Representando Aspectos de Sistemas Operacionais na Abordagem de Gerenciamento Baseado em Modelos

Diogo Ditzel Kropiwiec[1], **Paulo Lício de Geus**[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 6176 – 13083-970 – Campinas – SP – Brazil

{diogo,paulo}@las.ic.unicamp.br

Abstract. *Managing the configuration of security mechanisms of today's computers environment is becoming increasingly complex, especially with large scale networks. Security administrators face the challenge of designing and maintaining security policies for a huge number of heterogeneous mechanisms and operating systems to ensure the protection of these environments. To support the configuration of both network and operating system security in one single model, this work presents an extension to the Model-Based Management applied to networks that include operating system policy management.*

Resumo. *O gerenciamento da configuração de mecanismos de segurança em ambientes computacionais está se tornando cada vez mais complexo, especialmente em redes de computadores de larga escala. Administradores de segurança encaram o desafio de projetar e manter políticas de segurança para um enorme número de mecanismos heterogêneos e diferentes sistemas operacionais para garantir a proteção desses ambientes. Para permitir o gerenciamento de configurações de segurança de sistemas operacionais e de rede em um único modelo, este trabalho apresenta uma extensão da abordagem de Gerenciamento Baseado em Modelos aplicado a redes de computadores que inclui gerenciamento de políticas de sistemas operacionais.*

1. Introdução

O Gerenciamento de segurança de grandes sistemas computacionais está se tornando cada vez mais complexo. A necessidade de proteção em ambientes computacionais requer a integração de novos mecanismos de segurança, e ao administrador fica relegada a complexa tarefa de configurar esses mecanismos para atingir os requisitos de segurança de redes organizacionais. Como novas vulnerabilidades e técnicas de invasão são descobertas a cada dia, o processo de gerenciamento de políticas requer uma contínua reconfiguração para se equiparar com a evolução das ameaças.

A evolução das tecnologias de segurança para sistemas computacionais nos últimos anos é significante, mas a mesma atenção não é dada aos modelos de gerenciamento de políticas para essas tecnologias. O gerenciamento de segurança de sistemas computacionais em grandes organizações requer do administrador a compreensão e a configuração de sistemas muito diferentes, como sistemas operacionais distintos e uma variedade de mecanismos de rede, e cada um desses pode utilizar diferentes modelos de segurança, linguagens e formatos para especificar a política. O administrador de

segurança precisa operar sintaxes de configuração complicadas e heterogêneas, muitas das quais não intuitivas e, em alguns casos, que podem levar a erros na especificação.

Neste cenário, não é de se surpreender que a maioria dos incidentes de segurança derive de erros de configuração [Oppenheimer et al. 2003]. Uma simples diferença de configuração entre dois mecanismos pode levar uma vulnerabilidade que afete toda a rede. Portanto, abordagens cujo foco é auxiliar no gerenciamento de múltiplos sistemas usando um único modelo são essenciais para melhorar o processo de configuração e reduzir vulnerabilidades causadas por erro de configuração.

Há um número considerável de abordagens para especificação de políticas com foco em simplificar e melhorar a tarefa do administrador de segurança enquanto ele projeta, impõe e mantém políticas de segurança. A maioria dos modelos heterogêneos foca em mecanismos de rede [Teo and Ahn 2007, Franqueira and van Eck 2006, Porto de Albuquerque et al. 2005b], e os poucos que focam em sistemas operacionais geralmente consideram apenas um subconjunto de sistemas operacionais que implementam um único modelo de políticas de segurança [Loscocco and Smalley 2001]. Estes modelos carecem de uma maneira consistente de especificar ambos os sistemas—operacionais e de rede—para ambientes computacionais heterogêneos.

Para evitar a criação de mais um modelo, nós escolhemos como ponto de partida a abordagem de Gerenciamento Baseado em Modelos (*Model-Based Management* - MBM) [Lück et al. 2002], uma vez que ele foi aplicado com sucesso para o contexto de redes [Porto de Albuquerque et al. 2005b] e provê ferramentas que auxiliam no projeto de políticas para ambientes de rede grandes e complexos [Porto de Albuquerque et al. 2005a]. Este modelo possui um conjunto interessante de características: ele suporta o projeto de políticas de segurança em diferentes níveis de abstração; utiliza refinamento e validação automática entre níveis de abstração auxiliando no processo de especificação; e utiliza uma representação gráfica que permite a manutenção de uma política consistente em todos os níveis de abstração.

Este artigo apresenta uma extensão para o modelo proposto em [Porto de Albuquerque et al. 2005b] que inclui abstrações de segurança de sistemas operacionais. O foco da extensão é permitir a especificação de uma política de sistemas operacionais abstrata em conjunto com a política de redes, de forma que a mesma possa ser especializada para diferentes sistemas operacionais, sem adicionar muito mais complexidade ao modelo original. Desta forma, o risco de vulnerabilidades causadas por políticas conflitantes no ambiente de redes e sistemas operacionais será reduzido.

O artigo é organizado da seguinte maneira: A Seção 2 apresenta os trabalhos relacionados, sendo que o MBM é apresentado em mais detalhes na Seção 3. Na Seção 4, apresentamos a extensão para sistemas operacionais projetada e as Seções 5 e 6 apresentam uma política exemplo e a conversão de parte dela para um sistema operacional específico a partir da nova extensão. A Seção 7 conclui o conteúdo apresentado e apresenta os trabalhos futuros relacionados.

