Computers & Security (2004) **23**, 459–468

ELSEVIER

Computers & Security

www.elsevier.com/locate/cose

The use and usability of direction-based filtering in firewalls

Avishai Wool

School of Electrical Engineering, Tel Aviv University, Ramat Aviv 69978, Israel

Received 5 October 2003; revised 5 October 2003; accepted 16 February 2004

KEYWORDS
Firewalls;
Packet filtering;
Network security;
Anti-spoofing

Abstract The common match fields in firewall rules refer to a packet's source and destination IP addresses, protocol, and source and destination port numbers. However, most firewalls are also capable of filtering based on a packet's *direction*: which network interface card the packet is crossing, and whether the packet is crossing the interface from the network into the firewall ("inbound") or vice versa ("outbound"). Taking a packet's direction into account in the firewall's rules is extremely useful: it lets the firewall administrator protect against source address spoofing, write effective egress-filtering rules, and avoid unpleasant side-effects when referring to subnets that span the firewall.

Unfortunately, the firewall's definition of a packet's direction is different from what users normally assume. If interface `eth0` connects the firewall to the internal network, then, from a user's perspective, "inbound on `eth0`" is actually "Outbound" traffic. This discrepancy makes it very confusing for firewall administrators to use the packet direction correctly, and creates a significant usability problem.

In this paper we review the usefulness of direction-based filtering, identify the usability problem, and critically review the approaches taken by several major firewall vendors. Most vendors expose the raw and confusing functionality to the firewall administrators, while one vendor (Check Point) hides the functionality entirely. Both approaches leave much to be desired. However, recent advances in firewall research show that better alternatives exist: the *Firmato* prototype demonstrates that the firewall management software can *compute* the directions algorithmically for a perimeter firewall.

E-mail address: yash@acm.org.

0167-4048/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cose.2004.02.003

Introduction

Background

Firewalls are the cornerstones of corporate intranet security. Once a firewall is acquired, a security/systems administrator has to configure and manage it to realize an appropriate security policy for the particular needs of the company. This is a crucial task; quoting Rubin et al. (1997): "The single most important factor of your firewall's security is how you configure it". However, while firewalls themselves have seen some impressive technological advances (e.g., stateful inspection, transparency, performance, etc.), firewall configuration and management seem to be lagging behind.

A firewall configuration consists of a sequence of filtering rules. Each filtering rule has a matching part, and an action part. The matching part of a rule consists of values for various fields in a packet header. When a packet enters the firewall, the firewall goes over the sequence of rules until it finds a rule that matches the header values in this packet, and takes the action ("pass" or "drop"[1]) specified in that rule. If no rule matches, the default action (normally "drop") is taken.

The packet header fields that are usually tested by firewall rules are a packet's source and destination IP addresses, protocol, and source and destination port numbers. However, most firewalls are also capable of filtering based on a packet's *direction*: which network interface card the packet is crossing, and whether the packet is crossing the interface from the network into the firewall or vice versa. We call these last capabilities *direction-based filtering*.

The problem

Taking a packet's direction into account in filtering rules is extremely useful: it lets the firewall administrator protect against source address spoofing, write effective egress-filtering rules, and avoid unpleasant side-effects when referring to subnets that span the firewall.

Unfortunately, the firewall's definition of a packet's direction is different from what users normally assume. If interface `eth0` connects the firewall to the internal network, then, from a user's perspective, "inbound on `eth0`" is actually "Outbound" traffic. This discrepancy makes it very confusing for firewall administrators to use the packet direction correctly, and creates a significant usability problem.

Most firewall vendors (exemplified by Cisco and Lucent) seem to be unaware of the usability issues related to direction-based filtering. These vendors simply expose the raw and confusing direction-based filtering functionality to the firewall administrators. A notable exception is Check Point. In order to avoid the usability problem, Check Point chooses to keep its management interface simple, and hide the direction-based filtering functionality in such a way that most users are essentially unable to use it (indeed, many users do not even know that a Check Point FireWall-1 can perform direction-based filtering).

Evidence collected from detailed analyses of corporate firewalls (Wool, 2001b) shows that, in general, many firewalls are enforcing poorly written rule-sets, and in particular, direction-based filtering is often misconfigured or entirely unused. We suspect that direction-based filtering is underutilized in a great part due to the usability problem associated with the vendor's configuration tools.

Contributions

We start by pointing out why direction-based filtering is useful: it lets the firewall administrator protect against source address spoofing, write effective egress-filtering rules, and avoid unpleasant side-effects when referring to subnets that span the firewall.

Next, we identify the usability problem associated with direction-based filtering. This problem stems from the discrepancy between the users' global, network-centric, stance, and the firewall's local, device-centric, stance. While the problem may be familiar to firewall administrators who struggled to configure direction-based filtering, we are unaware of earlier reports of it in the literature.

