Filtering Postures: Local Enforcement for Global Policies *

Joshua D. Guttman
The MITRE Corporation
guttman@mitre.org

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.

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.

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. While we use the vocabulary of TCP/IP, the ideas and methods we introduce are also applicable to other protocols.

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 among the routers will ensure that the constraints will be enforced, no matter what path through the network a packet might take. This is a problem of localization.

This paper makes two contributions. The first is a straightforward way to define a security policy for a network, as a global policy about what packets can get where, regardless of path. The second is a method for solving the localization problem, to determine the filtering decisions of individual routers. These decisions can be based only on local information: namely, what interface the packet arrived at; what interface the packet will be routed out through; and what the headers say. An advantage of this approach is that it can be made fully rigorous [9], yielding an automated verification method for this particular security problem. A prototype implementation helped us refine the method and establish its feasibility.

Because we will consider only a logical description of the filtering to be done at each router interface, we will coin a new phrase. A "filtering posture" will mean an assignment of filter functional behavior to each router interface in a network. It 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 not discuss how to encode a filter configuration file that will correctly enforce those choices on the equipment actually available.

In Section 2 we will introduce an example of the network access control localization problem. Section 3 describes how we formalize security policies, and it introduces the

*Work supported by the National Security Agency under United States Army CECOM contract DAAB 07-96-C-E601. This paper appears in the *Proceedings, 1997 IEEE Symposium on Security and Privacy*.

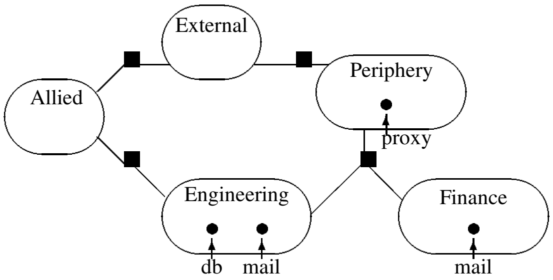

Figure 1. Example Corporate Network

network model within which we will work. Section 4 introduces a simple language that can be used to represent networks and policies. We encode our motivating example in the language. Section 5 turns to policy enforcement; it describes how to solve the localization problem. A conclusion (Section 6) summarizes and mentions some future work.

2 A Motivating Example

Suppose that a 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.

Users rely on the network 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. Database queries use remote procedure calls via UDP packets to port 1025 (let us say) on that server. 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 (the client) and the proxy host or else alternatively the proxy host and an external host (the server). 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. Conversely, all packets entering the internal networks from the allied networks should 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. Cobb [8] points out that internal

> firewalls can notably reduce the threat of internal hacking..., a problem which consistently outranks external hacking in all the surveys.

3 Formalizing the Security Goals

How can we formalize these security goals? Two types of ingredient appear to be relevant:

1. Which areas has the packet traversed; for instance, was it once in the external area, and has it now reached the engineering area?

2. What does the packet say? This in turn involves primarily four ingredients, although others (e.g. `syn` and `ack` bits) are 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;
    - Whether the packet is traveling from the client to the server or from the server to the client. Conceptually, this may be inferred from whether the recognizable server port appears as the source port or the destination port, although router hardware may use the `syn` and `ack` bits instead, in the case of the crucial packets that set up a TCP connection.

Ingredient 1 concerns the actual path of the packet as it traverses the network, regardless of what it claims. Ingredient 2 concerns only what the packet claims, not where it has really passed. These two kinds of information diverge when routers send packets through unexpected paths, or when packets are spoofed, or when packets are intercepted before reaching their nominal destinations. A useful notion of security policy must consider both kinds of information.

3.1 Policy Statements and Policies

We adopt a simple notion of network access control policy that balances actual trajectory and header contents. A policy statement concerns two distinct areas occurring in the actual path of the packet, one earlier network area and one later network area. If ϕ is some predicate of packets, and p ranges over packets, then

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

is a *policy statement* when $a_1 \neq a_2$. It requires that a_2 be protected against non-ϕ packets if they have ever been in a_1. For instance,

> If p was ever in the external area and later reaches the engineering area, then p should be an SMTP packet with its destination the mail host

would be a policy statement relevant to the corporate example.