2. Trabalhos relacionados

Outros projetos também são focados no desenvolvimento de modelos que gerenciem a segurança de redes e sistemas operacionais em uma única política. Na literatura de

políticas de segurança, a maioria das linguagens é específica para determinados mecanismos/sistemas e, portanto, não são viáveis para ambiente heterogêneos de larga escala. Dos modelos existentes, dois são apresentados a seguir.

O Chameleos-x [Teo and Ahn 2007] é um *framework* de políticas projetadas para distribuir políticas de segurança consistentemente tanto para mecanismos de rede com quanto sistemas operacionais. Foi projetado para ser flexível e extensível, e utiliza uma abordagem *bottom-up* para especificação de políticas. Extensões para diferentes sistemas podem ser adicionadas utilizando a meta-linguagem disponibilizada, o que permite ao administrador expandir o conjunto de regras base necessárias aos requisitos de segurança da rede. A meta-linguagem suporta a especificação de regras que podem ser convertidas diretamente para uma linguagem de política alvo, desta forma automatizando o processo de geração das configurações da política.

Apesar de suportar um conjunto variado de políticas de segurança, o Chameleos-x apresenta dois problemas principais: primeiro, ele necessita de diferentes subconjuntos de regras para cada sistema considerado, o que leva o administrador a especificar a mesma regra mais de uma vez para sistemas diferentes e, conseqüentemente, pode resultar em falhas na política; segundo, como a abordagem utilizada é *bottom-up*, a especificação de regras de alto-nível não pode ser feita diretamente no modelo.

O modelo proposto em [Franqueira and van Eck 2006] apresenta uma especificação formal de políticas utilizando o *framework* de Interação Governada por Leis para representar controle de acesso, redes e políticas de segurança física em um único modelo, sem a necessidade de interfaces e conexões entre domínios. Uma linguagem declarativa utilizando regras baseadas em eventos-condições-ações expressas em Prolog é utilizada para especificar a lógica da política, permitindo a verificação das regras da política e também a criação de rotinas arbitrariamente complexas para lidar com a inerente complexidade de sistemas computacionais de larga escala. Apesar das vantagens decorrentes do modelo unificado, existem alguns efeitos colaterais. Cada sistema coberto pela política requer a instalação de um *middleware* para impor as regras da política. Além disso, alguns detalhes da especificação podem tornar complexo o processo de verificação e correção de erros da política.

Estes dois modelos apresentam uma desvantagem significativa do ponto de vista do administrador de sistemas. O gerenciamento de políticas se torna cada vez mais difícil à medida que o ambiente computacional se expande devido ao uso de arquivos de configuração em formato texto. Sem o auxílio de uma ferramenta gráfica, o administrador precisa confiar em ferramentas de verificação automática ou então verificar manualmente todos os arquivos de especificação da política.

Dentre os modelos que se utilizam de interface gráfica para representação de políticas, dois se destacam. O Ponder [Damianou et al. 2002] provê um conjunto de ferramentas para especificação, distribuição e gerenciamento de políticas. A linguagem Ponder suporta a definição de políticas de gerenciamento e segurança e é baseada no gerenciamento de domínios, que no modelo são as estruturas hierárquicas utilizadas para agrupar os objetos gerenciados. O protótipo da ferramenta inclui um navegador de domínios que permite a visualização da política em uma estrutura de árvore hiperbólica. Apesar do foco da abordagem ser em políticas, este trabalho não provê uma representação da arqui-

tetura do sistema a ser gerenciado, dificultando o trabalho do administrador em associar as políticas ao seu modelo mental do sistema.

O outro modelo baseado em visualização é a abordagem de Gerenciamento Baseado em Modelos (MBM) [Lück et al. 2002, Porto de Albuquerque et al. 2005b]. O MBM apresenta a política em diferentes níveis de abstração, criando uma hierarquia de políticas que permite ao administrador especificar a política usando uma abordagem *top-down*, permitindo criar um mapeamento direto entre os requisitos de segurança organizacionais e a política de configuração derivada destes. Como o corrente trabalho propõe uma extensão para o modelo de MBM, ele será explicado em mais detalhes na próxima seção.

3. Gerenciamento Baseado em Modelos

A abordagem de Gerenciamento Baseado em Modelos (*Model-Based Management* - MBM) [Lück et al. 2002] tem por objetivo suportar o gerenciamento baseado em modelo pelo uso de um modelo orientado a objetos do sistema a ser gerenciado. Focando no gerenciamento de segurança de redes, o MBM foi aplicado com sucesso nesse ambiente [Porto de Albuquerque et al. 2005b], e a descrição a seguir se baseia neste último artigo.

