A policy-based approach to firewall management

Filipe Caldeira and Edmundo Monteiro
Polytechnic Institute of Viseu, ESTV - Department of Informatics {caldeira@di.estv.ipv.pt}
University of Coimbra, Centre for Informatics and Systems (CISUC) {edmundo@dei.uc.pt}

Abstract: This paper describes a policy-based approach to firewall management. The Policy-Based Networking (PBN) architecture proposed by the Policy Framework Group of IETF is analysed, together with the communication protocols, policy specification languages, and the necessary information models. The paper continues with a description of an application of the PBN architecture to firewall management. The proposed architecture is presented and its implementation issues are analysed with some usage examples. The paper concludes with the evaluation of the policy-based approach to firewall management.

Key words: Network security, Policy-Based Networking, COPS, COPS-PR, SPSL

1. INTRODUCTION

Several attempts have been made, along the last years, for the development of new network management methodologies. The traditional approaches are mainly oriented to the management of individual components, not having in consideration the network structure as a whole. Currently a great effort is being made to diminish the increasing complexity of networks management, with the use of new paradigms. The policy-based management model (PBN – Policy Based Networking) [Stevens 1999] intends to support the change from the actual configuration mechanisms, to an integrated management system approach.

In this paper a PBN approach to firewall management is presented. The work is structured in two areas. First, the Policy-Based Networking (PBN) architecture proposed by the Policy Framework Group of IETF is analysed, together with the communication protocols, policy specification languages, and the necessary information models. Then, the proposed architecture is presented and its implementation issues are analysed with some usage examples. The paper concludes with the evaluation of the PBN approach to firewall management.

2. POLICY-BASED MANAGEMENT

The Policy-Based Networking (PBN) architecture had origin in work developed by the Policy Framework Group of IETF [Stevens 1999]. The main objectives of this work are centralized network management, support of abstract definition of rules and policies, use of the same rules for different types of equipment and, automation of network management tasks. According to the PBN concept, the system administrator must describe what the network should do instead worrying about the way this will be implemented [Shepard 2000] [Mahon 1999].

This architecture defines four main entities (*Figure 1*): Management Console (MC), Policy Repository (PR), Policy Decision Point (PDP) and Policy Enforcement Point (PEP). Communication between entities is made with two different protocols: one for repository access and another for policy exchange.

Usually the PEP is implemented in network equipment, managing it in accordance with instructions received from a PDP. The PDP processes policies defined for the domain, together with other network administration data and makes decisions that, then, become new PEP's configurations. The Policy Repository stores policies defined inside the organization. In this architecture, the PDP uses the policy repository to obtain policies. The Management Console allows policy edition, translation and validation, in order to be able to store them in the repository and then be used by PDPs.

Using the PBN architecture, the set of policies defined by the administrator are implemented in the network by devices controlled by existing PEPs (*Figure 2*). This architecture can be seen on different two perspectives according to the *two-tier* or *three-tier* models [Raju 1999]. *Figure 2* shows the three-tier perspective, comprised by PEPs, PDPs, MC and PR. In this model, PEPs are entities that only enforce rules received from PDPs, without decision capacity. On the opposite, the two-tier approach allows components to make local decisions.

In the PBN architecture, interaction between PDPs and PEPs is done using a client-server model. In this context two kinds of operations can be identified: *outsourcing* and *provisioning* [IPHighway 2001].

In the outsourcing model, when an event occurs that the PEP doesn't know how to handle, a message is sent to the PDP notifying the occurrence. The PDP replies to the PEP sending the information needed to handle the event. This model is also known by *pull* or *reactive* because the PDP reacts to events originated in the PEPs.

In the provisioning model, when the PEP establishes the initial connection, the PDP sends all existent information applicable to that specific PEP. This information is kept in the PEP and all events that come to happen will be treated according to this information. This model is also called call *push* or *proactive*.

In the outsourcing model, the requests made by PEPs must be answered with only one decision (a 1:1 relation exists between requests and decisions). In the provisioning model this relation is not always 1:1 because the decision can be based on an external event known by the PEP.

