Universidade de Coimbra
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Informática

GESTÃO POR POLÍTICAS
APLICAÇÃO A SISTEMAS DE *FIREWALL*

Dissertação apresentada à Universidade de Coimbra,
para obtenção do grau de Mestre em
Engenharia Informática,
na área de especialização em Comunicações e Telemática

Filipe Manuel Simões Caldeira

Coimbra 2002

Dissertação elaborada sob a orientação do
Professor Doutor Edmundo Monteiro
Professor Auxiliar do Departamento de Engenharia Informática da
Faculdade de Ciências e Tecnologia da
Universidade de Coimbra

Agradecimentos

Muitos foram os que contribuíram, de forma directa ou indirecta, para a elaboração desta dissertação. Pretende-se, deste modo, expressar o reconhecimento e gratidão às pessoas e instituições cuja colaboração foi determinante para a sua concretização.

Ao Professor Doutor Edmundo Monteiro, orientador desta dissertação, um agradecimento destacado pelas valiosas sugestões e críticas pertinentes, bem como pelo constante incentivo e cordialidade.

À Escola Superior de Tecnologia de Viseu, pela oportunidade proporcionada de conciliação da actividade profissional de docência com a realização deste trabalho.

Aos familiares, amigos e colegas, pela amizade, apoio e colaboração prestados em diferentes momentos, em especial ao João Gil pelos muitos e bons jantares.

À Guida pelo apoio permanente e incondicional.

À Ana Filipa agradeço a paciência, por não ter estado mais tempo disponível.

Resumo

Nesta dissertação é feita uma abordagem à gestão de redes baseadas em políticas focando a arquitectura PBN (*Policy-Based Networking*) proposta no âmbito do grupo de trabalho *Policy Framework* do IETF (*Internet Engineering Task Force*). São evidenciados os principais aspectos desta arquitectura, desde os protocolos de comunicação até às linguagens de especificação de políticas, passando pelos modelos necessários à representação de informação.

Relativamente às linguagens de especificação de políticas, apresenta-se uma visão geral sobre a sua aplicabilidade na arquitectura PBN, sendo dado especial relevo à especificação de políticas referentes à área de segurança, através da descrição em pormenor da linguagem SPSL (*Security Policy Specification Language*).

No âmbito dos protocolos de difusão de política, destaca-se o protocolo COPS (*Common Open Policy Service*) e COPS-PR (*COPS for Policy provisioning*), os quais são alvo de estudo detalhado.

O documento termina com a descrição de uma aplicação de gestão de *firewalls* através do uso de políticas. Esta aplicação baseia-se na arquitectura de gestão por políticas (PBN) e aplica a linguagem SPSL e o protocolo COPS-PR.

Palavras chave: Segurança de redes, Gestão de redes por políticas, COPS, COPS-PR, SPSL.

Abstract

This dissertation describes a policy-based approach to network management. The Policy-Based Networking (PBN) architecture proposed by the Policy Framework Group of the 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 and their applicability to PBN architecture is presented paying particular attention to the specification of security policies through a detailed study of 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 studied in detail.

This document concludes with a description of an application of the PBN architecture to firewall management. This application uses the language SPSL and the COPS-PR protocol.

Keywords: Network security, Policy-Based Network Management, COPS, COPS-PR, SPSL.

Índice Geral

Índice de Figuras

Índice de Tabelas

1. Introdução

Este capítulo tem como principal objectivo efectuar o enquadramento do trabalho realizado na sua área de aplicação. São também apontadas as principais motivações subjacentes a este trabalho e apresentada a sua organização.

1.1 Enquadramento

Por gestão de sistemas informáticos e redes de computadores entende-se a execução de um conjunto de tarefas necessárias ao controlo, planeamento, coordenação e monitorização dos recursos existentes nos sistemas a gerir. Inclui funções como o planeamento inicial da rede, a sua configuração, o assegurar do seu normal funcionamento, a maximização da qualidade de serviço que é fornecida aos utilizadores (incluindo aspectos como produtividade, eficiência e garantia de serviço) e o assegurar da segurança dos sistemas geridos [ITS 1996].

A gestão de redes de computadores é um processo que inclui a configuração dos sistemas existentes e a sua monitorização ao longo do tempo, assim como a alteração da sua configuração de forma a permitir a utilização de novos serviços, aumentar o desempenho ou a resolução de problemas. Abrange todos os componentes de um sistema informático e de uma rede de computadores: *hardware* (incluindo todos os periféricos), *software* de sistema e aplicacional, infra-estruturas de comunicação e utilizadores do sistema.

Pode-se dividir a gestão de sistemas e redes em várias áreas funcionais, sendo que cinco dessas áreas estão definidas pela ISO (*International Standard Organization*)

[ISO 7498]: falhas, configuração, contabilização, desempenho e segurança – designados pelo acrónimo FCAPS (*Fault, Configuration, Accounting, Performance, Security*).

A gestão de falhas refere-se à detecção, resolução e isolamento de problemas da rede. A gestão da configuração destina-se à manutenção da configuração da rede (*hardware* e *software*), controlando todos os parâmetros que afectam o seu funcionamento normal. A gestão de contabilização está relacionada com a gestão de utilizadores no uso de recursos e serviços. A gestão de desempenho pretende maximizar o desempenho da rede, sendo que é uma área com fortes ligações a QoS (*Quality of Service*) e a factores como utilização de recursos, atrasos e perdas de pacotes [Hegering 1999].

A gestão de segurança tem por objectivo realizar a monitorização e controlo de todos os mecanismos que asseguram a segurança de um sistema ou rede de computadores. Os actuais sistemas de informação e as redes de computadores estão sujeitos a falhas de segurança cujos efeitos podem por em causa não só o normal funcionamento do sistema, mas também ter consequências mais graves ao nível da segurança da informação [Breton 1999]. Neste ambiente, existe uma forte necessidade de identificar os recursos do sistema ou rede que devem ser protegidos, controlar os acessos dos utilizadores a esses recursos, proteger e autenticar dados, utilizadores e comunicações, verificação e detecção de violações nas regras de segurança, etc [Hegering 1999].

Sendo uma actividade complexa, a gestão de segurança envolve áreas como a gestão de chaves (para autorização de acessos, encriptação de dados e autenticação de utilizadores), gestão de *firewalls*, criação de *logs* acerca de segurança, etc. Para além das vertentes descritas, a gestão de segurança deve também assegurar que existe uma elevada confiança por parte do utilizadores nos serviços disponibilizados.

Com o considerável crescimento em dimensão e complexidade das actuais redes telemáticas, as tarefas respeitantes à sua configuração, gestão e manutenção, têm vindo, gradualmente, a consumir mais recursos às organizações. Paralelamente, a complexidade dos próprios sistemas a gerir também sofreu um aumento considerável com o aparecimento de novas necessidades como, por exemplo, a garantia de qualidade de serviço ou a segurança na rede [Dinesh 2002].

Têm sido efectuadas, ao longo dos últimos anos, diversas tentativas para o desenvolvimento de novos métodos de gestão de redes. Os métodos tradicionais estão

centrados principalmente na gestão e configuração de equipamentos individuais, não tendo em consideração a estrutura de uma rede como um todo. Estes métodos mostraram-se satisfatórios até ao momento em que a tarefa de administração passou a consumir demasiados recursos nas organizações e as capacidades dos dispositivos aumentou. Actualmente está a ser feito um grande esforço para diminuir a complexidade crescente da administração de redes, recorrendo a novos paradigmas. O modelo de gestão baseado em políticas (PBN – *Policy Based Networking*) [Stevens 1999] pretende ser o resultado da mudança dos actuais mecanismos de configuração para um sistema integrado de gestão.

O modelo PBN pretende criar uma infra-estrutura que permita ao administrador um elevado grau de abstracção acerca das capacidades dos equipamentos que pretende gerir. Esta abstracção permite elevar o nível de controlo do equipamento para o nível da rede, representando também uma transição da gestão de equipamentos e serviços realizada de modo individual em cada equipamento, para a gestão de serviços e equipamentos efectuada no interior do próprio conceito de funcionamento da rede.

A gestão baseada em políticas pretende combinar todos os procedimentos e protocolos que permitem o controlo dos recurso disponíveis, de modo a que estes possam servir de suporte ao modelo de negócio da organização. A gestão baseada neste modelo, deve começar por identificar e conhecer a política global da organização e efectuar a sua aplicação nos sistemas e tecnologias de informação. Após a definição da política global, normalmente estabelecida a um nível hierárquico superior, a política pode então ser traduzida e aplicada nos sistemas e equipamentos. A Figura 1.1 pretende mostrar a natureza hierárquica dos passos necessários à definição de uma política.

Figura 1.1 – Hierarquia para definição de uma política [Follows 1999]

O objectivo a atingir com a gestão por políticas é a especificação de um conjunto de regras genéricas que podem ser aplicadas num ou em vários equipamentos de rede. Com as ferramentas apropriadas, estas regras genéricas, ou políticas, podem ser criadas, instaladas, monitorizadas, modificadas ou ainda eliminadas dos diversos equipamentos da rede com um pequeno custo operacional independentemente do tipo de equipamento ou fabricante dos equipamentos que constituem a rede. Para além da criação de políticas, este modelo de gestão pretende construir um sistema de informação uniforme que representa o modo como a rede deve funcionar.

Em termos de segurança, a heterogeneidade dos sistemas utilizados para a garantir numa rede de computadores leva a um aumento da dificuldade em definir uma política de segurança global. Para além da sintaxe usada por cada equipamento ser diferente, também o modo de "pensar" a segurança muda entre cada equipamento ou, pelo menos, entre cada marca de equipamento.

A maior dependência das organizações nos seus sistemas de informação e a ligação destes sistemas a outros exteriores, leva a novas necessidades em termos de segurança, implicando a especificação de uma política de segurança para toda a organização, na forma de uma declaração formal das regras às quais as pessoas a quem é dado acesso ao activo tecnológico e informação devem obedecer [Fraser 1997].

De acordo com [Fraser 1997], a política de segurança deverá existir para, entre outros factores, especificar os mecanismos pelos quais se podem atingir os requisitos de segurança, informar os utilizadores acerca dos requisitos obrigatórios em uso na organização e também para permitir estabelecer uma base a partir da qual se pode adquirir, configurar e auditar sistemas informáticos para atingir os requisitos de segurança pretendidos.

A definição de uma política de segurança deve envolver as seguintes entidades dentro de uma organização: responsáveis pela gestão da organização; administrador(es) da rede; representantes dos utilizadores; conselheiro legal (se apropriado) [Fraser 1997].

A declaração formal dessa política, deve definir claramente as responsabilidades dos utilizadores, administradores e gestores. Deve ser executável com ferramentas de segurança e prever sanções onde a prevenção não é tecnicamente viável. Deve ainda ser implementável através de procedimentos de administração dos sistemas O grau de

segurança implementado na rede é determinado pelas decisões expressas na política de segurança. De modo a tomar boas decisões, é necessário determinar quais os objectivos que se pretendem atingir, nomeadamente definir as acções que os utilizadores podem ou não levar a efeito na rede [Fraser 1997].

Os objectivos a atingir por uma política de segurança, são condicionados, de acordo com [Fraser 1997], pelos seguintes factos:

- Serviços oferecidos – Devem ser verificados os riscos que os serviços podem acarretar, tendo em conta as vantagens que trazem para a rede. Por vezes os riscos são superiores aos benefícios;
- Facilidade de uso – O sistema deve ser simples de usar sem descurar as questões de segurança. Deve ser encontrado um ponto de equilíbrio entre a facilidade de utilização do sistema e o grau de segurança implementado;
- Custo da segurança – Ao existirem vários custos associados à perda de informação, estes devem ser analisados tendo em conta o custo da segurança a implementar.

Um dos problemas encontrados na implementação de políticas é a falta de normas que permitam uma administração centralizada e homogénea de todos os elementos que implementam segurança numa rede. Nesta área da gestão de redes pode também ser aplicado o novo paradigma PBN.

Outro aspecto relacionado com a gestão de redes, prende-se com o facto de a Internet estar a mudar de um modelo de serviço onde todas as transmissões são consideradas iguais, sem nenhuma garantia de entrega, para um modelo que permita fornecer níveis de serviço com uma determinada qualidade previsível, específica para as exigências de cada serviço. Esta transição é necessária tanto pelas exigências de qualidade das comunicações necessária para as diversas aplicações emergentes, como pelo impulso do mercado para a diferenciação dos serviços [IPHighway 2000].

A introdução de serviços com garantia de qualidade de serviço, implica que os dispositivos da rede necessitem agora de efectuar uma distinção entre pacotes diferentes, efectuando um tratamento especifico para cada tipo de pacote, contrastando com as redes actualmente existentes que tratam todos os pacotes do mesmo modo.

O fornecimento de serviços com garantia de qualidade em redes com alguma dimensão está bastante dependente de uma estrutura de políticas que permitam, por exemplo, aos fornecedores de acesso à Internet (ISPs – *Internet Service Provider*) e aos administradores de grandes redes regulamentarem de uma forma geral a utilização das suas redes invés de configurarem cada equipamento individualmente.

Tendo em consideração os aspectos abordados, é sentida a necessidade de criar mecanismos que facilitem o trabalho do administrador de rede, permitindo ao mesmo tempo que a política de gestão de redes definida na organização seja posta em prática com maiores garantias de eficácia.

O trabalho objecto desta dissertação enquadra-se na área de gestão por políticas, efectuando uma aplicação deste paradigma para a área de segurança em redes, mais especificamente, na configuração de sistemas de *Firewall*.

1.2 Objectivos do trabalho

Pretende-se neste trabalho estudar a arquitectura de gestão por políticas PBN (*Policy Based Networking*) e a sua aplicação à gestão de redes de computadores, com especial detalhe na área de segurança. Assim, pretende-se analisar os mais recentes desenvolvimentos que têm sido levados a cabo sobre esta arquitectura, efectuando, para esse efeito, um estudo acerca dos seus principais componentes.

Para validar os conceitos estudados, pretende-se também propor, implementar e avaliar um sistema de gestão de *firewalls* baseado na arquitectura de gestão por políticas.

Com o intuito de conhecer e aprofundar os conhecimentos sobre os componentes da arquitectura PBN, são discutidas algumas das linguagens existentes para especificação de políticas, modelos para o seu armazenamento e protocolos para difusão dessas mesmas políticas.

A linguagem SPSL (*Security Policy Specification Language*) e o protocolo de difusão de políticas COPS (*Common Open Policy Service*) e COPS-PR (*COPS for Policy Provisioning*), serão alvo de especial atenção, pelo facto de serem usadas no protótipo desenvolvido.

Em síntese, pretende-se neste trabalho estudar o papel e as potencialidades da gestão por políticas na gestão de *firewalls*.

1.3 Organização

Relativamente à estruturação da dissertação, os seus capítulos são susceptíveis de ser agrupados em duas grandes áreas: uma primeira parte que pretende efectuar a apresentação de alguns dos paradigmas e tecnologias existentes na área e uma segunda parte descrevendo as tecnologias usadas no protótipo desenvolvido e a sua integração nesse mesmo protótipo.

Neste primeiro Capítulo apresenta-se, inicialmente, um enquadramento do trabalho efectuado sendo também apontadas as principais motivações que conduziram à criação desta dissertação. Seguidamente, faz-se uma exposição dos objectivos orientadores desta dissertação. A descrição da organização da dissertação, aqui a ser realizada, encerra este primeiro capítulo.

No segundo Capítulo, apresenta-se a arquitectura de gestão baseada em políticas (PBN), descrevendo os seus principais componentes. São também apresentadas algumas das vantagens decorrentes do uso deste modelo de gestão de redes.

No Capítulo 3 faz-se um estudo sobre dois dos modelos propostos para a criação de um repositório de políticas usado na arquitectura PBN: o modelo *Common Information Model* (CIM) e a estrutura de dados *Policy Information Base* (PIB).

No Capítulo 4 é feita a análise, no âmbito dos protocolos para a difusão de políticas, do protocolo COPS e do protocolo COPS-PR focando o seu modo de funcionamento.

No Capítulo 5 é feito o estudo de algumas das principais linguagens usadas para a descrição de políticas, apresentando em detalhe uma linguagem para descrição de políticas de segurança; linguagem SPSL (*Security Policy Specification Language*). Esta linguagem foi usada na implementação no protótipo construído.

No Capítulo 6 é apresentado um caso de estudo usando a arquitectura PBN, definida a arquitectura do sistema proposto e feita a descrição funcional dos aspectos considerados mais relevantes. Inclui-se ainda neste capitulo, a descrição da implementação do

protótipo desenvolvido aplicado à gestão de *firewalls*. Para concluir este capitulo é efectuada uma avaliação do trabalho desenvolvido no âmbito do seu enquadramento na vasta área que é a gestão por políticas.

Finalmente, no Capítulo 7, apresentam-se as conclusões do trabalho realizado assim como as perspectivas de trabalho futuro que a execução deste trabalho proporcionou.

2. Gestão por políticas

A arquitectura de gestão por políticas (PBN) é vista como uma proposta promissora para a gestão de redes. O conceito principal desta arquitectura é o uso de políticas, sendo estas políticas regras que determinam o funcionamento dos equipamentos existentes numa rede e consequentemente determinam o funcionamento global da rede. Neste capitulo, apresenta-se, com algum nível de detalhe, esta arquitectura e são descritos os seus principais componentes. São também discutidas algumas das vantagens obtidas com o uso desta arquitectura.

2.1 Arquitectura

A arquitectura de gestão por políticas (*Policy-Based Networking* – PBN) teve origem no trabalho desenvolvido pelo grupo *Policy Framework*[1] do IETF[2] (*Internet Engineering Task Force*) [Stevens 1999]. Este trabalho teve como principais objectivos, a administração centralizada da rede, suporte a uma definição abstracta das regras e políticas usadas, utilização das mesmas regras por vários equipamentos e automatização de tarefas de gestão de rede. A concretização destes objectivos permite uma maior consistência entre todos os elementos da rede. Neste conceito, o administrador de um sistema deve descrever o que a rede deve fazer, em lugar de se preocupar com o modo como isso deverá ser feito [Shepard 2000] [Mahon 1999].

[1] http://www.ietf.org/html.charters/policy-charter.html

[2] http://www.ietf.org/

A arquitectura de gestão baseada em políticas, definida pelo IETF [Stevens 1999], descreve, de um modo geral, as principais entidades existentes (Figura 2.1), sem contudo descrever nenhuma norma para a sua implementação. Esta arquitectura é composta por quatro entidades principais: consola de gestão de políticas, repositório de políticas, *Policy Decision Point* (PDP) e *Policy Enforcement Point* (PEP). A comunicação entre estas entidades é efectuada usando dois protocolos distintos: protocolo para acesso ao repositório e protocolo para difusão de políticas.

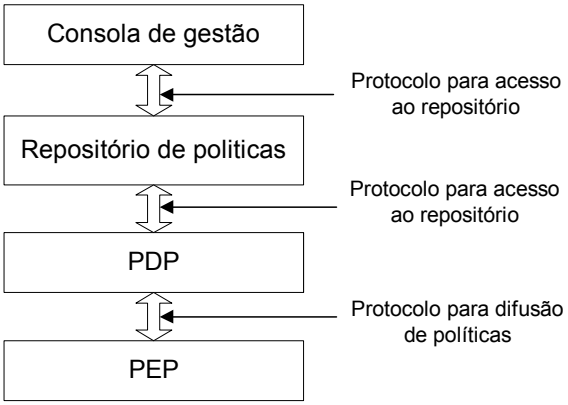

Figura 2.1 – Componentes da arquitectura PBN [INTAP 2001]

O PEP funciona normalmente nos equipamentos activos de uma rede, gerindo-os de acordo com instruções que recebe de um PDP. Um PDP efectua o processamento de políticas definidas para a rede, juntamente com outros dados relevantes para a administração da rede e toma decisões posteriormente transformadas em novas configurações para os PEPs. O repositório de políticas armazena o conjunto de políticas definidas dentro da organização. Nesta arquitectura o PDP acede ao repositório para obter todas as políticas que devem constar da sua configuração. A consola de gestão permite a edição, tradução e validação das políticas definidas de modo a poderem ser armazenadas no repositório de políticas e posteriormente postas em prática.

Usando a arquitectura PBN, o conjunto de políticas definidas pelo administrador é posto em prática pelos dispositivos de rede, após a configuração de cada PEP (Figura 2.2). A arquitectura PBN pode ser vista sobre duas perspectivas: O modelo de três camadas e o modelo de duas camadas [Raju 1999].

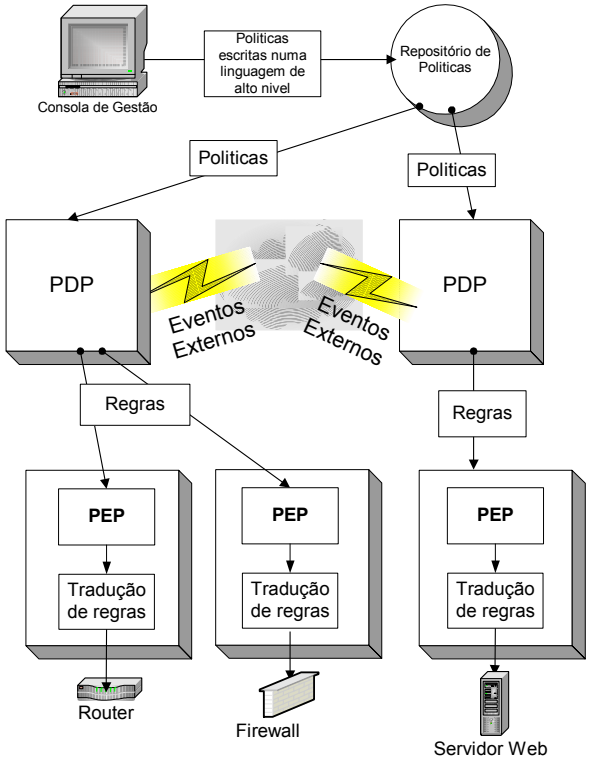

Figura 2.2 – A arquitectura PBN

A Figura 2.3 representa uma visão genérica do modelo de três camadas, sendo constituído por PEPs, PDPs e repositório de políticas. Neste modelo, o PEP é uma entidade que põe em prática as regras que recebe do PDP, não tendo contudo mais nenhuma capacidade de processamento.

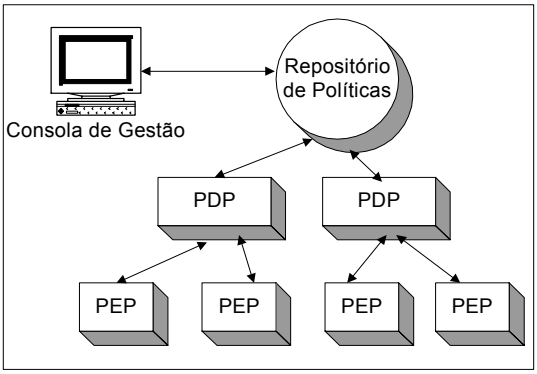

Figura 2.3 – Modelo de três camadas

O modelo de duas camadas apresentado na Figura 2.4 pode ser visto como uma simplificação do modelo de três camadas. Este modelo existe pelo facto de que alguns componentes da rede podem ser capazes de efectuar algum tipo de processamento, isto é, capazes de guardar regras e efectuar as suas próprias decisões. Como exemplo destes

equipamentos podemos considerar a maioria dos servidores de rede ou alguns *routers* que contenham um processador dedicado ao seu controlo de operação. Neste modelo, o PEP e o PDP estão integrados numa única entidade, que pode ser referida genericamente como "cliente de políticas".
Numa rede complexa é possível existir uma combinação destes dois modelos, existindo PDPs para os equipamentos com menos poder de processamento e a funcionalidade de PEP e PDP instalada nos servidores de rede.

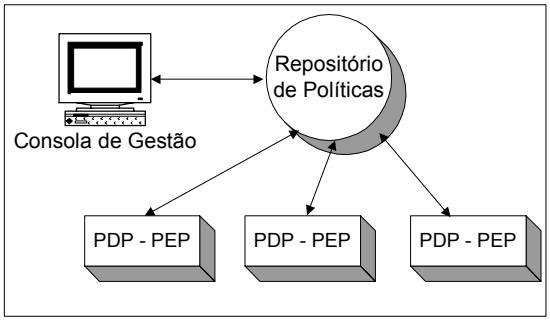

Figura 2.4 – Modelo de duas camadas

Os modelos *Outsourcing* e *Provisioning*

A arquitectura PBN efectua a interacção entre PDPs e PEPs usando o modelo cliente-servidor. Nesta arquitectura podem ser distinguidos dois modelos distintos de operação: *outsourcing* e *provisioning* [IPHighway 2001].

No modelo *outsourcing*, no caso de existir um evento que o PEP não saiba tratar de acordo com os critérios que tem instalados, é enviado um pedido ao PDP correspondente, notificando-o da ocorrência desse mesmo evento. O PDP responde ao PEP enviando informação que necessita de ser instalada de acordo com o evento. Este modelo também é conhecido por *pull* ou *reactive* [IPHighway 2001], visto que o PDP reage a eventos originados nos PEPs por ele controlados.

No modelo *provisioning*, quando o PEP efectua uma ligação inicial ao PDP este envia toda a informação existente, aplicável a esse mesmo PEP. Estas políticas são guardadas no PEP e todos os eventos que venham a ocorrer serão tratados usando esta informação. Podemos também chamar a este modelo *push* ou *proactive* [IPHighway 2001] pelo facto de o PDP enviar antecipadamente todas as políticas para os PEPs que controla.

Em ambos os casos os PDPs têm conhecimento das políticas postas em acção pelos PEPs pelo que pode decidir em qualquer altura efectuar uma actualização, instalando, apagando ou alterando a informação contida nos PEPs.

2.2 Componentes da arquitectura

Conforme foi referido, estão definidos na arquitectura PBN os seguintes componentes: consola de gestão de políticas, repositório de políticas, PDP, PEP, protocolo de acesso ao repositório e protocolo de difusão de políticas.

Consola para gestão de políticas

As políticas devem ser editadas usando uma ferramenta que permita ao administrador um grande nível de abstracção dos componentes da rede, assim como a descrição destas políticas deve ser feita numa linguagem de alto nível.

A consola de gestão deve fornecer uma interface para o administrador da rede utilizar o sistema de gestão por políticas. Deve integrar as funções de edição, tradução e validação de políticas. Com o editor de políticas, o administrador deve ser capaz de introduzir, visualizar e editar as políticas existentes no repositório. Deverá ser também possível, efectuar uma administração e monitorização do repositório de políticas, assim como do estado dos vários componentes do sistema. A representação das políticas pode tomar várias formas, dependendo da linguagem usada [Dinesh 2002].

Após uma política introduzida através do editor e antes de esta ser armazenada no repositório, deve ser efectuada uma validação contra possíveis conflitos com outras políticas já existentes. Paralelamente deve também ser efectuada uma tradução que permita transformar parâmetros de alto nível para parâmetros específicos da rede, por exemplo o nome de uma máquina deve ser transformado no seu endereço IP.

