TOWARDS INTEGRATED MANAGEMENT OF FIREWALL APPLIANCES

Luís Pinto, Edmundo Monteiro, Paulo Simões
CISUC – Dep. Eng. Informática
University of Coimbra, Pólo II
3030-290 Coimbra, Portugal

ABSTRACT

In the last few years we witnessed a remarkable proliferation of network firewalls, driven by security concerns and supported by the emergence of low-cost firewall solutions. However, the subsequent relegation of firewall management to less trained technicians – or even to final users, in some cases – also increased the risk of security problems due to configuration and administration malpractices. This is a troublesome situation. It is impossible to hire highly trained technicians to manage every firewall appliance, and it is also unfeasible – in most cases – to simplify firewall administration to the point where it requires no expertise at all.

This problem is particularly serious in small to medium-scale networks: they already require professional-level expertise, but they are still too small to justify the most sophisticated and expensive solutions.

In this paper we present a platform for remote management of firewall appliances that is particularly targeted to such environments. When compared with traditional practices, this platform brings two key advantages: it reduces installation and maintenance costs and, at the same time, it improves the configuration and monitoring practices.

Key Words

Network & Security Management, Firewall Management

1. INTRODUCTION

As the number of network firewalls in each network continues to grow, in order to deal with external and internal threats, related *operations and maintenance* (O&M) costs also increase at an almost proportional rate.

This problem is well known by the industry, which responded in two different ways.

For the corporate market, whose large networks easily justify the investment in expensive commercial products offering integrated and centralized firewall management, vendors such as *Checkpoint* [1], *NetScreen* [2], *Cisco* [3] or *Clavister* [4] already offer some degree of support for remote configuration and monitoring operations.

Protecting simple low-segment SOHO networks is also relatively straightforward, since low-cost Cable or ADSL routers generally include basic VPN and/or filtering support, as well as simplified Web-based configuration interfaces. Despite the lack of advanced security features or sophisticated remote/centralized management tools, those devices reasonably satisfy the basic needs of such networks.

There are also several noteworthy research projects addressing this problem, most of them focusing on the application of policy-based networking (PBN [5]) concepts for firewall management [6-9]. PBN goes beyond simple remote firewall management: it envisions a network where enterprise-wide high-level policies are automatically and dynamically reflected in the configuration of network devices such as routers, firewalls, switching equipment and servers. However, despite the promising results achieved so far, a lot of research is still needed before this long-term vision becomes reality.

Nevertheless, even with a relative abundance of firewall management proposals, our own field experience – based on the management of several geographically dispersed networks belonging to different clients – showed us that, sometimes, currently available solutions simply do not fit.

This is especially true for the growing segment of middle sized networks that already require a considerable degree of flexibility and functionality but are still too small to justify corporate-grade solutions. A large part of these networks uses firewalls based on *open source* BSD or GNU/Linux distributions, complemented with filtering tools, like *iptables* [10], and freely available VPN software [11-12]. This approach might be motivated by economic reasons, but its recognized flexibility and robustness are also determinant.

The problem is that installing and configuring such firewalls is time-consuming and error-prone. In order to simplify this task, there are several firewall kits based on GNU/Linux, such as *LRP* [13], *SmoothWall* [14] and *Leaf* [15]. These kits propose lightweight GNU/Linux installations – removing all the unnecessary applications and services – and, in a few cases, enhanced configuration methods. However, they still lack the remote/centralized management tools needed to scale down management costs associated with the management of several geographically dispersed firewalls.

The platform we present in this paper is inspired by those *open source* firewall kits but goes one step further, addressing the problems of simplified distribution, centralized management and O&M cost reduction.

It is also inspired by some PBN assumptions, namely the need for more automated high-level configuration interfaces. However, it follows a much more pragmatic approach, sacrificing the support for heterogeneous firewall implementations (focusing instead in a single reference implementation) and coordinated configuration of interrelated network nodes (each firewall configuration is self-contained).