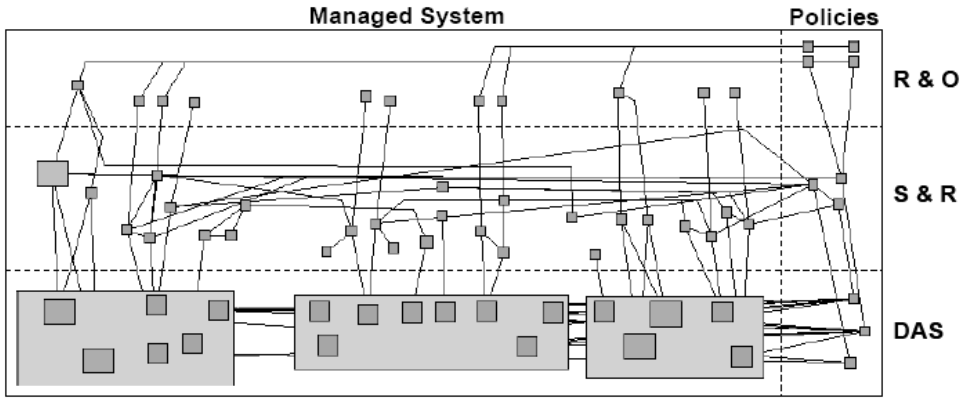

Figura 1. Visão Geral do Modelo

A estrutura do modelo é apresentada na Figura 1, onde três níveis de abstração podem ser diferenciados: Papéis e Objetos (*Roles & Objects* - RO), Sujeitos e Recursos (*Subjects & Resources* - SR) e Diagrama de Subsistemas Abstratos (*Diagram of Abstract Subsystems* - DAS). Cada nível é um refinamento do nível superior, no sentido de uma hierarquia de políticas [Moffett and Sloman 1993], e complementa o modelo em mais detalhes em um menor nível de abstração da visão do sistema. O modelo é também dividido em dois conjuntos: o Sistema Gerenciado (*Managed System*), que representa o sistema real; e as políticas (*Policies*), que representam as políticas de segurança que regulamentam o sistema.

O nível mais alto (RO) é baseado nos conceitos de Controle de Acesso Baseado em Papéis (*Role-Based Access Control* - RBAC) [Sandhu et al. 2000] e representa o modelo numa visão orientada a negócios da rede. Existem três classes principais nesse nível: *Roles* (Papéis), *Objects* (Objetos) e *AccessModes* (Modos de Acesso). *Roles* representam grupos de usuários com o mesmo conjunto de permissões e responsabilidades no ambiente modelado. Qualquer elemento do ambiente que pode ser sujeito a controle de acesso é definido por um *Object*; *AccessModes* representam as formas (permissões) de acessar

objetos. Uma quarta classe, *AccessPermission* (Permissão de acesso), conecta as três classes principais, representando que um sujeito personificando um *Role* pode acessar um *Object* particular da maneira definida por um *AccessMode*.

O segundo nível (SR) oferece uma visão do sistema do ponto de vista dos serviços que o sistema provê e, por conseguinte, consiste de um conjunto maior de classes. Objetos dessas classes representam: pessoas trabalhando no ambiente modelado (*User*); ambientes ou configurações típicas nas quais um usuário pode agir no sistema (*SubjectTypes*); serviços na rede utilizados para acessar recursos (*Services*); uma dependência de um serviço por outro serviço (*ServiceDependency*); e recursos da rede (*Resources*).

O nível inferior (DAS) descreve a estrutura do sistema como um todo em uma visão modular, isto é, ela separa o sistema em seus blocos constituintes e indica suas interconexões. Dessa forma, um DAS é um grafo composto de Subsistemas Abstratos (*Abstract Subsystems* - AS), que por sua vez contêm uma visão abstrata de um determinado segmento do sistema. Existem quatro elementos principais nesse nível: *Actors*—grupo de indivíduos que possuem um comportamento ativo; *Mediators*—elementos que intermediam a comunicação; *Targets*—elementos passivos que respondem às chamadas de *Actors*; e *Connectors*—interfaces entre dois ASs distintas.

Adicionalmente, cada AS em um DAS está também associado a uma visão detalhada dos mecanismos presentes no sistema, como um quarto nível de abstração. Essa visão expandida engloba objetos que representam terminais, processos e interfaces de rede do sistema e que auxiliam no processo de geração automática de parâmetros de configuração. Essa visão expandida está relacionada ao nível Processos e Recursos (*Process & Resources* - PR) dos primeiros trabalhos sobre MBM [Lück et al. 2002]

Um modelo de exemplo é apresentado na Figura 2. Ele representa uma visão simplificada da regra "permitir que funcionários acessem o servidor web da empresa". A *AccessPermission* no nível superior (RO) é uma abstração dessa regra e é refinada através dos níveis até uma representação mais próxima do sistema real, que inclui terminais, interfaces de rede, etc.

O MBM também provê uma ferramenta de modelagem que auxilia na edição, visualização e configuração de políticas de segurança. Ela auxilia o usuário por meio de um editor gráfico com funções adicionais para a verificação de restrições dependentes do modelo. A ferramenta suporta refinamento automático da política de segurança abstrata (RO), passando pelos níveis intermediários (SR e DAS), até atingir arquivos de configuração para os mecanismos de segurança suportados. Usando o conceito de foco e contexto [Card et al. 1999], como visão olho-de-peixe e zoom semântico, ela auxilia o administrador de segurança ao exibir informações que são mais úteis ao foco de sua análise e, conseqüentemente, reduz o risco de falhas de configuração.