Uma tecnologia amplamente aceite para implementação do interface da consola de gestão é o uso de "*Web Browsers*", pois actualmente é uma interface com um largo espectro de utilização, mais do que os outros tipos de GUIs (*Graphic User Interface*). Este tipo de tecnologia está disponível na grande maioria dos equipamentos actuais sendo paralelamente uma tecnologia bastante bem aceite pelos utilizadores. A natureza

hipermédia deste tipo de interface permite uma boa apresentação e navegação em áreas como a definição de políticas de alto nível. Este tipo de tecnologia é independente da localização do utilizador e da plataforma em que deve funcionar, reduzindo significativamente os problemas de implementação do sistema. Num sistema baseado nesta tecnologia, as tarefas relacionadas com a gestão do sistema deixam de ser necessariamente efectuadas em equipamentos específicos, passando a poder ser efectuadas de qualquer local com acesso à rede.

Policy Decision Point (PDP)

Esta entidade é responsável por efectuar a tradução de políticas de alto nível para um nível inferior de abstracção, mais próximo do formato usado para configurar os equipamentos. O PDP recebe do repositório de políticas, a informação que foi definida através da consola de gestão e processa-a juntamente com outros dados que conhece, como por exemplo informação sobre o estado da rede. A combinação desta informação dá origem à configuração necessária a cada PEP que o PDP controla. A configuração de cada PEP é gerada de acordo com as capacidade de cada equipamento a que o PEP está associado. Sempre que existam alterações no estado da rede ou nas políticas existentes, o PDP pode ter que reajustar a configuração dos diversos equipamentos enviado nova informação aos PEPs [Burns 2001].

Policy Enforcement Point (PEP)

O *Policy Enforcement Point* (PEP) é o componente que lida directamente com os equipamentos existentes na rede, sendo responsável pela sua configuração, no que concerne às políticas definidas para cada equipamento que controla. Localiza-se normalmente associado aos dispositivos que efectuam algum tipo de tratamento ao tráfego da rede, como por exemplo *routers* ou servidores de rede. O PEP também pode ser visto como um componente operacional que executa acções como filtragem de pacotes, marcação de pacotes, gestão de recursos, etc, no caso de este estar integrado no equipamento activo da rede. Estas duas abordagens referem-se à aplicação dos modelos de duas ou três camadas da arquitectura de gestão por políticas.

O papel do PEP é também o de assegurar que os dados enviados pelo PDP que o controla são postos em prática no equipamento que lhe está subjacente, efectuando aqui

uma tradução destes dados para comandos reconhecidos pelo equipamento que controla. O PEP deve sempre obedecer aos comandos enviados pelo PDP, podendo também ele próprio enviar comandos e informação para o PDP, por exemplo no caso de a sua configuração física, ou dos equipamentos por ele controlados, sofrer alterações. Por exemplo, o caso de um *router* que passe a ter mais uma interface activa, esta informação deve ser enviada ao PDP de modo a que este possa ajustar as políticas que devem ser postas em prática por esse mesmo *router*, considerando agora as suas novas capacidades.

Repositório de políticas

O repositório de políticas é o local onde as políticas definidas para um determinado domínio estão armazenadas. Este repositório pode estar localizado num único dispositivo físico no interior do domínio que se pretende gerir, assim como pode estar replicado por vários equipamentos existentes na rede. O sistema usado para armazenar a informação pode ser um simples ficheiro de texto, uma base de dados armazenada num sistema de gestão de bases de dados ou um servidor proprietário. As políticas são armazenadas no repositório através da consola de gestão de políticas. Este componente é alvo de estudo detalhado no Capítulo 3 do presente trabalho.

Protocolos para difusão de políticas

A interligação dos componentes existentes na arquitectura de gestão por políticas implica o uso de um mecanismo que permita efectuar o intercâmbio de informação. Nesta arquitectura podemos identificar dois tipos distintos de necessidades de comunicação entre os seus componentes: a comunicação entre PDPs e PEPs e o acesso ao repositório de políticas [Condell 2000-2].

A comunicação entre PDPs e PEPs é normalmente efectuada usando protocolos especialmente desenvolvidos para o transporte de políticas. Este facto deve-se a uma tentativa para normalizar a comunicação entre equipamentos de fabricantes diferentes, de modo a atingir os objectivos do paradigma de gestão por políticas.

O protocolo COPS (*Common Open Policy Service*) [Durham 2000] foi desenvolvido para a difusão de políticas entre servidores e clientes. Originalmente desenvolvido para

responder às necessidades do modelo *outsourcing* da arquitectura PBN, rapidamente evoluiu para dar resposta às necessidades existentes no modelo *provisioning*, dando origem ao protocolo COPS-PR (*COPS for Policy Provisioning*) [Chan 2001].

No protocolo COPS-PR o cliente (PEP) recebe do servidor (PDP) toda a informação acerca das políticas, aplicáveis a esse cliente. Após receber as políticas, o cliente deverá implementar essas mesmas políticas sem a existência de acções adicionais do servidor. Caso existam alterações nas políticas existentes no servidor, este é responsável pela manutenção da informação existente no cliente.

A comunicação entre o PDP, consola de gestão e o repositório de políticas pode ser efectuada de várias formas, dependendo do modo como o repositório está implementado. Por exemplo, se o repositório é uma directoria de rede, a escolha da maioria dos fabricantes para efectuar esta comunicação é o protocolo LDAP (*Lightweight Directory Access Protocol*) [Yeong 1995]. Quando o repositório é um sistema de gestão de bases de dados, são usadas consultas SQL (*Structured Query Language*) normais para aceder aos dados pretendidos.

O protocolo Diameter [Calhoun 2001] é utilizado para a autenticação de utilizadores remotos quando pretendem efectuar uma ligação para a sua rede de origem. Existem também propostas para expandir este protocolo de modo a suportar o uso de políticas para QoS.

No Capítulo 4, do presente trabalho, apresentam-se detalhadamente os protocolos COPS e COPS-PR.

Linguagens para descrição de políticas

Na gestão baseada em políticas, o administrador necessita de uma ferramenta para descrever o comportamento que deseja que o seu sistema apresente. Essa descrição deve ser feita num formato independente dos equipamentos existentes na rede, assim como deverá permitir descrever um leque bastante alargado de equipamentos e acontecimentos. Neste contexto, o uso de uma linguagem genérica permite ao administrador representar essa mesma política independentemente do sistema de gestão que a deverá implementar. As linguagens de especificação de políticas são alvo de um estudo detalhado no Capítulo 5 deste trabalho.

2.3 Uso de Políticas

O conceito fundamental subjacente à arquitectura PBN é o uso de políticas que permitem descrever o funcionamento pretendido para a rede em geral e para cada elemento de uma rede em particular. Inovador nesta arquitectura, é o facto de as políticas descreverem objectivos e não procedimentos a efectuar.

Na metodologia de gestão tradicional, o administrador define determinados objectivos e de seguida cria um conjunto de procedimentos que implementem esses mesmos objectivos. Por exemplo se fosse pretendido negar o acesso de (e para) o exterior da sub-rede que serve a contabilidade de uma organização, o administrador iria criar algo similar ao representado na Figura 2.5:

```
If ((SourceIp matches 10.10.1.0/24) or (DestinationIp matches
10.10.1.0/24)) Deny
```

Figura 2.5 – Regra para filtragem de pacotes

Na arquitectura PBN , a política deve definir o objectivo a atingir (Figura 2.6):

```
Política 1:
     Domínio: rede_contabilidade
     Acção: Nega acesso ao exterior
```

Figura 2.6 – Política para filtragem de pacotes

Obviamente que neste caso o administrador terá que fornecer informação adicional que permita que a política possa ser interpretada. Neste exemplo, deveria definir o que significa rede_contabilidade e "nega acesso ao exterior", não estando esta informação definida explicitamente na política. Assim, se por exemplo a rede_contabilidade fosse alterada para 10.10.2.0/24, o administrador apenas teria que alterar esta situação num único local, ficando todas as políticas relativas a esta sub-rede automaticamente validadas.

Vantagens do uso de políticas

O uso de políticas para descrever objectivos invés de procedimentos faz com que os detalhes da rede estejam separados das políticas definidas. Esta abordagem tem algumas vantagens sobre as tradicionais técnicas de gestão de rede [3COM 1998-1]:

- Alto nível de abstracção – As políticas são escritas de uma forma abstracta, se possível numa linguagem de alto nível, de uma forma o mais independente possível da topologia da rede à qual se destinam. Devem também ser independentes dos protocolos, serviços e aplicações usadas. Qualquer administrador deve ser capaz de determinar o comportamento previsto da rede lendo as políticas definidas, mesmo não tendo sido o autor das mesmas. Alterações na topologia da rede, protocolos de comunicação, serviços, equipamentos ou aplicações não afectam a consistência das políticas definidas pois os objectivos continuam os mesmos;

- Automatização e consistência – A arquitectura PBN leva a uma grande automatização de tarefas. Esta mesma automatização dá origem a uma maior consistência no comportamento entre os vários equipamentos da rede e reduz de modo significativo o tempo dedicado à configuração dos vários equipamentos;

- Políticas dinâmicas – Pelo facto de as políticas estarem separadas dos detalhes da rede e a junção destes dois tipos de informação ser um procedimento dinâmico que ocorre nos servidores de políticas, sempre que a configuração da rede seja alterada, as políticas também o são de modo a reflectirem essas alterações. Este facto permite que novos tipos de políticas sejam definidos, fornecendo uma maior flexibilidade aos administradores da rede. Por exemplo, supondo que é definida uma política que permite ao administrador efectuar acessos à rede exterior independentemente do local de onde efectuou o seu *login*, o PDP será informado de que o administrador efectuou um *login* na máquina X e rescreve as regras de modo a permitir o acesso ao exterior de (e para) essa máquina. Este tipo de políticas são bastante difíceis de implementar usando um sistema de gestão tradicional.

O tráfego na rede pode ser configurado de modo a suportar as prioridades existentes na organização. Os gestores relacionados com as tecnologias de informação podem deste modo trabalhar com os seus parceiros de gestão de modo a determinar quais as aplicações criticas para o negócio, assim como as horas em que são usadas e por quem. Por exemplo, a aplicação de gestão integrada da organização pode requerer a mais alta prioridade de tráfego em toda a rede, enquanto que aos utilizadores que naveguem na *Web* durante o horário de trabalho é dada a prioridade mais baixa. As ferramentas de gestão por políticas devem oferecer esta possibilidade e definir prioridades para conjuntos de serviços e utilizadores.

Com o uso deste tipo de gestão é possível reduzir o custo de manutenção dos sistemas. Esta redução é conseguida através da automatização de tarefas referentes à administração dos sistemas, libertando o administrador para outras tarefas. Por outro lado permite também rentabilizar melhor os recursos existentes, por exemplo afectando-os às operações prioritárias da organização.

A produtividade dos gestores e dos utilizadores pode aumentar. Por exemplo, um "utilizador móvel" passa a ter a possibilidade de aceder aos recursos que pretende sem ter que estar sempre a realizar ou requerer tarefas de configuração sempre que altera a sua localização.

De um modo geral, a consistência e controlo do serviço prestado aos utilizadores é melhor. O gestor pode assim controlar os recursos e definir o seu uso, fazendo através de um ponto central, uma "gestão real" dos serviços tendo em conta apenas o comportamento que o sistema gerido deve apresentar.

A exposição do sistema gerido a riscos de segurança poderá ser minimizada devido a uma melhor implementação de mecanismos de segurança. A gestão de segurança pode ser efectuada de um modo bastante mais coerente, evitando algum tipo de "esquecimento" que pode acontecer na gestão individualizada de cada equipamento.

2.4 Síntese

Neste capítulo, apresentou-se a arquitectura de gestão baseada em políticas (PBN), descrevendo os seus principais componentes e os vários modelos existentes, nomeadamente os modelos de duas ou três camadas, *provisioning* e *outsourcing*. São também apresentadas algumas das vantagens decorrentes da sua aplicação na gestão de sistemas e redes de computadores.

No capitulo seguinte são discutidos modelos de dados com os quais pode ser implementado um repositório de políticas na arquitectura de gestão por políticas (PBN).

3. Repositório de políticas

Conforme apresentado no capitulo anterior, o repositório de políticas pode ser visto como sendo o local onde as políticas associadas a um determinado domínio estão armazenadas, ou, de uma forma geral, como uma base de dados contendo essa mesma informação.

Neste contexto, existem várias possibilidades para efectuar o armazenamento físico dos dados (ficheiros de texto, SGBD, etc.), contudo este armazenamento deve, necessariamente, obedecer a modelos de dados utilizados para modelar a informação. Neste capitulo são apresentados o modelo CIM (*Common Information Model*) e o modelo baseado na estrutura de dados PIB (*Policy Information Base*).

3.1 Common Information Model

As actividades existentes para criar estruturas que permitam armazenar a informação necessária à gestão por políticas, têm resultado em trabalhos genéricos e ainda algo difusos. O grupo de trabalho de gestão por políticas, *Policy Framework,* do IETF (*Internet Engineering Task Force*), está actualmente a trabalhar na definição de uma terminologia normalizada para as políticas e de um modo geral pretende criar uma arquitectura genérica para o armazenamento de políticas de uma rede. O IETF e o grupo *Directory Enabled Networking*[3] (DEN) da organização DMTF[4] (*Distributed Management Task Force*) estão a concentrar os seus esforços na criação de um modelo

[3] http://www.dmtf.org/standards/standard_den.php

[4] http://www.dmtf.org

de dados orientado a objectos, o *Common Information Model* (CIM) [Moore 2000]. Este modelo de dados pretende vir a ser a base para a representação de políticas referentes a diferentes áreas, como a qualidade de serviço ou a segurança em redes.

O principal objectivo do modelo CIM é a descrição dos dados necessários para efectuar a gestão de sistemas de informação assim como os relacionamentos existentes entre estes dados, estando definido usando a linguagem UML (*Universal Modeling Language*) [Booch 1996]. Esta abordagem define um modelo de dados e não uma linguagem, ou implementação, para a definição desses mesmos dados. Para este problema existem várias abordagens possíveis, não existindo ainda um amplo consenso sobre qual a linguagem que deve ser utilizada de uma forma generalizada [Tosic 1999].

O modelo CIM utiliza os conceitos do paradigma orientado a objectos para a descrição das entidades existentes no sistema, como por exemplo os computadores, *software*, utilizadores e redes. Permite a criação de um modelo de dados onde estão definidas de modo abstracto, entre outros, os seguintes dados: perfis de utilização e políticas, equipamentos, protocolos, serviços e utilizadores [CIM 1999].

Sendo baseado no paradigma orientado a objectos, existem no modelo CIM noções de classes, instâncias dessas classes, tipos de dados, herança, regras de associação entre objectos e métodos.

O modelo CIM está baseado numa arquitectura conceptual de três camadas ou esquemas de dados, sendo constituída por *core model*, *common model* e *extension model* (Figura 3.1).

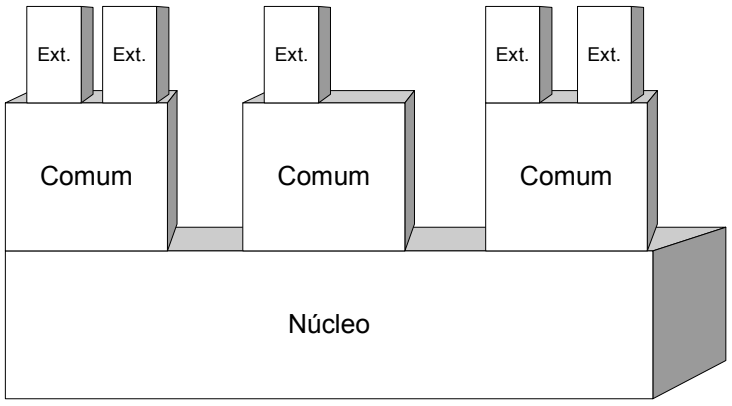

Figura 3.1 – Arquitectura conceptual do modelo CIM [Oliveira 2000]

A camada *Core Model* (núcleo), permite armazenar informação geral ou conceitos relevantes para todos os domínios de gestão. É constituída por um pequeno conjunto de classes, associações e propriedades que fornecem um vocabulário inicial para descrever os sistemas a gerir. Representa um ponto de partida para que o analista possa determinar como deve expandir a camada seguinte. Actualmente pensa-se que esta camada não deverá sofrer grandes alterações no futuro, no que respeita à definição de novos objectos [CIM 1999].

Na Figura 3.2 é apresentado a descrição UML simplificada dos principais objectos existentes nesta camada. O modelo é composto por diversas classes base, interrelacionadas através de relações do tipo "*is-a*" (herança), "*has-a*" (agregação) e "*uses-a*" (dependência funcional). Esta ultima relação, ao exprimir dependências, permite descrever algo como um objecto que não pode existir sem a existência de um outro [CIM 1999].

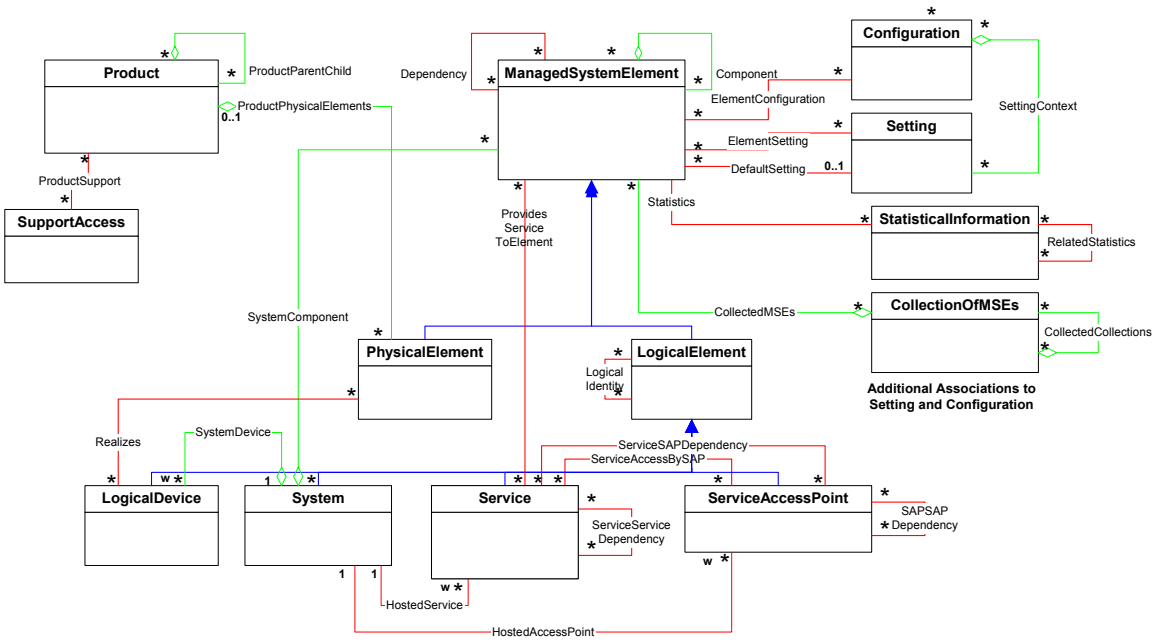

Figura 3.2 – Descrição UML simplificada do esquema "CIM Core" [Jennings 2001]

A classe *Managed System Element* é a classe base para a definição da hierarquia dos elementos do sistema, podendo representar sistemas, componentes de sistemas, *software* e redes que sejam de alguma forma significantes para a gestão do sistema. Qualquer componente independente de um sistema é candidato a ser incluído nesta classe caso a

sua gestão seja necessária. Esta classe é uma generalização de duas outras sub-classes: *Logical* e *Physical Elements*. Os modelos *Core* e *Common Models* permitem uma maior definição destas sub-classes, por exemplo inserindo elementos como sistema, dispositivo lógico, serviço, etc. [CIM 1999].

A outro nível, hierarquicamente mais baixo, considerando o modelo como uma árvore hierárquica, são definidos os modelos *Common* de modo a resolver questões específicas como a definição de redes, utilizadores e aplicações, entre outros.

Na Figura 3.3 é apresentado o modelo CIM v2.5, apresentando a camada *Core* e as camadas *Common* já definidas.

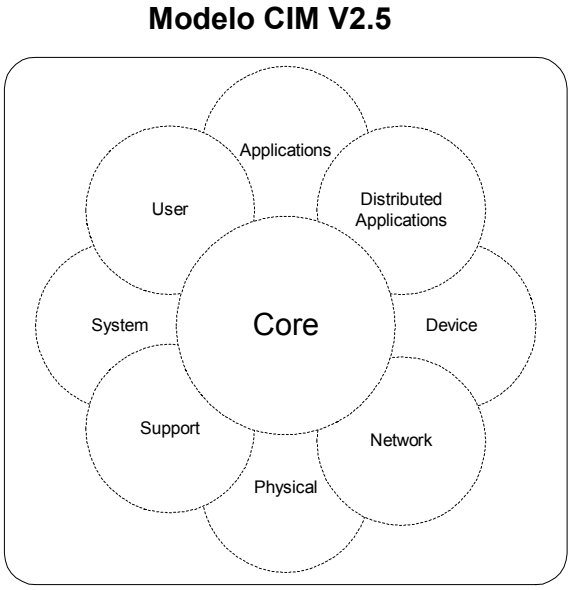

Figura 3.3 – Esquemas definidos no modelo CIM [Bumpus 2001]

Os esquemas *Common Models* (esquemas comuns), consistem num conjunto de classes destinadas a definir um modelo de dados para áreas mais específicas, sendo ainda independentes das várias tecnologias ou implementações. As áreas existentes vão desde os equipamentos, aplicações, redes utilizadores, até à gestão do suporte a utilizadores.

As classes, propriedades, associações e métodos definidos nestes esquemas pretendem fornecer uma visão global dos objectos a modelar em cada uma das áreas a que se destinam. Os esquemas *Core* e *Common* designam-se genericamente por esquema CIM.

Algumas das áreas já definidas encontram-se representadas na Figura 3.3. Não tendo como objectivo detalhar os modelos existentes em todas estas áreas, é apresentado o modo de funcionamento dos considerados mais relevantes [CIM 1999].

- Sistemas – O modelo comum para sistemas descreve os vários objectos de um sistema a gerir com um alto nível de abstracção. As representações mais usuais são de computadores de vários tipos, sistemas aplicacionais e sistemas de rede.
- Componentes – Este modelo é usado para representar elementos existentes num sistema, como a sua capacidade de armazenamento, capacidade de processamento e funcionalidades de comunicação de dados.
- Redes – As redes são representadas usando um modelo comum que modela os seus diversos aspectos. Isto inclui a topologia da rede, as suas ligações e os vários protocolos e serviços necessários ao funcionamento e acesso a essa mesma rede.

Na Figura 3.4 é apresentada, como exemplo de um modelo comum, a descrição UML do modelo comum para redes.

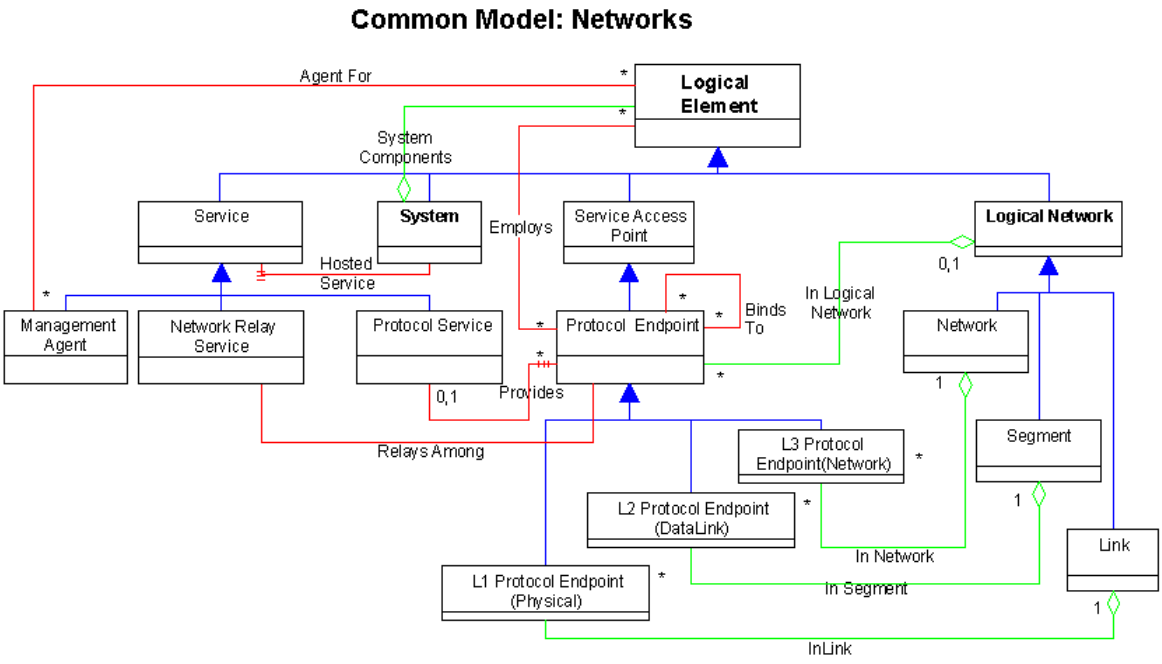

Figura 3.4 – Modelo CIM para descrição de Redes [CIM 1999]

Os *Extension Models* (Esquemas extendidos), fornecem extensões aos esquemas *Common* existentes, de forma a poder ser representada informação sobre ambientes específicos como por exemplo os sistemas operativos (*Unix*, *Microsoft Windows*, etc.).

De entre as diversas definições de esquemas *Common* e *Extension*, pode ser salientado o esquema relacionado com a definição de políticas. Este esquema é uma extensão CIM que permite a definição de objectos do tipo *Política*, permitindo aos administradores representar políticas relacionadas com diversas áreas: segurança, QoS, etc. Na Figura 3.5 apresenta-se, como exemplo, a descrição UML do esquema Políticas.