This paper proceeds as follows: Section 2 identifies the key requirements we took into account in the design of the platform. Section 3 presents the architecture of the platform, which is further detailed in Section 4 (firewall appliances) and Section 5 (Management Station). Section 6 presents the associated high-level configuration language, while Section 7 concludes the paper.

2. KEY REQUIREMENTS

The platform was designed to tackle with very specific problems we had in the outsourcing services we provide to a number of local customers. In the context of these services, we are responsible for the management of more than a dozen different networks, belonging to several distinct clients and totaling more than twenty firewalls. Each network comprises a few hundred desktops, as well as the typical suite of internet and intranet servers.

In order to keep competitive costs, time-consuming tasks such as user helpdesk and desktop maintenance are locally handled by technicians assigned to a single network/customer, while a centralized core of specialists in areas like security, databases and network engineering provides support to the whole set of managed networks. This set-up, combined with the error-prone configuration mechanisms of GNU/Linux firewalls, was a fertile ground for unnoticed installation problems and/or faulty configurations. Furthermore, even with strict operation guidelines, in order to normalize managed firewall and stimulate scale economies, costs were still almost proportional to the number of installed firewalls. We needed an alternative approach, focused on the following requirements:

- extensive remote management functionality, reducing as much as possible the need for local interventions.
- homogeneous managed firewall appliances. Even slight versioning differences from device to device – or minor configuration variations due to human factors – imply an increased effort to tackle with security threats, affecting scale economies.
- reduced probability of human errors. This implies minimizing the need for operator interventions (local or remote) and concentrating configuration in a single, easier-to-use, higher-level interface.
- centralized configuration storage, in order to simplify security audits and to accelerate emergency repairs in the case of hardware fault.
- monitoring mechanisms. These must include both on-line monitoring, to check the current health of managed devices, and off-line fine-grain analysis, based on the examination of the log files retrieved from each firewall.

3. ARCHITECTURE

From an architecture viewpoint, the platform is a classic remote management infrastructure (see Figure 1). It is centered on the Management Station, which controls the settings and execution of managed devices, stores management information in a relational database (including individual firewall configurations, historical log data, security and administrative settings, etc.) and provides a Web-based console to the systems operator. Section 5 discusses the key aspects of the management station, while Section 6 describes the user interface.

The managed appliances consist of classic GNU/Linux firewall kits, comprising *iptables*, DNS, NTP and DHCP servers and HTTP proxies. They are also enhanced with support for high-level firewall configuration syntaxes and the so-called Remote Management Layer, which controls the device configuration and handles communication with the Management Station. Section 4 provides a more detailed presentation of the managed firewalls.

