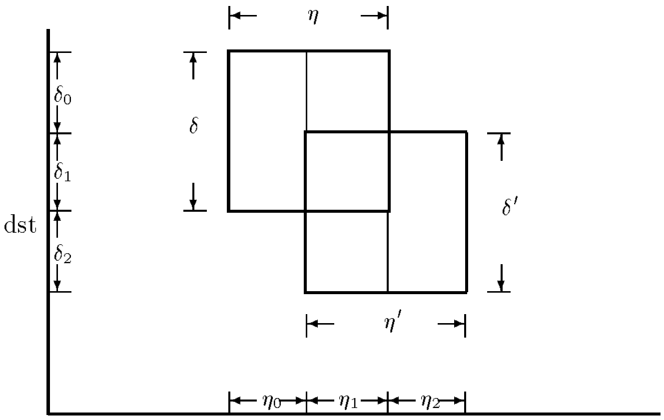

Figure 3: Notation for Proof of Lemma 5

is equivalent to R_1. Moreover, the rectangles $\langle \eta_1, \delta_1, S\rangle$ and $\langle \eta_1, \delta_1, S'\rangle$ have the same domain, while all the other pairs of rectangles shown from X_0 and X_1 have disjoint domains. Hence, X_0 and X_1 are rules.

Although X_0 and X_1 may not be compatible, their incompatibility points are a proper subset of those of R_0 and R_1.

Because $\eta_1 \neq \emptyset$ and $\delta_1 \neq \emptyset$, there exists at least one $p \in \mathrm{dom}(\langle \eta_1, \delta_1, S'\rangle)$. It is an incompatibility point of R_0, R_1 but not of X_0, X_1. On the other hand, there are no new incompatibility points of X_0, X_1. A new incompatibility point could arise only in two cases, namely if $\langle \eta, \delta, S\rangle$ had the same domain as some rectangle r_1 in R_1 or (symmetrically) if $\langle \eta', \delta', S'\rangle$ had the same domain as some rectangle r_0 in R_0; let us suppose the former. Then r_1 would have overlapping applicability with $\langle \eta', \delta', S'\rangle$, contradicting the assumption that R_1 is a rule. Hence, the incompatibility points of X_0, X_1 are properly included in the incompatibility points of R_0, R_1.

Since $\mathcal{N}$, S are finite structures, the set of packet descriptions is finite. Therefore, there exist compatible rules Y_0 and Y_1 equivalent to R_0 and R_1 respectively. ∎

A number of important operations on rules are easily carried out when the operands are compatible.

Definition 15 (Rule comparisons, operators)
Let R_0 and R_1 be rules.

1. *R is a* subrule *of R' if, for all packets p, if R applies to p, then R' applies to p, and if R allows p, then R' allows p.*[5]

[5] Altered 19 May 1997.

2. *R_2 is a* complement *of R_0 if R_2 allows a packet iff R_0 disallows it, and disallows a packet iff R_0 allows it.*

3. *R_2 is a* completion *of R_0 if R_2 allows a packet iff R_0 allows it, and R_2 disallows all other packets.*

4. *R_2 is a* loosening *of R_0 and R_1 if R_2 allows a packet iff either R_0 or R_1 allow it, and R_2 applies to a packet iff either R_0 or R_1 applies to it.*

5. *R_2 is a* tightening *of R_0 and R_1 if R_2 allows a packet iff both R_0 and R_1 allow it, and R_2 applies to a packet iff either R_0 or R_1 applies to it.*

6. *R_2 is a* difference *of R_0 and R_1 if R_2 is a tightening of R_0 and a complement of R_1.*

Clearly the choice of R_2 is unique (within equivalence) when it exists. First we consider the comparison operators.

Lemma 6 (Compatible rules: comparisons)
Suppose that R_0 and R_1 are compatible rules.

1. *R_0 and R_1 are equivalent iff $R_0 = R_1$ in the extensional sense of containing exactly the same rectangles.*

2. *R_0 is a subrule of R_1 if, for all $\langle h_1, h_2, S\rangle \in R_0$, there exists a S' such that $S \subseteq S'$ and $\langle h_1, h_2, S'\rangle \in R_1$.*

Next, we consider some combining operators.

Lemma 7 (Complement, completion)
Suppose R is a rule, and let $R' = \{\langle h_1, h_2, \overline{S}\,\rangle \mid \langle h_1, h_2, S\rangle \in R\}$, where $\overline{S}$ is the set complement of S. Then R' is a complement of R.