It would also be possible to consider more complicated policy statements, involving e.g. three areas. As an example, we might require a packet that came from the external area via the allied area and eventually reached the engineering area to have:

- an external address as its IP source field;
- an internal address as its IP destination field;
- a source or destination port of 25, indicating that it is an SMTP packet.

Other packets could not pass through the allied area.

However, realistic security goals appear to be expressible using two-area policy statements. In the case of our example, we could replace this three-area policy statement with a (slightly stronger) pair of two-area policy statements. The first would require that if a packet p that was in the external area reaches the allied area, and if p has a destination address in the internal areas, then p's source address should be in the external area and p's service should be SMTP. The second would require that if a packet p that was in the allied area reaches the engineering area, then p's destination address should be in one of the internal areas. If this pair of two-area statements are satisfied, then the three-area requirement will also be satisfied. The extra strength of these two-area statements was probably desired anyway: namely, that the corporation's internal networks should not be used as a pass-through from the allied organization.

Another advantage of using only two-area policy statements is that efficient graph algorithms can solve the localization problem.

Therefore, a *policy statement* will henceforth be a two-area statement, asserting that any packet p that was in one area and later arrives in a different area meets some constraint $\phi(p)$.

A *policy* will mean a set of policy statements, one for each pair of distinct areas a_1, a_2. The constraint may be vacuously true, allowing everything to pass between them; or else at the other extreme, unsatisfiable, requiring that nothing pass.

Source	Destination	Service
1. external	proxy host	`ftp`, `http`, `telnet` (from server)
2. external	mail servers	`smtp` (to/from server)
3. allied	mail servers	`smtp` (to/from server)
4. proxy host	internal	`ftp`, `http`, `telnet` (from server)

Table 1. Packet Constraints for Inbound Traffic

3.2 Policy for the Corporate Example

Table 1 illustrates the properties of packet headers relevant for packets traveling from the external area or the periphery to the internal corporate networks.

- If a packet p traveled from the external area to the periphery area, then one of the first three constraints in Table 1 holds of p.
- If a packet p traveled from the external area to the engineering or financial area, then constraint 2 or constraint 3 in Table 1 holds of p.
- If a packet p traveled from the periphery area to the engineering or financial area, then constraint 2, constraint 3, or constraint 4 in Table 1 holds of p.

No other packets should be permitted to enter any internal area, if they have ever previously been in the external or periphery areas. All of the security goals we have described can be codified in this way.

3.3 Network Model

We regard a network as a bipartite graph. The nodes of the graph consist of the areas we wish to separate—finance, engineering, periphery, external, and allied, in our example—together with the routers (or dual-homed hosts) that connect the areas and move packets between them. There is an (undirected) edge between a router and an area if the router has an interface on that area.

Intuitive notions such as a path through the network may be formalized by natural mathematical concepts [9]. A path through the network is a sequence of immediately connected nodes on the associated bipartite graph. Thus, we ignore issues of routing, so that our conclusions will hold even on the conservative assumption that routing tables may change unpredictably.

Formalizing a real-world network takes some care. We can express access control policies on the network only if they involve flow of packets from one area to a different area; we cannot express requirements on packets traveling within a single area. Nor could we enforce these requirements. Thus, our security goals must determine the granularity of the model.

In addition, we must ensure that all of the real-world connectivity between distinct areas in our networks is represented. We cannot enforce access controls on the traffic between areas if we do not know what routers (or dual-homed hosts) may move packets from one area to another. On the other hand, the areas may represent large collections of physical networks that have many routers within them. Those internal routers are of no interest for our analysis.

3.4 Abstract Addresses and Abstract Packets

We do not care whether a packet is destined for one desktop machine or another. We need only distinguish addresses if they lie in different areas, or if they represent distinguished hosts such as a proxy host or a mail host.