We then critically review the direction-based filtering capabilities provided by several major firewall vendors. Vendor's solutions generally fall into two categories: either expose the raw and confusing direction-based filtering functionality to the firewall administrators, or hide the functionality entirely.

We conclude that current solutions offered by vendors do not adequately address the usability problem associated with direction-based filtering. However, this situation is not inevitable: for instance, the *Firmato* prototype system (Bartal et al., 1999) includes the Rule Assignment and

[1] There are other possible actions, such as Network Address Translation (NAT), logging, content-filtering, all of which we ignore for simplicity.

DIrection Setting (RADIS) algorithm, which automatically computes the correct direction for every rule. If firewall vendors adopt such an approach and integrate RADIS-like capabilities into their configuration tools, direction-based filtering is likely to be used much more frequently and with fewer misconfigurations. This would both improve corporate network security, and make the public Internet more secure against spoofing attacks.

Organization

In the next section, we describe direction-based filtering, and identify the usability issues associated with it. Next, we discuss various firewall vendors' approaches to direction-based filtering. Finally, we conclude.

Direction-based filtering

Why use direction-based filtering?

Anti-spoofing

It is well known that the source IP address on an IPv4 packet is not authenticated.[2] Therefore, source addresses may be spoofed (forged) by attackers in an attempt to circumvent the firewall's security policy (cf. Chapman and Zwicky, 1995, p. 155). For instance, consider the very common firewall rule "From IP addresses in `MyNet`, to anywhere, any service is allowed", which we denote

$$\texttt{MyNet} \rightarrow * : *. \qquad (1)$$

Assuming that `MyNet` is behind the firewall, this rule is supposed to allow all Outbound traffic from hosts in `MyNet`. However, an attacker on the Internet may spoof a packet's source address to be inside `MyNet`, and set the destination address to some IP address behind the firewall. Such a spoofed packet would clearly match rule (1), and be allowed to enter. Obviously, the attacker will not see any return traffic, but damage has already been done: this is enough to mount a DoS[3] attack against hosts behind the firewall, and is sometimes enough to hijack a `tcp` session (Bellovin, 1989).

[2] If IPSec is used, then the external source IP address may be authenticated, once the IPSec security association is established. However, when IPSec is used to create a Virtual Private Network (VPN), the internal encapsulated source addresses are not necessarily authenticated, and may be hidden from the firewall by the encryption.

[3] Denial of Service.

The main defense against such spoofing attacks, at the perimeter firewall, is based on direction-based filtering.[4] Legitimate packets with source IP addresses that belong to `MyNet` should only enter the firewall from its internal interface. Therefore, giving rule (1) a direction would cause spoofed packets not to match, and to be dropped by a subsequent rule.

Egress-filtering

The primary goal of a firewall is to protect the network behind it. However, it is also important to filter egress traffic—traffic that exits the network. Otherwise, the network may become a launching point of attacks, and in particular, DoS attacks, against other organizations on the Internet, or against other zones in the internal network. During such attacks, the attacking host usually sends spoofed packets, to conceal its true location. Such an attack can come from a compromised host on the internal network, or from any other network that is routing through the internal network (e.g., a business partner). A well-configured firewall can prevent most DoS attacks originating behind it. This is called egress-filtering.

Again, since the problem at hand is rooted in source address spoofing, the most effective way to combat it in IPv4 is by direction-based filtering. The solution outlined in the previous section essentially works as an egress-filtering rule too: if the *only* packets that are allowed to enter the firewall via its internal interface (on their way Out) are those packets with source addresses in `MyNet`, then the firewall will drop all the spoofed DoS attack packets. This forces the attacker to use legitimate source addresses, and makes the attack host easier to trace back.

Note that in the previous section we dealt with protecting the internal network. Egress-filtering deals with protecting other networks from being attacked from the internal network. But since both problems are manifestations of source address spoofing, the solution is very similar and utilizes direction-based filtering.

See Edmead (2002) and SANS Institute (2000) for recommendation on how to write effective egress-filtering rules, for various types of firewalls. The same recommendations also appear in RFC 2827 (Ferguson and Senie, 2000). Interestingly, the language in the RFC speaks of *ingress* filtering, because it is written for an audience of Internet Service Providers to whom source-spoofed traffic is inbound. This is another illustration of the

[4] Other measures that help, to a lesser degree, are preventing ICMP redirects and IP source routing.

confusion surrounding packet directions (see section "Usability problems with direction-based filtering") below.

Zone-spanning

Firewall administrators often define objects that span more than one zone ("side") of the firewall. A typical case is to define the `MyNet` group so that it contains both the internal net and the DMZ[5] (see Fig. 1). When the `MyNet` group is zone-spanning, rule (1) has some unintended side-effects, besides the spoofing vulnerability we already discussed. Specifically, rule (1) now allows all traffic between the DMZ and the internal network, in both directions, because both subnets belong to `MyNet`, and both subnets obviously belong to "*". This defeats the whole purpose of having a DMZ, since a compromised machine in the DMZ has full access to the internal network. Additionally, rule (1) also allows unrestricted Outbound traffic to originate from the servers in the DMZ, which is not considered prudent (cf. Wool, 2002).