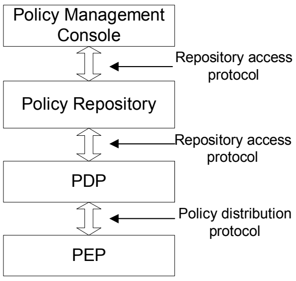
Policy Management Console
Repository access protocol
Policy Repository
Repository access protocol
PDP
Policy distribution protocol
PEP

Figure 1. PBN architecture components [INTAP 2001]

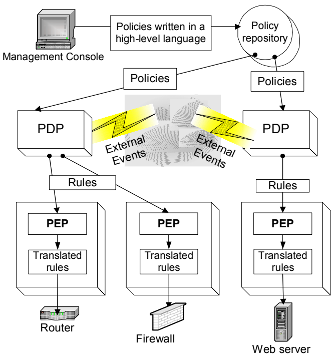

Figure 2. PBN architecture [Boutaba 2001]

The policy repository can be seen as a database that contains policy information. Several possibilities exist to store policy information (e.g. text files, database), however they must, necessarily, obey to a specific data model used to represent policy information.

In the IETF the Directory Enabled Networking group (DEN) of the DMTF (Distributed Management Task Force) is concentrating efforts in the creation of an object-oriented model than can represent policy information. This data model – Common Information Model (CIM) [Moore 2000] – is intended to provide the foundations for policy representation for different application areas, like QoS and network security. CIM's main objective is to describe information necessary to system and network management and relations between this information [Booch 1996].

2.1 Policy exchange protocols

The PBN architecture provides standard protocols for information exchange between its entities. These protocols are necessary to allow communication without restrictions among products from different manufacturers and to ensure the openness of the PBN solution.

2.1.1 COPS (Common Open Policy Service)

The Common Open Policy Service (COPS) [Durham 2000] has origin in work carried out at IETF to create a new standard protocol for communication between PDPs and PEPs. This protocol and its extensions are being developed by the Resource Allocation Protocol group of IETF with special focus on the Resource Reservation Protocol (RSVP) [Braden 1997] applications, however, due to its generic architecture, the protocol is motivating other working groups in different research areas.

COPS can be divided into three distinct layers: the base protocol, the client specific commands and the data representation. The base protocol defines the communication mechanisms that allow information exchange between PDPs and PEPs. Following a different approach from SNMP protocol, COPS is based on TCP connections and is stateful, meaning that the server keeps client's state. The *type of client* concept, allows the addition of a second level of abstraction to COPS constituted by client specific commands, allowing COPS to be used in different areas with the addition of new client specific layers. The main differences among the layers are message's *type* and *data*.

2.1.2 COPS for Policy Provisioning

The Common Open Policy Service for Policy Provisioning (COPS-PR) [Chan 2001] is an extension to COPS that uses the provisioning model described before. With COPS-PR each PEP connect to the primary PDP, informing its local capabilities (e.g. hardware type, software version, configuration information) and requesting a set of policies to store and enforce locally. In case that the PDP has no support for one specific type of PEP an error message containing the address of an alternative PDP is generated [Chan 2001]. COPS-PR is useful when PBN is used for configuration of network devices instead of its operational management only.

With COPS-PR, PEPs must have a local data structure called Policy Information Base (PIB) where the received information can be stored [Fine 2000]. This database can use a data model like the one defined by the CIM. PIBs are similar to MIBs (Management Information Base) used by SNMP. Tools are available to convert a MIB into a PIB [Fine 1999].

PIBs are structured by policy type or class according to a tree structure, where branches represent policy types or classes (PRCs – Policy Rule

Classes) and leaves represent the policy content (PRIs – Policy Rule Instances). Each PRC can contain multiple PRIs. Each PRI is identified with a PRID – Provisioning Instance Identifier. Policies are made of a set of PRIs. PDPs can install new PRIs, remove or modify existing PRIs in PEPs, trough COPS messages.

PIBs are defined, in COPS-PR, as abstract structures. The details are specified in separate documents. Already defined PIBs cover different areas of management, such as network security and QoS. The PIB definitions are made with a high level of abstraction, hiding hardware details. With this abstraction level, different network equipment can be controlled using the same data structure. The use of PIBs offers a good data abstraction, allowing the recognition of the same data in different equipment and PDPs to ignore the specific PEP implementation, providing thus, an additional level of abstraction in network and system management.

3. POLICY SPECIFICATION LANGUAGES

To integrate the PBN architecture some kind of language is needed to allow network managers to describe the behaviour they want to impose to the managed system. Currently there is no standard language for this purpose, only some draft proposals exist, each one concerning a different application field.

A policy specification language must support the concepts of the PBN architecture in a simpler and understandable way by network managers, for the different areas of network and system management. It must also support multiple management activities. The policies must be organized in groups and not treated individually in order to facilitate the policy specification in complex systems. The language must allow some type of policy composition and the support for consistency verification and conflict analysis between policies. The language must be expandable in order to allow new policy types [Stone 2001].