This leads to the notion of an *abstract address*. An abstract address is the name of a distinguished host or the name of an area. An area name will represent the address of any of the ordinary, undistinguished hosts of that area, while a distinguished host name represents the address of that host. We will regard an abstract address as a single item in our mathematical model, even though it may represent many real, concrete `ip` addresses. We simply do not care to differentiate those `ip` addresses, because our security goals treat them uniformly. In our corporate example, there are just nine abstract addresses—five areas and four distinguished hosts—despite the fact that the corporation and its allied organization may use hundreds or even thousands of `ip` addresses.

We define an *abstract packet* p to consist of:

- An abstract address called the source of p;
- An abstract address called the destination of p;
- A service;
- An orientation, which is one of the values `to_server` and `from_server`.

The service of a `tcp` or `udp` packet may be inferred from its destination port or its source port, depending whether its orientation is `to_server` or `from_server`. The service of an `icmp` message may be inferred from its `icmp` header `type` and `code` fields. However, nothing in the analysis described below depends on how the services are modeled, so other notions of service can be incorporated. In addition, other packet attributes can be added, beyond the orientation attribute; for instance, an attribute could be used to indicate whether the header was authenticated, or whether the body is a tunneled, encrypted packet [1, 2].

We regard an abstract packet as a single item in our mathematical model, even though it represents many concrete `ip` packets. These `ip` packets are simply indiscernible, as far as we are concerned, so our theory identifies them into a single abstract packet.

Since the policy in our corporate example concerns six different protocols (counting the FTP control and data connections separately), there are $9 \times 9 \times 6 \times 2 = 972$ different abstract packets. Thus, a reasonably complex network reduces to a very modest number of significantly different cases. Our methods are practical, however, even in specifications where the number of abstract packets is far larger. Neither the user-supplied specifications nor our algorithms need to enumerate individual abstract packets, since the notion of a *rectangle* (Section 4.2) allows us to treat large collections of abstract packets uniformly.

Given the notion of an abstract packet, we may formalize the constraints used in expressing policy: a constraint ϕ is simply a set of abstract packets.

3.5 Filtering Postures

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 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 roughly corresponds with the filtering facilities of commercially available routers, for instance, Cisco routers [6]. Some 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. Since an interface is determined by a choice of an area and a router, we formalize a filtering posture as a pair of functions $\langle \mathtt{inb}, \mathtt{outb} \rangle$. Each of these functions, when given as arguments an area a and a router r, delivers a constraint ϕ as its value. We interpret ϕ as the set of abstract packets permitted to pass the filter at that interface in the direction

```
(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/allied-router     engineering allied)
 (allied/ext-router     allied external))

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

Table 2. Specifying the Corporate Network

indicated.

4 A Specification Language

We 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.

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 forms for our example are in Table 2. A comment stretches from a semicolon to the end of the line. Services are introduced by protocol and server port number.

4.2 Sets of Hosts and Sets of Services

In this subsection we will introduce the linguistic support we need to express policy constraints of the kinds illustrated in Table 1. 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 concern 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 always maintain rules in a form in which all of their rectangles are disjoint.

