An Integrated Firewall Management Platform

Luís Pinto, Edmundo Monteiro, Paulo Simões

CISUC — Dep. Eng. Informática
University of Coimbra, Pólo II
3030-290 Coimbra, Portugal

{lmpinto,edmundo,psimoes}@dei.uc.pt

ABSTRACT

Security concerns in computer networks resulted in a remarkable proliferation of network firewalls. However, the mere presence of more firewalls in the network is not a guarantee of more secure networks. The average expertise of today's firewall administrator is probably lower than it was a few years ago, resulting in an increased possibility of faulty installations, poor configuration and ineffective monitoring. This scenario calls for safer and more cost-efficient procedures for firewall deployment and maintenance.

This paper presents a framework we have developed for integrated management of firewall appliances. The advantages of such a framework are manifold: it reduces installation and maintenance costs (several firewalls become remotely installable and manageable from a single remote location); it simplifies firewall configuration (firewall policies are definable using an easy-to-use graphical interface, rather than the error-prone mechanisms required by most firewall implementations); it simplifies the resolution of hardware failures (since firewall configuration if kept in a remote database, it is sufficient to replace the faulty hardware and automatically upload the saved configuration into the new hardware); it enables the administration of geographically distributed firewall environments; and it provides an integrated monitoring facility that concentrates logs and health reports from several firewalls in a single database.

Keywords: Network Management, Security Management, Firewall Management

1. Introduction

Not long ago firewalls were very expensive systems, requiring expensive software and/or expensive security experts. However, in the last few years, firewalls seem to have become more accessible: low-cost network appliances already include basic firewall functionalities; GNU/Linux and FreeBSD provide the basis to build reliable firewalls with a little effort; and even Windows features a number of firewalling capabilities. Although the most complex firewalls still require a lot of money and effort, low cost solutions are apparently available to the average network manager.

However, this democratization brings itself a whole new set of problems. Installation and maintenance by non-experts translates into increased probability of insecure installations, poor configurations and ineffective monitoring. The average network manager faces two options: either he becomes a security expert, investing a significant effort in the maintenance of the infrastructure security; or he starts relaxing security maintenance, increasing the risks of dangerous incidents.

Undoubtedly, this fundamental dilemma – to decide how much should be invested in security – will always be present. Nevertheless, we feel it is possible to significantly reduce the setup and running costs of network firewalls while improving configuration and monitoring practices.

Motivated by our own field experience, with more than twenty firewalls distributed across a dozen different locations and eight different clients, we developed a very simple and pragmatic platform that lets us remotely install, reconfigure and monitor all these firewalls using a central database and a simple web-browser. This platform has significantly improved management practices while reducing both the probability of human error and the required human effort.

In this paper we present the platform. Section 2 describes the scenario that motivated us in the development of such a platform. Section 3 discusses related work, including the application of Policy-based Management to firewall configuration and remotely manageable commercial platforms. Section 4 presents the whole architecture of the platform, further detailed in Section 5 (Firewall Architecture) and Section 6 (Server Architecture). Section 7 discusses implementation issues, while Section 8 concludes the paper.

2. Motivation

As already mentioned, the key motivation to develop the platform was the need to reduce the running costs of the *outsourcing* service we provide to several local enterprises. This service consists of network and systems management for small and medium-scale networks. The typical network ranges within a few hundred desktops, as well as the typical suite of servers (Email, DNS, Web, User Accounts, Backoffice Systems, Database Servers and so on).

Our team includes both eclectic field managers and specialists in fields such as security, database management and network engineering. While field managers are dedicated to a single client, taking care of the most time consuming problems and providing first line local support, specialists are involved with several clients at once, remotely performing most of their interventions from a central location. This way we can provide a cost-effective service without compromising its global quality.

Although some of our managed networks do require complex and custom-fit firewalls, we found out that in general a reduced set of services and functionalities was enough. Most cases could indeed be solved using a simple firewall based on GNU/Linux and *iptables* [1], complemented with DNS, NTP (*Network Time Protocol*) and DHCP (*Dynamic Host Configuration Protocol*) services and Web Proxies.

Our first step to reduce installation and maintenance costs was obvious: we needed to "standardize" our firewalls, defining a set of rules and operations guidelines for their installation and maintenance. This step did reduce the effort and the problems associated with simple firewalls, but there were still many undetected mistakes and misconfigurations. The installation process and the definition of firewalling rules – using *iptables* – were still too error-prone. Besides, each firewall needed individual and manual attention for tasks such as software upgrades, configuration management, and monitoring. Costs were still almost proportional to the number of managed firewalls.