3.1 Security Policy Specification Language (SPSL)

Since this work is focused in the study of the PBN application to firewall management, Security Policy Specification Language (SPSL) was the choice due to its application domain [Condell 2000]. Sponsored by IETF, SPSL was initially developed to be used for IPSec policy management. However it is possible to expand its syntax to be used in other application scenarios. The language syntax was derived from the Routing Policy Specification Language (RPSL) [Alaetinouglu 1998].

SPSL defines a set of classes that can be used to instantiate objects. Each object has a set of attributes associated that contain relevant data for the definition of policies. Objects do not contain methods and it is not possible in this specification to create new types of objects.

In order to allow one global security policy, SPSL deals with policies based on nodes and domains. To support these concepts the classes, "node", "gateway", "polserv" and "domain" are defined. One "node" is a representation of a single network entity. The "gateway" is an entity that implements a security policy. The "polserv" defines a policy server who is capable to deal with SPSL rules and, has upcoming feature, capable to translate these rules into the specific commands of various equipments.

One "domain" contains a set of "nodes". The policies are represented using the class "policy" that must be associated to one or more "node" or "domain". Objects must be signed by a digital certificate. The order by which objects are defined define their application precedence. In packet filters objects, for instance, the first rule that matches the conditions is the one applied.

The class "policy" can have one or more "policy” attributes. It is possible, for example, to specify a network connection using its destination and origin addresses, transport protocols, ports and traffic direction. Actions supported in rules are permit, deny and forward.

Security of policies written in SPSL, have to be assured by applications that manipulate the policy file. The attributes "mnt-by" and "signature" enable policy integrity validation with standard authentication mechanisms.

To enable firewall management some extensions to SPSL where proposed in this work. These extensions include two new actions: reject and redirect port.

4. APPLICATION TO FIREWALL MANAGEMENT

To evaluate the application of the PBN architecture to firewall management, an application model was conceived using COPS-PR as policy exchange protocol and SPSL as policy description language. Since each firewall model has its own set of commands and features, the application needs to generate a different set of rules for each firewall, based on the global security policy defined for the domain. The implementation was done in Linux operating system and can be used to manage IPChains Firewalls (IPTables in recent kernel versions) [Russel 2000] and Cisco Access Control Lists (ACL) [Cisco 2001]. The application developed can be easily extended to other equipments or functions, for example, to QoS management in the routers.

In *Figure 3* the application scenario is shown. In the scenario, the security policy server – PDP function is used to manage two firewalls PEP function. The two different outside connections with different security requirements will enable the evaluation of the application in the management of the firewalls with different policies.

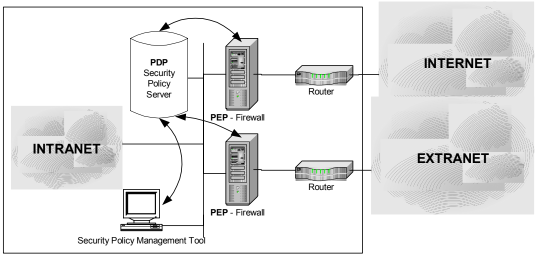

Figure 3. Evaluation scenario

4.1 Implementation issues

The implementation model of the application is shown in *Figure 4*. This model has three main components: Management Console, PDP and PEP. The Policy Repository considered in PBN architecture is integrated in the PDP.

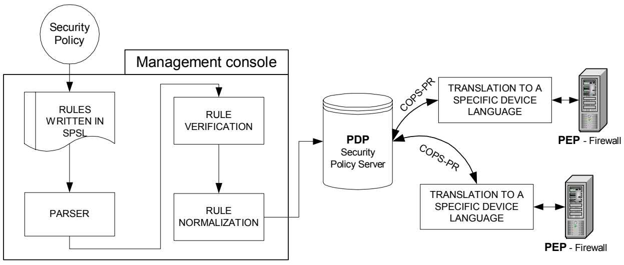

Figure 4. Proposed implementation model

The Management Console is a tool that allows generation and handling of policies written in SPSL. The tool includes a set of modules for language processing and policy description. It includes a parser and mechanisms for rule verification and normalization. The policy rules processed are stored in the policy repository maintained by the PDP.

The PDP (policy server) is the responsible for policy distribution to PEP located in managed network elements. PEPs are then responsible for policy translation into equipment commands and policy installation and enforcing.

