Modelagem de Sistemas de Segurança em Ambientes de Redes de Larga Escala

João Porto de Albuquerque[1,2]*, **Heiko Krumm**[2], **Paulo Lício de Geus**[1]

[1]Instituto de Computação – Universidade Estadual de Campinas
13083-970 Campinas/SP Brazil
{jporto, paulo}@ic.unicamp.br

[2]FB Informatik – University of Dortmund
44221 Dortmund Germany
{Joao.Porto, Heiko.Krumm}@udo.edu

Abstract. *As the use of computers and data communication technologies spreads, network security systems are becoming increasingly complex, due to the incorporation of a variety of mechanisms necessary to fulfil the protection requirements of the upcoming scenarios. The integrated design and management of different security technologies and mechanisms are thus of great interest. Especially in large-scale environments, the employment of security services and the design of their configurations shall be supported by a structured technique which separates the consideration of the system as a whole from the detailed design of the subsystems. To accomplish this goal, this paper presents a scalable approach for the modelling of large security systems, relying on the concepts of policy-based management and model-based management.*

Resumo. *À medida que o uso de computadores e de tecnologias de comunicação de dados se amplia, os sistemas de segurança de redes tornam-se crescentemente complexos, devido à incorporação dos variados tipos de mecanismos necessários para satisfazer os requisitos de proteção dos novos cenários. Nesse contexto, o projeto e o gerenciamento integrados de diferentes tecnologias e mecanismos de segurança adquirem grande relevância. Especialmente em ambientes de larga escala, é desejável que o emprego de serviços de segurança e o projeto de suas configurações sejam apoiados por uma técnica estruturada que separe a consideração do sistema como um todo do desenho detalhado de seus subsistemas. Visando atingir esse objetivo, o presente trabalho apresenta uma abordagem escalável para a modelagem de sistemas de segurança de redes, fundado nos conceitos de gerenciamento baseado em políticas e gerenciamento baseado em modelos.*

1. Introdução

A ampla utilização de computadores e tecnologias de comunicação de dados, conectados a uma crescente Internet, requer a adoção de medidas de proteção para controlar o risco de ataques através da rede. Com tal intuito são empregados *sistemas de segurança de redes*, os quais podem ser vistos como um conjunto de *serviços de segurança*, visando prover a operação e comunicação seguras da rede, nos moldes ditados pela política de segurança da organização. Cada um desses serviços de segurança é implementado por um ou mais *mecanismos de segurança*.

*Financiado pelo Serviço Alemão de Intercâmbio Acadêmico (DAAD).

Em consonância com as necessidades de proteção dos ambientes atuais, as tecnologias dos sistemas de segurança de redes têm-se tornado significativamente mais complexas. Aos tradicionais *firewalls* combinados à fortificação das configurações de sistemas [Garfinkel and Spafford, 1996, Cheswick et al., 2003, Zwicky et al., 2000] incorporam-se, então, mecanismos como Redes Privadas Virtuais (VPNs), associações criptográficas fim-a-fim (utilizando, por exemplo, IPSec), uso de pontos de aplicação distribuídos para uma política global (*firewalls distribuídos*) [Bellovin, 1999], serviços de autenticação (como Kerberos) e autorização, como também diversos sistemas para monitoramento, registro de informações críticas (*logging*), auditoria e detecção a intrusões (*intrusion detection systems*).

À medida que esses serviços e mecanismos de segurança são gradualmente empregados — resultando, então, em cenários ofuscantemente emaranhados —, a importância e o custo do gerenciamento da segurança crescem exponencialmente. Inicialmente, as tarefas desse gerenciamento abrangem a instalação e configuração dos serviços de segurança, às quais se adicionam, durante a operação, o monitoramento, a auditoria, a adaptação e a reconfiguração dos mesmos. Dessa forma, abstração adequada, integração e ferramentas de suporte se tornam fatores chave para facilitar as tarefas de gerenciamento.

As abordagens de hierarquias de política (*policy hierarchies*) [Moffett and Sloman, 1993, Wies, 1995] e gerenciamento baseado em políticas (*policy-based management*) [Sloman, 1994] podem ser utilizadas com proveito nesse contexto, visto que almejam automatizar tarefas de gerenciamento em ambientes complexos. Essas abordagens funcionam em conjunto da seguinte maneira: *hierarquias de políticas* podem ser construídas tomando-se um conjunto inicial de políticas de alto nível e refinando-o, em seguida, ao longo de níveis intermediários, até alcançar políticas de baixo nível de abstração, que sejam mecanicamente executáveis. Posteriormente, o *gerenciamento baseado em políticas* utilizará essas políticas de baixo nível, valendo-se de agentes distribuídos que se intercomunicarão, interpretarão e executarão aquelas políticas que são destinadas especificamente ao seu respectivo papel de gerenciamento (*management role*) [Lupu and Sloman, 1996]. Alternativamente, para mecanismos de segurança não capazes de interpretar políticas de gerenciamento (*policy-unaware*), estas deverão ser convertidas em arquivos de configuração com sintaxe própria a cada mecanismo, para que possam ser efetivamente implementadas.

A abordagem de *gerenciamento baseado em modelos* (*model-based management*) [Lück et al., 1999, Lück et al., 2001, Lück et al., 2002], por sua vez, oferece suporte à construção de hierarquias de políticas por meio de um projeto gráfico interativo. Ela adota conceitos de ferramentas de projeto orientado a objetos e emprega um modelo do sistema a ser gerenciado que é verticalmente estruturado em camadas. Nesse modelo, os objetos e associações pertencentes a uma camada representam o sistema em um certo nível de abstração.

Um problema comum a essas abordagens ocorre no tratamento de sistemas de grande dimensão, pois o modelo tende a perder muito de sua compreensibilidade, tornando-se obscuro devido a seu elevado número de componentes (como experienciado por [Geist, 2003]). A maneira clássica de lidar com problemas desse tipo consiste no uso do princípio de *divisão e conquista*, segundo o qual a segmentação de um sistema em módulos menores permite-nos tratar detalhadamente cada um destes em separado, e considerar o sistema como um todo mediante uma visão mais abstrata da interação entre suas partes.