The best way to avoid such side-effects is to completely eradicate zone-spanning definitions, at least in "pass" rules. Unfortunately, zone-spanning definitions are often convenient and intuitive, so avoiding them may be difficult. In particular, the "*" (or "Any") built-in definition is zone-spanning.

A less draconian measure would be to set the direction on rule (1), as we suggested in section "Anti-spoofing". For instance, if the rule is applied only to traffic leaving the firewall via its external interface, then traffic between the DMZ and Inside zones will not match the rule and will be dropped.

Usability problems with direction-based filtering

As we saw, direction-based filtering is a highly useful technique to combat various types of spoofing attacks. Unfortunately, though, configuring firewalls to actually use direction-based filtering seems to involve a significant usability problem. This problem is caused by the clash between the user's global, network-centric, stance, and the firewall's local, device-centric, stance. While the author is not aware of systematic research, which quantifies the extent of the problem, there is ample anecdotal evidence supporting its existence on firewall mailing lists such as Firewall Wizards (1997–2003), and, indirectly, in the amount of documentation devoted to explaining the use of direction-based filtering.

[5] DeMilitarized Zone.

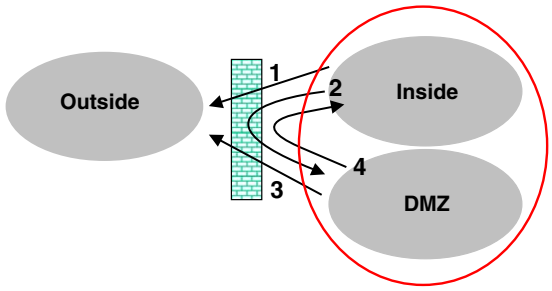

Figure 1 The side-effects of zone-spanning definitions when the rule is "`MyNet` → * : *", and `MyNet` includes both the Inside and the DMZ subnets. Arrow 1 indicates the intended traffic, arrows 2 and 4 indicate side-effects. Arrow 3 could be either intended or side-effect; however, current "best practice" suggests not to allow unrestricted Outbound traffic from the DMZ.

To a firewall administrator, IP addresses are usually split into two disjoint sets: "Inside" invariably means "my protected network" and "Outside" is the rest of the Internet. Traffic directions such as "Inbound" draw their meaning from this dichotomy. When there are other networks involved, such as DMZs, the distinctions blur somewhat, since the DMZ can be viewed as part of either the Inside or the Outside. Still, an Inbound flow of traffic is always understood to be traffic flowing from a less trusted IP address to a more trusted IP address, the latter being within the organization's perimeter. We use capitalized words when referring to such user-level concepts.

To the firewall, however, "inbound on interface `eth0`" means "crossing interface `eth0` from the adjacent network into the firewall". This may completely contradict the user's notion of "Inbound": if `eth0` connects the firewall to the internal, protected, network, then "inbound on `eth0`" is actually "Outbound" traffic (see Fig. 2). Furthermore, the *same* Outbound traffic is both "inbound on `eth0`" and "outbound on `eth1`", assuming that `eth1` is the external interface. This discrepancy makes it very confusing for firewall administrators to use the packet direction correctly.

Note that the difficulty is not merely syntactic or specific to a particular firewall vendor's configuration language. A typical firewall is not aware of the levels of trust given to the networks attached to each of its interfaces.[6] In the absence of such global knowledge, the *only* way to specify a direction for traffic flowing through the firewall is

[6] An interesting exception is Cisco's PIX Firewall, which gives numerical trust-levels to each interface and has syntactical mechanisms to specify directions. See section "Cisco: PIX Firewall" for more details.

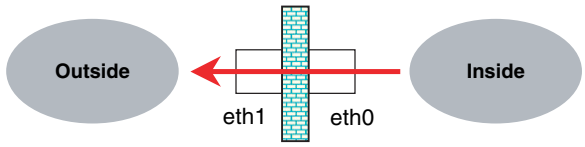

Figure 2 A traffic flow that is Outbound, and is both inbound on `eth0` and outbound on `eth1`.

device-centric, per interface. This implies that, *unavoidably*, the definitions in the firewall configuration language will often clash with the users' understanding.

Vendors' approaches to direction-based filtering

Since there is no standardization in firewall configuration languages, it is no surprise that each vendor has its own syntactical mechanisms to specify rule directions. What is interesting is that vendors have rather different *semantics* associated with their solutions. In this section we review the direction-based aspects of the two leading firewall vendors' configuration tools.

The first vendor we discuss is Cisco, in both its router and PIX Firewall product lines. Cisco's approach is typical of most firewall vendors: it exposes the raw and confusing direction-based filtering functionality to the firewall administrators. Other vendors that follow the same approach (with different syntactical mechanisms) include, among others, Lucent, NetScreen, and open-source tools such as ipchains and netfilter.