```
(defined-host-sets        ; define some host sets
  (internal               ; new name
   ((areas engineering    ; two areas
           financial)))
  (corporate
   ((areas periphery      ; three areas
           engineering financial)))
  (mail-hosts
   ((with                 ; two disting. hosts
     eng-mail-server financial-mail-server))))
```

Table 3. Host Sets for the Corporate Example

Any set of abstract address may be defined as:

> All the hosts within zero or more areas,
> omitting zero or more distinguished hosts, and
> including zero or more distinguished hosts.

We present a host set in the form:

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

where any of the keywords `areas`, `without`, and `with` may be omitted if it introduces no names. They may be combined using boolean operations such as `union`, `difference`, and so on.

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 are presented as an example in Table 3.

Oriented services are presented in the form `(service-name orientation)`. A set of oriented services is currently presented by the symbol `all` or by a possibly empty parenthesized list of oriented services.

4.3 Rectangles and Rules

A rectangle may be specified by giving:

- a source host set *src*;
- a destination host set *dst*;
- a list *osvcs* of the oriented services permitted.

A rectangle $\rho = \langle src, dst, svcs \rangle$ *applies* to an abstract packet p if p's source is in *src* and p's destination is in *dst*. If ρ

applies to p and p's service and orientation are in *osvcs*, then ρ *allows* p. If ρ applies to p but does not allow p, then ρ *prohibits* p. We may visualize ρ as being determined by an interval *src* on the x-axis, an interval *dst* on the y-axis, and a coloring. The coloring is the set *osvcs* of oriented services allowed for packets to which ρ applies. Two rectangles ρ_0 and ρ_1 are *disjoint* if there is no p such that both ρ_0 and ρ_1 apply to p; hence, they are disjoint if their source hosts sets are disjoint or their destination host sets are disjoint.

We will represent constraints on packets as *rules*. A rule is a set of mutually disjoint rectangles. Because the rectangles that make up a rule are always disjoint, there is no question about the order in which they are applied. In fact, rules in our sense are logical (declarative). When rules are combined to introduce more complex rules, an explicit operator such as `disjoin` or `conjoin` makes the logical role of the component rules clear. The declarative semantics of our rules is the main contrast with the languages used for current router configuration files, which are order-dependent.

Operators on rules include `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 `reflection` represents the other direction of the same conversations. For a detailed presentation of rectangles and rules, see [9].

4.4 Policy for the Corporate Example

To illustrate the workings of the specification language, we will now present part of a formalization of the security policy for the corporate example of Section 2. We start in Table 4 by introducing some notation, using a `defined-rules` form to give names to useful rules. These clauses define three rules. Each rule consists of a single rectangle. The first rule contains a rectangle that applies to packets with sources in `allied` or `external` and any destination. The rectangle is colored to allow any oriented service. The rectangle in the second rule applies to packets with any source, so long as the destination is the single distinguished host `proxy-host`. Again, the rectangle is colored to allow any oriented service. Finally, the third rule contains a rectangle that applies to packets with any source, so long as the destination is one of the two mail hosts; the coloring allows `smtp` with either orientation.

Turning to the corporate security policy, we require that if p has been in the `external` area and arrives in the `periphery` area, then p's source address must lie either in the `external` area or in the `allied` area. Moreover, p's destination must be either the proxy host (located in the `periphery` area) or else one of the mail hosts (located in the `engineering` and `financial` areas). If its destination is one of the mail hosts, then it must be an `smtp` packet, although its orientation may be either `to_server` or `from_server`.

```
(defined-rule-sets
  (source-non-corporate          ; rule name
   (((areas allied external))    ; sources
     all                         ; dests
     all)))                      ; services
  (dest-proxy
   ((all                         ; sources
     ((with proxy-host))         ; dests
     all)))                      ; services
  (dest-mail-hosts
   ((all                         ; sources
     mail-hosts                  ; dests
     ((smtp to_server)           ; services
      (smtp from_server))))))
```

Table 4. Auxiliary Rules for the Corporate Policy

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

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

Table 5. Rules for the Periphery and External Areas

Conversely, if p was ever in the `periphery` area, and later reaches the `external` area, then p should satisfy the reflection (Section 4.3) of this rule.

The policy statements for the `external` and `periphery` areas are formalized in Table 5. The remainder of the specification is equally straightforward.

The input specification language also permits a user to specify filtering rules for particular router interfaces. We assign a filter by giving the name of the router, the direction, the area in which the interface lies, and a rule specification. For instance, in the corporate example, we could specify one of the filters for the router between `external` and `periphery` as shown in Table 6. The prototype uses the same notation for output when it generates localized filtering rules. Hence in practice, it is not necessary to write router interface filtering specifications directly; the proto-

