Linguagem Lógica Formal para Expressar Segurança em Ambientes Pervasivos

Leonardo Mattes, Luis G. Kiatake, Eder Antonio R. Marques , João Antonio Zuffo

Laboratório de Sistemas Integráveis (LSI) – University of São Paulo (USP)
{leo, kiatake ,ederarm, jazuffo}@lsi.usp.br

Resumo: *Aplicações em meios pervasivos necessitam de modelos de seguranças coordenados por políticas mais flexíveis que permitam a definição dinâmica dos mecanismos a serem adotados em um processo de comunicação. O presente trabalho apresenta a LEPS (Linguagem Lógica para Expressar Políticas de Segurança), linguagem a ser utilizada por sistemas pervasivos heterogêneos para representar e fazer cumprir diferentes modelos de políticas para os serviços de controle de acesso, autenticação, integridade, privacidade, auditoria e irretratabilidade. No intuito de fornecer uma visão prática, também é apresentado um estudo de caso da utilização da LEPS para especificar e manter a segurança em uma rede corporativa.*

Abstract*: Pervasive applications need security models driven by more flexible policies that allow a dynamic definition of the mechanisms to be adopted in a communication. This paper presents LEPS (Linguagem Lógica para Expressar Políticas de Segurança – Logical Language to Express Security Policies), a language to be used by heterogeneous pervasive systems to represent and enforce different polices models for services like access control, authentication, integrity, privacy, auditing and non-repudiation. In order to provide some hints for a practical use of LEPS, a study case is presented for LEPS application in a corporative network.*

1. Introdução

Sistemas pervasivos são caracterizados pela integração de diversos dispositivos (PDAs, PCs, celulares e aparelhos domésticos) por meio de diferentes tecnologias de redes (Ethernet, Wireless, ATM). Aplicações em tais sistemas exigem mecanismos de segurança mais flexíveis e configuráveis do que os tradicionais. Por exemplo: mecanismos específicos de autenticação, privacidade e integridade podem vir a inviabilizar sistemas que combinem de forma alternada necessidades de alto nível de segurança e baixo consumo de recursos computacionais [Peikari 2003]; mecanismos de controle de acesso baseados em modelos específicos não oferecem respostas às necessidades de uma rede com recursos heterogêneos [Jajodia 97]; dispositivos do tipo *firewall* não produzem efeito em redes com tecnologia *wireless*, pois estas não permitem um controle centralizado de acesso à rede.

Uma solução pode provir de modelos de segurança baseados em políticas, uma vez que estes separam a implementação da escolha comportamental, permitindo que, em um dado processo de comunicação, regras, parâmetros e mecanismos possam ser estabelecidos de forma dinâmica e flexível [Undercoffer 2003]. O presente trabalho apresenta LEPS (Linguagem Lógica para Expressar Políticas de Segurança), linguagem que tem por objetivo representar e fazer cumprir modelos flexíveis de políticas de segurança em sistemas pervasivos heterogêneos. É apresentado um estudo de como a

utilização de LEPS pode coordenar o oferecimento de segurança em uma rede corporativa pervasiva.

Esse artigo é estruturado da seguinte forma: na seção 2 temos um estudo de caso de LEPS para coordenar segurança em uma rede corporativa que permeará as demais seções; a seção 3 apresenta os trabalhos relacionados estudados; a seção 4 descreve LEPS de acordo com os serviços que oferece suporte; por final são apresentados as conclusões e trabalhos futuros.

2. Modelo de Segurança

Nesta seção apresentamos a rede *heterogenea.com* que servirá de exemplo para ilustrar as capacidades da linguagem LEPS em representar e fazer cumprir políticas de segurança flexíveis. Tal rede, representada pela figura 1, possui dispositivos dos tipos PCs - *personal computer*, LAPTOPs e PDAs conectados via rede local, wireless ou Internet. Nesse cenário, para permitir que os serviços da rede possam ser oferecidos sem comprometer o desempenho e a segurança, num contexto de comunicação, o sistema deve ponderar sobre o poder computacional dos dispositivos envolvidos e o tipo de acesso utilizado, permitindo que PDAs e celulares façam uso de mecanismos mais leves e que conexões vindas de redes wireless ou da Internet demandem maior nível de segurança.