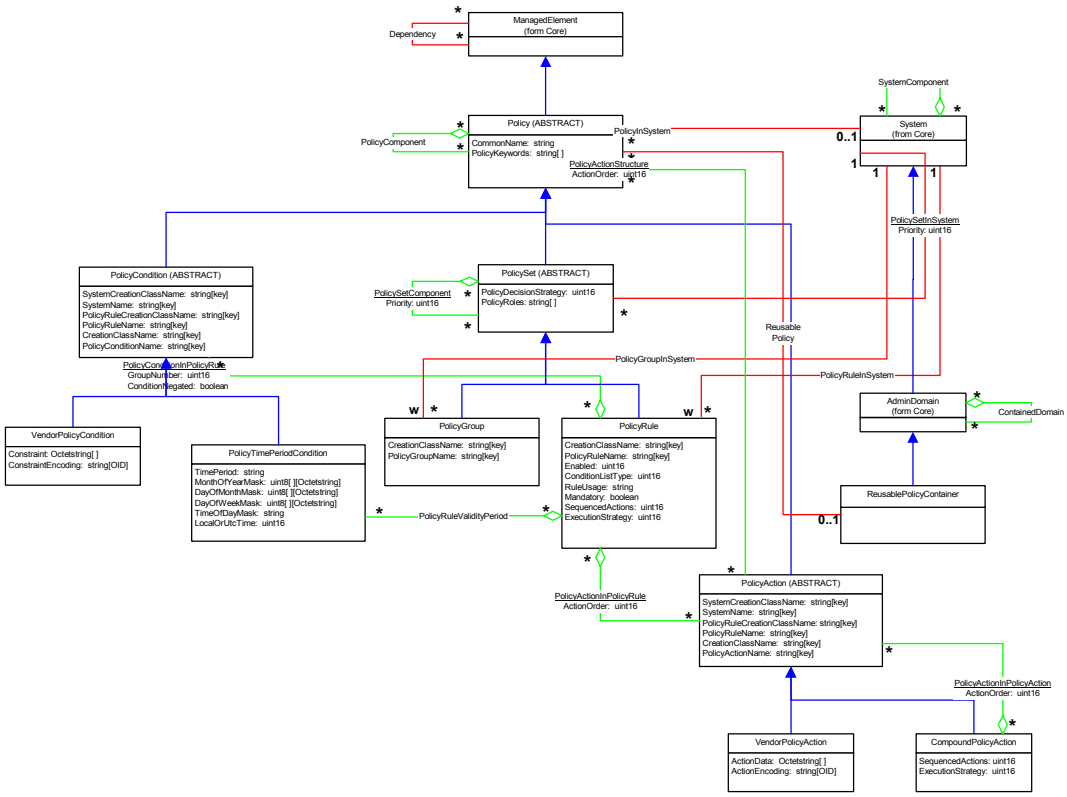

Figura 3.5 – Modelo CIM para descrição de Políticas [Jennings 2001]

O CIM é um modelo conceptual permitindo, assim, que a implementação de um modelo baseado neste conceito possa ser efectuada, por exemplo, usando XML (*eXtensible Markup Language*) ou mesmo uma base de dados relacional, podendo assim ser uma opção a considerar para a criação de um repositório de políticas numa arquitectura de gestão baseada em políticas.

3.2 Policy Information Base

O modelo de dados PIB (*Policy Information Base*) é apresentado no âmbito da sua utilização com os protocolos de difusão de políticas COPS (*Common Open Policy Service*) e COPS-PR(*COPS for Policy Provisioning*).

No protocolo COPS-PR, cada cliente tem que conter uma base de dados chamada *Policy Information Base* (PIB) [Fine 2000] onde deverá armazenar a informação recebida. Esta base de dados pode ter origem num modelo de dados mais abrangente como o CIM.

A estrutura de uma PIB é em tudo semelhante ao de uma MIB (*Management Information Base*) usada no protocolo SNMP (*Simple Network Management Protocol***)**, existindo inclusive mecanismos que permitem efectuar a conversão de uma PIB para uma MIB [Fine 1999]. A PIB contêm toda a informação estruturada de acordo com o tipo ou classe das políticas definidas. Uma PIB pode ser representada por uma estrutura em árvore (Figura 3.6), onde cada ramo da árvore corresponde a um diferente tipo de políticas ou classe de políticas (PRCs – *Policy Rules* ou *Policy Rule Classes*) e as folhas representam o conteúdo dessa mesma política – instâncias desse tipo de política (PRIs – *Policy Rules* ou *Policy Rule Instances*). Cada PRC pode conter múltiplas PRIs. Cada PRI é identificada por um identificador (PRID – *Provisioning Instance Identifier*), sendo que um PRID pode ser considerado como um nome único nos objectos COPS. Um exemplo de um PRID pode ser o nome "1.2.3.4.5" em que os primeiros quatro números representam a classe PRC ("1.2.3.4") e o ultimo número representa a PRI ("5").

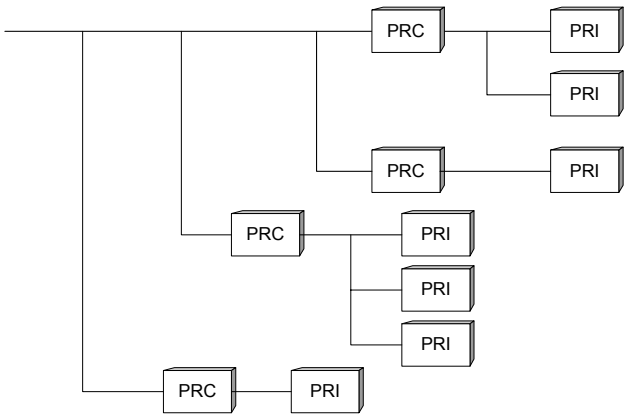

Figura 3.6 – Estrutura da PIB

De modo similar às MIBs usadas com SNMP, as PIBs usam a linguagem de definição de dados ASN.1 (*Abstract Syntax Notation One*) [MS 2000] e a informação deve respeitar as regras BER (*Basic Encoding Rules*) [MS 2000] de modo a que algum código SNMP possa ser reutilizado [Sanchez 1999].

As PIBs estão definidas no âmbito do protocolo COPS-PR apenas como estruturas abstractas, sendo que o detalhe referente a cada PIB (PRCs e semântica) estão especificados em documentos separados (como *internet drafts* ou documentos proprietários de determinado fabricante). São definidas diferentes PIBs de modo a cobrir diversas áreas de gestão, como segurança, QoS, etc.

A definição de uma PIB é efectuada com um elevado nível de abstracção, permitindo que os detalhes de implementação do *hardware* sejam escondidos. Desta forma, pode-se controlar o comportamento dos vários equipamentos existentes na rede, usando a mesma estrutura de dados para todos.

O PDP (*Policy Decision Point*) pode instalar novas PRIs ou alterar as PRIs existentes num PEP (*Policy Enforcement Point*), enviando as respectivas mensagens COPS. Pode ainda eliminar determinada PRI através de uma decisão que contenha o PRID do PRI que se pretende eliminar. As políticas são construídas através de um conjunto de PRIs existentes na PIB. Deste modo pelo facto de o PDP poder acrescentar, alterar ou remover PRIs de cada PEP ele pode implementar as políticas que pretende em cada equipamento.

É importante salientar que as políticas que cada PIB pode implementar estão pré-definidas no documento que define a PIB. De forma ao PDP poder controlar um determinado equipamento, este terá que conseguir mapear as políticas que lhe foram dadas através da consola de gestão, nas políticas que podem ser suportadas pela PIB em determinado PEP. As classes que são comuns a todas as PIBs estão definidas no documento "*Framework Policy Information Base*" [Fine 2000]. Esta PIB deve ser implementada em todos os clientes de COPS-PR.

O uso de PIBs permite uma grande abstracção dos dados, que passam assim a ser reconhecidos por todos os equipamentos existentes no domínio da rede em questão, deixando de existir a necessidade de criar novas políticas exclusivamente para determinado PEP, de um ou outro fabricante especifico. O uso de PIBs, COPS e COPS-

PR introduz um novo nível de abstracção, permitindo que o PDP não tenha conhecimento da arquitectura específica de cada PEP, desde que este tenha implementado algum tipo de suporte para as PIBs em uso e obviamente também suporte COPS e COPS-PR.

De modo a exemplificar o funcionamento de PIBs no âmbito do protocolo COPS-PR é apresentada uma pequena PIB que permite guardar informação sobre filtro de pacotes.

O cenário usado no exemplo, pode ser considerado como parte de uma rede numa organização com a seguinte topologia (Figura 3.7):

- Uma sub-rede publica (10.1.1.*), contendo servidores públicos (servidores de ficheiros, *ftp*, *www*, etc.) e alguns servidores para administração (servidor de autenticação, PDP, etc.);
- Uma sub-rede dedicada aos administradores (10.1.2.*), acedida exclusivamente por administradores;
- Três sub-redes pertencentes a salas de trabalho (10.1.11.* - 10.1.13.*);
- Um *router* que efectua a ligação entre estas sub-redes.

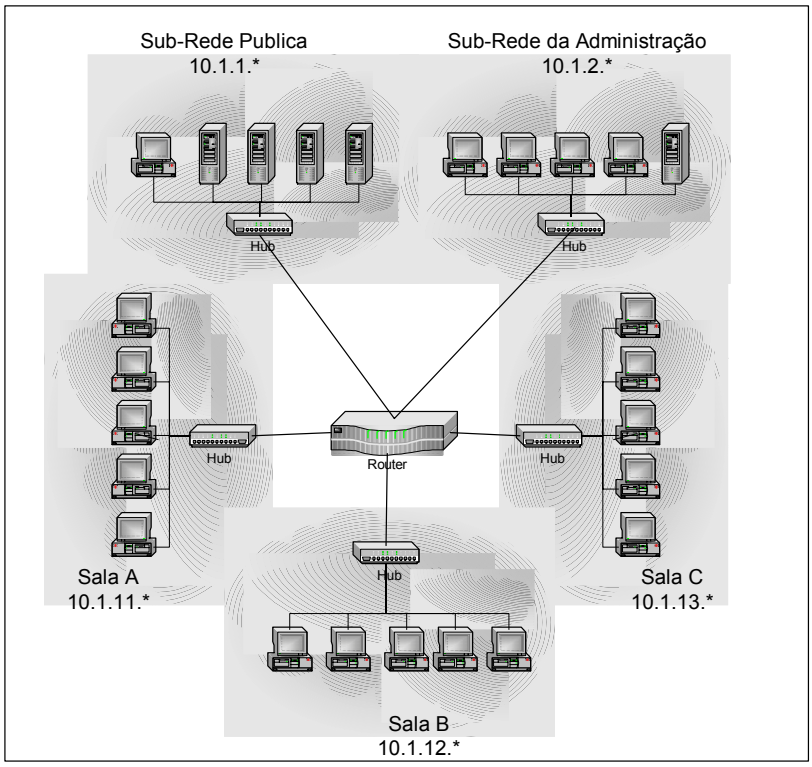

Figura 3.7 – Topologia da rede apresentada como exemplo

A política definida para a organização, determina as seguintes regras para administração da rede:

- As máquinas das salas de trabalho têm acesso à rede publica durante o período de almoço: Entre as 13 e as 14 horas. A sub-rede dos administradores deve ter sempre acesso à rede publica;
- Um administrador pode alterar a regra anterior e permitir o acesso a uma determinada sala para além do período inicialmente definido. Pode usar qualquer máquina das salas de trabalho e obter acesso à sua sub-rede;
- O acesso dos funcionários está restrito à sub-rede da sua sala trabalho e à rede publica;
- O tráfego do servidor de multimedia (10.1.1.5) é negado para as salas de trabalho sempre que exista congestionamento da rede.

Assume-se que num determinado momento o estado da rede é o seguinte: A hora é 11h:55m; um administrador está ligado numa máquina com o IP 10.1.11.5; o acesso de (e para) a sala A foi permitido durante a manhã (09h-13h); existe congestionamento de tráfego no *router*.

Suponhamos que o PEP subjacente ao *router* existente na rede suporta uma PIB com um único PRC. As PRIs desta PIB descrevem a origem e destino da comunicação como forma de filtragem de pacotes no interior da rede.

Uma PRI nesta PIB é uma regra no formato apresentado na Figura 3.8.

```
IF ((Src matches Srcaddr/Srcmask) and (Dst matches Destaddr/Destmask))
THEN Permit/Deny (Priority: P)
```

Figura 3.8 – Regra genérica usada no cenário apresentado

De modo a atingir o comportamento pretendido para a rede, o PDP terá que modificar a PIB para o formato apresentado na Figura 3.9. As prioridades definidas para cada PRI são escolhidas pelo PDP de modo a que o comportamento pretendido seja atingido.

O tráfego que não corresponde a nenhuma das regras definidas é negado.

Index	SrcAddr	SrcMask	DstAddr	Dstmask	Permit	Priority	
1	10.1.2.*	24	10.1.1.*	24	1	250	// Administradores → Rede Publica
2	10.1.1.*	24	10.1.2.*	24	1	250	// Rede Publica → Administradores
3	10.1.1.5	24	*.*.*.*	24	0	200	// Trafego Multimédia
4	10.1.11.*	24	10.1.1.*	24	1	100	// A → Rede Publica
5	10.1.1.*	24	10.1.11.*	24	1	100	// Rede Publica → A
6	10.1.11.5	24	10.1.2.*	24	1	50	// Comp. Administ. → Administ.
7	10.1.2.*	24	10.1.11.5	24	1	50	// Administ. → Comp. Administradores

Legenda	
Index	Índice da PRI
DstAddr	Endereço IP de destino
DstMask	Mascara do endereço de destino
SrcAddr	Endereço Ip de Origem
SrcMask	Mascara do endereço de origem
Permit	1=Permite / 0=Negar
Priority	0:Baixa / 255:Alta

Figura 3.9 – Exemplo de PIB resultante do cenário apresentado

O exemplo apresentado implica a existência de mecanismos adicionais que permitam ao PDP ser informado de eventos relacionados com o comportamento da rede. Por exemplo, pode existir um *script* que informa o PDP sempre que um administrador efectua um *login* numa máquina situada na rede dos funcionários. Paralelamente, o PDP pode usar o seu relógio, um serviço de relógio da rede ou um qualquer sinal externo para verificar se o evento "hora de almoço" está a ocorrer ou não.

O exemplo anterior demostra de um modo geral o modo como o PDP transforma as políticas definidas para a rede incluindo dados sobre o estado da rede, de modo a gerar a configuração apropriada para enviar aos PEPs.

Neste caso, podemos verificar que a inteligência do sistema está concentrada no PDP. O PEP limita-se a executar os comandos recebidos no equipamento que gere, não tomando decisões baseadas em eventos, mesmo que estes eventos já tenham ocorrido anteriormente (não é o caso em que as decisões a tomar constam na sua PIB).

Por exemplo, durante a hora de almoço, o acesso das salas de trabalho para o exterior deve ser permitido. De modo a implementar esta regra, todo o conjunto de PRIs que contêm esta regra devem ser enviadas para o PEP. Neste caso informar o PEP que o intervalo de almoço começou ou acabou, não é suficiente. O mesmo acontece quanto um administrador termina o seu trabalho numa determinada máquina da rede dos funcionários: O PEP necessita de receber todas as instruções provenientes do PDP, mesmo que isso já tenha acontecido no passado. Neste modelo, sempre que o estado da rede sofre alterações, o PDP deve instalar e/ou desinstalar em cada PEP as PRIs correspondentes, mesmo que este facto já tenha ocorrido no passado.

No modelo COPS-PR com este tipo de PIB, o PEP baseia o seu funcionamento no PDP. Caso aconteça uma falha no PDP ou apenas na comunicação entre PEP e PDP, o PEP mantêm uma PIB desactualizada o que pode não ser desejável em algumas situações. Por exemplo, no cenário apresentado, uma falha no PDP durante a hora de almoço poderia fazer com que a restrição de acesso à rede publica não fosse aplicada no final do intervalo para almoço. Outra situação de falha no PDP poderia fazer com que uma máquina que estava a ser usada por um administrador mantive-se o seu acesso à rede dos administradores mesmo que este já tivesse terminado a sua utilização.

Existe uma tentativa para limitar alguns destes problemas, usando o protocolo COPS-PR com meta-políticas [Boutaba 2001].

As meta-políticas são regras com o seguinte formato:

```
if (condição) then {acções}
```

onde "condição" é uma expressão lógica e "acções" é um conjunto de comandos pré-definidos semelhantes aos definidos numa PIB em COPS-PR.

Como as acções já contêm especificação de regras, uma regra do tipo meta-política é uma regra que indica como é que as políticas devem ser aplicadas. Estas meta-políticas são geradas pelo PDP e enviadas para o PEP. O PEP verifica as condições de cada meta-política e caso a condições seja verdadeira, aplica as acções correspondentes. Ambos os atributos "condição" e "acções" podem conter parâmetros, como por exemplo "congestionamento da rede", "data/hora" ou "IpAdministrador". Estes parâmetros são então verificados pelo PEP sempre que necessita de aplicar regras.

Com o uso de meta-políticas, o PEP não necessita de ser informado do modo como deve reagir a uma alteração ao estado da rede, precisando apenas de ser informado desse mesmo estado recebendo valores dos parâmetros incluídos nas meta-políticas. Por exemplo, se o PEP estiver a gerir um *router*, este pode obter o parâmetro "congestionamento de rede" directamente desse mesmo *router* e tomar as decisões necessárias sem o envolvimento directo do PDP.

O modelo de PIB com meta-políticas, está actualmente em estudo [Boutaba 2001] e deverá ser implementado como funcionalidade adicional nos sistemas futuros.

3.3 Síntese

O repositório de políticas é um componente importante da arquitectura de gestão por políticas. Genericamente descrito como sendo uma base de dados que mantém informação actualizada sobre as políticas definidas para todo o seu domínio de aplicação, este repositório necessita de estar construído segundo um modelo de dados que permita normalizar essa mesma informação.

Neste capitulo foram estudadas duas formas de modelar essa informação através do uso do modelo CIM (*Common Information Model*) e do modelo baseado na estrutura de dados PIB (*Policy Information Base*).

No capitulo seguinte discutem-se os protocolos utilizados na difusão de políticas, com especial detalhe no protocolo COPS (*Common Open Policy Service*) e COPS-PR (*COPS COPS for Policy Provisioning*).

4. Protocolos de difusão de políticas

Na arquitectura PBN, independentemente de se usar o modelo de duas ou três camadas e o modelo *provisioning* ou *outsoursing*, existe a necessidade de utilizar protocolos normalizados para os dispositivos efectuarem a troca de informação entre os vários componentes da arquitectura. Estes protocolos são necessários de modo a permitir uma comunicação sem restrições entre produtos de diversos fabricantes e assegurar que a solução para a gestão por políticas tenha uma âmbito de aplicação bastante alargado.

Neste contexto pretende-se, neste capitulo, dar a conhecer o protocolo COPS (*Common Open Policy Service*) e o protocolo COPS-PR (*COPS for Policy Provisioning*), focando o seu modo de funcionamento conforme é apresentado em [Durham 2000] e [Chan 2001].

4.1 Common Open Policy Service

O protocolo COPS (*Common Open Policy Service*) [Durham 2000] teve origem no trabalho realizado pelo IETF (*Internet Engineering Task Force*) com o objectivo de criar uma nova norma para efectuar a comunicação entre os PDPs (*Policy Decision Points*) e os PEPs (*Policy Enforcement Points*). Este protocolo e as suas extensões está a ser desenvolvido pelo grupo de trabalho do RAP[5] (*Resource Allocation Protocol*)

[5] http://www.ietf.org/html.charters/rap-charter.html

[RAP 2001] com especial incidência para a área do RSVP (*Resource Reservation Protocol*) [Braden 1997], contudo, devido à sua arquitectura genérica, este protocolo está a captar o interesse de outros grupos a trabalhar nas mais diversas áreas.

O protocolo COPS pode ser dividido em três camadas distintas: o protocolo base, os comandos específicos do tipo de cliente e a representação dos dados.

O protocolo base define os mecanismos de comunicação que permitem a troca de informação entre um PDP e os PEPs que lhe estão associados. O PEP fica com o papel de cliente e o PDP é o servidor. Diferenciando-se do protocolo SNMP, o protocolo COPS está baseado em ligações TCP (*Transmission Control Protocol*) e é um protocolo baseado em estados, significando isto que o servidor guarda o estado do cliente ou clientes, reagindo sempre que necessário de modo apropriado, mesmo que não tenha existido uma solicitação para o efeito.

O conceito de tipo de cliente existente no protocolo COPS, permite a adição de um segundo nível constituído por comandos específicos de cada cliente, permitindo assim a sua utilização em diversas áreas com a criação de novas camadas deste tipo. A principal diferença entre as camadas definidas para cada área específica é o tipo de mensagens que são trocadas entre o cliente e o servidor, bem como os dados que essas mesmas mensagens transportam.

O protocolo COPS é bastante flexível no sentido que permite suportar objectos definidos para vários contextos. Neste protocolo cada cliente suporta uma estrutura de dados própria (PIB – *Policy Information Base*) contendo também esta estrutura, objectos comuns a todos os clientes e contextos.

4.1.1 Funcionamento

O protocolo COPS é usado para difundir as políticas a implementar na rede. No COPS, cada PDP pode ter um ou mais clientes de tipos diferentes (diferentes áreas de aplicação); cada tipo de cliente pode representar uma área diferente de especificação de políticas (segurança, QoS, controlo de admissão, entre outras.). Ao suportar o tipo de cliente apropriado, o PDP fornece um modo de controlar os vários aspectos de configuração de um determinado equipamento.

O protocolo COPS dispõe de procedimentos de sincronização entre os PDPs e os PEPs, utilizando para tal troca de mensagens definidas para o efeito, assim como tem instruções de como reagir se a ligação se perder. Para além destes aspectos, este protocolo define mecanismos que asseguram a segurança e integridade das mensagens que são trocadas.

A semântica ou o formato dos dados transportados pelo protocolo COPS não estão definidos no âmbito do protocolo. O COPS fornece apenas um meio para trocar informação sendo que a definição do formato dos dados a serem enviados e recebidos é definida individualmente para cada tipo de cliente existente.

De acordo com [Durham 2000] podem-se identificar as seguinte características principais no protocolo COPS:

- Funciona segundo o modelo cliente/servidor, no qual o PEP envia pedidos, informa alterações e remoções de informação para o PDP. O PDP devolve decisões ao PEP;
- Usa o protocolo TCP para o transporte das mensagens;
- Foi criado com o objectivo de ser extensível. Desta forma pode suportar novos tipos de clientes sem existir a necessidade de efectuar alterações ao nível do protocolo;
- Fornece mecanismos de segurança para efectuar autenticação, protecção e controlo de integridade nas mensagens enviadas. Podem também ser usados protocolos já existentes como o IPSec (*IP Security Protocol*) [Keromytis 1999] e TLS (*Transport Layer Security*) [Dierks 1999] de forma a garantir uma ligação segura entre servidor e clientes;
- É baseado em estados pelo facto de o estado de um pedido ou de uma decisão ser do conhecimento tanto do servidor como do cliente e o estado de vários eventos (pedido/decisão) poder estar interligado, permitindo aos participantes na comunicação uma melhor gestão da comunicação e da informação partilhada. Adicionalmente, podemos dizer que o protocolo está baseado em estados porque este permite ao servidor enviar informação de configuração para o cliente, podendo de seguida em qualquer altura remover total ou parcialmente essa mesma configuração sempre que já não seja aplicável ao contexto em causa.

Na Figura 4.1 encontra-se representado um modelo COPS genérico. Neste exemplo é usado o protocolo COPS para efectuar a comunicação de políticas entre um PEP e um PDP remoto no âmbito do contexto de um determinado tipo de cliente. O PDP Local (LPDP) é um componente opcional usado com o objectivo de manter um conjunto de regras activas no PEP, funcionando mesmo na ausência do PDP. O PEP pode comunicar com um servidor de políticas de forma a obter decisões ou directivas, sendo responsável por iniciar uma ligação TCP persistente para o PDP. O PEP usa então essa ligação TCP para enviar pedidos e receber decisões provenientes do PDP.

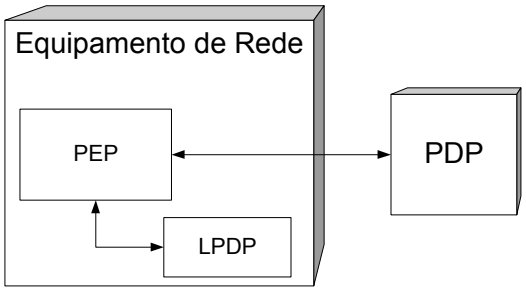

Figura 4.1 – COPS - Modelo genérico

Troca de mensagens

No protocolo COPS, quando o PEP é inicializado, inicia uma ligação a um PDP identificando-se e informando das suas capacidades e limitações. No modelo *outsourcing*, se o PEP recebe um evento que não sabe como tratar, envia um pedido ao PDP, pedindo instruções sobre o modo como tratar o evento. No modelo *provisioning*, o PEP regista a sua configuração junto do PDP após o que este envia as políticas que este PEP em particular deve por em prática. Em ambos os casos, o PDP pode em qualquer altura conhecer a informação existente em cada PEP que lhe está associado e efectuar alterações sempre que necessário.

Em caso de quebra de ligação COPS, o PEP guarda a informação já enviada num PDP local (LPDP) e tenta restabelecer a ligação ao PDP.

Após o PEP efectuar o pedido de uma nova configuração, o PDP envia uma mensagem do tipo *decision* contendo toda a informação que o PEP necessita. Após a decisão do PDP ser aplicada no PEP, este envia um relatório (*Report*) com a confirmação da instalação dessa decisão.

Na Figura 4.2 está representado o funcionamento das mensagens COPS.

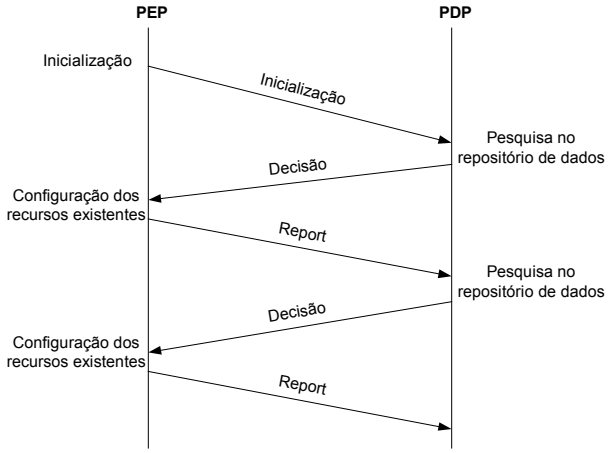

Figura 4.2 – Mensagens COPS – Funcionamento genérico

As comunicações entre o PEP e o PDP são, de um modo geral, baseadas na troca de mensagens do tipo pedido/decisão apesar de que ocasionalmente o PDP possa enviar informação não solicitada para o PDP de modo a forçar alterações em políticas previamente instaladas. A cada par pedido/decisão em COPS, podemos dar o nome de estado, sendo que cada pedido diferente dá origem a estados diferentes do sistema. O PDP também tem a capacidade de relatar ao PDP que implementou com sucesso as decisões que recebeu, sendo esta informação particularmente útil para efeitos de monitorização. O PEP é responsável por notificar o PDP quando um determinado pedido para configuração sofreu alterações. O PEP é também responsável pela remoção de algum estado (pedido/decisão) que já não seja aplicável devido a eventos ocorridos no cliente ou devido a novas decisões enviadas pelo servidor.

Quando um PEP emite um pedido de configuração este espera que o PDP envie um ou mais conjuntos de dados que permitem configurar o PEP para a situação em que foi efectuado o pedido. Estes dados são enviados através das mensagens implementadas no protocolo. Sempre que cada um destes dados é instalado no PEP, este envia uma mensagem do tipo *Report* confirmando a sua instalação. O servidor pode após receber o *Report*, caso pretenda, efectuar uma alteração ou remoção desses dados através de uma nova mensagem de decisão. Do mesmo modo sempre que pretenda remover determinada informação do PEP, este apaga a informação especificada e envia, como confirmação, uma mensagem de *Report* para o PDP.

O protocolo foi especificado de forma a poder comunicar objectos auto- identificativos, contendo os dados necessários à identificação dos pedidos de forma a permitir implementar, entre outros, os seguintes aspectos: estabelecimento do contexto de um pedido, identificação do tipo de pedido, referência a pedidos e respostas instaladas anteriormente, passagem de decisões, relatório de erros, controle de integridade das mensagens e transferência de informação genérica entre o cliente e o servidor.

De forma a suportar diferentes tipos de clientes, existe em cada mensagem do protocolo um atributo com informação acerca do tipo de cliente, pois clientes diferentes podem necessitar de diferentes tipos de decisões. Pretende-se que a cada novo tipo de cliente exista uma especificação referente à sua interacção com o protocolo.