Therefore, the next step had to be more ambitious. We wanted to:

- enforce an uniform firewall installation, at least for the generic installations;
- reduce the need for local interventions to a minimum;
- keep each firewall configuration in a central location, in order to enhance both routine auditing procedures and emergency repairs;
- define the configuration for each firewall from a single location, preferably using an user-friendly interface able to reduce configuration errors;
- add automated monitoring functionality, both collecting the most important logs in a central location and being able to evaluate the firewalls health in real-time.

Following a rather pragmatic approach, we could reduce our target – more complex firewalls could always be managed in the old way – and sacrifice heterogeneity – we chose to build a custom firewall image with integrated management functionalities, rather than trying to add a management layer on the top of several different firewalls.

3. Related Work

The idea of providing a common higher-level interface to manage several firewalls at once is obviously not new.

There are several research projects addressing the usage of policy-based management (PBM [2]) in security management [3-6]. The underlying idea is to build systems able to enforce a set of enterprise-wide high-level security policies by means of dynamic and coordinated configuration of the key network devices (routers, switches, firewalls, etc.). Our platform shares some of assumptions of PBM, such as the convenience of more automated higher-level configuration interfaces. However, we follow a simpler approach. Since we target the management of a set of homogeneous firewall systems, it is not necessary to translate high-level policies into distinct configurations for heterogeneous nodes. Furthermore, our high-level policies are kept within the firewall-level: there are not abstract enterprise-wide policies difficult to translate into the coordinated configuration of several interdependent firewalls.

Some commercial firewalls also claim supporting PBM to some extent, as well as centralized management. CheckPoint/Firewall-1 [7] is the most well-know, but other products also claim some sort of centralized management [8-9].

Our platform is much more related with Open Source projects attempting to deliver simple-to-use firewall kits, such as *LRP* [10], *Smoothwall* [11] and *Leaf* [12]. What we are trying to add to those basic kits is automation, remote management capability, higher-level configuration mechanisms and the possibility of managing a large number of firewalls from a single platform/location.

4. General Architecture

The general architecture of the platform is presented in Figure 1.

Managed firewalls, built on top of a stripped-down Linux distribution, are based on the well-known *iptables*, complemented with a high-level rule definition tool (*shorewall* [13]) and several network services, such as DNS and DHCP servers and web proxies. On top there is a management layer that provides configuration and management services using a SOAP-based interface [14]. Normal *syslog* data is also sent to the central station, using an SSL tunnel. Section 5 discusses managed firewalls in more detail.

The management station includes the management logic, the relational database that stores firewall configurations and monitoring reports, and a Web-based user interface. Management stations dynamically update firewall configurations, periodically check firewall status and configurations, and store the relevant logging data produced by the firewalls. A more detailed description of the management station is presented in Section 6.

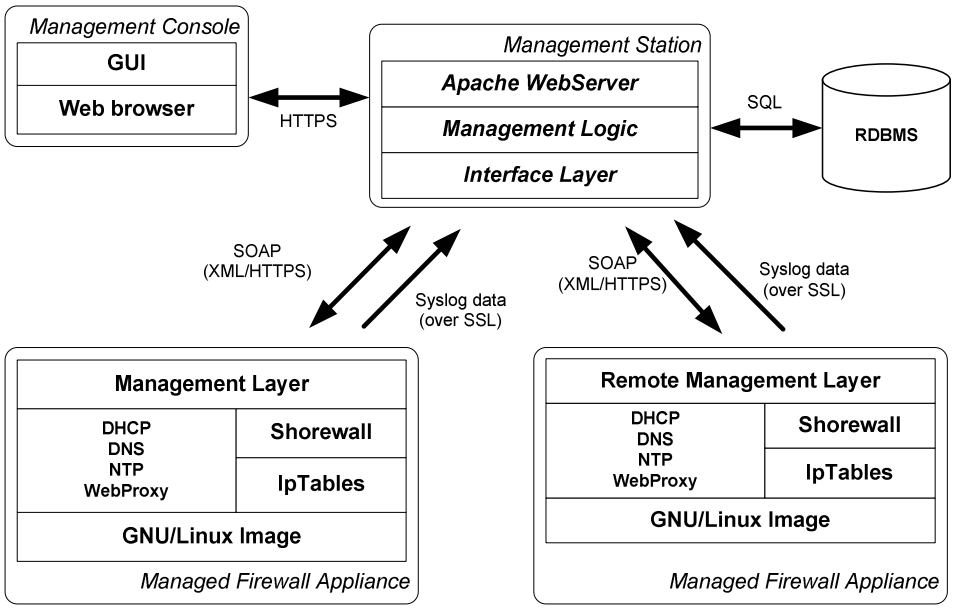


Figure 1: General Platform Architecture

5. Managed Firewall Appliances

As already mentioned, managed firewalls are basically an extension of the classic GNU/Linux firewall toolkit, with the inclusion of a remote management layer and a few configuration tweaks.

