SPML: A Visual Approach for Modeling Firewall Configurations

[1]Kleber Manrique Trevisani and [2]Rogério Eduardo Garcia

[1] Universidade do Oeste Paulista – Faculdade de Informática de Presidente Prudente, Rua José Bongiovani, 700 – Bloco H, 19050-920, Presidente Prudente – SP, Brazil
[2] Universidade Estadual Paulista – Depto de Matemática, Estatística e Computação, Rua Roberto Simonsen, 305, 19060-900, Presidente Prudente – SP, Brazil
kleber@unoeste.br, rogerio@fct.unesp.br

Abstract. This paper describes a graphical notation for modeling security policy, currently focused on firewalls, named SPML. Using SPML, it is possible to specify graphically the security policy to be implemented by firewalls and to configure firewalls at high level, since the rules can be translated to native configuration. To present the approach proposed, we show how to translate SPML models into firewall configuration.

1 Introduction

Visual modeling is the graphic representation of objects and systems of interest using graphical languages, as widely known. Modeling security policies using a graphical notation, instead a textual or a mathematical representation, make easier to analyze and to understand security policies. By using a formalized graphical notation to establish a security policy, it is possible to discuss better ways to improve it, to solve problems related to it and allow using automated tools to validate it [1].

In this paper we present SPML (Security Policy Modeling Language), which the main goal is to represent security policies into a formal graphical notation. To develop SPML we have considered related works, such as [2] and [3]. To make SPML models more useful, an important requirement is that they can be translated into native configurations of products that implement security policies. This requirement allows computing systems administrators deploying the security policy from graphical models, without any additional effort. Considering that security policies have several issues involved, initially we decided to focus on modeling some firewall functions – packet filtering and network address translation (NAT) –, since it has an important role in a security policy.

To present the notation proposed, this paper is organized as follow: the notation proposed and its components is described in the Section 2; examples of models using the proposed notation are presented in Section 3; the translation process (from SPML to native rules) is described in Section 4; finally, in Section 5 are presented the conclusions, current development and future works.

2 SPML Notation

The SPML notation has a set of graphical elements named *components*. In the current stage we are focusing on firewall configuration, which is represented using two types of diagrams: *Translation Diagrams* and *Filtering Diagrams*. Both types of diagrams have interconnected components to represent configuration and features of firewalls. For each component were defined attributes to describe its features [4]. It is important to note that some attributes are not visible in the graphical presentation, but are incorporated to components to allow translating SPML models into native firewalls configuration, as described in Section 4. Some attributes were omitted in this paper to make easier the comprehension of SPML.

The components are classified in *Firewall Components*, *Filtering Components*, *Translation Components* and *External Entities*. Filtering components are instantiated in *Filtering Diagrams* and Translation components are instantiated in *Translation Diagrams*. *Firewall components* and *External Entities* are allowed in both diagrams. In the following subsections are presented the SPML components and their attributes.

2.1 Firewall Components

The features and configuration of firewalls are represented in SPML by two components: *Firewall* and *Interface*. The first one – *Firewall* – is used to represent any hardware or software that executes function of firewall, and is depicted by a circle with the firewall identification, as shown in Figure 1. A firewall component has the following attributes: Hostname; Domain and Firewall Type (e.g. Linux Netfilter or OpenBSD Packet Filter).

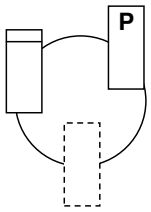


Fig. 1. A firewall with three network interfaces.

A firewall has one, or many, network interface cards, allowing other components to connect with. SPML does not allow any component to connect directly to a firewall. The connection is made by another component named *Interface*. According to the visual syntactic adopted, *Interface* is a rectangle overlapped in a firewall component, as shown in Figure 1. The attributes of interface component are: IP; NetMask; Name – string identifying the *Interface* component; Device – name of device that represents the interface (for example, eth0); Default Policy and Antispoof. The attribute *Default Policy* identifies the policy adopted for the interface: block or pass. Interfaces marked with ***P***, as depicted in Figure 1, use the pass policy, otherwise the block policy is assumed by default. Block policy denotes that all packets are blocked, unless explicitly allowed. The pass policy works contrary. A Flow (Section 2.2) connected to an interface with block policy, denotes permission, and connected to

an interface with pass policy, denotes denial. The *Antispoof* attribute identifies if the interface must implement protection against IP spoofing attacks – interfaces with spoofing protection disabled has a line segment on the top, as depicted in Figure 1. Virtual interfaces, which are used to associate more than a IP address to an physical network interface, is drawn using dashed lines, as depicted in Figure 1.