COPS-PR is used for policy exchange between the PDP and PEPs. In this work the COPS-PR implementation from Luleå University of Technology [Bergsten 2000] was used and improved.

SPSL revealed some limitations in what respects to the specification of packet filtering rules. The extensions to SPSL proposed [Caldeira 2000] include two new actions: reject and redirect-port. With these two actions IPChains rules can be directed derived from policy rules.

4.2 Rule generation and verification

Syntax verification of policy rules is the first issue addressed by the application [Caldeira 2000]. A parser generated using Bison [Donnelly 1995] and Flex [Paxson 1995] is used to verify the syntax of SPSL in BNF (Backus-Naur Form). At the same time a data structure containing all valid objects is created. When errors are found the erroneous object is excluded with an error report.

Each object is treated individually, considering its type, domain, policy and other relevant attributes. *Figure 6* shows the result of the use of the parser with the example policy depicted in *Figure 5* [Caldeira 2000].

```
domain:    IPV-SERVERS
coverage:  SERVER1,SERVER2
mnt-by:    FC
changed:   FC 20020623
signature: FC FC-CERT \
           KEYEDMD5
           ABBA007AB47E
```

```
policy-name: POL1
association: IPV-SERVERS
policy:    dst 193.37.7.1 src * \
           xport-proto 6 permit
mnt-by:    FC
changed:   FC 20020623
signature: FC FC-CERT \
           KEYEDMD5 ABBA007AB47E
```

```
policy-name: DEFAULT
association: IPV-SERVER
policy:    dst * src any deny
mnt-by:    FC
changed:   FC 20020623
signature: FC FC-CERT \
           KEYEDMD5ABBA007AB47E
```

Figure 5. SPSL policy example

```
N°: 1
Association: IPV-SERVERS
Name: POL1
Action: Permit
Origin:  *
Destination: 193.37.7.1
Protocol: 6
```

```
N°: 2
Association: IPV-SERVER
Name: DEFAULT
Action: Deny
Origin:  *
Destination:  *
```

Figure 6. Data structure generated from SPSL policy

After syntax verification, the application examines the resulting data structure and excludes incorrect rules. However, after this verification, some problems may remain, for example, two contradictory rules or rules whose network address ranges are not disjoint. The intersection of addresses will not cause problems in rule application. However, for the definition of a global policy, the existence of never applied rules induces extra complexity. In these cases an algorithm is applied to simplify the rules.

For example, in *Figure 7*, the first rule generalizes the second, allowing TCP connections from any IP to 193.137.7.2. Thus, the second rule is never applied. The simplification algorithm expands all addresses into address ranges, verifies the presence of range intersections and removes them. The outcome of this verification is shown in *Figure 8*. This algorithm also verifies intersections of other rule attributes, such as origin and destination ports and protocol numbers.

```
policy-name:    POL1
association:    IPV-SERVERS
policy:         dst not 193.137.7.1 \
                src * xport-proto 6 permit
mnt-by:         FC
changed:        FC 20020623
signature:    FC FC-CERT KEYEDMD5 ABBA007AB47E
```

```
policy-name:    POL2
association:    IPV-SERVER
policy:         dst 193.137.7.2  \
                src 187.100.100.0/24 \
                xport-proto 6 permit
mnt-by:         FC
changed:        FC 20020623
signature:   FC FC-CERT KEYEDMD5 ABBA007AB47E
```

Figure 7. SPSL Example

```
N°: 1
Association: IPV-SERVER
Name: POL1-01
Action: Permit
Origin: *
Destination: 0.0.0.0- 193.137.7.0, 193.137.7.2-255.255.255.255
Protocol: 6
```

Figure 8. Rule simplification

4.3 Rule storage, distribution and application

In this implementation, SPSL rules are stored in a text file. Other possibility is the use of SGBD with support for XML [Bray 1998]. To enable this, the application developed allows the creation of a XML file with the set of rules to be received and translated by PEPs. With this enhancement the application can be used with clients that do not support COPS-PR.

The implementation of PDP, PEP, COPS and COPS-PR was done to enable rule distribution and application. The PEP has the ability to translate rules in commands recognized by controlled equipment through a set of recognized commands generated automatically.

One of the problems found was the differences among equipment models. The decision was to generate rules closer to the original, informing users that rules can't be supported by the specific equipment command set.

The rule simplification algorithm, that expands address ranges, can lead to an increase in rule number. To solve this problem, ranges values are again converted into one or more atomic host or network IP addresses. Currently, the application supports the conversion of SPSL into IPChains rules and Cisco ACLs, and can be easily upgraded to support other equipment.

