ANDRÉ ROBLES ORTEGA

PFIM: Um Modelo de Informação para Gerenciamento de Filtros de Pacotes Baseado em Políticas

CURITIBA

2003

ANDRÉ ROBLES ORTEGA

PFIM: Um Modelo de Informação para Gerenciamento de Filtros de Pacotes Baseado em Políticas

Dissertação apresentada ao Programa de Pós-Graduação em Informática Aplicada da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática Aplicada.

Área de Concentração: *Metodologia e Técnicas de Computação*

CURITIBA

2003

Ortega, André Robles

PFIM: Um Modelo de Informação para Gerenciamento de Filtros de Pacotes Baseado em Políticas. Curitiba, 2003. 139 p.

Dissertação (Mestrado) – Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática Aplicada.

1. Segurança 2. Política 3. Gerenciamento 4. PCIMe.

I.Pontifícia Universidade Católica do Paraná. Centro de Ciências Exatas e de Tecnologia. Programa de Pós-Graduação em Informática Aplicada II-t

Aos meus pais, Silvio e Anadir, pelo incentivo
e apoio dados à execução deste trabalho.

Agradecimentos

Ao Professor Dr. Edgard Jamhour, pela sugestão do tema e brilhante orientação.

À minha esposa, Adriana, pela compreensão nos momentos dedicados a este trabalho.

À Xerox Comércio e Indústria Ltda, pelo incentivo ao desenvolvimento humano.

À PUCPR, pela oportunidade oferecida.

SUMÁRIO

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

CIM	Common Information Model
COPS	Common Open Policy Service
DEN	Directory Enabled Networks
DIT	Directory Information Tree
DMTF	Distributed Management Task Force
DN	Distinguished Name
IANA	Internet Assigned Numbers Authority
IETF	Internet Engineering Task Force
LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Data Interchange Format
LPDP	Local Policy Decision Point
MDL	Model Definition Language
MIB	Management Information Base
PCIM	Policy Core Information Model
PCIMe	Policy Core Information Model Extensions
PCLS	Policy Core LDAP Schema
PCELS	Policy Core Extensions LDAP Schema
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIB	Policy Information Base
QoS	Quality of Service
RBAC	Role Based Access Control
RDN	Relative Distinguished Name
RFC	Request for Comments
SNMP	Simple Network Management Protocol
TLS	Transport Layer Security

Resumo

O considerável crescimento das redes de comunicação revelou uma significante limitação na eficiência e escalabilidade nas técnicas tradicionais de gerenciamento de redes. O gerenciamento de redes baseado em políticas surge como um paradigma promissor na área de configuração e fornecimento de serviços. Neste sentido, o IETF, em conjunto com o DMTF propõem o PCIMe – *Policy Core Information Model Extensions* – que é uma padronização do formato de representação e gerência da estrutura das informações das políticas. Entretanto, o PCIMe não foi concebido para suportar apenas áreas de aplicação específicas. Políticas de segurança que são relacionadas às regras de negócio exigem a representação de informação e procedimentos não suportados diretamente pelo PCIMe. De acordo com o IETF, tais áreas de aplicações devem ser suportadas pela extensão do modelo PCIMe através da especialização de suas classes. Assim, este trabalho pretende definir um modelo de armazenamento e distribuição das regras associadas às políticas de segurança entre os diversos dispositivos de roteamento e segurança (*gateways* e *firewalls*) dispostos na rede com eficiência e escalabilidade estendendo o modelo proposto pelo PCIMe mantendo a conformidade com o padrão existente. A estrutura do PCIMe sugere a utilização de serviços de diretório como o repositório natural das informações de política. Neste caso, será utilizado o serviço LDAP através da implementação de um esquema derivado do PCIMe. Também é definida a arquitetura de distribuição das regras entre os pontos de decisão (PDP) e os pontos de aplicação das políticas (PEP) seguindo o padrão de comunicação definido pelo protocolo COPS – *Common Open Policy Services*.

Palavras-Chave: 1. Segurança 2. Política 3. Gerenciamento 4. PCIMe

Abstract

The considerable growth of the networks disclosed to a significant limitation in the efficiency and scalability in the traditional network management techniques. With the advent of Policy Based Network Management (PBNM) organizations can bind policies to the allocation of network resources. PBNM describes and implements high- level policies to the entire network rather than managing each device individually. The IETF in accordance with DMTF proposed the PCIMe – Policy Core Information Model Extensions – which is a standardization of the format of representation and management of the policy information structure. However, PCIMe was not conceived to support only specific areas. Security policies related to the business rules may represent information and procedures not supported by PCIMe. Such areas must extend the PCIMe framework through the specialization of its classes. This work intends to define a storage and distribution model of rules concerning to security policies. These policies will be used to automate the configuration of each device of the network. The PCIMe structure suggests the use of directory services as the natural repository for policy information. In this case, this work will demonstrate how to extend the PCIMe framework to map information about packet filters and store this information in an LDAP server. This work defines also an architecture of distribution of the policy rules between the policy decision points (PDP) and the policy enforcement points (PEP) following the Common Open Policy Services protocol – COPS.

Keywords: 1. Security 2. Policy 3. Management 4. PCIMe

Capítulo 1

Introdução

1.1. Desafio

Cada vez mais as pessoas e os negócios dependem de sistemas de computação interconectados que suportem aplicações distribuídas. Essas aplicações interagem entre si através de computadores dentro da mesma LAN, dentro de uma Intranet corporativa ou em qualquer outro lugar do mundo através da Internet. Para garantir a funcionalidade, oferecer facilidade de uso e uma administração economicamente eficiente dessas aplicações, as informações sobre os serviços, recursos, usuários e quaisquer outros objetos disponíveis na rede precisam estar organizados de maneira estruturada e de uma forma consistente. Atualmente, os serviços de diretório são capazes de armazenar informação a respeito de milhões de objetos dentro de um ambiente de rede. Muito dessas informações pode estar compartilhado entre as próprias aplicações, mas há a necessidade de proteção para que se evite acessos não autorizados aos dados.

Nos últimos anos, as técnicas de *firewalls* baseados em filtragem de pacotes receberam muitas novas funcionalidades, tais como "*statefull inspection*", transparência e desempenho entre outras, que representam um incremento tecnológico considerável. Entretanto, enquanto os *firewalls* receberam tais avanços, as técnicas de configuração e gerenciamento para toda essa tecnologia de segurança de informações receberam pouca atenção. Os *firewalls* não são simplesmente um software ou hardware que se tira da caixa, instala e está pronto para proteger uma rede. O administrador de segurança precisa configurar e gerenciar o *firewall* segundo uma política de segurança que atenda às necessidades da corporação a que pertence.

O problema se torna ainda maior quando uma rede possui mais que um único *firewall*. Esses *firewalls* dividem a Intranet da corporação em zonas, tais como recursos humanos, finanças, vendas, engenharia entre outras. Neste caso, a política de segurança é vista como um enorme conjunto de regras, dispostas entre diversos *gateways* que interconectam as diversas zonas. Assim, a maneira como essas regras deverão interagir entre si deve ser cuidadosamente examinada para que o sistema de segurança se comporte da forma esperada.

1.2. Motivação

Toda empresa possui regras para acesso aos seus recursos, sejam eles computacionais ou não. No caso de segurança de informações, essas regras podem ser reunidas em um conjunto denominado, genericamente, de "políticas de segurança". São as políticas de segurança que traduzem como as regras de negócio devem ser aplicadas na alocação dos recursos da rede. A esse tipo de gerenciamento de redes dá-se o nome de rede baseada em políticas. Assim, as políticas descrevem o comportamento da rede quando receber uma requisição de usuário ou quando os serviços de rede interagem entre si.

As políticas podem ser definidas a partir de duas perspectivas [RFC3198]:

1. Um objetivo, direcionamento ou método de ação utilizado para tomada de decisões presentes e futuras executadas dentro de um contexto particular.

2. Um conjunto de regras para administrar, gerenciar e controlar o acesso aos recursos disponíveis na rede.

Note-se que essas duas abordagens não são contraditórias, mas que se complementam.

A partir da segunda definição, entende-se que uma política de segurança é uma abstração das regras que devem ser implementadas em um *firewall* a fim de se obter o grau de segurança desejado. Entretanto, apesar do enorme crescimento tecnológico que os *firewalls* têm vivenciado nos últimos anos, a área de gerenciamento foi deixada pra trás.

De fato, uma pesquisa da revista Fortune relata que, para a maioria das grandes empresas norte-americanas, a maior dificuldade encontrada na área de segurança na Internet reside na falta de ferramentas de gerenciamento de segurança [BAR96].

Nos *firewalls*, o gerenciamento das políticas é uma tarefa bastante complexa uma vez que esses dispositivos são baseados em configuração de regras. No caso de um *firewall*

protegendo uma única Intranet – por exemplo, a LAN de uma pequena empresa – um conjunto de regras instrui o *firewall* sobre que pacotes podem ou não passar para dentro da rede. De maneira similar, outro conjunto define quais serviços têm permissão de acessar o mundo externo.

O administrador precisa implementar as políticas de segurança de alto nível sob a forma de regras de baixo-nível para que o *firewall* possa entender [BAR99]. A interface de configuração permite que o administrador defina grupos de hosts (faixas de endereços IP) e grupos de serviços, mas quando se examina um *firewall* real, percebe-se que esta estratégia freqüentemente conduz a configurações incorretas, com a inserção de regras redundantes e outras que estão faltando. Isto leva à necessidade de se reorganizar as regras novamente até que se alcance a implementação da política de segurança desejada. Esforços tais como *Distributed Firewalls* [BEL99] esbarram em limitações dos sistemas operacionais ou pela construção de complicadas APIs que implementam uma interface entre o software e o sistema operacional. Outras soluções comerciais sofrem do mal de serem produtos proprietários e não serem padronizados [BAR99].

O DMTF - *Distributed Management Task Force* –, fundado em 1992, tem se dedicado bastante ao desenvolvimento de padrões de gerenciamento para ambientes distribuídos. Entre esses padrões se destaca o CIM – *Commom Information Model* –, que é um modelo conceitual utilizado para a descrição da informação de gerenciamento, objetivando o estabelecimento de um padrão que possibilite o compartilhamento desta informação entre sistemas e aplicações. Por exemplo, o CIM fornece modelos que permitem a representação de serviços baseados em rede, dispositivos, sistemas, redes, entre outros.

Outra iniciativa, proposta em 1997, foi o DEN – *Directory Enabled Networks.* O DEN especifica um modelo de dados que mapeia os objetos CIM em serviço de diretório LDAP – *Lightweight Directory Access Protocol* [RFC2252]. O objetivo é habilitar a implementação e o uso de políticas através de um serviço comum e conceitos de usuários, especificando seu mapeamento e armazenamento em repositório LDAP para tornar regras de políticas independentes de fabricantes [RFC3198]. A partir de 1998, o DEN foi incorporado ao CIM. Ter um diretório como elemento central capacita a integração da informação de gerenciamento de uma grande variedade de fontes, fornecendo as bases para o gerenciamento baseado em políticas. Em 2001, a partir das padronizações do CIM, o IETF e o DMTF

propuseram um modelo para descrever o paradigma de políticas: O PCIM – *Policy Core Information Model* – [RFC3060]. O PCIM fornece um modelo para representar, gerenciar, aplicar e compartilhar a informação de política. As classes que compõem o PCIM têm como objetivo servir como uma hierarquia extensível (através de especialização) para a definição de objetos de política de diversos tipos [RFC3060]. O impacto que o PCIM causou na comunidade de gerenciamento de redes e QoS – *Quality of Service* – principalmente, foi excelente. É possível encontrar diversas implementações de sistemas de gerenciamento baseados em políticas. [NAB03] propõem uma extensão do modelo proposto pelo PCIM para implementar um sistema de controle de acesso RBAC. Em 2003, o PCIM sofreu algumas alterações que refinaram o modelo. Esta nova padronização recebeu o nome de PCIMe – *Policy Core Information Model Extensions* – [RFC3460].

[REY03] propõe um esquema de diretório para o mapeamento do PCIMe em LDAP. Esta proposta tende a padronizar os esquemas de armazenamento de objetos de políticas nesse tipo de serviço de diretório. Além disso, o LDAP é um serviço de diretório aberto que já está implementado em muitos sistemas operacionais.

Para a disposição das políticas entre os diversos dispositivos na rede, distinguem-se duas entidades básicas: o PDP – *Policy Decision Point* e o PEP – *Policy Enforcement Point*. O PEP tipicamente reside no dispositivo gerenciado. Seu papel é implementar os comandos recebidos do PDP na forma de dados de configuração. O PDP é o responsável pelas tomadas de decisão. Seu papel é atender as requisições dos PEPs e processar as políticas junto com outros dados tais como o estado da rede e gerar os dados de configuração que serão enviados para os PEPs.

O IETF padroniza a comunicação entre o PDPs e PEPs utilizando o protocolo COPS – *Common Open Policy Service* – [RFC2748] e suas extensões. Essas extensões definem detalhes para o formato e a semântica sobre como as mensagens e os dados de configuração serão intercambiados entre os PDPs e os PEPs. Nesse contexto, o COPS apresenta duas estratégias: o modelo *outsourcing* e o modelo *provisioning*. No modelo *outsourcing*, a cada nova requisição que chega ao PEP, é feita uma consulta ao PDP. No modelo *provisioning* o PEP se conecta ao PDP, reporta suas capacidades e limitações e solicita as políticas a ele associadas. Caso haja uma mudança no estado da rede ou uma alteração na política, o PDP é o responsável por enviar as novas configurações ao PEP [POL02]. Um esforço no sentido de se padronizar o modelo *provisioning* é o COPS-PR [RFC3084].

Todas as padronizações contidas nos trabalhos descritos acima se apresentam como os elementos necessários para a construção de um sistema de gerenciamento de *firewalls* capaz de ser simples na sua operação, independente de topologia e que minimize os erros de configuração através da aplicação de uma política de segurança, contrapondo um paradigma de configuração de regras. Será necessário estender o modelo proposto pelo PCIMe através de classes que acomodem os novos objetos propostos por este trabalho. O RBPIM [NAB03] já demonstrou como o PCIM pode ser estendido para suportar o modelo RBAC.

1.3. Proposta

O objetivo deste trabalho é propor um modelo orientado a objetos de configuração e gerenciamento das políticas de segurança que serão aplicadas, em termos de regras, em *firewalls* dispostos em ambientes distribuídos. Para tanto, pretende-se, primeiramente, estender o modelo de classes proposto pelo CIM [DMT00] para que seja possível a representação de um *firewall.* Além disso, para que o modelo suporte as regras de *firewall*, que implementam as políticas de segurança, será necessário estender o modelo de classes sugerido pelo PCIMe [RFC3460].

A implementação desse modelo em serviços de diretório baseado em LDAP será possível através da adaptação do esquema PCIMe ao modelo de objetos utilizado pelo serviço LDAP. O *Policy Core LDAP Extension Schema* – PCELS [REY03] apresenta um esquema para mapeamento entre as classes do PCIMe e um diretório LDAP. Entretanto, será necessário mapear, também, as novas classes propostas por este trabalho.

Ainda no contexto do PCIM, este trabalho pretende definir o PDP, o PEP e propor uma arquitetura que implemente a transferência das regras entre o repositório LDAP e o PDP e entre o PDP e os PEPs através do protocolo COPS-PR [RFC3084].

Desta forma, através da definição dos algoritmos de busca dos objetos de políticas contidos no LDAP será possível encaminhar cada uma das regras que compõem uma política de segurança aos dispositivos de rede (*firewalls*) aos quais estejam atribuídas.

As regras deverão ser submetidas a um interpretador que estará convertendo os dados armazenados no diretório em uma MIB que poderia ser utilizada para a configuração de um *firewall*. Assim, a partir dos padrões definidos pelo IETF, qualquer organização poderia

utilizar o modelo proposto como uma ferramenta de gerenciamento de suas políticas de segurança, independente dos modelos de *firewall* que se tenha disponível.

1.4. Organização

O desenvolvimento do trabalho apresentado nesta dissertação está estruturado em sete capítulos, organizados da seguinte forma:

- Capítulo 1 – Introdução: Faz uma apresentação geral do contexto deste trabalho.
- Capítulo 2 – Modelos de Configuração de *Firewalls*: Apresenta o estado da arte no que se refere às técnicas e tecnologias de *firewalls* baseados em filtros de pacotes, demonstrando a necessidade de um gerenciamento das regras centralizado.
- Capítulo 3 – Padrões IETF: Descreve os principais conceitos relacionados ao gerenciamento de redes baseado em políticas, arquitetura de implementação e os métodos de mapeamento, armazenamento e recuperação das políticas em um ambiente distribuído.
- Capítulo 4 – PFIM: Uma Proposta de Extensão do PCIMe para Configuração de Filtros de Pacotes: Apresenta o modelo proposto por este trabalho para o mapeamento e armazenamento das regras de política relacionadas a filtros de pacotes.
- Capítulo 5 – Estudo de Caso: Realiza um estudo de caso envolvendo o controle seletivo de pacotes baseado no modelo do PFIM em uma empresa de tecnologia onde um roteador deve ser configurado para permitir ou negar a passagem de solicitações entre a rede a Internet e entre as sub-redes do exemplo. Também apresenta as consultas LDAP que tornam possível a recuperação das condições e ações associadas às regras a partir do diretório.
- Capítulo 6 – Conclusões e Trabalhos Futuros: Descreve as conclusões obtidas a partir desta proposta e apresenta as possíveis evoluções que este trabalho pode receber.

Capítulo 2

Modelos de Configuração de Firewalls

2.1. Introdução

Este capítulo aborda o problema do gerenciamento do controle de acesso de redes de grande porte, formadas por um conjunto de redes administradas independentemente, interligadas através de uma infra-estrutura de comunicação compartilhada, como a Internet.

A conexão de uma rede com a Internet cria um problema: como evitar que recursos computacionais da rede interna sejam indevidamente expostos a usuários externos conectados à Internet?

Uma das técnicas adotadas para responder a essa pergunta consiste em estabelecer um controle seletivo de acesso entre a rede interna e a Internet. Uma parte do controle seletivo é obtida através de técnicas bem conhecidas de *firewall.* A função do *firewall* neste contexto é a de ser um elemento restritivo.

Muitos dos serviços oferecidos pela Internet são inerentemente inseguros. O *firewall* policia o tráfego de mensagens para esses serviços. É o *firewall* quem implementa as políticas de segurança, permitindo a passagem apenas dos pacotes os quais possuem uma regra específica definida [ZWI00].

2.2. Filtros de Pacotes

Uma das técnicas mais utilizadas em implementações de *firewalls* é a filtragem de pacotes. Os filtros de pacotes controlam seletivamente o fluxo de dados de/para a rede interna.

A filtragem é executada, normalmente, quando um pacote está sendo roteado de uma rede para outra [ZWI00]. Ela é feita com base nas informações introduzidas no cabeçalho dos protocolos da arquitetura TCP/IP [COM95a], [COM95b].

Os filtros de pacotes trabalham com as estruturas de dados correspondentes à camada de rede (denominados pacotes ou datagramas). Os pacotes contêm, basicamente, três tipos de informação: o cabeçalho de rede e, em alguns casos, o cabeçalho de transporte e aplicação. Os filtros de pacotes trabalham com as informações das camadas de rede e transporte. A camada de aplicação é utilizada pelos *gateways* de aplicação (servidores proxy).

As informações principais utilizadas para efetuar a filtragem são [ZWI00]:

- Endereço IP de origem
- Endereço IP de destino
- Protocolo (se o pacote é do tipo TCP, UDP ou ICMP)
- Porta TCP ou UDP de origem
- Porta TCP ou UDP de destino
- Tipo da mensagem ICMP
- Tamanho do pacote

Os filtros de pacotes são implementados no ponto de conexão entre a rede interna e a rede externa. Esse ponto é geralmente um roteador, mas pode-se utilizar também um computador com duas placas de rede. Os roteadores que executam a função de filtragem de pacotes são denominados *screening routers* [ZWI00]. A maioria dos roteadores comerciais modernos traz embutida a função de filtragem de pacotes. Existem também muitos produtos comerciais que permitem utilizar um computador com duas placas de rede como filtro de pacotes. Estes computadores são denominados *bastion hosts.*

A configuração do filtro de pacotes é feita através de um conjunto de regras de filtragem que define a política de segurança da empresa em relação à conectividade com o mundo externo. O formato das regras de filtragem é do tipo:

SE condição *ENTÃO* ação

A ação pode ser do tipo ACEITAR ou REJEITAR (o pacote). A condição determina um conjunto de condições que devem ser validadas para que a ação seja implementada. As condições são estabelecidas em função das informações encontradas no cabeçalho dos pacotes. As principais informações são listadas a seguir:

- No cabeçalho do protocolo IP analisa-se os endereços IP de origem e destino. Analisa-se também o tipo de protocolo, que é um campo no cabeçalho IP que identifica o tipo de informação transportada pelo pacote. Por exemplo: TCP (0x06), UDP (0x11) ou ICMP (0x01).
- No cabeçalho do protocolo de transporte analisam-se os endereços das portas de origem ou destino, no caso dos protocolos TCP e UDP. Para o protocolo TCP, utiliza-se também a informação do bit ACK, que indica o estabelecimento de uma conexão TCP (ACK = 0 indica o pedido de abertura de conexão) [COM95b]. No caso do ICMP analisa-se o tipo de mensagem.
- Também é possível utilizar padrões de bits no corpo ou cabeçalho do pacote. Esse método genérico permite associar regras a seqüências especiais de bits que apareçam em qualquer parte do pacote.

Outra informação relevante é o sentido do pacote, isto é, se ele está entrando ou saindo da rede interna. Para se alcançar um alto grau de segurança, as regras de entrada dos pacotes são definidas de maneira independente das regras para os pacotes que saem da rede. Além disso, alguns roteadores permitem definir regras isoladamente para cada porta.

Observe que a combinação de portas e endereços IP de origem e destino permite implementar as políticas de segurança de maneira bastante flexível. Por exemplo, a Tabela 2.1 exprime o conjunto de regras que permite que usuários da rede interna acessem recursos externos usando o protocolo de aplicação HTTP, mas proíbe que qualquer computador externo inicie uma conexão com um computador interno.

Tabela 2.1: Exemplo de regras de filtragem

REGRA	INTERFACE/ SENTIDO	PROTOCOLO	IP ORIGEM	IP DESTINO	PORTA ORIGEM	PORTA DESTINO	FLAG ACK	AÇÃO
1	rede interna/ para fora	TCP	interno	externo	> 1024	80	*[1]	aceitar
2	rede externa/ para dentro	TCP	externo	interno	80	> 1023	1	aceitar
3	*	*	*	*	*	*	*	rejeitar

[1] O símbolo "*" indica que qualquer valor é aceitável para regra.

As regras do filtro de pacotes são avaliadas em seqüência, sendo que a primeira regra que for satisfeita, independentemente de haver outras regras na seqüência, define a ação sobre o pacote [JAM99]. Alguns dispositivos mantêm registros dos pacotes que estão sendo roteados. Estes dispositivos são denominados *statefull packet filters*[2] (são chamados assim, pois esses dispositivos mantêm informações a respeito do estado das transações). Esses dispositivos são muito úteis na inspeção de pacotes UDP. Pacotes UDP têm estrutura similar à estrutura de pacotes TCP. Um cabeçalho UDP contém os números de porta UDP de origem e destino assim como em um cabeçalho TCP. Entretanto, um cabeçalho UDP não contém qualquer informação sobre números de seqüência ou *flags* como os que são utilizados pelo TCP. Em particular, um cabeçalho UDP não possui o bit ACK, o que torna impossível para o dispositivo determinar se o pacote é um pedido de conexão de um cliente externo para um cliente interno ou se é uma resposta de um servidor externo para um cliente interno. Assim, a manutenção dos estados das transações permite que sejam construídas regras tais como:

"Permita a entrada de pacotes UDP se, e somente se, forem respostas a pacotes UDP que já tenham saído" [ZWI00].

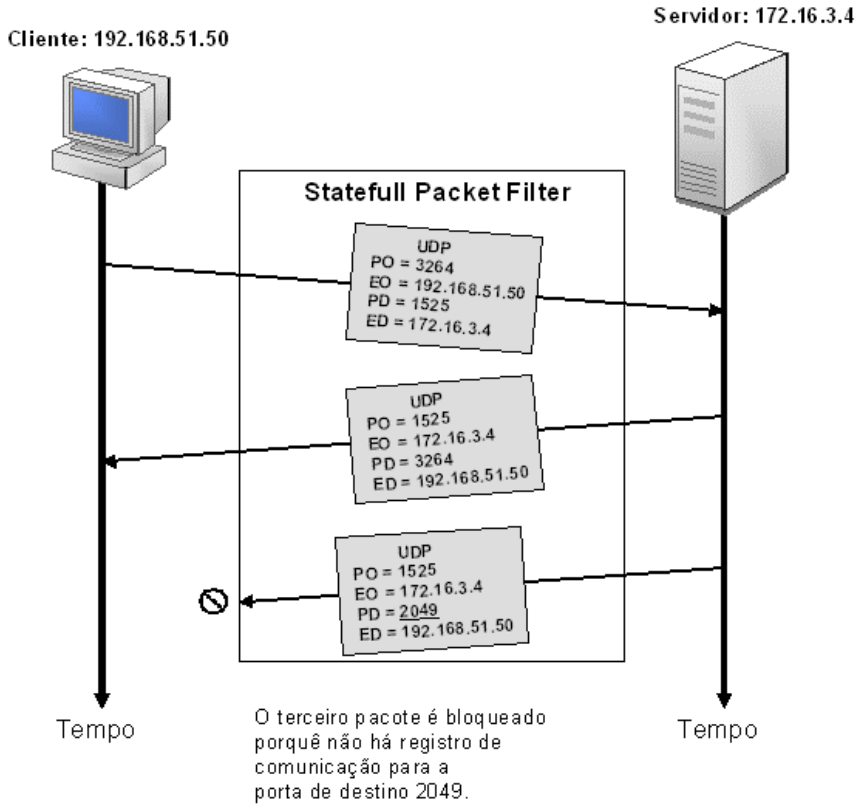

Figura 2.1 – *Statefull Packet Filter* - [ZWI00]

[2] Também podem ser chamados de *Dynamic Packet Filters* ou *Intelligent Packet Filters*.

Na Figura 2.1, os dois primeiros pacotes podem passar, pois são referentes a uma mesma comunicação a qual o dispositivo identifica inspecionando o valor das portas e endereços de origem e de destino. O terceiro pacote é endereçado a uma porta que não faz parte do contexto da comunicação entre o cliente e o servidor, sendo, portanto, bloqueado.

2.3. O Firewall SonicWall

A SonicWALL, empresa fundada em 1991, é especializada no projeto, desenvolvimento e manufatura de soluções relativas à segurança na Internet. Os produtos da SonicWALL prometem fornecer proteção para redes de vários tamanhos, incluindo SOHO, SMB, VPN Corporativa, provedores de acesso, governo, educação e saúde. A partir de 2002, a SonicWALL vem sendo citada em diversas publicações pela facilidade de uso e alto desempenho de seus produtos.

Recentemente, a SonicWALL vem desenvolvendo parcerias com empresas tais como Cisco, 3COM, NetGear, McAfee, Verisign, Cerberian e Websense. Os conhecimentos obtidos a partir dessas parcerias permitiram que a SonicWALL também desenvolvesse ferramentas que oferecem aos administradores da rede a capacidade de gerenciar as políticas de segurança e regras dos *firewalls* de forma centralizada.

Os *firewalls* da SonicWALL são preparados para prover as seguintes funcionalidades:

- filtragem de pacotes;
- servidor Proxy;
- NAT – *Network Address Translation*;
- reconhecimento de ataques DoS e DDoS;
- mantém o estado das sessões abertas – *Statefull Inspection*;
- inspeção de vírus, *worms* e cavalos de tróia anexados aos e-mails;
- filtragem de conteúdos.

Os *firewalls* da SonicWall possuem uma configuração *default*:

- permitir todas as sessões originadas na LAN com destino na WAN;
- bloquear todas as sessões originadas na WAN com destino na LAN.

Novas regras podem ser definidas para estender ou sobrepor as regras *default*, como, por exemplo:

- bloquear todo o tráfego relativo ao protocolo IRC originado na LAN para a Internet;
- permitir o acesso a um servidor *web* localizado na LAN (via NAT);
- restringir o uso de certos protocolos (por exemplo, FTP ou Telnet) apenas a usuários da LAN autorizados.

Estas regras atuam pela avaliação dos cabeçalhos das mensagens tais como IP Origem, IP Destino, Tipo do Protocolo comparando com as informações configuradas pelo administrador de segurança.A janela de Serviços, apresentada na Figura 2.2, contém uma tabela que mostra as regras, de acesso à rede, definidas para o *firewall*. As regras são ordenadas de cima para baixo, da mais restritiva para a mais genérica. A última regra é a regra default.

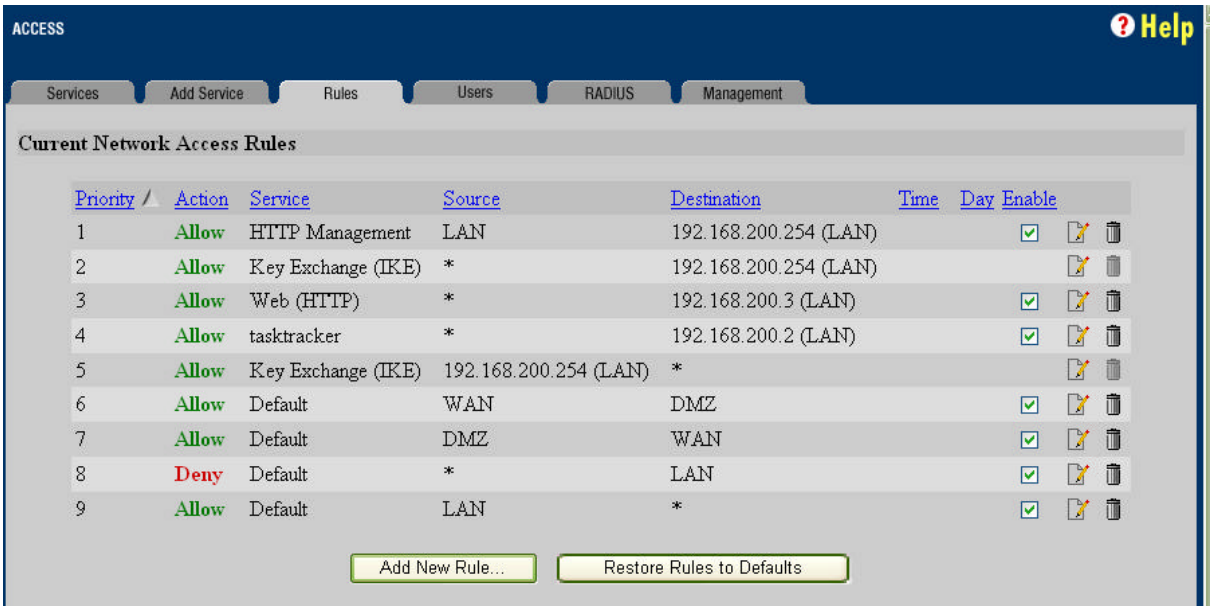

Figura 2.2 – Janela de configuração de regras do *firewall* SonicWall

Cada coluna da tabela de regras representa uma informação a ser filtrada:

- *Action*: Definida como *"Allow"* ou *"Deny"*, dependendo da intenção projetada para a regra.

- *Service*: É o nome do serviço (Protocolo + Número da Porta) ao qual a regra se refere.
- *Source*: Há três parâmetros para serem configurados para esta coluna: LAN, DMZ e WAN. Esses parâmetros são as faixas de endereços IP que estão presentes na rede a ser protegida pelo *Firewall.* Suas nomenclaturas oferecem uma boa indicação a respeito da localização física da origem da mensagem. A utilização do caractere "*" indica que qualquer endereço IP é aceitável para a regra.
- *Destination*: Idem à coluna *Source*, mas referentes ao destino das mensagens.
- *Time*: Este parâmetro permite especificar horários nos quais uma regra deve atuar. A condição *default* é sempre atuar.
- *Enable*: Este parâmetro indica se a regra está ou não ativa.

As regras de filtragem dos *firewalls* da SonicWall sempre são relacionadas a um tipo de serviço. Diversos tipos de serviços pré-definidos podem ser visualizados e configurados manualmente a partir da janela de configuração de serviços como mostra a Figura 2.3. Novos serviços, tais como serviços que atendem a uma necessidade exclusiva da rede podem ser adicionados de forma a oferecer flexibilidade para as regras de filtragem. A coluna *Service* define um parâmetro que necessita de configuração. Um serviço é a união entre um tipo de protocolo (TCP, UDP ou ICMP) e uma porta bem conhecida (1 – 1023) [ZWI00]. Por exemplo, o serviço http é definido como sendo o protocolo TCP na porta 80. Assim, é preciso especificar quais protocolos e quais portas referenciam um determinado serviço identificado por um nome único ou *alias*.

Os parâmetros a serem configurados para os serviços são:

- nome do serviço ou *alias*;
- porta a qual o serviço está associado;
- tipo do protocolo ao qual o serviço efetua a comunicação.