No presente trabalho tratou-se de aplicar esse princípio, de forma a obter uma abordagem para modelagem de sistemas de segurança de redes baseada na segmentação

de um sistema em subsistemas abstratos (*Abstract Subsystems*, abreviado como AS). O diagrama de subsistemas abstratos (*Diagram of Abstract Subsystems*, DAS) consiste, portanto, em uma representação da estrutura geral de um sistema, cujos detalhes são ocultos e tratados separadamente na especificação interna de cada subsistema. Essa abstração viabiliza, dessa forma, a decomposição dos processos de projeto e análise do sistema, contribuindo para a melhoria da compreensibilidade e da escalabilidade do modelo.

O DAS baseia-se numa visão de sistema orientada a políticas, oferecendo, ainda, uma camada de ligação entre uma visão de sistema orientada a serviços e a representação de seus reais mecanismos de rede. Além disso, essa técnica de modelagem é assistida por uma ferramenta de software, a qual proporciona um editor gráfico para o desenho de modelos e funções adicionais para a verificação de restrições de integridade e para a condução do processo de refinamento de políticas através das diferentes camadas hierárquicas do sistema.

Dado que nosso trabalho fundamenta-se na abordagem de gerenciamento baseado em modelos, apresentamos uma introdução a esta na seção seguinte. Subseqüentemente, o conceito de subsistema abstrato é exposto na Seção 3, servindo de base para as discussões posteriores acerca do diagrama de subsistemas abstratos (Seção 4), da modelagem sistemática de ASs (Seção 5) e do refinamento automático do modelo na Seção 6. Por fim, analisamos trabalhos correlatos na Seção 7 e tecemos conclusões para este trabalho na Seção 8.

2. Gerenciamento Baseado em Modelos

O conceito de gerenciamento baseado em modelos foi proposto inicialmente em [Lück et al., 1999] e aplicado posteriormente à configuração de vários tipos de mecanismos de segurança, como, por exemplo, filtros de pacotes [Lück et al., 2001] e VPNs [Lück et al., 2002]. Essa abordagem tenciona apoiar o gerenciamento baseado em políticas mediante um modelo orientado a objetos do sistema a ser gerenciado. Esse modelo viabiliza a execução de um refinamento das políticas de segurança, por meio do qual parâmetros de configuração para mecanismos de segurança podem ser automaticamente derivados.

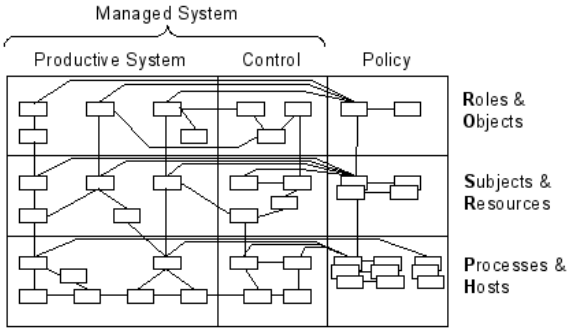

Figura 1: Visão Geral do Modelo

A estrutura do modelo é apresentada na Figura 1 (reproduzida de [Geist, 2003]), a qual possui três níveis de abstração: Papéis & Objetos (RO: *Roles & Objects*), Sujeitos & Recursos (SR: *Subjects & Resources*) e Processos e Máquinas (PH: *Processes & Hosts*). Cada nível é um refinamento do nível superior imediato, no sentido de uma hierarquia de política [Wies, 1995, Moffett and Sloman, 1993]. O nível mais alto representa a visão de negócios da rede, enquanto o nível mais baixo, relaciona-se à visão técnica. A linha vertical, por sua vez, marca a diferença entre o modelo propriamente dito (com elementos de produção e controle), do lado esquerdo, e as políticas que regulam o sistema, do

lado direito. Essas políticas de segurança são representadas por objetos que expressam permissões e requisitos; cada um destes se refere a componentes do modelo localizados em seu mesmo nível de abstração.

O nível mais alto (RO) do modelo erige-se sobre conceitos provenientes do Controle de Acesso Baseado em Papéis (*Role Based Access Control*) [Sandhu et al., 1996]. As principais classes desse nível são:

Role: Papéis assumidos pelas pessoas dentro do ambiente modelado;
Object: Objetos do ambiente modelado que devem ser submetidos a controle de acesso;
AccessMode: Modos de acesso aos objetos;
AccessPermission: Autorizações para que o indivíduo atuando em um papel (*Role*) acesse um objeto (*Object*) particular, da maneira definida pelo modo de acesso (*AccessMode*).

O segundo nível de abstração (SR na Figura 1) consiste num conjunto mais complexo de classes. Os objetos dessas classes representam: (a) pessoas que trabalham no ambiente modelado (classe *User*); (b) sujeitos (ou sessões) que agem a comando dos usuários, executando suas requisições (*SubjectType*); (c) serviços na rede que são utilizados para acessar recursos (*Service*); (d) dependências entre serviços (*ServiceDependency*); (e) recursos da rede (*Resources*) que são utilizados por serviços. A classe *ServicePermission* define políticas de autorização nesse nível, representando a permissão para que um sujeito, agindo no interesse de um usuário, acesse um recurso através do uso de um serviço.

O nível SR oferece, portanto, uma transição da visão voltada para o negócio, representada no nível RO, para uma perspectiva mais técnica, baseada em serviços. Para tanto, utiliza-se de uma abordagem de gerenciamento orientado a serviços (*service-oriented management* [McBride, 1998]), de forma a obter uma visão ainda relativamente abstrata do sistema gerenciado — o qual é definido sob o ponto de vista dos serviços que deverá prover.

O terceiro nível de abstração (PH), por sua vez, é responsável por modelar os mecanismos que serão utilizados para implementar os serviços do nível SR. Ele conterá, portanto, um número ainda maior de classes que seu predecessor, as quais representarão, por exemplo, as máquinas da rede (classe *Host*), com suas respectivas interfaces de rede (classes *Interface*), e os processos (classe *Process*), que realizam ações comunicativas através de *sockets* (classe *Sockets*). Permissões de protocolo (*ProtocolPermissions*) são utilizadas para representar políticas de autorização desse nível, exprimindo permissões para o tráfego de pacotes entre processos. Os autores definem, ainda, diversas outras classes, as quais não são aqui mencionadas por razões de brevidade.

Para apoiar a modelagem e a configuração interativa dos mecanismos, uma ferramenta de *software* é oferecida — cuja interface se apresenta na Figura 2. Essa ferramenta proporciona ao projetista um editor gráfico para a confecção dos diagramas, com funções adicionais para a verificação de restrições de integridade no modelo.