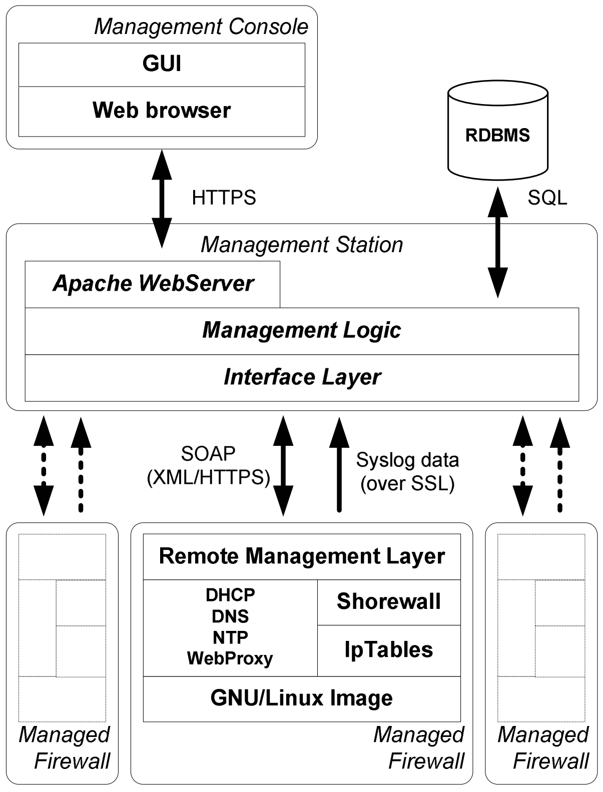


Figure 1: Platform Architecture

4. MANAGED FIREWALL APPLIANCES

Managed appliances are GNU/Linux-based firewalls that differ from other available firewall kits in three areas: the distribution and installation mechanism; the support for high-level firewall configuration syntaxes; and the Remote Management Layer that provides an interface with the centralized management station.

4.1. Normalized, CDROM-based Appliances

When managing a large number of firewalls, even small differences – such as different software versions or minor configuration changes – become annoying. They are difficult to keep track off, and often result in system malfunctions or security threats.

Furthermore, installing each firewall from scratch is a time-consuming and error-prone task, even when adopting rigid operation guidelines. It takes hours to install the base operating system, remove unnecessary components, install specific firewall services, configure them and perform the final tests before launching the firewall into production.

The need for normalization, reduced installation costs and reduced exposure to human-errors motivated us to follow a less traditional approach: to run firewalls directly from a bootable CDROM that contains the operating system, the firewalling services and the remote management modules.

In our case the CDROM was built by making a few adjustments to a regular Debian Linux distribution [16], in order to reduce its footprint and make it work directly from the CDROM. The very few node-specific settings (such as network addresses, firewalling rules, and security keys for communication with the server) were moved to a floppy disk-based partition, and system logs are sent directly to the Management Station.

This simple solution turns firewalls into diskless appliances running stable and normalized software, with no file system corruption problems or unnoticed installation faults. Furthermore, a number of security attacks become unfeasible (or not worthwhile) when the target firewall is based on a read-only file system.

Firewall installation also becomes simpler: the on-site technician just inserts the CDROM and a blank floppy disk and boots the system. He is then prompted to provide the basic data the firewall needs to contact the central server (a couple of network addresses and security keys). After that the firewall automatically contacts the server to retrieve its full configuration (filtering rules, monitoring settings, DHCP and DNS configuration, etc.) which should be already stored in the central database. This also accelerates hardware repairs, since configuration is rapidly restored after replacing the faulty equipment.

There are, however, a few downsides. The worst is the need for local interventions every time the software requires an upgrade. In our specific case the need to replace the firewall CDROM a few times a year is not relevant, since we already have an on-site technician. However, in other scenarios this might justify alternative approaches based on read-write file systems and dynamic software update mechanisms, such as PreOS management [17] or hot-swapping [18].

CDROM reliability is another potential drawback. However, although we found no studies comparing the reliability of hard drives and CDROMs, we believe they have more or less the same mechanical reliability. Furthermore, CDROM failures are simpler to detect and to solve (just replace the CDROM or the CDROM drive), while disk failures often result in unnoticed file system corruption and/or data loss. However, in some cases this could justify the investment in more robust solutions, such as flash memory.

4.2. Support for High-Level Configuration

Another distinctive feature of the managed firewalls is the support for a high-level configuration language. This language is based on an enhanced and extended version of the syntax used by *Shorewall* [19], a tool for high-level configuration of *iptables*.

This language is the platform's *lingua franca*. Its metaphors provide the basis for the firewall configuration GUI; it is used to store configurations in the central database; and it is used by the central station to upload configurations into the managed firewalls.

Managed firewalls feature a translation module that converts high-level configurations into low-level, service-specific configurations. In order to do this, the original *Shorewall* tool was extended to support other services, such as DNS, DHCP and NTP.

A more detailed discussion of the configuration language is presented in Section 6.

4.3. Remote Management Layer

The Remote Management Layer controls the firewall configuration on behalf of the Management Station.