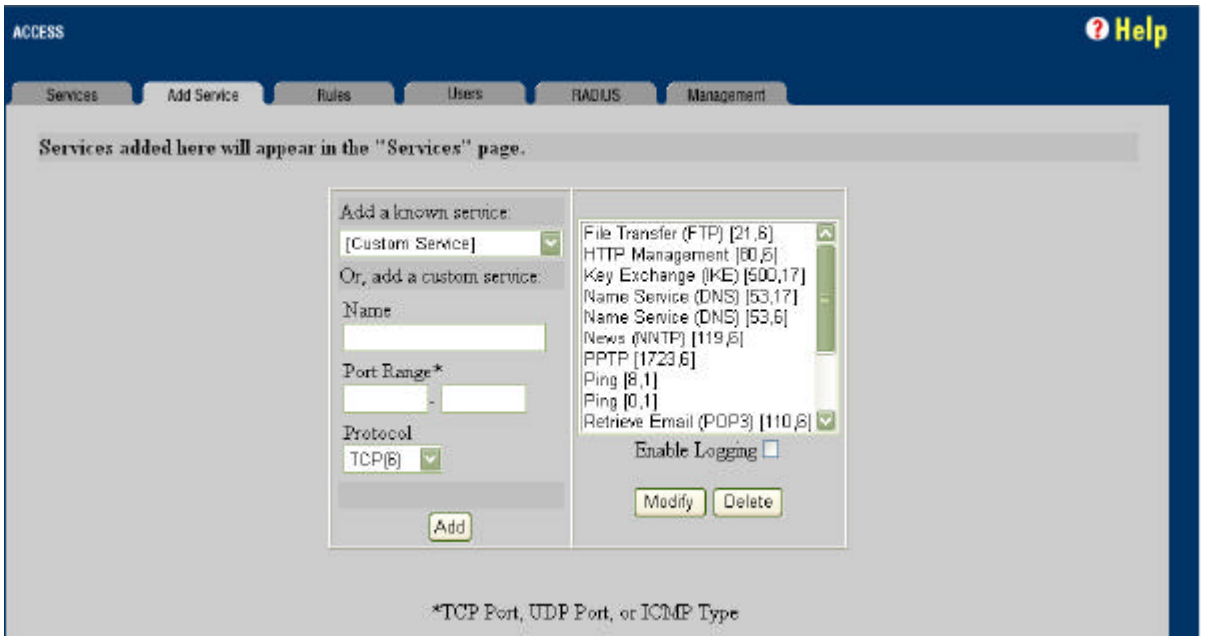

Figura 2.3 – Janela de configuração de serviços do *firewall* SonicWall

Também foi adicionada aos *firewalls* da SonicWall a funcionalidade de Gerenciamento de Banda (ou QoS). Desta forma, o administrador pode configurar a largura de banda mínima e/ou máxima em Kbps para um serviço em um determinado horário (ou sempre).

2.4. Firmato: Um Conjunto de Ferramentas para Gerenciamento de Firewalls

Em 1999, um grupo de pesquisadores dos Laboratórios Bell apresentaram o Firmato: um conjunto de ferramentas criado para facilitar o gerenciamento de *firewalls* orientado a arquivos [BAR99]. O grupo tinha quatro objetivos bem definidos:

- criar um meio de separação entre a definição das políticas de segurança e a sua implementação nos *firewalls* comerciais específicos existentes no mercado;
- tornar a definição das políticas de segurança independente de topologia da rede;
- gerar arquivos de configuração de *firewalls* automaticamente a partir da política de segurança;
- permitir uma metodologia de depuração de alto nível dos arquivos de configuração.

O modelo de configuração de *firewalls* apresentado pelo Firmato é composto pelos seguintes componentes:

- Um modelo entidade-relacionamento que oferece um *framework* para representação da política de segurança e a topologia da rede. Essa propriedade foi alcançada através da expressão das políticas de segurança em termos de "papéis" que definem as características assumidas pelos elementos da rede. Os papéis capturam perfeitamente a essência de uma política independente de topologia.
- Uma linguagem de definição – *Model Definition Language* (MDL) – utilizada como interface para definir uma instância do modelo entidade-relacionamento. Foi implementado um *parser* para a MDL que gera tais instâncias.
- Um compilador, que traduz um objeto instanciado em arquivos de configuração de um *firewall* específico. O conjunto de tais arquivos inclui a informação de topologia e regras.
- Uma interface gráfica que transforma os arquivos de configuração de *firewall* em uma representação gráfica das políticas de segurança sobreposta à topologia. Tal visualização permite uma avaliação rápida da viabilidade de uma dada política.

A maior parte do modelo proposto pelo Firmato está na definição de uma política de segurança independente de topologia. É nesse contexto que se introduz o conceito principal de "*papéis*". Um papel é uma propriedade que os diversos dispositivos de uma rede podem assumir. Os papéis definem as habilidades para ativação e aceitação de serviços. Por exemplo, o papel de um servidor de e-mail deveria definir a capacidade de aceitar o serviço *mail* (smtp – ou tcp na porta 25) de qualquer lugar da rede. De uma forma mais sofisticada, o dispositivo assume o papel de um servidor de e-mail.

Um papel não especifica somente as capacidades de aceitação de serviços, mas também, as características de ativação. Isto não é obrigatório, mas é natural que se defina, por exemplo, um papel tal como *web_enable*, que permite o acesso http ao mundo externo. Assim, os computadores da rede que assumirem esse papel, estariam habilitados a consultar páginas na *web*.

[BAR99**]** mostra, de forma clara, que um papel define, essencialmente:

1. Os serviços permitidos.

2. Os *peers* aos quais esses serviços se aplicam.

Os *peers* podem ser expressos em termos dos mesmos ou de outros papéis. Dessa forma, os papéis são atribuídos a dispositivos reais (máquinas), o que permite a união entre a política e a topologia. Por uma questão de conveniência, o Firmato propõe a utilização de grupos de papéis que são a reunião de diferentes tipos de papéis em conjuntos e que podem ser atribuídos aos dispositivos.

Papéis expressam capacidades positivas e, implicitamente, utilizam a estratégia "qualquer mensagem que não seja explicitamente permitida, deve ser bloqueada". Por exemplo, um dispositivo aceitará requisições http se e somente se a este dispositivo estiver atribuído o papel de servidor *web*.

Como a política de segurança é abstraída sob a forma de papéis, a topologia da rede também é modelada com uma abordagem entidade-relacionamento. Um *host* modela um dispositivo da rede através de seu nome e endereço IP. Um *host-group* representa um conjunto de dispositivos, definidos tanto pela sua faixa de endereços IP ou por um conjunto de outros grupos. Assim, os papéis podem ser atribuídos a ambos *host* e *host-groups*.

A utilização de herança surge, então, de maneira natural, tal que o conjunto de papéis assumidos por um host h_1 é o conjunto de todos os papéis assumidos pelo *host-group* H a que h_1 pertence[3].

Considere o exemplo da Figura 2.4. Um *gateway* conecta uma sub-rede de um setor de RH à Internet. É importante garantir que o acesso ao *gateway* esteja limitado apenas ao gerente que executa determinadas tarefas administrativas. Caso contrário, o controle de acesso e roteamento seriam facilmente quebrados. Assim, este *gateway* não poderá ser utilizado para outras tarefas e permanecerá escondido[4] dos outros operadores.

[3] *Host-group* H contém o host h_1 se o endereço IP de h_1 pertence à faixa de endereços IP de H.
[4] A terminologia correta é *stealthed*. O termo *stealthed* é utilizado para caracterizar a não visibilidade de um dispositivo ou serviço a outros dispositivos e usuários presentes na rede.

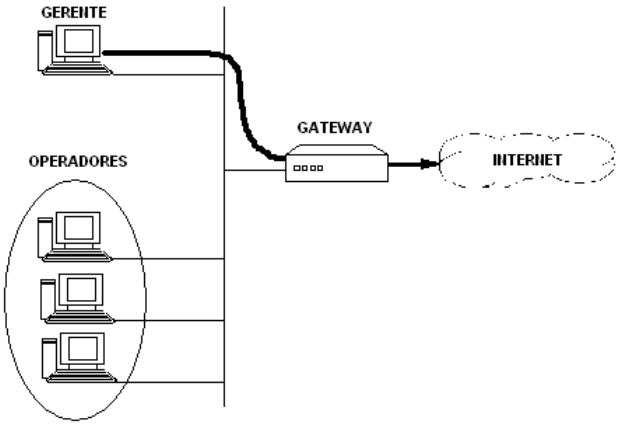

Figura 2.4 – Sub-rede do setor de RH conectada à Internet por um *gateway*. Apenas o gerente tem direito de acesso à Internet.

Enquanto a atribuição de papéis a um determinado *gateway* que permite o roteamento de um conjunto limitado de tipos de mensagens é simples, tal como no exemplo da Figura 2.4, este *gateway* poderia estar herdando outros papéis, o que poderia permitir um acesso indesejável. Para evitar esta possibilidade, o Firmato introduz o conceito de um grupo fechado de papéis.

Um grupo fechado de papéis é um grupo em que a herança de papéis não se aplica. Um *host* h_1, pertencente a um *host-group* H_1, que assume um grupo fechado de papéis, não pode herdar outros papéis atribuídos a qualquer outro *host-group* H_2 que contenha h_1. Um *host* pode assumir apenas um grupo fechado de papéis. De forma análoga, os grupos de papéis que podem herdar os papéis de outros grupos são ditos abertos. Por *default*, os grupos de papéis são sempre abertos. Conforme exposto, anteriormente, os papéis expressam uma capacidade positiva a um dispositivo. O mecanismo de grupo fechado de papéis oferece uma forma limitada de se expressar uma capacidade negativa. É possível notar que os papéis, por si mesmos, têm poder suficiente para implementar qualquer tipo de política de segurança. Isto pode ser alcançado pela separação dos grupos até que eles se tornem mutuamente exclusivos. Assim, não haveria herança de papéis. Mas isso pode se tornar uma tarefa muito extensa e de difícil manutenção. Através do exemplo acima, nota-se que um grupo fechado de papéis captura a idéia de exclusão de grupos, que é a forma mais comum de se expressar uma capacidade negativa em termos de política de segurança.

O Firmato foi um marco importante para a nova geração de ferramentas de gerenciamento de segurança. Esse trabalho demonstra que as tarefas de configuração e gerenciamento de *firewalls* pode ser feita em um nível alto de abstração. Muitos dos conceitos apresentados nessa iniciativa voltada ao gerenciamento de segurança podem ser vistos em trabalhos atuais. O Firmato é visto como o primeiro passo em direção à convergência entre segurança e gerenciamento de redes.

2.5. Distributed Firewalls

Firewalls convencionais são dependentes de restrições impostas pela topologia da rede para executar a filtragem das mensagens. No caso de ataques internos, existe a primitiva de que se um determinado tráfego não é visto pelo *firewall*, ele não pode ser filtrado; se for este o caso, há a necessidade de se implantar *firewalls* internos.

[IOA00] descreve várias desvantagens a respeito de *firewalls* convencionais, como:

- *firewalls* tendem a se tornar gargalos nas redes de comunicação;
- *firewalls* precisam manipular, ao mesmo tempo, antigos e novos protocolos, o que torna sua programação complicada;
- ataques internos precisam ser bloqueados através da implantação de *firewalls* internos;
- *firewalls* tradicionais não são capazes de identificar um ataque vindo através de um túnel VPN que tenha sido corrompido sem o conhecimento do usuário que estabeleceu a conexão;
- a dificuldade de se administrar redes com vários pontos de interconexão com a Internet;
- *firewalls* tradicionais são incapazes de analisar uma conexão criptografada fim-a-fim;
- sem a adição de grande poder de processamento, os *firewalls* tradicionais não são capazes de realizar uma filtragem de pacotes com maior detalhamento.

Para resolver esses problemas, ainda que mantendo as vantagens de se utilizar um *firewall* [BEL99] propõem o conceito de um *firewall* distribuído. Nesta abordagem, em um

firewall distribuído a política de segurança é definida de maneira centralizada, mas aplicada pelos dispositivos da rede individualmente (computadores e roteadores).

Em um ambiente organizacional típico, os usuários não são necessariamente os administradores dos computadores que utilizam. Ao contrário, para simplificar a administração do sistema e permitir algum nível de controle central, utiliza-se algum tipo de pacote de gerenciamento de sistemas para administrar as máquinas dos usuários. Esses pacotes permitem a instalação de *patches*, atualizações de antivírus, distribuição de novos softwares, etc. O controle de um *firewall* distribuído utiliza esse mecanismo [IOA00].

A política de segurança é aplicada por cada um dos hosts que participam do *firewall* distribuído individualmente. O administrador de segurança – que não precisa mais ser, necessariamente, um administrador local, uma vez que a topologia não é mais uma restrição, define a política de segurança em termos de identificadores de hosts. O conjunto de políticas resultante (provavelmente, mas não necessariamente, compilada para algum formato interno) é enviado aos hosts que fazem parte do *firewall* distribuído. A cada mensagem de entrada ou saída, esse conjunto de políticas é consultado para verificar se a mensagem está de acordo com as regras. É natural que se pense que tudo esteja acontecendo na camada de rede ou de transporte, mas as políticas podem ser aplicadas igualmente até a camada de aplicação. Por exemplo, uma política de segurança poderia definir que não é permitido aos *web browsers* locais executarem códigos Java ou JavaScript.

Firewalls distribuídos são baseados a partir de três noções [BEL99]: (a) uma linguagem de política que define que conexões são permitidas ou que devem ser bloqueadas; (b) algum tipo de ferramenta de gerenciamento de sistema (Microsoft SMS ou ASD); (c) IPSEC, um mecanismo de criptografia e autenticação para TCP/IP [RFC2401].

A aplicação da política se torna mais útil quando o host é identificado através de um certificado. Assim, a máquina poderá afirmar com muito mais segurança que ela é realmente quem diz ser.

A distribuição das políticas pode se dar de várias formas. Por exemplo, as regras poderiam ser enviadas diretamente aos hosts que deverão implementá-las, ou então, poderiam ser atribuídas credenciais aos usuários que deverão utilizá-las na comunicação com outros hosts da rede, ou poderia ser uma combinação de ambas.

Na implementação de um *firewall* distribuído são necessários três componentes:

- uma linguagem que expresse as políticas de segurança;
- um mecanismo que distribua as políticas de forma segura;
- um mecanismo que implemente as políticas de segurança.

A idéia básica é muito simples. Um compilador (ou interpretador) traduz a linguagem da política para algum formato interno. O sistema de gerenciamento distribui as políticas para todos os hosts que deverão estar protegidos pelo *firewall* distribuído e os pacotes de dados serão aceitos ou rejeitados por cada host de acordo com a política e/ou credencial autenticada do usuário que esteja enviando o pacote. É possível utilizar muitos tipos de linguagens para se descrever as políticas, incluindo-se esquemas orientados a arquivos similares ao Firmato **[**BAR99**]**, as GUIs – que são encontradas nos mais modernos *firewalls* comerciais e outras linguagens de descrição de políticas genéricas tais como o KeyNote **[**RFC2704]. A natureza da linguagem é irrelevante, desde que esta seja hábil o suficiente para expressar corretamente a política desejada. Suponha o exemplo de política, abaixo:

```
inside_net = x509{name="*.example.com"};
mail_gw = x509{name="mailgw.example.com"};
time_server = Ipv4{10.10.10.3};
allow smtp(*, mail_gw);
allow smtp(mail_gw, inside_net);
allow ntp(time_server, inside_net);
allow *(inside_net, *);
```

Pacotes SMTP de fora, somente poderão alcançar a máquina através da apresentação de um certificado de identificação ao servidor de e-mail. O servidor de e-mail pode enviar pacotes SMTP para todas as máquinas internas. Pacotes NTP podem ser distribuídos a partir de um dado endereço IP a todas as máquinas internas. Finalmente, qualquer chamada para fora da rede é permitida.

Um ponto importante é como os hosts serão identificados. *Firewalls* atuais dependem da topologia da rede; assim, as interfaces de rede são classificadas como sendo "internas", "externas", "DMZ", etc. Uma vez que o conceito de *firewalls* distribuídos traz uma abordagem de independência da topologia, a idéia de mundo interior e exterior é descartada.

O segundo identificador de host mais comum é o seu endereço IP. Ou seja, é dada a condição de confiança aos pacotes "de" ou "para" um endereço IP específico na rede,

relativos a um ou diversos protocolos tais como solicitações ao serviço telnet ou FTP, recebimento de pacotes SMTP da Internet, etc. *Firewalls* distribuídos utilizam o endereço IP somente para identificação de host considerando-se um nível reduzido de segurança. Em *firewalls* distribuídos é preferível utilizar-se o nome contido no "certificado criptografado" usado com IPSEC. Os certificados podem ser considerados muito confiáveis enquanto um identificador único. Eles são independentes da topologia e também são muito difíceis de serem forjados – prática utilizada pelo método de *spoofing*. Se a uma determinada máquina são autorizados certos privilégios baseados em seu certificado, esses privilégios estarão garantidos independentemente da localização física dessa máquina.

Mesmo com todas as vantagens apresentadas pelos idealizadores da proposta de *firewalls* distribuídos, é muito difícil que se abandone a idéia de se utilizar *firewalls* convencionais. É bem possível que, em uma configuração ideal, as duas técnicas se complementem no sentido de se oferecer maior segurança às redes de comunicação. Além disso, a adição de IPSEC pode ser utilizada no desenvolvimento de *firewalls* mais poderosos.

Entretanto, não importando a técnica que se utilize na implementação do *firewall*, seja ela convencional ou distribuída, um conceito é comum: as regras definidas pela política de segurança devem permanecer centralizadas. Dessa forma é possível diminuir os riscos de que um ataque à rede tenha sucesso.

2.6. Conclusões do Capítulo

Este capítulo apresentou os diversos aspectos relacionados à segurança de redes pela utilização de *firewalls*.

A Seção 2.2 demonstrou como um *firewall* pode ser implementado a partir da definição das regras para filtros de pacotes, estáticos e dinâmicos. Para exemplificar, a Seção 2.3 apresentou um *firewall* comercial e os seus modos de configuração.

Os problemas de gerenciamento de segurança introduzidos pela implantação de diversos filtros de pacotes nos diversos pontos de entrada de redes corporativas são abordados nas Seções 2.4 e 2.5. A seção 2.4 apresentou o Firmato [BAR99]: um conjunto de ferramentas voltadas ao gerenciamento e distribuição das regras de *firewalls*. O Firmato também introduz o conceito de política de segurança e papéis. Será demonstrado, nos próximos capítulos, que estes dois conceitos são o objeto de estudo principal deste trabalho.

Na Seção 2.5 foi apresentada uma nova abordagem ao problema de gerenciamento e distribuição das políticas de segurança. No paradigma de *firewalls* distribuídos, [BEL99] propõe que cada computador da rede é, potencialmente, um *firewall*, com suas próprias regras que permitem ou bloqueiam os acessos aos seus recursos compartilhados.

Capítulo 3

Padrões do IETF

3.1. Introdução

Este capítulo descreve os principais conceitos e padrões definidos pelo IETF relacionados ao gerenciamento de redes baseado em políticas.

A Seção 3.2 introduz o conceito de políticas. As Seções 3.3 a 3.6 descrevem os modelos de informação propostos pelo DMTF e pelo IETF para a representação das políticas. A Seção 3.7 demonstra as possibilidades de mapeamento dos modelos em um serviço de diretório baseado em LDAP. Na Seção 3.8 são apresentados os conceitos relativos ao protocolo COPS, utilizado para a distribuição das políticas de segurança. A Seção 3.9 conclui o capítulo.

3.2. Políticas

Política, em sua forma mais simples, é um conjunto de regras que descrevem a ação a ser tomada quando uma condição específica existir [QOS99]. As políticas descrevem o comportamento dos recursos da rede ao receber uma requisição de usuário ou quando os serviços de rede interagem entre si.

A [RFC3198] descreve as políticas a partir de duas perspectivas:

- Um objetivo, direcionamento ou método de ação utilizado para tomada de decisões presentes e futuras executadas dentro de um contexto particular.

- Um conjunto de regras para administrar, gerenciar e controlar o acesso a recursos.

Uma regra de política é normalmente composta por outras regras de políticas, o que, em efeito, pode-se dizer que políticas contêm políticas, ou que políticas podem ser agrupadas em grupos e os grupos podem ser aninhados para representar uma hierarquia de políticas [RFC3060]. Esta noção de hierarquia é muito importante, uma vez que habilita a construção de políticas complexas a partir de um conjunto de políticas simples e, também, simplifica o gerenciamento.

Como ilustração das regras de políticas, considere o conjunto de políticas de segurança relativas ao serviço *web* para a topologia de uma pequena rede conforme a Figura 3.1:

- tráfego interno da LAN é sempre permitido;
- a sub-rede dos administradores pode sempre acessar páginas *web* na Internet, não importando o horário;
- a sub-rede de empregados pode acessar páginas *web* na Internet, apenas entre 11:30h e 13:30h;
- computadores externos podem acessar o servidor *web* localizado na DMZ;
- qualquer outra opção deve ser bloqueada.

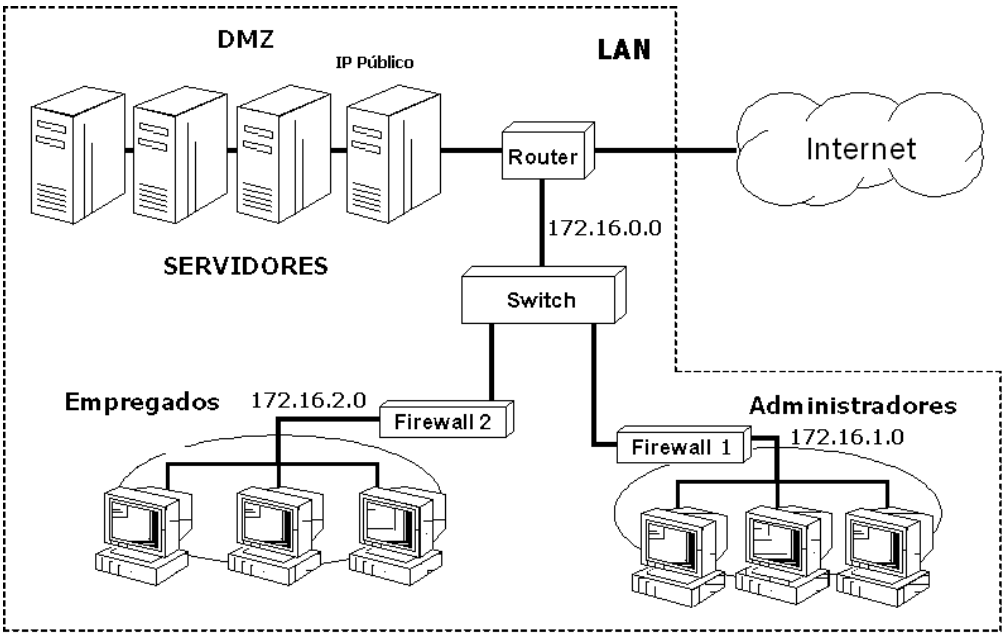

Figura 3.1 – Topologia da rede de exemplo.

3.3. Directory Enable Networks - DEN

O DEN – *Directory Enabled Networks* – é uma iniciativa de padronização de um esquema de diretório extensível para redes heterogêneas. Dessa forma, seria permitido às aplicações utilizar a infra-estrutura da rede em benefício de seus usuários, desenvolvedores e administradores de redes. O DEN representa os aspectos físicos, lógicos e de políticas dos elementos de rede e seus serviços [WON01]. A Figura 3.2 demonstra a filosofia de operação do DEN segundo o DMTF [BUP99].

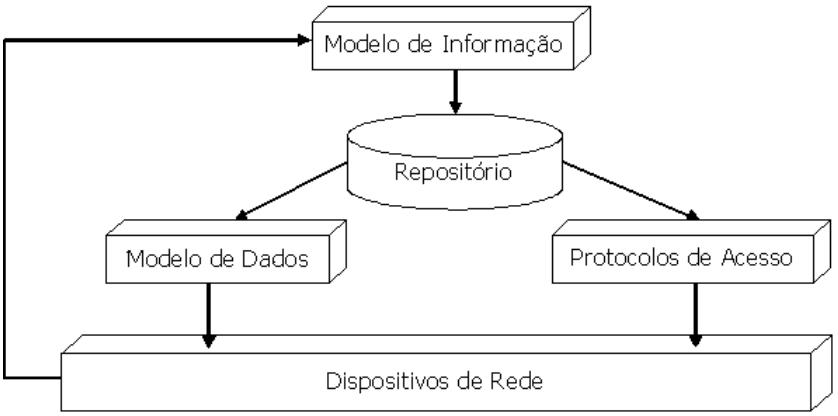

Figura 3.2 – Filosofia de operação do DEN

3.4. Common Information Model – CIM

A dificuldade de integração entre sistemas trouxe a necessidade de se padronizar um modelo de informações unificado para que dados pudessem ser compartilhados entre aplicações. A tradução de dados de um sistema para outro pode ser muito difícil, ineficiente e de alto custo. O CIM é um modelo de informações utilizado para descrever informações de gerenciamento que não estejam limitadas a uma implementação particular. Esse modelo foi concebido em 1997 pelo DMTF e descreve objetos, atributos e relacionamentos em um sistema único, rede de sistemas ou corporações. A partir desse modelo é possível definir uma política de controle e gerenciamento para esses objetos. O modelo definido pelo CIM se concentra na representação do sistema e no gerenciamento das políticas. Ele está projetado para manipular uma grande variedade de informações tais como aplicações, usuários, serviços, redes e/ou dispositivos. O CIM é expresso através de diagramas UML – *Unified Modeling Language* – e na linguagem MOF – *Managed Object Format*.

O esquema do CIM permite que aplicações de fabricantes diferentes, em plataformas diferentes, possam descrever dados de gerenciamento em um formato padrão. Dessa forma, esses dados podem ser compartilhados entre uma grande variedade de aplicações de gerenciamento.

O CIM ainda está em fase de definição e requer especificações adicionais em sua hierarquia de classes. Para as classes relativas às políticas, o DMTF, em conjunto com o IETF, estendeu as classes do CIM de forma a refinar um modelo de classes de políticas. Este trabalho recebeu o nome de PCIM – *Policy Core Information Model* – [RFC3060].

3.5. Policy Core Information Model – PCIM

O PCIM apresenta um modelo de informação orientado a objetos para a representação de informações de políticas desenvolvido como uma extensão ao CIM. As classes de políticas e associações definidas pelo PCIM são suficientemente genéricas para representar políticas relacionadas a qualquer área de atuação.

Um exemplo disso pode ser visto através do documento *Service Level Agreement* – SLA – e em seus objetivos e métricas, no documento *Service Level Objectives* – SLO – que são utilizados na especificação dos serviços que uma rede deve oferecer a um dado usuário ou cliente. O SLA é normalmente escrito em termos de negócio utilizando uma linguagem de alto nível. O SLO refere-se a métricas específicas a respeito dos acordos definidos no SLA. Estas descrições de alto nível a respeito dos serviços e métricas da rede precisam ser traduzidas para uma especificação de nível mais baixo, ainda que independentes de dispositivos e/ou fabricantes. As classes do PCIM servem como base para a tradução da descrição dos serviços de rede e métricas definidas pelo SLA e SLO para um nível mais baixo de especificação ainda que independente de dispositivos e fabricantes [RFC3060].

3.5.1. Descrição do Modelo

O modelo do PCIM define duas hierarquias de classes de objetos:

- classes estruturais representam as informações e controle das políticas;
- classes de associação indicam como as instâncias das classes estruturais se relacionam umas com as outras.

As classes definidas pelo PCIM servem como uma hierarquia de classes extensível (através de especialização) para a definição de objetos de políticas que habilita desenvolvedores, administradores de rede e administradores de políticas a representar políticas de tipos diferentes.

Uma rede baseada em políticas pode ser modelada por uma máquina de estados onde as políticas são utilizadas para controlar em que estado determinado dispositivo deveria estar ou que novo estado este dispositivo pode assumir em um determinado tempo. A partir desta abordagem, uma política pode ser caracterizada pelo conjunto de várias regras de políticas. Cada regra é composta por um conjunto de condições e um conjunto de ações conforme a Figura 3.3.

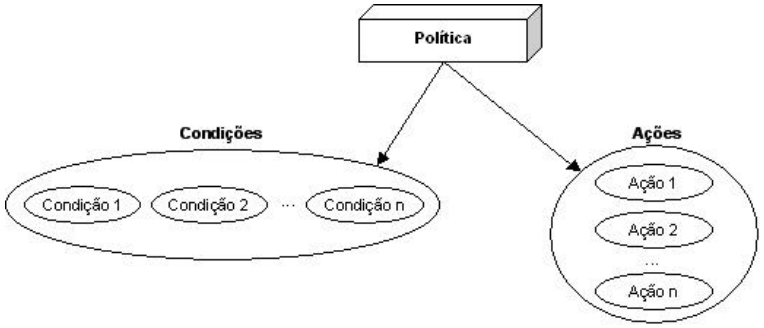

Figura 3.3 – Composição de políticas

Essas regras podem ser agregadas sob a forma de grupos de regras e, por sua vez, os grupos podem ser aninhados para representar uma hierarquia de políticas. A Figura 3.4 demonstra a estrutura da organização hierárquica das políticas.

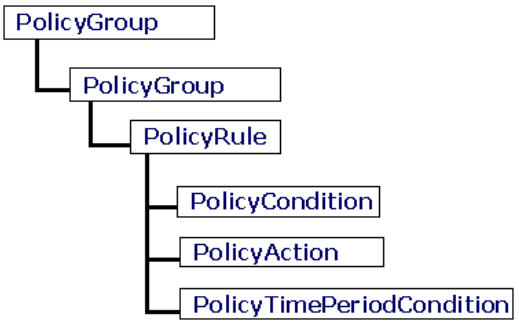

Figura 3.4 – Estrutura da organização hierárquica das políticas no PCIM.

Na Figura 3.4, *PolicyRule* representa a semântica "se **Condição** então **Ação**" associada à política. Assim, se um conjunto de condições associadas a uma regra de política for avaliado como verdadeiro, então o conjunto de ações associadas à mesma regra deverá ser

executado. Dessa forma é possível manter um objeto congelado em um determinado estado ou iniciar uma transição entre os estados permitidos para o objeto. É possível, também, especificar a ordem de execução das ações associadas às regras, assim como se uma determinada ordem de execução é recomendável ou obrigatória. O PCIM permite, opcionalmente, que seja definido um conjunto de intervalos de tempo, nos quais uma determinada regra de política pode ser ativada.

A Figura 3.5 representa o modelo de classes definido pelo PCIM.

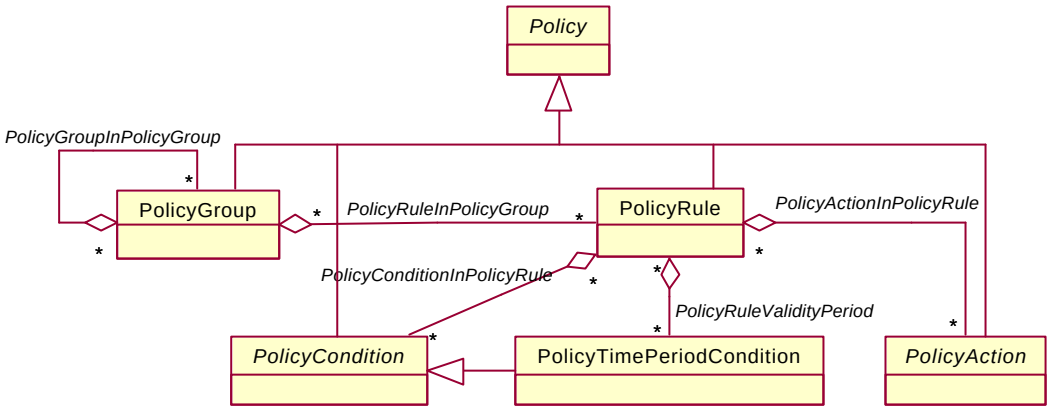

Figura 3.5 – Hierarquia de classes do PCIM

O PCIM divide condições e ações em duas categorias: 1) específicas, quando as condições e ações estão associadas a apenas uma regra; 2) reusáveis, quando as condições e ações podem estar associadas a mais de uma regra. Uma classe denominada *PolicyRepository* representa um *container* para o armazenamento de condições e ações reusáveis [RFC3060].

A classe abstrata *Policy* atua como base da hierarquia do PCIM. A classe *PolicyGroup* representa uma agregação de *PolicyRules* ou agregação de *PolicyGroups*, mas não de ambas. As *PolicyGroups* permitem que sejam modeladas interações entre objetos que têm uma interdependência bastante complexa. Não há restrição quanto ao nível de agregação entre *PolicyGroups*. As condições e ações associadas com uma regra de política são modeladas, respectivamente, com subclasses de *PolicyCondition* e *PolicyAction*. Como mencionado anteriormente, as regras de políticas também podem ser associadas com um ou mais períodos de tempo, indicando quando a política poderá ser aplicada ou não. Esses períodos de tempo são modelados através da classe *PolicyTimePeriodCondition* [RFC3060].

Os objetos que representam regras (*PolicyRule*) podem ser priorizados em relação a outros, fornecendo os instrumentos para a solução de conflitos entre eles. As ações associadas a cada uma destas regras podem ser ordenadas, possibilitando a especificação da seqüência das ações a serem executadas.

O conjunto de condições de políticas pode ser expresso de duas formas:

- *Disjunctive Normal Form* – DNF: um conjunto de condições construídas com o operador lógico AND e unidas através do operador lógico OR.
- *Conjunctive Normal Form* – CNF: um conjunto de condições construídas com o operador lógico OR e unidas através do operador lógico AND.