The second vendor we discuss (Check Point) takes a different, and unique, approach: it hides the confusing functionality from the firewall administrator.

Cisco: router access control lists

Technical details

Every Cisco router, running a recent version[7] of the IOS operating system (Held and Hundley, 1999), has packet filtering capabilities, and as such can be used as a "firewall". The basic mechanism for packet filtering is via `access-list` commands. An Access Control List (ACL) is a sequence of `access-list` commands, all of which have the same identifier (number or name). Each individual command can be either `permit` or `deny`, and this action is applied to the class of packets which match the command's filter. A basic[8] `access-list` command matches a packet based on its source and destination IP addresses, protocol, port numbers, and optionally some IP header flags. Thus the `access-list` commands themselves do not express any sense of direction (see Fig. 3).

However, an ACL is only active if it is associated with one of the router's interfaces. Each interface can have up to two ACLs associated with it: one in the `in` device-centric direction and one in the `out` direction. Therefore, the administrator can filter based on the direction, at the granularity of a whole ACL, by selecting both the interface and the device-centric direction to which an ACL will be associated.

On a router with N interfaces, the filtering rules may need to be fragmented into $2N$ different ACLs, two per interface. To mitigate some of this complexity, note that each packet must cross two interfaces along its path, one as an inbound packet and one as an outbound packet. Therefore, if we associate an ACL with *every* interface in the `in` direction, we can safely not associate ACLs in the `out` direction—and still be able to control and filter every packet that goes through the router.[9] Experienced network administrators use this observation to reduce the number of ACLs they need to maintain from $2N$ to N (Limoncelli, 2001).

Discussion

On the one hand, Cisco's approach on its IOS-based routers does give the administrator control over direction-based filtering, and the command syntax is relatively straightforward and self-consistent.

On the other hand, the mechanisms, which control direction-based filtering suffer from the following disadvantages. (1) The syntax for specifying direction is device-centric (into and out of the router), which conflicts with the administrator's global view of Inside and Outside. (2) The ACLs implementing the security policy need to be fragmented into many parts. Furthermore, rules that are related (e.g., refer to traffic to and from the same server) need to be written far apart from each other, since `access-list` commands for each direction need to be in different ACLs, and possibly associated with different interfaces.

[7] For instance, IOS v11 or later has the features mentioned here.

[8] So-called a standard or extended access list, in Cisco's parlance.

[9] Associating all the ACLs with the `out` direction is inferior because it would not control packets that are sent to the router itself.

```
IOS (tm) GS Software (RSP-JV-M), Version 11.1(22)

Hssi8/1/0 is up, line protocol is up
   Internet address is 10.0.0.1/24
   Outgoing access list is not set
   Inbound access list is 160
FastEthernet10/1/0 is up, line protocol is up
   Internet address is 10.0.20.2/16
   Outgoing access list is 104
   Inbound access list is not set

Extended IP access list 104
   deny ip host 10.2.26.26 any
   deny ip any host 10.2.26.26
   permit ip 10.2.0.00.0.255.255 any
   permit ip 10.33.20.00.0.0.255 any
   deny ip any any
Extended IP access list 160
   deny ip host 10.17.233.37 any
   deny ip any host 10.17.233.37
```

Figure 3 Parts of a Cisco router configuration, printed by the `show ip interface` and `show ip access-list` commands. We can see the definitions for two interfaces, one with an inbound `access-list`, and the other with an outbound `access-list`. Note that, without additional information, it is impossible to determine whether "`inbound on Hssi8`" refers to an Inbound or Outbound direction.

Cisco: PIX Firewall

Technical details

PIX (Chapman and Fox, 2001) is Cisco's main line of firewalls. Its configuration language has undergone significant changes between versions, and some of the changes affect the issue of direction-based filtering.

A unique feature of the PIX is that it requires a "trust-level" to be assigned to each interface. This is a numerical value in the range 0—100, with trust level 100 assigned to the most trusted interface, normally called `inside`, and trust-level 0 assigned to the least trusted interface, normally called `outside`. Other interfaces, connected to DMZs, business partner connections, etc., should have intermediate (and distinct) trust-levels.

Using these trust-level values, PIX documentation defines the term "inbound" as "from a low trust-level to a high trust-level", and "outbound" as the reverse. Within the section discussing PIX, we shall use the lower-case words *inbound* and *outbound* as they are defined in the PIX documentation.

Note that if the firewall only has two interfaces, then these definitions match the user's notion of Inbound and Outbound. However, if the firewall has more than two interfaces, the trust-level-based directions may not match the user's notions (e.g., is traffic between DMZ-1 and DMZ-2 inbound or outbound?).

Unfortunately, the PIX configuration language uses *different* commands for inbound and outbound traffic. Furthermore, these commands have internally inconsistent syntax and semantics (see Fig. 4).

