Policy-Based Networking: Applications to Firewall Management

Filipe Caldeira
Polytechnic Institute of Viseu, ESTV, Department of Informatics
Campus Politécnico de Repeses
3504-510 Viseu
Portugal
{caldeira@di.estv.ipv.pt}

Edmundo Monteiro
University of Coimbra, CISUC/DEI, Laboratory of Communications and Telematics
Polo II, Pinhal de Marrocos
3030-290 Coimbra
Portugal
{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 Internet Engineering Task Force (IETF) is analysed, together with the communication protocols, policy specification languages, and the necessary information models.

An overview of policy specification languages applicability to PBN architecture is presented paying particular attention to the specification of security policies through Security Policy Specification Language (SPSL).

The Common Open Policy Service protocol (COPS) and its variant, COPS for Policy provisioning (COPS-PR), both used for the transport of policy information, are also presented.

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, Firewall, COPS, COPS-PR, SPSL

1. INTRODUCTION

With the considerable growth in dimension and complexity of the actual computer networks, tasks regarding to its configuration, administration and maintenance, have come, gradually, to consume more resources in organizations. The managed systems also grow in complexity with the emergence of new service requirements, like Quality of Service and network security [Dinesh 2002].

Several attempts have been made, during the last years, in the development of new network management methodologies. The traditional approaches are mainly oriented to the management of individual components, not considering the network structure as a whole. These methods were shown satisfactory until the administration tasks started to consume too many resources in organizations.

Currently a great research effort is being made to reduce the increasing complexity of networks management systems, 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 mechanisms, to an integrated management system approach.

In this paper a PBN approach to firewall management is presented. The work is structured in two parts. 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 firewall management system 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 has 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, the 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.

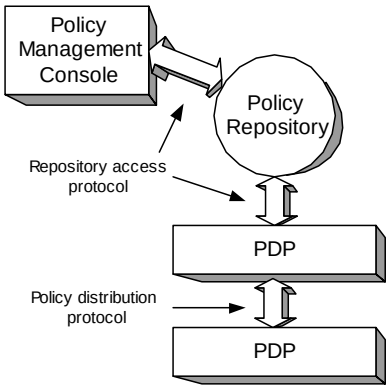

Figure 1. PBN architecture components [INTAP 2001]

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 regarded from two different 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. This model can be used when some network components are able to make some type of processing, like keeping rules or making their own decisions.

In the PBN architecture, interaction between PDPs and PEPs follows the 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 the PEP doesn't know how to handle occurs, 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. We can call this model push or proactive [IPHighway 2001] because the PDP sends all the information in advance.

In the outsourcing model, the requests made by PEPs must be answered with a single 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.

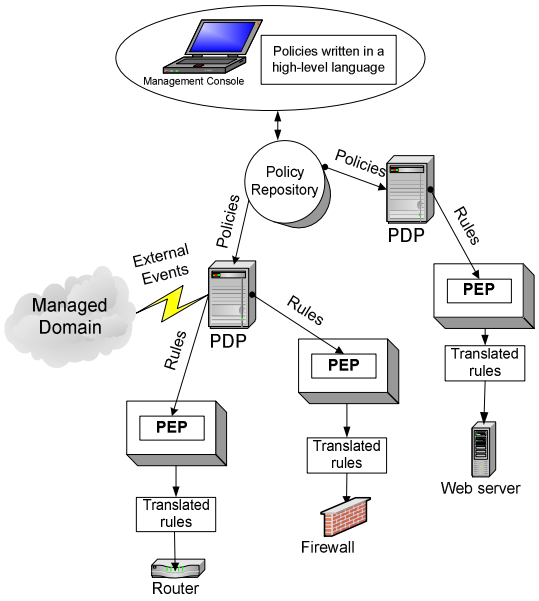

Figure 2. PBN architecture [Boutaba 2001]

The policy repository can be seen as the place where the policies associated to a particular domain are stored, or, in general, as a database that contains policy information. There are several ways to store policy information (e.g. text files, SGBD). However they must obey to a specific data model used to represent policy information.