Nota-se, a partir da discussão prévia, que o nível PH — o qual deve representar o sistema inteiro, com seus processos, máquinas, interfaces de rede, entre outros — torna-se rapidamente complexo, ao passo que o tamanho do sistema modelado cresce. Esse comportamento encontra-se também ilustrado na Figura 2, a qual apresenta o modelo de um cenário muito simples, com apenas um objeto *AccessPermission* em seu nível mais alto de abstração que expressa a seguinte política: os empregados da empresa têm autorização para acessar o servidor de páginas *web* corporativo. Não obstante, apesar da simplicidade desse nível RO, o modelo desdobra-se em um número considerável de objetos em seu nível mais baixo (fundo da fig. 2), a fim de representar mecanismos como mascaramento de endereços IP (*IP-masquerading*), *firewalls* e distribuidores de carga (*load balancers*).

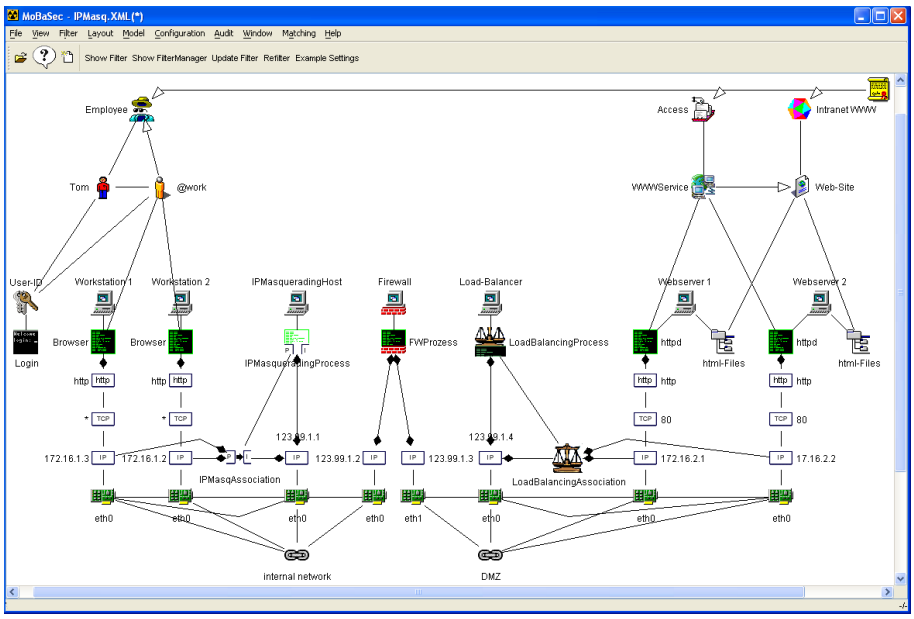

Figura 2: Ferramenta de Suporte

Devido à simpleza do exemplo, o modelo resultante é ainda razoável; contudo, infere-se daí que modelos de sistemas reais, de grande dimensão, tendem a tornar-se bastante confusos. Com o intuito de solucionar esse problema introduzimos na próxima seção o conceito de subsistema abstrato.

3. Conceito de Subsistema Abstrato

Um subsistema abstrato (*Abstract Subsystem*, abreviado como AS) consiste em uma visão abstrata de um segmento do sistema, isto é, em uma representação simplificada de um certo grupo de componentes do sistema. Dessa forma, um AS omite muitos detalhes que lhe são internos, exibindo, ao invés disso, apenas os aspectos relevantes para uma visão global da estrutura do sistema. Tais aspectos são escolhidos com base em uma visão do sistema que se pauta pelas políticas a ele aplicadas (*policy-oriented*) — ilustrada na Figura 3.

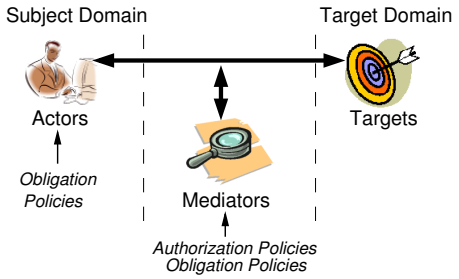

Figura 3: Componentes de Subsistemas Abstratos

Nesse esquema, podem-se distinguir três tipos de elementos: *Actors* (atores), *Mediators* (mediadores) e *Targets* (alvos). O primeiro tipo (*Actors*) consiste em grupos de indivíduos que possuem um comportamento *ativo* no sistema, isto é, eles iniciam comunicações e executam operações obrigatórias de acordo com as políticas de obrigação do sistema (*obligation policies*, ver [Sloman and Lupu, 2002]).

O segundo tipo de elementos abrange *Mediators*, os quais intermediam comunicações, recebendo requisições, inspecionando o tráfego, filtrando e/ou transformando o