2.2 Filtering Components

Filtering components, as suggested, define the rules for packets filtering in a SPML modeling. These components represent data flows among an *External Entity* (Section 2.4) and an *Interface* component, and they may be used only in *Filtering Diagrams*. Depending on the direction, the flows can be considered *Incoming flow* or *Outgoing flow*, described next.

(a) Incoming flow with FID = 1 (b) Outgoing flow with the Source FIDs

Fig. 2. Examples of Flows.

All flows, *Incoming and Outgoing*, has the following attributes: Name; Address Family – IPv4 or IPv6; Protocol – its values can be Any, TCP, UDP or ICMP; Source Ports; Destination Ports; Stateful – to indicate if the firewall should keep the state of connection for that flow (flows with the attribute Stateful=no are depicted by a doubled pointed arrow, presented in Figure 2a; Log – indicates if the firewall should log packets that match to the flow; Policy – it can assume the values *Drop* or *Return*. If the policy is *Return*, hosts that have packets blocked will be warned. *Drop* policy does not warn the source of packet about blocking.

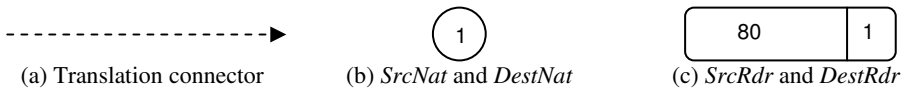
An **Incoming flow** is used to describe flows arriving on firewall. To be considered an incoming Flow, it must connect an *External Entity* (see Section 2.4) to an *Interface* component, respectively a source and a destination (arrowhead extremity connected to an Interface). In addition to the flow attributes, *Incoming Flows* has the FID (Flow ID) attribute. It is an integer number, as shown in Figure 2a, that identifies the flow and allows assigning an Incoming Flow to one or more Outgoing Flows. A firewall can not have more than one flow with the same FID (it must be unique). Flow assignment is important to the translation of SPML modeling into native configuration of firewalls, as described in Section 4.

An **Outgoing flow** is used to describe flows departing from firewall. To be considered as outgoing, the flow must be connected to an Interface component and the arrowhead must point to an *External Entity*. Instead of the FID, an outgoing flow has the Source FIDs attribute to define an assignment between incoming flow and outgoing flow. It is possible to assign more than one incoming flow to an outgoing flow. In this case, the Source FIDs are separated by comma. In Figure 2b is shown the outgoing flow *web_out* assigned with two incoming flows with FID 1 and 2. It is important do note that the incoming and outgoing flows must have the same protocol (protocol attribute with same value).

2.3 Translation Components

Translation components are divided into NAT (Network Address Translation) and RDR (Service Redirecting) components. In *Translation Diagrams* components are connected by dashed arrows, as depicted in Figure 3a.

NAT components are used to represent network address translation of N:1[1] and 1:1[2] [5] [6]. There are two NAT components: *SrcNat* and *DestNat*, both represented by a circle (see Figure 3b). *SrcNat* must be connected to one or more *External Entities* to describe the source of packets. It has the following attributes: Address Family, Protocol, Source Ports, Destination Ports and NID (Nat ID) – an integer number used not only to identify the component, but also to indicate the assignment between *SrcNat* and *DestNat*. *DestNat* must be connected to an Interface component and one or more *External Entities*. It component describes the network interface where NAT is done and the destination address which translation is allowed. A *DestNat* can not be represented without a *SrcNat* assigned, and vice-versa. *DestNat* has only one the NID attribute.



(a) Translation connector (b) *SrcNat* and *DestNat* (c) *SrcRdr* and *DestRdr*

Fig. 3. Translation Components.