Up to PIX v4.4, inbound traffic filtering was controlled by the `conduit` command, and inbound NAT was controlled by the `static` command. For outbound traffic, filtering was controlled by a combination of the `outbound` and `apply` commands, and NAT was controlled by a combination of the `global` and `nat` commands. The default setting also differed between inbound and outbound: for inbound traffic, the default was "deny", while for outbound traffic, the default was "permit". In other words, by default, PIX v4.4 allows all outgoing traffic (from an interface *i* to any other interface with a trust-level lower than *i*'s).

The syntax of these six commands is not uniform: e.g., some use positional arguments while others accept named arguments and keywords; on some commands the first IP address is the source while on others it is the destination. The syntax of the `outbound` command is notoriously confusing, since it only accepts a single IP address field (possibly a subnet) and a single port field. Both fields can be interpreted as either a source or a destination, *depending* on the arguments of the `apply` command and on the optional `except` sub-commands.

This myriad of commands make the task of configuring a PIX firewall rather difficult, especially for novice users. An informal survey of postings to firewall mailing lists such as Firewall Wizards (1997—2003) seems to show basic configuration questions being posted much more frequently for PIX than, say, for Check Point. Also, data gathered from the Lumeta Firewall Analyzer (Wool, 2001a,b) indicate that PIX users rarely use the `outbound` command, and those brave users that do use it only use it to drop specific services. Although theoretically possible, the author has never seen an active PIX (up to v4.4) configuration in which the default behavior for outbound traffic has been changed from "permit" to "deny". Thus PIX users rarely implement egress-filtering.

Apparently motivated by user complaints, and also to make the IOS and PIX languages more similar, Cisco made significant changes to PIX with the introduction of v5.0 in 1999. The main change was the introduction of an `access-list` command, whose syntax is almost identical to its IOS counterpart. Unlike IOS, though, there can only be a single PIX ACL (containing multiple `access-list` commands) attached to each interface,

```
PIX Version 4.4(5)
nameif ethernet0 outside security0
nameif ethernet1 inside esecurity100
nameif ethernet2 dmz1 security50

global (outside) 1 100.3.222.12-100.3.222.23
global (outside) 1 100.3.222.24
nat (inside) 1 192.100.100.0 255.255.255.0 0 0
static (dmz1,outside) 100.3.220.0 100.3.220.0 netmask 255.255.255.0 0 0

conduit permit tcp 100.3.220.0 255.255.255.0 eq www any
conduit deny tcp any range 0 65535 host 5.0.0.1
conduit permit tcp host 172.19.100.110 eq 7717 host 100.3.220.101
outbound 2 permit 0.0.0.0 0.0.0.0 0 tcp
outbound 2 except 100.3.220.0 255.255.255.0 1-19 tcp
apply (inside) 2 outgoing-src
```

Figure 4 Excerpts from a Cisco PIX v4.4 configuration file. The final digits in the `nameif` commands denote the trust levels 0, 100, 50 for the three interfaces. Note, for instance, that the `static` command uses a `netmask` keyword, but the `nat` and `outbound` commands accept a network mask as a positional argument. Also, un-intuitively, the first IP address argument to a `conduit` command is the destination rather than the source.

and it is always applied in the device-centric "inbound" direction. As we noted before, this does not reduce the expressiveness of the language, since every packet is inbound on some interface. When `access-list` commands are applied to a PIX interface, the default rule is always "drop", regardless of the interface trust-levels. Hence, the `access-list` command essentially ignores the trust-level associated with the interfaces. Note that trust-level-based commands are still needed for NAT, as there have been no replacements for the `static`, `global`, and `nat` commands.

As of this writing (with PIX at v6.3), the `conduit`, `outbound`, and `apply` commands are still supported, but Cisco recommends that users should migrate to the `access-list` command. Data gathered from the Lumeta Firewall Analyzer (Wool, 2001a) show that, despite Cisco's recommendation, many PIX owners still use the old command syntax even after migrating to new PIX versions.

Discussion

The explicit association of trust-level to interfaces is an intriguing approach. One could expect it to make direction-based filtering easier on the PIX, since it attempts to match the users' notions of Inbound and Outbound. The approach is at its best on very simple firewalls, with only two interfaces.

Unfortunately, the syntax and semantics used in the PIX configuration language up to v4.4 had much more severe usability problems than those associated with direction-based filtering. Overall, PIXes up to v4.4 were just complicated to configure. This general complexity makes it difficult for us to determine whether the trust-level approach *per se* could have been a good mechanism to configure direction-based filtering.

With PIX v5.0, Cisco introduced a marked improvement of the configuration language as a whole, since the `access-list` command is at least clear and self-consistent. However, as far as direction-based filtering goes, moving from trust-levels to the `access-list` command is a step back toward the more rudimentary capabilities of IOS routers, with all the drawbacks mentioned in section "Cisco router access control lists". The one language improvement PIX offers over IOS ACLs is the reduction of the number of ACLs per interface from two to one.