Além disso, as condições ainda podem ser negadas individualmente e agrupadas através da atribuição de um valor numérico do tipo inteiro, onde todas aquelas que possuírem valores idênticos são consideradas pertencentes ao mesmo grupo de condições.

Por exemplo, admitindo uma regra que agregue cinco *PolicyConditions*, chamadas de C1, C2,C3,C4,C5, e com os seguintes valores de agrupamento e negação:

- C1: *GroupNumber* = 1, *ConditionNegated* = FALSE
- C2: *GroupNumber* = 1, *ConditionNegated* = TRUE
- C3: *GroupNumber* = 1, *ConditionNegated* = FALSE
- C4: *GroupNumber* = 2, *ConditionNegated* = FALSE
- C5: *GroupNumber* = 2, *ConditionNegated* = FALSE

Para a forma DNF, a condição geral de validação desta regra será dada por:

(C1 AND (NOT C2) AND C3) OR (C4 AND C5)

Para o caso da forma CNF:

(C1 OR (NOT C2) OR C3) AND (C4 OR C5)

Em ambos os casos, estas expressões lógicas são testadas para verificar a possibilidade de execução das ações associadas a esta regra [RFC3060].

Além das classes estruturais, o *framework* do PCIM fornece diversas classes que dão o suporte para as associações e agregações que surgem a partir do relacionamento entre elas. Ainda, na Figura 3.5 podem ser identificadas as seguintes associações:

- *PolicyConditionInPolicyRule*
- *PolicyActionInPolicyRule*
- *PolicyRuleInPolicyGroup*
- *PolicyGroupInPolicyGroup*
- *PolicyRuleValidityPeriod*

3.5.2. Arquitetura de redes baseadas em políticas

O paradigma de redes baseadas em políticas é muito promissor no que diz respeito ao gerenciamento de redes de computadores. As políticas descrevem o comportamento desejado em uma linguagem de alto nível e independentemente dos dispositivos e topologia da rede [POL02].

Neste contexto, duas entidades se destacam: o PEP – *Policy Enforcement Point* e o PDP – *Policy Decision Point*. Os PEPs residem, tipicamente, nos dispositivos gerenciados. Seu papel é implementar os comandos recebidos, na forma de dados de configuração, dos PDPs. O PDP processa as políticas de alto nível com informações sobre o estado da rede e gera os dados de configuração para serem enviados aos PEPs. Caso haja alguma alteração nas políticas ou no estado da rede, o PDP é o responsável por reajustar o comportamento dos dispositivos através do envio dos dados de configurações atualizados aos PEPs.

A Figura 3.6 é uma ilustração desta arquitetura, a qual será utilizada como base para a implementação do modelo proposto por este trabalho.

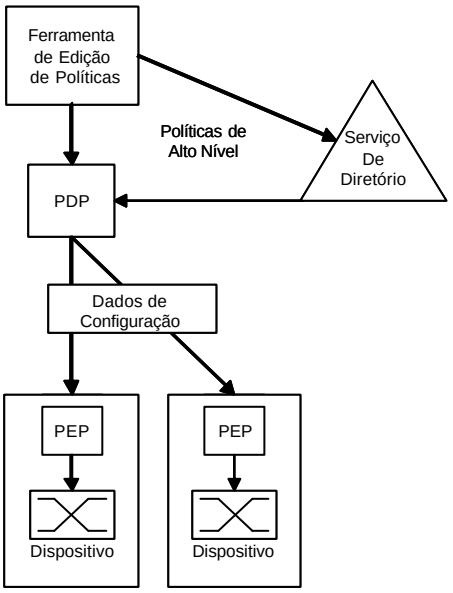

Figura 3.6 – Arquitetura de implementação do PCIM

Através da ferramenta de edição de políticas, as informações de políticas, de alto nível, são inseridas em um repositório, o qual é implementado, tipicamente, por um serviço de diretório tal como o LDAP. O PDP utiliza as informações contidas no repositório para efetuar as decisões de política.

Por exemplo, considere a seguinte política para a rede da Figura 3.1: A sub-rede de empregados (172.16.2.0) pode acessar páginas *web* (Protocolo TCP / Porta 80) na Internet (WAN), apenas entre 11:30h e 13:30h. Neste contexto, o administrador do sistema, através da ferramenta de gerenciamento, insere uma regra de política expressa, segundo o modelo do PCIM, como:

SE IP_Origem = "empregados" & Protocolo = TCP & Porta = 80 &
IP_Destino = "WAN" & Horário = 11:30:00 & Horário = 13:30:00 *ENTÃO*
permitir acesso
SENÃO
negar acesso
FIM-SE

Toda vez que o PEP é inicializado, ele solicita ao PDP que regras que devem ser implementadas para os papéis que o PEP esteja assumindo. A seguir, o PDP consulta o diretório em busca dessas regras e as devolve sob a forma de dados de configuração. O PEP, então, traduz as regras para um formato que o dispositivo de implementação das regras possa entender.

3.6. Policy Core Information Model Extensions – PCIMe

A padronização do modelo oferecido pelo PCIM resultou em outros trabalhos relacionados ao gerenciamento de redes baseado em políticas. Dentre esses trabalhos, destacam-se o *Policy QoS Information Model* – QPIM [SNI01], *IPSec Configuration Policy Model* – ICPM [JAS01], *Information Model for Describing Network Device QoS Datapath Mechanisms* – QDDIM [MOO01] e o RBPIM: Um Modelo de Políticas de Segurança Baseado em Papéis [NAB03].

Todos esses trabalhos demonstraram a necessidade de um modelo mais refinado que o fornecido pelo PCIM. A partir de suas definições, duas alterações foram efetuadas no modelo. A primeira foi a introdução de vários elementos completamente novos que permitiram ao PCIM abranger novas áreas, tais como a filtragem de pacotes (objeto de estudo deste trabalho). A segunda foi descontinuar alguns elementos e a definição de outros em seu lugar. Em ambos os tipos, as alterações foram feitas de tal forma a não comprometer a interoperabilidade com implementações projetadas no modelo original do PCIM.

3.6.1. Alterações no Modelo PCIM

Esta seção faz uma breve descrição das mudanças sofridas pelo *framework* do PCIM, que deram origem ao PCIMe [RFC3460]. A maioria dessas alterações foram contribuições obtidas a partir dos trabalhos QPIM [SNI01], ICPM [JAS01] e QDDIM [MOO01].

A Figura 3.7 apresenta um diagrama simplificado, baseado no novo modelo de informação introduzido pelo PCIMe [RFC3460]. Este diagrama apresenta as principais classes e associações relevantes ao escopo do modelo.

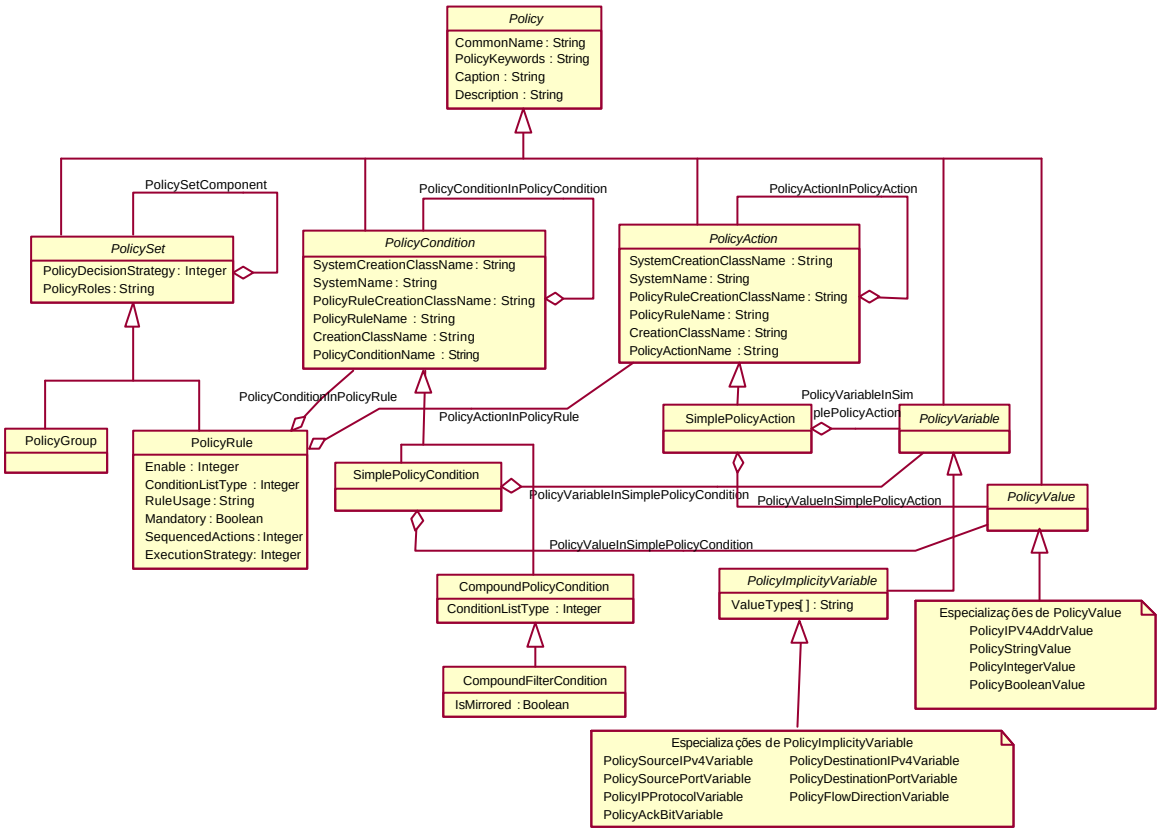

Figura 3.7 – Principais classes do modelo do PCIMe

O CIM especifica uma abstração da classe *ManagedSystemElement*, a classe *ManagedElement*, a qual, em conjunto com classe *Policy*, atua como classe base da hierarquia de classes do PCIMe.

A classe *PolicySet* é originada a partir da abstração de uma regra ou um conjunto de regras. É esta classe que define a semântica de aninhamento de regras ou de se formar conjuntos de regras dentro de grupos. Duas propriedades são definidas em *PolicySet*: *PolicyDecisionStrategy* e *PolicyRoles.* A primeira define o formato de avaliação das regras contidas em um conjunto de regras. Para isso, podem ser utilizadas duas estratégias: *FirstMatching* e *AllMatching.* A estratégia *FirstMatching* é utilizada para efetuar a validação das regras pertencentes a um conjunto de maneira tal que será executada a ação associada à primeira regra que for validada como verdadeira dentro do conjunto de regras. A estratégia *AllMatching* força o sistema a avaliar todas as regras do conjunto. Serão, então, executadas as ações associadas às regras somente se todas essas regras forem validadas como verdadeiras. Além disso, o conceito de papéis foi estendido à classe *PolicyGroups* (já presente na classe *PolicyRule*) [RFC3460]. Assim, a propriedade *PolicyRoles* é descontinuada de *PolicyRule* e recriada em *PolicySet*. Para *PolicyGroups*/*PolicyRules* aninhadas, quaisquer papéis assumidos por um grupo/regra de mais alta hierarquia são automaticamente herdados pelas *PolicyGroups/PolicyRules* aninhadas.

As agregações *PolicyGroupInPolicyGroup* e *PolicyRuleInPolicyGroup* foram combinadas em uma única agregação *PolicySetComponent* em *PolicySet*. Esta nova agregação introduz a semântica da representação de uma *PolicyRule* executando dentro do escopo de outra *PolicyRule*. A propriedade *PolicySetComponent.Priority* define a ordem[5] em que as regras devem ser executadas dentro de uma mesma árvore *PolicySet*. Entretanto, um recurso pode possuir várias árvores *PolicySet* disjuntas, que podem estar reunidas através de um papel ou pela combinação de papéis. Dessa forma, surge a necessidade de se estabelecer o conceito de prioridade entre árvores. A Figura 3.8 mostra a associação *PolicySetInSystem*, derivada de *PolicyInSystem* (do CIM) que também define uma propriedade *Priority* usada para atribuir uma prioridade relativa entre a uma instância *PolicySet*[6] S1 dentro de um escopo administrativo em que S1 não seja um componente de outra instância *PolicySet* S2.

Em função de uma possível confusão entre o componente de armazenamento de dados do *framework* de políticas "Serviço de Diretório" da Figura 3.6 e classe *PolicyRepository*, concluiu-se que este não é um bom nome para representar o container de políticas reusáveis.

[5] Quanto maior valor da propriedade Priority, maior a prioridade da regra.
[6] PolicySet é uma classe abstrata e, portanto, não pode ser instanciada. Entretanto, a referência a uma instância de PolicySet deve ser interpretada como uma ou várias instâncias de PolicyRule, PolicyGroup.

Assim, a classe *PolicyRepository* foi descontinuada e criada uma nova classe denominada *ReusablePolicyContainer*. A Figura 3.8 demonstra essa construção.

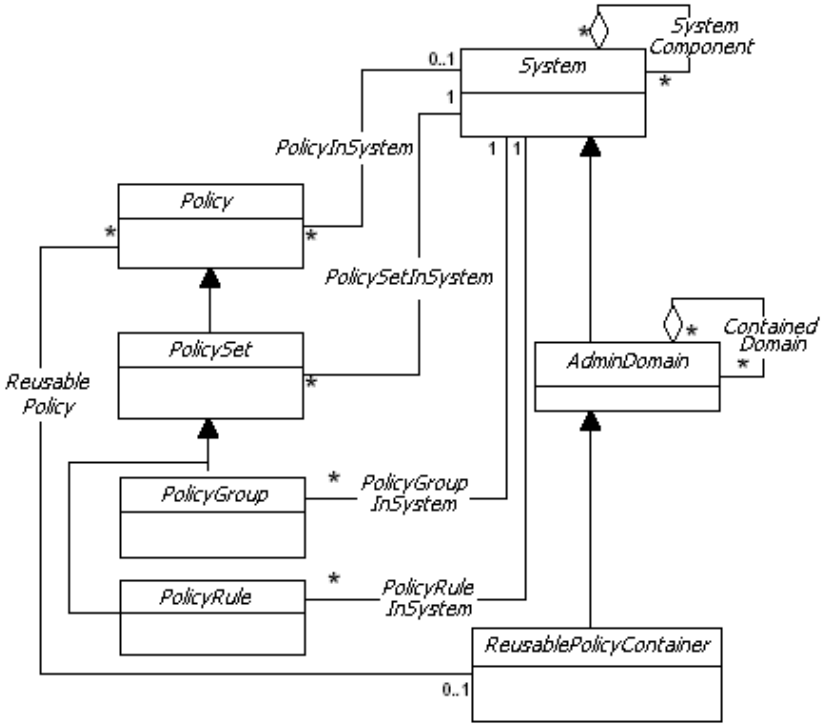

Figura 3.8 – Associações do PCIMe e a classe *ReusablePolicyContainer*

As associações *PolicyGroupInSystem* e *PolicyRuleInSystem* da Figura 3.8 são especializações de *PolicySetInSystem*. Conforme exposto na Seção 3.4.1, a classe *PolicyRule* define a semântica "se condição então ação". A propriedade *PolicyRule.ConditionListType* define a forma como as condições associadas à regra devem ser avaliadas, ou seja, se a composição será na forma DNF (1) ou CNF(2).

A introdução da classe *CompoundPolicyCondition* possibilitou a representação booleana do agrupamento de condições simples (*SimplePolicyConditions*). A classe *CompoundPolicyCondition* representa a semântica definida pela lógica DNF / CNF para a agregação de condições simples (da mesma forma que na agregação *PolicyConditionInPolicyRule* do PCIM) em uma única condição complexa. Este tipo de agrupamento será visto em detalhes na Seção 3.5.3.

Uma especialização de *PolicyCondition*, a classe *PolicyTimePeriodCondition* permite especificar o(s) período(s) de ativação de uma regra. Esta classe não está representada na Figura 3.7 para não prejudicar a visualização das classes principais.

A classe *CompoundPolicyCondition* agrupa instâncias de *SimplePolicyCondition* e *PolicyTimePeriodCondition* através da agregação *PolicyConditionInPolicyCondition*.

As agregações *PolicyConditionInPolicyRule* e *PolicyConditionInPolicyCondition* são especializações da classe *PolicyConditionStructure*. Esta classe define a propriedade 1) *PolicyConditionStructure.GroupNumber*: uma referência numérica para indicar o agrupamento ao qual as condições pertencem na construção de expressões na forma DNF e 2) *PolicyConditionStructure.ConditionNegated*: que indica se a condição deve ser negada.

Uma ação composta é uma construção conveniente para representar uma seqüência de ações a serem aplicadas como uma única ação dentro de uma regra. Similarmente à classe *CompoundPolicyCondition*, o PCIMe introduz o conceito de agrupamento de ações através da classe *CompoundPolicyAction*. Esta classe agrega instâncias de *SimplePolicyAction* através da agregação *PolicyActionInPolicyAction*. As agregações *PolicyActionInPolicyRule* e *PolicyActionInPolicyAction* são especializações da classe *PolicyActionStructure*. A propriedade *PolicyActionStructure.ActionOrder* define a ordem de execução das ações. A propriedade *PolicyActionStructure.SequencedActions* especifica se a ordenação das ações é obrigatória, recomendável ou indiferente.

A classe *SimplePolicyCondition* estende a estrutura básica da classe *PolicyCondition*, e introduz o par <VARIÁVEL>/<VALOR> para formar uma condição sob a forma de expressão booleana, onde a semântica "< VARIÁVEL > *MATCH* < VALOR >" define a validação da condição conforme a Figura 3.9.

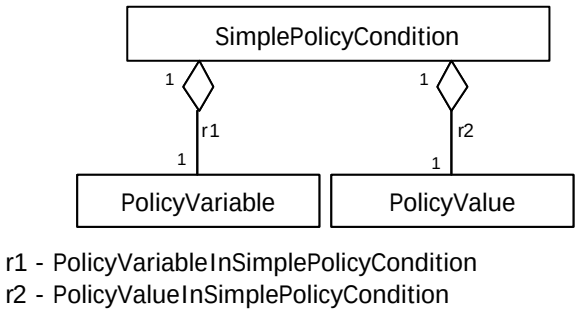

Figura 3.9 – Agregação do par < variável > / < valor > a uma condição

O operador *MATCH* fica implícito e não tem uma representação formal nas classes [RFC3460]. O par variável/valor associado a uma *SimplePolicyCondition* é criado, respectivamente, por instâncias das subclasses de *PolicyVariable* e *PolicyValue*.

De maneira similar à condição, uma *SimplePolicyAction* também estende a estrutura básica da classe *PolicyAction* através do uso do par <VARIÁVEL>/<VALOR>, mas com semântica *"SET* <VARIÁVEL > *TO* < VALOR >". Desta forma, é possível configurar a ação desejada. Duas agregações são utilizadas: *PolicyVariableInSimplePolicyAction* e *PolicyValueInSimplePolicyAction* conforme a Figura 3.10.

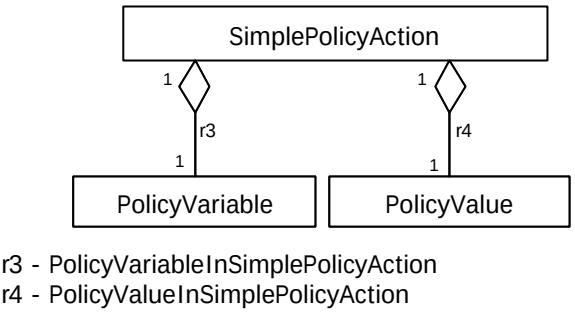

Figura 3.10 – Agregação do par < variável > / < valor > a uma ação

O PCIMe define duas especializações de *PolicyVariable*: *PolicyExplicitVariable* e *PolicyImplicitVariable*. A classe *PolicyExplicitVariable* define uma variável explícita que representa uma informação bem definida no contexto de modelagem do CIM, tal como diretórios, arquivos, serviços e aplicações. Por outro lado, pacotes trafegando em uma rede não podem ser explicitamente modelados ou instanciados, uma vez que não existe um gerenciamento ao nível de pacote (ao menos por enquanto) [RFC3460]. Assim, uma *PolicyCondition* fica impossibilitada de fazer uma referência a um modelo que represente, por exemplo, o endereço de origem de um pacote TCP/IP. Para esses casos, o PCIMe define uma *PolicyVariable* implícita ou *PolicyImplicitVariable*, que permite a modelagem dos parâmetros implícitos das mensagens que trafegam pela rede. Algumas dessas classes estão representadas na Figura 3.7. A maioria foi suprimida por questões de simplificação.

Uma especialização de *PolicyImplicitVariable* (que modela uma variável implícita particular) deve ser construída de forma que seu nome represente o significado da variável. Por exemplo, a classe que modela a porta de origem em uma comunicação TCP ou UDP recebe o nome *"SourcePort"*. O PCIMe define uma associação e um mecanismo de propósito geral que, juntos, caracterizam cada uma das variáveis implícitas:

- a associação *ExpectedPolicyValuesForVariable* especifica o conjunto de valores válidos para uma variável;

- uma lista de valores restritos que uma *PolicyVariable* pode conter (por exemplo, valores que uma variável deve validar) estão definidos nas propriedades de uma *PolicyValue* associada.

No exemplo citado acima, uma *PolicyImplicitVariable* seria instanciada por um objeto do tipo *PolicySourcePortVariable*. Esta instância representa a porta de origem para um pacote TCP. O atributo *ValueTypes* de *PolicySourcePortVariable* assumirá o valor "*PolicyIntegerValue*" que é o nome da classe que representa esse tipo de dado. Isto, por si só, restringe o tipo de dado armazenado pela instância de *PolicySourcePortVariable* para ser um valor inteiro. Ainda, é possível restringir os valores que *PolicySourcePortVariable* pode assumir através da configuração da propriedade *IntegerList* da classe *PolicyIntegerValue* (neste caso de 0 a 65535).

A Tabela 3.1 define os tipos de dados e a semântica para todas as especializações da classe *PolicyImplicitVariable* pré-definidos no PCIMe.

Tabela 3.1 – Especializações da classe *PolicyImplicitVariable*

Nome da Classe	**Tipo de Dado Permitido**
PolicySourceIPv4Variable	*PolicyIPv4AddrValue*
PolicyDestinationIPv4Variable	*PolicyIPv4AddrValue*
PolicySourceIPv6Variable	*PolicyIPv6AddrValue*
PolicyDestinationIPv6Variable	*PolicyIPv6AddrValue*
PolicySourcePortVariable	*PolicyIntegerValue(0..65535)*
PolicyDestinationPortVariable	*PolicyIntegerValue(0..65535)*
PolicyIPProtocolVariable	*PolicyIntegerValue (0..255)*
PolicyIPVersionVariable	*PolicyIntegerValue (0..15)*
PolicyIPToSVariable	*PolicyIntegerValue (0..255)* *PolicyBitStringValue (8 bits)*
PolicyDSCPVariable	*PolicyIntegerValue (0..63)* *PolicyBitStringValue (6 bits)*
PolicyFlowIdVariable	*PolicyIntegerValue (0..1048575)* *PolicyBitStringValue (20 bits)*
PolicySourceMACVariable	*PolicyMACAddrValue*
PolicyDestinationMACVariable	*PolicyMACAddrValue*
PolicyVLANVariable	*PolicyIntegerValue (0..4095)* *PolicyBitStringValue (12 bits)*
PolicyCoSVariable	*PolicyIntegerValue (0..7)* *PolicyBitStringValue (3 bits)*
PolicyEthertypeVariable	*PolicyIntegerValue (0..65535)* *PolicyBitStringValue (16 bits)*

PolicySourceSAPVariable	*PolicyIntegerValue (0..255)* *PolicyBitStringValue (8 bits)*
PolicyDestinationSAPVariable	*PolicyIntegerValue (0..255)* *PolicyBitStringValue (8 bits)*
PolicySNAPOUIVariable	*PolicyIntegerValue (0..16777215)* *PolicyBitStringValue (24 bits)*
PolicySNAPTypeVariable	*PolicyIntegerValue (0..65535)* *PolicyBitStringValue (16 bits)*
PolicyFlowDirectionVariable	*PolicyStringValue ('IN", "OUT")*

A classe abstrata *PolicyValue* é utilizada para modelar os valores e constantes usados pelas condições de políticas. São os diferentes tipos de *PolicyVariables* que definem os tipos de classes derivadas de *PolicyValue* para representar os atributos básicos de uma rede. A Figura 3.7 representa algumas dessas classes.

As sub-classes de *PolicyValue* definem três tipos de valores: inteiros, faixas e conjuntos. Por exemplo, uma porta poderia ser definida utilizando-se a classe *PolicyIntegerValue* através de um único valor (80 para http), uma faixa (80 – 88) ou por um conjunto (80, 88, 8080) de portas respectivamente.

O PCIMe define as seguintes sub-classes para *PolicyValue*:

- classes de propósito gerais:
  - *PolicyStringValue*
  - *PolicyIntegerValue*
  - *PolicyBitStringValue*
  - *PolicyBooleanValue*
- classes para valores referentes à informação de Camada 3:
  - *PolicyIPv4AddrValue*
  - *PolicyIPv6AddrValue*
- classes para valores referentes à informação de Camada 2:
  - *PolicyMACAddrValue*

3.6.2. Representação de Dispositivos no PCIMe

Foi observado que não existe, no PCIM, um mecanismo para atribuir papéis a recursos. Por exemplo, enquanto no PCIM é possível associar a uma *PolicyRule* o papel *RoleA*, não há um meio de indicar que interfaces atendem a esse critério. Assim, o PCIMe introduz a classe *PolicyRoleCollection* para representar a coleção de recursos associados a um papel em particular. A ligação entre uma *PolicyRule*/*PolicyGroup* e um conjunto de recursos é dada por uma instância de *PolicyRoleCollection* através da configuração de valores equivalentes da propriedade *PolicyRoles* de *PolicyRule*/*PolicyGroup* e da propriedade *PolicyRoleCollection.PolicyRole.*

No CIM, *System* é a classe base para a descrição de dispositivos de rede e domínios de administração. A associação *PolicyRoleCollectionInSystem* define como várias *PolicyRoleCollections* podem fazer parte de um mesmo sistema. Os dispositivos que compartilham um mesmo papel são agrupados por uma instância da classe *PolicyRoleCollection* através da agregação *ElementInPolicyRoleCollection*, conforme a Figura 3.11.

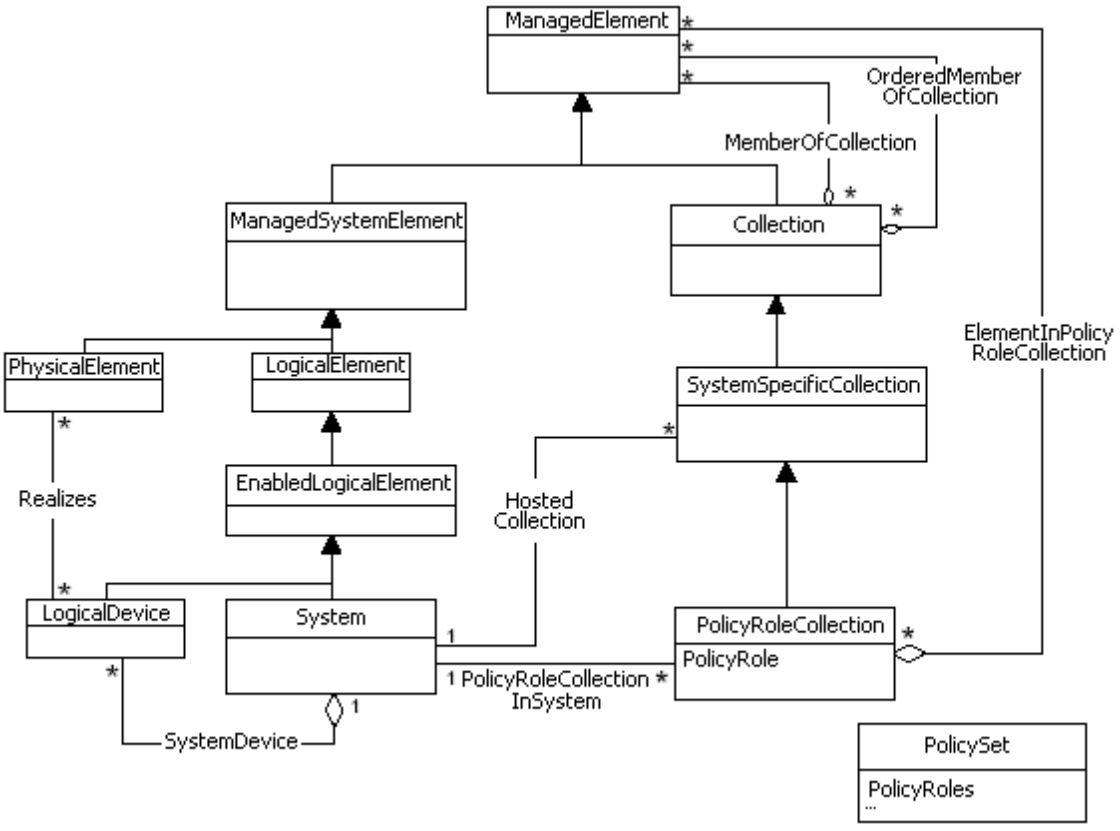

Figura 3.11 – Estrutura de representação de coleções e dispositivos no PCIMe

Para exemplificar como *PolicyRoleCollection* funciona, considere uma rede conectada à Internet que possua cinco servidores, que provêm os mais variados serviços (http, ftp, e-mail, telnet, etc.) aos seus usuários internos e externos. Um desses servidores disponibiliza o serviço http e pode ser acessado pela Internet. Assim, o administrador de políticas deve especificar várias regras que especificam que tipos de mensagens podem acessar esse servidor conforme a Tabela 3.2.

Tabela 3.2: Regras de filtragem para o serviço HTTP

REGRA	SENTIDO	PROTOCOLO	IP ORIGEM	IP DESTINO	PORTA ORIGEM	PORTA DESTINO	FLAG ACK	AÇÃO
1	IN	TCP	*	*	> 1023	80	0	pass
2	OUT	TCP	*	*	80	> 1023	1	pass
3	*	*	*	*	*	*	*	drop

Para todas as regras definidas na Tabela 3.2, a propriedade *PolicyRule.PolicyRoles* receberá o valor "Servidor_*HTTP*". Para indicar ao dispositivo que implementa o filtro de pacotes (por exemplo, um roteador) que este deve proteger uma sub-rede que contém um servidor http, será criada uma instância de *PolicyRoleCollection* com sua propriedade *PolicyRole* assumindo o mesmo valor de *PolicyRule.PolicyRoles*, ou seja, "Servidor_HTTP". A seguir, o administrador associa ao roteador a nova instância de *PolicyRoleCollection* através da agregação *ElementInPolicyRoleCollection*. Levando-se em consideração a arquitetura das redes baseadas em políticas apresentada na Seção 3.4.2, esta estratégia permite ao PDP determinar que as regras que assumem o papel "Servidor_HTTP" devem ser enviadas a todos os PEPs que também assumem o papel "Servidor_HTTP".

Note-se, então, que o papel é utilizado para otimizar a recuperação das políticas. Assim, a utilização de papéis implica na capacidade dos dispositivos (PEPs) de informar ao PDP os papéis assumidos por estes dispositivos. Esta estratégia é definida pelo protocolo COPS – *Common Open Policy Services* – e será demonstrada na Seção 3.7.

3.6.3. Domain-Level Packet Filtering no PCIMe

Além de preencher as lacunas do gerenciamento de políticas deixadas pelo PCIM, o PCIMe propõem um mecanismo unificado para expressar condições de políticas no domínio dos filtros de pacotes. Na Figura 3.12, cada *SimplePolicyCondition* representa um único campo a ser filtrado: Direção do Fluxo, Endereço IP de Origem, Porta de Origem, etc. O conjunto de várias *SimplePolicyCondition* referentes a uma única regra deve ser agrupado por

uma instância da classe *CompoundFilterCondition*. Esta classe é uma especialização da classe *CompoundPolicyCondition*.

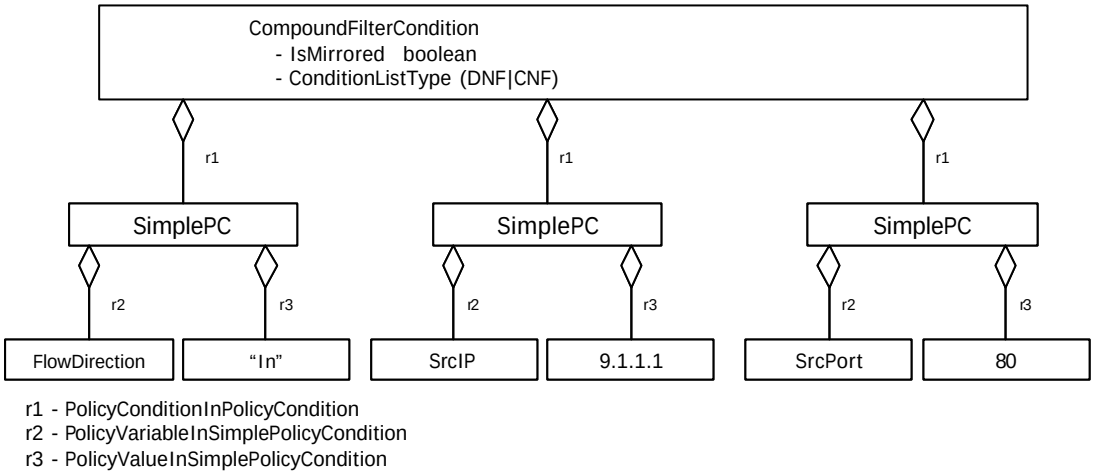

Figura 3.12 – Estrutura de condições complexas

As *SimplePolicyConditions* podem ser negadas individualmente quando agregadas em uma *CompoundFilterCondition* e a estratégia de avaliação segue a forma DNF / CNF, através da propriedade *ConditionListType* da classe *CompoundFilterCondition*, conforme a Figura 3.12. Além disso, esta classe introduz uma nova propriedade: *IsMirrored* (booleana). O propósito desta propriedade é validar pacotes viajando em ambas as direções em uma conexão de alto nível tal como numa sessão TCP. Quando esta propriedade assume o valor TRUE, pacotes adicionais também são validados pelo filtro, além daqueles que normalmente seriam validados. Para ilustrar o funcionamento da propriedade *IsMirrored*, considere o exemplo da Figura 3.12 na forma DNF:

- *Flow Direction* = "*In*"
- *Source IP* = 9.1.1.1
- *Source Port* = 80

Desconsiderando-se o valor de *IsMirrored*, os pacotes de entrada serão validados por uma *CompoundFilterCondition* se *Source IP address* = 9.1.1.1 e *Source port* = 80. Entretanto, se o valor de *IsMirrored* for *TRUE*, um pacote de saída também será validado se *Destination IP address* = 9.1.1.1 e *Destination port* = 80.