The structures actually used to store the information needed for policy-based management are quite generic and vague. 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 of different application areas, like QoS and network security.

The main objective of CIM is to describe the information necessary to system and network management and the existing relations between these categories of information [Booch 1996]. CIM is not a language or implementation. Instead, it just defines a data model for the storage of management information.

2.1 Policy exchange protocols

Independently of using the two-tier or three-tier models, in the PBN architecture we need to use normalized protocols for information exchange between devices that compose the architecture. This 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] was originated from work carried out at the IETF to create a new standard protocol for communication between PDPs and PEPs. This protocol and its extensions were developed by the Resource Allocation Protocol Group [RAP 2001] 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 related research areas.

The COPS protocol is organised in 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 (being PEP the client and PDP the server). Following a different approach from SNMP, COPS is based on TCP connections and is statefull, meaning that the server keeps client's state and reacts whenever necessary, even without a solicitation from the client. The *type of client* concept, allows the addition of a second level of abstraction to COPS with 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*. COPS is quite flexible supporting objects defined for several different contexts.

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.

In the outsourcing model, the requests made by PEP must be answered with a single decision because a 1:1 relation exists between requests and decisions. In the provisioning model the relation is not always 1:1 because the PDP can make a decision based on an external event. A new policy can origin some modifications in the rules applied to a specific PEP, for example.

With COPS-PR, each PEP is connected to a primary PDP, informs him of its local capabilities and requests a set of policies to store and enforce locally.

The operation of COPS-PR begins when the client opens a COPS connection from the PEP to the primary PDP. If the connection is successfully established, the PEP sends to the PDP a REQ message containing client's specific information (hardware type, software version, configuration information). At this stage, the PEP can also specify the maximum size of COPS-PR messages. After receiving this information the PDP sends the PEP the policies to be enforced locally.

When the PDP has no support for a specific type of PEP an error message with the address of an alternative PDP is generated [Chan 2001]. COPS-PR is useful when the PBN model is only used for configuration of network devices instead of for its operational management.

With COPS-PR, PEPs must have a local data structure called Policy Information Base (PIB) where the received information is stored [Fine 2000]. This database can use a data model like the one defined by the CIM previously described. PIBs are similar to MIBs (Management Information Base) used with 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 composed 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. PIB definition is made at 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 use of the same data for 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 also 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) [Condell 2000] was the choice due to its application domain. SPSL was initially developed, under the sponsoring of IETF, 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], and can be used to work with rules relating to data communication, like packet filters.

3.1.1 Language Components

SPSL is based on objects, without being an object-oriented language. It 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. In the current specification of SPSL, objects do not contain methods and it is not possible 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". The object "mnter" keeps information about the person responsible for security policies (name, phone, email, certificates and other information). Objects must be signed with a digital certificate to guarantee its integrity. The order by which objects are created defines their 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”.

SPSL security policies have to be validated by applications that manipulate the policy files. The attributes "mnt-by" and "signature" enable policy integrity validation with standard authentication mechanisms.

To enable firewall management some extensions to SPSL were proposed in this work. These extensions include two new actions: “reject” and “redirect port” to match the filtering rules of the Linux firewall [Russel 2000].

3.1.2 Language usage examples

Following is an example of policy using class "policy":

```
policy-name: EXAMPLE
policy:      dst 193.197.128.43    src * xport-proto 6 permit
```

This policy allows all connections to address 193.197.128.43 from any origin, when the protocol is TCP. The same policy can be written using the long format of the same class. The long format contains more attributes that can be used, for example, if the address is IPv6.

The security of policies written in SPSL has to be assured by the applications that use the text file. The verification of the validity of the policy is done by the existing attributes "mnt-by" and "signature".