The Remote Management Layer tries to check local configurations (stored in the floppy disk) each time the firewall boots. If there are no local configurations – or if there are differences between local and central configurations – the central settings are retrieved and applied. On the other hand, during normal execution configuration updates are usually triggered by the Management Station.

The Remote Management Layer also comprises two monitoring services. The first one provides basic health check funcionality, while the second simply transfers raw system logs to the central database, for latter analysis. This raw transfer may sound inefficient, but in most cases its effective impact does not justify any optimizations.

This layer consists of a number of Perl Scripts [20]. Except for the log transfer service, that uses an SSL tunnel, communication with the Management Station in based on an XML/SOAP interface [21] over HTTPS, implemented using SOAP::Lite [22].

5. MANAGEMENT STATION

The Management Station provides firewall configuration management services and monitoring services.

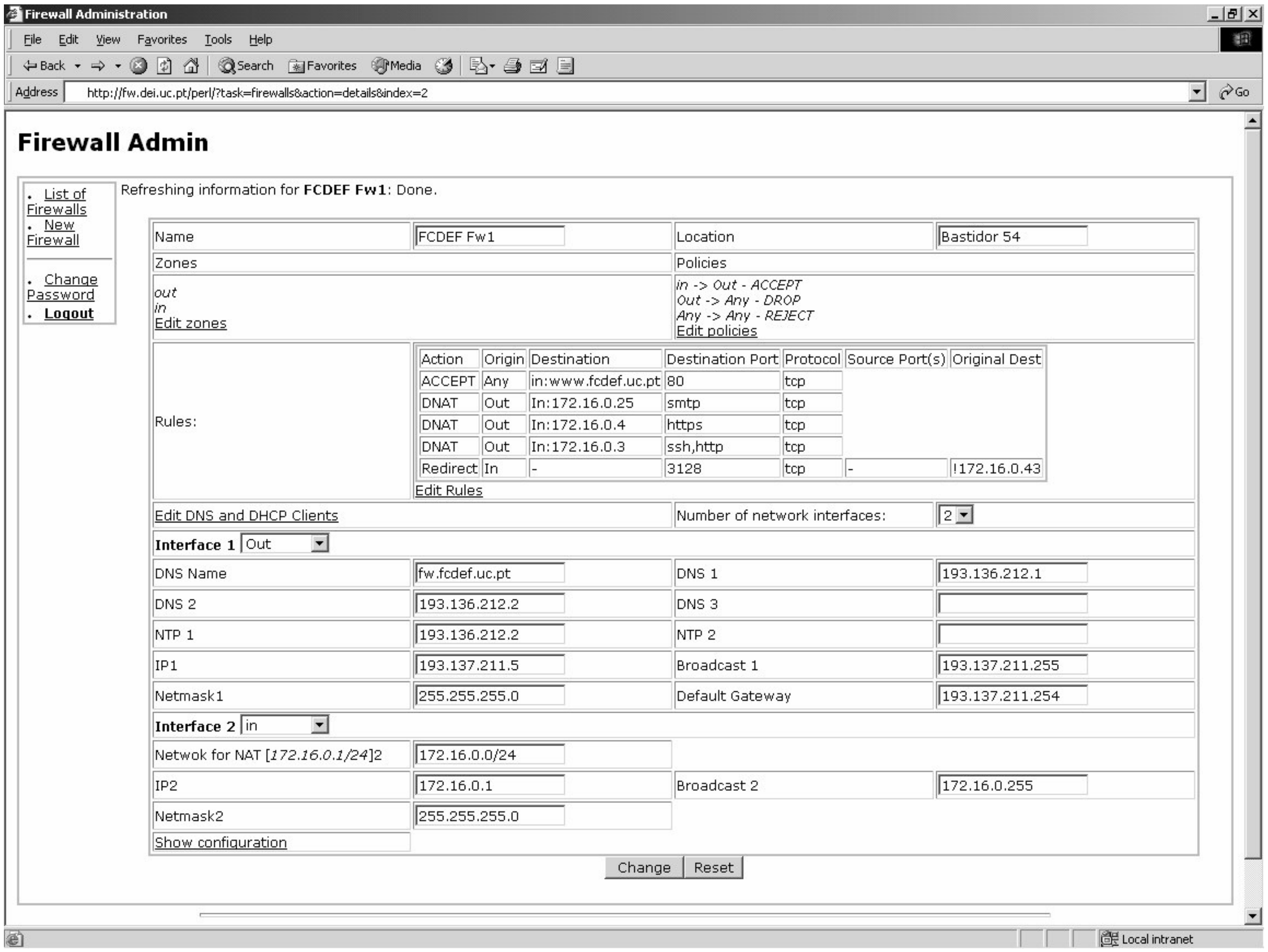
Installation of a new firewall begins with the registration of the firewall configuration, using a web-based interface. This configuration includes administrative data (e.g. physical location, hardware description and assigned technicians), IP settings, security keys, high-level filtering rules, and settings for services like DNS, DHCP and NTP. Configuration is then stored in the database, ready to be transferred to the firewall. As already mentioned, the local technician just needs to boot the firewall and provide a couple of settings (IP addresses and security keys) so that the firewall appliance can contact the management station to retrieve complete configuration settings. Later on, the same menus are used to review or update the firewall settings. Updated settings are registered in the database and, optionally, immediately transferred to the managed firewall. It is also possible to keep in the database several alternative configuration sets for the same firewall. Figure 2 shows the firewall configuration summary page.