IsMirrored executa uma "inversão" dos seguintes campos de cabeçalho:

- direção dos pacotes "*In*" / direção dos pacotes "*Out*"
- endereço IP de origem / endereço IP de destino

- porta de origem / porta de destino
- endereço MAC de origem / endereço MAC de destino
- SAP de origem [*layer-2*] / SAP de destino [*layer-2*]

3.7. O Mapeamento do PCIMe no LDAP

O modelo de representação definido pelo PCIMe não especifica em nenhum momento a sua forma de implementação deixando aos desenvolvedores a tarefa de especificar o formato de armazenamento e recuperação dos dados mais adequado. Dentre os trabalhos que estão sendo feitos destaca-se o *Policy Core LDAP Schema* – PCLS [MOO02]. Como o próprio nome sugere, o PCLS é uma iniciativa no sentido de se definir um mapeamento do PCIM de maneira que este possa ser implementado em um serviço de diretório baseado em LDAP.

As especificações do PCLS apenas definem o mapeamento das classes estruturais e associativas do PCIM [RFC3060] no LDAP. Assim, a partir das alterações e novas classes introduzidas pelo PCIMe [RFC3460], o PCLS também precisou acomodar os novos conceitos e alterações, passando a ser denominado *Policy Core Extensions LDAP Schema* – PCELS [REY03]. Este trabalho segue, ainda, como um grupo de desenvolvimento conjunto formado pelo DMTF e IETF.

Esta seção demonstra a proposta de mapeamento oferecida pelo PCELS para implementar o PCIMe no serviço LDAP.

3.7.1. Considerações sobre o Mapeamento de Classes no LDAP

O mapeamento de classes em um repositório LDAP é feito a partir de algumas regras. Neste sentido, a [RFC2252] define que uma classe em um serviço de diretório baseado no protocolo LDAP pode ser de três tipos: 1) abstrata, 2) estrutural e 3) auxiliar.

Para estes três tipos de classes, a [RFC2252] considera as seguintes regras:

- um objeto, em um serviço de diretório, não pode ser instanciado a partir de uma classe abstrata;
- classes estruturais são os únicos tipos de classe que podem ter instâncias em um diretório;

- classes auxiliares definem um conjunto de atributos que pode ser adicionado a objetos instanciados no diretório[7];
- uma classe abstrata pode ser uma subclasse apenas de outra classe abstrata;
- uma classe estrutural pode ser uma subclasse de uma classe abstrata ou de outra classe estrutural;
- um objeto em um serviço de diretório não pode ser instanciado a partir de uma classe auxiliar;
- classes auxiliares podem ser subclasses de classes abstratas ou de outras classes auxiliares.

3.7.2. O Mapeamento do PCIMe no LDAP

Como exposto na Seção 3.4, o *framework* proposto pelo PCIM define duas hierarquias de classes: 1) classes estruturais para representar as informações e controle das políticas e 2) classes de relacionamento, que indicam como as instâncias das classes estruturais estão relacionadas umas com as outras. Em ambos os casos, as hierarquias de classes precisam ser mapeadas para um tipo particular de armazenamento [MOO02].

O mapeamento das classes estruturais do PCIM (e do PCIMe) não é feito, na base de um para um: classes do modelo de informação são mapeadas por classes LDAP e as propriedades das classes do modelo de informação são mapeadas por atributos das classes LDAP. [DMT02] apresenta as seguintes estratégias de mapeamento LDAP:

- classes abstratas dos modelos de informação são mapeadas através de classes abstratas no modelo de objetos do LDAP;
- classes estruturais dos modelos de informação são mapeadas através de um conjunto composto por três classes LDAP:
  - uma classe abstrata, a qual define a hierarquia de classes LDAP equivalente à hierarquia dos modelos de informação;
  - uma classe estrutural do LDAP;

[7] Através da utilização das classes auxiliares é possível adicionar atributos a um objeto do diretório sem que a classe utilizada para os instanciar contenha estes atributos em sua definição.

- uma classe auxiliar, a qual permite que informações definidas pelos modelos sejam anexadas[8] a objetos pré-existentes no diretório.

A Tabela 3.3 apresenta o mapeamento entre as principais classes abstratas e estruturais do PCIMe no LDAP.

Tabela 3.3. Mapeamento entre as principais classes do PCIMe no LDAP.

PCIM / PCIMe	**Classe LDAP**
Policy	*pcimPolicy (ab)*
PolicySet	*pcimPolicySet (ab)*
PolicyGroup	*pcimPolicyGroup (ab)* *pcimPolicyGroupAuxClass (aux)* *pcimPolicyGroupInstance (struc)*
PolicyRule	*pcimPolicyRule (ab)* *pcimPolicyRuleAuxClass (aux)* *pcimPolicyRuleInstance (struc)*
PolicyCondition	*pcimConditionAuxClass (aux)*
PolicyAction	*pcimActionAuxClass (aux)*
PolicyTimePeriodCondition	*pcimTPCAuxClass (aux)*
SimplePolicyCondition	*pcimSimpleConditionAuxClass (aux)*
CompoundPolicyCondition	*pcimCompoundConditionAuxClass (aux)*
CompoundFilterCondition	*pcimCompoundFilterAuxClass (aux)*
SimplePolicyAction	*pcimSimpleActionAuxClass (aux)*
CompoundPolicyAction	*pcimCompoundActionAuxClass (aux)*
PolicyVariable	*pcimVariable (ab)*
PolicyExplicitVariable	*pcimExplicitVariableAuxClass (aux)*
PolicyImplicitVariable[9]	*pcimImplicitVariableAuxClass (aux)*
PolicyValue[10]	*pcimValueAuxClass (aux)*
PolicyRoleCollection	*pcimRoleCollection (struc)*
ReusablePolicyContainer	*pcimReusableContainer (ab)*

[8] O termo “anexada” será usado neste trabalho para indicar que os atributos definidos por uma classe auxiliar serão adicionados a uma instância (objeto) do diretório. Quando se diz que uma classe auxiliar será anexada a uma classe estrutural significa dizer que objetos desta classe estrutural poderão utilizar os atributos definidos por esta classe auxiliar [NAB03].

[9] As especializações de *PolicyImplictyVariable* também são mapeadas por classes auxiliares mas foram suprimidas por não agregarem nenhuma informação vital ao entendimento do mapeamento. Podem ser conferidas no Anexo A.

[10] Idem para as especializações de *PolicyValue*.

	pcimReusableContainerAuxClass (aux) *pcimReusableContainerInstance (struc)*

Para o mapeamento das associações, o PCLS define que pode ser de três formas:

1) através de classes LDAP auxiliares;

2) atributos representando referências DN[11] (*Distinguished Name*);

3) relacionamento superior-subordinado inerentes à DIT (*Directory Information Tree*) do diretório ou, simplesmente *DIT containement*[12].

A Tabela 3.4 apresenta o mapeamento entre o modelo de associação e o LDAP:

Tabela 3.4 – Mapeamento entre o modelo de associação e o LDAP.

Associação	**Atributo / Classe LDAP**
PolicySetComponent	*pcimPolicySetComponentList em pcimPolicySet e pcimPolicySetDN em pcimPolicySetAssociation*
PolicySetInSystem	*DIT Containment e* *pcimPolicySetDN em pcimPolicySetAssociation*
PolicyGroupInSystem	*DIT Containment e* *pcimPolicySetDN em pcimPolicySetAssociation*
PolicyRuleInSystem	*DIT Containment e* *pcimPolicySetDN em pcimPolicySetAssociation*
PolicyConditionStructure	*pcimConditionDN em pcimConditionAssociation*
PolicyConditionInPolicyRule	*pcimConditionList em pcimPolicyRule e pcimConditionDN em pcimConditionAssociation*
PolicyConditionInPolicyCondition	*pcimConditionList em pcimCompoundConditionAuxClass e* *pcimConditionDN em pcimConditionAssociation*
PolicyActionStructure	*pcimActionDN em pcimActionAssociation*
PolicyActionInPolicyRule	*pcimActionList em pcimPolicyRule e* *pcimActionDN em pcimActionAssociation*
PolicyActionInPolicyAction	*pcimActionList em pcimCompoundActionAuxClass e pcimActionDN em pcimActionAssociation*

[11] O DN (*Distinguished Name*) de um objeto é o identificador único de uma entrada no diretório. Além de identificar a entrada, ele também indica a localização desta entrada no diretório.

[12] Uma das características de um diretório é que um objeto pode conter outros objetos, não importando o nível desta associação. Desta forma, fica caracterizado o conceito de *DIT containement*.

PolicyVariableInSimplePolicyCondition	*pcimVariableDN em pcimSimpleConditionAuxClass*
PolicyValueInSimplePolicyCondition	*pcimValueDN em pcimSimpleConditionAuxClass*
PolicyVariableInSimplePolicyAction	*pcimVariableDN em pcimSimpleActionAuxClass*
PolicyValueInSimplePolicyAction	*pcimValueDN em pcimSimpleActionAuxClass*
ReusablePolicy	*DIT containment*
ExpectedPolicyValuesForVariable	*pcimExpectedValueList em pcimVariable*
ContainedDomain	*DIT containment ou pcimReusableContainerList em pcimReusableContainer*
EntriesInFilterList	*pcimFilterEntryList em pcimFilterListAuxClass*
ElementInPolicyRoleCollection	*DIT containment ou pcimElementList em pcimRoleCollection*
PolicyRoleCollectionInSystem	*DIT Containment*

A classe *Policy* é mapeada pela classe abstrata *pcimPolicy* no modelo de objetos do LDAP. Ela atua como classe base da hierarquia de classes que definem objetos relacionados a políticas. A classe *PolicySet*, definida no PCIMe como uma abstração para um conjunto de grupos ou regras, é mapeada pela classe *pcimPolicySet*. A classe *PolicyGroup* é mapeada por três classes. Esta estratégia de mapeamento dá flexibilidade à utilização do modelo, uma vez que possibilita duas alternativas para a especificação de containeres *PolicyGroup*:

1) anexando a classe *pcimPolicyGroupAuxClass* a objetos já existentes no diretório;

2) criação de novos objetos inseridos a partir da classe estrutural *pcimPolicyGroupInstance*.

Estas duas classes são especializações da classe abstrata *pcimPolicyGroup*. De maneira similar, a classe *PolicyRule* do PCIM também é mapeada por três classes:

1) uma classe abstrata chamada *pcimPolicyRule*, a qual define os atributos dos objetos *PolicyRule*;

2) duas especializações criadas com o mesmo propósito daquelas da classe *pcimPolicyGroup*: *pcimPolicyRuleAuxClass* e *pcimPolicyRuleInstance*.

A Figura 3.13 demonstra essa hierarquia:

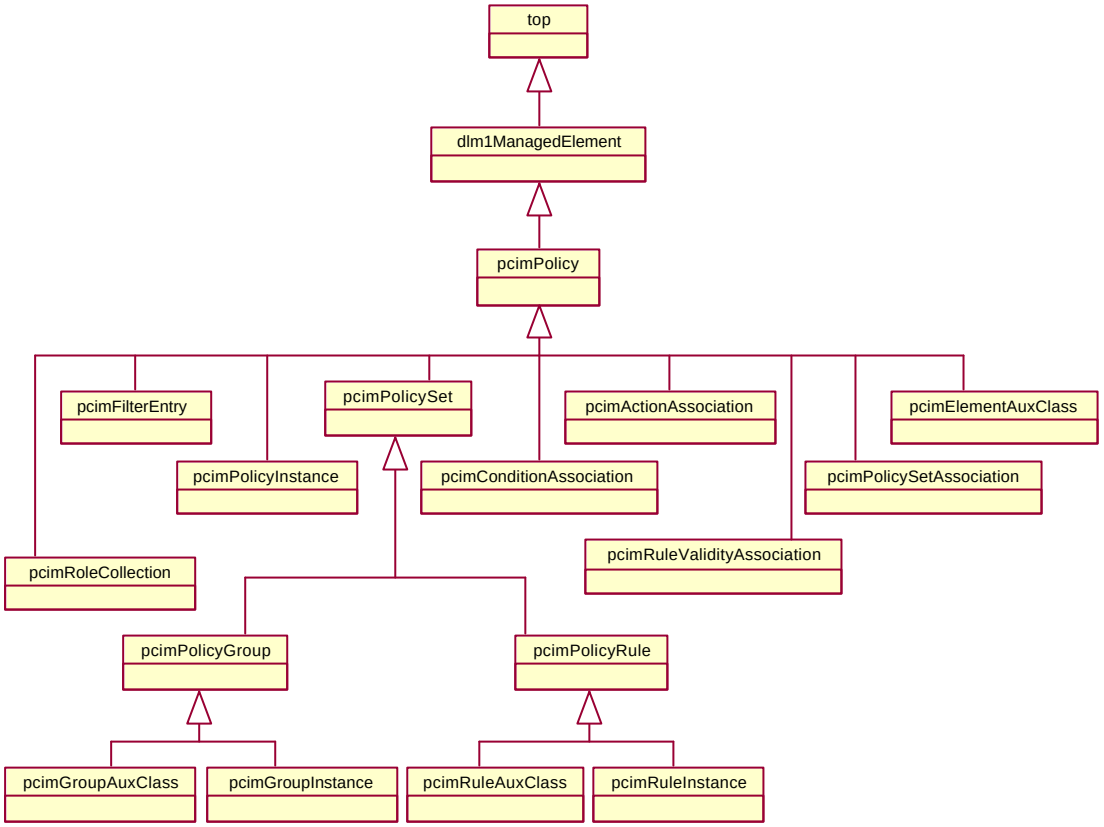

Figura 3.13 – Hierarquia de classes do PCELS (Parte 1)

Como demonstrado na Seção 3.5, as agregações *PolicyGroupInPolicyGroup* e *PolicyRuleInPolicyGroup* foram combinadas em uma única agregação denominada *PolicySetComponent*. Esta agregação e sua capacidade de associação entre uma política e a classe *ReusablePolicyContainer* oferecem muitas novas possibilidades de reutilização de regras. A agregação *PolicySetComponent* é mapeada pelo atributo *multi-value pcimPolicySetList* da classe *pcimPolicySet* e pelo atributo *pcimPolicySetDN* na classe *pcimPolicySetAssociation*. Estes atributos referem-se a regras e grupos aninhados.

As classes *PolicyCondition*, *PolicyAction* e *PolicyTimePeriodCondition* são mapeadas por três classes auxiliares do LDAP, respectivamente, *pcimConditionAuxClass*, *pcimActionAuxClass* e *pcimTPCAuxClass*. A Figura 3.14 demonstra essa hierarquia.

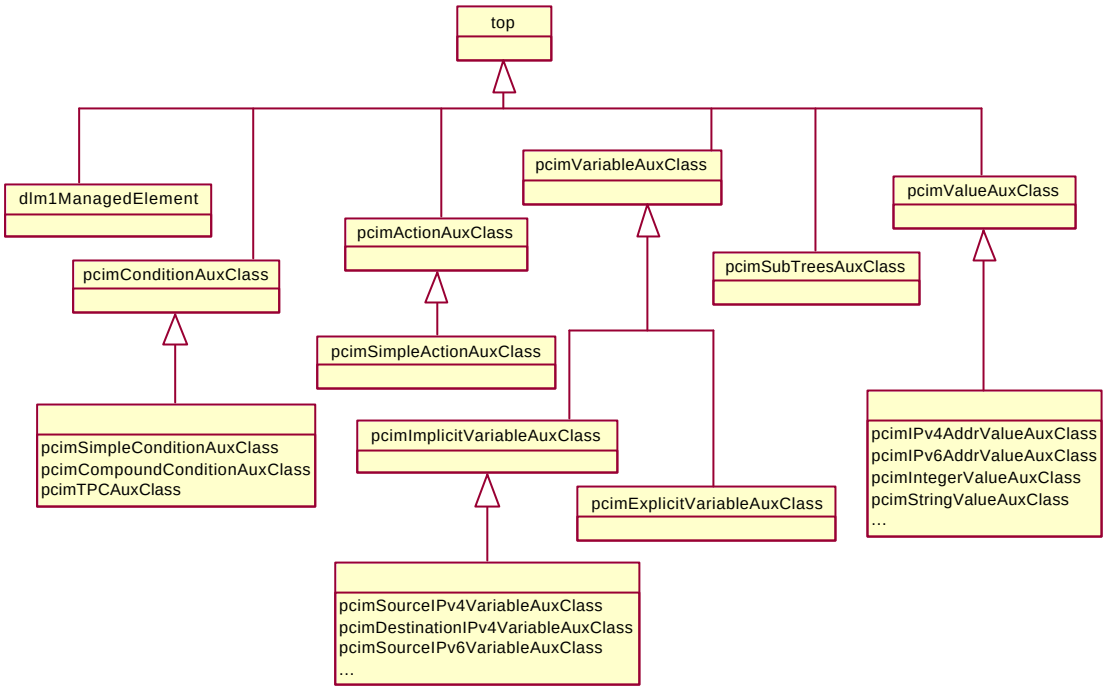

Figura 3.14 – Hierarquia de classes do PCELS (Parte 2)

As classes *pcimConditionAuxClass* e *pcimActionAuxClass* não representam as "condições" e "ações" reais. Estas são introduzidas por suas subclasses. O que *pcimConditionAuxClass* e *pcimActionAuxClass* introduzem são as semânticas de condições de política associadas a ações, que serão herdadas por todas as suas subclasses. Dentre essas semânticas está a que representa a utilização de condições/ações específicas e a que representa condições/ações reusáveis. Neste sentido, [REY03] propõe classes que possibilitam o mapeamento LDAP destas associações para os dois casos. Como visto na seção 3.5, o PCIMe introduz as classes *CompoundPolicyCondition* e *CompoundPolicyAction* como especializações de *PolicyCondition* e *PolicyAction*, respectivamente. As condições/ações compostas (condições/ações que são formadas pela composição de condições/ações simples reunidas na forma DNF/CNF) definidas no PCIMe estendem a capacidade de uma regra para associar, agrupar e avaliar/executar condições/ações[13].

A partir da Tabela 3.3, a classe *CompoundPolicyCondition* será mapeada por *pcimCompoundConditionAuxClass*, enquanto a classe *CompoundPolicyAction* será mapeada por *pcimCompoundActionAuxClass*.

[13] Para que se evite confusões, condições/ações agregadas por CompoundPolicyConditions serão designadas pelo termo "composições".

Como regra geral de armazenamento, condições/ações específicas são subordinadas (via DIT) a uma regra ou a uma composição que as agrega e são anexadas às instâncias das classes de associação. Condições/Ações reusáveis são subordinadas a instâncias da classe *pcimReusableContainerInstance* e anexadas a instâncias da classe *pcimPolicyInstance*. A hierarquia de classes para o armazenamento de condições/ações reusáveis é apresentado na Figura 3.15.

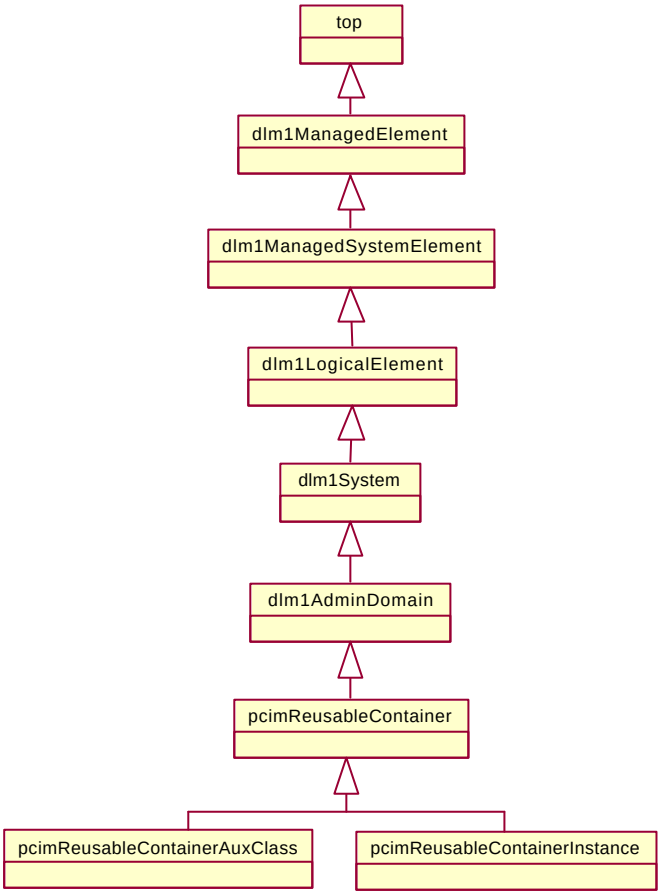

Figura 3.15 – Hierarquia de classes do PCELS (Parte 3)

[REY03] demonstra as quatro possibilidades de construção de condições/ações específicas/reusáveis, simples/ composições:

Primeiro Caso: composição específica formada por condições/ações específicas.

Como as condições/ações pertencem a uma regra específica, elas devem ser anexadas como classes auxiliares às instâncias de *pcimConditionAssociation* ou *pcimActionAssociation*. Estas classes estruturais representam a associação entre a uma regra e as composições de condições/ações. As condições/ações específicas são, portanto, subordinadas (via DIT) à regra.

As condições/ações específicas são agregadas às composições da mesma forma que as composições são agregadas pelas regras, conforme a Figura 3.16. As classes associativas implementam a associação entre as composições e suas condições/ações específicas.

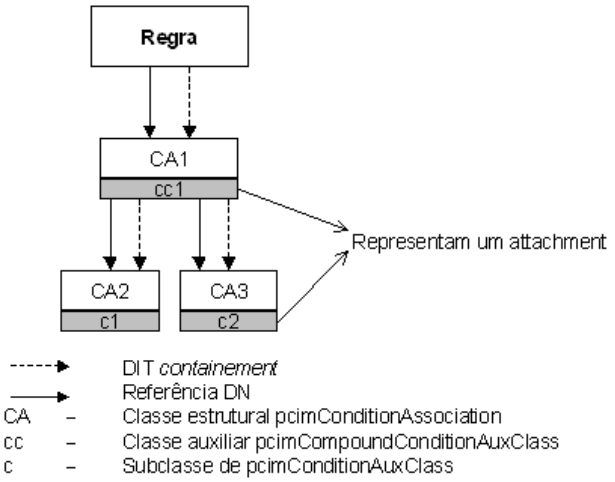

Figura 3.16 – Composição específica formada por condições/ações específicas

Segundo Caso: Composição específica formada por condições/ações reusáveis.

Este caso é similar ao primeiro. As condições/ações são reusáveis, portanto, não podem ser anexadas às classes associativas, mas são anexadas a classes estruturais no container reusável. As classes associativas são referenciadas por DN às condições/ações localizadas no container, conforme a Figura 3.17.

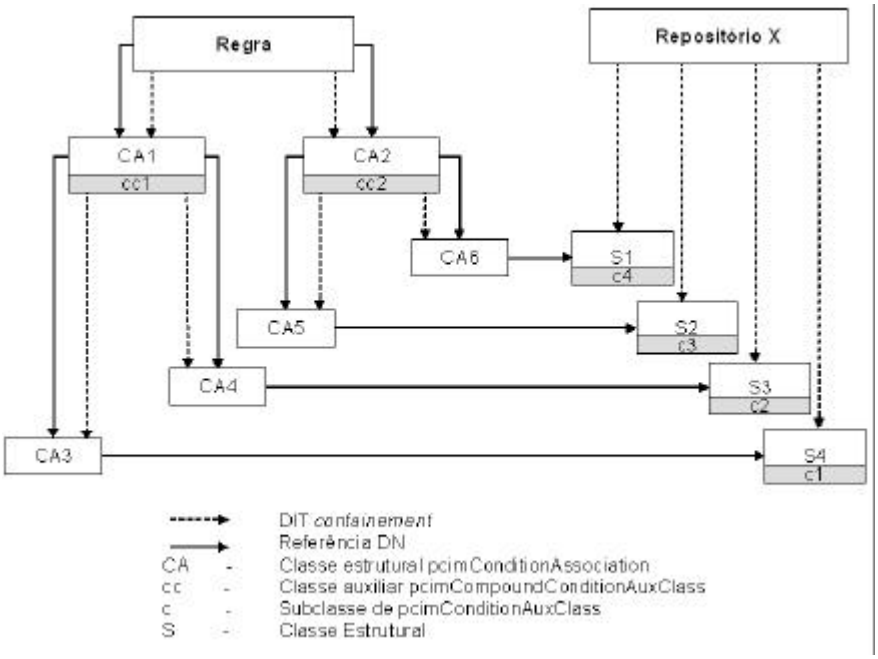

Figura 3.17 – Composição específica formada por condições/ações reusáveis

Terceiro Caso: Composição reusável formada por condições/ações específicas.

Composições reusáveis são anexadas em classes estruturais e armazenadas em um container de políticas reusáveis. Essas composições são associadas às regras através de referências DN nas classes de associação. As condições/ações específicas são anexadas às instâncias das classes de associação e subordinadas (via DIT) às composições de condições/ações, conforme a Figura 3.18.

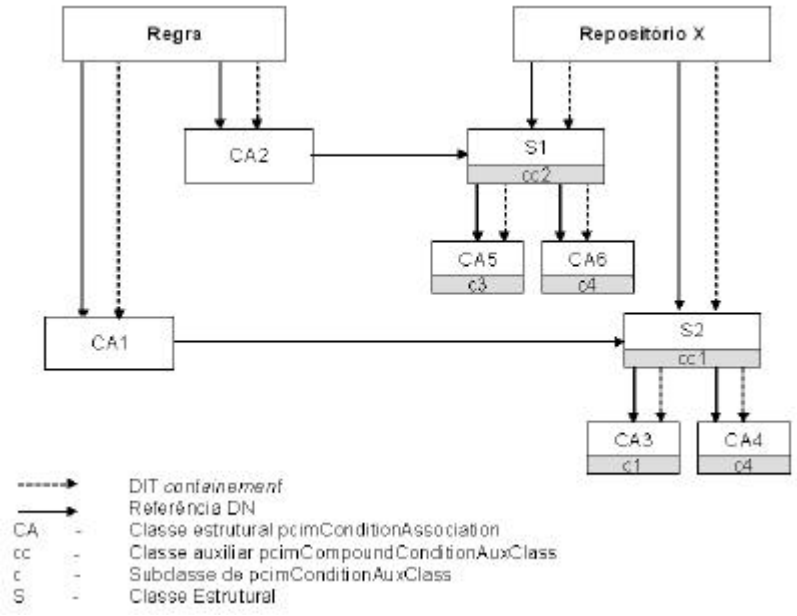

Figura 3.18 – Composição reusável formada por condições/ações específicas

Quarto Caso: Composição reusável formada por condições/ações reusáveis.

Todas as condições/ações são reusáveis. Portanto, todas são armazenadas em containeres reusáveis. A Figura 3.19 ilustra dois containeres de políticas reusáveis diferentes (Repositório A e B), mas o número de containeres do sistema é definido por questões administrativas. As condições/ações (cc1 e cc2) são reusáveis, portanto, não podem ser anexadas às classes associativas (CA1 e CA2), mas são anexadas a classes estruturais (S1 e S2) no container reusável (Repositório A). As classes associativas (CA1 e CA2) são referenciadas por DN às condições/ações localizadas no container reusável (Repositório A).

Como as condições/ações (c1, c2, c3 e c4) que formam as composições reusáveis (cc1 e cc2) também são reusáveis e pertencem a outro container (Repositório B), essas condições/ações (c1, c2, c3 e c4) devem ser anexadas a classes estruturais (S3, S4, S5 e S6) do Repositório B para serem referenciadas pelas condições/ações reusáveis do Repositório A (CA6, CA5, CA4 e CA3) por referência DN.

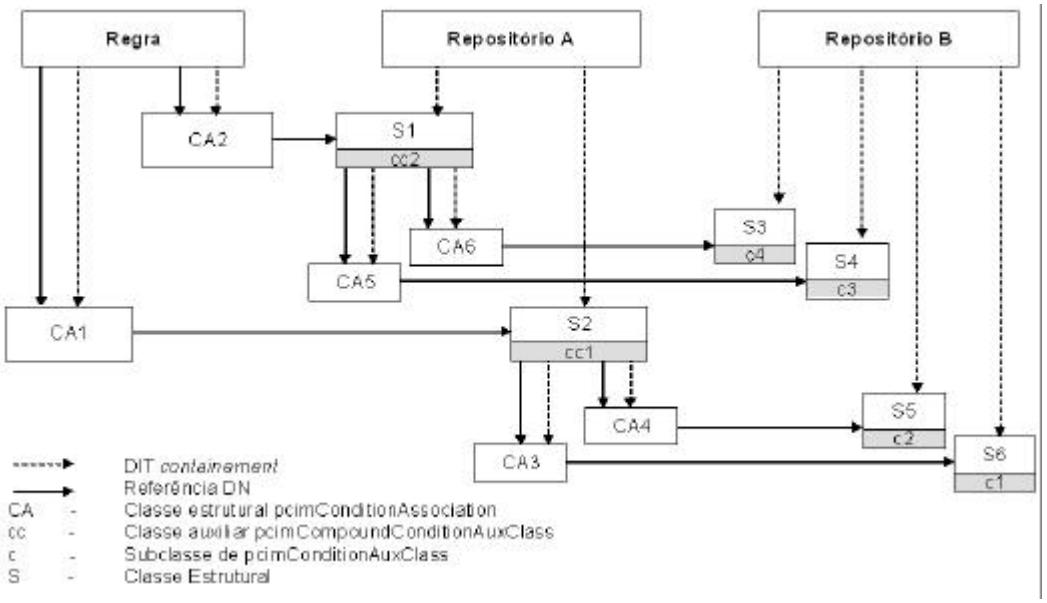

Figura 3.19 – Composição reusável formada por condições/ações reusáveis

3.8. O Protocolo COPS-PR

3.8.1. Introdução

O considerável crescimento das redes de computadores revelou as limitações nas técnicas tradicionais de gerenciamento no que tange eficiência e escalabilidade. Então, o paradigma de gerenciamento de redes baseado em políticas surgiu como um conceito promissor na área de gerenciamento de redes de computadores [SHE00]. Para tanto, é preciso que as políticas descrevam o comportamento desejado em uma linguagem de alto nível, independentemente dos dispositivos e topologia da rede [BAR99] [POL02].

Todos esses conceitos, detalhados nas seções anteriores, introduzem a idéia de como uma rede baseada em políticas pode ser utilizada para centralizar o gerenciamento de segurança e alocação de recursos. Entretanto, a descrição e armazenamento das políticas não são suficientes. É necessário que haja um protocolo de comunicação que implemente a interface entre os dispositivos de segurança e os servidores de políticas [SAL02]. O protocolo COPS – *Common Open Policy Service* – surge, então, como um padrão do IETF para descrição de um modelo de comunicação para troca de mensagens sobre informações de políticas entre um servidor de políticas, (PDP), responsável pela tomada das decisões e um cliente de políticas (PEP), responsável por solicitar e implementar essas decisões [FEN02]. Esta seção demonstra, brevemente, a arquitetura oferecida pelo protocolo COPS.

3.1.1. Padrões de Implementação

A [RFC2753] define três arquiteturas possíveis para a implementação do controle baseado em política. Essas três possibilidades de implementação têm em comum, a definição de duas entidades principais: o PDP – *Policy Decision Point* – e o PEP – *Policy Enforcement Point*. As Figuras 3.20 a 3.22 ilustram a estratégia de implementação de cada uma das três arquiteturas.