Suppose that R' is a loosening of R and the universal prohibition rule

$$\{\langle any, any, \emptyset\rangle\}$$

Then R' is a completion of R.

Lemma 8 (Compatible rules: combinations)
Suppose that R and R' are compatible rules. If f is a binary operator on sets, let R_2 be:

$$\begin{array}{lll}\{\langle h_1, h_2, S\rangle \mid & & \langle h_1, h_2, S\rangle \in R \wedge (\forall S' \,.\, \langle h_1, h_2, S'\rangle \notin R') \\ & \vee & \langle h_1, h_2, S\rangle \in R' \wedge (\forall S' \,.\, \langle h_1, h_2, S'\rangle \notin R) \\ & \vee & \exists S_1, S_2 \,.\, S = f(S_1, S_2) \\ & & \quad \wedge \langle h_1, h_2, S_1\rangle \in R \wedge \langle h_1, h_2, S_2\rangle \in R'\}\end{array}$$

1. *If f is set union, then R_2 is a loosening of R and R'.*

2. *If f is set intersection, then R_2 is a tightening of R and R'.*

Example	Areas	Spec Size	Filter Size	Run time (s)
Tactical	6	1122	4518	0.11
Corporate	5	4538	11361	0.63
2 Corps.	8	9009	22082	3.72
3 Corps.	11	14167	42975	17.3

Table 22: Timings for Filter Generation

3. If f is set difference, then R_2 is a difference of R and R'.

Since, by Lemma 5, for every pair of rules there is a pair of compatible rules to which they are respectively equivalent, there always exist loosenings, tightenings, and differences of given rules. As we have already noted, the result must be unique to within equivalence; this is the finest individuation of rules we will need.

Corollary 9 *Loosening, tightening, and difference are binary functions on rules (to within equivalence).*

These operations are the basic ones needed to express policies and to reason about whether a posture enforces a policy. Luckily, they may be implemented somewhat more efficiently than by using the algorithm implicit in the proof of Lemma 5. Also useful are conjunction and disjunction operators, which complete their arguments:

Definition 16 (Conjunction, disjunction)
If R_0 and R_1 are rules, then the conjunction *of R_0 and R_1 is the tightening of their completions. Their* disjunction *is the loosening of their completions.*

This machinery of rectangles and rules leads to a reasonably efficient implementation. A prototype, not carefully optimized for memory allocation (e.g.), has been implemented using the T programming language [10]. Although the T compiler was a state-of-the-art compiler at the time it was developed, it is now about a decade old. Newer language implementations are considerably more efficient.

Nevertheless, run times for these algorithms are negligible. Timings given in Table 22 were made on a HyperSparc processor (not the faster UltraSparc) with ample memory. It is a 125MHz, 131 SPECint_92 machine. The table displays the number of areas for each example, the size of the input specification file in bytes, the size of the generated filtering posture in bytes, and the run time in seconds. This is the time used to generate filters using the approach of Section 5.2.2. We show the tactical communications example, the corporate example, and two expansions of the corporate example.

The first expansion was created from it by adding three new areas, corresponding to the `engineering`, `finance`, and `periphery` areas of a new corporation. They are connected to the `allied` and `external` areas as was the original corporation. Policy for the new corporation is similar to that for the old one, except that several protocols were renamed to increase the total number of services. The second expansion adds yet another corporation in the same way, but without renaming the protocols again. We suspect that many realistic examples will be smaller than the largest shown here.

6 Conclusion and Future Work

We believe that this approach to specifying and analyzing network access control policies has substantial benefits. It provides a compact, unambiguous statement of the security goals for a particular network. It provides a fairly mechanical solution to the localization problem for deriving filter behavior that will enforce a security policy. And it provides a fully mechanical check as to whether a proposed localization successfully enforces a policy.

Several areas for future work remain. First, as we emphasized in the Introduction, in addition to the localization problem, there is also a matter of implementation, namely encoding a filter configuration file that will correctly enforce those choices on the equipment actually available.

Considerable work in this direction has been carried out by colleagues at MITRE. An immediate goal is to reimplement the methods described here in a form compatible with that work. It should lead to automatic generation of reasonably efficient filtering configuration files for such commercial routers as those produced by Cisco (version 11) and Network Systems Corporation. A comprehensible user interface will also be a concern.

Second, we have not concentrated on aspects of network access control needed to protect the routers themselves, as opposed to using the routers to protect the hosts on the various network areas. These security goals can be modeled in the framework we have developed, but it is probably desirable to introduce some extra machinery to do so in a more natural way.