On-line monitoring consists of a simple health monitor service that periodically checks the firewall status, looking for interface failures and other signs of potential problems. This is mostly a poll-based operation, even though managed firewalls may also trigger alarms in a few exceptional circumstances.

Off-line monitoring is no more than a centralized *syslog* service that transfers log data from the firewalls to the central database, for later analysis.

The Management Station also features a fine-grained administrative model. Administration is shared by a team of technicians with different roles. Field technicians are usually assigned to a single location (and a small number of firewall appliances) and, in most cases, just perform monitoring and hardware maintenance tasks. Remote managers, on the other way, are typically responsible for the configuration of a large number of devices belonging to different costumers. In order to reflect this model, permissions to access the Management Station are definable per user, per managed firewall and per task.

The Management Station was implemented using a GNU/Linux distribution. The Web interface is provided by an Apache WebServer [23], while the associated database uses MySQL [24]. Like the Remote Management Server, on the firewall side, the Management Station's logic was mostly developed using Perl scripts and SOAP::Lite.



http://fw.dei.uc.pt/perl/?task=firewalls&action=details&index=2

Firewall Admin

Refreshing information for **FCDEF Fw1**: Done.

Name	FCDEF Fw1	Location	Bastidor 54
Zones		Policies	
out *in* Edit zones		*in -> Out - ACCEPT* *Out -> Any - DROP* *Any -> Any - REJECT* Edit policies	

Rules:

Action	Origin	Destination	Destination Port	Protocol	Source Port(s)	Original Dest
ACCEPT	Any	in:www.fcdef.uc.pt	80	tcp		
DNAT	Out	In:172.16.0.25	smtp	tcp		
DNAT	Out	In:172.16.0.4	https	tcp		
DNAT	Out	In:172.16.0.3	ssh,http	tcp		
Redirect	In	-	3128	tcp	-	!172.16.0.43

Edit Rules

Edit DNS and DHCP Clients		Number of network interfaces:	2
Interface 1 Out			
DNS Name	fw.fcdef.uc.pt	DNS 1	193.136.212.1
DNS 2	193.136.212.2	DNS 3	
NTP 1	193.136.212.2	NTP 2	
IP1	193.137.211.5	Broadcast 1	193.137.211.255
Netmask1	255.255.255.0	Default Gateway	193.137.211.254
Interface 2 in			
Netwok for NAT [*172.16.0.1/24*]2	172.16.0.0/24		
IP2	172.16.0.1	Broadcast 2	172.16.0.255
Netmask2	255.255.255.0		
Show configuration			