5.1 Firewall Distribution

One of the problems we wanted to solve was the small differences from firewall to firewall. These differences – for instance slightly different service releases – were difficult to track and often resulted, later on, into minor but annoying configuration problems. For this reason, a stable and uniform distribution is preferable, even at the cost of not always using the latest versions of non-critical components.

In order to enforce this uniformity, and to minimize the need for on-site system reinstallation due to hard-disk failure or file-system corruption, we took a somehow radical approach. Our firewall

appliances have no hard-disks. Instead, there is just a bootable CD distribution – with the OS and the necessary services – and a floppy disk that stores node-specific settings (network definitions, filtering rules, DNS and DHCP configuration, security keys, etc.).

Using read-only CDROM distributions, managed firewalls are guaranteed to be uniform and the risk of undetected change of critical files and applications (either accidentally or intentionally) is reduced. Furthermore, the installation of a new firewall appliance 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, which in most cases is already stored in the central database. Another interesting side-benefit stems from the fact that several security attacks are not successful (or not worthwhile, even when successful, since a simple reboot restores original settings) when used against a firewall based on a read-only file system.

On the other hand, new distribution releases imply local interventions on every managed firewall. In our case the need to update the CDROM 1 to 4 times per year is not an issue, since we already have on-site technicians able to spend 5 minutes in this straightforward task. CDROM reliability might also be questioned. Although we found no studies comparing the reliability of hard drives and CDROMs, we believe they have more or less the same mechanical reliability. However, 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. In the future we plan to support more reliable solutions – such as flash memory or RAM drives associated with preOS management solutions [15] – that will also make possible remote upgrade of the firewall distribution itself.

5.2 Firewall Management

The lower layers of the firewall are no different from other firewall kits: the GNU/Linux base (in our case Debian GNU/Linux [16] stripped-down and adapted to boot from a CDROM), *iptables*, DNS, NTP and DHCP servers, and a Web Proxy. The main differences are in the two upper layers: *Shorewall* and the Management Layer.

Shorewall is an opensource tool that enables the configuration of *iptables* using high-level rules. It is much simpler to use than plain *iptables* and, although it is just text-based, it was relatively easy to extend it with a Web-based GUI to further simplify rule-definition. For this reason, we decided to use an enhanced version of its syntax to store configurations at the server side and to download those configurations into the firewall. The *Shorewall* Layer is where those high-level rules are finally translated into low-level *iptables* rules. We also evaluated multiplatform alternatives [3-6,17] but, in the strict context of *iptables*, *shorewall* had the best balance between functionality, flexibility and easiness of use for the final user.

The Management Layer controls the firewall. When the firewall boots it looks for configuration files in the floppy disk. If the floppy is empty (or if its content is not coherent) the local technician is prompted to define the firewall's IP settings, the server's address and the security keys to communicate with the server. Then it contacts the central server to retrieve its full configuration and stores it in the floppy disk. Subsequent configuration changes are usually triggered by the server, but the firewall may also ask for a periodically configuration refresh (e.g. daily or when rebooting).

The Management Layer also features two monitoring services. The first provides the server with basic health data, either periodically (poll-based) or asynchronously (event-triggered). The second is no more than an SSL tunnel that sends raw *syslog* data to the server.

With the exception of the already mentioned SSL tunnel, every management service was developed using Perl Scripts and a SOAP interface over HTTPS [18-19].

6. Management Station

6.1 Firewall Administration

The management station supports the typical firewall administration operations: initial configuration, posterior configuration adjustments, health monitoring and raw logging analysis.

Installation of a new firewall begins with the registration of the firewall's configuration, using the web-based interface. This configuration includes the firewall's name and location, IP settings, security keys, high-level filtering rules, and DNS, DHCP and NTP definitions. These settings are stored in the central database, ready to be transferred to the firewall. As already mentioned, the local technician just needs to boot the firewall appliance and provide a couple of settings (IP addresses and security keys) so that the firewall appliance can start communication with the management station services.

Later on, the same menus are available 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, for instance to test new settings without losing the original configurations. Figure 2 shows the summary configuration page.