Check Point FireWall-1

Technical details

The filtering policy for the Check Point FireWall-1 product (cf. Welch-Abernathy, 2002) can be configured in two different ways: via the Graphical User Interface (GUI), or via the text-based INSPECT language. Interestingly, when the firewall administrator chooses to install the policy on the firewall from the GUI, Check Point's software in fact produces an intermediate INSPECT file, which is then compiled into the firewall's bytecode. The distinction between the GUI and the INSPECT language is relevant, because the capabilities they expose to the user are different.

The INSPECT language is similar to assembly language: it is very powerful, but of very low level. Using it, the firewall administrator may essentially filter based on any bit in the packet header, while

taking the firewall's state into account. Furthermore, INSPECT can filter based on the interface the packet is crossing, and on the direction (device-centric inbound or outbound). Direction-based filtering may be applied at a rule-by-rule granularity. Thus, the INSPECT language offers the full functionality of direction-based filtering.

However, the vast majority of Check Point users configure the firewall through the GUI, for the same reasons that people use high-level programming languages instead of assembly. And, unlike the underlying INSPECT language, the GUI does not have a sense of direction (see Fig. 5). Using the GUI, the firewall administrator has *no way* of specifying the interface that a rule will be applied to, nor the direction that a rule should have. The GUI maintains a single rule-base, which is applied to *all* interfaces, in both directions. Thus, the Check Point GUI hides the direction-based filtering capabilities that its filtering engine has.

It is the author's opinion that Check Point deliberately chose to omit the direction-based capabilities from the GUI. The GUI has had the same (lack of) direction-based capabilities since FireWall-1 v3.0, despite three major code releases (4.0, 4.1, NG) and several minor code releases over the last few years. Since the GUI produces an INSPECT file, this choice could not have been an oversight. Furthermore, the Check Point GUI is very rich in detail, controlling dozens of features and options. So omitting direction-based capabilities was not done to reduce the visual or operational complexity of the GUI. It is reasonable to conclude that Check Point's design choice attempted to reduce the *conceptual* complexity of the GUI.

To compensate for the lack of direction-based filtering capabilities within the rules, Check Point provides an "anti-spoofing" capability. Anti-spoofing is not configured at a rule-by-rule granularity, but per interface. To activate anti-spoofing, the firewall administrator needs to click on the firewall object's icon, and modify the anti-spoofing setting on each interface. Doing so produces implicit rules in the GUI-produced INSPECT file. These implicit rules all have the form "`Spoofed` → * : * : drop", where the meaning of `Spoofed` depends on the interface to which the rule is attached, and the networks behind it. The implicit anti-spoofing rules are directional, and specify both the interface to which they apply and the device-centric traffic direction they control.

Discussion

It is to Check Point's credit that, unlike their competitors, they identified the usability issue surrounding direction-based filtering. Making the