Apesar do MBM ter várias características de segurança do ponto de vista da administração de políticas, especialmente a representação da política em múltiplos níveis de abstração e a automação do processo de refinamento, ele carece de um importante aspecto da segurança de sistemas computacionais: o sistema operacional. Se a segurança de um sistema operacional em uma rede é comprometida, os pressupostos sobre a segurança da rede não são mais válidos [Loscocco et al. 1998]. Adicionalmente, alguns aspectos tratados pelo modelo não são facilmente controláveis sem o auxílio do sistema operacio-

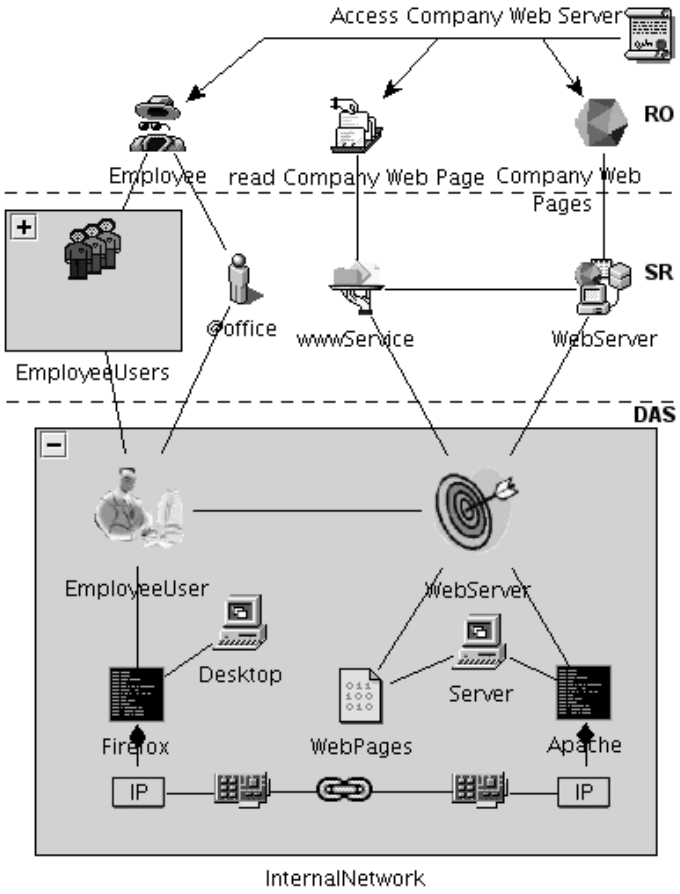

Figura 2. Modelo exemplo da política "Funcionários podem acessar o servidor web da empresa" utilizando MBM. Cada nível (RO, SR e DAS) é separado por uma linha tracejada

nal. Por exemplo, a maioria dos protocolos de rede não suporta identificação de usuário, o que implica na identificação do usuário em termos das credenciais de seu computador (Endereço IP, por exemplo), o que reduz a cobertura da política. Para também representar o sistema operacional, uma extensão foi proposta e é detalhada na próxima seção.

4. Representando Políticas de Sistemas Operacionais usando MBM

Dada as vantagens provenientes do uso de MBM, iniciou-se um trabalho de pesquisa para analisar a viabilidade de introduzir no modelo aspectos de sistemas operacionais, aumentando assim a cobertura do modelo original. Nesta seção, será apresentado um breve resumo dos aspectos de segurança de sistemas operacionais relacionado ao ambiente de redes, seguido pela descrição da extensão desenvolvida para o MBM.

Sistemas operacionais modernos permitem controlar o acesso de usuários e aplicações aos recursos de sistema com maior granularidade do que seus antecessores. Regras de segurança permitem não só definir domínios mais isolados entre usuários, mas também entre aplicações executadas por usuários e aplicações de sistema (serviços e/ou servidores). Dessa forma, pode-se obter maior isolamento entre os diversos serviços providos pelo sistema operacional, evitando que o comprometimento de uma aplicação afete outras aplicações, e assim restringindo o impacto de invasões bem sucedidas.

Entretanto, a maior granularidade dificulta o trabalho do administrador de segurança, uma vez que aumenta significativamente o número de regras a serem geridas—em especial em sistemas operacionais, que precisam controlar um número elevado de processos e recursos—ao ponto de impossibilitar a verificação e validação de políticas devido ao número de regras. Mesmo considerando que as regras suportam agrupamento de elementos (como definição de grupos, tipos e outros), ainda assim é uma tarefa complexa.

Ao se considerar que em o ambiente de redes mais comum em grandes organizações contam com mais de um tipo de sistema operacional, e que normalmente esses sistemas possuem formas distintas de especificar as regras de segurança, o trabalho do administrador torna-se mais complicado, uma vez que ele precisa garantir que todos os sistemas respondem a mesma política de segurança. E é exatamente porque o MBM lida tão bem com a heterogeneidade dos elementos de rede que o escolhemos como base para nossa extensão.

Em nossa extensão, nos empenhamos em representar a maioria dessas abstrações sem adicionar complexidade ao modelo original. Estratégias de visualização do modelo original foram empregadas a extensão, de forma a permitir ao administrador visualizar as regras relevantes ao sistema operacional quando necessário, e se abstrair delas ao analisar os aspectos de rede.

O foco principal da extensão foi o controle de acesso. Informações sobre quais usuários podem acessar determinadas máquinas, e quais serviços podem se conectar pela rede podem ser obtidas diretamente do modelo original, porém informações referentes ao controle de acesso do sistema operacional de cada máquina, como quais processos podem acessar quais recursos do sistema e com quais permissões, e quais processos/usuários são responsáveis por gerenciar os serviços precisaram ser mapeados.