RDR components, used to represent service redirecting, are useful to become available services provided by hosts with private IP, protected by firewall. Similar to NAT components, there are *SrcRdr* and *DestRdr* components, that are represented by a rounded rectangle (see Figure 3c). *SrcRdr* must be connected to one or more *External Entities*, to define the packet source address, and connected to an *Interface*, to define the packet destination address. *SrcRdr* has as attributes: Address Family, Protocol, Source Ports, Destination Ports and Destination RID (Redirection Identification) – an integer number used to identify the component. RID is also used to assign a *SrcRdr* to a *DestRdr* (using the same RID to a *DestRdr*). This assignment defines where packets are redirected – a *SrcRdr* can not be used without its *DestRdr* corresponding, and vice-versa. Visually, both the Destination Ports (number *80* in Figure 3c) and RID (number *1* in Figure 3c) are presented in the inner region of rounded rectangle used to represent each RDRcomponent.

DestRdr must be connected to only one External Entity, used as destination to the redirected packets. The address of the *External Entity*, connected to *DestRdr* component, substitutes the original destination address of the packet. The *DestRdr* attributes are: Port – represented by a unique integer number (number *80* in Figure 3c), that will replace original port of the packet, and RID.

[1] *N* addresses are translated to one address.
[2] *1* (one) address is translated to another address.

2.4 External Entities

External Entities define source and destination addresses in translation and filtering diagrams. When these components are used in *Translation Diagrams*, they can be connected to *SrcNat*, *DestNat*, *SrcRdr* and *DestRdr*. Used in *Filtering Diagrams*, they can be connected only to *Interface* component. *External Entities* can not be connected each other. These components are:

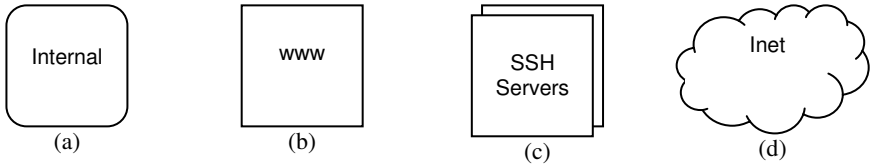


Fig. 4. External entities: (a) Network named Internal; (b) Host named www; (c) Hostlist of SSH Servers; (d) Internet Representation.

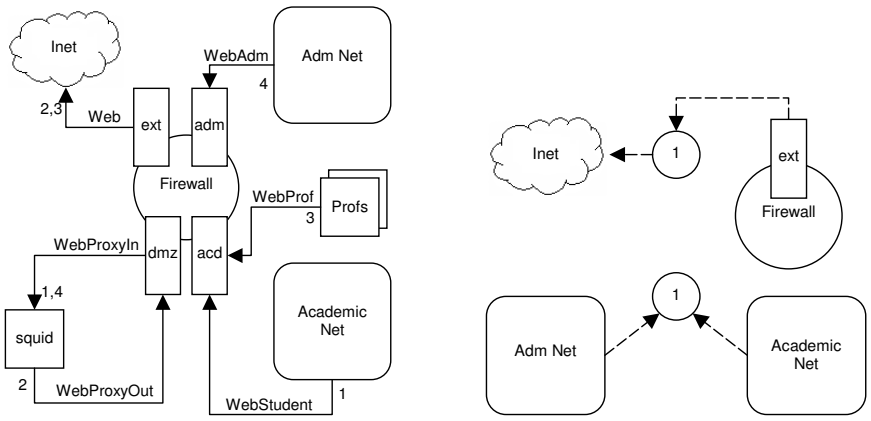
- Network: represents a TCP/IP network, depicted by a square with rounded corner, as showed in Figure 4a. The attributes of this component are: Name – to identify uniquely a network, Network Address and Netmask.
- Host: This component describes any device capable to receive an IP address, and involved with security policy modeled. It is represented by a square, as depicted in Figure 4b. Its attributes are: Hostname, Domain, IP Address and Netmask.
- Hostlist: Defines a group of hosts involved in security policy modeled. It is represented by two squares overlapped, as depicted in Figure 4c. It attributes are: Name – identifies set of hosts uniquely (e.g. SSH Servers) and a list of hosts (e.g. sshserver1, sshserver2, sshserver3).
- Inet: Used to represent the Internet or other network with unknown IP address. This component is graphically represented by a cloud, as depicted in Figure 4d. And it has only one attribute: Name (a symbolic name).

3 SPML Modeling

The SPML notation has a set of formal rules to interconnect its components described in [4]. In this section are presented some diagrams representing a firewall configuration to deal with web access from a University (with academic network and administrative network) connected to Internet, considered as case of study.