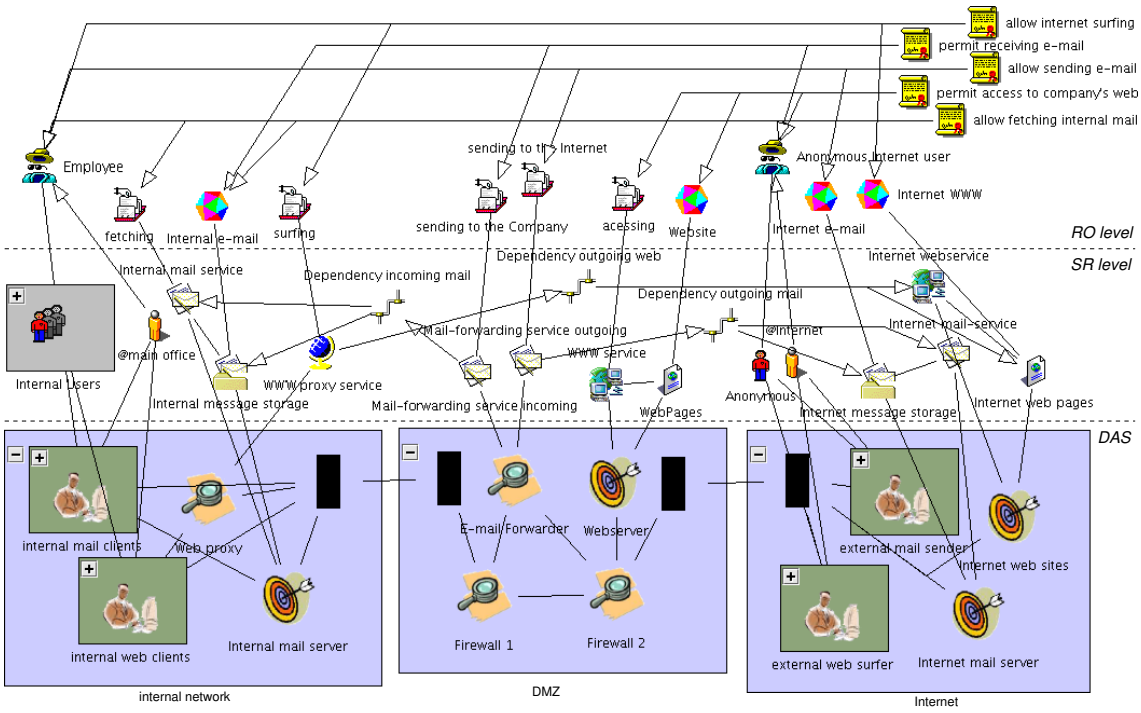

Figura 4: Modelo em Três Níveis de Abstração

fluxo de dados de acordo com políticas de autorização (*autorisation policies*). Eles também podem realizar operações obrigatórias previstas por políticas, tais como o registro de informações acerca de fluxos de dados. Os *Targets*, por sua vez, são elementos passivos; eles contêm informações relevantes, as quais são acessadas por *Actors*.

Tomando-se esse esquema como fundação, podemos agora redefinir um AS como um grafo não-orientado cujas arestas representam comunicação potencial (bidirecional) entre seus nós. Cada um destes pode pertencer ou a um dos tipos acima mencionados (*Actors*, *Mediators* e *Targets*) ou, ainda, à categoria *Connectors* (conectores). Esse último tipo de componentes é adicionado para representar os pontos de interface entre dois ASs, ou seja, para possibilitar o fluxo de informações de um AS para outro.

4. Diagrama de Subsistemas Abstratos

Com base nos conceitos apresentados na seção anterior, introduzimos aqui um novo nível de abstração na modelagem de sistemas de segurança: o diagrama de subsistemas abstratos (*Diagram of Abstract Subsystems* – DAS). Essa camada localiza-se imediatamente abaixo da visão do sistema orientada a serviços (nível SR na fig. 1) e acima da camada PH, a qual representa os mecanismos reais do sistema.

O objetivo principal de um DAS é descrever de forma modular a estrutura geral do sistema a ser gerenciado, isto é, através da definição de seus blocos constituintes (ASs) e da indicação de conexões entre eles. Tendo em vista que esses blocos consistem em visões abstratas e baseadas em políticas dos componentes reais de cada subsistema (ver sec. 3), infere-se que um DAS provê uma visão concisa e inteligível da arquitetura do sistema — em um sentido similar ao proposto por [Porto de Albuquerque and de Geus, 2003]. Além disso, esse diagrama viabiliza a consideração da estrutura do sistema em conjunto com as políticas de segurança executáveis que a ele se aplicam, explicitando, assim, a distribuição dos diferentes participantes dessas políticas sobre o sistema.

Em uma definição formal, o DAS é um grafo composto de ASs como nós e de arestas que representam a possibilidade de comunicação bidirecional entre dois ASs,

como apresentado na parte inferior da Figura 4. De uma perspectiva descendente em níveis de abstração (ou seja, do nível SR para baixo), esse diagrama acrescenta quatro tipos de informação a seu predecessor: (a) a segmentação de usuários, serviços e recursos em subsistemas; (b) as conexões estruturais entre elementos e subsistemas; e, por conseguinte, (c) as conexões estruturais entre os diferentes participantes de uma política (*Actors*, *Mediators* e *Targets*); e, por fim, (d) elementos mediadores que não estão diretamente relacionados com os serviços do nível SR, mas que tomam parte na comunicação, filtrando e transformando os dados (e.g. *firewalls*).

Para esclarecer a aplicação prática do DAS, expomos na seção seguinte uma sistemática para o mapeamento de uma visão de sistema orientada a serviços para uma representação por meio de subsistemas abstratos, e, em seguida, desta para a modelagem dos reais mecanismos do sistema. Cada etapa desse mapeamento será exemplificada através de um cenário-teste.

5. Modelagem Sistemática de Subsistemas Abstratos

A Figura 5 ilustra o cenário-teste a ser considerado. Esse cenário consiste em uma versão simplificada do estudo de caso realizado em [Geist, 2003], o qual se baseia em um típico ambiente empresarial de rede conectado à Internet.

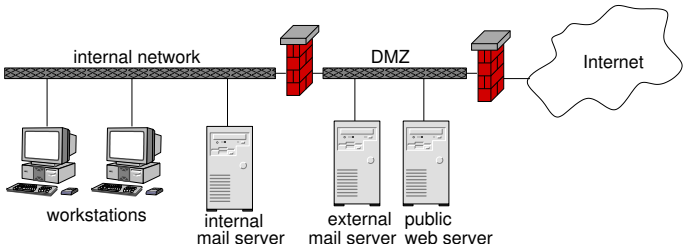

Figura 5: Cenário-teste

Para esse cenário, prescrevem-se as seguintes políticas de segurança: 1) os usuários são autorizados a navegar na Internet a partir das estações de trabalho internas; 2) os usuários podem enviar e-mails para a Internet a partir de suas estações de trabalho; 3) o correio eletrônico oriundo da Internet deve ser autorizado a chegar ao servidor de correio externo, o qual, por sua vez, poderá encaminhá-lo ao servidor de correio interno; 4) usuários externos utilizando a Internet podem acessar o servidor *web* público da companhia; e 5) os usuários são autorizados a, a partir das estações de trabalho internas, ler e-mails localizados no servidor de correio interno.

A modelagem dessas políticas, de acordo com os princípios mencionados na Seção 2, resulta nos dois primeiros níveis da Figura 4 (os detalhes desse procedimento estão fora do escopo do presente trabalho e podem ser encontrados em [Lück et al., 2002, Lück et al., 2001]). O nível mais alto (RO) desse modelo contém um objeto *AccessPermission* para cada uma das políticas enumeradas acima, as quais serão referenciadas doravante como AP1–AP5.

O desenvolvimento do correspondente DAS terá, portanto, o modelo no nível SR como ponto de partida, e será executado em três etapas: (i) a segmentação do sistema em módulos (ASs), de acordo com o respectivo cenário de rede; (ii) o mapeamento dos elementos do nível SR (usuários, serviços, recursos) para os componentes internos a cada AS; e (iii) o estabelecimento de conexões estruturais entre os elementos dentro de cada AS e entre os ASs. Em seguida, cada AS será expandido independentemente, de forma a produzir uma representação detalhada de seus mecanismos no nível PH, viabilizando,

assim, a geração automatizada de seus correspondentes arquivos de configuração. Essas etapas serão correspondentemente descritas nas próximas seções.