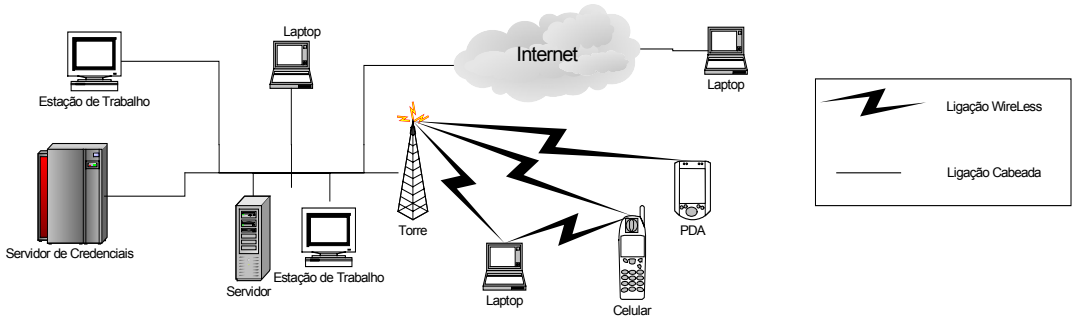

FIG. 1 Rede corporativa heterogênea: *heterogenea.com*

Para ilustrar a capacidades de LEPS, adotou-se um sistema de segurança para a rede *heterogenea.com* com as seguintes características:

- para que duas entidades possam estabelecer um canal de comunicação, é realizado primeiramente os processo de autenticação e troca segura de credenciais de acesso;
- em um processo de comunicação, cada entidade da rede toma suas decisões de segurança por meio de um interpretador LEPS, tendo como parâmetros suas políticas de segurança e as informações contidas nas credenciais de acesso da entidade parceira;
- a credencial de acesso de uma entidade é escrita conforme a sintaxe de LEPS e contem as políticas de segurança da entidade e mais as seguintes variáveis de contextualização: data e hora da autenticação na rede, método de autenticação utilizado; meio de acesso (rede local, wireless, Internet);
- o servidor de credenciais é responsável em autenticar e distribuir credenciais de acesso a todas as entidades que desejem oferecer e/ou consumir recursos na rede. O trabalho [Mattes 2003] apresenta uma análise comparativa entre dois modelos para tal processo.

3. Trabalhos Relacionados

Entre os trabalhos estudados, a linguagem LEA [Jajodia 97] utiliza-se de um formalismo matemático para representar os diferentes modelos de controle de acesso, prevendo a associação de atributos a usuários, grupos e papéis. Contempla também a utilização de regras configuráveis para derivação de privilégios. No que diz respeito a controle de acesso, boa parte da especificação de LEPS assemelha-se a LEA, a qual adicionou-se suporte para as condições que definem a incorporação de papéis por usuários, conceito extraído do projeto Dragon Slayer [Horst 2001].

A linguagem utilizada para o projeto Dragon Slayer, restrita a controle de acesso, objetiva integrar diferentes hierarquias em modelos de controle de acesso baseados em papéis, não permitindo assim a atribuição direta de privilégios a usuários.

A especificação de LEPS para políticas de irretratabilidade e auditoria teve como inspiração o conceito de atribuir a execução de rotinas quando da realização de determinadas ações do sistema, conceito este encontrado nos projetos REI [Kagal 2003] e PONDER [Damianou 2001] [Lymberopoulos 2003].

O projeto REI consiste em uma linguagem que tem como objetivo expressar segurança em ambientes pervasivos, oferecendo suporte aos serviços de controle de acesso e prevendo a execução de rotinas quando realizadas ações em determinados contextos. Entre suas funcionalidades não se encontra o suporte para resolução de conflitos no processo de derivação de privilégios.