Para representar abstrações do sistema operacional, o modelo de *Domain and Type Enforcement* (DTE)[Walker et al. 1996] foi escolhido. Entre as características interessantes, ele permite agrupar os elementos do sistema em abstrações (tipos e domínios) e expressar as relações entre eles em termos de permissões. O modelo já é utilizado em alguns sistemas operacionais (o SELinux é mais usado entre eles [Loscocco and Smalley 2001]), o que garante uma conversão direta do modelo para uma política real. Como nosso objetivo é permitir a representação de políticas de segurança para um conjunto maior de sistemas operacionais, pretendemos adicionar informação suficiente a estes elementos para permitir a conversão também para sistemas operacionais baseados em ACL (como o Windows [Swift et al. 2002]), dessa forma expandindo o espectro de sistemas cobertos pela extensão.

Alguns elementos de sistemas operacionais já são representados no MBM. Usuários são mapeados pelo objeto *User*, e grupos de usuários com mesmas permissões derivam de um objeto *Role*. Processos de rede, arquivos de dados e conexões de rede são representados pelos objetos correspondentes no nível DAS, representando todos os aspectos relevantes à rede necessários ao sistema operacional para nossa extensão. Elementos do sistema operacional não relacionados à rede têm de ser mapeados em novos objetos e arestas no modelo, como comunicação entre processos (IPCs), arquivos relacionados ao controle e segurança dos processos (arquivos de configuração, arquivos de log, etc.) e dispositivos e outros recursos de sistema.

Uma vez que especificar uma política completa de um sistema operacional envolve a representação de centenas de regras, adotamos a estratégia usada pela *Targeted Reference Policy* do SELinux [Mayer et al. 2006]. Essa política tem por objetivo isolar processos de alto risco de outros processos não sujeitos a nenhuma política específica. Dessa forma, o administrador pode focar nos serviços importantes, como processos acessíveis pela rede—e, portanto, sujeitos a ataques—, relegando processos locais a serem gerenciados por medidas de segurança padrão. Se especificado pelos requisitos de segurança, o administrador será capaz de também gerenciar esses serviços no mesmo modelo.

Em nossa extensão, domínios derivam de *Actors*, *Mediators* e *Targets*. Cada processo associado a um desses será mapeado ao domínio respectivo que representam. No MBM, o objeto *process* representa mais uma aplicação do que um processo propriamente dito. Muitas aplicações dependem de processos auxiliares para executar seus serviços, e do ponto de vista de sistemas operacionais, seus domínios devem ser restringidos ainda mais para reduzir o impacto de ataques a esses processos. Para representar esse aspecto sem adicionar complexidade à visão de rede do ambiente, cada *process* possui agora uma visão expandida que representa os relacionamentos entre o processo principal e seus processos auxiliares. Cada um desses processos pode estar associado ao seu próprio subdomínio, se necessário, permitindo a especificação de políticas de segurança mais restritas. Arestas entre processos na visão expandida representam regras de transição e de acesso entre eles. Adicionalmente, tanto os objetos *process* quanto as arestas interligando-os possuem atributos que definem informações adicionais sobre os domínios e subdomínios, e as permissões/restrições de acesso e transição entre eles, respectivamente.

No modelo DTE, um tipo representa um conjunto de arquivos e/ou dispositivos de sistema com uma finalidade específica. Em nossa extensão, tipos derivam de duas origens principais: do objeto *file*, e de todos os objetos relacionados à rede da visão expandida de um AS. Além de representar arquivos visíveis do ponto de vista da rede, o primeiro representa também em nossa extensão arquivos e dispositivos relevantes aos serviços do sistema operacional—como arquivos de configuração e temporários—, porém esses só serão apresentados na visão expandida do *process* a eles associado. Atributos associados a esses arquivos especificam quais arquivos e dispositivos do sistema são representados por um *file* específico.

Arestas conectando *process* (domínios) e *files* (tipos) representam permissões de acesso, e o conjunto de permissões é definido nos atributos de cada aresta. Como um tipo pode ser acessado por mais de um domínio, ao invés de obscurecer o modelo adicionando arestas cruzando bordas entre domínios, um determinado tipo pode aparecer mais de uma vez no modelo, porém com um único conjunto de atributos.

Do ponto de vista da administração do sistema operacional, dois tipos de arquivos precisam ser diferenciados dos demais arquivos do sistema, e por isso um novo objeto para representar cada um foi criado: o objeto *config file* (arquivo de configuração) e o objeto *log file* (arquivo de *log*). Eles foram adicionados para aproveitar as funcionalidades providas pela ferramenta de edição de políticas do MBM. A meta-linguagem utilizada para representar a linguagem da política permite a especificação de restrições sobre objetos e arestas interligando-os, restringindo as conexões e atributos possíveis associados a eles. Novas regras serão associadas à linguagem para restringir o acesso de proces-

sos "normais" (que serão diferenciados mais adiante) a esses arquivos diferenciados, de forma que apenas permissões de leitura sejam permitidas entre os objetos *process* e *config files*, e apenas permissões de edição ao final do arquivo sejam permitidas entre os objetos *process* e *log file*. Essas restrições evitarão que privilégios excessivos sejam dados a uma aplicação que possam resultar em vulnerabilidades no sistema.

As restrições envolvendo esses arquivos especiais criaram a necessidade de um novo tipo de objeto para representar processos com privilégios administrativos: o objeto *controller*. O *controller* representa processos administrativos que possuem permissões de controle sobre a execução de processos associados a serviços do sistema e também de edição sobre arquivos de configuração (entre outras permissões), e por isso podem ser associados somente a *Actors* e *Mediators* do modelo. Por utilizar uma representação diferente do processo comum, os processos administrativos podem ser facilmente identificados no modelo e estão restritos a domínios controlados por usuários legítimos da rede em questão, mitigando o risco de um atacante externo utilizar um serviço como ponto de entrada na rede.