5.1. Segmentação em ASs

A divisão do sistema em ASs deverá orientar-se pelos blocos estruturais do ambiente analisado. Dessa forma, os componentes abstratos do DAS serão agregados de acordo com os grupos de mecanismos já existentes no sistema real, tais como departamentos, lugares de trabalho e funções desempenhadas.

Um importante critério a ser considerado nessa segmentação é a unidade semântica de um AS, i.e. o grupo de mecanismos abarcado por um AS deve possuir uma propriedade comum claramente distinguível. Essa propriedade garante, portanto, a *coesão* do AS, fazendo com que este represente um agrupamento lógico identificável no ambiente real (tais como aqueles anteriormente referidos), ao invés de constituir em mero agregado de elementos heterogêneos.

Analisando o cenário ilustrado na Figura 5, a existência de uma subdivisão estrutural em três segmentos é clara, a saber: a rede interna, a zona desmilitarizada (DMZ, no acrônimo em inglês) e a rede externa (a Internet). Logo, o DAS para o nosso cenário-teste compor-se-á de um AS para cada um desses segmentos.

5.2. Mapeamento de Actors

Um objeto *Actor* será criado, basicamente, para cada grupo de máquinas e processos dentro de um certo AS que iniciam comunicações, de modo a atuar em conformidade com uma política. Essas políticas, em nosso modelo, são representadas por objetos *ServicePermission* no nível SR. Portanto, um ator corresponde ao domínio-sujeito (*subject domain*, ver [Lupu and Sloman, 1996]) de uma *ServicePermission*, ou, mais precisamente, à parte desse domínio que está localizada em um certo AS.

Por outro lado, um *Actor* pode ser compartilhado por diferentes *ServicePermissions*, contanto que estas tenham em comum a parte de seu domínio-sujeito representada pelo *Actor*. Esse compartilhamento contribui para uma representação mais compacta, beneficiando, assim, a escalabilidade dos modelos.

No nível SR o domínio-sujeito das *ServicePermissions* é representado por objetos dos tipos *User* e *SubjectType* (ver sec. 2); logo, cada *Actor* estará conectado a um ou mais pares de objetos desses tipos. No contexto do gerenciamento baseado em modelos, entretanto, *ServicePermissions* não são diretamente modeladas pelo projetista do sistema, mas, antes, geradas automaticamente a partir dos objetos *AccessPermission* definidos no nível mais alto do modelo (RO). Sendo assim, a determinação dos *Actors* no sistema dever-se-á iniciar de uma *AccessPermission*, identificando a *Role* a ela associada, e, então, prosseguir com a definição de seus objetos *User* e *SubjectType* correspondentes. Posteriormente, cria-se um objeto *Actor* ligado a estes objetos e, assim, à *ServicePermission*, contemplando, conseqüentemente também a a *AccessPermission* considerada inicialmente.

Voltando-nos, agora, ao nosso cenário-teste, o objeto "internal web clients" dentro do AS que representa a rede privada da companhia ("internal network") é criado para mapear a primeira política — a qual é modelada pela primeira *AccessPermission* na Figura 4, a saber, AP1:"allow Internet surfing" (permitir navegação na Internet). De maneira análoga, para a *AccessPermission* AP3 ("allow sending e-mail" — permitir o envio de e-mails) o *Actor* "internal mail clients" é criado no mesmo AS, enquanto que, no AS "Internet", os atores "external mail sender" e "external web surfer" correspondem, respectivamente, aos objetos AP2:"permit receiving e-mail" (permitir o recebimento de

e-mails) e AP4:“permit access to own web” (permitir o acesso ao servidor web próprio). Por fim, a *AccessPermission* AP5: “allow fetching e-mail” (permitir a recuperação de e-mails) pode-se utilizar do *Actor* previamente criado “internal mail clients”, visto que seu domínio-sujeito é o exatamente o mesmo que o de AP3.

5.3. Mapeamento de Mediators

Em relação aos *Mediators*, dois tipos podem ser distinguidos: *Mediators* do primeiro tipo são refinamentos a partir de serviços que executam as funções de “homem-do-meio” (*middleman functions*) descritas na Seção 3, como por exemplo, *proxy* e encaminhamento de e-mails. Dessa forma, eles serão obtidos através de um mapeamento direto a partir de serviços no nível SR, criando-se, para tanto, um objeto correspondente e posicionando-o no AS apropriado. Um serviço pode, também, ser refinado por meio de mais de um *Mediator*, cada um dos quais residindo em um AS diferente. Por outro lado, um dado *Mediator* pode também mapear mais de um serviço.

Em nosso cenário-teste, o *Mediator* “E-Mail-Forwarder”, localizado no AS “DMZ”, representa os serviços de tratamento tanto e-mails que chegam, como também daqueles que são através dele enviados. No AS “internal network”, o *Mediator* “Web Proxy” mapeia o serviço “WWWProxyService”.

O segundo tipo de *Mediators*, por sua vez, representa mecanismos que não são modelados no nível SR, mas que são necessários para o controle da comunicação, isto é, eles transformam e/ou filtram-na de acordo com políticas de autorização (realizando, por exemplo, filtragem de pacotes e mascaramento de endereços IP), ou inspecionam os dados de acordo com políticas de obrigação (e.g. IDS, monitores de eventos). Assim sendo, deve o projetista do sistema criar objetos desse tipo sempre que alguma dessas funcionalidades for requerida, ou seja, um *Mediator* estará presente sempre que se encontrar um mecanismo de segurança como os supra-citados no respectivo ambiente de rede real.

Exemplos para esse segundo tipo de *Mediators* são apresentados na Figura 4 através dos objetos “Firewall 1” e “Firewall 2”. Estes foram introduzidos no AS “DMZ”, exatamente nos lugares em que *firewalls* são encontrados no cenário da Figura 5.