O contexto de cada pedido corresponde ao tipo de evento que lhe deu origem. O objecto *Cops Context* identifica o tipo de pedido e mensagem que geraram um evento, acedendo aos atributos *message type* e *request type*.

O protocolo COPS identifica três tipos de eventos: a chegada de uma mensagem; alocação de recursos locais e o envio de uma mensagem. Cada um destes eventos requer que diferentes decisões sejam tomadas. Neste sentido, o conteúdo de uma mensagem COPS do tipo pedido/decisão deve depender do contexto.

O PEP pode ainda ter a capacidade de tomar decisões próprias usando o seu PDP local (LPDP), continuando apesar disso a ser o PDP a máxima autoridade no sistema. Neste caso, todas as decisões locais que sejam consideradas de alguma forma relevantes devem ser comunicadas posteriormente ao PDP, pois este manter uma informação actualizada sobre o sistema de modo a poder tomar decisões que afectam a sua globalidade

Devido ao facto de que este protocolo está normalmente associado a gestão de equipamentos, nomeadamente ao nível de gestão de segurança, são implementados mecanismos que permitem efectuar gestão de falhas, fazendo com que o PEP e o PDP verifiquem, com intervalos regulares, a ligação existente, através de mensagens do tipo *Keep-Alive.* Quando existir uma falha de ligação, o PEP deve tentar efectuar uma nova ligação ao PDP ou tentar efectuar a ligação para um servidor alternativo que esteja descrito na sua configuração. Quando a ligação for restabelecida, o PEP deve informar o PDP dos eventos ocorridos durante o período em que a ligação esteve cortada.

Paralelamente, o PDP pode efectuar um pedido de sincronização da informação existente entre este e o PEP.

A implementação do protocolo no PDP deve permitir ao sistema esperar ligações numa porta conhecida (*well-known TCP port*). No caso do protocolo COPS é usada a porta 3288. O PEP é responsável por iniciar a ligação ao PDP. A localização do PDP pode ser configurada no PEP ou obtida através de algum mecanismo de localização de serviços.

Um único PEP pode enviar várias mensagens *Client Open*, cada uma referente a um tipo de cliente, usando uma ou mais ligações TCP para o mesmo ou diferentes PDPs (Figura 4.3). Do mesmo modo, um PDP a funcionar num determinado endereço e porta, pode suportar vários tipos de clientes. Fornecendo-lhe o tipo de cliente, este tem a capacidade de o aceitar, rejeitar ou reencaminhar para outro PDP. Podem ser usadas portas diferentes para a existência de vários PDPs no mesmo equipamento.

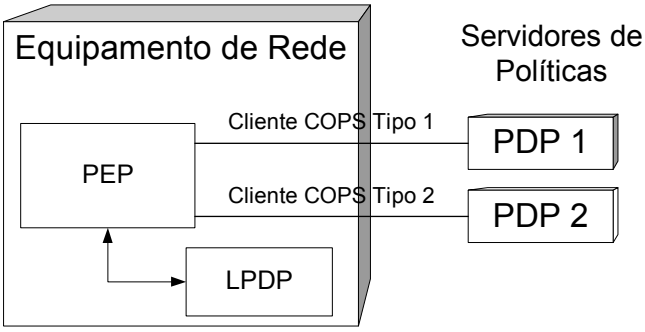

Figura 4.3 – Exemplo de múltiplos PDPs

Na especificação do protocolo COPS é usado um identificador (*handle*), escolhido pelo PEP, para permitir reconhecer individualmente cada estado de um PEP por cada tipo de cliente que esse PEP use. O PDP usa o identificador de modo a identificar univocamente o pedido efectuado por um cliente em particular, numa determinada ligação TCP. Estes identificadores são inicializados aquando de uma mensagem *Request* e são usados posteriormente nas mensagens seguintes. Estes identificadores têm obrigatoriamente que ser únicos no contexto de uma ligação e de um tipo de cliente.

Este protocolo disponibiliza um objecto de forma a garantir a autenticação e verificação da integridade das mensagens que são transmitidas do cliente para o servidor e do servidor para o cliente. De modo a assegurar que o cliente (PEP) está a comunicar com o servidor (PDP) correcto é necessário efectuar uma autenticação destes dois sistemas

usando uma chave secreta, por exemplo com o uso de PKI [Branchaud 1997]. Esta chave requer uma configuração manual (não integrada no protocolo) de um conjunto de chaves possíveis, partilhadas pelo cliente e pelo servidor. A chave escolhida é usada em conjunto com o conteúdo de uma mensagem COPS, de forma a criar um determinado valor que passa a fazer parte de um objecto do tipo *Integrity*. Este objecto pode então ser usado para validar todas as comunicações de um PEP para um PDP numa determinada ligação TCP. Este objecto também implementa uma funcionalidade baseada em números sequenciais de modo a tornar a comunicação mais segura. O PDP escolhe o numero inicial da sequência para o PEP e o PEP escolhe outro numero para o PDP. Estes valores são incrementados após o envio de cada mensagem de e para o PDP/PEP. O valor inicial deve ser escolhido de modo a que este nunca se repita para uma determinada chave [Durham 2000].

4.1.2 Tipo de Mensagens

O modelo de dados do protocolo pode ser definido pela descrição do formato das suas mensagens e dos objectos trocados entre o PDP e o PEP.

As mensagens COPS são iniciadas por um cabeçalho seguido de um conjunto de objectos auto-identificados. O cabeçalho da mensagem (Figura 4.4) identifica o tipo de mensagem (*Op Code*) que está a ser enviada.

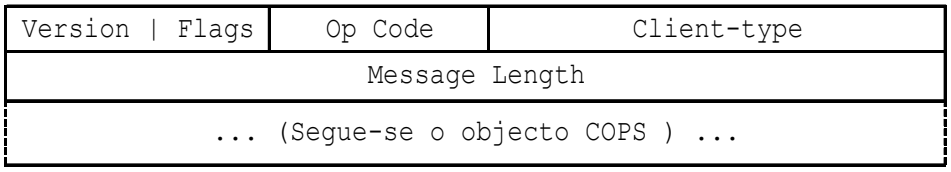

Figura 4.4 – Cabeçalho de uma mensagem COPS

Estão definidas em COPS dez tipos de mensagens diferentes [Durham 2000], apresentadas na Tabela 4.1:

N.º	Mensagem	Direcção	Descrição
1	*Request* (REQ)	PEP→PDP	Envia objectos com o estado actual do PEP, identificando o tipo de cliente e número de sequência. Deste modo o PDP já pode enviar informação para esse cliente especifico.
2	*Decision* (DEC)	PDP→PEP	Após o PDP receber uma mensagem do tipo REQ, responde com uma mensagem DEC para o número de cliente que efectuou o pedido contendo um ou mais objectos do tipo decisão. Estes objectos podem conter informação de configuração ou decisões para o PEP. Para além da informação solicitada, o PDP pode enviar uma mensagem deste tipo sempre que necessário
3	*Report State* (RPT)	PEP→PDP	Informa o PDP sobre o sucesso na recepção e implementação da decisão recebida.
4	*Delete Request State* (DRQ)	PEP→PDP	Notifica sobre o "*Delete*" de um determinado estado, que pode ter sido pedido pelo PDP ou por motivos próprios do PEP.
5	*Synchronise State Request* (SSQ)	PDP→PEP	Pede uma nova mensagem (RPT) ao PEP.
6	*Client-Open* (OPN)	PEP→PDP	Informa o PDP que o PEP vai enviar pelo menos uma mensagem do tipo (REQ)
7	*Client-Accept* (CAT)	PDP→PEP	Mensagem (OPN) aceite.
8	*Client-Close* (CC)	PEP←→PDP	Informa o receptor do fim do serviço. Pode também ser usado pelo PDP para reencaminhar o PEP para outro PDP.
9	*Keep-Alive* (KA)	PEP→PDP e resposta do PDP	Mensagem transmitida para verificar que a ligação está activa, quando não existem outro tipo de mensagens.
10	*Synchronise Complete* (SSC)	PEP→PDP	Informa o PDP que a sincronização iniciada com (SSQ), está completa. É enviada após a mensagem (RPT) ser transmitida.

Tabela 4.1 – Mensagens COPS [Durham 2000]

Nas figuras seguintes é apresentada a troca de mensagens COPS nos vários estágios de funcionamento do protocolo.

Na Figura 4.5 apresenta-se a negociação inicial entre o PEP e o PDP. Esta negociação tem como objectivo, estabelecer parâmetros de segurança que permitam a a abertura de uma ligação COPS.

Após a definição de parâmetros comuns ao PEP e ao PDP, a ligação pode ser estabelecida pelo PEP (Figura 4.6). Uma vez estabelecida a ligação, a troca de mensagens processa-se sempre que o PEP ou o PDP tenham necessidade de comunicar para a operação correcta do sistema (Figura 4.7). Durante o funcionamento do sistema são enviadas regularmente mensagens entre os intervenientes do processo, de modo a garantir que a ligação se mantém estabelecida (Figura 4.8).

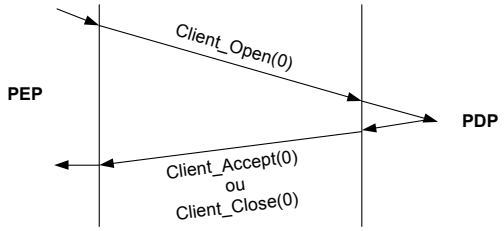

Figura 4.5 – Negociação de segurança e número de sequência

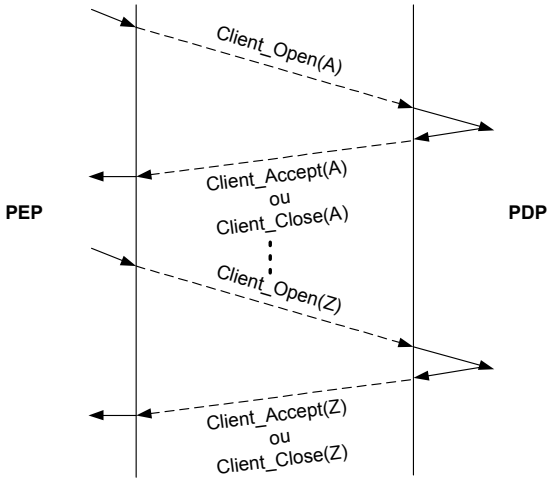

Figura 4.6 – Inicialização do PEP

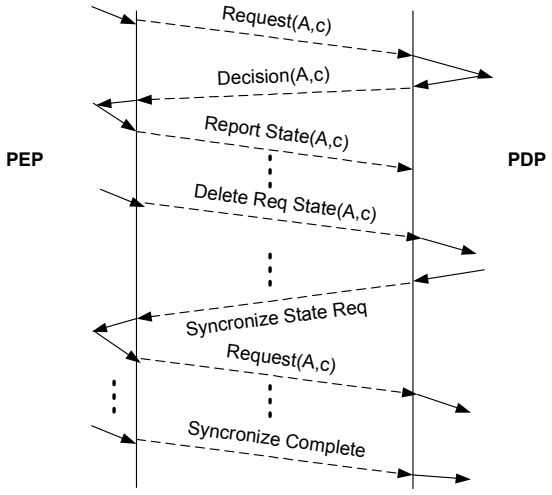

Figura 4.7 – Operações Regulares

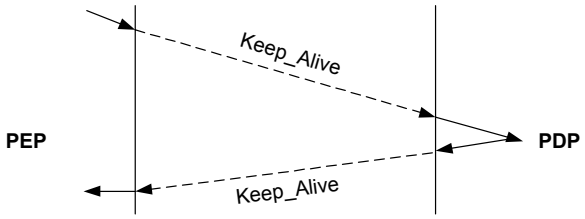

Figura 4.8 – Manutenção da Ligação

4.2 COPS for Policy Provisioning

O protocolo *COPS for Policy Provisioning* (COPS-PR) [Chan 2001] é uma extensão do protocolo COPS que funciona apenas no modelo *provisioning*, no qual o PDP transfere todas as políticas relevantes para o PEP. Com este protocolo, os clientes ligam-se a um PDP informando-o das suas capacidades e limitações, requerendo que seja enviado um conjunto inicial de políticas que estes irão armazenar localmente.

No modelo *outsourcing*, os pedidos efectuados por um PEP devem ser respondidos por uma única decisão, existindo assim uma relação 1:1 entre pedidos e decisões. No modelo *provisioning* a relação não é sempre 1:1 pois o PDP pode enviar uma decisão baseada num evento externo, por exemplo uma nova introdução de uma política na consola de gestão de rede pode desencadear uma alteração numa regra do PEP ou mesmo que o PDP envie toda uma nova informação de configuração para o PEP.

Como já foi referido, não existe uma relação directa entre os pedidos do PEP e as decisões do PDP. Os eventos enviados pelo PEP contêm informação sobre esse mesmo PEP e acerca de todos os parâmetros que podem ser alvo de configuração. Sempre que exista alguma alteração no PEP, por exemplo, se for acrescentada uma interface num *router*, este envia um novo pedido de decisão.

Funcionamento

O funcionamento do protocolo COPS-PR inicia-se com a abertura de uma ligação COPS, do PEP para o PDP primário. Após a ligação ter sido efectuada com sucesso, o PEP envia uma mensagem do tipo REQ contendo informação específica desse cliente (ex. tipo de *hardware*, versão de *software*, informação de configuração) para o PDP. Nesta fase, o PEP pode também especificar o valor máximo do tamanho de cada

mensagem COPS-PR. Após ter recebido esta informação, o PDP guarda esta informação e caso contenha políticas para serem aplicadas por esse PEP estas serão enviadas. Caso o PDP não suporte o tipo de cliente que está a tentar ligar, pode transmitir uma mensagem de erro onde pode estar contido o endereço de um PDP adicional que contenha suporte para esse mesmo cliente [Chan 2001].

Após a ligação ter sido efectuada com sucesso, a comunicação efectua-se como foi apresentado na secção 4.2.

Na Figura 4.9 é ilustrado o funcionamento genérico do protocolo COPS-PR.

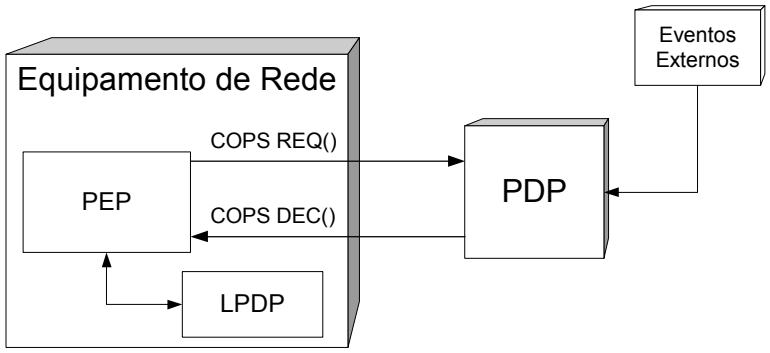

Figura 4.9 – Funcionamento COPS-PR

O protocolo COPS-PR utiliza, para o seu funcionamento, a estrutura de dados PIB (Policy Information Base) apresentada na secção 3.2.

O funcionamento do protocolo COPS-PR introduz vantagens nas situações em que se pretenda usar gestão por políticas para efectuar a configuração de equipamentos e não apenas controlar o seu funcionamento. Por exemplo, a configuração de um filtro de pacotes usando o protocolo COPS implicaria que a cada pacote, ou ligação, o PEP deveria pedir uma decisão do PDP. Com o uso de COPS-PR a informação acerca dessa filtragem de pacotes é enviada para o PEP e utilizada posteriormente de forma independente do PDP.

4.3 Síntese

Sendo um componente fundamental da arquitectura de gestão por políticas, os protocolos de difusão de políticas são necessários de modo a permitir uma comunicação sem restrições entre produtos de diversos fabricantes.

O protocolo COPS (*Common Open Policy Service*) foi criado especialmente para o transporte de políticas. Atendendo às suas características, é um protocolo bastante adequado para ser usado na arquitectura de gestão por políticas. Neste capitulo, apresentou-se o protocolo COPS e o protocolo COPS-PR (*COPS for Policy Provisioning*), sendo estes protocolos utilizados no sistema proposto apresentado no Capítulo 6.

De modo a poder representar as políticas subjacentes à arquitectura de gestão por políticas, apresenta-se, no capitulo seguinte, um estudo sobre linguagens que permitem efectuar a sua especificação.

5. Linguagens de especificação de políticas

Neste capitulo é feita a apresentação de mecanismos, linguagens formais, que permitem a descrição de políticas a implementar numa rede de computadores. É dado especial relevo à especificação de políticas referentes à área de segurança. É apresentada em pormenor a linguagem SPSL (*Security Policy Specification Language*) [Condell 2000-1], devido ao facto de ser uma linguagem vocacionada para a especificação de políticas relacionadas com aspectos de segurança, principalmente no que concerne a especificação de políticas para filtragem de pacotes. Esta linguagem foi escolhida para a implementação do protótipo apresentado no Capítulo 6.

5.1 Motivação

De forma a representar as políticas usadas na arquitectura de gestão baseada em políticas, existe a necessidade da utilização de algum modelo de dados. Este modelo, pode, neste ponto, ser considerada como uma linguagem através da qual o gestor de rede descreve o comportamento que pretende que a sua rede tenha. Actualmente ainda não existe um norma para uma linguagem deste tipo, existindo sim, vários documentos orientadores, cada um vocacionado para a área de gestão que se pretende efectuar.

A linguagem a criar, como norma, deve fornecer uma plataforma única que suporte os novos conceitos das arquitecturas baseadas em políticas que estão a ser desenvolvidos

actualmente. Neste cenário, devem-se identificar os requisitos que uma linguagem para especificação de políticas deve suportar.

Uma linguagem de especificação de políticas deve ser simples e facilmente compreensível pelos utilizadores. Deve permitir suporte para múltiplas actividades de gestão, nomeadamente políticas de segurança e controlo de acessos. As políticas devem ser agrupadas e não serem apenas tratadas de forma individual de modo a facilitar a especificação de políticas relacionadas com grandes redes de dados. Para que as acções referentes a segurança e gestão da rede possam estar relacionadas entre si, sempre que necessário, a linguagem deve permitir efectuar algum tipo de composição de políticas, com a possibilidade de as analisar em termos de conflitos com outras políticas, bem como verificar a sua consistência dentro da especificação global. Por ultimo, a linguagem deve ser expansível, de modo a permitir novos tipos de políticas que previsivelmente irão surgir no futuro. Este objectivo pode ser atingido, utilizando algo semelhante à herança das linguagens orientadas a objectos [Stone 2001].

A especificação de políticas pode ser efectuada através do uso de uma linguagem e um conjunto de acessórios que permitam manipular essa mesma linguagem (Figura 5.1). Um acessório importante para a especificação de políticas será a criação de um interface gráfico que permita ao gestor uma maior abstracção da linguagem.

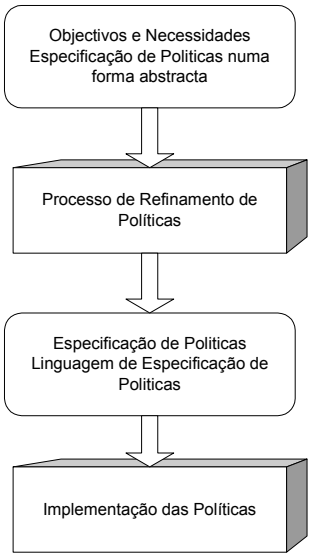

Figura 5.1 – Especificação de políticas [Damianou 2000]

Existindo várias especificações, cada uma com a sua própria sintaxe e semântica, deve ser escolhida uma linguagem que seja bastante extensível e com um grande grau de

escalabilidade, pois estão constantemente a surgir novas necessidades de gestão o que implica que novas funcionalidades devem ser adicionadas à linguagem sem grande esforço adicional.

5.2 Security Policy Specification Language

A linguagem SPSL (*Security Policy Specification Language*) [Condell 2000-1] pretende ser uma linguagem independente das plataformas e/ou do vendedor do equipamento para a qual se pretende especificar políticas, especialmente políticas relacionadas com aspectos de segurança. Com a generalização do uso de *firewalls* com elevadas funcionalidades assim como o uso de redes privadas virtuais (VPNs - *Virtual Private Network*) com vários níveis de encriptação de dados, surge a necessidade de criar uma forma mais simples e eficaz de gerir esses mesmos mecanismos [Ioannidis 2000], nomeadamente através do uso de uma gestão baseada em políticas. A linguagem SPSL permite que as políticas de segurança possam ser especificadas numa linguagem genérica, que pode ser guardada no seu formato original ou transformada num qualquer modelo de dados para posterior processamento e aplicação das suas políticas nos equipamentos que implementam segurança. Existe deste modo uma separação entre a linguagem que representa as políticas e os equipamentos que as devem por em prática. Neste sentido uma linguagem como a SPSL é um componente essencial na concepção de um tipo de gestão baseada em políticas [Keromytis 1999].

Neste contexto, apresenta-se a linguagem SPSL de acordo com o definido em [Condell 2000-1].

5.2.1 A linguagem

Desenvolvida no âmbito do grupo de trabalho *IP Security Policy* (*ipsp*)[6], do IETF[7] (*Internet Engineering Task Force*), esta linguagem foi definida inicialmente para ser usada na gestão de IPSec e com sistemas de gestão de chaves IKE (*Internet Key Exchange*) [Blaze 2001]. Contudo é possível usar esta linguagem para outros fins, expandindo a sua sintaxe. A sua sintaxe foi derivada da linguagem de especificação de

[6] http://www.ietf.org/html.charters/ipsp-charter.html

[7] http://www.ietf.org/

políticas de *routing* (RPSL) [Alaetinouglu 1998], pelo que está bastante direccionada para lidar com regras referentes à comunicação de dados, como por exemplo a especificação de regras para *firewalls*.

A linguagem SPSL foi desenvolvida, na sua versão base, tendo em consideração as seguintes necessidades [Condell 2000-1]:

- Suporte para IPSec/IKE e de um modo geral permitir o suporte para especificação de políticas de segurança em comunicações;
- Suporte para modelo baseados em nós de uma rede e para um modelo baseado em domínios;
- Possibilidade da existência de múltiplos servidores de políticas;
- Existência de mecanismos de autenticação e autorização de modo a facilitar a gestão das políticas;
- Elevado grau de flexibilidade e possibilidade de expansão da linguagem.

Uma política é definida na linguagem SPLS, na forma de uma associação entre um conjunto de condições sobre a comunicação e um correspondente conjunto de acções de segurança que devem ser tomadas quando as condições da comunicação forem verdadeiras. Esta abstracção é usada de forma genérica para especificar políticas de segurança sobre as comunicações. Se uma comunicação que está estabelecida(ou ainda vai ser estabelecida) verifica uma das condições existentes, então uma das acções definidas deve ser tomada de forma a proteger essa mesma comunicação. Este género de abstracção generaliza as práticas correntes deste tipo de gestão.

Em SPSL, as acções definidas numa política podem afectar a segurança da comunicação de diferentes formas, por exemplo [Condell 2000-1]:

- Pode simplesmente especificar acções de filtragem de pacotes: eliminar o pacote, deixar passar ou reencaminhar esse pacote para uma outra entidade na rede;
- Especificação de configuração necessária para protecção da troca de chaves IKE;

- Criar túneis IPSec para a passagem de determinado conjunto de pacotes, incluindo mecanismos para a especificação e configuração desse tipo de comunicação.

São definidas duas metodologias para a definir a associação de políticas com as entidades existentes na rede: O modelo *baseado em nós* e o *baseado em domínios*, podendo de um modo geral estes modelos serem usados em simultâneo na especificação de uma política de segurança global [Condell 2000-1].

No modelo *baseado em nós*, as políticas de segurança estão associadas aos vários componentes existentes na rede que implementam segurança, por exemplo *firewalls*, servidores, etc. As políticas associadas a um nó especifico, descrevem a protecção que deve ser efectuada nas comunicações de e para esse mesmo nó de rede. É suposto que esses tipos de políticas sejam postas em práticas individualmente por cada um desses nós.

No modelo *baseado em domínios*, as políticas estão associadas de uma forma geral a todo um domínio que se pretende proteger. Um domínio de segurança pode aqui ser definido como um conjunto interligado de entidades existentes na rede que são geridas em termos de configuração de segurança, por um (ou mais) PDP geral e por um conjunto de PEPs referentes a cada entidade. Cada PEP contribui de alguma forma para a aplicação da política genérica do domínio. O domínio de segurança pode ser constituído por vários subdominios e também por equipamentos isolados, permitindo assim modelar mais facilmente a realidade.

A segurança referente à própria política de segurança escrita usando a linguagem SPSL está implementada usando algumas classes de objectos definidas para esse efeito, suportando os seguintes serviços [Condell 2000-1]:

- Garantia da integridade dos dados existentes nas políticas definidas e autenticação da origem desses mesmos dados: Cada objecto do tipo política é protegido através do uso de uma assinatura do tipo chave publica. Este facto fornece também um meio para a não-repudiação do autor dessa mesma política;
- Mecanismos de autorização e autenticação dos autores das políticas: os objectos de gestão contêm associados um certificado de modo a permitir a autenticação das

políticas criadas por ele ou apenas para a sua identificação para controle de acesso a essas mesmas políticas.

Com o uso destes serviços, os utilizadores da linguagem SPSL podem sempre efectuar uma verificação da integridade e da origem das políticas definidas, assim como podem efectuar algum controlo sobre quem pode efectuar a manutenção das mesmas.

A linguagem SPSL é bastante flexível e extensível. Flexível devido ao facto de que a sua sintaxe actual permite a definição de políticas para uso em diferentes áreas. Pode, por exemplo, ser usada para definição de regras de filtragem de pacotes ou para a criação de VPNs que utilizem IPSec. É extensível pois permite a criação de novas classes de objectos seguindo uma regra semelhante à herança, no paradigma objecto. Consequentemente a linguagem pode ser expandida de modo a permitir a especificação de políticas para diferentes tipos de comunicações, protocolos de segurança ou aplicações.

5.2.2 Componentes

A linguagem SPSL baseia-se em objectos, sem contudo ser uma linguagem orientada a objectos pelo facto de os objectos não conterem métodos, nem ser possível, na actual especificação, criar novos tipos de objectos. Está definido um conjunto de classes que podem servir para instanciar outros objectos. Cada objecto tem associado um conjunto de atributos que podem ser opcionais ou obrigatórios e que contêm os dados relevantes para a definição das políticas [Condell 2000-1].

Cada classe é composta por um conjunto de atributos que permitem armazenar informação acerca dos objectos existentes nessa classe. Os atributos podem ser obrigatórios ou opcionais. Um atributo obrigatório tem que ser definido para todos os objectos da classe e um atributo opcional pode ser omitido. Os atributos podem também conter um único valor ou vários valores. Um atributo com um único valor, apenas pode aparecer uma vez em cada objecto. Um atributo com múltiplos valores, pode aparecer mais do que uma vez por cada objecto. Cada objecto é identificado univocamente pelo atributo chave da classe.

Como os objectos em SPSL são distinguidos e referenciados pelo seu atributo chave, o valor deste atributo (que normalmente corresponde ao seu nome) tem que ser único em todo o conjunto de objectos definidos na especificação de uma política em SPSL.