Na próxima seção, uma política exemplo utilizando essa extensão é apresentada.

5. Política exemplo

Baseado na política apresentada na Figura 2, um novo requisito de segurança—"permitir ao administrador gerenciar o servidor Apache"—foi adicionado a política para representar a tarefa administrativa sobre o Apache. A Figura 3 apresenta o nível DAS dessa nova política do ponto de vista da rede. Três novos elementos estão presentes: o *Actor* `AdminUser`, o *Controller* `ConfigShell` e o *config file* `ApacheConfig`.

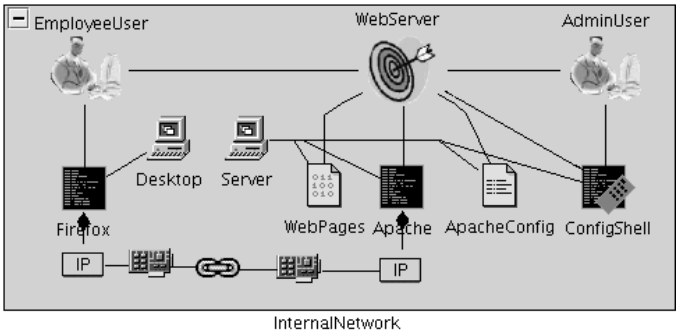

Figura 3. Nível DAS da política exemplo.

Do ponto de vista da rede, não são representadas diretamente as relações entre processos e arquivos, somente a quais máquinas eles pertencem, quais possuem acesso a rede e quais os elementos do DAS (*Actors*, *Mediators* e *Targets*) estão associados a eles. A relação entre o processo `ConfigShell` e os arquivos de configuração ficam subentendidas pelo fato do primeiro ser um elemento *Controller*. Apesar de restrita do ponto de vista do sistema operacional, essa forma de visualização permite ao administrador identificar rapidamente quais aplicações tem acesso direto a rede, de forma que ele possa configurar os mecanismos de rede para protegê-las de ataques.

A especificação de uma política de sistema operacional pode ser visualizada através da visão expandida de um *process*. A Figura 4 apresenta a visão expandida dos processos associados ao *target* `WebServer` (`Apache` e `ConfigShell`).

O processo `ConfigShell` é relativamente simples. Como pode ser observado na Figura 4, é permitido ao processo acessar o arquivo de configuração `ApacheConfig` e controlar a execução do processo `Apache`. Por padrão, a permissão de acesso básica de um *controller* a um *config file* é de leitura e escrita, porém uma política mais ampla ou restritiva pode ser definida atualizando os atributos da aresta que ligam os dois elementos.

Observe que não está sendo representado regras específicas a um determinado sistema operacional, como acesso a bibliotecas compartilhadas, outros recursos de sistema não relevantes a segurança, etc. Para reduzir a sobrecarga do modelo com milhares de regras, outras estratégia da *Targeted Reference Policy* do SELinux [Mayer et al. 2006] está sendo utilizada. Cada processo possui um conjunto padrão de permissões requeridas à execução do processo em cada sistema, que é definido em outro lugar. Se necessário, o administrador pode sobrescrever essas regras para um determinado processo especificando um conjunto mais detalhado de regras usando a mesma representação da extensão. Dessa forma, a política pode ser mantida abstrata o suficiente para representar diferentes sistemas operacionais, e ainda assim considerar aspectos específicos a cada um deles.

O processo `Apache` possui um conjunto de regras mais complexo. Na política de exemplo, o `Apache` possui acesso não apenas aos arquivos visíveis à rede, mas também a arquivos temporários e de *log*. Ele pode também executar dois processos auxiliares, os interpretadores de Perl e PHP (arquivos associados a estes não são apresentados). As permissões de somente leitura a arquivos de configuração e de edição somente ao final de arquivos de *log* podem ser impostas pela ferramenta do MBM, mas permissões específicas a outros arquivos e processos precisam ser definidas nos atributos das arestas.

A partir da Figura 4, dois domínios principais e cinco tipos podem ser identificados. Os domínios correspondem aos processos `Apache` e `ConfigShell`. Os tipos correspondem aos quatro objetos *file* e à conexão de rede. A lista de arquivos representada por cada *file* é especificada em seus atributos.

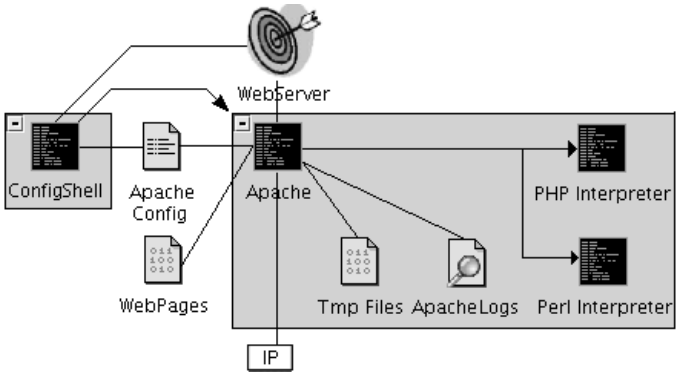

Figura 4. Visão expandida dos processos ConfigShell e Apache.