```
(interface-filtering-specs
 (per/ext-router outbound      ; to
  periphery
  (disjoin dest-proxy dest-mail-hosts)))
```

Table 6. Sample Filtering Rule: Out to Periphery

type generates a collection, and we may tailor their functionality by editing them.

5 Reasoning about Policies and Postures

The ideas introduced in Sections 3–4 suggest algorithms that exploit the boolean operations on constraints in combination with the graph structure of the underlying network specification. 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).

Both of these algorithms depend on the notion of the feasibility set of a path. Given a filtering posture $\langle \mathtt{inb}, \mathtt{outb} \rangle$, the feasibility set of a path σ is the set of all abstract packets that survive all of the filters traversed along the path. That is, if σ traverses router r, entering it from area a_1, then an abstract packet p is in the feasibility set of σ only if $p \in \mathtt{inb}(a_1, r)$. If σ enters area a_2 from r, then p is in the feasibility set of σ only if $p \in \mathtt{outb}(a_2, r)$.

We can compute the feasibility set of a path iteratively by starting with the set of all packets; as we traverse the inbound step from a_1 to r, we take an intersection with $\mathtt{inb}(a_1, r)$; as we traverse the outbound step from r to a_2, we take an intersection with $\mathtt{outb}(a_2, r)$. The rectangle representation introduced in Section 4.3 allows us to carry out such computations reasonably efficiently.

We use this idea in both of the following sections.

5.1 Checking a Posture

To check that a posture enforces a policy P, we examine each path between areas to ensure that the feasibility set for that path is included in the policy statement for the areas it connects. If σ is a path starting at area a_0 and terminating at area a_i, we must check that the feasibility set for σ is included in $P(a_0, a_i)$, i.e., the set of abstract packets that can actually traverse the path is a subset of the set of abstract packets permitted to travel from a_0 to a_i.

Algorithmically, it is enough to check this property for noncyclic paths, as the feasibility set for a cyclic path σ_1 must be a subset of the feasibility set for any noncyclic sub-path σ_0. The set of noncyclic paths is fairly small for reasonable examples; in the case of the corporate example, 20 noncyclic paths begin and end at areas (rather than at routers).

```
Violations found in passing
from: external
to:   engineering
along path through: <allied>
Violations:
((((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 7. Error Report: Spoofing an Allied Source

We implement the checking algorithm by a depth-first graph traversal.

Posture Checking: Corporate Example. Using this method, we learn that we must filter packets passing from the `external` area to the `allied` area to enforce the corporate policy. One might have thought–perhaps naïvely—that no filtering would be needed at that router, as there is no policy statement constraining traffic between them.

However, the checking algorithm detected a violation. Output, presented in part in Table 7, 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
- they select a destination and service that would have been permissible had the packet really originated in `allied`.

Since one wants these packets to enter `engineering` if they have originated within `allied`, one must prevent these packets from ever entering `allied` from `external`. There are two ways to do so. One could reject them because they have destination addresses in other areas ("no pass-through"), or because their arrival from `external` with source addresses in `allied` is fishy ("no spoofing").

In the case of a particular firewall familiar to the author, the first approach was taken. The router at the point of entry to the allied organization refuses to pass inbound packets with corporate IP addresses as their destination. This is implemented in the device's routing configuration, rather than in its filtering configuration. The configuration contains no-route assertions, which stipulate that there is no route to corporate IP addresses. The no-route assertions are static in that the device will not update its routing tables no matter what information reaches it via routing protocols such as `egp` or `ospf`. The no-route assertions have an advantage over using filtering rules for this purpose, namely 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 would happen if the filtering configuration eliminated the misguided packets.

5.2 Generating a Posture

Creating a posture is a more open-ended problem. There are essentially different solutions, different ways to assign filtering behavior, possibly to different routers or to different interfaces of a router, such that the net result enforces the global security policy.

The choice between "no pass-through" and "no spoofing" just mentioned is one example; others are easy to construct.