Em [Condell 2000-1] estão definidas 4 categorias principais de objectos:

- Dados primitivos – contendo dados usados na definição dos objectos, por exemplo o nome do objecto, um endereço IP, etc;
- Agentes de Manutenção – pretendem guardar informação referente à pessoa que pode modificar a informação contida nos objectos. Integra também um conjunto de informações sobre certificados que irão servir para autenticar cada objecto;
- Entidades da Rede – entidades relevantes para a especificação de uma política global de segurança: domínios, nós de rede, etc;
- Políticas – contém a especificação pormenorizada de cada política de segurança. Existem dois tipos de políticas já definidas: *class policy* que define filtragem de pacotes e *class ipsec-policy* que define acções *IPSec*.

De modo a permitir uma política de segurança global, a SPSL pode lidar com políticas baseadas em nós ou em domínios. Para servir de suporte a estes conceitos estão definidas as classes, *node*, *gateway*, *polserv* e *domain*. Um *node* é uma representação de uma entidade de rede que tenha pelo menos um interface e um nome. Um *gateway* é uma entidade que implementa as políticas de segurança. Um *polserv* define um servidor de políticas que é capaz de lidar com regras SPSL e futuramente com recurso a esta aplicação, também deverá ser capaz de as traduzir para a linguagem proprietária de cada equipamento.

Um *domain* engloba um conjunto de *nodes*, podendo ter mais do que um *gateway* que implemente as políticas para esse domínio. As políticas são representadas usando a classe *policy*, estando cada uma associada a um ou mais *node* ou *domain*. Os objectos *mnter* guardam informação sobre a pessoa responsável pela política de segurança, como nome, telefone, *email*, certificados, etc.

Cada objecto deve ser assinado por um certificado digital de modo a garantir a sua genuinidade.

5.2.3 Estrutura

A estrutura da linguagem de especificação de políticas SPSL é apresentada segundo a definição de [Condell 2000-1].

Um objecto em SPSL é representado textualmente na forma de uma lista de pares *atributo-valor*. A representação dos objectos inicia-se com a definição da chave da classe. O nome de cada atributo representa-se antes de um sinal ":" (dois pontos), sendo seguido pelo valor do atributo. O valor do atributo pode ser dividido por várias linhas sendo cada linha deste tipo separada por um sinal "\". A representação de um objecto termina com uma linha em branco ou com o final do ficheiro onde está armazenado.

A ordem pela qual os atributos são definidos no objecto também é relevante. A chave da classe deve ser sempre o primeiro atributo. Se o atributo que define o conjunto de caracteres usado (*char-set*) estiver presente, este deve estar definido antes de qualquer atributo que utilize texto nos seus valores. O conjunto de caracteres definido por defeito é o ISO 8859-1 (*iso Latin-1*). O ultimo atributo definido no objecto tem que ser a sua assinatura.

A especificação dos objectos permite também o uso de comentários. Um comentário pode aparecer em qualquer posição da definição do objecto. Um comentário inicia-se com o caracter "#" e termina com o fim da linha onde está definido.

Os objectos são escritos num único ficheiro de texto, podendo existir ficheiros secundários incluídos usando a instrução *$INCLUDE*. A ordem segundo a qual os objectos são escritos é de extrema importância, pois no caso da filtragem de pacotes, a primeira regra que satisfaça as condições é aplicada, pelo que a política a aplicar por defeito a um dado domínio deverá ser sempre a última política para esse mesmo domínio.

O ficheiro deverá definir as entidades relevantes existentes na rede usando os objectos *node*, *gateway* ou *polserv*. Estes objectos têm entre outros os atributos *name* e *ifaddr*, respectivamente o nome DNS (*Domain Name System*) e endereço(s) IP da entidade modelada.

De modo a facilitar a administração da política de segurança, as entidades podem ser agrupadas em conjuntos, usando os atributos *node-sets* e *gateway-sets*. O objecto

domain define um conjunto de entidades na rede às quais será aplicado o mesmo conjunto de políticas de segurança. O seu atributo *coverage* lista as entidades que pertencem ao domínio, podendo ter como valores objectos *node*, *node-sets* ou ainda uma lista de endereços IP.

Cada política deverá ter o atributo *association* que define o domínio ou entidades a que esta se aplica.

A classe *policy* poderá ter um ou mais atributos do tipo *policy*, sendo possível especificar uma ligação usando o seu destino, origem, porto de destino e origem, protocolo de transporte e direcção. As acções base a executar que estão definidas são *permit*, *deny* e *forward*.

Um objecto da classe política especifica uma associação entre um conjunto de condições referentes a uma comunicação e um conjunto de acções a por em prática nessa mesma comunicação.

Actualmente estão formalmente definidas dois tipos de políticas: A definição de políticas associadas a filtragem de pacotes e as políticas associadas a comunicações IPSec.

Os objectos do tipo "Política" podem ser representados de duas formas distintas: o formato pequeno ou longo. Cada um destes formatos é apropriado para diferentes aplicações no caso de ser necessário representar mais ou menos atributos.

Ambos os formatos deste objecto partilham três atributos. Na Figura 5.2 está representada a definição de um objecto (formato pequeno) de modo a serem apresentados os atributos comuns. Entre os atributos, temos como sendo comuns, o atributo *policy-name* que representa o nome da política, *cache-expiry* que indica em segundos o período máximo que esta política pode estar num cliente sem ser verificada a existência de uma nova (se este atributo não estiver presente é considerado que não existe um período máximo). O atributo *association* especifica os nomes de um ou mais nós da rede ou domínios aos quais esta política está associada.

O formato pequeno do objecto *policy* especifica uma política usando um único atributo da seguinte forma:

```
      Atributo          Valor                                    Tipo

      policy-name:      <policy-name>                            man, s-v, key
      char-set:         <char-set>                               opt, s-v
      notes:            <free-form>                              opt, m-v
      association:      <node-name> | <node-set-name>
                        | <gateway-name> | <gateway-set-name>
                        | <domain-name>                          man, s-v
      cache-expiry:     <integer>                                opt, s-v
      policy:           Ver Figura 5.3                           opt, m-v
      mnt-by:           list of <mntner-name>                    man, m-v
      changed:          <mntner-name> <date>                     man, m-v
      signature:                                                 man, m-v

Tipo [man: Obrigatório; opt:Opcional ; s-v: valor único; m-v: multiplos valores; key: chave]
```

Figura 5.2 – Atributos da classe "policy" (formato pequeno)

Especificação dos atributos do objecto *Policy* (formato pequeno):

```
policy:
      dst * | any | [not] list of <ip-address>, <address-range>
            [port (* | opaque | any |
            [not] list of <port>, <port-range>)
            [dynamic [<port-range>]]]
      [src * | any | [not] list of <ip-address>, <address-range>
            [port (* | opaque | any |
            [not] list of <port>, <port-range>)
            [dynamic [<port-range>]]]]
      [xport-proto * | opaque | any |
            [not] list of <proto>, <proto-range>]
      [direction (inbound | outbound) [, symmetric]]
permit [, forward <dest> [<proto> <port>]]
| deny [, forward <dest> [<proto> <port>]]
| reject [, forward <dest> [<proto> <port>]]
```

Figura 5.3 – Atributos do objecto *Policy* (formato pequeno)

O sub-atributo *dst* especifica uma lista de endereços IP de destino ou uma gama de endereços aos quais a política deve ou não ser aplicada. Este endereço pode ser especificado também como *any* ou "*", indicando tratar-se de todos os endereços. A gama de endereços pode ser especificada recorrendo a valores mínimo e máximo ou usando uma especificação endereço/máscara. Uma lista de endereços pode ser precedida pelo qualificador *not*, indicando então o conjunto inverso dos endereços especificados. Ao sub-atributo *dst* pode ser opcionalmente acrescentada uma porta ou lista de portas para as quais a política deve ou não aplicar-se. Para a porta pode também ser definida o valor *dynamic*, especificando desta forma uma comunicação estabelecida numa determinada porta pode usar outras portas de forma dinâmica no resto da comunicação que efectuar. Por defeito, a não especificação da porta implica que seja considerada qualquer porta e que a opção *dynamic* não será usada.

O sub-atributo *src* é equivalente ao anterior, considerando aqui tratar-se do endereço de origem de uma determinada comunicação.

O protocolo de transporte pode ser especificado usando o sub-atributo *xport-proto* ao qual está associado por defeito o valor "*". Podemos neste caso utilizar um único valor para o protocolo, uma lista de valores ou ainda uma gama de valores na forma *min-max*.

A *direction* especifica se a política deve ser aplicada a pacotes que estão a entrar no domínio ou a sair deste. Pode ser usada a opção *symetric* para aplicar a política em ambas as situações.

A opção *transfer action* do tipo *permit*, *deny* ou *reject* é aqui especificada de modo a referir o destino dado aos pacotes que satisfazem essa mesma política. Adicionalmente a estas três opções, podemos acrescentar a opção de reencaminhar o pacote para outro destino.

Como apresentado na Figura 5.4, no formato longo do objecto *Policy*, cada componente da política é expresso individualmente num atributo usado para o efeito.

Atributo	**Valor**	**Tipo**
policy-name:	<policy-name>	man, s-v, key
char-set:	<char-set>	opt, s-v
notes:	<free-form>	opt, m-v
association:	<node-name> | <node-set-name> | <gateway-name> | <gateway-set-name> | <domain-name>	man, s-v
cache-expiry:	<integer>	opt, s-v
valid-period:	list of <valid-time>	opt, m-v
dst:	Ver Figura 5.5	opt, m-v
src:	Ver Figura 5.5	opt, m-v
xport-proto:	Ver Figura 5.5	opt, m-v
direction:	(inbound | outbound) [, symmetric]	opt, s-v
userid:	* | any | [not] list of n822 <email-addr>, dn <distinguished-name>	opt, m-v
systemname:	* | any | [not] list of <general-name>, dn <distinguished-name>	opt, m-v
ipv6-class:	* | any | [not] list of <integer-range>	opt, m-v
ipv6-flow:	* | any | [not] list of <integer-range>	opt, m-v
ipv4-tos:	* | any | [not] list of <integer-range>	opt, m-v
seclabel:	* | any | [not] list of <seclabel>	opt, m-v
tfr-action:	Ver Figura 5.5	opt, m-v
mnt-by:	list of <mntner-name>	man, m-v
changed:	<mntner-name> <date>	man, m-v
signature:		man, m-v

Figura 5.4 – Atributos da classe "policy" (formato longo)

Especificação dos atributos do objecto *Policy* (formato longo):

```
dst: * | any | [not] list of <ip-address>, <address-range>
             [port (* | opaque | any | [not] list of <port>, <port-range>)
          [dynamic [<port-range>]]]

src: * | any | [not] list of <ip-address>, <address-range>
             [port (* | opaque | any | [not] list of <port>, <port-range>)
          [dynamic [<port-range>]]]

xport-proto: * | opaque | any | [not] list of <proto>, <proto-range>

tfr-action: permit [, forward <dest> [<proto> <port>]]
            | deny [, forward <dest> [<proto> <port>]]
            | reject [, forward <dest> [<proto> <port>]]


<valid-time>: [year yyyy-yyyy] [month 000000000000]
     [day-of-month 00000000000000000000000000000000]
     [reverse-day-of-month 00000000000000000000000000000000]
     [day-of-week 0000000] [time [not] hh:mm:ss-hh:mm:ss]
```

Figura 5.5 – Atributos do objecto *Policy* (formato longo)

O atributo *valid-period* descreve um ou mais períodos de tempo para os quais a política definida deve ser aplicada. O formato usado para representar estes períodos pode ser consultado na definição da linguagem[Condell 2000-1].

Os atributos *dst*, *src*, *xport-proto*, *direction* e *tfr-action* são equivalentes aos usados no formato pequeno

O atributo *user-id* permite a especificação de um endereço de correio electrónico ou um nome para identificar um único utilizador. O atributo *systemname* pode usar um nome DNS ou endereço para identificar um sistema em particular.

Os atributos *ipv6-flow* e *ipv6-class* podem conter uma lista de valores ou gama de valores correspondendo aos *label* de um fluxo Ipv6 e classe de transporte. O atributo *ipv4-tos* pode ser uma lista ou gama de valores inteiros e refere-se ao tipo de serviço especificado no cabeçalho de um pacote Ipv4.

As políticas escritas na linguagem SPSL podem usar qualquer um dos formatos apresentados ou ainda uma combinação desses mesmos formatos. Podendo especificar a mesma política num ou noutro formato, cada um implica algumas vantagens e desvantagens. O formato pequeno permite que a especificação de políticas mas simples, como uma filtragem de pacotes, seja feita usando um formato mais condensado, permitindo inclusive que um único objecto especifique mais do que uma política. Contudo, este formato não é indicado para a especificação de políticas mais complexas.

O formato longo permite a especificação de uma forma mais perceptível, logo mais adequado para situações complexas. A combinação destes dois formatos traz uma maior flexibilidade à representação do sistema a modelar.

5.2.4 Exemplos de aplicação

Exemplo de uma política definida usando uma classe *policy*:

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

Figura 5.6 – Extracto de uma política em SPSL – classe *policy*

Esta política permite todas as ligações para o endereço 193.197.128.43 a partir de qualquer origem, desde que o protocolo seja o TCP. A mesma política pode ser escrita usando o formato longo da mesma classe, no qual existem mais atributos que podem ser usados, por exemplo se os endereços forem IPv6.

A mesma política especificando individualmente todos os atributos:

```
policy-name:     EXEMPLO
dst:             193.197.128.43
src:             *
xport-proto:     6
tfr-action :     permit
```

Figura 5.7 – Extracto de uma política em SPSL – Atributos separados

Na Figura 5.8 é apresentado um ficheiro contendo uma política de segurança usando SPSL. Este exemplo contém uma política de segurança que está associada apenas à máquina *www.ipv.pt* (WEB), declarando que apenas permite ligações TCP de qualquer origem para a máquina WEB.

No que diz respeito à segurança das próprias políticas usando SPSL, esta deverá ser assegurada pelas aplicações que utilizem este ficheiro, existindo para tal os atributos *mnt-by* e *signature* que permitem verificar a validade da política. Sendo simples alterar o ficheiro de texto, é contudo difícil de escrever uma assinatura válida para o objecto sem a respectiva chave privada, pelo que a integridade dos objectos deverá está

garantida. Por outro lado, terá que haver outras preocupações de segurança sobre este ficheiro, pois o acto de apagar uma política não será reconhecido por uma aplicação que apenas verifique as assinaturas dos objectos. Neste caso, deverá ser dada atenção à validação da integridade de todo o ficheiro.

```
                                          changed:         FILIPE 20000829
mntner:          FILIPE                   signature:       FILIPE FILIPE-CERT \
auth:            crypt-pw slijhygsjhsku                    dsa-sha1 gkudodx7
address:         IPV - Viseu
phone-number:    +351-232480500           policy-name:     TCP-WEB
email:           caldeira@di.estv.ipv.pt  association:     WEB
mnt-by:          FILIPE                   policy:          dst 193.137.7.1 \
certs:           FILIPE-CERT                               src * \
changed:         FILIPE 20000829                           xport-proto 6 permit
signature:       FILIPE FILIPE-CERT \     mnt-by:          FILIPE
                 dsa-sha1 klj498dg        changed:         FILIPE 20000829
                                          signature:       FILIPE FILIPE-CERT \
cert:            FILIPE-CERT                               dsa-sha1 k75fnmkj
certlocation:    x509_sig  file  filename  \
                 /home/FILIPE/.x509_cert  policy-name:     DEFAULT
mnt-by:          FILIPE                   association:     WEB
changed:         FILIPE 20000829          policy:          dst * src * deny
signature:       FILIPE FILIPE-CERT \     mnt-by:          FILIPE
                 dsa-sha1 sli4nflk        changed:         FILIPE 20000829
                                          signature:       FILIPE FILIPE-CERT \
node:            WEB                                       dsa-sha1 o29fnmkj
name:            www.ipv.pt
ifaddr:          193.137.7.1              #
mnt-by:          FILIPE
```

Figura 5.8 – Política em SPSL

5.3 Outras propostas

Tendo em conta a existência de diversas linguagens, são apresentadas algumas consideradas mais relevantes.

eXtensible Mark-up Language (XML)

Uma das possibilidades para representar uma linguagem de especificação de políticas é o uso da *eXtensible Mark-up Language* (XML) [Bray 1998], pois de entre as várias meta-linguagens deste género, a XML é já uma norma sendo cada vez mais adoptada para armazenar dados estruturados. Devido ao facto de a XML ser normalizada, existem vários "*parsers*" para esta linguagem, para além de várias APIs (*Aplication Programming Interface*) para a sua integração com uma linguagem de programação tradicional como o *C++* e o *Java*. Neste contexto a XML pode então ser usada como uma ferramenta de especificação de políticas.

Policy Framework Definition Language (PFDL)

O PFDL [Strassner 1998] é uma linguagem proposta pelo grupo de trabalho de gestão por políticas do IEFT, foi definida para permitir transformar as necessidades de especificação dos equipamentos existentes numa rede, numa especificação formal independente do formato usado pelos equipamentos. O benefício deste tipo de linguagem numa arquitectura de rede baseada em políticas é o facto de permitir a coexistência de dispositivos heterogéneos a suportar a mesma política.

A linguagem PFDL é baseada no modelo de dados CIM actualmente a ser desenvolvido pelo DMTF. Este modelo define uma hierarquia de classes de objectos que podem ser usados para representação de políticas. As classes e os relacionamentos entre elas existentes no modelo CIM são usadas para definir a gramática da linguagem como se pode ver na Figura 5.9. O ponto de partida é o de que uma política é uma agregação de regras. Uma regra define uma sequência de acções que serão executadas quando o correspondente conjunto de condições seja verdadeiro [Strassner 1998]. Estão definidas cinco classes para suporte ao modelo CIM [CIM 1999]: As classes *ComplexPolicy*, *SimplePolicy*, *PolicyRule*, *PolicyCondition* e *PolicyAction*. O relacionamento existente entre estas classes está também representado na Figura 5.9.

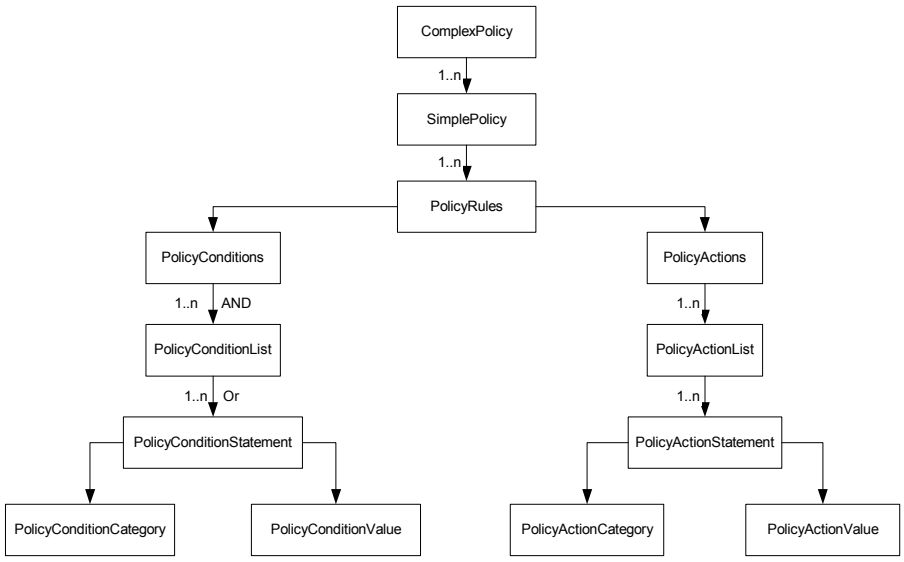

Figura 5.9 – PFDL - Relacionamento entre classes [Strassner 1998]

Uma *PolicyRule* contêm um conjunto de *PolicyConditions* e um conjunto de *PolicyActions*. Quando o conjunto de *PolicyConditions* é satisfeito, as acções existentes em *PolicyActions* são executadas.

Um *PolicyConditionStatement* é composto por um par *categoria-valor*. Estes dois componentes são específicos para um determinado domínio de aplicação, podendo esse domínio ser QoS, segurança ou outro. A existência de *condições-acções* para um determinado domínio permite à linguagem ser aplicada nas várias áreas existentes.

Uma classe *PolicyAction* consiste numa acção ou uma lista de acções. Estas acções podem ser executadas numa ordem específica ou arbitrária. Paralelamente à ordenação das acções a serem realizadas, a linguagem permite ainda executar uma ou mais acções baseadas no resultado da acção anterior.

Esta linguagem também permite de um modo geral lidar com o tratamento de conflitos entre as políticas definidas.

Ponder Policy Specification Language (PONDER)

A linguagem PONDER [Damianou 2001], desenvolvida no departamento de computação do Imperial College of Science, Technology and Medicine – Londres[8], fornece um meio para especificar políticas que podem ser mapeadas em vários mecanismos que as implementem, nomeadamente *firewalls*, sistemas operativos e bases de dados. Suporta políticas do tipo condição-acção que podem ser executadas após a ocorrência de algum evento. Esta linguagem pode ser usada em sistemas de gestão por políticas ou mesmo para sistemas distribuídos. Pode ainda ser usada para a gestão de actividades relacionadas com a gestão de sistemas de segurança, como por exemplo, o registo de novos utilizadores, controle de acessos e auditoria de acessos a recursos críticos ou auditoria de violações de segurança.

Os conceitos principais da linguagem são o uso de regras que agrupam políticas relacionadas com uma determinada tarefa na organização, a definição de relacionamentos que representam iterações entre tarefas e estruturas de gestão, permitindo criar regras que são adequadas a cada departamento ou área da organização.

[8] http://www-dse.doc.ic.ac.uk/Research/policies/ponder.html

O uso de regras compostas por políticas permite obter um elevado grau de abstracção acerca dos sistemas existentes na organização.

A linguagem PONDER é declarativa e baseada em tipos de dados do tipo orientado a objecto. Este facto permite obter desta linguagem algum grau de flexibilidade no que concerne à sua adaptação e extensão a várias áreas da gestão de redes.

Na Figura 5.10 é apresentado um exemplo de uma política, que permite especificar filtragem de tráfego numa rede, escrita usando a linguagem PONDER.

```
inst auth+ filter1 {
        subject /Agroup + /Bgroup ;
        target USAStaff – NYgroup ;
        action VideoConf(BW, Priority)
                { in BW=2 ; in Priority=3 ; } // default filter
                if (time.after("1900")) {in BW=3; in Priority = 1; }
}
```

Figura 5.10 – Exemplo de uma política na linguagem PONDER [Damianou 2001]

A política apresentada na Figura 5.10, determina que os membros dos grupos pré-definidos *Agroup* e *Bgroup* podem usar o serviço de vídeo-conferência para o grupo *USAStaf* com excepção do grupo *Nygroup*. Se este serviço for usado após as 19 horas, os parâmetros usados para estabelecer a ligação especificam que a largura de banda disponível é de 3Mb/s com prioridade 1. Nas restantes situações, os parâmetros especificados são de uma largura de banda de 2 Mb/s e prioridade 3.

Simple Ruleset Language (SRL)

A linguagem SRL [Brownleee 1999], desenvolvida na Universidade de Auckland por Brownlee, pretende descrever políticas para medição de tráfego em tempo real (RTFM - *Realtime Traffic Flow Measurement*) numa rede. Estas políticas especificam os fluxos de tráfego que devem ser medidos assim como a quantidade de informação que deve ser armazenada. Os atributos usados para verificar os padrões usados são entre outros o endereço de origem e destino, número de portos, etc. A identificação dos pacotes é efectuada usando a estrutura condicional *IF* (Se <condição> <acção1> senão <acção2>).

As políticas escritas na linguagem SRL têm como objectivo a identificação dos pacotes que se pretendem analisar e após esta identificação, aplicar acções a estes pacotes. As acções possíveis limitam-se a armazenamento de informação sobre os fluxos de tráfego e a manutenção de informação estatística acerca dessa mesma informação.

Na Figura 5.11 está representado um exemplo da linguagem SRL que permite representar uma política destinada a efectuar a contagem de pacotes TCP/IP cuja porta de destino é a porta de *Telnet*, armazenando também todos os pares origem/destino de cada pacote contado.

```
#
# Classify IP port numbers
#
define Ipv4 = 1; # Address Family number from RFC 1700
define telnet = 23; # Well-known Port numbers from RFC 1700
define tcp = 6; # Protocol numbers from RCF 1700
#
if SourcePeerType == Ipv4 save;
else ignore; # Not an Ipv4 packet
#
if (SourceTransType == tcp && DestTransAddress == telnet)
save, store FlowKind := 'T';
#
save SourcePeerAddress /32;
save DestPeerAddress /32;
count
#
```

Figura 5.11 – SRL: Exemplo da linguagem [Brownleee 1999]

Routing Policy Specification Language (RPSL)

Em resultado de actividades do grupo de trabalho *Routing Policy System*[9] do IETF foi definida a linguagem RPSL [Meyer 1999]. A RPSL teve origem no desenvolvimento da linguagem RIPE-81 (*Réseaux IP Européens - Representation of IP Routing Policies in the RIPE Database*) que foi uma das primeiras linguagens a ser definida para especificação de políticas de encaminhamento. Ao especificar políticas usando RPSL, um gestor de rede pode especificar políticas de encaminhamento de modo a serem armazenadas no IRR (*Internet Routing Registry*). O sistema IRR guarda as políticas de

[9] http://www.academ.com/nanog/may1995/rps.html

várias organizações, de forma a que estas políticas possam ser vistas por outros usando o serviço "*WHOIS*". Cada objecto armazena os vários componentes de informação referentes a cada política, sendo constituído por vários atributos denominados *Keys*. A linguagem RPSL foi desenvolvida com o objectivo de que a configuração de um *router* possa ser gerada partindo da especificação da política escrita nesta linguagem [Meyer 1999].

Na Figura 5.12 está representada uma política simples. O atributo *autnum* representa o número do sistema, neste caso AS2 representa o sistema autónomo 2. *as-name* e *desc*" representam respectivamente o nome e descrição associada ao sistema. Os atributos mais importantes neste exemplo são as políticas de *import* e *export*. A clausula *import* especifica as políticas de importação de regras de *routing* e a *export* especifica as políticas de exportação.

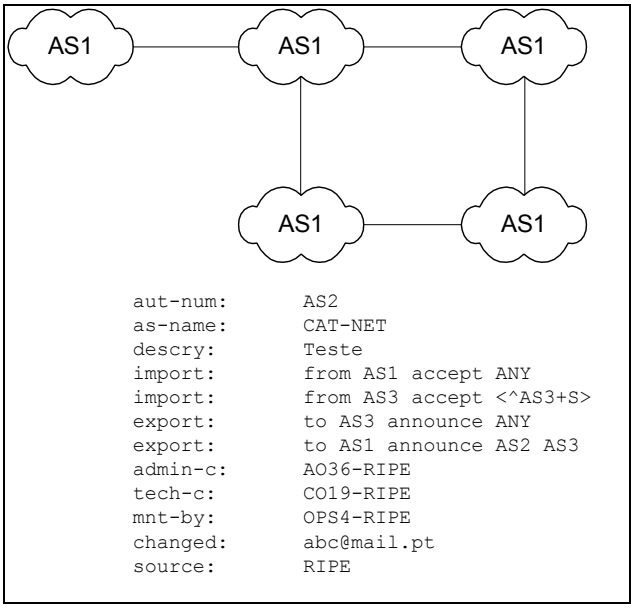

Figura 5.12 – Diagrama RPSL e exemplo de uma política [Meyer 1999]

Neste exemplo, a regra de importação "*from AS1 accept ANY*" indica que o sistema AS2 irá aceitar qualquer tipo de anúncios que o sistema AS1 enviar. A segunda regra de importação existente indica que o sistema AS2 aceita apenas anúncios originados no sistema AS3 e que contêm no seu caminho apenas referência do sistema AS3

A regra de exportação "*to AS3 announce ANY*" indica que qualquer rota que os sistema AS2 contenha na sua tabela irá ser passada para o sistema AS3. A segunda regra permite os anúncios de quaisquer rotas aprendidas pelos sistemas AS2 ou AS3 para o sistema AS1.