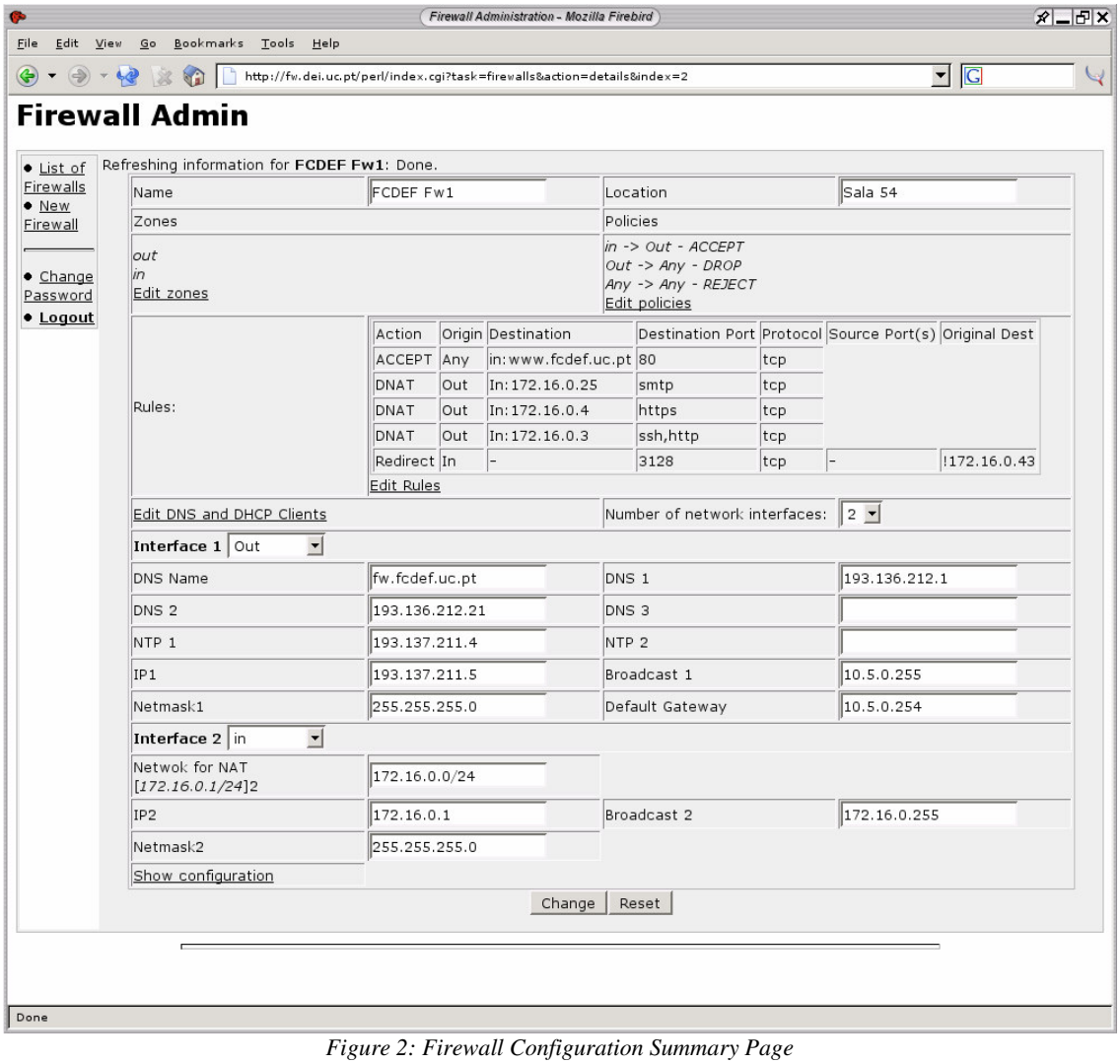


Figure 2: Firewall Configuration Summary Page

Filtering rules definition is the most important component of the firewall configuration, since it is typically the most technically challenging and error-prone configuration step. As already mentioned, our rule definition metaphor is based on the *shorewall* syntax, with its concepts of *zones*, *policies* and *rules*. There were a few minor changes in the syntax, mainly to enhance rule storage (in the database) and rule manipulation in the web-based user interface (the original *shorewall* only supports plain text configurations). Nevertheless, the metaphor is essentially the same.

This high-level syntax is also used to transfer filtering rules to the managed firewalls, were they are finally translated into low-level *iptables* rules. This means that in the future it could be possible to support other firewall implementations (if *shorewall* eventually gets ported to those implementations).