While the text file can be easily modified, it is however difficult to write a valid signature for objects without the respective private key. This approach guarantees the integrity of the object, but security concerns about the text file remain an issue. Therefore, the removal of one policy will not be recognized by an application that only verifies the objects signatures. In this case we must pay attention to the validation of the file integrity.

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 the Linux operating system and can be used to manage IPChains Linux firewalls (IPTables in more 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.

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

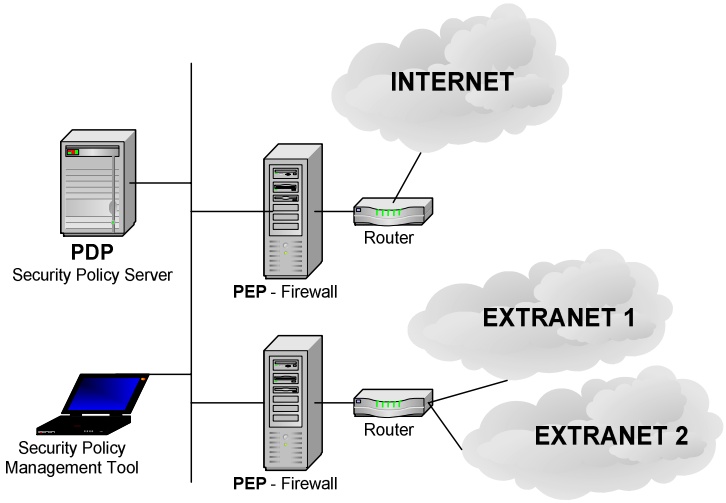

Figure 3. Evaluation scenario

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.

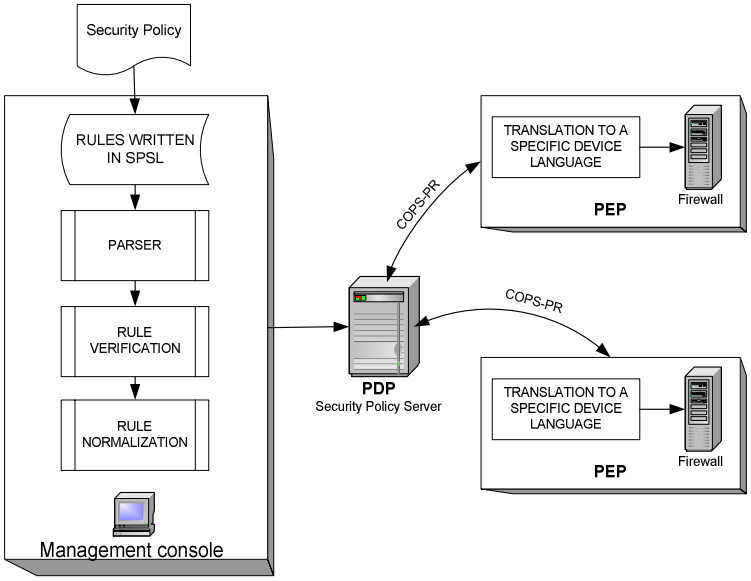

Figure 4. Proposed implementation model

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, 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 regarding the specification of packet filtering rules. The extensions to SPSL proposed and described in previous work [Caldeira 2000] include two new actions: reject and redirect-port. With these two actions IPChains rules can be directly derived from policy rules.

4.2 Rule generation

The security policy defined for the organization is described using language SPSL, extended with the extensions already mentioned. The implemented system uses a text file where policy is stored and generates a set of rules that must later be stored in a policy repository. In this implementation, we can consider that the policy repository is the text file with SPSL information. After the necessary processing to this file, the rules are sent immediately to the PDP.

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, allowing the process to continue in remaining rules, without compromising the global organization security.

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].