PONDER é um projeto que especifica uma linguagem para representar políticas de segurança em organizações de grandes proporções, prevendo representação de direitos, obrigações e regras a objetos e ações. PONDER apresenta uma visão diferente da adotada por LEPS, não se preocupando com questões relativas a interoperabilidade entre diferentes modelos de controle de acesso.

De forma geral, LEPS se destaca dos trabalhos estudados por duas razões principais. Sua especificação é a única a oferecer suporte integrado aos serviços de autenticação, integridade, privacidade, controle de acesso, auditoria e irretratabilidade. Outra característica importante da linguagem é o fato de LEPS oferecer suporte a interoperabilidade tecnológica, especificamente para os serviços de autenticação, integridade e privacidade, para os quais utiliza-se de parâmetros de alto nível para definir dinamicamente a tecnologia a ser utilizada e seus atributos específicos.

4. LEPS

Para uma melhor compreensão de LEPS, sua descrição realizada nessa seção é segregada de acordo com os serviços para os quais oferece suporte. Os exemplos a serem mostrados nessa seção foram realizados a partir do modelo de segurança da rede *heterogenea.com* descrito na seção 2.

4.1. Definições e Informações Básicas

Para uma melhor compreensão da linguagem, esta seção apresenta os conceitos de sujeitos, objetos, ações, tipos de conexão, parâmetros do sistema e rotinas.

- **Sujeitos -** são entidades para as quais os parâmetros de autorização, autenticação, integridade e privacidade são atribuídos. Existem três tipos de sujeitos, usuários (u), grupos (g) e papéis(p), caracterizados da seguinte forma:
    1. ***Usuários*** (u) - representam indivíduos conectados ao sistema;
    2. ***Grupos*** (g) - são conjuntos de usuários e ou outros grupos, cujos privilégios são estendíveis a seus membros. Sua estrutura hierárquica, demonstrada na figura 2, pode representar cargos, departamentos e funções dentro de uma

corporação. Propõem-se dois modelos de grupos: grupos para serviço de controle de acesso, onde sujeitos podem pertencer a mais de um grupo; grupos para serviços de autenticação, privacidade e integridade, onde sujeitos podem pertencer somente a um grupo;

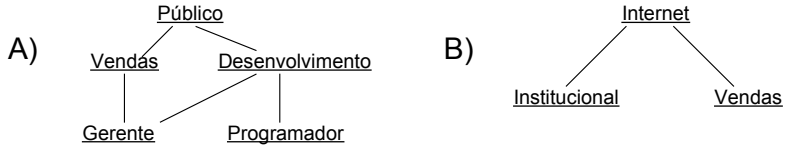

FIG. 2 Modelo hierárquico de Grupos: (A) Modelo para controle de acesso; (B) Modelo para privacidade, integridade e autenticação.

3. ***Papéis*** (p) - são coleções de privilégios e parâmetros que podem ser incorporados por usuários. Sua estrutura hierárquica (figura 3) pode representar funções e cargos dentro de uma corporação, onde sujeitos incorporam e ou desativam papéis de acordo com decisões providas pela política de segurança do sistema.

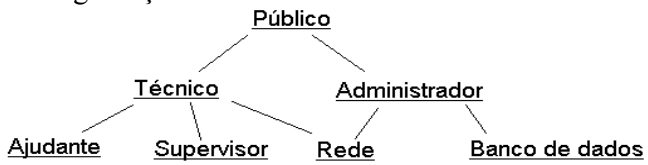

FIG. 3 Modelo hierárquico de papéis

Como exposto acima, grupos e papéis possuem uma estrutura hierárquica e têm como objetivo agregar atributos de segurança para serem transmitidos aos sujeitos relacionados. Usuários herdam invariavelmente as prerrogativas de segurança do grupo ao qual pertence, enquanto que privilégios de segurança de papéis são delegados a usuários em determinados contextos.