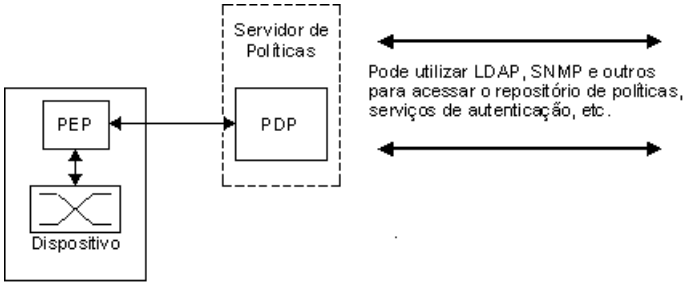

Figura 3.20 – Arquitetura do controle baseado em políticas – Arquitetura 1

Para a arquitetura apresentada na Figura 3.20, o PEP é um componente presente em um nó da rede, enquanto o PDP é uma entidade remota que fornece um serviço baseado em decisões de política [RFC2753]. O PEP interage com as entidades da rede e tem como responsabilidade garantir que uma decisão de política seja cumprida. O PDP é o responsável por processar as políticas junto com outros dados tais como informações sobre o estado da rede e gerar e enviar os dados de configuração ao PEP para que sejam implementadas. Ainda, o PDP precisa utilizar mecanismos adicionais para executar suas funções. Entre eles estão protocolos e serviços de autenticação para acesso ao repositório de informações LDAP.

As Figuras 3.21 e 3.22 são variações da arquitetura indicada pela Figura 3.20. Na Figura 3.21, o PDP também se apresenta como um nó da rede. Esta variação é indicada no caso de políticas que dependem fortemente da informação e do estado de uma entidade em particular na rede, situação caracterizada pelo aspecto dinâmico das informações [RFC2753].

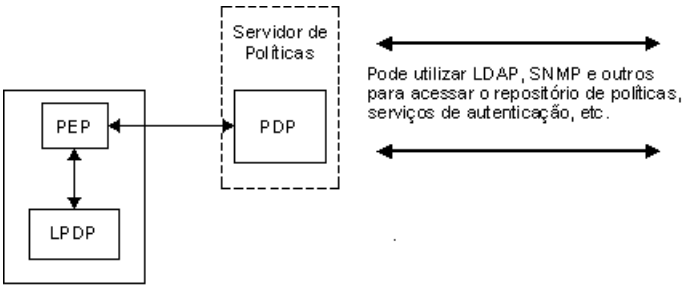

Figura 3.21 – Arquitetura do controle baseado em políticas – Arquitetura 2

No cenário da Figura 3.21, a partir de um evento onde o PEP precisa tomar uma decisão baseada em política, ele contata seu LPDP – *Local Policy Decision Point* – e obtém um resultado parcial. Este resultado parcial de política, mais a requisição original são encaminhados ao PDP para a tomada de decisão. Por fim, o PDP toma a decisão final e a retorna ao PEP. Nesta situação, o LPDP e o PDP trabalham cooperativamente, onde o primeiro atua de forma subordinada ao segundo [RFC2753].

Na Figura 3.22, o PDP se apresenta inteiramente no próprio nó da rede, não havendo a necessidade do PEP entrar em contato com um servidor de políticas remoto. Este tipo de implementação é eficiente quando as políticas são dependentes de informações e condições locais, referentes a um roteador em particular ou quando a alteração nas políticas se dá de forma muito dinâmica [RFC2753].

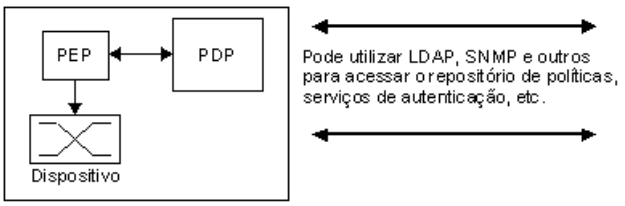

Figura 3.22 – Arquitetura do controle baseado em políticas – Arquitetura 3

3.8.2. Interações entre o PDP e PEP

Um aspecto vital na arquitetura mostrada na seção anterior é a interação entre o PDP e o PEP. No sentido de estabelecer um padrão para esta comunicação, o *IETF Networking Group* definiu um protocolo específico, o COPS – *Common Open Policy Service*, proposto pela [RFC2748]. Este protocolo emprega um modelo cliente/servidor onde o PEP envia requisições de políticas ao PDP e este retorna as decisões relacionadas de volta ao PEP. Com intuito de estabelecer a confiabilidade na troca de mensagens, o COPS utiliza o TCP como protocolo de transporte. Conforme padronização do IANA, entidades PDP devem aguardar pedidos de conexões na porta 3288, ficando o PEP responsável por iniciar a conexão TCP. O PEP é o responsável, também, por manter, permanentemente, a conexão TCP com o PDP.

O COPS é um protocolo *query/response* que suporta dois modelos para o controle baseado em políticas: o modelo *outsourcing* e o modelo *provisioning*.

O modelo *outsourcing* está relacionado a um cenário onde ocorrem eventos em que o PEP precisa de uma decisão de política instantânea, no qual ele envia uma requisição ao PDP e aguarda a decisão de política.

No modelo *provisioning* (ou *configuration*) não há correlação direta entre os eventos do PEP e tomada de decisões do PDP. Mesmo assim, o PDP também pode agir de maneira pró-ativa, fornecendo ao PEP as informações de política sem que este as tenha solicitado [RFC3084].

Essas informações de políticas são distribuídas por toda a rede em tempo real e é o PDP que determina se uma determinada regra deve ou não ser implementada por um PEP qualquer, baseado em vários critérios, tais como restrições de tempo e funcionalidades do dispositivo. Se um determinado critério for satisfeito, o PDP traduz as regras de políticas para um formato que possa ser entendido pelos dispositivos por ele controlados. Neste caso, o formato pode ser baseado em mensagens padrão COPS ou SNMP[14]. O COPS é utilizado como protocolo de comunicação entre o PDP e o PEP enquanto o SNMP é utilizado por elementos mais específicos tais como servidores de autenticação.

A Figura 3.23 oferece uma visão bastante abrangente da arquitetura de implementação do protocolo COPS e as entidades PEP e PDP.

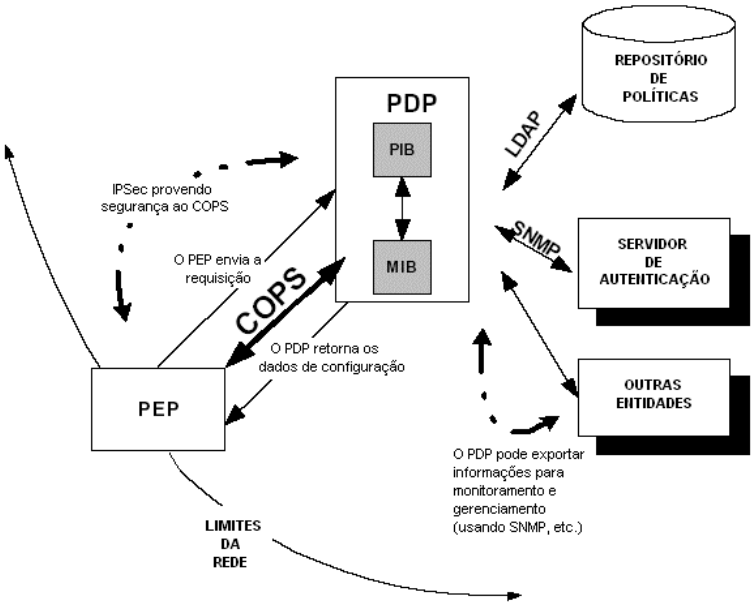

Figura 3.23 – Arquitetura COPS

O COPS fornece um esquema de segurança para autenticação, proteção contra *replays* e integridade, mas também pode usar outros protocolos para segurança tais como o *IPSec* ou *TLS* para autenticar e estabelecer um canal de comunicação seguro entre o PEP e o PDP [RFC2748].

[14] Apesar do SNMP ter evoluído muito desde a sua primeira versão, este protocolo ainda é incapaz de suportar com eficiência o gerenciamento de redes baseado em políticas devido à sua inabilidade de suportar dispositivos clientes iniciando a comunicação com seus servidores. Além disso, o SNMP oferece apenas a visão de um dispositivo local, enquanto o COPS oferece uma visão global e pró-ativa para o gerenciamento da rede.

3.8.3. O Modelo de Provisionamento – Protocolo COPS-PR

O protocolo COPS-PR é uma extensão (ou um tipo de cliente) do COPS desenvolvida pelo grupo de trabalho IETF RAP – *Resource Allocation Protocol*. Inicialmente, o COPS-PR foi planejado para efetuar o controle de políticas relacionadas a eventos não sinalizados tais como serviços de diferenciação (DiffServ) [XIA99] [FIN01]. Entretanto, este protocolo tem obtido sucesso em muitas outras áreas de gerenciamento: configuração de dispositivos de VPN, serviços de voz sobre IP (VoIP) [WON01] e segurança [OTT00]. Este trabalho pretende utilizar este modelo para a configuração de *firewalls* baseados em filtros de pacotes.

Como o seu nome sugere, o COPS-PR trabalha em modo de provisionamento. Os clientes (PEPs) se conectam ao PDP, reportam suas capacidades e limitações e requisitam as políticas que devem ser implementadas. O PDP processa a requisição de cada cliente e, de acordo com as informações globais de política e estado atual da rede, envia os dados de configuração de volta aos PEPs, os quais tratam cada novo evento de acordo com os dados de política recebidos do PDP. Caso haja uma alteração no estado da rede ou uma alteração na definição das políticas, o PDP deve atualizar todos os dados de configuração localizados nos PEPs que tenham alguma relação com as alterações que tenham ocorrido [RFC3084]. Assim é preservada a consistência no comportamento de todos os dispositivos envolvidos.

3.8.4. Policy Information Base – PIB

Nas arquiteturas que utilizam o COPS-PR, cada cliente deve manter um banco de dados especial, onde serão armazenados todos os dados de configuração recebidos do PDP. Este banco de dados recebe o nome de PIB – *Policy Information Base*. A PIB é uma estrutura similar a uma MIB – *Management Information Base* – e pode ser descrita como um espaço de nomes em forma de árvore onde os galhos representam as estruturas de dados ou classes de provisionamento (PRCs) e as folhas representam as instâncias dessas classes, chamadas instâncias de provisionamento (PRIs) [POL02], conforme a Figura 3.24.

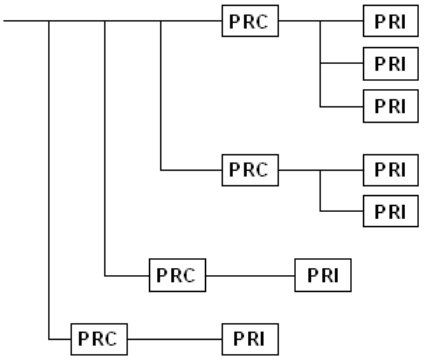

Figura 3.24 – Estrutura da PIB

As PIBs são abstrações de alto nível que tornam as informações de hardware transparentes ao PDP e oferecem um meio de se unificar o controle do comportamento dos dispositivos por toda a rede. As PIBs atuam como “pontes” entre um modelo de dispositivo e sua implementação em forma de produto específico de um fabricante. Dessa forma, a estrutura de cada PIB deve ser especificada segundo suas diferentes áreas de atuação e gerenciamento (por exemplo, DiffServ, VPN, VoIP, segurança). As políticas definidas em uma linguagem de alto nível armazenadas no repositório são traduzidas pelo PDP para parâmetros de baixo nível representados nas PIBs os quais podem ser entendidos pelos PEPs.

Novas regras podem ser implementadas apenas estendendo as classes da PIB, uma vez que esta mantém, associado a cada regra de política, um identificador único PRID – *Policy Rule Identifier*. Assim, ao extrair o PRID das mensagens COPS-PR, o PEP tem condições de decodificar e processar todas as informações referentes a uma regra de política, conforme a Figura 3.25.

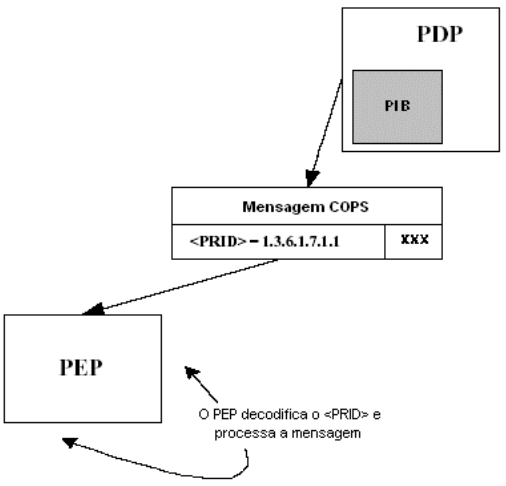

Figura 3.25 – O PEP Decodifica e Processa o PRID

SPPI – *Structure of Policy Provisioning Information* – é a linguagem utilizada para definir uma PIB [RFC3159]. Esta linguagem especifica a informação de política que pode ser transmitida para um dispositivo da rede a fim de se configurar uma política. SPPI utiliza codificação ASN.1 e é similar à estrutura da linguagem SMI – *Structure of Management Information* – para gerenciamento de informações SNMP. Conseqüentemente, a PIB tem uma estrutura hierárquica similar a uma MIB. Além disso, os identificadores de objetos (OIDs) utilizados para identificar objetos localizados na estrutura hierárquica da MIB podem ser reutilizados pelas PIBs. Estas características de similaridade simplificam a comunicação entre o PDP e o PEP uma vez que permitem que uma PIB converta uma MIB localizada em um PEP em uma MIB localizada no PDP. Assim, é possível ao PEP enviar os seus papéis ao PDP para descrever suas interfaces e o PDP retornará toda a PIB de volta ao PEP.

3.9. Conclusões do Capítulo

O paradigma do gerenciamento baseado em política vem ganhando força junto a comunidade científica, concentrando-se nas áreas de autorização de acesso a recursos dentro de uma rede e na área da qualidade de serviço. As políticas podem ser vistas como um conjunto de regras para administrar, gerenciar e controlar o acesso a recursos. Através da popularização dos serviços de diretórios, o DEN e o CIM são iniciativas no sentido de se padronizar o armazenamento das políticas em repositórios abertos (no caso LDAP).

Ao atrair o interesse do IETF, o modelo de informações de políticas proposto pelo DMTF recebeu um refinamento, o que deu origem ao PCIM e, posteriormente, ao PCIMe. O PCIMe vem rapidamente se tornando o padrão para o gerenciamento de redes baseado em políticas. A parceria entre o DMTF e o IETF também propiciou o desenvolvimento de um esquema padrão para a representação das classes do PCIMe no LDAP – o PCELS. Finalmente, para efetuar a distribuição das políticas entre os vários elementos de segurança da rede, o IETF propõe o protocolo COPS e suas duas abordagens: o modelo *outsourcing* e o modelo *provisioning*. Cada uma dessas abordagens tem a sua aplicação específica relacionada à natureza de controle de acesso. Modelos em que as políticas de segurança têm um aspecto dinâmico tais como QoS e RBAC [NAB03] utilizam a abordagem *outsourcing*, enquanto que modelos em que as políticas de segurança têm um aspecto estático devem utilizar a abordagem *provisioning*.

Capítulo 4

PFIM: Proposta de Extensão do PCIMe para Configuração de Filtros de Pacotes

4.1. Introdução

O modelo introduzido pelo PCIM [RFC3060] estabeleceu um padrão para representar e gerenciar informações de políticas. O modelo de classes e associações produzido pelo PCIM fornece a base necessária à representação de políticas de qualquer natureza. A partir deste conceito, o PCIM deu origem a diversos grupos de estudos que demonstraram a sua capacidade de ser estendido para diversas áreas a partir do modelo original. Conforme exposto no Capítulo 3, dentre esses trabalhos, destacam-se, na área de QoS, o QPIM [SNI01] e o QDDIM [MOO01], na área de IPSec, o ICPM [JAS01], o e o RBPIM [NAB03] para controle de acesso.

Todos esses trabalhos, de alguma forma, contribuíram para que o padrão proposto pela RFC3060 recebesse novas classes para torná-lo ainda mais genérico: o PCIMe [RFC3460].

Neste contexto, será apresentado o PFIM – *Packet Filtering Information Model* – uma proposta de extensão do PCIMe capaz de suportar a descrição e o gerenciamento das regras de filtros de pacotes definidas através de políticas de segurança.

Este capítulo também demonstra o mapeamento deste modelo em serviços de diretório baseados no LDAP. Para tanto, será utilizado o esquema de diretório definido pelo PCELS [REY03] estendendo-se esse esquema para acomodar as novas classes propostas pelo PFIM.

4.2. A Proposta PFIM

A presente seção detalha o modelo de informação proposto. Este modelo propõe a utilização do *framework* do PCIMe para o mapeamento das regras de *firewall*, especificamente, filtros de pacotes. Para tal, esta proposta utiliza os conceitos e especificações apresentados nos Capítulos 2 e 3, propondo uma extensão ao PCIMe e PCELS a partir da estratégia adotada pelo Firmato [BAR99]. Esta proposta inova por ser uma alternativa para a descrição, o gerenciamento e a implementação das políticas nos dispositivos de segurança em conformidade com objetivos principais abordados no Firmato:

- separar a definição das políticas de segurança da sua implementação;
- tornar a definição das políticas independente da topologia da rede;
- gerar arquivos de configuração de filtros de pacotes automaticamente a partir da política de segurança.

4.2.1. Considerações e Estratégias

O Capítulo 2 apresentou uma breve descrição do funcionamento de um *firewall* baseado em filtros de pacotes e padrões comerciais e acadêmicos para sua implementação. O Capítulo 3 apresentou os padrões definidos pelo IETF e DMTF para a representação das informações de políticas necessárias ao gerenciamento de sistemas computacionais, tais como a informação sobre configurações, sistemas, redes, serviços, aplicações, dispositivos e componentes que podem ser gerenciados. A especificação X.500 [CCI88] define uma série de padrões que especificam como a informação pode ser armazenada e acessada em um diretório. As implementações do serviço de diretório LDAP mantêm as regras de nomes e o modelo de informação do X.500.

A estratégia adotada para a concepção deste modelo visa o aproveitamento das padronizações disponibilizadas pelo CIM, PCIM e PCIMe. Essa estratégia é importante, pois é muito provável que os serviços de diretório comerciais adotem a proposta do DMTF, passando a fornecer as classes do CIM de forma pré-definida. A compatibilidade com o CIM permitirá reduzir bastante o número de informações a serem adicionadas ao diretório, a fim de suportar esta proposta [NAB03]. Assim, as organizações que utilizam serviços de diretório LDAP para armazenar seus objetos, definidos a partir dos esquemas definidos pelo CIM, serão beneficiadas. Estas organizações poderão incorporar o armazenamento e o

gerenciamento das políticas de segurança a serem implementadas pelos *firewalls*, através da adição de objetos de políticas PFIM descritos segundo o modelo proposto, sem que haja a necessidade de efetuar alterações nos objetos que representam a estrutura computacional existente.

4.2.2. O Modelo de Classes PFIM

A Figura 4.1 apresenta um diagrama baseado no modelo de informação do PCIMe [RFC3460]. Este diagrama apresenta as principais classes e associações relevantes ao escopo do modelo, assim como as novas classes e associações introduzidas por este trabalho.

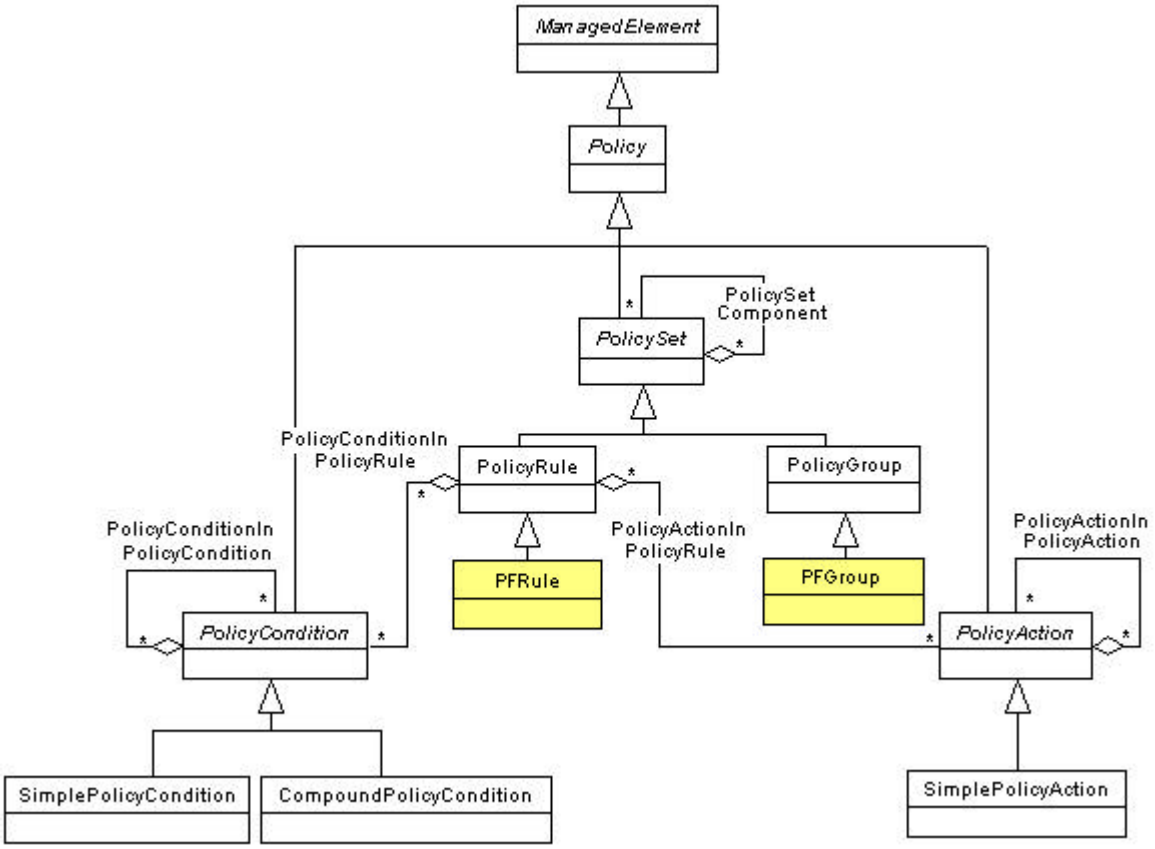

Figura 4.1 – Modelo de Classes do PFIM Simplificado

A classe *ManagedElement* (uma especialização da classe *ManagedSystemElement* do CIM) em conjunto com classe *Policy*, atuam como a base da hierarquia de classes do PFIM.

As classes *PFGroup* e *PFRule* são especializações das classes *PolicyGroup* e *PolicyRule* do PCIMe, respectivamente, e representam os grupos e regras de filtros de pacotes. Através da hierarquia mostrada na Figura 4.1, a classe *PFRule* herda as propriedades

e a semântica "se condição então ação" da classe *PolicyRule*. Seu objetivo é caracterizar que a regra associada é do tipo regra de filtros de pacotes. Desta forma serão evitadas possíveis confusões entre regras de diferentes naturezas contidas no diretório tais como, por exemplo, regras relacionadas a QoS, IPSec ou RBAC. O mesmo se aplica a *PFGroup*.

A classe *PolicySet* define a semântica de aninhamento de regras ou de se formar conjuntos de regras dentro de grupos. Através do conceito de herança, essa semântica também será implementada por *PFRule* e *PFGroup*.

Os filtros de pacotes executam, normalmente, as ações associadas à primeira condição que seja validada como verdadeira. Para informar ao PEP que seu dispositivo associado deve assumir esse comportamento, a propriedade *PolicyDecisionStrategy* deverá assumir o valor associado à estratégia *FirstMatching*. A propriedade *PolicyRoles* oferece uma forma de se caracterizar os recursos de rede aos quais o conjunto de políticas se aplica. Este conceito será mais profundamente abordado na Seção 4.2.3.

O aninhamento de regras ou a formação de conjuntos de regras é possível através do uso da classe de agregação *PolicySetComponent*. A propriedade *PolicySetComponent.Priority* define a ordem em que as regras devem ser executadas dentro de uma mesma árvore *PolicySet*. Entretanto, um recurso pode possuir várias árvores *PolicySet* disjuntas, que podem estar reunidas através de um papel ou pela combinação de papéis. Dessa forma, a associação *PolicySetInSystem*, derivada de *PolicyInSystem* (do CIM) também define uma propriedade *Priority* que estabelece o conceito de prioridade entre árvores. Conforme exposto no Capítulo 3, a classe *PolicyRule* define a semântica "se condição então ação".

A natureza de funcionamento dos filtros de pacotes especifica que as regras associadas às condições sejam agrupadas através de um operador relacional AND. É a propriedade *PolicyRule.ConditionListType* quem define a forma como as condições associadas à regra devem ser avaliadas. Assim a composição das condições assumirá a forma DNF (1).

As agregações *PolicyConditionInPolicyRule* e *PolicyActionInPolicyRule* oferecem uma forma de se agrupar objetos das classes *CompoundFilterCondition* e *SimplePolicyAction*, respectivamente, a objetos da classe *PFRule*.

Como exposto no Capítulo 3, a classe *CompoundPolicyCondition* agrega objetos *SimplePolicyCondition* para formar expressões complexas através da agregação *PolicyConditionInPolicyCondition*. As agregações *PolicyConditionInPolicyRule* e

PolicyConditionInPolicyCondition são especializações da classe *PolicyConditionStructure*. Esta classe define as propriedades *GroupNumber*: que é uma referência numérica que indica o agrupamento aos quais as condições pertencem na construção das expressões na forma DNF e *ConditionNegated*: que indica se a condição deve ser negada. Como a proposta é gerenciar regras de filtros de pacotes, este trabalho assumirá sempre o valor "zero" para *GroupNumber* e *"FALSE"* para *ConditionNegated*.

Similarmente a *PolicyConditionInPolicyRule*, as agregações *PolicyActionInPolicyRule* e *PolicyActionInPolicyAction* são especializações da classe *PolicyActionStructure*. Apesar do modelo definido pelo PCIMe prever a execução de várias ações a partir da validação de uma ou mais condições, este trabalho não prevê tal tipo de composição de ações. Assim, será utilizada apenas a agregação *PolicyActionInPolicyRule*. Ainda, não será necessária a utilização da propriedade *ActionOrder*, uma vez que somente uma ação estará associada à validação de cada *CompoundPolicyCondition* como verdadeira. As únicas ações definidas para filtros de pacotes, segundo a estratégia adotada pelo Firmato são *"pass"* ou *"drop"*.

Também é possível especificar o período de ativação de uma regra através da classe *PolicyTimePeriodCondition*, mas a implementação desta condição será objeto de estudo de trabalhos futuros.

4.2.3. Aplicação do Conceito de Papéis

O conceito de papel é central na proposta de um *framework* baseado em políticas para configuração de filtros de pacotes. A idéia é muito simples. Ao invés de configurar individualmente cada um dos dispositivos da rede, o administrador de políticas simplesmente atribui, a cada dispositivo, um ou mais papéis e, então, especifica as políticas, sob a forma de regras, referentes a cada um desses papéis [BAR99]. O *framework* de políticas é o responsável por configurar cada um dos dispositivos associados com o papel para este se comportar de acordo com as políticas especificadas para o papel. Quando o estado da rede se altera e é necessário alterar qualquer política, o administrador executará a alteração e o *framework* de políticas irá garantir que sejam feitas as atualizações em todos os dispositivos associados aos papéis que contêm as políticas alteradas [RFC3060]. Assim, a topologia da rede se torna transparente ao administrador de políticas.

Formalmente, um papel pode ser definido como [RFC3198]:

> "Uma característica administrativa específica de um elemento da rede (por exemplo, uma interface). É um seletor para as regras de política que determina a aplicabilidade de uma regra a um elemento da rede em particular".

É importante notar que um papel é muito mais que um atributo. É o papel quem define a função do elemento da rede identificando o respectivo comportamento a ele associado.

A propriedade *PolicyRoles* está definida na classe *PolicySet*. Portanto, o conceito de papéis pode ser naturalmente estendido a grupos de regras *PolicyGroups* (e *PFGroups*) e *PolicyRules* (e *PFRules*) através de herança. Assim, uma instância de *PFGroup* poderia assumir uma determinada combinação de papéis de maneira que cada *PFRule* contida nesse grupo herdará, implicitamente, todos os papéis assumidos pelo grupo. Um cuidado deve ser tomado neste ponto para evitar que uma determinada regra assuma um papel indevido. Conforme demonstrado no Capítulo 2, [BAR99] oferece uma forma de se manipular esse tipo de inconsistência através do conceito de grupo aberto ou fechado de papéis.

A semântica de que um papel é um "seletor de políticas" pode estender à idéia de grupos[15] de papéis como uma reunião de regras de origens distintas. Assim, os papéis podem ser utilizados para selecionar uma ou mais políticas específicas referentes a um ou mais dispositivos dentro de um enorme conjunto de políticas disponíveis. Ou seja, é possível pré-definir um grande número de políticas genéricas que estarão sempre disponíveis, mas que somente serão atribuídas após a identificação dos dispositivos que assumem os papéis referentes a essas políticas. O processo de seleção das políticas associadas a um grupo de papéis passa pela escolha das políticas associadas ao próprio grupo, políticas associadas a cada um dos sub-grupos e políticas associadas a cada um dos papéis contidos no grupo.

Em [WAL03] as regras de políticas são da forma "SE <*policyFilter*> ENTÃO <*policyAction*>", onde <*policyFilter*> é um conjunto de condições que são utilizadas para determinar se uma política deve ou não ser aplicada a um objeto. O filtro de políticas pode executar operações de comparação entre variáveis SNMP pré-definidas nas MIBs (por exemplo, *"ifType = = ethernet"*). [WAL03] define uma tabela para associar os elementos aos papéis. Enquanto a semântica do papel, como seletor de políticas, é a mesma, esta abordagem difere do PCIMe no seguinte sentido: no PCIMe, os papéis são descritos através da propriedade *PolicyRoles* das regras de políticas.

[15] O termo "grupo" não tem nenhuma relação com *PolicyGroup* e/ou suas especializações.

Assim, é necessário que um elemento externo identifique quais objetos estão associados a um determinado papel. Ou seja, a implementação dessa semântica é de responsabilidade do PDP, que é o referido elemento externo. Entretanto, [WAL03] não exige que nenhum processamento especial seja feito para executar a semântica de papéis. Ao contrário, os papéis são tratados como qualquer outra variável SNMP e as comparações de seus valores podem ser incluídas como mais um filtro na regra de política. Além disso, o PCIM não define um método formal de associação entre papéis e objetos. Como dito anteriormente, [WAL03] utiliza tabelas para efetuar esta associação entre papéis e objetos.

Para solucionar este problema, o PCIMe introduz uma nova classe *PolicyRoleCollection*, derivada da classe *Collection* do CIM, conforme a Figura 4.2. Detalhes sobre como o PCIMe associa um dispositivo a um papel podem ser encontrados na Seção 3.5.2.

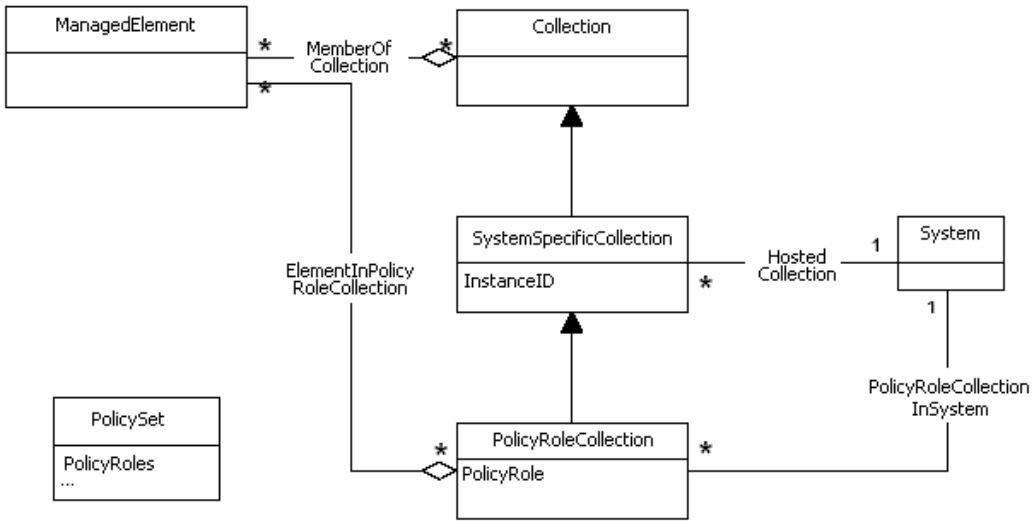

Figura 4.2 – Modelo de associação entre papéis e objetos no PCIMe

4.2.4. Variáveis e Valores

A natureza de operação dos filtros de pacotes sugere a utilização apenas de variáveis implícitas. Portanto, este trabalho não pretende demonstrar o uso de variáveis explícitas. [NAB03] é uma boa referência para descrição e utilização de variáveis explícitas.

O Capítulo 3 demonstrou como a classe *SimplePolicyCondition* refina a estrutura básica da classe *PolicyCondition*, através do uso do par <variável>/<valor> para formar uma condição de semântica "<variável> *match* <valor>". O par <variável>/<valor> associado a

uma *SimplePolicyCondition* é criado, respectivamente, por instâncias das subclasses de *PolicyVariable* e *PolicyValue*. As agregações *PolicyVariableInSimplePolicyCondition* e *PolicyValueInSimplePolicyCondition* são as responsáveis pela associação entre uma *SimplePolicyCondition* e as instâncias das subclasses de *PolicyVariable* e *PolicyValue*, respectivamente. A Figura 4.3 ilustra esta estrutura.