5.4. Mapeamento de Targets

Targets são obtidos por um mapeamento direto a partir de pares de objetos *Service* e *Resource* (no nível SR) que abarquem o domínio-alvo de uma *ServicePermission*, ou a parte desse domínio que localiza-se em um certo AS. Dessa maneira, cada *Target* deve conectar-se a, pelo menos, um par de objetos *Service* e *Resource* no nível SR. Entretanto, um Target também pode ser compartilhado por diferentes *ServicePermissions* e, neste caso, relações com mais de um par daqueles objetos estarão presentes — esse compartilhamento também contribuirá para a concisão do modelo. Inversamente, cada par *Service–Resource* pode ser mapeado em vários *Targets*, cada um dos quais localizado em diferentes ASs, de maneira similar ao que ocorre com *Mediators*.

Como no caso de *Actors*, a identificação de *Targets* também deve ter seu ponto de partida nas *AccessPermissions* do nível RO. Aqui, contudo, o que se deve considerar inicialmente é o *Object* relacionado a cada *AccessPermission*, a fim de obter, então, os correspondentes *Resource* e *Service* no nível SR. Por fim, cria-se um *Target* no DAS para mapear esse par de objetos.

Aplicando esse método ao nosso cenário, criamos, para a política AP1, o *Target* “Internet web sites” que refina o par *Service–Resource* composto dos objetos “Internet webservice” e “Internet web pages”, visto que este se relaciona com o *Object* “Internet

WWW" de AP1. Vale notar que, neste caso, o objeto *Service* refinado a partir do *AccessMode* de AP1 (a saber: "WWWProxyService") não é o mesmo do que aquele que relacionamos anteriormente ao *Target*; tal fato pode ocorrer apenas quando existe uma dependência de serviços entre os dois *Services* (representada pelo objeto *ServiceDependency*, ver sec. 2). De fato, o serviço "WWWProxyService" não pode prover, sozinho, acesso ao recurso "Internet WWW", pois precisa valer-se do serviço "Internet webservice" para efetuá-lo.

Procedendo da mesma maneira, criamos os *Targets* "internal mail server" (no AS "internal network"), "Webserver" (em "DMZ") e "Internet mail server" (em "Internet") para mapear, respectivamente, as políticas modeladas por AP2, AP3 e AP4. Quanto à AP5, ela pode compartilhar o *Target* "internal mail server" com AP2 de forma que não há necessidade da criação de um outro objeto.

5.5. Estabelecimento de Conexões Estruturais

Para completar o modelo formado pela aplicação dos procedimentos das seções anteriores, necessitamos introduzir as associações entre os objetos do DAS, ou, dito de maneira formal, acrescentar as arestas do grafo. Essas associações tem um significado diferente daquele possuído pelos mapeamentos entre um elemento no nível SR e um objeto no DAS: enquanto que estes representam refinamentos de abstração — no sentido de relacionar diferentes níveis de uma hierarquia de política —, cada uma daquelas associações representa uma conexão estrutural, i.e. uma possibilidade de comunicação no sistema real. Todavia, apenas as conexões relevantes à visão abstrata do sistema devem ser representadas no DAS; essas conexões são, portanto, aquelas que interconectam os diferentes participantes de políticas executáveis: *Actors*, *Mediators* e *Targets*.

O estabelecimento dessas conexões inicia-se, então, novamente, a partir de uma *AccessPermission*. Cada objeto no DAS que corresponde a essa *AccessPermission* é identificado e associações são estabelecidas, a fim de construir os caminhos entre os relevantes *Actors* e *Targets*, atravessando, também, os *Mediators* necessários. Em cada ponto em que um desses caminhos entrar ou sair de um AS será inserido um objeto *Connector*, o qual representará uma interface de comunicação do AS. O número de *Connectors* em um AS corresponde, portanto, ao número de interfaces físicas disponíveis no sistema real.

Seguindo esse procedimento, obtemos, para nosso cenário-teste, os *Connectors* (retângulos) e arestas de conexão (linhas) exibidos na Figura 4.

5.6. Expansão de ASs

Partindo do modelo criado até esse ponto (fig. 4), o próximo passo do desenvolvimento será a expansão de cada AS em separado. Isso significa que, para cada AS, os mecanismos internos a ele devem ser modelados de acordo com o procedimento usual (descrito em [Geist, 2003]), resultando em uma representação detalhada dos mesmos, ou seja, o nível PH. Posteriormente, as associações entre os componentes do nível PH e os objetos do respectivo AS devem ser estabelecidas, definindo, assim, uma relação de refinamento de abstração.

Cada objeto *Actor* de um AS será, então, relacionado com seus correspondentes componentes de tipo *Process* e *UserID* no nível PH, de tal forma que o *Actor* efetue a associação destes componentes com os respectivos objetos *SubjectType* e *User* do nível SR. Como mencionado na Seção 5.2, um Actor pode ser compartilhado por mais de um par *SubjectType–User*, correspondendo, dessa forma, a várias *ServicePermissions*. Portanto, o objeto *Actor* deve manter um tabela de 4-tuplas (contendo *User*, *SubjectType*, *UserID* e *Process*) a fim de armazenar cada associação em particular.

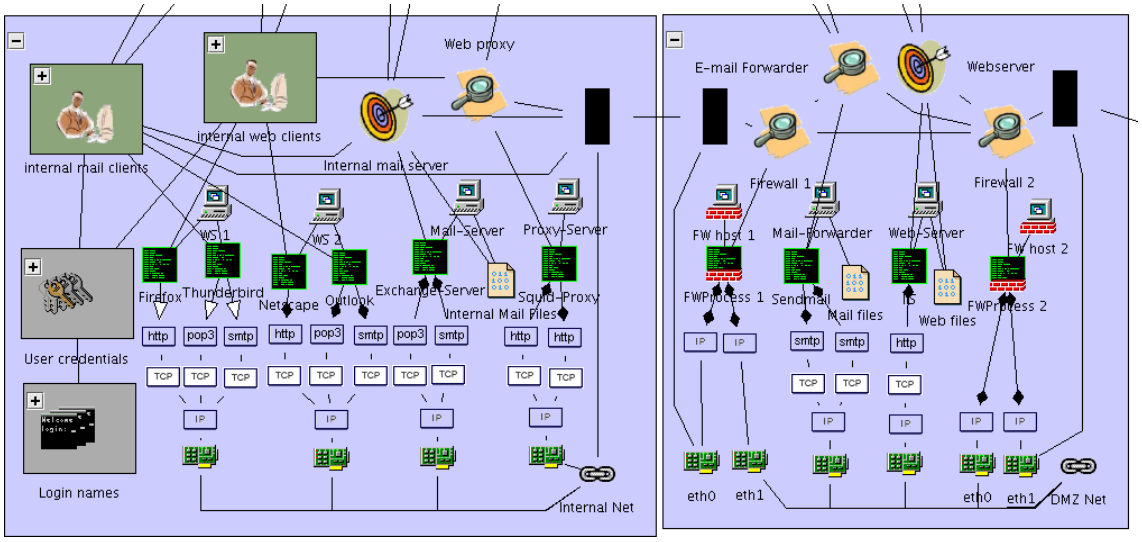

Figura 6: ASs Expandidos

Os *Mediators* de um AS, por sua vez, serão relacionados simplesmente a um ou mais componentes do tipo *Process*, os quais implementam suas correspondentes funcionalidades. Quanto aos *Targets*, a cada um deles associar-se-á um ou mais pares de objetos dos tipos *Process* e *Object* que implementam os respectivos serviços. Neste caso, um objeto *Process*, no nível PH, relacionar-se-á, através de um *Target* no DAS, a um *Service* no nível SR, enquanto um *Object*, da mesma maneira, a um *Resource*.