Os atributos *admin-c* (administrativo), *tech-c* (técnico), *mnt-by* (mantido por) e *changed* (ultima alteração por) são atributos que contêm informação de contacto usada pelo serviço *whois*. Os valores associados a estes atributos são identificadores únicos que pretendem identificar pessoas.

Ao permitir especificar políticas de *routing*, a linguagem RPSL não foi originalmente definida para ser usada numa outra área de implementação. Contudo o desenvolvimento desta linguagem, nomeadamente para a área de segurança, deu origem à linguagem SPSL (*Security Policy Specification Language*) [Condell 2000-1].

5.4 Síntese

Apesar de ainda não existir uma norma sobre qual a linguagem de especificação de políticas que deve ser utilizada pelos fabricantes de equipamentos, neste capitulo foram discutidas algumas das linguagens existentes: XML, PFDL, PONDER, SRL, RPSL e SPSL. Identifica-se também neste capitulo, alguns requisitos que uma linguagem para especificação de políticas deve suportar

No que concerne à linguagem SPSL, é alvo de uma descrição pormenorizada, apresentado a sua sintaxe e exemplos de aplicação. Pelo facto de estar vocacionada para especificar políticas relacionadas com segurança, a linguagem SPSL foi usada na implementação de um sistema de gestão por políticas apresentado em detalhe no capitulo seguinte deste trabalho.

6. Aplicação à gestão de *firewalls*

No presente capitulo, é apresentada a aplicação desenvolvida que permite estudar a aplicação da arquitectura de gestão por políticas à configuração de *firewalls*. Esta apresentação engloba a descrição da aplicação, dos componentes desenvolvidos para a sua implementação, da forma como as regras são geradas, distribuídas e traduzidas para comandos reconhecidos pelos tipos de equipamentos que suporta. Após esta descrição é efectuada uma avaliação da aplicação desenvolvida no contexto da gestão por políticas na qual se enquadra.

6.1 Descrição funcional

O sistema implementado é um protótipo que permite efectuar a aplicação da arquitectura PBN (*Policy-Based Networking*) à gestão de *firewalls,* utilizando como protocolo de difusão de políticas e linguagem de descrição, respectivamente, o protocolo COPS-PR (*Common Object Policy Service for Policy Provisioning***)** e a linguagem SPSL (*Security Policy Specification Language***)**.

De um modo simplista, uma *firewall* pode ser considerada como um mecanismo existente na interligação entre duas ou mais redes com o objectivo de controlar o tráfego que existe entre essas mesmas redes. Estes mecanismos lidam com o problema de filtrar as comunicações não pretendidas [Monteiro 2000].

A razão para a instalação de uma *firewall* é quase sempre proteger uma rede privada contra intrusões. Em muitos casos, o seu objectivo é também evitar que utilizadores não autorizados acedam a recursos existentes na rede privada e muitas vezes prevenir a

exportação não autorizada de informação. É de salientar que o termo *firewall* não se refere exclusivamente a uma máquina ou *software*, mas sim a um esquema de segurança construído para controlar o acesso entre redes de computadores.

Existem vários tipos de *firewalls* que são actualmente utilizados, sendo diferenciados entre si pelo tipo de tecnologia que está subjacente ao seu funcionamento. Na implementação apresentada foi utilizado o Filtro de Pacotes (*Packet Filters*), funcionalidade que permite ao administrador da rede decidir o tratamento a dar a cada pacote, nomeadamente, rejeitar o pacote, permitir a sua passagem ou redireccionar esses mesmo pacote para outro destino.

Nos Anexos A e B encontra-se, respectivamente, a descrição do funcionamento das listas de acesso da *Cisco* e do *software IPChains* que constituem exemplo de *Firewalls* por filtragem de pacotes.

Pelo facto de cada tipo de *firewall* ter um conjunto de instruções próprias para efectuar a sua configuração, pretende-se com a aplicação desenvolvida, partir da política de segurança global da organização e gerar um conjunto de regras que serão entendidas por cada um dos tipos de *firewall* que se pretenda gerir.

A implementação foi efectuada no sistema operativo *Linux RedHat* e pode ser utilizada para a configuração de sistemas de *firewall* que utilizem o *software IPChains* [Russel 2000] e/ou listas de acesso da *Cisco* (ACL) [Cisco 2001]. Foram utilizados estes dois sistemas de *firewall* devido à sua grande divulgação. No caso do *IPChains*, este *software* foi substituído pelo *software IPTables* a partir da versão 2.3.15 do núcleo do sistema operativo *Linux*. Pelo facto de o *IPTables* não trazer novidades relevantes para o sistema desenvolvido, manteve-se o uso do *IPChains*.

Na Figura 6.1 apresenta-se um cenário de utilização do protótipo desenvolvido. Neste cenário, o servidor de políticas de segurança (PDP – *Policy Decision Point*) é usado para efectuar a configuração de duas *firewall's* (PEP – *Policy Enforcement Point*). No cenário apresentado, podemos verificar a existência de dois domínios diferentes para a aplicação de políticas de segurança: A ligação à Internet e a ligação a uma *extranet*. Nesta situação, as políticas a por em prática em cada um dos domínios serão necessariamente diferentes. A aplicação desenvolvida pode ser facilmente integrada com outras que operem em áreas distintas, por exemplo, no caso apresentado, pode

também ser usado o mesmo servidor de políticas para efectuar alguma configuração de qualidade de serviço nos *routers*, desde que o PDP e os PEPs suportem esse tipo de cliente/serviço.

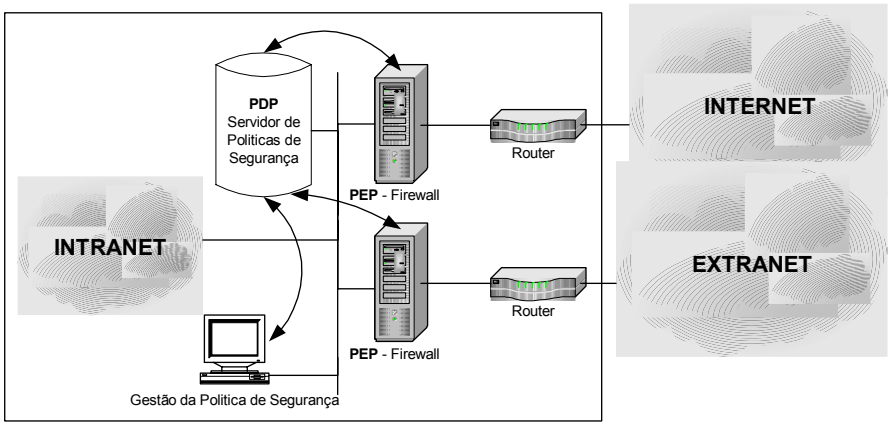

Figura 6.1 – Cenário de utilização da aplicação desenvolvida

6.2 Aspectos de implementação

Tendo em consideração a arquitectura de gestão por políticas (PBN), assim como o funcionamento do protocolo COPS e COPS-PR, desenvolveu-se um sistema cuja arquitectura está representada na Figura 6.2. Esta arquitectura é composta por três componentes principais: consola de gestão, PDP (*Policy Decision Point*) e PEP (*Policy Enforcement Point*).

A consola de gestão é uma ferramenta que permite a introdução e o tratamento de políticas de segurança escritas na linguagem SPSL. Neste protótipo, a consola de gestão funciona sem interface gráfica, contendo, apesar da sua simplicidade, um conjunto de módulos de grande relevância para o processamento da linguagem de descrição de políticas, nomeadamente, integra um *parser* e mecanismos de verificação e normalização de regras. Deste modo, um ficheiro contendo uma descrição de políticas em SPSL, é tratado na consola de gestão, gerando informação passível de ser armazenada no repositório de políticas ou, como no actual protótipo, passível de ser integrada na estrutura de dados existente no PDP.

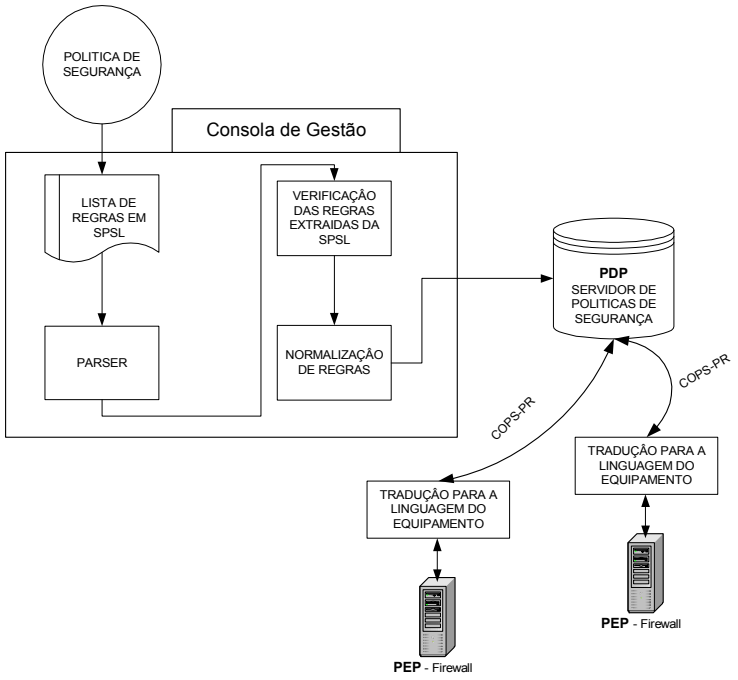

Figura 6.2 – Arquitectura do sistema proposto

O servidor de políticas (PDP) está desenvolvido de modo a ser o módulo responsável pela distribuição das políticas pelos elementos da rede. Na aplicação construída, o PDP integra também o repositório de políticas proposto na arquitectura PBN.

O destinatário das políticas enviadas pelo PDP, o PEP, é responsável pela tradução e instalação dessas mesmas políticas nos equipamentos que as devem implementar. No sistema proposto o PEP está em funcionamento no próprio equipamento.

A linguagem de especificação de políticas (SPSL) usada para esta implementação, apresenta algumas limitações no que respeita à sua utilização para especificar regras referentes à filtragem de pacotes, pois apenas tem definidas as acções *permit*, *deny* e *forward*. Com o objectivo de alargar o seu âmbito de aplicação, foi definido como extensão à linguagem a acção *reject* e porta para re-direccionamento, pois estas acções podem ser usadas nas regras implementadas na *firewall* do *kernel* do *linux*, recorrendo à aplicação que permite efectuar a manutenção destas regras, *IPChains.* [Caldeira 2000]

Os protocolos COPS e COPS-PR estão implementados de forma independente da aplicação, tendo neste âmbito sido desenvolvida uma PIB (*Policy Information Base*), utilizada pelo protocolo COPS-PR para o transporte de políticas referentes a *firewalls*.

Os componentes referidos, são apresentados em detalhe ao longo do presente capitulo.

Após a definição da política de segurança global criada para a organização, esta deve ser escrita num formato bastante simples e com um elevado grau de abstracção – a linguagem SPSL. Esta linguagem é processada ao nível da consola de gestão onde são verificadas regras como a sintaxe da linguagem ou num nível superior a verificação de sobreposição de objectivos de segurança [Caldeira 2000].

Depois da colocação da política na consola de gestão, esta informa o PDP de alterações efectuadas à política de segurança que deve ser posta em prática no seu campo de acção. O PDP inicia assim a sua tarefa de informar todos os PEPs em funcionamento, das regras que estes devem instalar. Cada PEP ao obter informação sobre as políticas que deve utilizar, é responsável pela sua tradução para o equipamento que lhe está subjacente assim como pela sua instalação do equipamento.

A informação é armazenada numa PIB definida para o efeito, sendo que todos os intervenientes no processo devem ser capazes de interpretar essa mesma PIB.

Implementação do protocolo COPS / COPS-PR

Na implementação do protocolo COPS/COPS-PR foi usado como ponto de partida um protótipo de uma API (*Application Programming Interface*) desenvolvida no decorrer de investigação realizada no laboratório TELIA – Suécia[10] . Esta API[11] foi melhorada para permitir todos os tipos de mensagens existentes no RFC actual que define estes protocolos.

A API resultante contêm o protocolo COPS e baseado neste protocolo foi feita a implementação do protocolo COPS-PR.

O resultado deste trabalho é a existência de dois módulos de *software*. Um contendo o protocolo COPS e outro com o seu modelo *provisioning* – COPS-PR. As APIs de acesso a estes módulos são livrarias escritas na linguagem C, compatíveis com os sistemas *Linux* e *FreeBSD*.

[10] http://extwww.lulea.trab.se/

[11] Disponível em http://extwww.lulea.trab.se/cops/

A API recebe comandos do utilizador (programa) e devolve eventos. Está dividida em duas componentes principais: A API para implementar o PDP e outra para o PEP.

São disponibilizados mecanismos de autenticação usando o algoritmo HMAC (*Keyed-Hashing for Message Authentication*) [Krawczyk 1997], sendo que a gestão de chaves é da responsabilidade do programador.

Quando é necessária alguma intervenção do utilizador da API, esta desencadeia eventos de um determinado tipo. Um evento é uma estrutura *typedef* que contem informação relacionada com o acontecimento que ocorreu na sessão COPS. A definição de todos os eventos pode ser encontrada no ficheiro *cops_event.h* existente na livraria.

Todos os eventos têm um campo em comum, o campo *sock_field*, usado para descrever qual a ligação PEP-PDP em que ocorreu o evento. Os eventos implementados na API para o protocolo COPS são os apresentados na Tabela 6.1.

A API COPS-PR está implementada usando as funcionalidades existentes na API do COPS, fornecendo funções adicionais não existentes no protocolo base.

Os eventos existentes são passados da livraria para o utilizador usando o mesmo modelo especificado para o protocolo COPS. Um evento pode conter informação que não necessita de nenhuma acção imediata mas que pode ser de interesse para o utilizador. De modo semelhante à API do COPS, cada evento é uma estrutura do tipo *typedef* contendo informação relacionada com algo que ocorreu numa sessão COPS-PR. Os eventos gerados pela API COPS são primeiro recebidos pela API COPS-PR e caso façam sentido para o uso deste protocolo são processados e enviados ao utilizador no formato COPS-PR. Caso não façam sentido para a livraria COPS-PR, são enviados directamente para o utilizador. Este facto significa que uma aplicação que utilize o protocolo COPS-PR deverá ter que tratar eventos originados pela livraria COPS e COPS-PR. Os eventos implementados na API, específicos para o protocolo COPS-PR, são apresentados na Tabela 6.2.

Evento	Aplica-se a	Descrição
COPS_EVENT_CONNECT	PDP	Após uma ligação ser iniciada por um PEP, o PDP recebe este evento antes de qualquer mensagem ser enviada ou recebida.
COPS_EVENT_DISCONNECT	PDP e PEP	Gerado quando uma ligação é terminada por uma das partes. É o ultimo evento gerado para uma ligação.
COPS_EVENT_REQUEST	PDP	Correspondendo à mensagem *Request Message*, definida no protocolo COPS, significa que um PEP efectuou um pedido ao PDP. Este evento deve tratado com o envio de uma mensagem do tipo decisão para o cliente. A estrutura de dados gerada, contem a especificação do cliente (características) assim com dados sobre a informação que o cliente possa conter no seu repositório local.
COPS_EVENT_DECISION	PEP	Evento indicativo de que uma mensagem foi recebida do PDP. Um evento deste tipo contêm associado uma decisão, uma lista de decisões ou ainda uma mensagem de erro, por exemplo no caso de que o PDP não suporte o tipo de PEP especificado aquando do pedido de ligação.
COPS_EVENT_ACCEPT	PEP	Quando este evento é recebido no PEP significa que a sessão COPS com um determinado PDP foi aberta com sucesso.
COPS_EVENT_REPORT	PDP	Este evento contém informação recebida numa mensagem "*COPS Report*"
COPS_EVENT_REDIRECT	PDP e PEP	Do lado do PDP, este evento é recebido no caso de o PEP ter sido redireccionado para outro PDP. Um redireccionamento pode ocorrer nos casos em que o PDP não suporte o tipo de cliente para o qual o PEP pretende obter serviços. Este evento é recebido pelo PEP quando o PDP informa que não suporta este tipo de cliente. Neste caso o PEP recebe uma mensagem com algum tipo de erro ou então um endereço de um novo PDP que suporte o tipo de cliente pretendido. A responsabilidade do conteúdo desta mensagem é do programador e não do protocolo.
COPS_EVENT_ERROR	PDP e PEP	Corresponde ao recebimento de uma mensagem de erro, normalmente correspondendo a um fecho da sessão.
COPS_EVENT_OPEN	PDP	Quando uma mensagem "*Accept*" foi enviada para um PEP, o PDP recebe este tipo de evento. É de salientar que este evento não garante que o PEP receba a mensagem que foi enviada, significando apenas que essa mesma mensagem foi enviada. A mensagem "*Accept*" é enviada automaticamente quando uma mensagem do tipo "*Open*" é recebida de um PEP.
COPS_EVENT_DELETE	PDP	Este evento corresponde a uma mensagem do tipo "*COPS Delete*".
COPS_EVENT_SYNC	PDP e PEP	De modo semelhante ao evento "*Redirect*", este evento significa situações diferentes caso ocorra no lado do PEP ou no lado do PDP. No caso de ocorrer no PEP, corresponde a um pedido de sincronização do protocolo COPS. Se ocorrer no PDP significa que o emissor deste evento terminou de efectuar a sincronização prevista no protocolo.
COPS_EVENT_AUTH	PDP	Este evento ocorre durante a fase de negociação de autenticação caso esta exista. A função de autenticação existente na mensagem deve ser executada, pois caso não o seja a ligação será terminada. A livraria actual não contem nenhum serviço de gestão de chaves, sendo o programador responsável por executar este tipo de operações.

Tabela 6.1 – Resumo dos eventos implementados na API do protocolo COPS

Evento	Aplica-se a	Descrição
COPS_PR_EVENT_REQUEST	PDP	Evento gerado quando a fase de ligação de um PEP ao PDP for concluída. É usado para informar que um PEP efectuou uma tentativa para estabelecer ou restabelecer uma ligação ao PDP. Neste evento existe uma mensagem associada contendo informação acerca do PEP.
COPS_PR_EVENT_REPORT	PDP	Deve ser emitido um evento deste tipo sempre que um PEP envie uma mensagem do tipo "*Report*" para o seu PDP. Actualmente esta funcionalidade não se encontra implementada.
COPS_PR_EVENT_DECISION	PEP	Contem a indicação fornecidas pelo PDP, de quais as alterações que o PEP deve efectuar na sua PIB local. Este evento deve ser tratado pelo utilizador da API, de modo a que estas alterações sejam correctamente implementadas na PIB do PEP. A mensagem associada contem informação de quais as políticas que devem ser adicionadas, assim como de quais devem ser alteradas ou removidas.
COPS_PR_EVENT_HANDLE	PDP	Sempre que um PEP emita uma mensagem do tipo "*Request*", nos casos em que pretende efectuar uma sincronização ou uma nova ligação, este evento ocorre. Contem a identificação do PEP e caso seja necessário uma nova identificação.

Tabela 6.2 – Resumo dos eventos implementados na API do protocolo COPS-PR

Os módulos de *software* que implementam os protocolos COPS e COPS-PR permitem ao utilizador um elevado nível de abstracção do protocolo, contudo, existem muitas situações em que o programador tem que efectuar as acções adequadas.

Após o inicio de uma sessão aberta pelo PEP, o utilizador terá que ter a preocupação de introduzir a informação de configuração na nova PIB criada no PEP. No caso de a ligação ser interrompida, é da responsabilidade do programador executar as acções necessárias ao seu restabelecimento. Por exemplo, se o PEP não puder contactar o PDP durante um determinado período de tempo, este pode tentar a ligação para outro PDP (caso exista). Este tipo de mecanismos não são automaticamente realizados pala API. É também da responsabilidade do utilizador a validação dos dados existentes na PIB, assim como verificar a validade do seu conteúdo, caso a sua ligação seja perdida.

Implementação do PDP

Como o sistema é baseado no modelo cliente / servidor, podemos considerar o PDP como o servidor do sistema em análise.

Na solução apresentada, é dado conhecimento ao PDP, na forma de parâmetros na linha de comando, da localização do ficheiro que contém as políticas escritas em SPSL. Ao utilizar este ficheiro, o PDP efectua as operações necessárias à geração de regras (descritas anteriormente).

Ao ter conhecimento das regras existentes, o PDP cria uma PIB com essa mesma informação, no seu espaço de memória.

Após a PIB ter sido criada, o PDP cria também estruturas de dados sobre as quais serão guardadas informações sobre os clientes ligados, os tipos de clientes que suporta assim como diversas informações necessárias ao seu funcionamento, como por exemplo uma lista de PDPs alternativos para os quais ele pode redireccionar os seus clientes se assim o entender.

Neste ponto é inicializado o protocolo COPS através de funções definidas para o efeito na API, o qual começa a aguardar ligações dos PEPs.

Implementação do PEP

Pelo facto de o PEP ser o responsável pela implementação das políticas existentes nos equipamentos que gere, foram criadas duas implementações diferentes, tendo em consideração o equipamento que está subjacente ao PEP: uma *firewall IPChains* e um *router Cisco*. As duas implementações são bastante semelhantes, diferenciando-se apenas no modulo de tradução de regras e na forma como é feita a implementação dessas mesmas regras.

Cada PEP ao ser iniciado, cria uma PIB local no seu espaço de memória, contendo inicialmente informação respeitante ao equipamento que suporta (tipo de equipamento, n.º de portas, domínio de gestão, etc.). Após a criação desta estrutura de dados, estabelece uma ligação ao PDP especificado a sua configuração, enviando-lhe a informação existente na sua PIB local. Uma vez estabelecida a ligação, o sistema passa a aguardar mensagens enviadas pelo PDP. A informação enviada inicialmente vai permitir informar o PDP de quais as regras que devem ser postas em prática por um determinado PEP, por exemplo caso o PEP forneça um domínio de acção, o PDP apenas vai enviar regras referentes a esse mesmo domínio.

No protótipo implementado, a PIB local criada em cada PEP contém já uma regra que nega todo o tráfego no equipamento, de modo a garantir a segurança de todo o sistema durante a inicialização do mesmo.

Para cada mensagem do tipo *decision* recebida do PDP, a PIB local é alterada de modo a reflectir as decisões recebidas.

Sempre que é efectuada uma alteração na PIB local do PEP é efectuada a tradução das regras recebidas para a sintaxe reconhecida pelo equipamento que as implementa.

No caso do *IPChains*, o PEP efectua directamente a sua configuração, pois é suposto que nesta situação o PEP esteja a funcionar na mesma máquina que o *IPChains*. Deste modo o conjunto de regras *IPChains* existente no *kernel Linux* é mantido permanentemente sincronizado com as regras existentes no PDP, para o domínio em questão.

Na implementação efectuada para suportar as ACLs existentes nos equipamentos da *Cisco*, para cada alteração na PIB local, o PEP gera um ficheiro de texto contendo as novas ACLs a serem implementadas posteriormente no *router*. Dependendo do equipamento, o PDP pode também enviar essa informação para o *router*, sincronizando assim as regras de uma forma automática, utilizando por exemplo o protocolo COPS (caso seja suportado pelo *router*), SNMP ou ainda editando o seu ficheiro de configuração.

A aplicação subjacente ao PEP entra assim num ciclo infinito esperando mensagens do PDP e efectuando as acções correspondentes. Durante a execução deste ciclo é verificada também a ligação ao PDP, sendo tomadas as medidas necessárias caso esta ligação seja quebrada.

Como já foi referido, a PIB utilizada armazena informação sobre variados aspectos, como por exemplo a configuração de cada PEP. Na implementação efectuada, foi utilizado uma PIB definida para QoS, a qual foi expandida para suportar regras referentes a filtros de pacotes. Na Figura 6.3 encontra-se representada a estrutura de dados (excerto da PIB global) que é utilizada pelo protocolo COPS-PR para enviar as regras do PDP para os PEPs.

```
typedef struct pib_ip_firewall {
int pri_type;
   u_int32_t index;
   int ref_count;
   int simu_ref_count;

  int   numero;
  char  nome[255];
  char dominio[255];

  char accao[255]; /* deny ou permit */

  int  forward; /* 0 se não existe forward ou 1 se existe */
  char ipforward[20]; /* Endereço para forward */
  char portforward[20]; /* Porta para forward */
  char protoforward[20]; /* Protocolo para forward */

  int  src_not; /* 0 ou 1 se existe o not nesta gama de endereços */
  int  src_wild; /* 0 ou 1 se esta gama e constituída por asterisco */
  int  src_mask_min_max;/* 0:max é mask, -1:max nao existe, 1:max e máximo */
  char src_min[20];   /* string com o valor mínimo da gama */
  char src_max[20]; /* string com o valor máximo da gama */

  int  dst_not;
  int  dst_wild;
  int  dst_mask_min_max;
  char dst_min[20];
  char dst_max[20];

  int  src_port_not;
  int  src_port_wild;
  int  src_port_mask_min_max;
  char src_port_min[20];
  char src_port_max[20];

  int  dst_port_not;
  int  dst_port_wild;
  int  dst_port_mask_min_max;
  char dst_port_min[20];
  char dst_port_max[20];

  int  proto_port_not;
  int  proto_port_wild;
  int  proto_port_mask_min_max;
  char proto_port_min[20];
  char proto_port_max[20];

  char dir[255]; /* Direcção da regra */

   u_int32_t acl_idx;
} pib_ip_firewall;
```

Figura 6.3 – Excerto da PIB implementada

Este excerto da PIB utilizada (*pib_ip_firewall*), pretende representar todos os dados utilizados na linguagem SPSL que se referem a filtros de pacotes. Cada PEP ao receber um conjunto de estruturas deste tipo é capaz de traduzir esta informação para a linguagem do equipamento.

6.3 Geração e verificação de regras

A política de segurança definida para a organização é descrita usando a linguagem SPSL, actualizada com as extensões já referidas. O protótipo implementado usa um

ficheiro de texto onde está armazenada essa mesma política e gera um conjunto de regras que devem ser armazenadas num repositório de políticas e posteriormente postas em prática. Nesta implementação, podemos considerar que o repositório de políticas é o ficheiro de texto com a informação em SPSL. Após o necessário processamento a esse ficheiro, as regras são de imediato enviadas para o PDP.