Outbound Filtering. Various posture generation algorithms can be based on the idea of "correcting" a preexisting filtering posture $F = \langle \mathtt{inb}, \mathtt{outb} \rangle$. Suppose that σ is a path from area a_0 to a_i that enters a_i from router r, and suppose that the feasibility set for σ is ϕ. If ϕ is not a subset of the policy constraint $P(a_0, a_i)$, then we can update F to a new filtering posture $F' = \langle \mathtt{inb}', \mathtt{outb}' \rangle$ where F' differs from F only in that

$$\mathtt{outb}'(a_i, r) = \mathtt{outb}(a_i, r) \setminus (\phi \setminus P(a_0, a_i))$$

where $\phi \setminus \psi$ is the set difference of ϕ and ψ. F' tightens[1] F to prevent any policy violations that would otherwise occur on the last step of σ. This change cannot cause any new policy violations, because it cannot increase any feasibility set. It can only reduce the feasibility sets of other paths that also traverse this edge.

Hence, if we start from an arbitrary filtering posture F_0 and iterate this correction process for every cycle free path σ, we will obtain a filtering posture that satisfies the policy P. We organize this process as a depth-first traversal of the graph starting from each area in turn. It performs the tightening by side-effecting data structures that hold the filters for the individual router interfaces. However, this recipe for generating a posture does not say how to use the inbound filters effectively.

Inbound Filtering. We use the inbound filters for protection against spoofing, because they know which interface the packet has arrived through, which the outbound filter does not. Many human-constructed firewalls use inbound filters for this purpose.

As a heuristic, we assume that packets from one area should not take a detour through another area to reach a directly connected router. Our expectation is that there will normally be good connectivity within any one area, and that a packet originating anywhere in an area will easily be able to reach a router if the router has an interface anywhere in that area. Although this expectation may not always be met—for instance when an area, like `external` in the corporate example, consists of most of the Internet—a security policy may choose to require that packets arrive as expected, and act defensively otherwise.

We may easily formalize this heuristic. Suppose a packet p reaches a router r through its interface to area a, but the source field of p asserts that it originates in area a' where $a' \neq a$. If r also has an interface on a', then we want to discard p. For, if p had really originated where it claims to have originated, then p should have reached r through its interface on a'. We will refer to the inbound filters that implement this idea as $\mathtt{inb}_0$.

We apply our correction technique starting with $\mathtt{inb}_0$ as inbound filters. As outbound filters, we start from the fully permissive filters $\mathtt{inb}_0$, defined so that $\mathtt{outb}_0(a, r)$ always permits all abstract packets to pass. The correction process constrains the outbound filters to enforce the policy P.

In constructing $\mathtt{inb}_0$ we have used only the structure of the network specification, not the policy or any pre-existing filtering posture. These ingredients may be consulted to produce somewhat more finely tuned filtering postures.

Filter Generation: Corporate Example. The filter generation algorithm just described produces a good filtering posture in the corporate example. The inbound filtering detects spoofed packets entering the `allied` area, so the error in the hand-coded version (presented previously in Table 7) is eliminated from the start. The filtering specification for packets inbound from `external` into `allied/ext-router` is given in Table 8. 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. The filters for the router between `engineering` and `allied` are shown in Table 9.

This filter generation algorithm is by no means ideal for all purposes. A human user can sometimes improve its results by editing the output, using the filter checking algorithm to ensure that the new version still enforces the security policy. Variants of the filter generation algorithm can also improve the filtering postures somewhat.

[1] *F' tightens F* if $\mathtt{inb}'(a,r) \subseteq \mathtt{inb}(a,r)$ and $\mathtt{outb}'(a,r) \subseteq \mathtt{outb}(a,r)$, for all a and r.