- **Ações** - são exemplos de ações ler, escrever, modificar, apagar e executar.
- **Objetos** - são entidades do sistema sobre as quais ações podem ser executadas. São exemplos de objetos arquivos, diretórios, impressoras, serviços e recursos computacionais. Os objetos podem ser agrupados e classificados por tipos, de acordo com seu conteúdo ou funcionalidade no sistema, por exemplo arquivos com as terminações ".mpeg", ".ps" e ".dat" poderiam ser agrupados respectivamente nos tipos de arquivos de vídeo, documento e banco de dados.
- **Tipos de Conexão -** são utilizados para agrupar conexões de acordo com o nível de segurança exigido. Tal classificação se mostra útil na administração de sistemas distribuídos seguros, permitindo a configuração dos níveis de segurança das conexões de uma forma unificada e detalhada.
- **Parâmetros do Sistema** - são as variáveis encontradas em uma sessão de comunicação, tais como data, endereço IP e *login*. Por meio de operadores lógicos, tais parâmetros podem contextualizar conexões na rede, permitindo assim, definir as condições a serem respeitadas para a validade das regras de segurança.
- **Rotinas** - são procedimentos armazenados que realizam determinadas operações quando acionadas por eventos. Dois tipos de rotinas são considerados**:**

1. ***Irretratabilidade***: são rotinas que têm o objetivo de gerar e armazenar provas irrefutáveis sobre a realização de uma ou mais ações de um determinado usuário.
2. ***Auditoria***: são rotinas que armazenam ou notificam informações relativas a ações do sistema.

4.2.Autenticação

A variedade de métodos de autenticação existente possui diferentes níveis de segurança, vulnerabilidade e custos computacionais, sendo que a utilização de determinados métodos de autenticação pode acarretar em degradação de performance de um sistema, principalmente quando se faz o uso de máquinas com baixo poder computacional.

Dentro de um ambiente corporativo com aplicações com diferentes níveis de segurança, uma política eficiente deve analisar o contexto de estabelecimento de conexão para determinar quais métodos de autenticação devem ser utilizados de forma a não comprometer a performance e a segurança do sistema.

Definição: A política para autenticação tem como objetivo mapear tuplas contendo u (usuários), t (tipo de conexão) e m (método de autenticação) para o conjunto {+a (autorizado), -a (negado)}, com a seguinte representação: f{u,t,m}→{+a, -a}.

LEPS faz uso de cinco predicados para representar políticas de autenticação: *atribui* e *pertence* são utilizados respectivamente para atribuir e verificar a relação de um sujeito como membro de grupo; *permite_m* representa os métodos de autenticação aceitos por sujeitos; *aceito_m* representa as regras para derivação de privilégios de grupos, definindo se, em um determinado contexto de estabelecimento de comunicação, certo método de autenticação pode ou não ser utilizado por um sujeito. Segue a definição de tais predicados.

Definição: *atribui(s, g)* - representa a ligação de um sujeito(s) como membro do grupo(g).

Exemplos: seguem duas expressões da rede *heterogenea.com* que associam os usuários Pedro e Luiz como membros do grupo programadores: 1. *atribui(*Pedro, programadores*);* 2. *atribui(*Pedro, programadores*);*

Definição: *pertence(s, g)* → {verdadeiro, falso} - verificar a relação de um sujeito s como membro do grupo g.

Exemplos: as duas expressões a seguir verificam se Maria e Pedro são membros do grupo programadores: 1. *pertence (*Maria, programadores*)* → {falso}*;* 2. *pertence (*Pedro , programadores*)* → {verdadeiro}*;*

Definição: *permite_m(s, m, t)* ← {L} - Representa uma regra L para utilização de um método de autenticação (m) por um sujeito (s) em um tipo de conexão (t).