After syntax verification, the application examines the resulting data structure and excludes incorrect rules, for example the use of an address range where the minimum has a smaller value than the maximum, (193.134.56.7 - 193.134.55.1) or even for invalid network addresses (193.3.3.5/24).

```
domain:       ESTV-SERVERS
coverage:     SERVER1,SERVER2
mnt-by:       CAL
changed:      CAL 20030523
signature:    CAL CAL-CERT \
              KEYEDMD5 ABBA007AB47E

policy-name:  POL1
association:  ESTV-SERVERS
policy:       dst 193.137.7.1  src * \
              xport-proto 6 permit
mnt-by:       CAL
changed:      CAL 20030523
signature:    CAL CAL-CERT \
              KEYEDMD5 ABBA007AB47E
```

```
policy-name:  POL2
association:  ESTV-SERVER
policy:       dst 193.137.7.2 \
              src 193.137.6.0/24  \
              xport-proto 6 permit
mnt-by:       CAL
changed:      CAL 20030523
signature:    CAL CAL-CERT \
              KEYEDMD5 ABBA007AB47E

policy-name:  DEFAULT
association:  ESTV-SERVER
policy:       dst * src any deny
mnt-by:       CAL
changed:      CAL 20030523
signature:    CAL CAL-CERT \
              KEYEDMD5 ABBA007AB47E
```

Figure 5. SPSL policy example

```
.......
Nº: 2
Association: ESTV-SERVERS
Name: POL1
Action: Permit
Origin:  *
Destination: 193.137.7.1
Protocol: 6
```

```
Nº: 3
Association: ESTV-SERVER
Name: POL2
Action: Permit
Origin: 193.137.6.0/255.255.255.0
Destination: 193.137.7.2
Protocol: 6
```

```
Nº: 4
Association: ESTV-SERVER
Name: DEFAULT
Action: Deny
Origin:  *
Destination:  *
```

Figure 6. Data structure generated from SPSL policy

4.3 Rule verification

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 rules that are never applied 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 address 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:  ESTV-SERVERS
policy:       dst not 193.137.7.1 \
              src * xport-proto 6 permit
mnt-by:       CAL
changed:      CAL 20030523
signature:    CAL CAL-CERT \
              KEYEDMD5 ABBA007AB47E
```

```
policy-name:  POL2
association:  ESTV-SERVER
policy:       dst 193.137.7.2  \
              src 187.100.100.0/24 \
              xport-proto 6 permit
mnt-by:       CAL
changed:      CAL 20030523
signature:    CAL CAL-CERT \
              KEYEDMD5 ABBA007AB47E
#
```

Figure 7. SPSL Example

```
Nº: 1
Association: ESTV-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

4.4 Rule storage

In this implementation, SPSL rules are stored in a text file. Other possibility is to use SGBD with XML support [Bray 1998] or, at a higher level of abstraction, use a data model based on CIM [CIM 1999]. When using SGBD, the relational or object-relational models can be used. The queries made by PDPs to the repository can be made using SQL, which will return a XML file, that will be processed by the PDP.

A Web based management console can then be easily implemented to allow database interactions in XML format. The utilization of XML is advantageous because XML is already a standard created by W3C and is sufficiently extensible in what concerns the data to be exchanged.

To enable XML use, 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.

4.5 Rule distribution and application

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 the controlled equipment through a set of recognized commands automatically generated.

After the global security policy is written and processed, we have to apply it to one or more equipment, like firewalls or routers. Our intention is that equipments can be instructed through a set of recognized commands generated automatically by a translator.

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.

SPSL rules have more flexibility than usually found in equipments. For example, one initial aspect of conversion is the translation of ranges values to atomic values, a functionality that is not supported by the majority of equipments.