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

Table 8. Generated Inbound Filtering: Inbound from External

```
(interface-filtering-specs
 (eng/allied-router
  outbound     ; to
  engineering
  ((((areas allied))             ; srcs
    ((areas
      allied financial           ; dsts
      periphery external)
     (without
      financial-mail-server))
    ())
   (((areas                      ; srcs
      financial periphery
      external))
    any                          ; dsts
    ())
   (((areas allied))             ; srcs
    ((with db-server))           ; dsts
    ((db-query to_server)))
   (((areas allied))             ; srcs
    eng-untrusted                ; dsts
    ((http to_server)
     (ftp_data to_server)
     (ftp to_server)
     (telnet to_server)))
   (((areas allied))             ; srcs
    mail-hosts                   ; dsts
    ((smtp from_server)
     (smtp to_server)))
   (((areas engineering))        ; srcs
    any                          ; dsts
    all))))
```

Table 9. Generated Outbound Filtering: The Allied/Engineering Router

Example	Areas	Spec Sz	Filter Sz	Time
Corporate	5	4538	11361	0.63
2 Corps.	8	9009	22082	3.72
3 Corps.	11	14167	42975	17.3

Table 10. Timings for Filter Generation

5.3 Prototype Implementation

The machinery of rectangles and rules leads to an efficient implementation. A prototype has been implemented using the T programming language [10].

Although many improvements and optimization remain possible, run-times for these algorithms are negligible. Timings given in Table 10 were made on a HyperSparc processor, a 125MHz, 131 SPEC INT_{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 correction approach. We show the corporate example and two expansions of it. In these expansions, two or three corporations (respectively) are connected to the `allied` and `external` areas as the corporation is in the original example. Each corporation has its own periphery, engineering and financial areas. Their policies are similar to the one presented. We suspect that many realistic examples will be smaller than the three-corporation example 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 mechanical solution to the localization problem for deriving filter behavior that will enforce a security policy. It provides a fully mechanical check as to whether a proposed localization successfully enforces a policy.

Several areas remain for future work. First, 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. Work in this direction is under way at MITRE. 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. Some extra machinery would allow us to model this in a natural way. Third, current interest in authenticated headers [3, 1] and in using tunneled, encrypted packets to support virtual private networks [2] will call for some extensions to the methods described here.

There are also some extensions of larger scope under development, namely *specifications of service* and *router security testing*.

A network *service policy* is dual to a network access control policy. It characterizes the minimum level of service between areas that should be assured by the filtering posture. A method dual to that of Section 5.1 allows us to compute whether a given filtering policy respects a service policy. The combination of this service checking method with our security methods should allow a human operator quickly to converge on a reasonable filtering posture if any exists. Alternatively, the minimal desired level of service may be incompatible with the maximal permissible level of secure access. In this case compromises may have to be made, or additional routers purchased to change the graph structure of the network.

The real world being what it is, a development method, no matter how systematic, calls for a method for testing the results. One would like to take each individual router, one at a time, to test which concrete packets it transmits. A large number of packets are needed to exercise all aspects of the configuration file. For each packet, one needs to predict whether it should be transmitted or discarded. When supplemented with information about IP addresses and subnet masks for the distinguished hosts and the areas, our network model should be highly effective, because abstract packets distinguish just those ingredients that should make a difference. Generating a set of test cases by translating each abstract packet to concrete `ip` packets—possibly several packets using different addresses in the same area—should achieve substantial coverage of the relevant cases with moderate numbers of packets.

These methods systematically enforce global network access control policies by combining the local effects of filtering routers.

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

I am also grateful to Lee Benzinger of Lockheed-Martin, who suggested many improvements.

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. B. Chapman and E. D. Zwicky. *Building Internet Firewalls*. O'Reilly and Associates, Sebastopol, CA, 1995.

[5] W. R. Cheswick and S. 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] S. Cobb. NCSA firewall policy guide. White paper, National Computer Security Association, Carlisle, PA, 1996.

[9] J. D. Guttman. Filtering postures: Local enforcement for global policies. MTR 97B007, The MITRE Corporation, November 1996.

[10] D. Kranz, R. A. Kelsey, J. A. Rees, P. Hudak, J. Philbin, and N. 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*, 3.0 edition, May 1996. See especially the "Packet Control Facility" chapter.

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

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

Contents

List of Figures

List of Tables