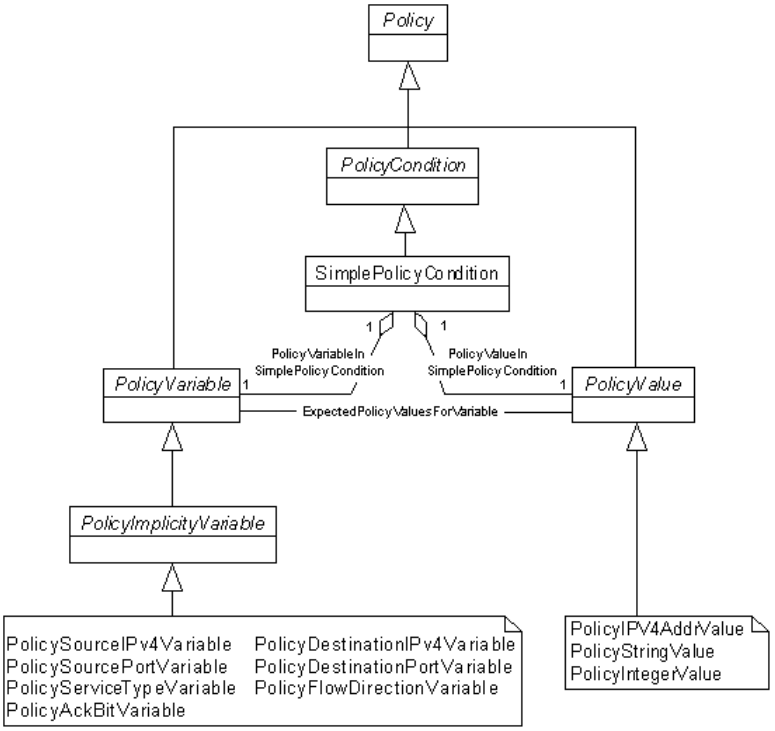

Figura 4.3 – Diagrama de classes de variáveis e valores

PolicyImplicitVariable é uma especialização de *PolicyVariable* e representa informações referentes aos cabeçalhos das mensagens que trafegam pela rede. Como pode ser visualizado na Figura 4.3, o modelo proposto pelo PFIM utiliza cinco classes (*PolicySourceIPv4Variable*, *PolicyDestinationIPv4Variable*, *PolicySourcePortVariable*, *PolicyDestinationPortVariable* e *PolicyFlowDirectionVariable*) das vinte e uma, especializações de *PolicyImplicitVariable* pré-definidas no PCIMe. As outras duas classes (*PolicyServiceTypeVariable* e *PolicyAckBitVariable*) que aparecem na Figura 4.3 são extensões propostas por este trabalho, conforme será exposto a seguir. A associação *ExpectedPolicyValuesForVariable* especifica o conjunto de valores que podem ser validados por uma variável dentro de uma condição simples. Nenhuma das especializações de *PolicyImplicitVariable* permite ao administrador de políticas especificar os serviços oferecidos pela rede através de uma descrição de alto nível. Por exemplo, classes pré-definidas tais como *PolicyIPProtocolVariable* são dependentes de códigos que necessitam de

tabelas de conversão para tornar sua utilização transparente. Outra informação referente a filtros de pacotes que não foi coberta pelo PCIMe é a representação do *flag* ACK do cabeçalho de mensagens TCP o qual indica o pedido de abertura de conexão [RFC3514].

Para cobrir essas lacunas, o PFIM propõe uma extensão do modelo do PCIMe através de duas novas especializações da classe *PolicyImplicitVariable*: a classe *PolicyServiceTypeVariable* que é uma abstração para a especificação de uma string que represente o serviço a ser filtrado. Por exemplo: “http”, “FTP”, “SMTP”, “TELNET” e outros. A classe *PolicyAckBitVariable* representa o *flag* ACK. O PFIM também define as propriedades *PolicyServiceTypeVariable.ValueTypes*, que deve receber um valor do tipo *PolicyStringValue* (livre) e *PolicyAckBitVariable.ValueTypes*, que deve receber um *PolicyIntegerValue* (limitado por: 0 – pedido de conexão; 1 – resposta; 2 – indiferente).

Conforme a figura 4.3, o modelo proposto pelo PFIM utiliza apenas três especializações de *PolicyValue* (*PolicyIPv4AddrValue*, *PolicyStringValue* e *PolicyIntegerValue*) para representar os valores válidos a que se referem as variáveis.

De maneira similar, a classe *SimplePolicyAction* refina a estrutura básica da classe *PolicyAction*, também através do uso do par <variável>/<valor>, mas com a semântica “*set* <variável> *to* <valor>”. O par <variável>/<valor> associado a uma *SimplePolicyAction* também é criado, respectivamente, por instâncias das subclasses de *PolicyVariable* e *PolicyValue*. As agregações *PolicyVariableInSimplePolicyAction* e *PolicyValueInSimplePolicyAction* são as responsáveis pela associação entre uma *SimplePolicyAction* e as instâncias das subclasses de *PolicyVariable* e *PolicyValue*, respectivamente, conforme a Figura 4.4.

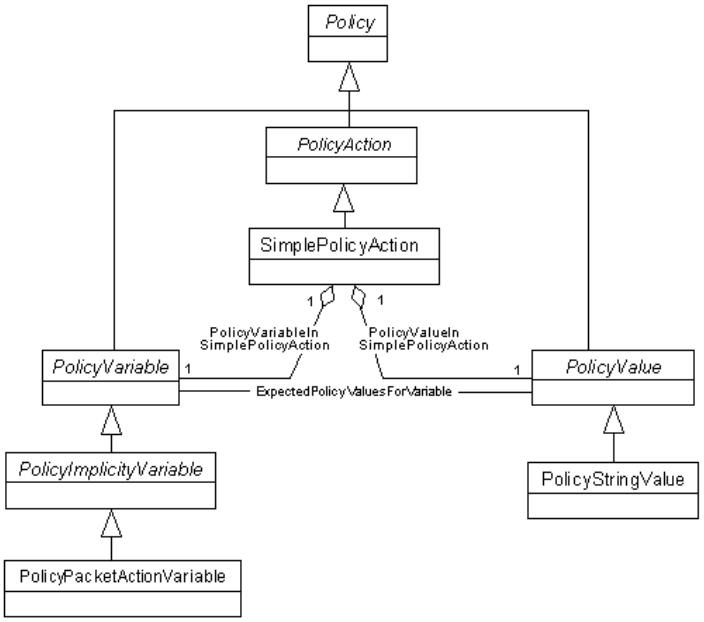

Figura 4.4 – Diagrama de classes de variáveis e valores para *SimplePolicyAction*

Para representar a ação a ser tomada pelo filtro de pacotes ao validar uma condição como verdadeira, o PFIM propõe uma nova especialização da classe *PolicyImplicitVariable*: a classe *PolicyPacketActionVariable.* A propriedade *PolicyPacketActionVariable.ValueTypes*, deve receber um valor do tipo *PolicyStringValue* (limitado por: *"pass"* ou *"drop"*).

4.2.5. Filtros de Pacotes

O Capítulo 3 demonstrou uma das formas definidas no PCIMe para implementar filtros de pacotes. Nesta Seção, será apresentada a estratégia de armazenamento das regras de um filtro de pacotes do PFIM a partir da tabela de filtros, demonstrando o mapeamento das regras do filtro no modelo do PCIMe. Para tanto, considere as regras de filtragem para o acesso de um computador interno ao serviço FTP localizado fora da sub-rede (o servidor FTP pode estar localizado em outra sub-rede ou na Internet) listadas na Tabela 4.1.

Tabela 4.1: Regras de filtragem para o serviço FTP

REGRA	SENTIDO	PROTOCOLO	IP ORIGEM	IP DESTINO	PORTA ORIGEM	PORTA DESTINO	FLAG ACK	AÇÃO
3	OUT	FTP	INTERNO	*	> 1023	21	*	PASS
2	IN	FTP	*	INTERNO	21	> 1023	1	PASS
1	*	*	*	*	*	*	*	DROP

Cada célula das colunas SENTIDO, PROTOCOLO, IP ORIGEM, IP DESTINO, PORTA ORIGEM, PORTA DESTINO E FLAG ACK devem ser representadas por uma instância de *SimplePolicyCondition*, ou seja, cada célula dessas colunas representa uma

condição. Cada célula da coluna AÇÃO deve ser representada por uma instância de *SimplePolicyAction* no modelo PFIM.

O conjunto de colunas representadas por instâncias de *SimplePolicyCondition* é representado, linha a linha, por uma instância da classe *CompoundFilterCondition*.

A associação entre uma instância da classe *CompoundFilterCondition* à respectiva instância da classe *SimplePolicyAction* pertencente à mesma linha, forma a semântica da representação de uma regra. Portanto, cada linha da Tabela 4.1 é representada por uma instância de *PFRule* dentro do modelo proposto pelo PFIM.

O conteúdo das células da coluna REGRA define a prioridade entre as regras. A semântica desta propriedade define que o maior valor possui a maior prioridade.

Para esta política, sugere-se o nome "*Enable_FTP*". Todas as três regras assumem esse papel. Ainda, essas regras podem estar agrupadas em uma instância da classe *PFGroup* [BAR99].

Dessa forma, o modelo proposto pelo PFIM oferece uma estrutura flexível para se montar e gerenciar uma biblioteca de regras genéricas que atendem, virtualmente, a qualquer tipo de *firewall* baseado em filtros de pacotes para implementar o controle seletivo de mensagens.

4.2.5.1. Variáveis e Valores

Como exposto anteriormente, uma condição refina a semântica definida para uma regra através da introdução do par variável / valor através da expressão variável *match* valor.

Assim, a cada condição da Tabela 4.1 será agregado um par formado por instâncias das subclasses de *PolicyImplicitVariable* e *PolicyValue*. A Tabela 4.2 demonstra a correspondência entre as colunas e as instâncias.

Tabela 4.2 – Agregação do par variável / valor a uma condição

FILTRO	PAR VARIÁVEL / VALOR
SENTIDO	***PolicyVariable**: PolicyFlowDirectionVariable* ***PolicyValue**: PolicyStringValue* (*"IN", "OUT"*)
PROTOCOLO	***PolicyVariable**: PolicyServiceTypeVariable* ***PolicyValue**: PolicyStringValue* (*"HTTP","SMTP", "FTP", etc.*)

IP ORIGEM	***PolicyVariable**: PolicySourceIPv4Variable* ***PolicyValue**: PolicyIPv4AddrValue* (Endereço IP válido)
IP DESTINO	***PolicyVariable**: PolicyDestinationIPv4Variable* ***PolicyValue**: PolicyIPv4AddrValue* (Endereço IP válido)
PORTA ORIGEM	***PolicyVariable**: PolicySourcePortVariable* ***PolicyValue**: PolicyIntegerValue (0..65535)*
PORTA DESTINO	***PolicyVariable**: PolicyDestinationPortVariable* ***PolicyValue**: PolicyIntegerValue (0..65535)*
FLAG ACK	***PolicyVariable**: PolicyAckBitVariable* ***PolicyValue**: PolicyIntegerValue* (0 – pedido de conexão; 1 – resposta; 2 – indiferente)

Da mesma forma como definido para uma condição, uma ação refina a semântica definida para uma regra através da expressão "*set* variável *to* valor".

Assim, a cada ação da Tabela 4.1 será agregado um par formado por instâncias das classes de *PolicyPacketActionVariable* e *PolicyStringValue*. A Tabela 4.3 demonstra a correspondência entre as colunas e as instâncias.

Tabela 4.3 – Agregação do par variável / valor a uma ação

FILTRO	**PAR VARIÁVEL / VALOR**
AÇÃO	***PolicyVariable**: PolicyPacketActionVariable* ***PolicyValue**: PolicyStringValue* (*"PASS", "DROP"*)

4.2.6. O Mapeamento do PFIM no LDAP

[REY03] apresenta uma maneira pela qual o *framework* do PCIMe pode ser mapeado por um esquema LDAP através do PCELS. Nesta seção, serão apresentadas as classes e os atributos específicos utilizados para fazer o mapeamento do *framework* proposto pelo PFIM, no LDAP, utilizando os conceitos do PCELS.

O modelo proposto pelo PFIM introduz as classes *PFGroup* e *PFRule*. A Tabela 4.4 apresenta o mapeamento dessas classes no LDAP. O Anexo A apresenta o mapeamento de todas as classes envolvidas no modelo, assim como seus atributos e a descrição de suas funcionalidades.

Tabela 4.4 – Classes PFIM no LDAP

PFIM	Classe LDAP
PFGroup	*pcimGroup(ab)* *pfimGroupAuxClass (aux)* *pfimGroupInstance (struc)*
PFRule	*pcimPolicyRule (ab)* *pfimRuleAuxClass (aux)* *pfimRuleInstance (struc)*
PolicyServiceTypeVariable	*pfimServiceTypeVarAuxClass (aux)*
PolicyAckBitVariable	*pfimAckBitVariableAuxClass (aux)*
PolicyPacketActionVariable	*pfimPacketActionVarAuxClass (aux)*

Conforme a Tabela 4.4, a classe *PFGroup* é mapeada por duas classes. Esta estratégia de mapeamento mantém a flexibilidade do modelo proposto pelo PCELS, uma vez que possibilita duas alternativas para a especificação de grupos *PFGroup*:

- objetos já existentes no diretório podem ser anexados através da classe *pfimGroupAuxClass;*
- novos objetos podem ser inseridos a partir da classe estrutural *pfimGroupInstance.*

Esta flexibilidade também está presente no mapeamento da classe *PFRule* que também é mapeada por outras duas classes: a classe abstrata *pcimPolicyRule* e suas especializações *pfimRuleAuxClass* (auxiliar) e *pfimRuleInstance* (estrutural), onde a estratégia de mapeamento é similar à usada para *PFGroup*.

As classes *pcimConditionAuxClass* e *pcimActionAuxClass* introduzem as semânticas de condições de política associadas a ações. Dentre essas semânticas está a que representa a utilização de condições/ações específicas e a que representa condições/ações reusáveis. Neste sentido, [REY03] propõe classes que possibilitam o mapeamento LDAP destas associações para os dois casos. Como visto na seção 3.5, o PCIMe introduz as classes *CompoundPolicyCondition* e *CompoundPolicyAction* como especializações de *PolicyCondition* e *PolicyAction*, respectivamente. As condições/ações complexas ou simplesmente composições, definidas no PCIMe, estendem a capacidade de uma regra para

associar, agrupar e avaliar/executar condições/ações por oferecerem um meio simples de se montar expressões complexas (Vide a Figura 3.12).

De forma a oferecer maior flexibilidade no armazenamento e montagem dos filtros, das quatro possibilidades de construção de condições/ações específicas/reusáveis, simples/complexas oferecidas pelo PCELS, o PFIM utiliza o quarto caso: composição de condições/ações reusáveis formada por condições/ações reusáveis. Todas as condições/ações são reusáveis.

Portanto, todas são armazenadas em containeres reusáveis. A Figura 4.5 ilustra esta situação. Dois containeres de políticas reusáveis diferentes são utilizados para armazenar as condições/ações. No Repositório A estão armazenadas as composições (condições/ações complexas) reusáveis. No repositório B ficam armazenadas as condições/ações simples, que irão formar as composições.

As classes associativas CA1 e CA2 são instâncias de *pcimConditionAssociation* ou *pcimActionAssociation*. São agregadas à Regra através do atributo Regra.*pcimConditionList* ou Regra.*pcimActionList*. Também podem ser agregadas por referência DN através do atributo *pcimPolicySetList*. Por serem reusáveis, as composições cc1 e cc2 não podem ser anexadas diretamente às classes associativas CA1 e CA2. Essa associação será efetuada por referência DN às instâncias S1 e S2. Assim, cc1 e cc2 são anexadas às classes estruturais S1 e S2 (instâncias de *pcimReusableContainerInstance*), respectivamente, contidas no Repositório A. As instâncias de *pcimReusableContainerInstance* são agregadas pelo container Repositório A por referência DN (atributo *pcimReusableContainerList*).

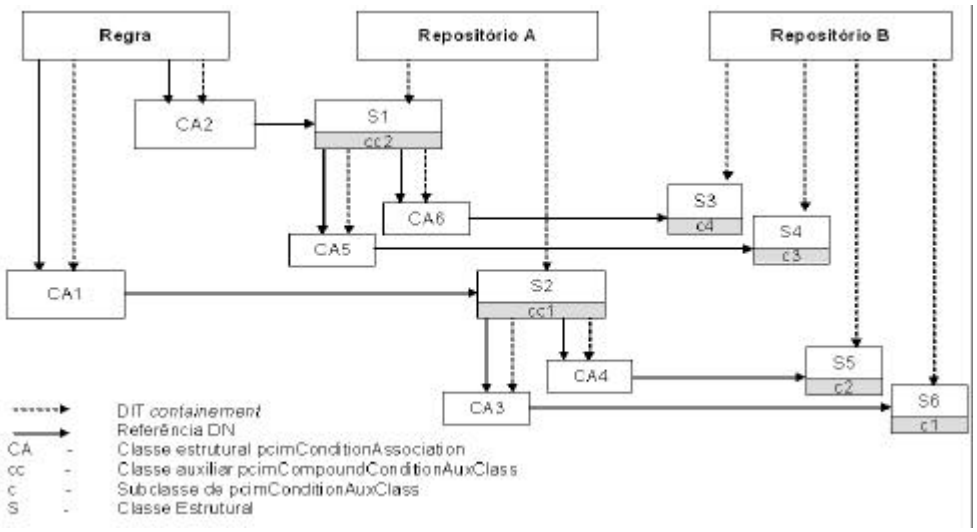

Figura 4.5 – Combinação reusável formada por condições/ações reusáveis

As condições/ações simples c1, c2, c3 e c4 que formam as composições reusáveis (cc1 e cc2) também são reusáveis. Assim, por uma questão administrativa, essas condições/ações estão armazenadas no *container* Repositório B. Da mesma forma que no Repositório A, as condições/ações simples c1, c2, c3 e c4 são anexadas às classes estruturais S3, S4, S5 e S6 (instâncias de *pcimReusableContainerInstance*), respectivamente, contidas no Repositório B. Essas instâncias S3, S4, S5 e S6 de *pcimReusableContainerInstance* são agregadas pelo *container* Repositório B por referência DN.

Novamente, a associação entre as condições/ações reusáveis CA3, CA4, CA5 e CA6 do Repositório A e as condições/ações simples c1, c2, c3 e c4 do Repositório B será dada por referência DN para as instâncias S3, S4, S5 e S6 de *pcimReusableContainerInstance*.

4.3. Conclusões do Capítulo

Este capítulo apresentou uma proposta de implementação para o framework do PFIM baseado nas padronizações propostas pelo DMTF e IETF do Capítulo 3. Para este trabalho, buscou-se por uma alternativa para a descrição, o gerenciamento e a implementação das políticas nos dispositivos de segurança em conformidade com objetivos principais abordados no Firmato [BAR99]. Para tanto, foi necessário estender as classes do modelo do PCIMe [RFC3460] e, posteriormente, mapeá-las no serviço de diretório LDAP, segundo as regras descritas no PCELS [REY03]. A partir das definições contidas neste capítulo, o Capítulo 5 apresentará um estudo de caso e a implementação do protótipo, objetivando avaliar a proposta contida neste trabalho.

Capítulo 5

Estudo de Caso

5.1. Introdução

O Capítulo 4 apresentou um esquema de diretório para o mapeamento do PFIM no LDAP. Através da utilização deste esquema, os administradores de políticas poderão especificar as regras que devem ser executadas pelos dispositivos de segurança que implementem a técnica *packet filtering*. A utilização de serviço de diretório baseado em LDAP disponibiliza benefícios tais como interoperabilidade, centralização lógica e distribuição física [NAB03].

Com o objetivo de validar e demonstrar como o modelo e o esquema de diretório do PFIM podem ser utilizados para a definição dos filtros de pacotes, este capítulo realiza um breve estudo de caso envolvendo a configuração de um roteador que protege uma rede com alguns servidores. Também serão apresentadas as consultas LDAP que tornam possível a recuperação das regras a partir do diretório.

5.2. Estudo de Caso

Este estudo de caso foi baseado nas técnicas usuais de configuração de filtros de pacotes observadas por administradores de segurança em redes corporativas.

5.2.1. Cenário

Os objetivos de uma corporação ao conectar sua rede à Internet são oferecer algum tipo de serviço a usuários remotos e/ou utilizar algum tipo de serviço remoto. A conexão de uma rede interna à Internet cria um problema: como evitar que recursos computacionais da rede interna fiquem indevidamente expostos a outros usuários conectados à Internet? Apenas os computadores que disponibilizam serviços básicos como páginas web, servidores de arquivos e correio eletrônico devem ser liberados para acesso externo. Assim, modelo de gestão de segurança adotado para resolver esse problema consiste em estabelecer um controle seletivo de acesso entre a rede interna e a Internet. A função do *firewall*, como um filtro de pacotes, neste contexto, é altamente restritiva. Ele deve bloquear qualquer acesso oriundo da rede externa, exceto aqueles associados aos serviços básicos da Internet que estarão explicitamente declarados através da tabela de filtragem.

Para ilustrar essa situação, considere a Figura 5.1, onde está representada a topologia da rede XNC, uma rede típica para uma pequena empresa.

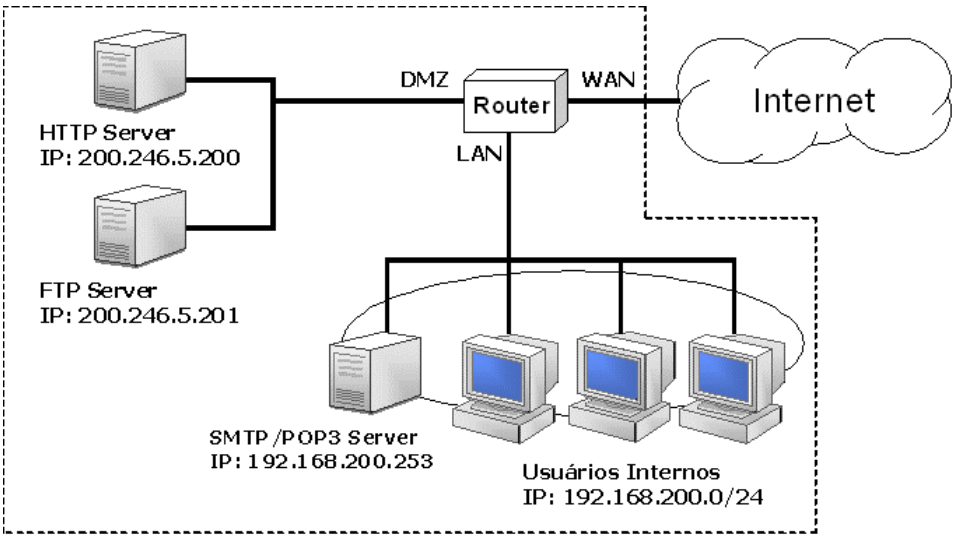

Figura 5.1 – Topologia da Rede XNC

Nesta topologia, é possível observar que há usuários localizados na rede interna e servidores localizados na DMZ[16]. O roteador possui três interfaces que atuam na conexão entre rede interna e a Internet. Nas interfaces LAN/WAN e LAN/DMZ é necessário que o roteador implemente a funcionalidade NAT – *Network Address Translation* – para que usuários da rede interna acessem servidores localizados na Internet e na DMZ, respectivamente. Na interface WAN/DMZ não é necessária esta funcionalidade. O roteador também é o responsável por implementar os filtros de pacotes.

Os usuários internos têm direito de fazer o *download* de arquivos e acessar páginas *web* localizadas tanto na DMZ quanto na Internet. Os usuários externos não devem acessar os computadores localizados na rede interna, mas podem acessar os computadores localizados na DMZ (apenas para os serviços oferecidos pelos servidores).

O serviço de correio eletrônico é oferecido aos usuários internos pelo servidor SMTP/POP3 e não deve ser acessado por usuários externos.

A Tabela 5.1 apresenta as regras de filtragem a serem implementadas pelo roteador.

Tabela 5.1: Regras de filtragem para o estudo de caso.

SENTIDO	PROTOCOLO	IP ORIGEM	IP DESTINO	PORTA ORIGEM	PORTA DESTINO	FLAG ACK	AÇÃO
OUT	HTTP	INTERNO	*	*	80	*	PASS
IN	HTTP	*	INTERNO	80	*	1	PASS
IN	HTTP	EXTERNO	200.246.5.200	*	80	*	PASS
OUT	HTTP	200.246.5.200	*	80	*	1	PASS
OUT	FTP	INTERNO	*	*	21	*	PASS
IN	FTP	*	INTERNO	21	*	1	PASS
IN	FTP	EXTERNO	200.246.5.201	*	21	*	PASS
OUT	FTP	200.246.5.201	*	21	*	1	PASS
OUT	SMTP	192.168.200.253	EXTERNO	*	25	*	PASS
IN	SMTP	EXTERNO	192.168.200.253	25	*	1	PASS
*	*	*	*	*	*	*	DROP

No exemplo deste estudo de caso é apresentado apenas um roteador, mas a arquitetura de redes baseadas em políticas permite a configuração de tantos filtros quantos forem necessários. O caso extremo seria a utilização de *Personal Firewalls*, onde cada computador na rede possui suas próprias regras de filtragem, com a vantagem de que o gerenciamento, utilizando-se o modelo PFIM, passa a ser centralizado.

[16] DMZ - *DeMilitarized Zone* ou Zona Desmilitarizada. Recebe este nome, pois nessa rede são colocados os computadores que estão sujeitos a contato direto com o mundo externo. A rede interna fica protegida por um filtro de pacotes, que limita também o acesso da DMZ em relação à rede interna. Dessa forma, computadores da rede externa não poderão utilizar esses servidores como ponte para acessar a rede interna [ZWI00].

A partir dos conceitos apresentados nos Capítulos 3 e 4, as regras da Tabela 5.1 podem ser mapeadas e armazenadas em um serviço de diretório LDAP. O serviço de diretório LDAP utilizado como repositório das informações de políticas do PFIM foi o *Sun ONE Directory Server* 5.1. A Tabela 5.2 apresenta os papéis associados a cada um dos servidores especificados neste estudo de caso. Conforme exposto no Capítulo 4, o papel assumido por um dispositivo é fundamental na definição das regras associadas ao dispositivo.

Tabela 5.2: Os papéis dos Servidores no Estudo de Caso da Rede XNC.

Servidor	**Papel**
200.246.5.200	pcimroles=httpserver
200.246.5.201	pcimroles=ftpserver
192.168.200.253	pcimroles=smtpserver

5.2.2. A DIT para a Rede do Estudo de Caso

Tipicamente, as informações estão contidas em um diretório a partir de uma entrada que representa uma dada organização (atributo RDN **o**). Dentro desta entrada existem unidades organizacionais (atributo RDN **ou**), as quais permitem agrupamentos lógicos de entradas em função de suas representatividades dentro da organização.

Por exemplo, pode existir uma unidade organizacional para cada sub-rede da topologia. Nesta linha, este trabalho define que entradas referentes às unidades organizacionais serão containeres para entradas referente a grupos de regras, condições e ações reusáveis. Seguindo o esquema do PFIM, isto é conseguido com o *attachment* da classe auxiliar *pfimGroupAuxClass* à entrada da unidade organizacional, o que a transforma em um *PolicyGroup*. De maneira similar, este mesmo método pode ser aplicado para a montagem de containeres de condições/ações compostas/simples reusáveis através do *attachment* das classes *pcimReusableContainerAuxClass* a uma unidade organizacional as quais fornecem os atributos necessários para a especificação das referências DNs aos objetos contidos. A Figura 5.2 apresenta a DIT do diretório, utilizada para este estudo de caso. Nesta figura, estão sendo mostrados os principais containeres do diretório com seus atributos RDN.

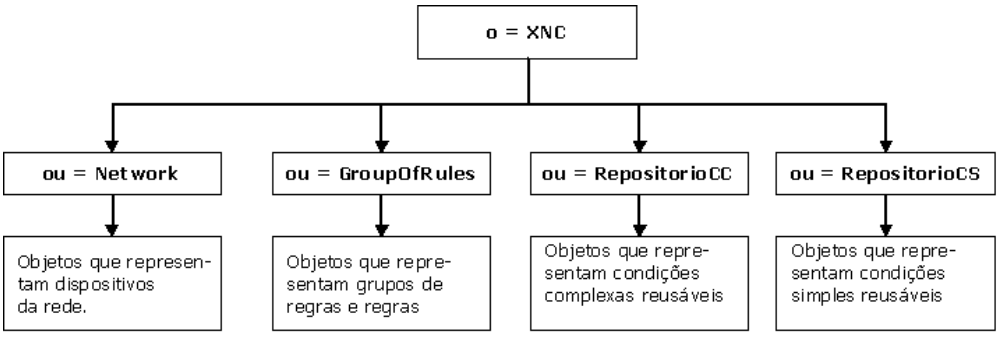

Figura 5.2 – DIT do Diretório para o Caso da Rede XNC

5.2.3. Objetos do Diretório

Segundo a estratégia de mapeamento do PFIM no LDAP apresentada no Capítulo 4, a Figura 5.3 representa como as informações relativas ao mapeamento das regras de políticas são armazenadas no diretório.

A Figura 5.3 mostra, esquematicamente, uma regra e seus principais objetos associados. Seguindo o modelo de armazenamento das composições (condições/ações simples reunidas na forma de uma condição/ação complexa) sugerido por [REY03], onde composições reusáveis são formadas por condições/ações simples reusáveis, uma regra agrega composições reusáveis por referência DN a uma composição anexada a uma instância de um *container*. Por sua vez, uma composição referencia, pelo DN, as condições/ações simples reusáveis que irão formar a composição, armazenadas em uma instância de um outro *container*. Aqui, as condições/ações simples também são *attachments.* Vale observar que uma regra pode agregar várias condições e ações.

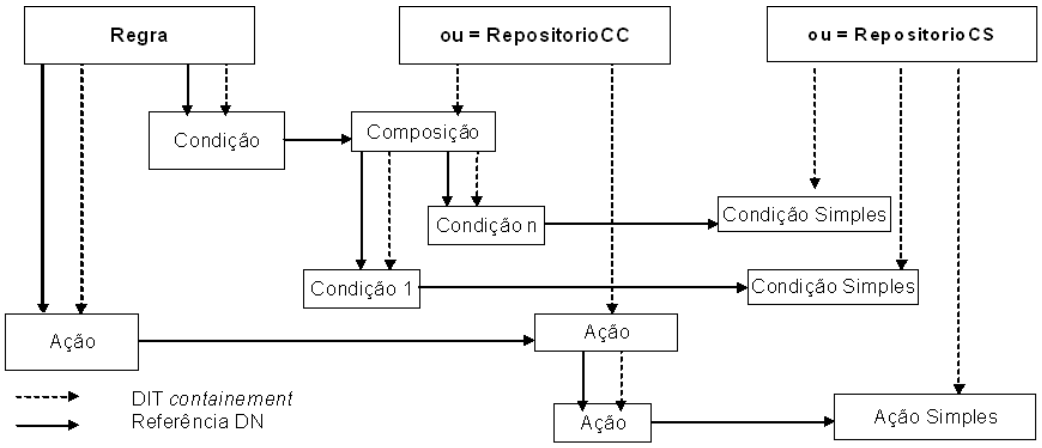

Figura 5.3 – Uma regra e seus principais objetos

5.2.4. As Consultas LDAP

Para recuperar os dados contidos em um diretório, a arquitetura proposta pelo PFIM utiliza os papéis desempenhados pelos dispositivos da rede para determinar quais regras pertencem a um determinado dispositivo.

Consultas LDAP seguem a sintaxe [RFC1959] [RFC1558]:

ldap[s]://<host>:<porta>/<base_dn>?<atributos>?<escopo>?<filtro>

onde :

- <host> é o nome (ou endereço IP) do servidor LDAP.
- <porta> é a porta do servidor LDAP. Se nenhuma porta for especificada, assume-se a porta padrão 389.
- <base_dn> é o DN de uma entrada no diretório. Esse DN identifica a entrada que é o ponto inicial da procura. Se esse componente estiver vazio, a procura começa no DN da raiz.
- <atributos> são os atributos que serão retornados. Os atributos são separados por vírgula (por exemplo, *pcimPolicySetList* e *pcimRuleName*). Se nenhum atributo for especificado na URL, todos os atributos do objeto serão retornados.
- <escopo> Define o escopo da procura. Pode ser um dos seguintes valores:
  - *one*: procura as informações abaixo do atributo (<base_dn>) da URL. A entrada de base não é incluída nesse escopo.
  - *base*: procura as informações somente no objeto especificado pelo atributo (<base_dn>) da URL.
  - *sub*: procura as informações sobre as entradas de todos os níveis abaixo do atributo (<base_dn>) da URL, inclusive da própria entrada de base. Se nenhum escopo for especificado, adota-se uma busca do tipo base.
- <filtro> especifica o critério de procura a ser aplicado na busca. Os filtros LDAP são basicamente relações do tipo (atributo = valor). Podem-se montar

critérios de filtragem concatenando atributos e utilizando caracteres "coringas" tais como "*". A sintaxe do filtro é definida com detalhes na [RFC1558].

Neste contexto, toda vez que o PEP for inicializado ele deve abrir uma sessão com o PDP enviando todos os papéis a ele associados. O PDP deve, então, fazer uma busca dentro do diretório para determinar quais são as regras definidas associadas aos papéis enviados pelo PEP. Essa busca deve ser feita, em passos, na seguinte seqüência:

Passo 1: determinar a partir do papel informado pelo PEP quais os grupos e regras associados ao papel.