The platform's interface for rule definition is presented in Figure 3. This interface is still not appropriate to be used by end-users, but it does represent a significant improvement over plain *iptables* configuration files, even for experienced technicians.

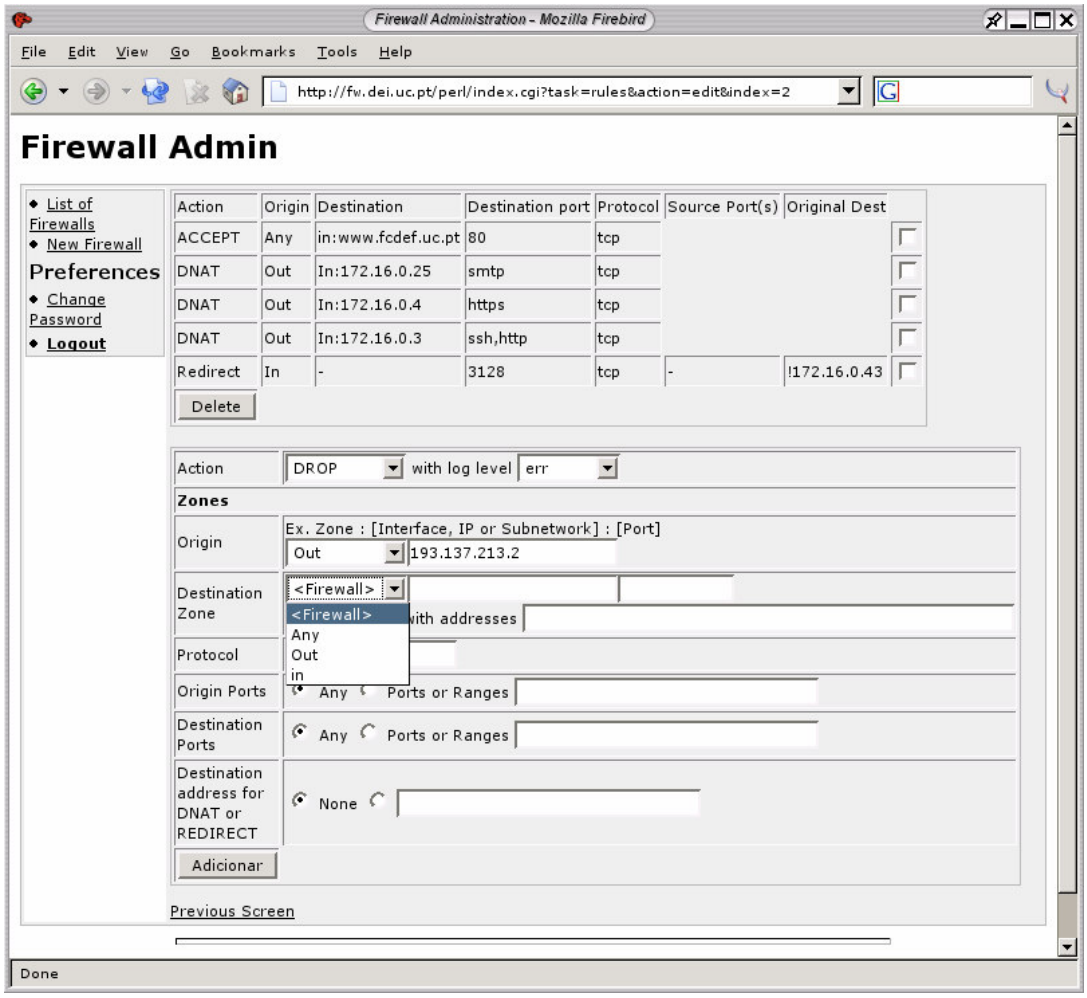


Figure 3: Rule Definition Interface

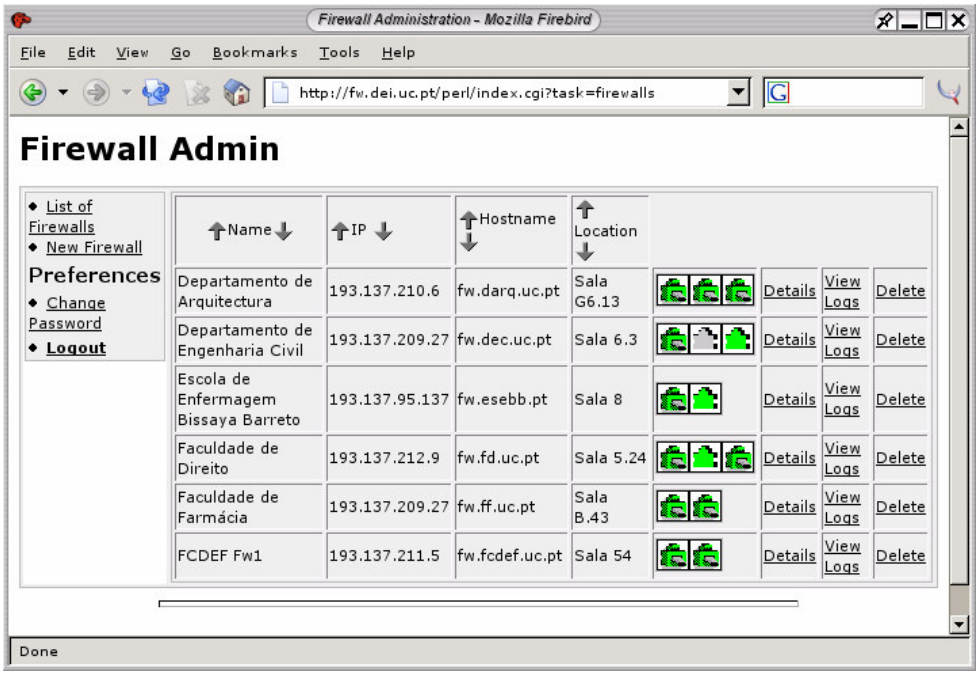
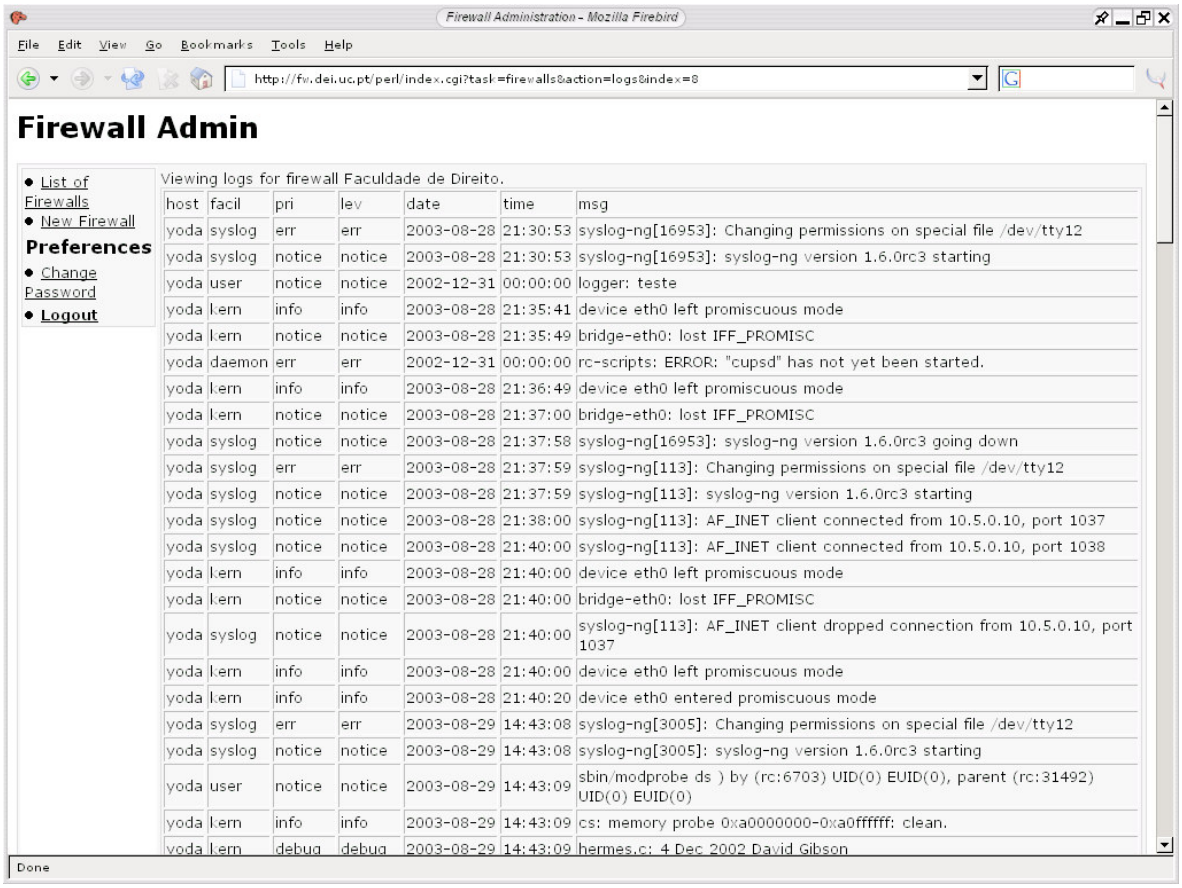