4.4 Usage example

This usage example is related to the evaluation scenario described above. The application of SPSL policy rules in *Figure 9* result in a new set of rules after parsing, validation and normalization. The resultant rules are sent to the PDP's PIB and then, by the PDP, to all the PEPs that convert them into

IPChains or Cisco ACLs. *Figure 10* shows the translation into IPChains of policy rules concerning the evaluation scenario.

```
policy-name:    POL1
association:    IPV-SERVER
policy:
   dst 193.137.7.2 port 110 \
   src 193.137.6.0/24 \
   xport-proto 6,17 \
   direction inbound permit
mnt-by:     FC
changed:    FC 20020623
signature:  FC FC-CERT \
         KEYEDMD5 ABBA007AB47E
```

```
policy-name:    POL2
association:    IPV-SERVER
policy:
    dst 193.137.7.3 port 80 \
    src * xport-proto 6 \
    direction inbound deny, \
    forward port 8080
mnt-by:     FC
changed:    FC 20020623
signature:  FC FC-CERT \
         KEYEDMD5 ABBA007AB47E
```

```
policy-name:    DEFAULT
association:    IPV-SERVER
policy:
    dst any src any reject



mnt-by:     FC
changed:    FC 20020623
signature:  FC FC-CERT \
         KEYEDMD5 ABBA007AB47E
```

Figure 9. A global policy written in SPSL

```
Rule N° 1, POL1, assoc: IPV-SERVER
ipchains -A input -p 6 –s 193.137.6.0/255.255.255.0 -d 193.137.7.2 110 –j ACCEPT
Rule N° 2, POL1, assoc: IPV-SERVER
ipchains -A input -p 17 -s 193.137.6.0/255.255.255.0 -d 193.137.7.2 110 –j ACCEPT
Rule N° 3, POL2, assoc: IPV-SERVER
ipchains -A input -p 6 -d 193.137.7.3 80 –j REDIRECT 8080
Rule N° 4, DEFAULT, assoc: IPV-SERVER
ipchains -A input –j REJECT
```

Figure 10. IPChains translation

The results for the Cisco router ACLs are presented in *Figure 11*. The translation of rules 4 and 5 is different from the translation in the case of IPChains. In rule 4, the packet redirect functionality has different forms of application. For the Cisco router two new lines are created in the ACL together with a line that specifies the translation of an address (this last line does not belong to ACLs definition). Rule 5 is implemented by default by the router, making it redundant.

```
Rule N° 1, POL1 , assoc: IPV-SERVER
access-list 101 permit tcp 193.137.6.0 0.0.0.255 host 193.137.7.2 eq 110
Rule N° 2, POL1 , assoc: IPV-SERVER
access-list 101 permit udp 193.137.6.0 0.0.0.255 host 193.137.7.2 eq 110
Rule N° 3, POL2 , assoc: IPV-SERVER
access-list 101 permit tcp any host 193.137.7.3 eq 80
access-list 101 permit tcp any host 193.137.7.3 eq 8080
ip nat inside source static tcp 193.137.7.3 80 193.137.7.3 8080
```

Figure 11. Cisco ACL translation

4.5 Evaluation

From the global security policy specification, the developed application can manage the implementation of this policy in existing network equipment in the evaluation scenario. With this approach, network administrators don’t need to have a deep knowledge about languages and techniques used in every type of equipment supported by the application.

In this phase the application only deals with packet filtering policies. This introduces limitations due to the lack of mechanisms to configure other equipment aspects.

The rule translation module for IPChains and Cisco ACL can originate non optimal configurations due to syntax and semantic translation limitations. The management console uses a command line interface preventing administrators to have a general view of existing policies.

To conclude this evaluation, it can be said that this application brings advantages to an organization if seen as a centralized configuration manager of supported firewalls.

5. CONCLUSION AND FUTURE WORK

The present study shows the difficulty of implementing the PBN approach to manage a large diversity of network equipment. In general, this architecture has some potential that allows for cost reduction and improvement on security and QoS in computer networks.

The "plug and play" network configuration with a set of policies written in a high-level language is not, yet, a generalized solution. Many manufacturers propose solutions for their own equipment. The use of these solutions with the equipment of other manufacturers is still hard. The work developed supports this conclusion, specifically for security management.

Through this study, it can also be concluded that COPS and COPS-PR protocols have good potential to be chosen to support PBN solutions. SPSL was the available choice to describe security policies. However, it revealed weaknesses concerning policy specification for other management areas.