Consulta 1: obter todas as regras e grupos de regras que possuam os papéis informados pelo PEP.

Base_dn:	o = XNC
Atributos:	pcimroles=httpserver
Escopo:	sub
Filtro:	(&(|(objectclass=pfimRuleInstance)(objectclass=pfimRuleAuxClass))(pcimroles=<papel>))

Esta consulta deve retornar os DNs de todos os grupos e regras que assume o papel enviado pelo PEP. No estudo de caso, para o papel "*httpserver*", o valor retornado será, conforme a listagem LDIF[17] a seguir:

```
dn: pcimrulename=Rule1HTTPServer,ou=GroupOfRulesHTTPServer, o=XNC, dc=com
dn: pcimrulename=Rule2HTTPServer,ou=GroupOfRulesHTTPServer, o=XNC, dc=com
dn: pcimrulename=Rule3HTTPServer,ou=GroupOfRulesHTTPServer, o=XNC, dc=com
dn: pcimrulename=Rule4HTTPServer,ou=GroupOfRulesHTTPServer, o=XNC, dc=com
dn: pcimrulename=Rule5HTTPServer,ou=GroupOfRulesHTTPServer, o=XNC, dc=com
```

Passo 2: ao se determinar quais são as regras, é possível determinar as composições de condições/ações associadas às regras a partir da seguinte consulta para cada uma das entrada mostradas na listagem LDIF acima:

[17] *LDIF – LDAP Data Interchange Format* – é um formato para definição de entradas de um diretório no formato texto. A maioria dos serviços de diretórios permite a importação/exportação de entradas a partir de um arquivo no formato LDIF.

Consulta 1: busca dos DNs das condições complexas (executada para cada uma das entradas retornadas pelo Passo 1:

Base_dn:	pcimrulename=Rule1HTTPServer, ou=GroupOfRulesHTTPServer, o=XNC, dc=com
Atributos:	pcimConditionDN
Escopo:	one
Filtro:	(objectclass=pcimconditionassociation)

Retorno:

pcimConditionDN:pcimreusablecontainername=CCrule1httpserver,ou=RepositorioCC,o=XNC,dc=com
pcimConditionDN:pcimreusablecontainername=CCrule2httpserver,ou=RepositorioCC,o=XNC,dc=com
pcimConditionDN:pcimreusablecontainername=CCrule3httpserver,ou=RepositorioCC,o=XNC,dc=com
pcimConditionDN:pcimreusablecontainername=CCrule4httpserver,ou=RepositorioCC,o=XNC,dc=com
pcimConditionDN:pcimreusablecontainername=CCrule5httpserver,ou=RepositorioCC,o=XNC,dc=com

Consulta 2: busca dos DNs das ações:

Base_dn:	pcimrulename=Rule1HTTPServer, ou=GroupOfRulesHTTPServer, o=XNC, dc=com
Atributos:	pcimActionDN
Escopo:	one
Filtro:	(objectclass=pcimactionassociation)

Retorno:

pcimActionDN: pcimreusablecontainername=ACrule_pass, ou=RepositorioCC, o=XNC, dc=com
pcimActionDN: pcimreusablecontainername=ACrule_pass, ou=RepositorioCC, o=XNC, dc=com
pcimActionDN: pcimreusablecontainername=ACrule_pass, ou=RepositorioCC, o=XNC, dc=com
pcimActionDN: pcimreusablecontainername=ACrule_pass, ou=RepositorioCC, o=XNC, dc=com
pcimActionDN: pcimreusablecontainername=ACrule_pass, ou=RepositorioCC, o=XNC, dc=com
pcimActionDN: pcimreusablecontainername=ACrule_drop, ou=RepositorioCC, o=XNC, dc=com

Passo 3: conhecendo as condições/ações complexas retornadas pelo Passo 2 é possível determinar os DNs de cada condição/ação simples a partir das seguintes consultas:

Consulta 1: busca dos DNs das condições simples (deve ser executada para cada uma das condições complexas):

Base_dn:	pcimreusablecontainername=CCrule1httpserver,ou=RepositorioCC,o=XNC,dc=com
Atributos:	PcimConditionList
Escopo:	Base
Filtro:	(objectclass=pcimcompoundconditionauxclass)

Retorno para cada uma das condições simples recuperadas no Passo 2:

dn:pcimreusablecontainername=CCrule1httpserver,ou=RepositorioCC,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=FlowDirection_OUT,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=Protocolo_HTTP,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourceIP_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestIP_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourcePort_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestPort_80,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=ACK_BIT_ANY,ou=RepositorioCS,o=XNC,dc=com

dn:pcimreusablecontainername=CCrule2httpserver,ou=RepositorioCC,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=FlowDirection_IN,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=Protocolo_HTTP,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourceIP_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestIP_LAN,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourcePort_80,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestPort_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=ACK_BIT_1,ou=RepositorioCS,o=XNC,dc=com

dn:pcimreusablecontainername=CCrule3httpserver,ou=RepositorioCC,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=FlowDirection_IN,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=Protocolo_HTTP,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourceIP_WAN,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestIP_HTTPServer,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourcePort_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestPort_80,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=ACK_BIT_ANY,ou=RepositorioCS,o=XNC,dc=com

dn:pcimreusablecontainername=CCrule4httpserver,ou=RepositorioCC,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=FlowDirection_OUT,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=Protocolo_HTTP,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourceIP_HTTPServer,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestIP_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourcePort_80,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestPort_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=ACK_BIT_1,ou=RepositorioCS,o=XNC,dc=com

dn:pcimreusablecontainername=CCruleDefault,ou=RepositorioCC,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=FlowDirection_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=Protocolo_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourceIP_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestIP_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourcePort_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestPort_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=ACK_BIT_ANY,ou=RepositorioCS,o=XNC,dc=com

Consulta 2: busca dos DNs das ações (deve ser executada para cada uma das ações):

Base_dn:	pcimreusablecontainername=CCrule1httpserver,ou=RepositorioCC,o=XNC,dc=com
Atributos:	PcimValueDN
Escopo:	Base
Filtro:	(objectclass=pcimsimpleactionauxclass)

Retorno para cada ação recuperada no Passo 2:

dn:pcimreusablecontainername=ACrule_pass,ou=RepositorioCC,o=XNC,dc=com
pcimvaluedn:pcimreusablecontainername=ACTION_PASS,ou=RepositorioCS,o=XNC,dc=com

dn:pcimreusablecontainername=ACrule_drop,ou=RepositorioCC,o=XNC,dc=com
pcimvaluedn:pcimreusablecontainername=ACTION_DROP,ou=RepositorioCS,o=XNC,dc=com

Passo 4: conhecendo os DNs das condições/ações simples retornadas pelo Passo 3 é possível determinar os valores associados a cada uma das condições/ações associadas a cada regra:

Consulta 1: busca dos valores das condições simples (deve ser executada para cada uma das condições simples de cada uma das regras):

Base_dn:	pcimreusablecontainername=FlowDirection_OUT,ou=RepositorioCS,o=XNC,dc=com
Atributos:	pcimStringList
Escopo:	base
Filtro:	(objectclass=pcimstringvalueauxclass)

Retorno:

dn:pcimreusablecontainername=FlowDirection_OUT,ou=RepositorioCS,o=XNC,dc=com
pcimStringList:OUT

Os outros valores dos filtros podem ser observados na listagem LDIF do Anexo B.

Consulta 2: busca dos DNs das ações:

Base_dn:	pcimreusablecontainername=ACTION_PASS,ou=RepositorioCS,o=XNC,dc=com
Atributos:	pcimStringList
Escopo:	base
Filtro:	(objectclass=pcimstringvalueauxclass)

Retorno:

dn:pcimreusablecontainername=ACTION_PASS,ou=RepositorioCS,o=XNC,dc=com
pcimStringList:PASS

Os outros valores dos filtros podem ser observados na listagem LDIF do Anexo B.

A unidade organizacional ou=Network serve para associar dispositivos da rede mapeados no diretório LDAP seguindo a estrutura proposta pelo CIM. A listagem LDIF abaixo apresenta a definição dos objetos no diretório.

```
dn: pfimdevicename=Router, ou=Network, o=XNC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: dlm1ManagedSystemElement
objectClass: dlm1LogicalElement
objectClass: dlm1LogicalDevice
objectClass: pfimdevice
pfimDeviceName: Router

dn: pfimdevicename=ServidorHTTP, ou=Network, o=XNC, dc=com
dlmDeviceID: 200.246.5.200
objectClass: top
objectClass: dlm1ManagedElement
objectClass: dlm1ManagedSystemElement
objectClass: dlm1LogicalElement
objectClass: dlm1LogicalDevice
objectClass: pfimdevice
pfimDeviceName: ServidorHTTP
```

A partir da definição de coleções de papéis, é possível associar um elemento de rede aos papéis definidos para PEPs quando estes solicitam as regras aos PDPs. Essa funcionalidade oferece uma forma fácil de se gerenciar as regras, pois fica simples a visualização de quais papéis pertencem aos dispositivos mapeados no diretório. A listagem LDIF abaixo apresenta a forma como uma coleção agrega os dispositivos mapeados no diretório através do atributo pcimElementList : pfimDeviceName = ServidorHTTP, ou = Network, o = XNC, dc = com. O mesmo conceito pode ser estendido aos outros servidores, conforme a listagem apresenta.

```
dn: pcimrolecollectionname=ServidoresHTTP, ou=Network, o=XNC, dc=com
pcimRoleCollectionName: ServidoresHTTP
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimrolecollection
pcimRole: httpserver
pcimElementList: pfimdevicename=ServidorHTTP, ou=Network, o=XNC, dc=com

dn: pcimrolecollectionname=ServidoresFTP, ou=Network, o=XNC, dc=com
pcimRoleCollectionName: ServidoresFTP
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimrolecollection
pcimRole: ftpserver
pcimElementList: pfimdevicename=ServidorFTP, ou=Network, o=XNC, dc=com
```

```
dn: pcimrolecollectionname=ServidoresSMTP, ou=Network, o=XNC, dc=com
pcimRoleCollectionName: ServidoresSMTP
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimrolecollection
pcimRole: smtpserver
pcimElementList: pfimdevicename=ServidorSMTP, ou=Network, o=XNC, dc=com
```

5.3. Conclusões do Capítulo

Este capítulo realizou um breve estudo de caso envolvendo cenários tipicamente observados em redes corporativas e como o PFIM poderia ser utilizado para modelar, armazenar e recuperar as regras de firewall definidas pelas políticas de segurança.

Cada *firewall* do estudo de caso assume um papel que especifica quais regras de segurança devem ser aplicadas de forma a se proteger a sub-rede ao qual o *firewall* está conectado. Ao se estabelecer uma política de controle de acesso aos recursos compartilhados pelas sub-redes, a conversão para regras de *firewall* é simples. Para a atribuição dessas regras aos dispositivos físicos, foi utilizada a própria definição dos dispositivos que já estava presente no diretório em uma unidade organizacional qualquer. Para realização desta avaliação as consultas LDAP seguiram a sintaxe definida na [RFC1959] e [RFC1558].

Capítulo 6

Conclusões e Trabalhos Futuros

Este trabalho apresentou o PFIM, um modelo para a definição e gerenciamento de regras de políticas para filtros de pacotes em redes corporativas. O modelo oferecido pelo PFIM é baseado nos objetivos propostos pelo Firmato [BAR99]. Esta proposta inova por ser uma alternativa para a descrição, gerenciamento e implementação das políticas nos dispositivos de segurança em conformidade com seus objetivos principais:

- separar a definição das políticas de segurança da sua implementação;
- tornar a definição das políticas independente da topologia da rede;
- gerar arquivos de configuração de filtros de pacotes automaticamente a partir da política de segurança.

Uma das contribuições importantes deste trabalho é a proposta de utilização do *framework* do PCIMe no gerenciamento da distribuição das regras de políticas em dispositivos de segurança de redes.

A estratégia de extensão do PCIMe foi mostrada no Capítulo 4 através do modelo do PFIM. Este modelo está em total acordo com os padrões definidos pelo DMTF e IETF para o mapeamento de regras de políticas PCIM e PCIMe [RFC3060] [RFC3460]. Todas as classes do modelo do PFIM são extensões que atendem às definições padronizadas pelo PCIMe. Isto permite a representação de políticas de forma que possibilite a softwares de fabricantes diferentes interpretarem um mesmo conjunto de políticas.

O Capítulo 4 também apresentou um esquema de diretório que pode ser utilizado para o armazenamento das informações de políticas PFIM em serviços de diretório LDAP,

estratégia recomendada pelas especificações do DMTF. [REY03] propõe o mapeamento dos objetos de políticas PCIMe em uma linha que tende a se tornar o padrão para implementações de regras de políticas em diretórios baseados em LDAP. Ainda, o modelo do PFIM requer informações sobre os dispositivos que implementam as políticas que podem ser mapeadas por objetos definidos pelo CIM. O CIM fornece modelos que permitem o armazenamento destas informações de forma padronizada em repositórios LDAP.

Estes dois últimos aspectos são importantes à medida que há diversas situações onde um mesmo conjunto de regras de políticas deve ser implementado em roteadores e *firewalls* que implementam as funcionalidades de filtros de pacotes, de diferentes fabricantes, com diferentes tecnologias. Em um caso extremo, considere a possibilidade da instalação de *Personal Firewalls* em cada um dos computadores de uma rede. A cada inicialização, seriam carregadas as regras de filtragem referentes a cada um dos papéis assumidos por esses computadores. A grande vantagem é que o gerenciamento das regras é centralizado através de uma ferramenta de gerenciamento das políticas. Dessa forma, fica caracterizado um novo tipo de *firewall* distribuído, diferente da estratégia apresentada por [BEL99] no Capítulo 2.

Esta facilidade de distribuição das regras pode ser alcançada com a opção da arquitetura PDP/PEP no modelo *provisioning*. Entretanto, a complexidade apresentada por este mecanismo forçou a sua definição a cargo de um trabalho futuro.

A proposta de armazenamento de regras formadas for condições e ações reusáveis abre as portas para a extensão do modelo proposto pelo PFIM para configuração de outras funcionalidades como, por exemplo, segurança em redes virtuais privadas (VPN) [TOK99]. Ao se conectar, o cliente VPN poderia informar o seu papel ao servidor que retorna os dados de configuração de volta ao cliente. Esses dados são função do nível de autorização do cliente. Esta estratégia poderia prevenir que usuários não autorizados utilizem canais VPN para acessar servidores e informações da rede interna.

Por fim, também é preciso mencionar a possibilidade de se introduzir aspectos temporais no serviço de controle de acesso baseado em filtros de pacotes. Tal proposta introduz a possibilidade de se definir filtros de pacotes dinâmicos que podem variar ao longo do dia ou mesmo em dias diferentes. Esquemas com esta característica permitem aos administradores controlar, por exemplo, o acesso de usuários internos a páginas *web* durante

determinados horários. Também é possível, utilizando esta técnica, priorizar o acesso aos recursos da rede, estratégia importante no caso de acessos muitos concorridos.

Referências Bibliográficas

[BAR96] Barth, C; Erwin,B.; Howe, C. D.; Elliot, S. *What's beyond firewalls? The Forrester Report*, Novembro, 1996.

[BAR99] Bartal, Y.; Mayer, A.; Nissim, K.; Wool, A., *Firmato: A Novel Firewall Management Toolkit*, *IEEE Symposium on Security and Privacy*, Oakland, Califórnia, Maio, 1999. URL: http://www.computer.org/proceedings/s&p/0176/01760017abs.htm

[BEL99] Bellovin, S. M., *Distributed Firewalls*, 1999. URL: http://www.research.att.com/~smb/papers/distfw.pdf

[BUP99] Bumpus, W.; Strassner, J.; Weiss, W. *The DEN-CIM Connection: A Roadmap to Directory-Enabled Networks*, 1999. URL: http://www.dmtf.org

[CCI88] *The Directory: Overview of Concepts, Models and Service. CCITT Recommendation X.500*, 1988.

[COM95a] Comer, D.E., *Internetworking with TCP/IP: Volume I Principles, Protocols, and Architecture*, Prentice-Hall, Inc., 1995.

[COM95b] Comer, D.E.; Stevens, D.L., *Internetworking with TCP/IP: Volume II Design Implementation and Internals*, Prentice-Hall, Inc., 1995.

[DMT00] DMTF – *CIM Specification Version 2.4*, 2000. URL: http://www.dmtf.org

[DMT02] DMTF – *CIM Core Model V2.5 LDAP Mapping Specification*, 2002 URL: http://www.dmtf.org

[FEN02] Fenger, R.; Hedge, H.; Larson, D.; Sahita, H., *Simplifying Support of New Network Services Using COPS-PR, White Paper*, 2002. URL: ftp://download.intel.com/labs/manage/cops/download/simpnetservice.pdf

[FIN01] Fine, M.; McCloghrie , K.; Seligson, J.; Chan, K.; Hahn, S.; Bell, C.; Smith, A.; Reichmeyer, F., *Differentiated Services Quality of Service Policy Information Base. Internet Draft*, 2001. URL: http://www.cs.utk.edu/~moore/ID-PDF/draft-ietf-diffserv-pib-05.pdf

[IOA00] Ioannidis, S.; Keromytis, A. D.; Bellovin, S. M.; Smith, J. M., *Implementing a Distributed Firewall*, 2000. URL: http://www1.cs.columbia.edu/~angelos/Papers/df.pdf

[JAM99] Jamhour, E., *Proposta de um Serviço para Controle de Acesso em Redes Globais*, Trabalho Apresentado em Concurso para Professor Titular PUCPR, Novembro, 1999.

[JAS01] Jason, J.; Rafalow, L.; Vyncke, E., *Ipsec Configuration Policy Model, Internet Draft*, 2001.
URL: http://www.ietf.org/proceedings/01aug/I-D/draft-ietf-ipsp-config-policy-model-03.txt

[MOO01] B. Moore, and D. Durham, J. Halpern, J. Strassner, A. Westerinen, W. Weiss, *Information Model for Describing Network Device QoS Datapath Mechanisms, Internet Draft*, 2001.
URL: http://www.ietf.org/proceedings/01dec/I-D/draft-ietf-policy-qos-device-info-model-06.txt

[MOO02] Moore, B; Ellenson, E; Strassner, J., *Policy Core LDAP Schema (PCLS), Policy Framework Working Group, Internet Draft*, 2002.
URL: http://www.ietf.org/internet-drafts/draft-ietf-policy-core-schema-15.txt

[NAB03] Nabhen, R.C., *RBPIM: Um Modelo de Políticas de Segurança Baseado em Papéis*, 2003. URL: http://www.ppgia.pucpr.br/ensino/defesas/Dissertacao_Ricardo_C_Nabhen.pdf

[OTT00] Ottensmeyer, J.; Bokaemper, M.; Roeber, K., *A Filtering Policy Information Base (PIB) for Edge Router Filtering Services and Provisioning via COPS-PR, Internet Draft*, 2000. URL: http://www.max.franken.de/ids/draft-otty-cops-pr-filter-pib-00.txt

[POL02] Polirakis, A.; Boutaba, R., *The Meta-Policy Information Base. IEEE Network Magazine, special issue on Policy-based Networking*, Vol.16, No. 2, pág. 40-48, 2002.

[QOS99] *Introduction to QoS Policies, White Paper*, 1999.
URL: http://www.csd.uwo.ca/faculty/hanan/cs434/qos1.pdf

[REY03] Reyes, A.; Barba, A.; Moron, D.; Brunner, M.; Pana, M., *Policy Core Extension LDAP Schema (PCELS), Internet Draft*, 2003.
URL: http://www.ietf.org/internet-drafts/draft-reyes-policy-core-ext-schema-03.txt

[RFC1558] Howes, T., *A String Representation of LDAP Search Filters*, 1993.

[RFC1777] Yeong, W.; Howes, T.; Kille, S., *Lightweight Directory Access Protocol*, 1995.

[RFC1959] Howes, T.; Smith, M., *An LDAP URL Format*, 1996.

[RFC3514] Bellovin, S., *The Security Flag in the IPv4 Header*, 2003.

[RFC2252] Wahl, M.; Coulbeck, A.; Howes, T.; Kille, S., *Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions*, 1997.

[RFC2401] Kent, S.; Atkinson, R., *Security Architecture for the Internet Protocol*, 1998.

[RFC2704] Blaze, M.; Feigenbaum, J.; Ioannidis, J.; Keromytis, A., *The KeyNote Trust-Management System Version 2*, 1999.

[RFC2748] Durham, D.; Boyle, J.; Cohen, R.; Herzog, S.; Rajan, R.; Sastry, A., *The COPS (Common Open Policy Service) Protocol*, 2000.

[RFC2753] Yavatkar, R.; Pendarakis, D.; Guerin, R., *A Framework for Policy-based Admission Control*, 2000.

[RFC3060] Moore, B.; Ellesson, E.; Strassner, J.; Westerinen, A., *A Framework for Policy-based Admission Control*, 2000.

[RFC3084] Chan, K., *et al*, *COPS Usage for Policy Provisioning (COPS-PR)*, 2001.

[RFC3159] McCloghrie, K.; Fine, M.; Seligson, J.; Chan, K.; Hahn, S.; Sahita, R.; Smith, A.; Reichmeyer, F., *Structure of Policy Provisioning Information (SPPI)*, 2001.

[RFC3198] Westerinen, A.; *et al*, *Terminology for Policy-Based Management*, 2001.

[RFC3460] Moore, B., *Policy Core Information Model (PCIM) Extensions*, 2003.

[SAL02] Salsano, S.; Veltri, L., *QoS Control by Means of COPS to Support SIP-Based Applications*. *IEEE Network Magazine*, pág. 27-33, Mar/Abr, 2002.

[SHE00] Shepard, S. J., *Policy-based Networks: Hype and Hope*, 2000. *IT Professional*, Vol. 2, No. 1, Janeiro 2000.

[SNI03] Snir, Y.; Ramberg, Y.; Strassner, J.; Cohen, R.; *Policy QoS Information Model*, *Internet Draft*, 2003. URL: http://www.potaroo.net/ietf/ids/draft-ietf-policy-qos-info-model-05.txt

[TOK99] Toktar, E.; Neves, E.; Costa, J.E., *Redes Virtuais Privadas*, 1999 URL: http://www.ppgia.pucpr.br/~alcides/Research/dissertations/VPN_Toktar_Neves_daCosta.zip

[XIA99] Xiao, X.; Ni, L.M., *Internet QoS: A Big Picture*, 1999. URL: http://www.cse.msu.edu/~xiaoxipe/papers/inetqos/inet.qos.bigpicture.pdf

[WAL03] Waldbusser, S.; Saperia, J.; Hongal, T., *Policy Based Management MIB*, 2003, *Internet Draft*, 2003. URL: http://www.ietf.org/internet-drafts/draft-ietf-snmpconf-pm-14.txt

[WON01] Wong, J.; Hunt, R., *Policy Based Network Management*, 2001. - *Department of Accountancy, Finance, and Information Systems. University of Canterbury. Christchurch.*

[ZWI00] Zwicky, E.D.; Cooper, S.; Chapman, D.B., *Building Internet Firewalls 2nd Edition*, O'Reilly, 2000

ANEXO A – MAPEAMENTO DAS CLASSES DO PFIM NO LDAP

Classes do PCIMe/PFIM Mapeadas no LDAP

Classe LDAP	Descrição	Atributos
pcimPolicy (ab)	Classe base que descreve todas as classes relacionadas às políticas	*cn*
pcimPolicySet (ab)	Classe que representa um conjunto de políticas relacionadas.	*pcimPolicySetName* *pcimDecisionStrategy* *pcimRoles* *pcimPolicySetList*
pcimGroup (ab)	Classe que representa um container de regras de políticas relacionadas.	*pcimGroupName*
pfimGroupAuxClass (aux)	Classe auxiliar que reúne um conjunto de regras de políticas relacionadas.	*nenhum*
pfimGroupInstance (struc)	Classe estrutural que reúne um conjunto de regras de políticas relacionadas.	*nenhum*
pcimPolicyRule (ab)	Classe base para a representação da semântica “se condição então ação” associada a uma regra de política.	*pcimRuleName* *pcimRuleEnabled* *pcimConditionListType* *pcimConditionList* *pcimActionList* *pcimRuleValidityPeriodList* *pcimRuleUsage* *pcimRuleMandatory* *pcimSequencedActions* *pcimExecutionStrategy*
pfimRuleAuxClass (aux)	Classe auxiliar para a representação da semântica “se condição então ação” associada a uma regra de política.	*nenhum*
pfimRuleInstance (struc)	Classe estrutural para a representação da semântica “se condição então ação” associada a uma regra de política.	*nenhum*
pcimConditionAuxClass (aux)	Classe auxiliar que representa uma condição a ser avaliada.	*nenhum*
pcimActionAuxClass (aux)	Classe auxiliar que representa uma ação a ser executada.	*nenhum*
pcimSimpleConditionAuxClass (aux)	Classe auxiliar que representa uma condição simples.	*pcimVariableDN* *pcimValueDN*

pcimCompoundConditionAuxClass (aux)	Classe auxiliar que representa uma condição comp lexa.	*pcimConditionListType pcimConditionList*
pcimCompoundFilterAuxClass (aux)	Classe auxiliar que representa uma condição complexa para filtros de pacotes.	*pcimIsMirrored*
pcimSimpleActionAuxClass (aux)	Classe auxiliar que representa uma ação simples.	*pcimVariableDN pcimValueDN*
pcimCompoundActionAuxClass (aux)	Classe auxiliar que representa uma ação complexa	*pcimActionList pcimSequencedActions pcimExecutionStrategy*
pcimRoleCollection (struc)	Classe estrutural utilizada para agrupar entradas no diretório que compartilham o mesmo papel.	*pcimRole pcimRoleCollectionName pcimElementList*
pcimReusableContainer (ab)	Classe abstrata para representar a informação de políticas reusáveis.	*pcimReusableContainerName pcimReusableContainerList*
pcimReusableContainerAuxClass (aux)	Classe auxiliar para agregar informação de políticas reusáveis.	*nenhum*
pcimReusableContainerInstance (struc)	Classe estrutural para agregar informação de políticas reusáveis.	*nenhum*

Classes de Variáveis do PCIMe/PFIM Mapeadas no LDAP

Classe LDAP	Descrição	Atributos
pcimVariable (ab)	Classe base para a representação de variáveis nas quais o valor contido deve ser comparado a um valor de referência.	*pcimVariableName pcimExpectedValueList*
pcimExplicitVariableAuxClass (aux)	Classe auxiliar que representa variáveis de políticas explícitas no contexto do esquema do CIM.	*pcimVariableModelClass pcimVariableModelProperty*
pcimImplicitVariableAuxClass (aux)	Classe auxiliar que representa variáveis de políticas implícitas as quais a avaliação depende do contexto em que se encontram. Suas subclasse especificam os tipos de dados e a semântica das variáveis.	*pcimExpectedValueTypes*
pcimSourceIPv4VariableAuxClass (aux)	Classe auxiliar que representa um endereço IPv4 de origem.	*nenhum*
pcimDestinationIPv4VariableAuxClass (aux)	Classe auxiliar que representa um endereço IPv4 de destino.	*nenhum*
pcimSourcePortVariableAuxClass (aux)	Classe auxiliar que representa uma porta de origem.	*nenhum*
pcimDestinationPortVariableAuxClass (aux)	Classe auxiliar que representa uma porta de destino.	*nenhum*

pcimFlowDirectionVariableAuxClass (aux)	Classe auxiliar que representa a direção do fluxo da mensagem.	*nenhum*
pfimServiceTypeVarAuxClass (aux)	Classe auxiliar que representa o nome do serviço (http, ftp, etc.)	*nenhum*
pfimAckBitVariableAuxClass (aux)	Classe auxiliar que representa o flag ACK de mensagens TCP.	*nenhum*

Classes de Valores do PCIMe/PFIM Mapeadas no LDAP

Classe LDAP	Descrição	Atributos
pcimValueAuxClass (aux)	Classe auxiliar base para a representação de um valor que pode ser confrontado ou configurado por uma variável específica.	*pcimValueName*
pcimIntegerValueAuxClass	Classe auxiliar base para a representação de um valor inteiro.	*pcimIntegerList*
pcimIPv4AddrValueAuxClass	Classe auxiliar base para a representação de um endereço IPv4.	*pcimIPv4AddrList*
pcimStringValueAuxClass	Classe auxiliar base para a representação de uma string	*pcimStringList*

ANEXO B – ARQUIVOS LDIF DO ESTUDO DE CASO DA REDE XNC

O *LDAP Data Interchange Format* (LDIF) é um formato para definir entradas de diretórios no formato texto. Uma entrada *LDIF* especifica as seguintes informações:

- O *distinguished name* (DN) que identifica a entrada: O DN de uma entrada identifica a sua localização no diretório. Duas entradas nunca podem ter o mesmo DN.
- As classes do esquema (*object classes*) utilizadas na entrada: Uma classe define os atributos obrigatórios e permitidos para uma entrada.
- Os atributos e seus valores correspondentes: Os atributos e seus valores especificam a informação sobre uma entrada.