Exemplos: Tomemos como exemplo as seguintes expressões contidas nos servidores da rede *heterogênea.com*:

1. *permite_m*(Pedro, Kerberos, segura) ← (Host <> wireless & Host <> Internet);
2. *permite_m* (público, Kerberos, fraca) ← (verdadeiro);
3. *permite_m* (público, login/senha, segura) ← (falso).

A primeira expressão representa a possibilidade do usuário "Pedro" poder se autenticar utilizando o protocolo "Kerberos" em conexões do tipo "segura", somente se não estiver utilizando acesso via Internet ou wireless. O segundo exemplo representa uma autorização para o grupo "público" utilizar o "Kerberos" em conexões do tipo "fraca". O último exemplo representa a impossibilidade de utilizar "*login*/senha" para o grupo público em conexões "seguras".

Definição: *aceito_m*(u, m, t) ← {L} → {+a, -a} - representa a regra L de derivação de privilégios, onde em um contexto de conexão retorna aceito (+a) ou não aceito (-a) o método de autenticação (m) para ser utilizado pelo usuário (u) em conexões do tipo (t).

O modelo hierárquico para grupos prevê privilégios transferidos de um sujeito hierarquicamente acima para os sujeitos hierarquicamente abaixo. Assim, além dos

privilégios diretamente associados a si, um sujeito pode derivar privilégios do grupo ao qual pertence. Como um método de autenticação não pode ter seu uso autorizado e negado simultaneamente ao mesmo sujeito, o processo de derivação deve impedir que sujeitos possuam duas regras para um determinado método de autenticação. O predicado *aceito_m* é utilizado para impedir tais conflitos e comandar a transferência de privilégios de autenticação. A tabela 1 representa os dois modelos de derivação, no primeiro, em caso de conflito, as regras do sujeito prevalecem sobre as do grupo, já no segundo o inverso ocorre.

Tabela 1 Exemplo de LEPS para representar resoluções de conflitos

Modelo	Expressão
Sobreposição do sujeito	*Aceito_m*(s, m t) ← (*permite_m*(g, m, t) & *pertence*(s, g) & (∅ = *permite_m*(s, m, t)) U (*permite_m*(s, m, t))
Sobreposição do grupo	*Aceito_m*(s, m, t) ←(*permite_m*(s, m, t) & (∅ = (*permite_m*(g, m, t) & *pertence*(s, g))) U (*permite_m*(g, m, t) & *pertence*(s, g))

Na expressão *aceito_m*(s, m, t) de sobreposição do sujeito, os privilégios de um grupo (g) "*permite_m*(g, m, t)" são derivados somente a seus membros "& *pertence*(s, g, t)" caso não haja privilégios diretamente relacionados a s "& (∅ = *permite_m*(s, m, t)", já os privilégios do sujeito, quando existentes, sempre prevalecem "U *permite_m*(s, m, t)".

Na expressão *aceito_m*(s, m, t) de sobreposição do grupo (s), quando existentes os privilégios do seu grupo, os privilégios do usuários são descartados "(*permite_m*(s,m, t) & (∅ = (*permite_m*(g, m, t) & pertence(s,g))".

Na rede *heterogenea.com* os exemplos descritos acima são armazenados no servidor de credenciais e, quando um usuário (u) entra na rede, o servidor de credenciais por meio da aplicação da regra "*aceito_m"* define e escreve na credencial de u quais os métodos de autenticação aceitos para cada tipo de conexão (t) do sistema.

4.3.Privacidade e Integridade

Os serviços de privacidade e integridade são responsáveis por garantir que, dentro de um contexto de comunicação, somente as entidades participantes e autorizadas poderão acessar, modificar e redigir mensagens, o que inclui a proteção à integridade da mensagem.