Figure 2: Configuration Management Interface

6. CONFIGURATION LANGUAGE

Firewall configuration is an error-prone task for two main reasons. First, because settings are disperse across several different files, each with its own structure. Network interfaces are defined in one place, DNS configurations are written somewhere else, *iptables* in a third file and so on. It is difficult to keep coherence between so many disperse configurations. The second reason has to do with the counterintuitive syntax used by *iptables* to define filtering rules.

To solve these problems, we decided to concentrate the firewall configuration in a single location and to use higher-level metaphors to define the filtering rules.

Instead of defining a whole new language from scratch, we chose to extend *Shorewall* [19], an *open source* tool that converts high-level filtering rules into *iptables* native definitions. It is much simpler to use than plain *iptables* and, although it is just text-based, it was easy to extend with a Web-based GUI and an XML structure. Other services, such as DNS and DHCP, were also added to the original language. We evaluated several multiplatform alternatives [6-9,25] but, in the context of *iptables*, *Shorewall* had the best balance between functionality, flexibility and easiness of use.

For the filtering rules section, for instance, the following abstraction entities are defined (see Figure 3):

- **interfaces** map physical or logical network interfaces;
- **zones** define the different zones connected to the firewall, such as the Internet, a private network, a demilitarized zone, etc;
- **policies** define general policies regarding establishment of connections between existing zones. They can specify the traffic to be accepted, denied, or rejected;
- **rules** are exceptions to the default policies. They fine-tune the default policies, for instance enabling traffic from an non-trusted network to one specific host;

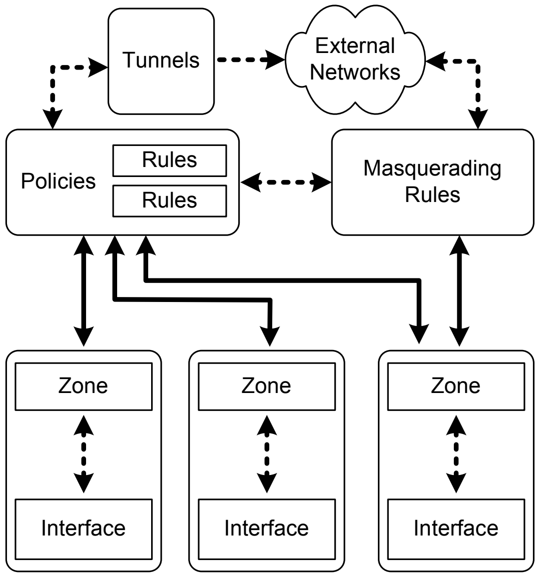


Figure 3: Abstractions for Rule Definition

- **masquerading Rules** define IP Masquerading and Source Network Address Translation (SNAT) rules, enabling one private network to access other zones using address translation;
- **tunnels** define IPSec, OpenVPN, PPTP and IPv6 to IPv4 tunnels.

The configuration language is the basis of communication between the Management Station and managed devices. The Management Station creates a Perl associative array with the firewall's network settings, filtering rules and the various service configurations (DNS, DHCP, etc.). That array is then translated into XML and sent over HTTPS. Figure 4 presents an excerpt of the SOAP conversation between the Management Station and the firewall.