As a final remark, the development of standard firewall configuration solutions using the PBN model is an interesting approach.

Future work will include language extensions (or even the adoption of a new language) for richer policy description and full support of equipment command set and the development of translation mechanisms to other devices types.

6. REFERENCES

[Alaetinouglu 1998] Alaetinouglu, C. et al., Routing Policy Specification Language (RPSL), RFC 2280, IETF, January 1998.

[Bergsten 2000] Bergsten Anders, Borg Niklas, Implementation and Evaluation of the Common Open Policy Service (COPS) Protocol and its use for Policy Provisioning, (extwww.lulea.trab.se/cops), 2000.

[Booch 1996] Booch, G. et al., Unified Method for Object-Oriented Development Document Set, Rational Software Corporation, 1996, (http://www.rational.com/uml).

[Boutaba 2001] Boutaba, R. et al., COPS-PR with meta-policy support, IETF independent publication, April 2001.

[Braden 1997] Braden, R. et al., Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification, RFC 2205, IETF, September 1997.

[Bray 1998] Bray, T. et al., eXtensible Markup Language (XML) 1.0, W3C, February 1998, (http://www.w3c.org/TR/REC-xml).

[Caldeira 2000] Caldeira, F., Monteiro, E., Descrição Geração e Difusão de Políticas de Segurança, in Proceedings of CRC'2000, November 2000.

[Chan 2001] Chan, K. et al., COPS Usage for Policy Provisioning (COPS-PR), RFC 3084, IETF, March 2001.

[CIM 1999] Common Information Model (CIM) Specification – Version 2.2, DMTF, June 1999 (http://www.dmtf.org/spec/cim_spec_v22/).

[Cisco 2001] Online manuals (http://www.cisco.com)

[Condell 2000] Condell, M. et al., Security Policy Specification Language, Internet draft, draft-ietf-ipsp-spsl-00.txt, IETF , March 2000.

[Donnelly 1995] Donnelly C., Stallman R., Bison - The YACC-compatible Parser Generator, (http://www.gnu.org/manual/bison/html_mono/bison.html), November 1995.

[Durham 2000] Durham D., Boyle J., Cohen R., Herzog S., Rajan R., Sastry A., The COPS (Common Open Policy Service) Protocol, RFC 2748, Network Working Group, IETF, January 2000.

[Fine 1999] Fine, M. et al., Quality of Service Policy Information Base, Internet draft, draft-mfine-cops-pib-01.txt, IETF, June 1999.

[Fine 2000] Fine, M. et al., Framework Policy Information Base, Internet draft, draft-ietf-rap-frameworkpib-04.txt, IETF, November 2000.

[INTAP 2001] Survey on Policy-Based Networking - Addressing Issues, Technological Trends, Future Prospects of Policy Exchange Methods in Multi-Domain Scenarios, INTAP, 2001, (http://www.net.intap.or.jp/INTAP/).

[IPHighway 2001] Policy Standards and IETF Terminology, White paper, Volume #2, IPHighway, January 2001.

[Mahon 1999] Mahon, H. et al., Requirements for a Policy Management System, Internet draft, draft-ietf-policy-req-02.txt, IETF, November, 1999.

[Moore 2000] Moore, B. et al., Policy Core Information Model – Version 1 Specification, Internet draft, draft-ietf-policy-core-info-model-04.txt, IETF March 2000.

[Paxson 1995] Paxson,V., Flex - A fast scanner generator, (http://www.gnu.org/manual/flex-2.5.4/html_mono/flex.html), March 1995.

[Raju 1999] Raju, Rajan et al., A policy framework for integrated and differentiated services in the internet, in IEEE Network, September 1999.

[Russel 2000] Russell, R., Linux IPChains HowTo, Online, July 2000.

[Shepard 2000] Shepard, S.,Policy-based networks: hype and hope; in IT Professional, Vol. 2, No. 1, January 2000.

[Stevens 1999] Stevens, M. et al., Policy Framework, Internet draft, draft-ietf-policy-framework-00.txt, IETF, September 1999.

[Stone 2001] Stone, G. et al., Network Policy Languages: A Survey and a New Approach, in IEEE Network, pag. 10-21, January 2001.

[Tosic 1999] Tosic, V. et al., The Common Information Model (CIM) Standard – An Analysis of Features and Open Issues, in Proceedings of the TELSIKS ’99, Nis, Yugoslavia, October 1999.