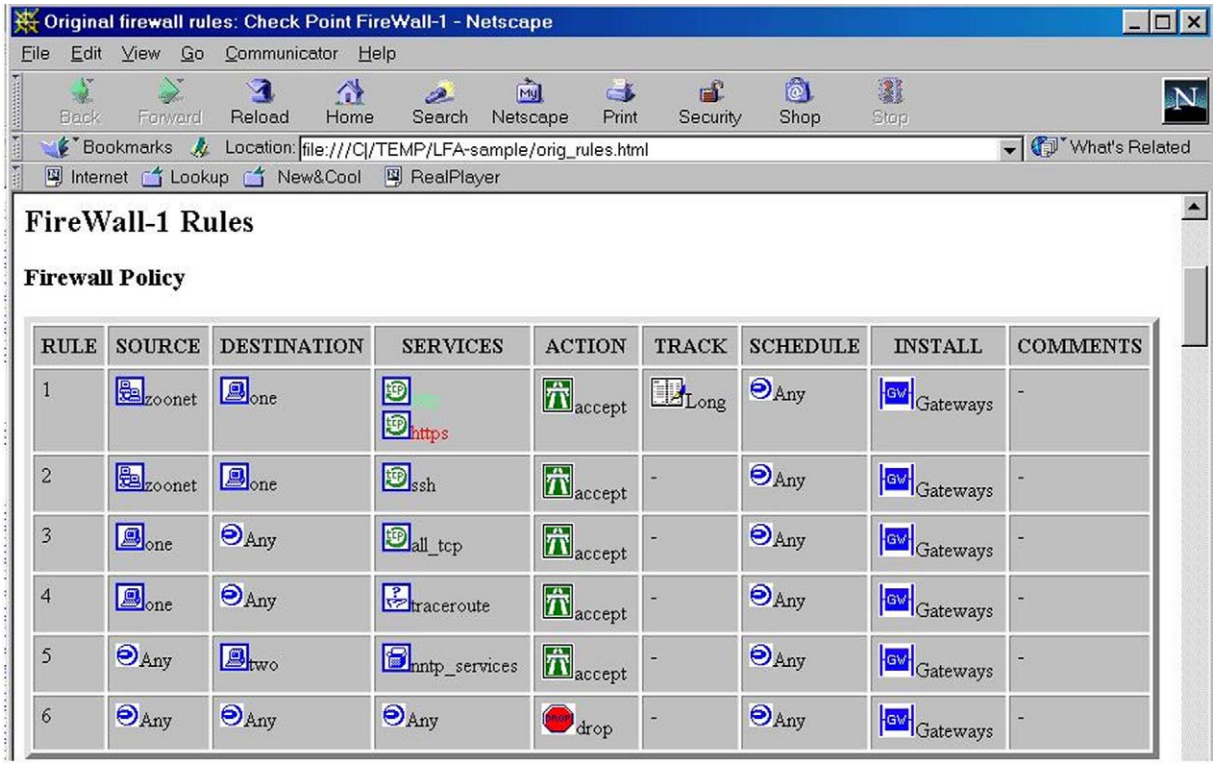

Figure 5 An HTML rendering of a rule-base within the Check Point GUI produced by the `fwrules` utility (Xu et al., 2000). Note the lack of any type of "direction" or "interface" column.

firewall configuration tools less confusing is a prerequisite for well-written firewall configurations.

Unfortunately, their solution, of hiding the confusing feature, is problematic. The fact that anti-spoofing is not controlled by the rules, but rather from a completely different area of the GUI, causes many users to overlook the anti-spoofing capability. Furthermore, the fact that the anti-spoofing rules are implicit also means that they are invisible to most users, so the rules' effects are not well understood. The consequence of Check Point's design choice is that administrators of FireWall-1 devices use direction-based filtering rarely (Wool, 2001b).

Conclusions

As we have seen, direction-based filtering is a useful tool to have in the firewall administrator's toolbox. Unfortunately, the direction-based filtering mechanisms currently offered by vendors are not very satisfactory. Most vendors force firewall administrators to deal with confusing low-level details, while Check Point essentially deprives users of this capability. However, vendors can do much better. This is demonstrated by the Rule Assignment and DIrection Setting (RADIS) algorithm, which was implemented within the *Firmato* prototype (Bartal et al., 1999).

An algorithmic solution

The *Firmato* prototype is a multi-vendor firewall rule compiler. Its input comprises a security policy, and a description of the network topology on which the policy is to be enforced. This information is written in *Firmato*'s Model Description Language (MDL). The compiler parses this input, transforms the data through several compilation phases, and produces firewall rules in the supported vendor's configuration languages.

The RADIS algorithm is the one-before-last phase of the compilation process—just before the rules are translated into some vendor-specific language. By the time that RADIS is activated, the security policy has already been converted from MDL into a rule-base. This is a list of rules, each matching on the standard fields: IP addresses, protocol, and port numbers. The directions for these rules are not specified. Thus, the input to RADIS is essentially equivalent in expressiveness to the rules produced by Check Point's GUI (recall section "Check Point FireWall-1").

The key to the power of the RADIS algorithm is that it is *routing-aware*. From the network topology description parts of the *Firmato* input, RADIS knows which IP addresses and subnets are located behind each firewall interface. As shown by Mayer et al. (2000) and Wool (2001a), this level of network topology information can be computed from the firewalls' routing tables. Given the network topology, the RADIS algorithm sets the direction field of each rule by casting the question "does a rule *r* have a chance of ever being relevant to a non-spoofed packet attempting to pass through interface *i* in direction *d*?" in graph-theoretic terms. This question can be efficiently answered using a graph algorithm, which searches for the path between the rule's source and destination across the firewall. The interested reader is referred to Bartal et al. (1999) for the full technical details of the RADIS algorithm.

Recommendations and future work

- One can easily envision the RADIS working in isolation, outside the *Firmato* framework. In such a system, the directionless rules would be written directly by the firewall administrator, and the network topology would be extracted by software from the firewall's routing table. Such a system would enjoy the best of both the worlds: on the one hand, it would abstract away the confusing details related to direction-based filtering, so the firewall administrator would not need to explicitly set the direction of every rule. On the other hand, it would automatically compute the correct direction for every rule. Hence, the rules would automatically activate anti-spoofing in all directions on the firewall, without requiring any explicit configuration. In particular, the internal network would always be protected from spoofed packets, and egress-filtering would be enabled. It could be interesting to add RADIS-like capabilities to existing firewall management systems.
- The usability of security tools and products is an area of research that deserves more attention. A few quantitative usability studies of security tools have been published, but much more can, and should be, done. After all, a tool or feature only improves an organization's security if it is *used*, and used *correctly* (see Schultz et al., 2001 for an overview of Human Factors in security tools). The picture emerging from the studies of systems like PGP (Whitten and Tygar, 1999), SSH (Kröger, 1999),

and public-key management (Jendricke and tom Markotten, 2000), is generally un-encouraging: Users find them difficult to use. Note, though, that these particular systems address the needs of very wide audiences. In contrast, firewalls are (supposed to be) managed by professional system administrators, who are expected to be trained. Nevertheless, we have seen that current mechanisms to configure direction-based filtering have significant usability problems, even for trained users. Furthermore, anecdotal evidence shows that corporate firewalls in general are not configured well (Wool, 2001b). Thus, firewall management may offer interesting opportunities for usability studies.