Neste contexto, o primeiro aspecto a ser tratado pela aplicação desenvolvida é a verificação da sintaxe do ficheiro que contém as políticas escritas usando linguagem SPSL [Caldeira 2000]. Para tal é usado um *parser* gerado usando *bison* [Donnelly 1995] e *flex* [Paxson 1995], de modo a verificar a sintaxe definida pela BNF (*Backus-Naur Form*) em que se baseia a linguagem. Ao ser verificada a sintaxe da linguagem é criada também um estrutura de dados que contém todos os objectos válidos. Além disso, são geradas mensagens de erro quando é encontrado algum objecto que não esteja escrito de forma correcta, excluindo apenas esse objecto da estrutura de dados resultante, permitindo assim continuar a processar as restantes regras, não comprometendo a segurança global da organização.

Os objectos são tratados individualmente, tendo em consideração o seu tipo, *domain*, *policy*, etc, de modo a poder ser criada uma nova estrutura que contenha apenas as políticas retiradas dos objectos *policy*, associando de imediato cada uma destas políticas ao objecto base que lhe deu origem, de modo a conseguir identificar, por exemplo, a que domínio essa política está associada.

Como exemplo [Caldeira 2000] podemos verificar que partindo dos objectos representados na Figura 6.4, ao executar o *parser* e a criação das estruturas de dados, o resultado será o apresentado na Figura 6.5.

O que é feito neste ponto é a extracção de regras da política de segurança global, referentes a um ou vários domínios.

```
domain:          IPV-SERVERS
coverage:        SERVER1,SERVER2
mnt-by:          FILIPE
changed:         FILIPE 20000823
signature:       FILIPE FILIPE-CERT \
                 KEYEDMD5 ABBA007AB47E

policy-name:     POL1
association:     IPV-SERVERS
policy:          dst 193.137.7.1  src * \
                 xport-proto 6 permit
mnt-by:          FILIPE
changed:         FILIPE 20000823
signature:       FILIPE FILIPE-CERT \
                 KEYEDMD5 ABBA007AB47E

policy-name:     POL2
association:     IPV-SERVER
policy:          dst 193.137.7.2 \
                 src 193.137.6.0/24  \
                 xport-proto 6 permit
mnt-by:          FILIPE
changed:         FILIPE 20000823
signature:       FILIPE FILIPE-CERT \
                 KEYEDMD5 ABBA007AB47E

policy-name:     DEFAULT
association:     IPV-SERVER
policy:          dst * src any deny
mnt-by:          FILIPE
changed:         FILIPE 20000823
signature:       FILIPE FILIPE-CERT \
                 KEYEDMD5 ABBA007AB47E
#
```

Figura 6.4 – Exemplo de uma política escrita em SPSL

```
.......
Nº: 2
Associado a: IPV-SERVERS
Nome: POL1
Accao: Permit
Origem:  *
Destino: 193.137.7.1
Protocolo: 6

Nº: 3
Associado a: IPV-SERVER
Nome: POL2
Accao: Permit
Origem: 193.137.6.0/255.255.255.0
Destino: 193.137.7.2
Protocolo: 6

Nº: 4
Associado a: IPV-SERVER
Nome: DEFAULT
Accao: Deny
Origem:  *
Destino:  *
```

Figura 6.5 – Estrutura de dados gerada a partir da definição de uma política

Depois de garantir que a sintaxe está correcta, o algoritmo utilizado para gerar as regras examina a sua estrutura, excluindo regras sempre que a lógica das mesmas seja incorrecta, por exemplo o uso de uma gama de endereços em que o valor mínimo seja menor que o valor máximo, (Ex: 193.134.56.7 - 193.134.55.1) ou mesmo para endereços de rede inválidos (Ex. 193.3.3.5/24).

Verificação de regras

Uma vez validadas as regras, poderão ainda existir alguns problemas quanto à sua lógica, por exemplo a existência de duas regras contraditórias ou a existência de duas ou mais regras cujas gamas de endereços se intersectem.

Uma vez que a aplicação das regras segue a ordem pela qual elas foram escritas, a intersecção de endereços não deverá causar problemas na sua aplicação, contudo como o que se pretende é a definição de uma política global, a existência de regras que nunca

irão ser aplicadas leva a um aumento da complexidade dessa mesma política. Neste caso, pode ser aplicado um algoritmo que tem como objectivo a simplificação de regras.

```
.......                                     policy-name:     POL2
policy-name:     POL1                       association:     IPV-SERVER
association:     IPV-SERVERS                policy:          dst 193.137.7.2  \
policy:          dst not 193.137.7.1 \                       src 187.100.100.0/24 \
                 src * xport-proto 6 permit                  xport-proto 6 permit
mnt-by:          FILIPE                     mnt-by:          FILIPE
changed:         FILIPE 20000823            changed:         FILIPE 20000823
signature:       FILIPE FILIPE-CERT \       signature:       FILIPE FILIPE-CERT \
                 KEYEDMD5 ABBA007AB47E                       KEYEDMD5 ABBA007AB47E
                                            #
```

Figura 6.6 – Exemplo SPSL

No exemplo apresentado na Figura 6.6, a primeira regra permite a ligação TCP, de qualquer endereço, para uma qualquer máquina do nosso domínio, à excepção da 193.137.7.1. A segunda regra permite a ligação TCP para a máquina 193.137.7.2, caso a ligação seja efectuada partindo dos endereços 187.100.100.0/24. Pode-se então verificar que a primeira regra generaliza a segunda ao permitir que qualquer máquina se ligue ao nosso endereço 193.137.7.2, pois este endereço é diferente de 193.137.7.1, fazendo com que a segunda regra nunca seja aplicada.

Usando o algoritmo para simplificação das regras, que transforma todos os endereços em gamas de endereços e verifica se existem intersecções, as duas políticas anteriores seriam transformadas apenas na seguinte regra, apresentada na Figura 6.7.

```
N°: 1
Associado a: IPV-SERVER
Nome: POL1-01
Accao: Permit
Origem: *
Destino: 0.0.0.0- 193.137.7.0,
         193.137.7.2-255.255.255.255
Protocolo: 6
```

Figura 6.7 – Simplificação de regras

Neste caso, o resultado final é equivalente à descrição anterior, usando apenas uma regra que permite ligações TCP de qualquer origem para o conjunto de endereços 0.0.0.0 – 193.137.7.0 e 193.137.7.2-255.255.255.255, isto é todas excepto a máquina com endereço 193.137.7.1.

O algoritmo usado também verifica intersecções com outros elementos das regras, como os portos de origem destino e o protocolo usado.

6.4 Armazenamento de regras

Nesta implementação, as regras estão armazenadas num ficheiro de texto, contendo as regras escritas em SPSL.

Uma outra abordagem possível no desenvolvimento desta arquitectura é o uso de um sistema de gestão de bases de dados (SGBD) com suporte para XML [Bray 1998] armazenando uma PIB ou ainda num nível mais elevado de abstracção, armazenando informação num modelo de dados baseado no CIM (*Common Information Model*) [CIM 1999]. Neste caso, as políticas podem ser armazenadas no SGBD na forma de tabelas relacionais ou ainda na forma de objectos, no caso do modelo objecto-relacional dos SGBD. As consultas feitas pelos PDPs ao repositório de políticas pode neste caso ser feita usando a linguagem SQL, a qual irá devolver um ficheiro XML, entendido pelo PDP.

Neste caso, pode também ser facilmente implementada uma consola de gestão baseada numa interface Web que permita interagir com a bases de dados no formato XML. Uma vantagem que pode existir no uso de XML é o facto de este ser já uma norma criada pelo W3C e ser bastante extensível no que respeita ao formato de dados a ser trocado.

A aplicação desenvolvida permite, criar um ficheiro XML com um DTD (*Document Type Definition*) associado, que define o conjunto de regras que podem posteriormente recebidas e traduzidas pelo módulo de tradução especifico de cada equipamento. Na Figura 6.8 é apresentado um ficheiro XML gerado pela aplicação, onde está descrito o DTD utilizado na versão actual, assim como é apresentada uma regra em formato XML obedecendo ao DTD definido. Este DTD pretende efectuar um mapeamento entre a estrutura de dados PIB existente e um ficheiro XML.

A implementação de uma estrutura de dados XML, permite que a aplicação possa distribuir informação de configuração para equipamentos que não tenham suporte ao protocolo COPS.

```
<?xml version="1.0" encoding="windows-1252"?>
<!DOCTYPE  ListaRegras [
                <!ELEMENT ListaRegras (regra)*>
                <!ELEMENT   regra   (nome,   dominio?,   accao,   forward?,   ipforward?,
                portforward?,       protoforward?,  src+,  dst+,  src_port*,  dst_port*,
proto*,         dir?, extras?)>
                <!ATTLIST regra  num  ID #REQUIRED>
                <!ELEMENT nome (#PCDATA)>
                <!ELEMENT dominio (#PCDATA)>
                <!ELEMENT accao (#PCDATA)>
                <!ELEMENT forward (#PCDATA)>
                <!ELEMENT ipforward (#PCDATA)>
                <!ELEMENT portforward (#PCDATA)>
                <!ELEMENT protoforward (#PCDATA)>
                <!ELEMENT src (not?, wild?, mask_min_max, min?, max?)>
                <!ELEMENT dst (not?, wild?, mask_min_max, min?, max?)>
                <!ELEMENT src_port (not?, wild?, mask_min_max, min?, max?)>
                <!ELEMENT dst_port (not?, wild?, mask_min_max, min?, max?)>
                <!ELEMENT proto (not?, wild?, mask_min_max, min?, max?)>
                <!ELEMENT dir (#PCDATA)>
                <!ELEMENT extras (#PCDATA)>
                <!ELEMENT not (#PCDATA)>
                <!ELEMENT wild (#PCDATA)>
                <!ELEMENT mask_min_max (#PCDATA)>
                <!ELEMENT min (#PCDATA)>
                <!ELEMENT max (#PCDATA)> ]>
<ListaRegras>
                <regra num="_1">
                     <nome> POL1 </nome>
                     <dominio> IPV-SERVERS </dominio>
                     <accao> PERMIT </accao>
                     <src>
                            <not> 0 </not>
                            <wild> 0 </wild>
                            <mask_min_max> 0 </mask_min_max>
                            <min> 193.137.6.0 </min>
                            <max> 255.255.255.0 </max>
                </src>
                     <dst>
                            <not> 0 </not>
                            <wild> 0 </wild>
                            <mask_min_max> -1 </mask_min_max>
                            <min> 193.137.7.2 </min>
                </dst>
                     <proto>
                            <not> 0 </not>
                            <wild> 0 </wild>
                            <mask_min_max> -1 </mask_min_max>
                            <min> 6 </min>
                     </proto>
                     <dir> inbound </dir>
                </regra>
</ListaRegras>
```

Figura 6.8 – Regra em formato XML obedecendo ao DTD definido

6.5 Distribuição e aplicação de regras

Com o objectivo de efectuar a distribuição e aplicação das regras geradas na consola de gestão de políticas, foi efectuada a implementação do PDP, PEP e dos protocolos de difusão de políticas COPS e COPS-PR.

A comunicação entre servidores e clientes de políticas pode ser realizada, na aplicação desenvolvida, de dois modos independentes: transferência de informação no formato

XML ou através do uso de um protocolo especifico para esse efeito, como é o caso do protocolo COPS. Neste sentido, houve a necessidade de implementar suporte ao protocolo COPS e COPS-PR, pois ambos os protocolos encontram-se ainda em fase de discussão no âmbito do grupo de trabalho RAP do IETF.
Para além das capacidades de intercâmbio de informação com o PDP, foi implementado no PEP a capacidade de traduzir as regras recebidas em comandos reconhecidos pelo equipamento que este está a controlar.

Uma vez escrita e processada a política de segurança global da empresa, esta deverá ser aplicada a um ou mais equipamentos, como sejam as *firewalls* ou *routers*. Pretende-se assim que cada equipamento possa ser instruído através de um conjunto de instruções que este reconheça, geradas automaticamente por um tradutor.

Um dos problemas com que nos deparamos é o da existência de potencialidades diferentes entre cada tipo de equipamento. Neste caso decidimos que seriam geradas as regras que utilizando a sintaxe do equipamento em questão, mais se aproximem da regra original. Esta situação é comunicada ao utilizador, avisando quais as alterações efectuadas, bem como qual a função que não pode ser implementada usando a linguagem pedida.

As regras escritas usando SPSL têm maior flexibilidade do que normalmente encontramos nos equipamentos. Por exemplo, um dos aspectos iniciais da conversão é a passagem de gamas de valores, não suportadas pela maioria dos equipamentos, para valores atómicos.

A regra apresentada na Figura 6.9, terá que ser convertida em 4 regras diferentes através das combinações de todos os valores que não são atómicos, resultando então um novo conjunto de regras, descritas na Figura 6.10.

```
.......
policy-name:     POL1
association:     IPV-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:          FILIPE
changed:         FILIPE 20000823
signature:       FILIPE FILIPE-CERT KEYEDMD5 ABBA007AB47E
```

Figura 6.9 – Representação de uma regra em SPSL

```
Nº: 1                                   Nº: 3
Associado a: IPV-SERVERS                Associado a: IPV-SERVERS
Nome: POL1                              Nome: POL1
Accao: Permit                           Accao: Permit
Origem:  193.1.1.1                      Origem:  194.1.1.1
Destino: 193.137.7.1                    Destino: 193.137.7.1
Protocolo: 6                            Protocolo: 6
Direccao: inbound                       Direccao: inbound

Nº: 2                                   Nº: 4
Associado a: IPV-SERVERS                Associado a: IPV-SERVERS
Nome: POL1                              Nome: POL1
Accao: Permit                           Accao: Permit
Origem:  193.1.1.1                      Origem:  194.1.1.1
Destino: 193.137.7.2                    Destino: 193.137.7.2
Protocolo: 6                            Protocolo: 6
Direccao: inbound                       Direccao: inbound
```

Figura 6.10 – Regras resultantes do processo de simplificação

Esta acção conjugada com o algoritmo de simplificação de regras, que expande todos os valores para gamas de endereços e as gamas de endereços escritas originalmente, pode levar a um grande aumento do número final de regras. Para minorar este problema, as gamas de valores são convertidas em um ou mais valores atómicos. No caso das gamas de endereços IP, estas são transformadas no conjunto das redes que as constituem. Por exemplo se a regra simplificada tiver como endereço de destino a gama de valores 193.137.7.0 – 193.137.7.255, deverá ser transformada em 193.137.7.0/24, valor este que já pode ser aplicado na sintaxe da maioria dos equipamentos, não implicando efectuar a expansão da gama de endereços, pelo que consequentemente o número de regras finais será menor.

Actualmente o sistema desenvolvido suporta a conversão de SPSL para *IPChains* [Russel 2000] e ACL da *cisco* [Cisco 2001], podendo facilmente evoluir para suportar outro tipo de linguagem proprietária dos equipamentos. Esta conversão é executada usando os mecanismos descritos anteriormente, efectuando de seguida uma comparação dos atributos existentes nas regras com a BNF do *IPChains* ou das ACL, gerando gradualmente regras em cada uma das cadeias *IPChains* ou gerando novas ACLs. Sempre que não existam dados suficientes na regra, são usados valores por defeito e comunicada essa atribuição ao utilizador.

6.6 Exemplo de utilização

A introdução, através da consola de gestão, do extracto de quatro regras de uma política global escrita em SPSL, descrita na Figura 6.11, dá origem a um novo conjunto de

regras, descritas na Figura 6.12 (existem cinco regras, pois os protocolos existentes na política POL2, foram expandidos para formar duas regras diferentes).

```
policy-name:     POL1                          policy-name:     POL3
association:     IPV-SERVERS                   association:     IPV-SERVER
policy:          dst 193.137.7.1  src * \      policy:          dst 193.137.7.3 port 80 \
                 xport-proto 6 \                                 src * xport-proto 6 \
                 direction inbound permit                        direction inbound deny, \
mnt-by:          FILIPE                                          forward port 8080
changed:         FILIPE 20000823               mnt-by:          FILIPE
signature:       FILIPE FILIPE-CERT  \         changed:         FILIPE 20000823
                 KEYEDMD5 ABBA007AB47E         signature:       FILIPE FILIPE-CERT \
                                                                 KEYEDMD5 ABBA007AB47E
policy-name:     POL2
association:     IPV-SERVER                    policy-name:     DEFAULT
policy:          dst 193.137.7.2 port 110 \    association:     IPV-SERVER
                 src 193.137.6.0/24 \          policy:          dst any src any reject
                 xport-proto 6,17 \            mnt-by:          FILIPE
                 direction inbound permit      changed:         FILIPE 20000823
mnt-by:          FILIPE                        signature:       FILIPE FILIPE-CERT \
changed:         FILIPE 20000823                                 KEYEDMD5 ABBA007AB47E
signature:       FILIPE FILIPE-CERT \
                 KEYEDMD5 ABBA007AB47E         #
```

Figura 6.11 – Exemplo políticas em SPSL

```
N°: 1                                          Origem: 193.137.6.0/255.255.255.0
Associado a: IPV-SERVERS                       Destino: 193.137.7.2
Nome: POL1                                     Porto de destino: 110
Accao: Permit                                  Protocolo: 17
Origem:  *                                     Direccao: inbound
Destino: 193.137.7.1
Protocolo: 6                                   N°: 4
Direccao: inbound                              Associado a: IPV-SERVER
                                               Nome: POL3
N°: 2                                          Accao: Deny e Forward para porto 8080
Associado a: IPV-SERVER                        Origem:  *
Nome: POL2                                     Destino: 193.137.7.3
Accao: Permit                                  Porto de destino: 80
Origem: 193.137.6.0/255.255.255.0              Protocolo: 6
Destino: 193.137.7.2                           Direccao: inbound
Porto de destino: 110
Protocolo: 6                                   N°: 5
Direccao: inbound                              Associado a: IPV-SERVER
                                               Nome: DEFAULT
N°: 3                                          Accao: Reject
Associado a: IPV-SERVER                        Origem:  *
Nome: POL2                                     Destino:  *
Accao: Permit
```

Figura 6.12 – Regras extraídas de uma política em SPSL

O conjunto de regras resultante é então introduzindo na PIB do PDP existente, após o que este as irá enviar para cada PEP que controla.

Utilizando a PIB que contêm as regras, o PEP, através do módulo de tradução efectua a passagem das regras para a linguagem que as vai implementar. No protótipo desenvolvido, o PEP pode converter o conjunto de regras recebidas para *IPChains* ou listas de acesso da *Cisco*.

Na Figura 6.13, encontram-se as regras *IPChains* que o PEP implementa no equipamento.

```
Regra Nº 1,POL1, assoc: IPV-SERVERS
ipchains -A input –p 6 -d 193.137.7.1 –j ACCEPT

Regra Nº 2, POL2, 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

Regra Nº 3, POL2, 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

Regra Nº 4, POL3, assoc: IPV-SERVER
ipchains -A input –p 6 -d 193.137.7.3 80 –j REDIRECT 8080

Regra Nº 5, DEFAULT, assoc: IPV-SERVER
ipchains -A input –j REJECT
```

Figura 6.13 – Tradução para *IPChains*

No caso de o PEP estar a controlar um *router cisco*, os comandos que este recebe são os apresentados na Figura 6.14. É de realçar a diferença existente na tradução das regras n.^{os} 4 e 5. No caso da regra n.º 4, a funcionalidade de redireccionamento de pacotes para uma porta diferente da original tem formas de aplicação diferentes nos dois equipamentos. Devido a esta situação, no caso da configuração para o *router cisco*, é feita a criação de duas entradas na ACL, assim como é criada uma linha que especifica a tradução de um endereço (esta ultima linha não pertence à definição de ACLs). Em relação à regra nº5 podemos verificar que esta não existe na configuração do *router*. Este facto deve-se a que a ACL implementa por defeito uma regra equivalente a esta, pelo que neste caso esta regra é redundante.

```
Regra Nº 1 com nome POL1 , associada a: IPV-SERVERS
access-list 101 permit tcp any host 193.137.7.1

Regra Nº 2 com nome POL2 , associada a: IPV-SERVER
access-list 101 permit tcp 193.137.6.0 0.0.0.255 host 193.137.7.2 eq 110

Regra Nº 3 com nome POL2 , associada a: IPV-SERVER
access-list 101 permit udp 193.137.6.0 0.0.0.255 host 193.137.7.2 eq 110

Regra Nº 4 com nome POL3 , associada a: 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
```

Figura 6.14 – Tradução para *Cisco*

6.7 Avaliação da solução

Através da utilização do protótipo desenvolvido podemos avaliar a funcionalidade da arquitectura de gestão por políticas aplicada à gestão de *firewalls* numa rede de computadores.

Existindo a definição de uma política de segurança global para a organização, o uso da aplicação apresentada pode melhorar a forma de implementar essa mesma política nos vários equipamentos existentes na rede, principalmente pelo facto de automatizar os procedimentos de configuração de um conjunto de sistemas independentes. Paralelamente, permite ao administrador da rede não necessitar de ter um conhecimento aprofundado das linguagens ou técnicas usadas para configurar cada um dos dois tipos de equipamentos suportados pela aplicação.

O facto de a aplicação proposta estar, nesta fase, vocacionada para lidar apenas com políticas referentes à filtragem de pacotes, introduz limitações relacionadas com a falta de mecanismos para configurar outros aspectos dos equipamentos. Por exemplo, ao permitir configurar listas de acesso de um *router cisco*, também deveria ser possível com o sistema proposto descrever o funcionamento da rede e gerar a adequada configuração do *router*.

A linguagem utilizada para especificação de políticas (SPSL) mostrou-se adequada para o cenário de aplicação previsto, permitindo com facilidade exprimir as políticas a aplicar na rede. Contudo, mesmo sendo expansível, verifica-se que esta linguagem não é suficientemente abrangente, no que concerne à especificação de políticas em várias áreas da gestão de redes, para ser usada num cenário mais alargado no âmbito de gestão por políticas.

O módulo de tradução de regras para comandos *IPChains* e ACL da cisco introduz a possibilidade de a configuração gerada para os equipamentos não ser a mais indicada, pois por vezes não se trata apenas uma questão de traduzir a sintaxe mas de manter a semântica pretendida.

No que concerne ao protocolo de difusão de políticas usado (COPS-PR), verifica-se que este protocolo permite que a comunicação entre os PDPs e os PEPs se processe de uma forma bastante simples e eficaz.

A consola de gestão utilizada, funcionando ao nível da linha de comandos, não traz vantagens relativamente ao modo tradicional de configurar os equipamentos. É sentida a ausência de uma interface gráfica (GUI) que permita ao administrador obter rapidamente uma visão geral acerca das políticas existentes.

Para concluir esta avaliação, pode-se afirmar que a aplicação traz vantagens a uma organização, se for vista como uma aplicação de gestão centralizada do tipo de *firewalls* que suporta, por exemplo, num cenário onde existam vários sistemas *IPChains* a aplicação é útil para garantir que esses sistemas são configurados correctamente segundo a política global da organização.

6.8 Síntese

Neste capitulo apresentou-se o protótipo desenvolvido que aplica o uso de políticas à gestão de *firewalls*. Para a implementação efectuada foram desenvolvidos módulos de forma a que a sua arquitectura corresponda ao paradigma de gestão por políticas apresentado no segundo capitulo deste trabalho. Neste contexto, discutem-se os componentes desenvolvidos para as tarefas de geração, verificação e armazenamento de regras, assim como é abordada a forma como as regras são distribuídas e convertidas para comandos reconhecidos pelas *firewalls* que o sistema suporta (*IPChains* e ACL da *cisco*). De modo a simplificar o entendimento da aplicação desenvolvida efectua-se a apresentação de um pequeno exemplo de utilização.
Conclui-se o capitulo, com uma avaliação do protótipo criado, no âmbito da área onde este se integra que é a gestão por políticas (PBN).

7. Conclusão e trabalho futuro

Neste capítulo apresenta-se a conclusão desta dissertação e as perspectivas de trabalho futuro a realizar na área de gestão por políticas.

7.1 Conclusão

A realização deste trabalho revelou-se uma tarefa exigente, mas mobilizadora e interessante. Como principal dificuldade salientam-se as inconsistências e omissões de detalhes em alguns documentos consultados, bem como o facto de estarmos perante tecnologias ainda em desenvolvimento. O contacto com a arquitectura de gestão por políticas desperta necessidades de aprofundar conhecimentos em várias vertentes. Este estudo é enriquecedor e contribui para a consolidação de conhecimentos mas, nalguns casos, acaba por ser abordado de forma abreviada. Dada a abrangência do tema, este trabalho é inevitavelmente uma obra incompleta. Porém, apesar das eventuais omissões, atingiram-se os principais objectivos orientadores da dissertação referidos no Capítulo 1.

Ao longo deste trabalho procurou-se apresentar as potencialidades e limitações da arquitectura de gestão por políticas, bem como evidenciar as diferenças entre as metodologias tradicionais de gestão de redes e esta nova abordagem.

A gestão por políticas em áreas específicas da gestão de redes de computadores representa uma visão do futuro a curto ou médio prazo, pois a expansão e o ritmo da evolução desta arquitectura leva-nos a ser optimistas. No entanto, do presente estudo, pode-se verificar que, no que concerne à gestão de todos os equipamentos da rede, a

gestão por políticas ainda é, actualmente, uma arquitectura cuja implementação generalizada está bastante distante. De um modo geral, é possível dizer que estamos perante uma arquitectura com elevado potencial no sentido de possibilitar a redução de custos e melhorar a qualidade do serviço prestado em redes de computadores. Por outro lado, actualmente, a oferta de produtos nesta área engloba apenas uma pequena parte das vantagens desejáveis. Neste momento, os projectos de investigação em curso começam a demonstrar as vantagens e serviços que o uso desta arquitectura pode proporcionar. A configuração "*plug and play*" de uma rede com equipamentos heterogéneos partindo de um conjunto de políticas escritas numa linguagem de alto nível ainda não é uma solução generalizada. Muitos fabricantes apresentam soluções proprietárias de gestão por políticas para os seus equipamentos, mas a sua aplicação a equipamentos de outros fabricantes é, na grande maioria dos casos, impossível.

A construção do protótipo permite concluir que a aplicação desta metodologia, de uma forma generalizada, à configuração de um grande numero de *firewalls* de diferentes fabricantes é uma tarefa árdua se não existir o apoio desses mesmos fabricantes.

Conclui-se também, através deste estudo, que os protocolos COPS e COPS-PR apresentam potencialidades interessantes para serem escolhidos em soluções de gestão por políticas.

No que concerne à linguagem de especificação utilizada, esta, sendo indicada para o trabalho desenvolvido, não é suficientemente abrangente para ser uma solução adoptada de uma forma generalizada para a gestão por políticas. Sendo bastante flexível, esta linguagem deverá ser expandida antes de se poder tornar um padrão, pois como o seu objectivo inicial foi a utilização de IPSec e IKE, existem algumas lacunas no que diz respeito à descrição de regras para configuração de *firewalls*.