6. Conversão de política para SELinux

Baseado no modelo apresentado pelas Figuras 3 e 4, são apresentados trechos da política gerada para a linguagem de especificação do SELinux.

Para simplificar tanto a notação quanto a geração das regras, foi utilizado o conjunto padrão de macros do SELinux [Smalley 2003], pois permite a definição das regras

em um conjunto menor e mais claro. A padronização de sufixos utilizada também foi mantida: os sufixos _u, _r representam nomes de usuários e de papéis, respectivamente. O sufixo _t representa tanto domínios quanto tipos.

Na Figura 5 é apresentado o conjunto de regras geradas a partir da definição do processo Apache. Para cada processo, dois tipos são criados: <process name>_t e <process name>_exec_t (linhas 1 e 2). O primeiro representa o domínio do processo em execução no sistema operacional e o segundo representa o tipo do arquivo binário da aplicação armazenada em disco. Processos derivados de serviços e recursos do nível SR são inicializados pelo sistema e, portanto, devem estar associados ao papel system_r (linha 3). A macro init_daemon_domain define que um processo no domínio init_daemon pode iniciar um processo no domínio apache_t a partir de uma executável do tipo apache_exec_t (linha 4).

```
1: type apache_t;
2: type apache_exec_t;
3: role system_r types apache_t;
4: init_daemon_domain(apache_t,apache_exec_t)
```

Figura 5. Regras do domínio Apache geradas a partir da política exemplo

As permissões de acesso aos tipos a partir de um domínio derivam das restrições impostas pela meta-linguagem e dos atributos definidos nas arestas e tipos. Caso nenhuma restrição seja definida, o modelo padrão permite a leitura e escrita dos arquivos. A Figura 6 apresenta a política para o tipo TmpFiles e as permissões de acesso do domínio Apache. As linhas 1 e 2 definem o tipo e inicializam atributos comuns a todos os arquivos. As linhas 3 e quatro estabelecem que o domínio apache_t pode acessar arquivos e diretórios do tipo tmpfiles_t, e a linha 5 define que o domínio apache_t pode criar arquivos do tipo tmpfiles_t.

```
1: type tmpfiles_t;
2: files_file(tmpfiles_t)
3: manage_dirs_pattern(apache_t,tmpfiles_t,tmpfiles_t)
4: manage_files_pattern(apache_t,tmpfiles_t,tmpfiles_t)
5: files_tmp_filetrans(apache_t, tmpfiles_t, { file dir })
```

Figura 6. Regras sobre o tipo TmpFile

No caso do tipo WebPages, não cabe ao Apache modificar seu conteúdo, portanto os atributos da aresta conectando ambos devem limitar o Apache a ler das WebPages. Caso sejam utilizadas as macros de somente leitura fornecidas pelo SE-Linux, as regras geradas pela ferramenta seriam as apresentadas na Figura 7.

Tipos de arquivos especiais utilizam um conjunto predefinido de regras, simplificando a especificação de políticas pelo administrador. Por exemplo, o acesso aos arquivos de *log* pelo domínio Apache deve ser limitado à leitura e a edição ao final (*append*), caso contrário um invasor poderia tentar remover os registros de sua invasão. A Figura 8 apresenta a política gerada para o tipo ApacheLogs. De maneira similar, o tipo ApacheConfig seria limitado a leitura pelo processo Apache.

```
1: type webpages_t;
2: allow apache_t webpages_t:dir list_dir_perms;
3: read_files_pattern(httpd_t,webpages_t,webpages_t)
4: read_lnk_files_pattern(httpd_t,webpages_t,webpages_t)
```

Figura 7. Regras sobre o tipo WebPages

```
1: logging_log_file(apachelogs_t)
2: allow apache_t apachelogs_t:dir setattr;
3: create_files_pattern(apache_t,apachelogs_t,apachelogs_t)
4: append_files_pattern(apache_t,apachelogs_t,apachelogs_t)
5: read_files_pattern(apache_t,apachelogs_t,apachelogs_t)
6: read_lnk_files_pattern(apache_t,apachelogs_t,apachelogs_t)
```

Figura 8. Regras sobre o tipo ApacheLogs

O acesso a aplicações auxiliares pode ser feito mantendo o mesmo domínio do processo principal, ou então pode ser definido um novo subdomínio para a aplicação auxiliar. Na política exemplo, supondo que os atributos entre os processos `Apache` e `PhpInterpreter` definam que o segundo deve ser executado em seu próprio subdomínio, um novo conjunto de regras seria gerado (Figura 9). As linhas 1 e 2 criam os dois tipos básicos relacionados ao processo, enquanto as linhas 3, 4 e 5 definem as regras de transição e a linha 6 associa o domínio `PhpInterpreter` ao papel `system_r`.

```
1: type apachephp_t;
2: type apachephp_exec_t;
3: domain_type(apachephp_t)
4: domain_entry_file(apachephp_t,apachephp_exec_t)
5: domtrans_pattern(apache_t, apachephp_exec_t, apachephp_t)
6: role system_r types apachephp_t;
```

Figura 9. Regras de transição de domínio entre o Apache e o PhpInterpreter

Não são incluídos aqui as regras geradas para o domínio `ConfigShell` e as permissões de acesso aos tipos associados a este domínio, pois os mesmos seguem o modelo já apresentado acima. A geração de regras da política derivadas das configurações de rede do sistema operacional associadas ao processo `Apache` também não são apresentadas, pois a inclusão de algumas macros ainda são necessárias para permitir a sua representação. Entretanto, os resultados obtidos até o momento demonstram que um modelo similar ao já usado para arquivos e domínios poderá ser empregado também para os aspectos de rede.