According to the policy adopted by the University, academic network must be isolated from administrative network to prevent students to have access to data and resources forbidden to them. Additionally, all users have Internet access limited (by content, for example), except the professors. Figure 5a shows a filtering diagram for that configuration, considering only the outgoing rules. The diagram has a firewall with four network interfaces: one of them connected to Internet; two connected to academic and administrative networks; the fourth interface is connected to demilitarized zone (DMZ). In such schema, packets from academic or administrative

network must pass through *squid* host (a HTTP proxy) before reach the Internet, if allowed. Note that the *WebStudent* and *WebAdm* flows, with FID 1 and FID 4 respectively, are assigned to *WebProxyIn* flow. The *squid* host has permission to send packets with any destination through *dmz* interface: the *WebProxyOut* and *Web flows* are assigned by FID 2. Packets from professors' hosts can pass directly to Internet, without a proxy (following the FID 3).



(a) Filtering Diagram

(b) Translations diagram using NAT components

Fig. 5. Example of SPML Diagrams.

Hosts from academic and administrative network have private IP addresses. Therefore, it is need to change the source IP address of packets from such hosts before send them to Internet. The translation diagram, exposed in Figure 5b, represents the NAT configuration to deal with such problem. In that example, the translation is performed on *ext* interface, so packets from academic and administrative network have their source IP addresses changed to *ext* interface IP address.

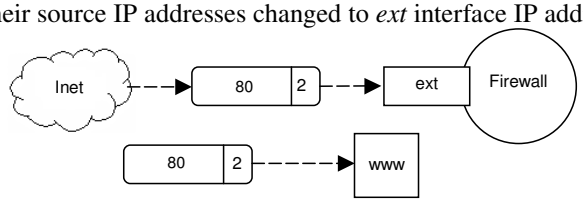


Fig. 6. Modeling of Service Redirection.

Since *www* host is a Web server and has a private IP, it can not be reached from the Internet. This problem is treated using the RDR components with RID 2, shown in Figure 6, where the packets from the Internet, arriving through *ext* interface, with destination port 80, are redirected to *www* host. Similar is made to *Academic Net* and *squid*. Note that the filtering rules allowing redirection to *www* host were not presented in the filtering diagram, shown in Figure 5a, but they could be modeled in such diagram or in another filtering diagram.

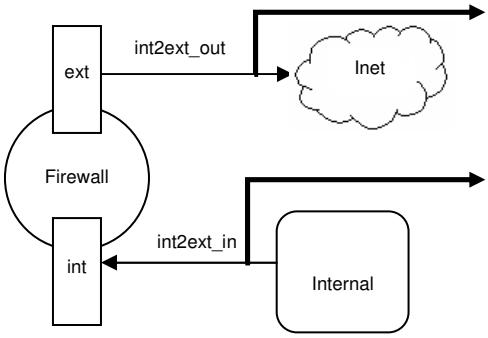
4 Translation to Native Rules

Only graphical modeling might not be interesting for network administrator, since it still require writing native rules. To support the definition (by creating or modifying) and the translation of SPML models into native configurations was developed a tool named SOH (Security On Hands), written in Java. Such tool is organized in modules: edition module; translation module and instantiation module. The first one, as suggested, supports edition of SPML diagrams [7]. The second one translates SPML diagrams into XML documents [8], and the third one translates the XML documents to native firewall configuration [7].

The use of XML document aims to facilitate exchange SOH data structures. XML documents are used as abstract representation of firewall configuration, what allows the development of new translators, from SPML to native firewall configuration, without knowledge about how SOH represents SPML diagrams internally. The XML structure used is named FwXML, and was defined by [4] and [9]. This approach allows writing specific translators to each firewall product, instantiating the abstract rules from XML documents to native rules.

4.1 SPML to FwXML

The first step for translating SPML to FwXML focuses on firewall features and external entities components. The next step focuses on translation diagrams, dealing with NAT and RDR components. The last step focuses on the filtering rules. Figure 7 shows a filtering diagram and its respective FwXML representation.




```
<spmlmodel>
 <ruleset>
 <flow name="$int2ext_out">
   <rule action="pass" direction="out"
        interface="$ext" proto="tcp">
     <parameters log="no" af="inet"
          keepstate="yes"/>
     <from>$internal</from>
     <to dst_port="80">$inet</to>
   </rule>
 </flow>
 <flow name="$int2ext_in">
   <rule action="pass" direction="in"
        interface="$int" proto="tcp">
     <parameters log="no" af="inet"
          keepstate="yes"/>
     <from>$internal</from>
     <to dst_port="80">$inet</to>
   </rule>
 </flow>
 </ruleset>
</spmlmodel>
```

Fig. 7. FwXML of filtering diagram.