The rule presented in *Figure 9* will be converted into four different rules through the combination of values that are not atomic, resulting in a new set of rules, described in *Figure 10*.

```
.......
policy-name:  POL1
association:  ESTV-SERVERS
policy:       dst 193.137.7.1-193.137.7.2 \
       src 193.1.1.1, 194.1.1.1 xport-proto 6 direction inbound permit
mnt-by:       CAL
changed:      CAL 20030523
signature:    CAL CAL -CERT KEYEDMD5 ABBA007AB47E
```

Figure 9. SPSL rule

Nº: 1 Assoc: ESTV-SERVERS Name: POL1 Action: Permit Origin: 193.1.1.1 Dest: 193.137.7.1 Protocol: 6 Direction: inbound	Nº: 2 Assoc: ESTV-SERVERS Name: POL1 Action: Permit Origin: 193.1.1.1 Dest: 193.137.7.2 Protocol: 6 Direction: inbound	Nº: 3 Assoc: ESTV-SERVERS Name: POL1 Action: Permit Origin: 194.1.1.1 Dest: 193.137.7.1 Protocol: 6 Direction: inbound	Nº: 4 Assoc: ESTV-SERVERS Name: POL1 Action: Permit Origin: 194.1.1.1 Dest: 193.137.7.2 Protocol: 6 Direction: inbound

Figure 10. Rules resultant from the simplification process

The rule simplification algorithm, that expands address ranges, can lead to an increase in rule number. To solve this problem, range values are again converted into one or more atomic host or network IP addresses. For example, if the simplified rule has the destination address range 193.137.7.0 - 193.137.7.255, it will be transformed into 193.137.7.0/24, and can thus be applied using syntax of most equipment.

Currently, the application supports the conversion of SPSL into IPChains rules and Cisco ACLs, and can be easily upgraded to support other equipment.

4.6 Usage example

This usage example is related to the evaluation scenario described above. The application of SPSL policy rules in *Figure 11* results 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 12* shows the translation of policy rules into IPChains.

```
policy-name:  POL1
association:  ESTV-SERVER
policy:
  dst 193.137.7.2 port 110 \
  src 193.137.6.0/24 \
  xport-proto 6,17 \
  direction inbound permit
mnt-by:    CAL
changed:   CAL 20030523
signature: CAL CAL-CERT \
      KEYEDMD5 ABBA007AB47E
```

```
policy-name:  POL2
association:  ESTV-SERVER
policy:
    dst 193.137.7.3 port 80 \
    src * xport-proto 6 \
    direction inbound deny, \
    forward port 8080
mnt-by:   CAL
changed:  CAL 20030523
signature: CAL CAL-CERT \
      KEYEDMD5 ABBA007AB47E
```

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



mnt-by:    CAL
changed:   CAL 20030523
signature: CAL CAL-CERT \
      KEYEDMD5 ABBA007AB47E
```

Figure 11. A global policy written in SPSL

```
Rule Nº 1, POL1, assoc: ESTV-SERVER
ipchains -A input –p 6 –s 193.137.6.0/255.255.255.0 -d 193.137.7.2 110 –j ACCEPT

Rule Nº 2, POL1, assoc: ESTV-SERVER
ipchains -A input –p 17 -s 193.137.6.0/255.255.255.0 -d 193.137.7.2 110 –j ACCEPT

Rule Nº 3, POL2, assoc: ESTV-SERVER
ipchains -A input –p 6 -d 193.137.7.3 80 –j REDIRECT 8080