A Figura 6 apresenta o modelo PH correspondente aos ASs "internal network" (à esquerda) e "DMZ" (à direita) em nosso exemplo (fig. 4). Nessa figura, a relação entre *Actors*, *Mediators* e *Targets* dos ASs e os objetos do nível PH — obtida meramente seguindo-se os princípios acima expostos — é graficamente indicada. Dessa forma, a correspondência entre a visão abstrata de cada subsistema (AS) e seus reais mecanismos (PH) é estabelecida. Comparando-se a visão simplificada desses subsistemas (no DAS da fig. 4) e a detalhada (fig. 6), nota-se claramente que a modelagem por meio de subsistemas abstratos oferece vantagens concretas no que tange à concisão e à compreensibilidade do modelo.

6. Ferramenta de Suporte e Refinamento Automatizado

A técnica de modelagem apresentada no presente trabalho é assistida por uma ferramenta de *software*, a qual inclui um editor gráfico de diagramas (por meio do qual as figs. 4 e 6 foram produzidas) por meio do qual se pode definir os objetos do modelo e seus relacionamentos, assim como atribuir-lhes propriedades. Dessa forma, cada passo da modelagem explicada nas seções anteriores é apoiado pela ferramenta, a qual verifica a concordância do modelo com restrições de integridade estruturais na medida em que objetos em particular são definidos. Uma vez que o modelo está completo, executam-se uma série de testes a fim de garantir a consistência do modelo como um todo.

Embora uma derivação totalmente automática de políticas executáveis (de baixo nível de abstração) a partir de um conjunto de especificações abstratas não seja praticável no caso geral (cf. [Sloman and Lupu, 2002]), a técnica de modelagem aqui apresentada torna possível uma automatização da construção de hierarquias de política, tomando-se por base um modelo do sistema que é estruturado em diferentes níveis de abstração. Assim, a análise dos objetos do sistema em conjunto com seus relacionamentos recíprocos e com as políticas que lhes são prescritas, todos localizados em um determinado nível

de abstração, viabiliza a geração de políticas de nível mais baixo, considerando ainda o modelo do sistema neste nível e os relacionamentos entre as entidades dos dois níveis[1].

A ferramenta de suporte aplica o referido procedimento, primeiramente, de forma a refinar cada um dos objetos do tipo *AccessPermission* (do nível RO) definidos pelo projetista, obtendo uma ou mais correspondentes *ServicePermissions* no nível SR (ver sec. 2). Em seqüência, um conjunto de objetos do tipo *ATPathPermission* são derivados a partir das *ServicePermissions*. Cada *ATPathPermission* consiste em um caminho no grafo DAS que representa a autorização para que um *Actor* alcance um certo *Target* passando através dos objetos *Mediator* e *Connector* que se encontram ao longo desse caminho.

O refinamento prossegue com a geração automatizada de *ProtocolPermissions* a partir das *ATPathPermissions*, valendo-se da informação detalhada contida no nível PH, de modo a obter políticas de segurança para este nível. Cada *ProtocolPermission* se relaciona com um conjunto de objetos no nível PH para representar a permissão para que um processo iniciante — ao qual uma credencial de usuário pode ser atribuída — comunique-se, através de sua entidade de protocolo relacionada e um protocolo remoto, com um processo servidor, a fim de acessar um certo recurso físico.

Finalmente, como última etapa da configuração de serviços de segurança baseada em modelos, uma série de módulos *backend* específicos são executados, cada um dos quais correspondente a um produto em particular que implementa um serviço de segurança (e.g. Kerberos, FreeS/WAN, Linux IP tables etc.). Essas funções *backend* avaliam, então, as *ProtocolPermissions* e os modelos PH, produzindo os arquivos de configuração adequados para cada um dos produtos de segurança (mais detalhes acerca desse procedimento podem ser encontrados em [Lück et al., 2001, Lück et al., 2002, Geist, 2003]).

7. Trabalhos correlatos

Há um número razoável de abordagens para especificação de políticas de segurança e de gerenciamento (ver [Sloman and Lupu, 2002] para uma recente resenha crítica)[2]. No entanto, a construção de hierarquias de política assistida por ferramentas e a automação do refinamento de políticas (culminando na geração automática de parâmetros de configuração para mecanismos) merecem, ainda, considerável esforço de pesquisa.

Uma abordagem nessa direção é a ferramenta *Firmato* [Bartal et al., 2004], a qual oferece suporte interativo ao projeto de políticas por meio de diagramas e derivação automática de parâmetros de configuração de dispositivos (como roteadores, *switches* e *softwares* para *firewalls*). O nível de abstração no qual as políticas são representadas nessa ferramenta é, entretanto, relativamente próximo aos parâmetros de configuração considerados, fato que restringe o seu escopo a tecnologias específicas e dificulta uma apreciação mais abstrata do sistema a ser gerenciado.

A ferramenta *Power* [Mont et al., 2000] traz uma abordagem diferente, a qual propicia suporte ao refinamento assistido de hierarquias de política a partir de políticas de alto nível de abstração até parâmetros de configuração. Não obstante, *Power* não oferece apoio à definição gráfica de políticas, usando, ao invés disso, moldes e assistentes pré-definidos. A visualização da distribuição dos participantes das políticas sobre o sistema torna-se, assim, obscura devido ao o uso exclusivo de uma representação textual das políticas.

[1]Detalhes a respeito do processo de refinamento automatizado e da validação realizada durante esse processo estão fora do escopo do presente trabalho e podem ser encontrados em [Porto de Albuquerque et al., 2005].