4.2 Translating FwXML to Native Configuration

As mentioned, the translation of FwXML to native configuration must be written specifically for each firewall product: there are SOH modules that generate native configuration to OpenBSD Packet Filter [10] and Linux Netfilter [11].

```
<spmlmodel>
  <ruleset>
    <flow name="$int2ext_out">
      <rule action="pass" direction="out"
          interface="$ext" proto="tcp">
        <parameters log="no" af="inet"
            keepstate="yes"/>
        <from>$internal</from>
        <to dst_port="80">$inet</to>
      </rule>
    </flow>
    <flow name="$int2ext_in">
      <rule action="pass" direction="in"
          interface="$int" proto="tcp">
        <parameters log="no" af="inet"
            keepstate="yes"/>
        <from>$internal</from>
        <to dst_port="80">$inet</to>
      </rule>
    </flow>
  </ruleset>
</spmlmodel>
```

(a) FwXML

```
# int2ext_out
pass out on $ext inet proto tcp
  from $extip to $inet port 80
  keep state

#int2ext_in
pass in on $int inet proto tcp
  from $internal to $inet port 80
  keep state
```

(b) OpenBSD Packet Filter native rules

Fig. 8. Translation from FwXML to OpenBSD Packet Filter native rules.

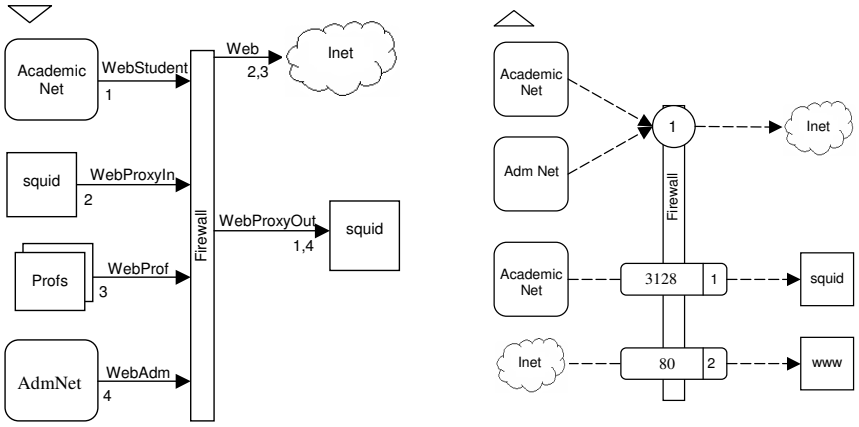
An example of filtering rules translation to Open BSD Packet Filter is presented in Figure 8. It is important to note that the translator must determine the source and the destination addresses, depending on firewall product. Considering the Packet Filter, NAT rules are performed before filtering, so filtering rules must use the network interface address as source address, for packets that have been applied NAT. Figure 8b shows *$extip* (in bold) chosen as source address, instead $internal, which is the original address before NAT application to the packet. The XML representing the NAT rule was omitted due to page limit.

4.3 Dealing with ordering rules problems

In a firewall, a set of rules is analyzed in sequential order. SPML models previously presented do not provide a way for ordering of filtering and NAT rules. Such order is used by firewalls to establish which action (block or pass) must be done when they treat packets. Usually, the firewalls adopt the following approaches:

1. The packets are compared with all rules before take any action. The last rule that matches with the packet defines the action to be done. The filtering of OpenBSD firewall works like this, unless for those rules qualified with reserved word quick [10];
2. The firewall compares the packets with each rule. The first rule that matches with the packet defines which action is taken, and the next rules are ignored. Firewall Aker [12] uses this approach for filtering and NAT. OpenBSD uses it for NAT.

For modeling filtering and NAT rules order, another graphical visualization was developed. The solution adopted represents the firewall in a cross section view, named *Side View*, which is another way to see components already modeled in filtering and NAT diagrams. For example, in Figure 9a is presented a *Side View* of the filtering diagram shown in Figure 5a, and Figure 9b presents a *Side View* of the translation diagrams shown in the figures 5b and 6. By moving the components in such views to up/down establishes the order of rules application: the translation process starts with the component at top, and follows going down. In this view, the Interface components are omitted, since they do not contribute for this modeling task, and their presentation would became the view polluted.