7.2 Trabalho futuro

Como trabalho futuro deve ser prevista a extensão da linguagem proposta, ou mesmo a adopção de uma nova linguagem, de forma a que a tradução das políticas para *IPChains* e ACL da *cisco* fique completa, o que permite facilitar a configuração de *firewalls* deste tipo numa organização de tamanho considerável. Para além desta funcionalidade, o sistema proposto deverá prever o desenvolvimento de mecanismos adicionais de

tradução para outros equipamentos. Tendo em consideração a modularidade do sistema proposto, efectuar alterações desta natureza implica apenas modificações ao nível da consola de gestão e a criação de novos módulos de tradução a incluir nos PEPs de forma a estes poderem efectuar a tradução das regras que recebem do PDP para a linguagem específica de um novo tipo de equipamento.

A construção de uma interface gráfica, que permita visualizar apenas o conjunto de regras pretendidas, como por exemplo um só domínio, ou a construção visual e teste de regras é um factor fundamental para o sucesso de um projecto deste tipo

Adicionalmente, é de todo o interesse apresentar soluções normalizadas para a configuração de *firewalls* usando o modelo proposto.

Para concluir, é necessário ter em atenção que esta é uma área em que se têm verificado grandes mudanças, onde os desenvolvimentos surgem constantemente e onde nada, ou quase nada, é tido como definitivo.

Anexo A – Listas de Acesso da Cisco

Uma lista de controle de acesso (ACL) é uma lista de regras ordenadas que permitem ou bloqueiam o tráfego de pacotes baseado-se em informações presentes nessa mesma lista. Sendo que uma ACL é uma lista de regras ordenadas, é importante salientar que a ordem de criação das regras na lista é muito importante. Um dos erros mais comuns durante a criação de listas de acesso é a configuração das regras por ordem incorrecta. No final da lista de acesso, existe uma regra implícita que bloqueia todo o tráfego. Um pacote que não é explicitamente permitido será bloqueado pela regra que bloqueia tudo. Outro erro comum que ocorre no processo de criação de uma lista de controle de acesso é o esquecimento deste facto. [Cisco 2001]

A informação que é geralmente analisada para a filtragem dos pacotes são os dados contidos nos cabeçalhos dos pacotes da camada 3 ou da camada 4. Normalmente, as listas de controle de acesso não podem usar informações de camadas superior a 4 para efectuar a filtragem. Por exemplo: uma lista de controle de acesso não possui a capacidade de filtrar comandos específicos de FTP.

O processo de filtragem de pacotes tem início quando uma interface recebe um pacote. O *router* verifica a tabela de *routing* à procura de uma rota para o pacote. Caso não tenha uma rota, este pacote será descartado sendo enviada para a origem uma mensagem de ICMP (*Host Unreachable*). Caso contrário, verifica-se se existe uma lista de controle de acesso aplicada à interface. Não existindo, o pacote é enviado para o *buffer* da porta de saída. Existindo, o pacote é analisado pela lista de controle de acesso da interface em questão. Uma vez que o fluxo de dados através da uma determinada interface é bidirecional, uma ACL pode ser aplicada a uma direcção específica da interface:

Entrada (*inbound*) - verifica se o processamento do pacote deve continuar após ter sido recebido por uma determinada interface;

Saída (*outbound*) - verifica se o pacote deve ser enviado para uma interface de saída ou bloqueado.

É importante salientar que os pacotes gerados pelo router para a troca de tabelas de *routing* não são afectados pelas regras aplicadas a uma interface no sentido *outbound*. A

única forma de controlar os pacotes gerados pelo router para actualização dessas tabelas é através de ACL de *inbound*.

A lista de acesso é verificada por ordem sequencial, ou seja, o pacote é testado a partir da primeira regra. Assim, se o pacote se enquadrar numa regra, é aplicada a acção correspondente a essa regra: é permitido ou bloqueado. Caso o pacote não se enquadre em nenhuma das regras, o mesmo será bloqueado pela última regra, a qual é implícita e bloqueia tudo o que não está explicitamente permitido, conforme já dito anteriormente. O funcionamento genérico de uma ACL encontra-se descrito na Figura A.1.

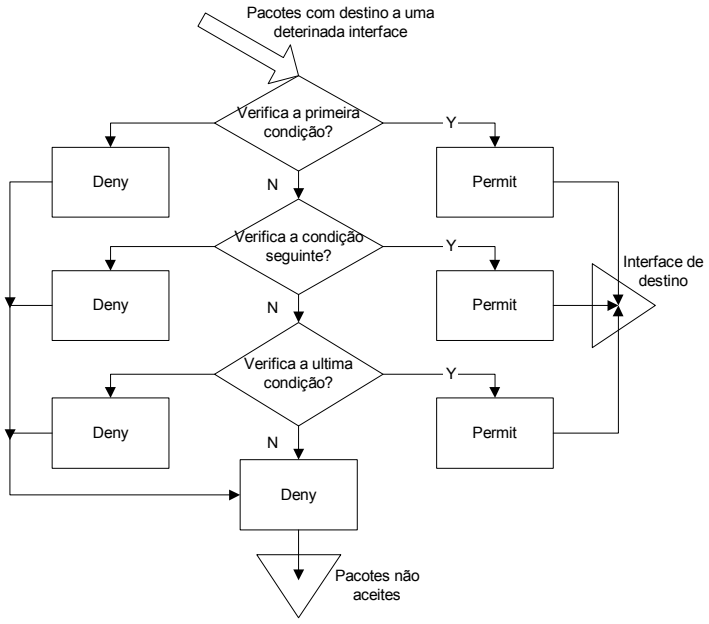

Figura A.1 – Funcionamento genérico de uma ACL

Nos routers *cisco* estão definidos dois principais tipos de listas de acesso:

Padrão (*standard*) - A lista de acesso padrão verifica o IP de origem de um pacote que pode ser alvo de *routing*. Baseado na rede/sub-rede/endereço do *host*, é permitido ou bloqueado o envio do pacote, ou seja, que o mesmo saia por outra interface.

Estendida (*extended*) - A lista de acesso estendida possui um número mais elevado de parâmetros que podem ser alvo de verificação. Neste caso, pode ser analisar o IP de origem, o IP de destino, a porta de origem, a porta de destino, os protocolos e alguns outros parâmetros de forma a permitir ao administrador uma maior flexibilidade na elaboração das regras.

Um aspecto importante na implementação das ACLs é o funcionamento dos *Wildcard* nos *routers cisco*. A filtragem dos endereços é efectuada através da utilização de máscaras *wildcard* para identificar o que é permitido ou bloqueado nos bits do IP. As máscaras *wildcard* para os bits de endereço IP utilizam o número 1 e o número 0 para a identificação do que deve ser filtrado.

O valor 0 significa que o bit deve ser verificado;

O valor 1 significa que o bit deve ser ignorado;

128	64	32	16	8	4	2	1	
0	0	0	0	0	0	0	0	= Verifrica todos os bits
0	0	0	0	0	1	1	1	= Ignora os ultimos 3 bits
1	1	1	1	0	0	0	0	= Verifica os ultimos 4 bits
1	1	1	1	1	1	1	1	= Ignora todos os bits do octeto

Figura A.2 – Exemplo de *Wildcard* em router *Cisco*

É importante salientar que estes bits não têm nenhuma relação com as máscaras de rede IP. Por exemplo, se um administrador pretender testar um conjunto de endereços IP 172.30.16.0 até 172.30.31.0, temos que os dois primeiros octetos correspondem à parte da rede (172.30). O terceiro octeto corresponde à sub-rede (16 até 31), e o quarto corresponde ao *host*. Se pretendermos efectuar a configuração de uma ACL usando máscaras *wildcard* para verificar as sub-redes 172.30.16.0 até 172.30.31.0, o processo será o seguinte:

Inicialmente, a máscara verifica os dois primeiros octetos 172.30. Para isso usa os zeros nos bits do *wildcard*; Como não existe interesse em filtrar a parte do *host*, o quarto octeto será ignorado, para isso os bits devem ser colocados a 1; No terceiro octeto, onde está localizada a sub-rede, a máscara irá verificar a posição correspondente ao valor binário 16, ou seja este bit deve estar com 0 bem como os bits superiores, já os bits abaixo do valor binário 16 devem ser ignorados, para que isso ocorra devem ser colocados a uns. Esta técnica é mostrada nas Figura A.2 e Figura A.3.

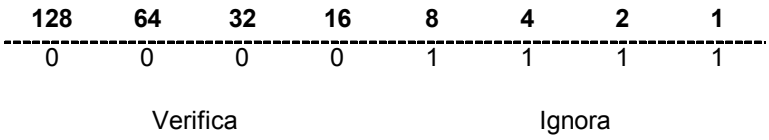

Figura A.3 – Verificação dos bits da máscara de rede

Temos então que para o endereço: 172.30.16.0, com a máscara: 0.0.15.255, serão verificadas as sub-redes: 172.30.16.0 a 172.30.31.0

A utilização de ACLs está bastante difundida para a construção de filtros de pacotes pois pode ser implementada na grande maioria dos *routers* comerciais, não implicando normalmente a compra de equipamento adicional pelas organizações, para implementar este nível de filtragem.

Anexo B – Linux IPChains

O *ipchains* [Russel 2000] é uma ferramenta *Linux* para a filtragem de pacotes utilizada para definir, manter e inspeccionar o *firewall* e as regras existentes no *kernel* do sistema operativo *Linux*. Pode-se dividir esta ferramenta em duas partes. Uma é a componente que permite efectuar a filtragem de pacotes IP ao nível do *kernel* e outra que é o módulo de administração das regras de filtragem.

Neste contexto, usando esta ferramenta, cada um dos pacotes IP é analisado, comparando a sua constituição com um conjunto de regras predefinidas. Com base na análise feita, o *kernel* decide o que fazer com cada pacote, podendo aceitá-lo, rejeitá-lo ou ainda dar-lhe algum tipo de tratamento especifico, como por exemplo redireccionar esse mesmo pacote para outra porta (Figura B.1).

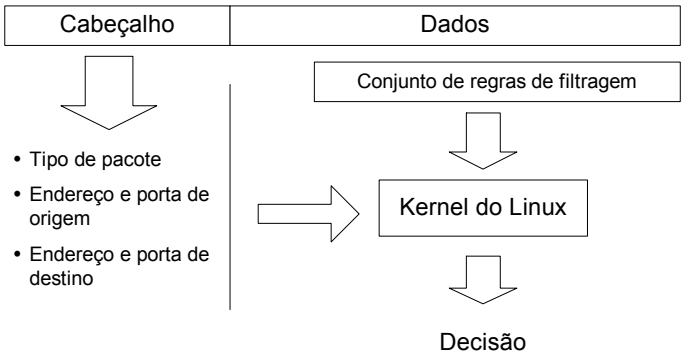

Figura B.1 – *IPChains* - Funcionamento genérico

As regras usadas para efectuar a filtragem dos pacotes são denominadas cadeias (*chains*), sendo que cada cadeia pode conter um sub-conjunto de regras e uma política por omissão. Cada regra usada no *ipchains* é constituída por um par condição-acção. Quando o *kernel* processa um pacote, verifica em cada cadeia, as regras existentes, sequencialmente até que uma delas se verifique. Nesta situação a acção definida na regra é executada. Se forem verificadas todas as regras e nenhuma verificar a condição do pacote em causa, é executada a política por omissão dessa cadeia. Normalmente esta

política por omissão consiste numa regra que obriga o *kernel* a rejeitar o pacote, apenas sendo aceites pacotes que verifique uma das condições descritas nas regras.

O *kernel* inicia a filtragem de pacotes com uma configuração que usa três cadeias predefinidas:

Input – Quando um pacote entra no sistema, depois de submetido aos testes de integridade, é validado pelo *kernel* usando as regras presentes nesta cadeia.

Forward – Se o pacote não foi descartado na cadeia anterior, o *kernel* efectua o seu reencaminhamento. Se o pacote tiver como destino uma outra qualquer máquina, o *kernel* efectua a sua validação usando a cadeia *forward*.

Output – Antes de enviar um pacote de novo para a rede, o *kernel* consulta esta cadeia e efectua a validação desse mesmo pacote.

Usando a ferramenta *ipchains*, o administrador pode criar também novas cadeias, ou apagá-las conforme as necessidades específicas do sistema a proteger. Numa dada regra, podemos incluir condições que testam vários parâmetros do pacote em análise, por exemplo, algumas das principais condições a testar são:

Endereço de origem `(-s ip_address / subnet_mask)`

Esta condição verifica-se quando o pacote provém de uma máquina cuju endereço pertence à gama de endereços definida pelo parâmetro *ip_adress/subnet_mask*.

Endereço de destino `(-d ip_address / subnet_mask)`

Equivalente à condição anterior, sendo testado aqui o endereço de destino.

Protocolo `(-p protocol)`

Verifica se o pacote transporta informação referente a um determinado protocolo. Permite o uso de um valor numérico do protocolo IP ou o nome do protocolo, como por exemplo, TCP, UDP ou ICMP.

Porta TCP ou UDP

Quando for especificado o protocolo TCP ou UDP, pode ser acrescentado o número de uma porta ou um conjunto de portas a serem verificadas.

Interface

Permite especificar qual o interface através do qual o pacote entra ou sai da máquina em que uma regra deve ser verificada.

A cada regra está normalmente associada uma acção que o *kernel* deve executar quando a condição dessa regra se verificar. As acções que se podem especificar usando *ipchains* são:

accept – O pacote é aceite.

reject – O pacote é rejeitado. Nesta situação o emissor é notificado com "*ICMP Host Unreachable*"

deny – O pacote é rejeitado não sendo devolvida nenhuma informação ao emissor do pacote.

masq – O pacote é "mascarado" usando uma outra funcionalidade do *kernel Linux*, o IP *Masquerade*.

redirect – Funcionando na cadeia *Input* com os protocolos TCP e UDP, o pacote é enviado para uma porta local especificada com a acção.

return – Não executa nenhuma acção, concluindo o processamento da cadeia. Esta acção poderá servir para efectuar *logs* de pacotes.

cadeia definida pelo utilizador – Com esta acção, o fluxo de verificação de regras é passado para a primeira regra existente na cadeia definida pelo utilizador que for especificada.

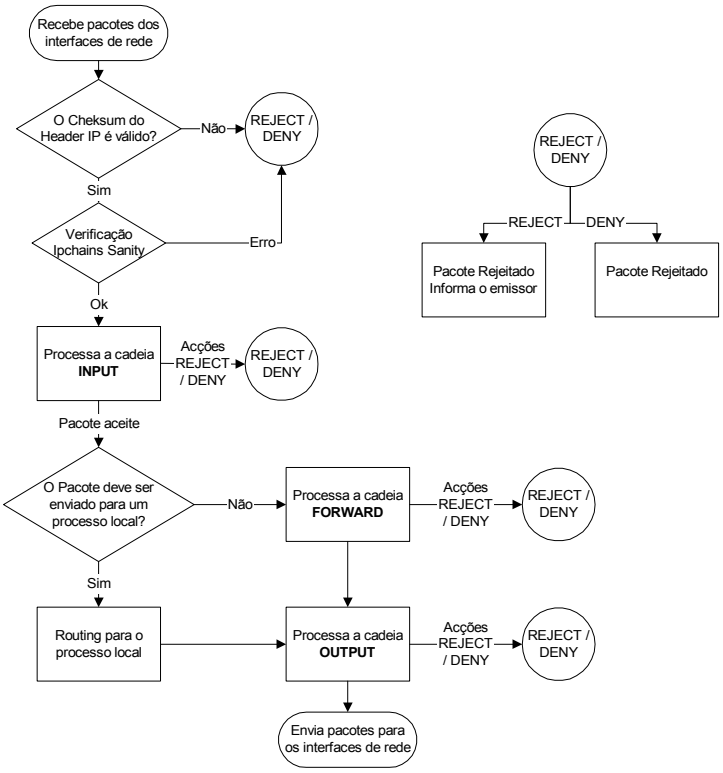

Figura B.2 – Esquema de funcionamento das cadeias IPChains

O *ipchains* permite ainda utilizar algumas opções adicionais, como por exemplo: efectuar o *log* dos pacotes que verificam determinadas regras; especificar que pretendemos verificar os pacotes que contêm a *flag* SYN, restringindo assim a possibilidade de iniciar uma determinada ligação; utilizar a *flag* –y (Bidireccional), que faz com que uma regra passe também a ser aplicada no sentido inverso ao que foi inicialmente definida.

A principal vantagem do uso desta ferramenta é sem duvida o facto de ser uma solução gratuita e *open source* a funcionar sobre o sistema operativo *linux*. Por outro lado, este mesmo motivo também torna o *ipchians* mais conhecido do publico, estando portanto sujeito a mais ataques.

Referências

[3COM 1998-1] *Policy-Powered Networking and the Role of Directories*; 3COM, White paper, Julho 1998.

[Alaetinouglu 1998] Alaetinouglu C., et al., *Routing Policy Specification Language (RPSL),* RFC 2280, Network Working Group, IETF, Janeiro 1998.

[Blaze 2001] Blaze Matt, Ioannidis John, Keromytis Angelos, Trust Management for IPsec, *in Proceedings of the Internet Society Symposium on Network and Distributed Systems Security (SNDSS) 2001*, pp. 139 – 151, Fevereiro 2001.

[Booch 1996] Booch Grady, Rumbaugh James, *Unified Method for Object-Oriented Development Document Set*, Rational Software Corporation, 1996, (http://www.rational.com/uml).

[Boutaba 2001] Boutaba R,, Polyrakis, Andreas, *COPS-PR with meta-policy support*, publicado como submissão independente no IETF, Abril 2001.

[Braden 1997] Braden R., Zhang L., Berson S., Herzog S., Jamin S., *Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification*, RFC 2205, Network Working Group, IETF, Setembro 1997.

[Branchaud 1997] Branchaud Marc, A Survey of Public-Key Infrastructures, *Department of Computer Science McGill University*, Montreal, Março 1997 (http://home.xcert.com/~marcnarc/PKI/thesis/).

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

[Breton 1999] Breton C., *Mastering Network Security*, Network Press, 1999.

[Brownleee 1999] Brownleee N., *SRL: A Language for Describing Traffic Flows and Specifying Actions for Flow Groups*, Internet draft, draft-ietf-rtfm-ruleset-language-07.txt, IETF, Agosto 1999.

[Bumpus 2001] Bumpus, Winston, *CIM, DEN, WBEM Key Technologies to Enable Holistic Management*, DMTF, Fevereiro, 2001.

[Burns 2001] Burns J., Gurung P., Martin D., Rajagopalan S., Rao P., Rosenbluth D., Surendran A., Automatic Management of Network Security Policy by Self-securing Networks, *in Proceedings of DISCEX II*, 2001.

[Caldeira 2000] Caldeira Filipe, Monteiro Edmundo, Descrição, Geração e Difusão de Políticas de Segurança, *in Actas da 3ª Conferência sobre Redes de Computadores, Tecnologias e Aplicações (CRC'2000)*, Novembro 2000.

[Calhoun 2001] Calhoun R., Akhtar H., Arkko J., Guttman E., Rubens A., Zorn G., *DIAMETER Base Protocol*, Internet draft, draft-ietf-aaa-diameter-07.txt, IETF, Julho 2001.

[Chan 2001] Chan, K., Seligson, J., Durham, D., Gai, S., McCloghrie, K., Herzog, S., Reichmeyer, F., Yavatkar, R. and A.Smith, *COPS Usage for Policy Provisioning (COPS-PR)*, RFC 3084, Network Working Group, IETF, Março 2001.

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

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

[Condell 2000-1] Condell M., Lynn C., Zao J., *Security Policy Specification Language*, Internet draft, draft-ietf-ipsp-spsl-00.txt, IETF , Março 2000.

[Condell 2000-2] Condell M., Sanchez L. A., *Security Policy Protocol*, Internet draft, draft-ietf-ipsp-spp-00.txt, IETF, Julho 2000.

[Damianou 2000] Damianou Nicodemos, Policy Specification and Implementation for Distributed Systems Management, *Imperial College of Science, Technology and Medicine*, Janeiro 2000.

[Damianou 2001] Damianou N., Dulay, N., Lupu, E., Sloman, M., Ponder: A Language for Specifying Security and Management Policies for Distributed Systems – The Language Specification, Version 1.11, *Imperial College Research Report DoC 2000/1*, Janeiro 2001, (http://www-dse.doc.ic.ac.uk/files/Ponder1.11.pdf).

[Dierks 1999] Dierks T., Allen C., *The TLS Protocol*, RFC 2246, Network Working Group, IETF, Janeiro 1999.

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

[Donnelly 1995] Donnelly C., Stallman R., *Bison - The YACC-compatible Parser Generator*, (http://www.gnu.org/manual/bison/html_mono/bison.html), Novembro 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, Janeiro 2000.

[Fine 1999] Fine, M., McCloghrie, K., Seligson, J., Chan, K., Hahn, S., A. Smith, *Quality of Service Policy Information Base*, Internet draft, draft-mfine-cops-pib-01.txt, IETF, Junho 1999.

[Fine 2000] Fine M., McCloghrie K., Seligson J., Chan K.; Hahn S., Sahita R., Smith A., Reichmeyer F.; *Framework Policy Information Base*, Internet draft, draft-ietf-rap-frameworkpib-04.txt, IETF, Novembro 2000.

[Follows 1999] Follows Jonathan, Straeten Detlef, Application-Driven Networking: Concepts and Architecture for Policy-Based Systems, *IBM - International Technical Support Organization*, Dezembro 1999, (http://www.redbooks.ibm.com).

[Fraser 1997] Fraser B., *The Site Security Handbook*, RFC 2196, Network Working Group, IETF, Setembro 1997.

[Hegering 1999] Hegering, Heinz-Gerd, Abeck, Sebastian and Neumair, Bernhard; *Integrated Management of Networked Systems*; Morgan Kaufman, 1999.

[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/).

[Ioannidis 2000] Ioannidis Sotiris, Keromytis Angelos , Bellovin Steven, Smith Jonathan , Implementing a Distributed Firewall, *in Proceedings of the ACM Computer and Communications Security (CCS) 2000*, Novembro 2000.

[IPHighway 2000] *Introduction to Policy-based Networking and Quality of Service*, White paper, IPHighway, Janeiro 2000.

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

[ISO 7498] ISO/IEC 7498-4, 1989.

[ITS 1996] Letter-by-Letter Listing, (www.its.bldrdoc.gov/fs-1037/dir-001/_0064.htm), 1996.

[Jennings 2001] Jennings, James, *Common Information Model: A Candidate Modeling Technology*, Tivoli Core Technologies, Junho, 2001.

[Keromytis 1999] Keromytis A., Richardson M., Sanchez L., *IPsec Policy - Discovery Architecture,* Internet draft, draft-keromytis-ipsp-arch-00.txt, IETF, Outubro 1999.

[Krawczyk 1997] Krawczyk, H., Bellare, M. and R. Canetti, *HMAC: Keyed-Hashing for Message Authentication*, RFC 2104, Network Working Group, IETF, Fevereiro 1997.

[Mahon 1999] Mahon H., Bernet Y., Herzog S., *Requirements for a Policy Management System*, Internet draft, draft-ietf-policy-req-02.txt, IETF, Novembro, 1999.

[Meyer 1999] Meyer D., Schmitz J., Orange C., Prior M., and Alaettinoglu C., *Using RPSL in Practice*, RFC 2650, Network Working Group, IETF, Agosto 1999.

[Monteiro 2000] Monteiro, Edmundo, Boavida, Fernando, *Engenharia de Redes Informáticas*, FCA – Editora, Agosto 2000.

[Moore 2000] Moore B., Ellesson E., Strassner J., *Policy Core Information Model – Version 1 Specification*, Internet draft, draft-ietf-policy-core-info-model-04.txt, IETF Março 2000.

[MS 2000] Microsoft Support Online, A Brief Introduction to ASN.1 and BER, *Doc. n.º Q252648*, Outubro 2000.

[Oliveira 2000] Oliveira, J., Aguiar R., Network and Desktop Management Convergence, *in Proceedings of the 33rd Hawaii International Conference on System Sciences*, Hawaii, Janeiro 2000.

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

[Raju 1999] Raju Rajan, Dinesh Verma, Sanjay Kamat, Eyal Felstaine, Shai Herzog, A policy framework for integrated and differentiated services in the internet, *in IEEE Network*, Setembro/Outubro 1999.

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

[Russel 2000] Russell Rusty, *Linux IPChains HowTo*, Online, Julho 2000.

[Sanchez 1999] Sanchez Luis, McCloghrie Keith, Jon Saperia, *Evaluation of COPS/PIB and SNMP/MIB approaches for configuration management of IP-based networks*, Internet draft, draft-ops-mumble-conf-management-00.txt, IETF, Outubro 1999.

[Shepard 2000] Shepard, Susan; Policy-based networks: hype and hope; *in IT Professional, Vol. 2, No. 1*, Janeiro-Fevereiro 2000.

[Stevens 1999] Stevens M., Weiss W., Mahon H., Moore B., Strassner J., Waters G., Westerinen A., Wheeler J., *Policy Framework*, Internet draft, draft-ietf-policy-framework-00.txt, IETF, Setembro 1999.

[Stone 2001] Stone G., Lundy B., Xie G., Network Policy Languages: A Survey and a New Approach, *in IEEE Network, pag. 10-21*, Janeiro/Fevereiro 2001.

[Strassner 1998] Strassner J. and Schleimer S., *Policy Framework Definition Language*, Internet draft, draft-ietf-policy-framework-pfdl-00.txt, IETF, Novembro 1998.

[Tosic 1999] Tosic, V., Djordjevic-Kajan S., The Common Information Model (CIM) Standard – An Analysis of Features and Open Issues, *in Proceedings of the TELSIKS '99 – 4th International Conference on Telecommunications in Modern Satellite, Cable, and Broadcasting Services, Nis, Yugoslavia*, Outubro 1999.

[Yeong 1995] Yeong W., Howes T., Kille S., *Lightweight Directory Access Protocol*, RFC 1777, Network Working Group, IETF, Março 1995.

Lista de acrónimos

Neste documento são utilizados vários acrónimos, enquanto abreviaturas de designações comuns. Os acrónimos mais utilizados são:

ACL	Access Control List
API	Application Program Interface
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
BNF	Backus-Naur Form
CIM	Common Information Model
COPS	Common Object Policy Service
COPS-PR	COPS for Policy Provisioning
DEN	Directory Enable Networking
DMTF	Distributed Management Task Force
DNS	Domain Name Service
DTD	Document Type Definition
FCAPS	Fault, Configuration, Accounting, Performance, Security
GUI	Graphics User Interface
HMAC	Hashed Message Authentication Code
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IP	Internet Protocol
IPSEC	IP Security Protocol
IRR	Internet Routing Registry
ISO	International Standard Organization
ISP	Internet Service Provider
LDAP	Lightweight Directory Access Protocol
LPDP	Local Policy Decision Point
MIB	Management Information Base
PBN	Policy-Based Networking
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PFDL	Policy Framework Definition Language

PIB	Policy Information Base
PKI	Public-Key Infrastructures
PONDER	Ponder Policy Specification Language
PRC	Provisioning Class
PRI	Provisioning Instance
PRID	Provisioning Instance Identifier
QoS	Quality of Service
RSVP	Resource ReSerVation Protocol
RTFM	Real-time Traffic Flow Measurement
SGBD	Sistema de Gestão de Bases de Dados
SNMP	Simple Network Management Protocol
SPSL	Security Policy Specification Language
SQL	Structured Query Language
SRL	Simple Ruleset Language
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UML	Unified Modelling Language
VPN	Virtual Private Network
WWW	World Wide Web
XML	eXtensible Markup Language