It is also likely that additional graph algorithms will support several new kinds of analysis. For instance, with current interest in authenticated headers [3, 1] and in using tunneled, encrypted packets to support virtual private networks [2], it is likely that new packet attributes will be needed in addition to orientation (to/from server). New operations will be needed to analyze security policies for tunneled packets.

There are also some extensions of larger scope, of which we will briefly consider two, namely specifications of service (Section 6.1) and router security testing (Section 6.2).

6.1 Service Specifications

In the tactical example (Section 5.2.4), we illustrated the fact that a filtering posture may be generated that does not allow the network to provide the functionality that we desire. In that case, packets with source address Mgt2 and destination in the `battalion` area could not be delivered, despite our wanting them to be. Nor does the security policy entail that they need to be stopped. We learnt this by inspecting the generated filtering posture directly.

A more systematic approach would be to introduce the idea of a network *service policy*, dual to the notion of a network access control policy introduced in Definition 4. A service policy characterizes the minimum level of service between areas that should be assured by the filtering posture. A service policy will associate a set of packet descriptions with each pair of distinct areas. A filtering policy will then be described as *respecting* a service policy if, for every pair of distinct areas a_0, a_1, and every packet description p in the associated set, there exists a feasible trajectory for p starting in a_0 and ending in a_1. A method dual to that of Lemma 2 will then allow us to compute whether a given filtering policy respects a service policy.

It is likely that the combination of service checking with the security methods of Section 5 will allow a human operator quickly to converge on a reasonable filtering posture if any exists. Alternatively, the minimal level of service to be assured may be incompatible with the maximal secure level of access. In this case compromises may have to be made, or additional routers may have to be purchased to change the graph structure of the network.

6.2 Router Security Testing

The real world being what it is, it is crucial to supplement a development method, no matter how systematic it is, with a method for testing the results. For one thing, commercial routers sometimes behave in unexpected ways, and bugs in the interpreters that execute filter configuration files should be expected. Second, one sometimes has a carefully hand-crafted configuration file in place, which may incorporate special hacks either for efficiency or to handle odd cases. One may prefer to test whether the file behaves as expected, rather than discard it. Third, one may have some router equipment from unusual vendors for which configuration files could not be generated automatically, in which case, having constructed the configuration file by hand, one would want to test it systematically.

In each of these situations, one would like to take each individual router, one at a time, and determine the packets it transmits using a test apparatus. One needs to generate a large number of packets that will exercise all aspects of the configuration file, together with a prediction for each packet whether it should be transmitted or discarded. One possible test driver would consist of a workstation running in promiscuous mode, connected to the networks on the different router interfaces. The workstation would inject the packets going inbound into the router, and record whether the transmitted packets correspond to the predictions.

The ideas that we have introduced seem highly effective for this purpose, supplemented—naturally—with information about the IP addresses and subnet masks for the areas and distinguished hosts. Packet descriptions, in the sense of Definition 3, abstract just those ingredients that should make a difference. Moreover, the interface filtering specifications predict exactly which packet descriptions should be acceptable to the router.

The number of packet descriptions is limited. For instance, in the case of the largest example, the three-corporation example mentioned in Table 22, there are 14,812 packet descriptions. If the source or destination of a packet description is an area, one will want to test a number of concrete IP addresses within the area. In some situations these IP addresses should be generated randomly; however, when we are testing an existing configuration file, it may be revealing to select them from addresses and subnet masks appearing in the configuration file. In the three corporation example, 3,388 packet descriptions have areas as both their source and their destination, while 7,392 packet descriptions have an area as source or destination but not both. Thus, generating a set of test cases to achieve very substantial coverage is by no means prohibitive.

These methods would seem to provide a more systematic basis for ensuring that every packet takes a safe trajectory in real networks.