The systems manager, however, does not deal with text configurations. Instead, he just deals with the Web-based interface (see Figure 2) that represents a structured view of the firewall's settings.

```
<conf xsi:type="namesp2:SOAPStruct">
   <zones xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="xsd:string[2]">
     <item xsi:type="xsd:string">
        In.Intranet.Internal Zone
     </item>
     <item xsi:type="xsd:string">
         Out.Internet.External
     </item>
   </zones>
   <dhcpd.conf xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="xsd:string[4]">
     <item xsi:type="xsd:string">
        option domain-name "fd.uc.pt";
     </item>
     <item xsi:type="xsd:string">
        option domain-name-servers "193.136.212.2";
     </item>
     <item xsi:type="xsd:string">
        option domain-name-servers "193.136.212.22";
     </item>
   </dhcpd.conf>
 </conf>
```

Figure 4: Excerpt of the SOAP Conversation

7. CONCLUSIONS

In this paper we presented a platform for centralized management of firewall appliances. This platform extends traditional *open source* firewall kits with high-level remote configuration and remote monitoring mechanisms.

Although sharing some assumptions with more ambitious approaches, such as PBN and high-end commercial products, is follows a much more pragmatic approach. Focusing in the reduction of O&M costs and the provision of easy-to-use interfaces, while sacrificing support for multiple firewall implementations and excessively abstract rule definition languages, it was possible to build a simple and cost-effective platform without much effort.

The current implementation is not a full-fledged product. It still lacks more sophisticated monitoring (such as dynamically defined threshold alarms and advanced log analysis), and VPN configuration also needs further integration work. Furthermore, the firewall distribution mechanism becomes questionable whenever there are no on-site technicians to periodically replace CDROMs.

Nevertheless, we are already using it to manage production firewalls, with a noticeable improvement of O&M costs and, more important, an enhanced control over each firewall's configuration correctness.

References

[1] Checkpoint Software Technologies, http://www.checkpoint.com/

[2] NetScreen Firewall, http://www.netscreen.com/products/firewall/

[3] Cisco Secure Policy Manager, http://www.cisco.com

[4] Clavister Firewall-SW Series, http://www.clavister.com/

[5] D. Kosiur, *Understanding Policy-Based Networking* (Willey, 2001).

[6] F. Caldeira & E. Monteiro, A Policy-Based Approach to Firewall Management, *Proc. of NetCon2002, Network Control and Engineering for QoS, Security and Mobility with focus on Policy-based Networking*, Kluwer Academic Publishers, Paris, 2002.

[7] V. Kurland & V. Zaliva, Firewall Builder Project, http://www.fwbuilder.org

[8] Filter Language Compiler, http://coombs.anu.edu.au/~avalon/

[9] S. Chudley & U. Ultes-Nitsche, An XML-based Approach to Modeling and Implementing Firewall Configurations, *Proc. of ISSA 2002 Information Security Conference*, South Africa, July 2002

[10] netfilter/iptables Homepage, www.netfilter.org/

[11] Linux FreeS/Wan Project, http://www.freeswan.org/

[12] OpenVPN Project, http://openvpn.sourceforge.net/

[13] Linux Router Project, http://www.linuxrouter.org/

[14] SmoothWall Project, http://www.smoothwall.org

[15] Leaf, Linux Embedded Appliance Firewall, http://leaf.sourceforge.net

[16] Debian Project, http://www.debian.org/

[17] T. Cruz & P. Simões, Enabling PreOS Desktop Management, *Proc. of the 8th IFIP/IEEE International Symposium on Integrated Network Management (IM'2003)*, Kluwer Academic Press, March 2003

[18] N. Feng, A. Gang, T. White and B. Pagurek, Dynamic Evolution of Network Management Software by Software Hot-Swapping, *Proc. of the 7th IFIP/IEEE International Symposium on Integrated Network Management (IM 2001)*, May 2001.

[19] Shorewall Project, http://www.shorewall.net/

[20] Perl, http://www.perl.com

[21] Simple Object Access Protocol (SOAP) 1.1, http://www.w3.org/TR/SOAP/

[22] SOAP::Lite, http://www.soaplite.com

[23] Apache Software Foundation, http://www.apache.org

[24] MySQL Database Server, http://www.mysql.com

[25] HLFL, High Level Firewall Language, http://www.hlfl.org/