Acknowledgements

I thank Tom Limoncelli for many stimulating discussions and for showing me some of his system administration tricks. Michele Rizack brought RFC 2267 to my attention.

References

Bartal Y, Mayer A, Nissim K, Wool A. Firmato: a novel firewall management toolkit. In: Proceedings of the 20th IEEE Symposium on Security and Privacy, Oakland, CA; May 1999. p. 17—31. To appear in ACM Transactions on Computer Systems.

Bellovin SM. Security problems in the TCP/IP protocol suite. Comput Commun Rev 1989;19(2):32—48.

Chapman DW, Fox A. Cisco secure PIX firewalls. Cisco Press; 2001.

Chapman DB, Zwicky ED. Building Internet firewalls. O'Reilly & Associates, Inc.; 1995.

Edmead MT. Egress filtering. TISC Insight January 2002;4(1). [Electronic newsletter. Available from: http://www.tisc2001.com/newsletters/41.html].

Ferguson P, Senie D. Network ingress filtering: defeating denial of service attacks which employ IP source address spoofing. [Internet Engineering Task Force RFC 2827; May 2000]. Available from: http://www.ietf.org/rfc/rfc2827.txt. [An earlier draft is RFC 2267; January 1998].

Firewall Wizards. Electronic mailing list; 1997—2003. Available from: http://honor.icsalabs.com/mailman/listinfo/firewall-wizards.

Held G, Hundley K. Cisco access lists. McGraw-Hill; 1999.

Jendricke U, tom Markotten DG. Usability meets security—the identity-manager as your personal security assistant for the Internet. In: Proceedings of the 16th Annual Computer Security Applications Conference; December 2000.

Kröger VP. Security of user interfaces—a usability evaluation of F-Secure SSH. Technical Report, Department of Computer Science and Engineering, Helsinki University of Technology; 1999.

Limoncelli TA. Personal communication; 2001.

Mayer A, Wool A, Ziskind E. Fang: a firewall analysis engine. In: Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA. IEEE; May 2000, pp. 177—87.

Rubin A, Geer D, Ranum M. Web security sourcebook. Wiley Computer Publishing; 1997.

SANS Institute, Global Incident Analysis Center. Egress filtering, v0.2; February 2000. Available from: http://www.incidents.org/protect/egress.php.

Schultz EE, Proctor RW, Lien M-C, Salvendy G. Usability and security: an appraisal of usability issues in information security methods. Comput Secur 2001;20(7):620—34.

Welch-Abernathy DD. Essential Checkpoint Firewall-1: an installation, configuration, and troubleshooting guide. Addison-Wesley; 2002.

Whitten A, Tygar JD. Why Johnny can't encrypt: a usability evaluation of PGP 5.0. In: Proceedings of the Eighth USENIX Security Symposium. Usenix Association; 1999.

Wool A. Architecting the Lumeta firewall analyzer. In: 10th USENIX Security Symposium, Washington, D.C. USENIX; August 2001a, pp. 85—97.

Wool A. How not to configure your firewall: a field guide to common firewall misconfigurations. Invited talk, 15th USENIX Systems Administration Conference (LISA); December 2001b.

Wool A. Combating the perils of port 80 at the firewall. ;login: Mag USENIX & SAGE August 2002;27(4):44—5.

Xu W, O'Neal S, Schoonover J, Moser S, Lamar F, Grasboeck G. fwrules50; 2000. Available from: http://www.phoneboy.com/fw1/.

Avishai Wool received a B.Sc. (Cum Laude) in Mathematics and Computer Science from Tel Aviv University, Israel, in 1989. He received an M.Sc. and Ph.D. in Computer Science from the Weizmann Institute of Science, Israel, in 1992 and 1996, respectively. Dr. Wool then spent four years as a Member of Technical Staff at Bell Laboratories, Murray Hill, NJ, USA. In 2000 Dr. Wool co-founded Lumeta corporation, a startup company specializing in network security. Since 2002 Dr. Wool has been an Assistant Professor at the School of Electrical Engineering, Tel Aviv University, Israel.

Dr. Wool is the creator of the Lumeta Firewall Analyzer. He is an associate editor of the ACM Transactions on Information and System Security. He has served on the program committee of the leading IEEE and ACM conferences on computer and network security. He is a senior member of IEEE, and a member the ACM and USENIX. His research interests include firewall technology, network and wireless security, data communication networks, and distributed computing.