Figure 4: Firewall Monitoring Page

Firewall monitoring comprises two distinct services. The first service is a simple health monitor that periodically checks the firewall status, looking for interface failures, CPU overflow and other signs of potential problems. In Figure 4, that presents the summary page of this service, the icons associated with each network interface indicate their health status. Like the configuration services previously discussed, health monitoring uses a SOAP-based interface for communication between the management server and the firewall appliances. Right now monitoring is mostly poll-based, although a few hard-coded alarms may also trigger asynchronous notifications. In the future we intend to add support for more flexible adjustments.

The second monitoring feature is no more than a centralized log service. Firewall logs provided by *syslog* are not stored locally (amongst other reasons, because there would probably be no space to store them in the floppy disk). Instead, they are sent to management server using an SSL tunnel. Right now this server just stores those logs in order to show them to the systems manager, eventually with basic filtering and ordering (see Figure 5). In the future it would be interesting to study the possibility of adding automated log analysis, with alarm correlation between distinct firewalls.



host	facil	pri	lev	date	time	msg
yoda	syslog	err	err	2003-08-28	21:30:53	syslog-ng[16953]: Changing permissions on special file /dev/tty12
yoda	syslog	notice	notice	2003-08-28	21:30:53	syslog-ng[16953]: syslog-ng version 1.6.0rc3 starting
yoda	user	notice	notice	2002-12-31	00:00:00	logger: teste
yoda	kern	info	info	2003-08-28	21:35:41	device eth0 left promiscuous mode
yoda	kern	notice	notice	2003-08-28	21:35:49	bridge-eth0: lost IFF_PROMISC
yoda	daemon	err	err	2002-12-31	00:00:00	rc-scripts: ERROR: "cupsd" has not yet been started.
yoda	kern	info	info	2003-08-28	21:36:49	device eth0 left promiscuous mode
yoda	kern	notice	notice	2003-08-28	21:37:00	bridge-eth0: lost IFF_PROMISC
yoda	syslog	notice	notice	2003-08-28	21:37:58	syslog-ng[16953]: syslog-ng version 1.6.0rc3 going down
yoda	syslog	err	err	2003-08-28	21:37:59	syslog-ng[113]: Changing permissions on special file /dev/tty12
yoda	syslog	notice	notice	2003-08-28	21:37:59	syslog-ng[113]: syslog-ng version 1.6.0rc3 starting
yoda	syslog	notice	notice	2003-08-28	21:38:00	syslog-ng[113]: AF_INET client connected from 10.5.0.10, port 1037
yoda	syslog	notice	notice	2003-08-28	21:40:00	syslog-ng[113]: AF_INET client connected from 10.5.0.10, port 1038
yoda	kern	info	info	2003-08-28	21:40:00	device eth0 left promiscuous mode
yoda	kern	notice	notice	2003-08-28	21:40:00	bridge-eth0: lost IFF_PROMISC
yoda	syslog	notice	notice	2003-08-28	21:40:00	syslog-ng[113]: AF_INET client dropped connection from 10.5.0.10, port 1037
yoda	kern	info	info	2003-08-28	21:40:00	device eth0 left promiscuous mode
yoda	kern	info	info	2003-08-28	21:40:20	device eth0 entered promiscuous mode
yoda	syslog	err	err	2003-08-29	14:43:08	syslog-ng[3005]: Changing permissions on special file /dev/tty12
yoda	syslog	notice	notice	2003-08-29	14:43:08	syslog-ng[3005]: syslog-ng version 1.6.0rc3 starting
yoda	user	notice	notice	2003-08-29	14:43:09	sbin/modprobe ds) by (rc:6703) UID(0) EUID(0), parent (rc:31492) UID(0) EUID(0)
yoda	kern	info	info	2003-08-29	14:43:09	cs: memory probe 0xa0000000-0xa0ffffff: clean.
yoda	kern	debug	debug	2003-08-29	14:43:09	hermes.c: 4 Dec 2002 David Gibson

Figure 5: Log Analysis Interface

6.2 Administrative Model

In our case the platform is used by two distinct classes of users: field technicians assigned to a single location (and therefore co-managing a reduced number of firewall appliances) and remote managers responsible for a larger number of firewall appliances (not always belonging to the same costumers). Furthermore, in some cases field technicians are not supposed to change firewall settings, although they might need to check firewall status or trigger firewall reboots. For this reason the platform features a flexible administrative model, based on user identification, which allows permissions to be defined for each user, each firewall and each management operation. Nevertheless, this administrative model easily scales-down to more simplistic application scenarios with less-granular security permissions.