Rule Nº 4, DEFAULT, assoc: ESTV-SERVER
ipchains -A input –j REJECT
```

Figure 12. IPChains translation

The results for the Cisco router ACLs are presented in *Figure 13*. The translation of rules 4 and 5 is different from the translation to 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 Nº 1, POL1 , assoc: ESTV-SERVER
access-list 101 permit tcp 193.137.6.0 0.0.0.255 host 193.137.7.2 eq 110

Rule Nº 2, POL1 , assoc: ESTV-SERVER
access-list 101 permit udp 193.137.6.0 0.0.0.255 host 193.137.7.2 eq 110

Rule Nº 3, POL2 , assoc: ESTV-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 13. Cisco ACL translation

4.7 Evaluation

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

In the current implementation, the application only deals with packet filtering policies. This introduces limitations due to the lack of mechanisms to configure other equipment aspects. For instance, when configuring Cisco ACLs, we would also like to have the

possibility to describe the network functioning and generate the adequate router configuration.

The SPSL language revealed adequate for policy based management, making easier the expression of policies. However, we verified that this is not good enough, concerning policy specification for other network management areas.

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. A graphical interface that allows administrators to quickly get a general vision of existing policies is needed.

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. RELATED WORK

In this area some work related to firewall configuration using rule translation mechanisms is presented. The main difference of our system is that we also manage the distribution of rules among the equipments using the standard COPS-PR.

One system, among others, that has successfully implemented a conversion process from a set description to various firewall configurations is the Filter Compiler Language project (FCL) [Reed 2001]. This system uses the C pre-processor to execute the conversions.

Using FCL we can generate firewall configuration from a set of “if statements” and variable mappings like the one in *Figure 14*.

```
if ( in ) then {
set protocol tcp
if ( from host BAR and opening ) then {
block .
}
if ( from foo and to host bar ) then {
log body and block .
}
if ( to port 2049 ) then {
log and block .
}
pass .
}
```

Figure 14. FCL rule example

This system would not be applicable to a wide range of services because it doesn't use the concept of policy and doesn't do policy distribution among equipments. One objective of this project is to introduce the possibility of simulation.

We can also find the Firewall Builder [Kurkland 2001] where nodes and other network elements are described using XML descriptions, and a GUI editor is used to configure the firewall.

The entire configuration is made in the GUI and stored in XML files. The XML file is then translated to a firewall specific language, but there isn't a mechanism to send the translated rules to the firewall.

Another proposal is Network Management using Abstracted XML Specifications [Simon 2002] with the following main goals:

- To provide a method to allow users to gather and display structural details about nodes and links on their local area network;
- To allow the specification of network services and configuration for each required node on the network. The actual behaviour will be generalised so that it reflects the various different implementations and versions of that service available on different platforms;
- To store the state of the network layout and the service descriptions of each node to a series of XML files. These give an abstracted description of the system as a whole, allowing it to be restored and analysed at a later date.

In the proposal XML node descriptions can then be parsed and converted into sets of configuration files specific to the required operating system and service types. Again, one of the main advantages of our proposal when compared to this work is to allow communication with equipments with a standard protocol.

6. CONCLUSION AND FUTURE WORK

The work presented in this paper analyzed the applicability of the PBN approach to the management of network equipment and to firewalls in particular. As a general conclusion it can be said that the PBN architecture has some potential that can allow for cost reduction and improvement on performance, availability, security and QoS in computer networks.

However, the "plug and play" network configuration with a set of policies written in a high-level language is not, yet, a generalized solution. Many equipment manufacturers

provide solutions for their own equipment. The use of these proprietary solutions with the equipment of other manufacturers is still hard to do. The experimental 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 support PBN solutions. SPSL was the available choice as language 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.

7. 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.

[Dinesh 2002] Dinesh Verma, Simplifying Network Administration using Policy based Management, in IEEE Network Magazine, March 2002.

[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.

[Kurkland 2001] V. Kurland and V. Zaliva. Firewall Builder, (http://www.fwbuilder.org/), 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.

[RAP 2001] Resource Allocation Protocol (rap); (http://www.ietf.org/html.charters/rap-charter.html), 2001.

[Reed 2001] Darren Reed, Filter language compiler specification (http://coombs.anu.edu.au/~avalon/flc.html), 2001

[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.

[Simon 2002] Simon R. Chudley and Ulrich Ultes-Nitsche An XML-based Approach to Modelling and Implementing Firewall Configurations, in proceedings of ISSA 2002 Information Security conference, Muldersdrift, Gauteng, South Africa, July 2002

[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.