(a) Side View of filtering rules (b) Side view of translation rules

Fig. 9. Modeling ordering rules with Side View.

Additionally, in this view is possible to select the approach for analyzing rules. A triangle, at top left position, indicates such approach. When the triangle is pointing to down, it signalizes that firewall must analyze all rules before any action about the packet (approach 1). Otherwise, the triangle indicates the firewall must apply the first matched rule (approach 2). If the triangle was pointed to up in Figure 9a, the professors' host would not have direct access to Internet, because the matched flow would be WebStudent (FID = 1), considering that professor's hosts are members of academic network too.

5 Conclusions and Further Works

We proposed a graphical notation for modeling security policies, named SPML, focusing on firewall configuration. The visualization of SPML diagrams allows understanding the actions performed by firewalls, what aiding decision making about maintenance of security policy (for adding, removing or changing of rules). SPML provides a friendlier, comparing to text based commands or proprietary management

systems. Also, it is possible to define a firewall policy in an independent way, without take into account technical details of specific products.

Modeling SPML aided by automated tools, such as SOH, allow deploying the firewall configuration without any additional effort, since the tool can translate – by using the FwXML mapping – and send to it. The current version of SOH supports translation to OpenBSD Packet Filter and Linux Netfilter firewalls. A pilot study was conducted by [9] to evaluate not only the use of visual approach proposed in SPML, but also to show the independency between SPML models and firewall products, validating the FwXML translation. In such study, the same SPML model was translated to these two firewalls products and penetrations tests showed that the behavior modeled in SPML was the same observed in both firewall products.

The initial evaluation has shown advantages on firewall configurations, what has motivated further investigations on extending SPML to deal with other security policies issues, as initially intended. One might, for example, extend the SPML to deal with VPN, proxies, IDS, access control, bandwidth control, load balance with NAT, so on. For that, new diagrams must be defined using or extending the already defined components, or creating new components. Nowadays, the most works in progress is related to enhance SOH tool by adding new functionalities, as new modules to translate FwXML into native configurations. Additionally, automated tools can be developed to validate firewall configurations according to security policies specified in SPML models [1].

References

[1] Adel El-Atawy et al.: An Automated Framework for Validating Firewall Policy Enforcement, POLICY '07: Proceedings of the Eighth IEEE International Workshop on Policies for Distributed Systems and Networks, 151-160, 2007.

[2] Bartal, Y. et Al. Firmato: A novel firewall management toolkit. In: ACM Transactions on Computer Systems. v. 22, n. 4, p.381-420, November 2004.

[3] L. Yuan, H. Chen, J. Mai, C.-N. Chuah, Z. Su, and P. Mohapatra. Fireman: a toolkit for firewall modeling and analysis. IEEE Symposium on Security and Privacy (2006).

[4] Khouri, K. T. and Trevisani, K. M. Definition of a subset of SPML language to modeling firewalls (in Portuguese). Technical Report, FIPP/UNOESTE (2005).

[5] Egevang, K. and Francis, P.: Network Address Translation (NAT) - Internet Request for Comments 1631 (1994).

[6] Tsirtsis, G. and Srisuresh, P.: Network Address Translation - Protocol Translation (NAT-PT) - Internet Request for Comment 2766 (2000).

[7] Soares, R. P. and Trevisani, K. M.: Integration of translating and transmission modules of SOH and extension of it functionalities (in Portuguese). Technical Report, FIPP/UNOESTE, (2006).

[8] Nagai, D. M. and Trevisani, K. M.: Development of a SOH module to translate SPML diagrams to XML documents. (In Portuguese) Technical Report, FIPP/UNOESTE (2006).

[9] Gamba, F. M. and Trevisani, K. M.: Validation of FwXML to firewalls products OpenBSD Packet Filter and Linux Netfilter. (In Portuguese) Technical Report, FIPP/UNOESTE (2007).

[10] Artymiak, J.: Building Firewalls with OpenBSD and PF. 2.ed. Poland, Sowa (2003).

[11] Russell, R.: The NetFilter Project, <http://www.netfilter.org>. Accessed in Aug. 2008.

[12] Aker Security Solutions: Firewall AKER Manual, São Paulo/SP (2003).