7. Implementation

The management station was implemented over GNU/Linux and the Apache WebServer [20]. The associated database uses MySQL [21], although porting to other database engines should be straightforward. Server/firewall communication is mostly based on SOAP over HTTPS, using SOAP::Lite [19] and Perl scripts. Log transfer is based on SSL tunnels.

The Firewall Appliance distribution is a Debian GNU/Linux distribution – specifically adapted to work directly from the read-only CD – configured to support the most popular PC hardware configurations. Special care was taken to support the most usual network interface cards.

We are now evaluating this platform in the real-world scenario described in Section 2. From an Operations & Maintenance perspective we are particularly interested in finding out how effective it is in reducing the running costs associated with firewall appliances. From the security viewpoint, we are evaluating the impact of high-level firewall rule-definition in the reduction of undetected configuration errors. The potential new security threats introduced by remote appliance management are also being assessed. The results obtained so far, although preliminary, were already satisfactory enough for us to decide to further deploy the platform.

8. Conclusions and Future Work

In this paper we presented a platform for centralized management of firewall appliances that reduces the management costs and simplifies firewall configuration.

This platform departs from other proposals – namely those inspired by the PBM field – due to its pragmatic approach. Focusing in the reduction of TCO costs and the provision of easy-to-use interfaces, while sacrificing support for multiple firewall implementations and excessively abstract rule definition languages (harder to translate into low-level equipment configurations), it was possible to build a simple but cost-effective platform without much effort. This platform is a natural evolution of traditional open source firewall kits: it provides the same basic security functionality, but it goes a step further in centralized management capability and high-level rule definition.

The current implementation is barely more than a proof-of-concept but, nevertheless, it is already successfully being used in the management of several appliances, with satisfactory results. In the future we intend to study more powerful solutions for firewall distribution (such as flash memory and/or PreOS management mechanisms [15]), to add more functionality to the firewall appliances (namely VPN support), and to improve monitoring services. Another area lacking more work is integrated configuration of several firewalls at once and integrated log analysis.

References

[1] Netfilter/Iptables Homepage, http://www.netfilter.org/
[2] D. Kosiur, “Understanding Policy-Based Networking”, Willey, 2001
[3] F. Caldeira, E. Monteiro, “A policy-based approach to firewall management”, Proc. of 'Net-Con'2002, Network Control and Engineering for QoS, Security and Mobility with focus on Policy-based Networking, Paris, 2002
[4] V. Kurland, V. Zaliva, Firewall Builder Project, http://www.fwbuilder.org
[5] Filter Language Compiler, http://coombs.anu.edu.au/~avalon/
[6] S. Chudley, U. Ultes-Nitsche, “An XML-based Approach to Modelling and Implementing Firewall Configurations”, in proceedings of ISSA 2002 Information Security Conference, South Africa, July 2002
[7] Checkpoint Software Technologies, http://www.checkpoint.com/
[8] NetScreen Firewall, http://www.netscreen.com/products/firewall/
[9] Clavister Firewall-SW Series, http://www.clavister.com/products/security_software_index.html
[10] Linux Router Project, http://www.linuxrouter.org/
[11] SmoothWall Project, SmoothWall Project, http://www.smoothwall.org
[12] Leaf, Linux Embedded Appliance Firewall, http://leaf.sourceforge.net
[13] Shorewall Project, http://www.shorewall.net/
[14] Simple Object Access Protocol (SOAP) 1.1, http://www.w3.org/TR/SOAP/
[15] T. Cruz, P. Simões, “Enabling PreOS Desktop Management”, Proc. of IM'2003 (IFIP/IEEE International Symposium on Integrated Network Management), Colorado Springs, USA, March/2003
[16] Debian Project, http://www.debian.org/
[17] HLFL, High Level Firewall Language, http://www.hlfl.org/
[18] Perl, http://www.perl.com
[19] SOAP::Lite, http://www.soaplite.com
[20] Apache Software Foundation, http://ww.apache.org
[21] MySQL Database Server, http://www.mysql.com