A Grammar for the Specification Language

In Table 23, we present a BNF-style grammar for the specification language. Square brackets [] enclose optional components; * expresses repetition zero or more times. Terminals are written here in upper case, although the language itself is case-insensitive.

```
areas         ::= (AREAS (name host-name*)*)
services        ::= (SERVICES (name [protocol server-number])*)
protocol        ::= TCP | ICMP | UDP | RPC/UDP
defined-host-sets
                ::= (DEFINED-HOST-SETS (name host-set-spec)*)
host-set-spec   ::= name | ([(AREAS name*)]
                            [(WITHOUT name*)] [(WITH name*)])
                  | (UNION host-set-spec1 host-set-spec2)
                  | (INTERSECTION host-set-spec1 host-set-spec2)
                  | (DIFFERENCE host-set-spec1 host-set-spec2)
                  | (COMPLEMENT host-set-spec1)
rectangle-spec  ::= (host-set-spec1 host-set-spec2 services-spec)
services-spec   ::= ALL | (oriented-service*)
oriented-service
                ::= (name service-dir)
service-dir     ::= FROM_SERVER | TO_SERVER
defined-rules   ::= (DEFINED-RULES (name rule-spec)*)
rule-spec       ::= (rectangle-spec*)
                  | name
                  | (REFLECT rule-spec)
                  | (EXTEND  rule-spec1  rule-spec2)
                  | (LOOSEN  rule-spec1  rule-spec2)
                  | (TIGHTEN rule-spec1  rule-spec2)
                  | (CONJOIN rule-spec1  rule-spec2)
                  | (DISJOIN rule-spec1  rule-spec2)
policy          ::= (POLICY (from-name to-name rule-spec)*)
connectivity    ::= (CONNECTIVITY (router-name area-name*)*)
filtering-specs ::= (INTERFACE-FILTERING-SPECS
                     (router-name interface-dir area-name rule-spec)*)
interface-dir   ::= OUTBOUND | INBOUND
no-spoofing-spec
                ::= (NO-SPOOFING names)
router-non-delivery
                ::= (NO-ROUTE
                     (router-name host-set-spec)*)
```

Table 23: Grammar for the Specification Language

B Full Specification for Corporate Example

```
(areas
 ;; name        distinguished hosts
 (external)
 (periphery    proxy-host)
 (engineering  eng-mail-server db-server)
 (financial    financial-mail-server)
 (allied))

(connectivity
 ;; router name         areas
 (per/ext-router        periphery external)
 (per/eng/fin-router    periphery engineering financial)
 (eng/fin-router        engineering financial)
 (eng/allied-router     engineering allied)
 (allied/ext-router     allied external))

(services
 (telnet                tcp  23)
 (ftp                   tcp  21)
 (ftp_data              tcp  20)
 (nfs                   udp  666)
 (x                     udp  6000)
 (db-query              rpc/udp 2342)
 (smtp                  tcp  25)
 (http                  tcp  80))

(defined-host-sets
  (internal                          ; new name
   ((areas engineering              ; two areas
            financial)))
  (corporate
   ((areas periphery                ; three areas
            engineering financial)))
  (mail-hosts
   ((with eng-mail-server            ; three disting. hosts
         financial-mail-server)))
  (trusted-hosts
   (union                            ; union of
    ((areas periphery))             ; area and
    (union mail-hosts                ; host set
          ((with db-server)))))
  (eng-untrusted
   (difference
    ((areas engineering))
    trusted-hosts)))

(defined-rule-sets
  ;; rule name
```

```
  (source-non-corporate
   ;; source host set                 dests     services
   ((((areas allied external))        all     all)))
  (source-allied
   ((((areas allied))                 all     all)))
  (destination-internal
    ((all                          internal     all)))
  (destination-proxy
    ((all             ((with proxy-host))     all)))
  (destination-mail-hosts
    ((all                        mail-hosts     all)))
  (destination-eng
    ((all                      eng-untrusted    all)))
  (db-query
    ((all              ((with db-server))     ((db-query
                                                to_server)))))
  (usual-service-requests
   ((all                                all     ((telnet to_server)
                                                (ftp to_server)
                                                (ftp_data to_server)
                                                (http to_server)))))
  (email-service
   ((all                                all     ((smtp to_server)
                                                (smtp from_server))))))

(defined-rule-sets
  (allied-incoming-addresses
   (conjoin source-allied destination-internal))

  (allied-to-engineering
   (conjoin
    allied-incoming-addresses
    (loosen
     db-query
     (loosen
      (conjoin destination-eng usual-service-requests)
      (conjoin destination-mail-hosts email-service)))))

  (allied-to-financial
   (conjoin
    allied-incoming-addresses
    (conjoin destination-mail-hosts email-service))))

(policy
 ;; Here are all the clauses that concern the
 ;; allied area
 ;;
 (allied external source-allied)
 ;; We leave external->allied unconstrained.
 ;; It's their problem not ours.
```

```
 ;;
 (allied engineering allied-to-engineering)
 (engineering allied (reflect allied-to-engineering))

 (allied financial allied-to-financial)
 (financial allied (reflect allied-to-financial))

 (allied periphery no-access)
 (periphery allied no-access))

(defined-rule-sets
  (external-to-periphery
   (conjoin
    source-non-corporate
    (loosen destination-proxy destination-mail-hosts)))

  (incoming-email
   (conjoin
     source-non-corporate
     (conjoin destination-mail-hosts email-service)))

  (outgoing-email (reflect incoming-email))

  (periphery-to-internal
   (loosen
    incoming-email
    (reflect
     (conjoin destination-proxy usual-service-requests)))))



(policy
 ;; Here are all the clauses that concern the
 ;; periphery area (but not involving allied)
 ;;
 (external periphery external-to-periphery)
 (periphery external (reflect external-to-periphery))

 (periphery engineering periphery-to-internal)
 (engineering periphery (reflect periphery-to-internal))

 (periphery financial periphery-to-internal)
 (financial periphery (reflect periphery-to-internal)))

(defined-rule-sets
  (financial-to-engineering-email
   ((((with financial-mail-server))
     ((with eng-mail-server))
     ((smtp to_server)
       (smtp from_server)))))
```

```
  (financial-to-engineering
   (loosen
    outgoing-email
    (loosen
     db-query
     (loosen
      financial-to-engineering-email
      (conjoin destination-eng usual-service-requests))))))
```

```
(policy
 ;; Here are all the clauses that concern the
 ;; engineering area (but not involving allied or periphery)
 ;;
 (external engineering incoming-email)
 (engineering external outgoing-email)

 (financial engineering financial-to-engineering)
 (engineering financial (reflect financial-to-engineering)))

(policy
 ;; The remaining clauses for financial concern
 ;; the external area.
 ;;
 ;;
 (financial external outgoing-email)
 (external financial incoming-email))
```