A listagem apresentada a seguir retrata o conteúdo do arquivo Rede_XNC.ldif, o qual contém as entradas utilizadas no estudo de caso.

```
dn:o=XNC,dc=com
objectClass:top
objectClass:organization
o:XNC

dn:ou=Network,o=XNC,dc=com
ou:Network
objectClass:organizationalunit
objectClass:top

dn:pfimdevicename=Router,ou=Network,o=XNC,dc=com
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1LogicalDevice
objectClass:pfimdevice
pfimDeviceName:Router

dn:pfimdevicename=ServidorHTTP,ou=Network,o=XNC,dc=com
dlmDeviceID:200.246.5.200
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1LogicalDevice
```

objectClass:pfimdevice
pfimDeviceName:ServidorHTTP

dn:pfimdevicename=ServidorFTP,ou=Network,o=XNC,dc=com
dlmDeviceID:200.246.5.201
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1LogicalDevice
objectClass:pfimdevice
pfimDeviceName:ServidorFTP

dn:pfimdevicename=ServidorSMTP,ou=Network,o=XNC,dc=com
dlmDeviceID:192.168.200.253
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1LogicalDevice
objectClass:pfimdevice
pfimDeviceName:ServidorSMTP

dn:pcimrolecollectionname=ServidoresHTTP,ou=Network,o=XNC,dc=com
pcimRoleCollectionName:ServidoresHTTP
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimrolecollection
pcimRole:httpserver
pcimElementList:pfimdevicename=ServidorHTTP,ou=Network,o=XNC,dc=com

dn:pcimrolecollectionname=ServidoresFTP,ou=Network,o=XNC,dc=com
pcimRoleCollectionName:ServidoresFTP
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimrolecollection
pcimRole:ftpserver
pcimElementList:pfimdevicename=ServidorFTP,ou=Network,o=XNC,dc=com

dn:pcimrolecollectionname=ServidoresSMTP,ou=Network,o=XNC,dc=com
pcimRoleCollectionName:ServidoresSMTP
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimrolecollection
pcimRole:smtpserver
pcimElementList:pfimdevicename=ServidorSMTP,ou=Network,o=XNC,dc=com

dn:pfimdevicename=LAN,pfimdevicename=Router,ou=Network,o=XNC,dc=com
dlmDeviceID:192.168.200.254
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1LogicalDevice
objectClass:pfimdevice
pfimDeviceName:LAN

dn:ou=RepositorioCC,o=XNC,dc=com
ou:RepositorioCC
objectClass:top
objectClass:organizationalunit
objectClass:pcimreusablecontainerauxclass
objectClass:pcimReusableContainer
objectClass:dlm1AdminDomain
objectClass:dlm1System
objectClass:dlm1LogicalElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1ManagedElement

dn:ou=RepositorioCS,o=XNC,dc=com
ou:RepositorioCS
objectClass:top
objectClass:organizationalunit
objectClass:pcimreusablecontainerauxclass
objectClass:pcimReusableContainer
objectClass:dlm1AdminDomain
objectClass:dlm1System
objectClass:dlm1LogicalElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1ManagedElement

dn:pcimreusablecontainername=FlowDirection_IN,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:FlowDirection_IN
pcimStringList:IN
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimflowdirectionvariableauxclass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimstringvalueauxclass
objectClass:pcimValueAuxClass
pcimExpectedValueTypes:pcimstringvalueauxclass
pcimExpectedValueList:IN,OUT

dn:pcimreusablecontainername=FlowDirection_OUT,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:FlowDirection_OUT
pcimStringList:OUT
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimflowdirectionvariableauxclass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimstringvalueauxclass

objectClass:pcimValueAuxClass
pcimExpectedValueTypes:pcimstringvalueauxclass
pcimExpectedValueList:IN,OUT

dn:pcimreusablecontainername=Protocolo_HTTP,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:Protocolo_HTTP
pcimStringList:HTTP
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pfimServiceTypeVarAuxClass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimstringvalueauxclass
objectClass:pcimValueAuxClass
pcimExpectedValueTypes:pcimstringvalueauxclass
pcimExpectedValueList:HTTP,FTP,SMTP,POP3,etc.

dn:pcimreusablecontainername=Protocolo_FTP,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:Protocolo_FTP
pcimStringList:FTP
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pfimServiceTypeVarAuxClass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimstringvalueauxclass
objectClass:pcimValueAuxClass
pcimExpectedValueTypes:pcimstringvalueauxclass
pcimExpectedValueList:HTTP,FTP,SMTP,POP3,etc.

dn:pcimreusablecontainername=Protocolo_SMTP,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:Protocolo_SMTP
pcimStringList:SMTP
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pfimServiceTypeVarAuxClass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimstringvalueauxclass
objectClass:pcimValueAuxClass
pcimExpectedValueTypes:pcimstringvalueauxclass

pcimExpectedValueList:HTTP,FTP,SMTP,POP3

dn:pcimreusablecontainername=SourceIP_LAN,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:SourceIP_LAN
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimsourceipv4variableauxclass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimipv4addrvalueauxclass
objectClass:pcimValueAuxClass
pcimExpectedValueTypes:pcimIPv4AddrValueAuxClass
pcimIPv4AddrList:192.168.200.0/24

dn:pcimreusablecontainername=SourceIP_wAN,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:SourceIP_wAN
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimsourceipv4variableauxclass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimipv4addrvalueauxclass
objectClass:pcimValueAuxClass
pcimExpectedValueTypes:pcimIPv4AddrValueAuxClass
pcimIPv4AddrList:0.0.0.0

dn:pcimreusablecontainername=SourceIP_HTTPServer,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:SourceIP_HTTPServer
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimsourceipv4variableauxclass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimipv4addrvalueauxclass
objectClass:pcimValueAuxClass
pcimExpectedValueTypes:pcimIPv4AddrValueAuxClass
pcimIPv4AddrLis t:200.246.5.200

dn:pcimreusablecontainername=SourceIP_FTPServer,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:SourceIP_FTPServer
objectClass:top

objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimsourceipv4variableauxclass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimipv4addrvalueauxclass
objectClass:pcimValueAuxClass
pcimExpectedValueTypes:pcimIPv4AddrValueAuxClass
pcimIPv4AddrList:200.246.5.201

dn:pcimreusablecontainername=SourceIP_SMTPServer,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:SourceIP_SMTPServer
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimsourceipv4variableauxclass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimipv4addrvalueauxclass
objectClass:pcimValueAuxClass
pcimExpectedValueTypes:pcimIPv4AddrValueAuxClass
pcimIPv4AddrList:200.246.5.202

dn:pcimreusablecontainername=DestIP_LAN,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:DestIP_LAN
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimdestinationipv4variableauxclass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimipv4addrvalueauxclass
objectClass:pcimValueAuxClass
pcimExpectedValueTypes:pcimIPv4Addrvalueauxclass
pcimIPv4AddrList:192.168.200.0/24

dn:pcimreusablecontainername=DestIP_wAN,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:DestIP_wAN
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain

objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimdestinationipv4variableauxclass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimipv4addrvalueauxclass
objectClass:pcimValueAuxClass
pcimExpectedValueTypes:pcimIPv4Addrvalueauxclass
pcimIPv4AddrList:0.0.0.0

dn:pcimreusablecontainername=DestIP_HTTPServer,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:DestIP_HTTPServer
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimdestinationipv4variableauxclass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimipv4addrvalueauxclass
objectClass:pcimValueAuxClass
pcimExpectedValueTypes:pcimIPv4Addrvalueauxclass
pcimIPv4AddrList:200.246.5.200

dn:pcimreusablecontainername=DestIP_FTPServer,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:DestIP_FTPServer
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimdestinationipv4variableauxclass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimipv4addrvalueauxclass
objectClass:pcimValueAuxClass
pcimExpectedValueTypes:pcimIPv4Addrvalueauxclass
pcimIPv4AddrList:200.246.5.201

dn:pcimreusablecontainername=DestIP_SMTPServer,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:DestIP_SMTPServer
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimdestinationipv4variableauxclass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable

objectClass:pcimipv4addrvalueauxclass
objectClass:pcimValueAuxClass
pcimExpectedValueTypes:pcimIPv4Addrvalueauxclass
pcimIPv4AddrList:200.246.5.202

dn:pcimreusablecontainername=SourcePort_21,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:SourcePort_21
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimSourcePortVariableAuxClass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimintegervalueauxclass
objectClass:pcimValueAuxClass
pcimIntegerList:21
pcimExpectedValueTypes:pcimintegervalueauxclass

dn:pcimreusablecontainername=SourcePort_25,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:SourcePort_25
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimSourcePortVariableAuxClass
objectClass:pcimI mplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimintegervalueauxclass
objectClass:pcimValueAuxClass
pcimIntegerList:25
pcimExpectedValueTypes:pcimintegervalueauxclass

dn:pcimreusablecontainername=SourcePort_80,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:SourcePort_80
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimSourcePortVariableAuxClass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimintegervalueauxclass
objectClass:pcimValueAuxClass
pcimIntegerList:80
pcimExpectedValueTypes:pcimintegervalueauxclass

dn:pcimreusablecontainername=DestPort_21,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:DestPort_21
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimDestinationPortVariableAuxClass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimintegervalueauxc lass
objectClass:pcimValueAuxClass
pcimIntegerList:21
pcimExpectedValueTypes:pcimintegervalueauxclass

dn:pcimreusablecontainername=DestPort_25,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:DestPort_25
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimDestinationPortVariableAuxClass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimintegervalueauxclass
objectClass:pcimValueAuxClass
pcimIntegerList:25
pcimExpectedValueTypes:pcimintegervalueauxclass

dn:pcimreusablecontainername=DestPort_80,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:DestPort_80
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimDestinationPortVariableAuxClass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimintegervalueauxclass
objectClass:pcimValueAuxClass
pcimIntegerList:80
pcimExpectedValueTypes:pcimintegervalueauxclass

dn:pcimreusablecontainername=ACK_BIT_0,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:ACK_BIT_0
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement

objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pfimAckBitVariableAuxClass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimintegervalueauxclass
objectClass:pcimValueAuxClass
pcimIntegerList:0
pcimExpectedValueTypes:pcimintegervalueauxclass

dn:pcimreusablecontainername=ACK_BIT_1,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:ACK_BIT_1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pfimAckBitVariableAuxClass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimintegervalueauxclass
objectClass:pcimValueAuxClass
pcimIntegerList:1
pcimExpectedValueTypes:pcimintegervalueauxclass

dn:pcimreusablecontainername=FlowDirection_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:FlowDirection_ANY
pcimStringList:ANY
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimflowdirectionvariableauxclass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimstringvalueauxclass
objectClass:pcimValueAuxClass
pcimExpectedValueTypes:pcimstringvalueauxclass
pcimExpectedValueList:ANY

dn:pcimreusablecontainername=Protocolo_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:Protocolo_ANY
pcimStringList:ANY
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain

objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pfimServiceTypeVarAuxClass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimstringvalueauxclass
objectClass:pcimValueAuxClass
pcimExpectedValueTypes:pcimstringvalueauxclass
pcimExpectedValueList:ANY

dn:pcimreusablecontainername=SourceIP_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:SourceIP_ANY
pcimStringList:ANY
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pfimServiceTypeVarAuxClass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimstringvalueauxclass
objectClass:pcimValueAuxClass
pcimExpectedValueTypes:pcimstringvalueauxclass
pcimExpectedValueList:ANY

dn:pcimreusablecontainername=DestIP_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:DestIP_ANY
pcimStringList:ANY
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pfimServiceTypeVarAuxClass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimstringvalueauxclass
objectClass:pcimValueAuxClass
pcimExpectedValueTypes:pcimstringvalueauxclass
pcimExpectedValueList:ANY

dn:pcimreusablecontainername=SourcePort_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:SourcePort_ANY
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimSourcePortVariableAuxClass

objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimintegervalueauxclass
objectClass:pcimValueAuxClass
pcimIntegerList:1-65535
pcimExpectedValueTypes:pcimintegervalueauxclass

dn:pcimreusablecontainername=DestPort_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:DestPort_ANY
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimDestinationPortVariableAuxClass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimintegervalueauxclass
objectClass:pcimValueAuxClass
pcimIntegerList:1-65535
pcimExpectedValueTypes:pcimintegervalueauxclass

dn:pcimreusablecontainername=ACK_BIT_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:ACK_BIT_ANY
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pfimAckBitVariableAuxClass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimintegervalueauxclass
objectClass:pcimValueAuxClass
pcimIntegerList:2
pcimExpectedValueTypes:pcimintegervalueauxclass

dn:pcimreusablecontainername=ACTION_PASS,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:ACTION_PASS
pcimStringList:PASS
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pfimPacketActionVarAuxClass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimstringvalueauxclass
objectClass:pcimValueAuxClass

pcimExpectedValueTypes:pcimstringvalueauxclass
pcimExpectedValueList:PASS

dn:pcimreusablecontainername=ACTION_DROP,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:ACTION_DROP
pcimStringList:DROP
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pfimPacketActionVarAuxClass
objectClass:pcimImplicitVariableAuxClass
objectClass:pcimVariable
objectClass:pcimstringvalueauxclass
objectClass:pcimValueAuxClass
pcimExpectedVa lueTypes:pcimstringvalueauxclass
pcimExpectedValueList:DROP

dn:pcimreusablecontainername=CCrule1httpserver,ou=RepositorioCC,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=FlowDirection_OUT,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=Protocolo_HTTP,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourceIP_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestIP_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourcePort_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestPort_80,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=ACK_BIT_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:CCrule1httpserver
pcimConditionListType:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimcompoundfilterauxclass
objectClass:pcimCompoundConditionAuxClass
objectClass:pcimConditionAuxClass

dn:pcimreusablecontainername=CCrule2httpserver,ou=RepositorioCC,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=FlowDirection_IN,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=Protocolo_HTTP,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourceIP_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestIP_LAN,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourcePort_80,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestPort_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=ACK_BIT_1,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:CCrule2httpserver
pcimConditionListType:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System

objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimcompoundfilterauxclass
objectClass:pcimCompoundConditionAuxClass
objectClass:pcimConditionAuxClass

dn:pcimreusablecontainername=CCrule3httpserver,ou=RepositorioCC,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=FlowDirection_IN,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=Protocolo_HTTP,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourceIP_WAN,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestIP_HTTPServer,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourcePort_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestPort_80,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=ACK_BIT_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:CCrule3httpserver
pcimConditionListType:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimcompoundfilterauxclass
objectClass:pcimCompoundConditionAuxClass
objectClass:pcimConditionAuxClass

dn:pcimreusablecontainername=CCrule1ftpserver,ou=RepositorioCC,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=FlowDirection_IN,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=Protocolo_FTP,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourceIP_LAN,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestIP_FTPServer,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourcePort_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestPort_21,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=ACK_BIT_0,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:CCrule1ftpserver
pcimConditionListType:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimcompoundfilterauxclass
objectClass:pcimCompoundConditionAuxClass
objectClass:pcimConditionAuxClass

dn:pcimreusablecontainername=CCrule2ftpserver,ou=RepositorioCC,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=FlowDirection_IN,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=Protocolo_FTP,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourceIP_WAN,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestIP_FTPServer,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourcePort_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestPort_21,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=ACK_BIT_0,ou=RepositorioCS,o=XNC,dc=com

pcimReusableContainerName:CCrule2ftpserver
pcimConditionListType:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimcompoundfilterauxclass
objectClass:pcimCompoundConditionAuxClass
objectClass:pcimConditionAuxClass

dn:pcimreusablecontainername=CCrule3ftpserver,ou=RepositorioCC,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=FlowDirection_OUT,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=Protocolo_FTP,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourceIP_FTPServer,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestIP_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourcePort_21,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestPort_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=ACK_BIT_1,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:CCrule3ftpserver
pcimConditionListType:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimcompoundfilterauxclass
objectClass:pcimCompoundConditionAuxClass
objectClass:pcimConditionAuxClass

dn:pcimreusablecontainername=CCruleDefault,ou=RepositorioCC,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=FlowDirection_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=Protocolo_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourceIP_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestIP_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourcePort_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestPort_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=ACK_BIT_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:CCruleDefault
pcimConditionListType:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimcompoundfilterauxclass
objectClass:pcimCompoundConditionAuxClass
objectClass:pcimConditionAuxClass

dn:pcimreusablecontainername=CCrule1smtpserver,ou=RepositorioCC,o=XNC,dc=com

pcimConditionList:pcimreusablecontainername=FlowDirection_OUT,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=Protocolo_SMTP,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourceIP_SMTPServer,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestIP_WAN,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourcePort_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestPort_25,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=ACK_BIT_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:CCrule1smtpserver
pcimConditionListType:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimcompoundfilterauxclass
objectClass:pcimCompoundConditionAuxClass
objectClass:pcimConditionAuxClass

dn:pcimreusablecontainername=CCrule2smtpserver,ou=RepositorioCC,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=FlowDirection_IN,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=Protocolo_SMTP,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourceIP_WAN,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestIP_SMTPServer,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourcePort_25,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestPort_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=ACK_BIT_1,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:CCrule2smtpserver
pcimConditionListType:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimcompoundfilterauxclass
objectClass:pcimCompoundConditionAuxClass
objectClass:pcimConditionAuxClass

dn:pcimreusablecontainername=ACrule_pass,ou=RepositorioCC,o=XNC,dc=com
pcimValueDN:pcimreusablecontainername=ACTION_PASS,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:ACrule_pass
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimsimpleactionauxclass
objectClass:pcimActionAuxClass

dn:pcimreusablecontainername=ACrule_drop,ou=RepositorioCC,o=XNC,dc=com
pcimValueDN:pcimreusablecontainername=ACTION_DROP,ou=RepositorioCS,o=XNC,dc=com

pcimReusableContainerName:ACrule_drop
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimsimpleactionauxclass
objectClass:pcimActionAuxClass

dn:pcimreusablecontainername=CCrule4httpserver,ou=RepositorioCC,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=FlowDirection_OUT,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=Protocolo_HTTP,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourceIP_HTTPServer,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestIP_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourcePort_80,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestPort_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=ACK_BIT_1,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:CCrule4httpserver
pcimConditionListType:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimcompoundfilterauxclass
objectClass:pcimCompoundConditionAuxClass
objectClass:pcimConditionAuxClass

dn:pcimreusablecontainername=CCrule4ftpserver,ou=RepositorioCC,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=FlowDirection_OUT,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=Protocolo_FTP,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourceIP_FTPServer,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestIP_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=SourcePort_21,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=DestPort_ANY,ou=RepositorioCS,o=XNC,dc=com
pcimConditionList:pcimreusablecontainername=ACK_BIT_1,ou=RepositorioCS,o=XNC,dc=com
pcimReusableContainerName:CCrule4ftpserver
pcimConditionListType:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1System
objectClass:dlm1AdminDomain
objectClass:pcimReusableContainer
objectClass:pcimreusablecontainerinstance
objectClass:pcimcompoundfilterauxclass
objectClass:pcimCompoundConditionAuxClass
objectClass:pcimConditionAuxClass

dn:ou=GroupOfRulesHTTPServer,o=XNC,dc=com
pcimDecisionStrategy:1
pcimRoles:httpserver

ou:GroupOfRulesHTTPServer
objectClass:top
objectClass:organizationalunit
objectClass:pfimgroupauxclass
objectClass:pcimGroup
objectClass:pcimPolicySet
objectClass:pcimPolicy
objectClass:dlm1ManagedElement
pcimGroupName:RulesHTTPServer
pcimPolicySetList:dn:pcimrulename=Rule1HTTPServer,ou=GroupOfRulesHTTPServer,o=XNC,dc=com
pcimPolicySetList:dn:pcimrulename=Rule2HTTPServer,ou=GroupOfRulesHTTPServer,o=XNC,dc=com
pcimPolicySetList:dn:pcimrulename=Rule3HTTPServer,ou=GroupOfRulesHTTPServer,o=XNC,dc=com
pcimPolicySetList:dn:pcimrulename=Rule4HTTPServer,ou=GroupOfRulesHTTPServer,o=XNC,dc=com
pcimPolicySetList:dn:pcimrulename=Rule5HTTPServer,ou=GroupOfRulesHTTPServer,o=XNC,dc=com

dn:ou=GroupOfRulesFTPServer,o=XNC,dc=com
pcimDecisionStrategy:1
pcimRoles:ftpserver
ou:GroupOfRulesFTPServer
objectClass:top
objectClass:organizationalunit
objectClass:pfimgroupauxclass
objectClass:pcimGroup
objectClass:pcimPolicySet
objectClass:pcimPolicy
objectClass:dlm1ManagedElement
pcimGroupName:RulesFTPServer
pcimPolicySetList:dn:pcimrulename=Rule1FTPServer,ou=GroupOfRulesFTPServer,o=XNC,dc=com
pcimPolicySetList:dn:pcimrulename=Rule2FTPServer,ou=GroupOfRulesFTPServer,o=XNC,dc=com
pcimPolicySetList:dn:pcimrulename=Rule3FTPServer,ou=GroupOfRulesFTPServer,o=XNC,dc=com
pcimPolicySetList:dn:pcimrulename=Rule4FTPServer,ou=GroupOfRulesFTPServer,o=XNC,dc=com
pcimPolicySetList:dn:pcimrulename=Rule5FTPServer,ou=GroupOfRulesFTPServer,o=XNC,dc=com

dn:ou=GroupOfRulesSMTPServer,o=XNC,dc=com
pcimDecisionStrategy:1
pcimRoles:smtpserver
ou:GroupOfRulesSMTPServer
objectClass:top
objectClass:organizationalunit
objectClass:pfimgroupauxclass
objectClass:pcimGroup
objectClass:pcimPolicySet
objectClass:pcimPolicy
objectClass:dlm1ManagedElement
pcimGroupName:RulesSMTPServer
pcimPolicySetList:dn:pcimrulename=Rule1SMTPServer,ou=GroupOfRulesSMTPServer,o=XNC,dc=com
pcimPolicySetList:dn:pcimrulename=Rule2SMTPServer,ou=GroupOfRulesSMTPServer,o=XNC,dc=com
pcimPolicySetList:dn:pcimrulename=Rule3SMTPServer,ou=GroupOfRulesSMTPServer,o=XNC,dc=com

dn:pcimrulename=Rule1HTTPServer,ou=GroupOfRulesHTTPServer,o=XNC,dc=com
pcimDecisionStrategy:1
pcimRoles:httpserver
pcimConditionListType:1
pcimExecutionStrategy:2
pcimPolicySetName:RulesHTTPServer
pcimRuleEnabled:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy

objectClass:pcimPolicySet
objectClass:pcimPolicyRule
objectClass:pfimruleinstance
pcimRuleName:Rule1HTTPServer

dn:pcimconditionname=ca,pcimrulename=Rule1HTTPServer,ou=GroupOfRulesHTTPServer,o=XNC,dc=com
pcimConditionGroupNumber:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimconditionassociation
pcimConditionName:ca
pcimConditionNegated:FALSE
pcimConditionDN:pcimreusablecontainername=CCrule1httpserver,ou=RepositorioCC,o=XNC,dc=com

dn:pcimactionname=aa,pcimrulename=Rule1HTTPServer,ou=GroupOfRulesHTTPServer,o=XNC,dc=com
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimactionassociation
pcimActionName:aa
pcimActionDN:pcimreusablecontainername=ACrule_pass,ou=RepositorioCC,o=XNC,dc=com
pcimActionOrder:1

dn:pcimrulename=Rule2HTTPServer,ou=GroupOfRulesHTTPServer,o=XNC,dc=com
pcimDecisionStrategy:1
pcimRoles:httpserver
pcimConditionListType:1
pcimExecutionStrategy:2
pcimPolicySetName:RulesHTTPServer
pcimRuleEnabled:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimPolicySet
objectClass:pcimPolicyRule
objectClass:pfimruleinstance
pcimRuleName:Rule2HTTPServer

dn:pcimconditionname=ca,pcimrulename=Rule2HTTPServer,ou=GroupOfRulesHTTPServer,o=XNC,dc=com
pcimConditionGroupNumber:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimconditionassociation
pcimConditionName:ca
pcimConditionNegated:FALSE
pcimConditionDN:pcimreusablecontainername=CCrule2httpserver,ou=RepositorioCC,o=XNC,dc=com

dn:pcimactionname=aa,pcimrulename=Rule2HTTPServer,ou=GroupOfRulesHTTPServer,o=XNC,dc=com
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimactionassociation
pcimActionName:aa
pcimActionDN:pcimreusablecontainername=ACrule_pass,ou=RepositorioCC,o=XNC,dc=com
pcimActionOrder:1

dn:pcimrulename=Rule3HTTPServer,ou=GroupOfRulesHTTPServer,o=XNC,dc=com

pcimDecisionStrategy:1
pcimRoles:httpserver
pcimConditionListType:1
pcimExecutionStrategy:2
pcimPolicySetName:RulesHTTPServer
pcimRuleEnabled:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimPolicySet
objectClass:pcimPolicyRule
objectClass:pfimruleinstance
pcimRuleName:Rule3HTTPServer

dn:pcimconditionname=ca,pcimrulename=Rule3HTTPServer,ou=GroupOfRulesHTTPServer,o=XNC,dc=com
pcimConditionGroupNumber:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimconditionassociation
pcimConditionName:ca
pcimConditionNegated:FALSE
pcimConditionDN:pcimreusablecontainername=CCrule3httpserver,ou=RepositorioCC,o=XNC,dc=com

dn:pcimactionname=aa,pcimrulename=Rule3HTTPServer,ou=GroupOfRulesHTTPServer,o=XNC,dc=com
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimactionassociation
pcimActionName:aa
pcimActionDN:pcimreusablecontainername=ACrule_pass,ou=RepositorioCC,o=XNC,dc=com
pcimActionOrder:1

dn:pcimrulename=Rule4HTTPServer,ou=GroupOfRulesHTTPServer,o=XNC,dc=com
pcimDecisionStrategy:1
pcimRoles:httpserver
pcimConditionListType:1
pcimExecutionStrategy:2
pcimPolicySetName:RulesHTTPServer
pcimRuleEnabled:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimPolicySet
objectClass:pcimPolicyRule
objectClass:pfimruleinstance
pcimRuleName:Rule4HTTPServer

dn:pcimconditionname=ca,pcimrulename=Rule4HTTPServer,ou=GroupOfRulesHTTPServer,o=XNC,dc=com
pcimConditionGroupNumber:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimconditionassociation
pcimConditionName:ca
pcimConditionNegated:FALSE
pcimConditionDN:pcimreusablecontainername=CCrule4httpserver,ou=RepositorioCC,o=XNC,dc=com

dn:pcimactionname=aa,pcimrulename=Rule4HTTPServer,ou=GroupOfRulesHTTPServer,o=XNC,dc=com

objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimactionassociation
pcimActionName:aa
pcimActionDN:pcimreusablecontainername=ACrule_pass,ou=RepositorioCC,o=XNC,dc=com
pcimActionOrder:1

dn:pcimrulename=Rule5HTTPServer,ou=GroupOfRulesHTTPServer,o=XNC,dc=com
pcimDecisionStrategy:1
pcimRoles:httpserver
pcimConditionListType:1
pcimExecutionStrategy:2
pcimPolicySetName:RulesHTTPServer
pcimRuleEnabled:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimPolicySet
objectClass:pcimPolicyRule
objectClass:pfimruleinstance
pcimRuleName:Rule5HTTPServer

dn:pcimconditionname=ca,pcimrulename=Rule5HTTPServer,ou=GroupOfRulesHTTPServer,o=XNC,dc=com
pcimConditionGroupNumber:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimconditionassociation
pcimConditionName:ca
pcimConditionNegated:FALSE
pcimConditionDN:pcimreusablecontainername=CCruledefault,ou=RepositorioCC,o=XNC,dc=com

dn:pcimactionname=aa,pcimrulename=Rule5HTTPServer,ou=GroupOfRulesHTTPServer,o=XNC,dc=com
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimactionassociation
pcimActionName:aa
pcimActionDN:pcimreusablecontainername=ACrule_drop,ou=RepositorioCC,o=XNC,dc=com
pcimActionOrder:1

dn:pcimrulename=Rule1FTPServer,ou=GroupOfRulesFTPServer,o=XNC,dc=com
pcimDecisionStrategy:1
pcimRoles:ftpserver
pcimConditionListType:1
pcimExecutionStrategy:2
pcimPolicySetName:RulesFTPServer
pcimRuleEnabled:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimPolicySet
objectClass:pcimPolicyRule
objectClass:pfimruleinstance
pcimRuleName:Rule1FTPServer

dn:pcimconditionname=ca,pcimrulename=Rule1FTPServer,ou=GroupOfRulesFTPServer,o=XNC,dc=com
pcimConditionGroupNumber:1

objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimconditionassociation
pcimConditionName:ca
pcimConditionNegated:FALSE
pcimConditionDN:pcimreusablecontainername=CCrule1ftpserver,ou=RepositorioCC,o=XNC,dc=com

dn:pcimactionname=aa,pcimrulename=Rule1FTPServer,ou=GroupOfRulesFTPServer,o=XNC,dc=com
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimactionassociation
pcimActionName:aa
pcimActionDN:pcimreusablecontainername=ACrule_pass,ou=RepositorioCC,o=XNC,dc=com
pcimActionOrder:1

dn:pcimrulename=Rule2FTPServer,ou=GroupOfRulesFTPServer,o=XNC,dc=com
pcimDecisionStrategy:1
pcimRoles:ftpserver
pcimConditionListType:1
pcimExecutionStrategy:2
pcimPolicySetName:RulesFTPServer
pcimRuleEnabled:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimPolicySet
objectClass:pcimPolicyRule
objectClass:pfimruleinstance
pcimRuleName:Ru le2FTPServer

dn:pcimconditionname=ca,pcimrulename=Rule2FTPServer,ou=GroupOfRulesFTPServer,o=XNC,dc=com
pcimConditionGroupNumber:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimconditionassociation
pcimConditionName:ca
pcimConditionNegated:FALSE
pcimConditionDN:pcimreusablecontainername=CCrule2ftpserver,ou=RepositorioCC,o=XNC,dc=com

dn:pcimactionname=aa,pcimrulename=Rule2FTPServer,ou=GroupOfRulesFTPServer,o=XNC,dc=com
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimactionassociation
pcimActionName:aa
pcimActionDN:pcimreusablecontainername=ACrule_pass,ou=RepositorioCC,o=XNC,dc=com
pcimActionOrder:1

dn:pcimrulename=Rule3FTPServer,ou=GroupOfRulesFTPServer,o=XNC,dc=com
pcimDecisionStrategy:1
pcimRoles:ftpserver
pcimConditionListType:1
pcimExecutionStrategy:2
pcimPolicySetName:RulesFTPServer
pcimRuleEnabled:1
objectClass:top

objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimPolicySet
objectClass:pcimPolicyRule
objectClass:pfimruleinstance
pcimRuleName:Rule3FTPServer

dn:pcimconditionname=ca,pcimrulename=Rule3FTPServer,ou=GroupOfRulesFTPServer,o=XNC,dc=com
pcimConditionGroupNumber:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimconditionassociation
pcimConditionName:ca
pcimConditionNegated:FALSE
pcimConditionDN:pcimreusablecontainername=CCrule3ftpserver,ou=RepositorioCC,o=XNC,dc=com

dn:pcimactionname=aa,pcimrulename=Rule3FTPServer,ou=GroupOfRulesFTPServer,o=XNC,dc=com
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimactionassociation
pcimActionName:aa
pcimActionDN:pcimreusablecontainername=ACrule_pass,ou=RepositorioCC,o=XNC,dc=com
pcimActionOrder:1

dn:pcimrulename=Rule4FTPServer,ou=GroupOfRulesFTPServer,o=XNC,dc=com
pcimDecisionStrategy:1
pcimRoles:ftpserver
pcimConditionListType:1
pcimExecutionStrategy:2
pcimPolicySetName:RulesFTPServer
pcimRuleEnabled:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimPolicySet
objectClass:pcimPolicyRule
objectClass:pfimruleinstance
pcimRuleName:Rule4FTPServer

dn:pcimconditionname=ca,pcimrulename=Rule4FTPServer,ou=GroupOfRulesFTPServer,o=XNC,dc=com
pcimConditionGroupNumber:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimconditionassociation
pcimConditionName:ca
pcimConditionNegated:FALSE
pcimConditionDN:pcimreusablecontainername=CCrule4ftpserver,ou=RepositorioCC,o=XNC,dc=com

dn:pcimactionname=aa,pcimrulename=Rule4FTPServer,ou=GroupOfRulesFTPServer,o=XNC,dc=com
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimactionassociation
pcimActionName:aa
pcimActionDN:pcimreusablecontainername=ACrule_pass,ou=RepositorioCC,o=XNC,dc=com
pcimActionOrder:1

dn:pcimrulename=Rule5FTPServer,ou=GroupOfRulesFTPServer,o=XNC,dc=com
pcimDecisionStrategy:1
pcimRoles:ftpserver
pcimConditionListType:1
pcimExecutionStrategy:2
pcimPolicySetName:RulesFTPServer
pcimRuleEnabled:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimPolicySet
objectClass:pcimPolicyRule
objectClass:pfimruleinstance
pcimRuleName:Rule5FTPServer

dn:pcimconditionname=ca,pcimrulename=Rule5FTPServer,ou=GroupOfRulesFTPServer,o=XNC,dc=com
pcimConditionGroupNumber:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimconditionassociation
pcimConditionName:ca
pcimConditionNegated:FALSE
pcimConditionDN:pcimreusablecontainername=CCruledefault,ou=RepositorioCC,o=XNC,dc=com

dn:pcimactionname=aa,pcimrulename=Rule5FTPServer,ou=GroupOfRulesFTPServer,o=XNC,dc=com
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimactionassociation
pcimActionName:aa
pcimActionDN:pcimreusablecontainername=ACrule_drop,ou=RepositorioCC,o=XNC,dc=com
pcimActionOrder:1

dn:pcimrulename=Rule1SMTPServer,ou=GroupOfRulesSMTPServer,o=XNC,dc=com
pcimDecisionStrategy:1
pcimRoles:smtpserver
pcimConditionListType:1
pcimExecutionStrategy:2
pcimPolicySetName:RulesSMTPServer
pcimRuleEnabled:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimPolicySet
objectClass:pcimPolicyRule
objectClass:pfimruleinstance
pcimRuleName:Rule1SMTPServer

dn:pcimconditionname=ca,pcimrulename=Rule1SMTPServer,ou=GroupOfRulesSMTPServer,o=XNC,dc=com
pcimConditionGroupNumber:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimconditionassociation
pcimConditionName:ca
pcimConditionNegated:FALSE
pcimConditionDN:pcimreusablecontainername=CCrule1smtpserver,ou=RepositorioCC,o=XNC,dc=com

dn:pcimactionname=aa,pcimrulename=Rule1SMTPServer,ou=GroupOfRulesSMTPServer,o=XNC,dc=com
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimactionassociation
pcimActionName:aa
pcimActionDN:pcimreusablecontainername=ACrule_pass,ou=RepositorioCC,o=XNC,dc=com
pcimActionOrder:1

dn:pcimrulename=Rule2SMTPServer,ou=GroupOfRulesSMTPServer,o=XNC,dc=com
pcimDecisionStrategy:1
pcimRoles:smtpserver
pcimConditionListType:1
pcimExecutionStrategy:2
pcimPolicySetName:RulesSMTPServer
pcimRuleEnabled:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimPolicySet
objectClass:pcimPolicyRule
objectClass:pfimruleinstance
pcimRuleName:Rule2SMTPServer

dn:pcimconditionname=ca,pcimrulename=Rule2SMTPServer,ou=GroupOfRulesSMTPServer,o=XNC,dc=com
pcimConditionGroupNumber:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimconditionassociation
pcimConditionName:ca
pcimConditionNegated:FALSE
pcimConditionDN:pcimreusablecontainername=CCrule2smtpserver,ou=RepositorioCC,o=XNC,dc=com

dn:pcimactionname=aa,pcimrulename=Rule2SMTPServer,ou=GroupOfRulesSMTPServer,o=XNC,dc=com
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimactionassociation
pcimActionName:aa
pcimActionDN:pcimreusablecontainername=ACrule_pass,ou=RepositorioCC,o=XNC,dc=com
pcimActionOrder:1

dn:pcimrulename=Rule3SMTPServer,ou=GroupOfRulesSMTPServer,o=XNC,dc=com
pcimDecisionStrategy:1
pcimRoles:smtpserver
pcimConditionListType:1
pcimExecutionStrategy:2
pcimPolicySetName:RulesSMTPServer
pcimRuleEnabled:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimPolicySet
objectClass:pcimPolicyRule
objectClass:pfimruleinstance
pcimRuleName:Rule3SMTPServer

dn:pcimconditionname=ca,pcimrulename=Rule3SMTPServer,ou=GroupOfRulesSMTPServer,o=XNC,dc=com
pcimConditionGroupNumber:1
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimconditionassociation
pcimConditionName:ca
pcimConditionNegated:FALSE
pcimConditionDN:pcimreusablecontainername=CCruledefault,ou=RepositorioCC,o=XNC,dc=com

dn:pcimactionname=aa,pcimrulename=Rule3SMTPServer,ou=GroupOfRulesSMTPServer,o=XNC,dc=com
objectClass:top
objectClass:dlm1ManagedElement
objectClass:pcimPolicy
objectClass:pcimactionassociation
pcimActionName:aa
pcimActionDN:pcimreusablecontainername=ACrule_drop,ou=RepositorioCC,o=XNC,dc=com
pcimActionOrder:1

dn:pfimdevicename=WAN,pfimdevicename=Router,ou=Network,o=XNC,dc=com
dlmDeviceID:200.246.5.254
pfimDeviceName:WAN
objectClass:top
objectClass:dlm1ManagedElement
objectClass:dlm1ManagedSystemElement
objectClass:dlm1LogicalElement
objectClass:dlm1LogicalDevice
objectClass:pfimdevice