7. Conclusão

Neste artigo foi apresentada uma extensão para o modelo MBM que inclui abstrações de segurança de sistemas operacionais, e apresenta a conversão de uma política exemplo para a linguagem de especificação do sistema operacional SELinux. Apesar de novos elementos terem sido acrescidos ao modelo, e ao acréscimo do número de regras a serem representadas, não houve aumento na complexidade da modelagem dos aspectos de rede,

uma vez que as novas regras são representadas em uma visão expandida de processos. Apesar de nosso modelo poder resultar em uma política de acesso mais amplo por não mapear todas as especificidades de um modelo de segurança de sistema operacional específico, ele oferece uma forma concisa de mapear mais de um sistema operacional, e a relação custo-benefício parece ser suficientemente razoável para a maioria dos ambientes computacionais.

Trabalhos futuros envolvem estender o meta-modelo do MBM para suportar a geração de políticas de sistemas operacionais não só para o SELinux, mas para outras linguagens de especificações de políticas de segurança de sistemas operacionais, de forma a ampliar o escopo das políticas representadas. Aspectos de sistemas operacionais não considerados em nossa simplificação também devem ser considerados, viabilizando sua inclusão e dessa forma criando meios de definir políticas mais restritivas com relação ao princípio de privilégio mínimo. Finalmente, à medida que aumentamos a granularidade do modelo de segurança, modelos de segurança mais recentes, como o UCONabc [Park and Sandhu 2004] podem ser agregados ao MBM, ampliando a aplicabilidade do modelo.

Referências

Card, S. K., Mackinlay, J. D., and Shneierman, B. (1999). *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers, San Francisco, CA, USA.

Damianou, N., Dulay, N., Lupu, E., Sloman, M., and Tonouchi, T. (2002). Tools for domain-based policy management of distributed systems. In *In proceedings of Network Operations and Management Symposium, 2002.*, pages 203 – 217.

Franqueira, V. N. L. and van Eck, P. A. T. (2006). Towards alignment of architectural domains in security policy specifications. In *Proccedings of the 8th International Symposium on System and Information Security*, Sao Jose dos Campos, Brazil. Fundacao Casimiro Montenegro Filho - CTA/ITA.

Loscocco, P. and Smalley, S. (2001). Integrating flexible support for security policies into the linux operating system. In *Proccedings of the FREENIX Track: 2001 USENIX Annual Technical Conference*, Boston Mass.

Loscocco, P. A., Smalley, S. D., Muckelbauer, P. A., Taylor, R. C., Turner, S. J., and Farrel, J. F. (1998). The inevitability of failure: The flawed assumption of security in modern computing environment. In *Proceedings of the 21st National Information Systems Security Conference*, pages 303–314.

Lück, I., Vögel, S., , and Krumm, H. (2002). Model-based configuration of vpns. In *NOMS 2002: Proceedings of the 8th IEEE/IFIP Network Operations and Management Symposium*, pages 243–255, London, UK. Springer-Verlag.

Mayer, F., MacMillan, K., and Caplan, D. (2006). *SELinux by Example: Using Security Enhanced Linux (Prentice Hall Open Source Software Development Series)*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Moffett, J. J. D. and Sloman, M. S. (1993). Policy hierarchies for distributed systems management. *IEEE JSAC Special Issue on Network Management*, 11(9):1404–1414.

Oppenheimer, D., Ganapathi, A., and Patterson, D. (2003). Why do internet services fail, and what can be done about it.

Park, J. and Sandhu, R. (2004). The uconabc usage control model. *ACM Trans. Inf. Syst. Secur.*, 7(1):128–174.

Porto de Albuquerque, J., Isenberg, H., Krumm, H., and de Geus, P. L. (2005a). Improving the configuration management of large network security systems. In *Proceedings of 16th IFIP/IEEE Internation Workshop on Distributed Systems: Operations and Management, DSOM 2005*, Barcelona, Spain.

Porto de Albuquerque, J., Krumm, H., and de Geus, P. L. (2005b). Policy modeling and refinement for network security systems. In *POLICY '05: Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks*, pages 24–33, Washington, DC, USA. IEEE Computer Society.

Sandhu, R., Ferraiolo, D., and Kuhn, R. (2000). The nist model for role-based access control: Towards a unified standard. In *Proceedings of the 5th ACM Workshop on Role-Based Access Control*, pages 47 – 63, Berlim, Germany. ACM Press, New York, NY, USA.

Smalley, S. D. (2003). Configuring the selinux policy. Technical report, National Security Agency of United States of America.

Swift, M. M., Hopkins, A., Brundrett, P., Van Dyke, C., Garg, P., Chan, S., Goertzel, M., and Jensenworth, G. (2002). Improving the granularity of access control for windows 2000. *ACM Trans. Inf. Syst. Secur.*, 5(4):398–437.

Teo, L. and Ahn, G.-J. (2007). Managing heterogeneous network environments using an extensible policy framework. In *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 362–364, New York, NY, USA. ACM.

Walker, K. W., Bagder, D. F., Petkac, M. J., Sherman, L., and Oostendorp, K. A. (1996). Confining root programs with domain and type enforcement. In *Proceedings of The 6th USENIX Security Symposium*, San Jose, California.