[2]O PCIM [Moore et al., 2001] é uma abordagem nessa direção, a qual difere da utilizada neste artigo por basear-se em uma sintaxe de políticas constituída por regras *condição-ação*, além de operar em um nível de abstração único.

Além disso, não se encontra em *Power*, nem tampouco em *Firmato*, a uma preocupação com fatores de escalabilidade, o que se reflete no fato de que essas abordagens não possuem mecanismos para uma representação modular do sistema modelado.

8. Conclusão

Apresentamos, no presente trabalho, uma técnica de modelagem que oferece suporte ao gerenciamento de sistemas de segurança de redes. Essa modelagem emprega um modelo do sistema estruturado verticalmente em níveis de abstração e obtém escalabilidade mediante a segmentação do sistema em subsistemas abstratos, a qual viabiliza que os processos de desenvolvimento e análise de modelos sejam efetuados de forma modular. Por basear-se em uma visão de sistema orientada a políticas, nossa modelagem proporciona, ainda, a consideração da arquitetura geral do sistema *vis-à-vis* as políticas de segurança que lhe são prescritas, possibilitando, assim, a apreciação dos meios pelos quais essas políticas serão implementadas no sistema.

A sistemática do mapeamento a partir de uma visão do sistema orientada a serviços até a obtenção de um diagrama de subsistemas abstratos foi exposta em detalhes, abarcando tanto a escolha dos elementos a serem representados na visão abstrata, como também o estabelecimento da correspondência entre esses elementos e os mecanismos do ambiente real. Além disso, a modelagem assistida e o refinamento de políticas automatizado viabilizados por nossa ferramenta de *software* foram também abordados.

Adicionalmente, realizamos um estudo de caso abrangente (com o cenário utilizado em [Geist, 2003]), do qual os exemplos deste trabalho foram extraídos. Os resultados têm mostrado melhorias na escalabilidade propiciadas pelo uso de nossa técnica de modelagem que são similares às obtidas nos exemplos aqui apresentados.

Atualmente estamos integrando técnicas de manipulação de grafos como visualização olho-de-peixe (*fischeye views* [Furnas, 1986]) e zoom semântico (*semantic zooming* [Köth and Minas, 2002]) à ferramenta, com o propósito de incrementar as características de exibição e navegação dos modelos. Trabalhos futuros planejados incluem a definição formal dos modelos e dos refinamentos de abstração.

Referências

Bartal, Y., Mayer, A. J., Nissim, K., and Wool, A. (2004). Firmato: A novel firewall management toolkit. *ACM Transactions on Computer Systems*, 22(4):381–420.

Bellovin, S. M. (1999). Distributed firewalls. *;login: magazine, special issue on security*.

Cheswick, W. R., Bellovin, S. M., and Rubin, A. D. (2003). *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison-Wesley, 2nd edition.

Furnas, G. W. (1986). Generalized fisheye views. In *Proceedings of ACM CHI'86 Conference on Human Factors in Computing Systems*, Visualizing Complex Information Spaces, pages 16–23.

Garfinkel, S. and Spafford, G. (1996). *Practical Unix & Internet Security*. O'Reilly & Associates, Sebastopol, CA, 2nd edition.

Geist, G. (2003). Model-based management of security services: Integrated enforcement of policies in company networks. Master's thesis, University of Dortmund, Germany. in German.

Köth, O. and Minas, M. (2002). Structure, abstraction, and direct manipulation in diagram editors. *Lecture Notes in Computer Science*, 2317.

Lupu, E. C. and Sloman, M. (1996). Towards a role based framework for distributed systems management. *Journal of Networks and Systemss Management, Plenum Press*.

Lück, I., Schäfer, C., and Krumm, H. (2001). Model-based tool-assistance for packet-filter design. In M. Sloman, J. Lobo, E. L., editor, *Proc. IEEE Workshop Policy 2001: Policies for Distributed Systems and Networks*, number 1995 in Lecture Notes in Computer Science, pages 120–136, Heidelberg. Springer Verlag.

Lück, I., Schönbach, M., Mester, A., and Krumm, H. (1999). Derivation of backup service management applications from service and system models. In R. Stadler, B. S., editor, *Active Technologies for Network and Service Management, Proc. DSOM'99*, number 1700 in Lecture Notes in Computer Science, pages 243–255, Heidelberg. Springer Verlag.

Lück, I., Vögel, S., and Krumm, H. (2002). Model-based configuration of VPNs. In Stadtler, R. and Ulema, M., editors, *Proc. 8th IEEE/IFIP Network Operations and Management Symposium NOMS 2002*, pages 589–602, Florence, Italy. IEEE.

McBride, D. (1998). Successful deployment of it service management in the distributed enterprise. White Paper, Hewlet-Packard Company.

Moffett, J. D. and Sloman, M. S. (1993). Policy hierarchies for distributed system management. *IEEE JSAC Special Issue on Network Management*, 11(9).

Mont, M., Baldwin, A., and Goh, C. (2000). POWER prototype: Towards integrated policy-based management. In Hong, J. and Weihmayer, R., editors, *Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS2000)*, pages 789–802, Hawaii, USA.

Moore, B., Ellesson, E., Strassner, J., and Westerinen, A. (2001). Policy core information model—version 1 specification. RFC 3060. Internet Engineering Task Force.

Porto de Albuquerque, J. and de Geus, P. L. (2003). A framework for network security system design. *WSEAS Transactions on Systems*, 2:139–144.

Porto de Albuquerque, J., Krumm, H., and de Geus, P. L. (2005). Policy modeling and refinement for network security systems. In *IEEE 6th International Workshop on Policies for Distributed Systems and Networks*, Stockholm, Sweden.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. (1996). Role-based access control models. *IEEE Computer*, 29(2):38–47.

Sloman, M. (1994). Policy driven management for distributed systems. *Journal of Network and Systems Management*, **2**:333.

Sloman, M. and Lupu, E. C. (2002). Security and management policy specification. *IEEE Network, Special Issue on Policy-Based Networking*, 16(2):10–19.

Wies, R. (1995). Using a classification of management policies for policy specification and policy transformation. In Sethi, A. S., Raynaud, Y., and Fure-Vincent, F., editors, *Integrated Network Management IV*, volume 4, pages 44–56, Santa Barbara, CA. Chapman & Hall.

Zwicky, E. D., Cooper, S., and Chapman, D. B. (2000). *Building Internet Firewalls*. O'Reilly and Associates, Sebastopol, CA, 2nd edition.