References

[1] R. Atkinson. IP authentication header. Internet Request for Comments 1826, August 1995. Available at `http://ds.internic.net/rfc/rfc1826.txt`.

[2] R. Atkinson. IP encapsulating security payload (ESP). Internet Request for Comments 1827, August 1995. Available at `http://ds.internic.net/rfc/rfc1827.txt`.

[3] R. Atkinson. Security architecture for the Internet Protocol. Internet Request for Comments 1825, August 1995. Available at `http://ds.internic.net/rfc/rfc1825.txt`.

[4] D. Brent Chapman and Elizabeth D. Zwicky. *Building Internet Firewalls.* O'Reilly and Associates, Sebastopol, CA, 1995.

[5] William R. Cheswick and Steven M. Bellovin. *Firewalls and Network Security: Repelling the Wily Hacker.* Addison-Wesley, Reading, MA, 1994.

[6] Cisco Systems, San Jose, CA. *Router Products Command Reference*, 10th edition, 1994. Chapters 10 to 17 (especially Chapter 16).

[7] Cisco Systems, San Jose, CA. *Router Products Configuration Guide*, 10th edition, 1994. Chapters 10 to 17 (especially Chapter 16).

[8] Stephen Cobb. NCSA firewall policy guide. White paper, National Computer Security Association, Carlisle, PA, 1996.

[9] Joshua D. Guttman. Filtering postures: Local enforcement for global policies. In *Proceedings, 1997* IEEE *Symposium on Security and Privacy*, Los Alamitos, CA, May 1997. IEEE Computer Society Press.

[10] David Kranz, Richard A. Kelsey, Jonathan A. Rees, Paul Hudak, Jim Philbin, and Norman I. Adams. ORBIT: An optimizing compiler for Scheme. *SIGPLAN Notices*, 21(7):219–233, June 1986. Proceedings of the '86 Symposium on Compiler Construction.

[11] Network Systems Corporation, Minneapolis, MN. *The Security Router*, 3.0rd edition, May 1996. See especially the "Packet Control Facility" chapter.

[12] Karanjit Siyan and Chris Hare. *Internet Firewalls and Network Security.* New Riders Publishing, Indianapolis, IN, 1995.

[13] W. Richard Stevens. TCP/IP *Illustrated, Volume 1: The Protocols.* Addison-Wesley, Reading, MA, 1994.

Distribution List

INTERNAL

G021

D. J. Bodeau
W. M. Farmer
W. R. Gerhart
J. D. Guttman (8)
L. J. LaPadula
M. C. Michaud
F. J. Thayer Fábrega

G022

INFOSEC Library

PROJECT

National Security Agency
9800 Savage Road
Ft. Meade, MD 20755-6000

R23

Mr. Gene Myers (3)
Mr. Grant Wagner