O serviço de integridade é implementado por funções espalhamento, já o serviço de privacidade principalmente por algoritimos de criptografia simétrica. Entre as diversas funções do tipo espalhamento, destacam-se MD4, MD5 e SHA-1 [Stallings 98]; de criptografia simétrica 3DES, RC4, RC5 e IDEA [Stallings 98]. Duas grandes questões permeiam o oferecimento de privacidade e integridade em aplicações distribuídas:

- garantir que as aplicações suportem atualizações de novos algoritmos de forma dinâmica, sem perder a compatibilidade com os utilizados anteriormente;
- possibilitar o uso de algoritmos computacionalmente mais leves diante de máquinas com baixo poder computacional.

Os dois exemplos a seguir ilustram as questões mencionadas:

Uma determinada aplicação utiliza-se do 3-DES, um algoritmo de criptografia simétrica. Uma atualização na aplicação para passar a utilizar o RC-5 com chave de 128 bits mais eficiente e robusto [Guelfi 2002], pode tornar-se uma tarefa exaustiva, visto que além de reescrever a aplicação, o desenvolvedor deve impedir que a aplicação se torne incompatível com parceiros de comunicação que ainda utilizam 3-DES.

Outra aplicação utiliza-se do RC-5 com chave de 128 bits para manter privacidade, porém, tal algoritmo se mostra computacionalmente custoso para clientes com máquinas

de baixo poder computacional; uma negociação poderia permitir a utilização de algoritmos computacionalmente mais leves como o RC4 com chave de 40 bits.

A LEPS tem também como objetivo permitir a representação de regras e parâmetros para políticas de privacidade e integridade de forma que a escolha dos mecanismos utilizados por uma conexão seja realizada dinamicamente.

Definição: A política para privacidade e integridade mapeia tuplas contendo u (usuário requisitante), t (tipo de conexão), c (algoritmo de criptografia), tam (tamanho de chave) e f (função espalhamento) para o conjunto {+a (autorizado), -a (negado)}, com a seguinte representação: f(u, t, c, tam, f) →{+a, -a}.

A fim de melhor compreender os recursos de LEPS para políticas de privacidade e integridade, será introduzido dois novos predicados: *permite_c* representa a regra que permite um sujeito fazer uso de um algoritmo de criptografia e uma funçâo espalhamento em conexões de determinado tipo; *aceito_c* representa as regras para derivação de privilégios de grupos, definindo para um determinado contexto de comunicação, qual algoritmo de criptografia e função de espalhamento podem ou não ser utilizados por um usuário. Segue a definição de tais predicados.

Definição: *permite_c(*s, t, c, tam, f*)* ← {L}: Representa a regra (L) para o uso de um algoritmo de criptografia e uma função de espalhamento por um sujeito em um determinado tipo de conexão. Recebe como parâmetros uma entidade do sistema (s), um tipo de conexão (t), um algoritmo de criptografia (c), tamanho da chave criptográfica (tam) e a função de espalhamento (f).

Exemplos: Segue as expressões contidas nos servidores da rede *heterogena.com*, que oferecem tratamento especial a conexões vindas da Internet ou redes WireLess :

1. *Permite_c(*público, segura, RC5, 128, MD5*)* ← (verdadeiro) ;
2. *Permite_c(*público, segura, RC4, 40, f*)* ← (Host <> wireless & Host<>Internet);
3. *Permite_c(*público, fraca, RC4, 40, MD5*)* ← (Host <> Internet) .

A primeira expressão representa a possibilidade do grupo público utilizar o algoritmo RC5 128 bits e a função MD5 para o tipo de conexão segura; a segunda expressão abre a possibilidade do grupo público não conectados via wireless ou Internet utilizarem RC4 40 bits e MD5 em conexões seguras; a última expressão define RC4 40 bits e MD5 utilizáveis para o grupo público em conexões do tipo fraca que não tiverem acesso a Internet.

Definição: *aceito_c(*u, t, c, tam , f*)* ←{L} → {+a, -a} - representa a regra L de derivação de privilégios, onde em um contexto de estabelecimento de conexão, retorna aceito (+a) ou não aceito (-a) dos mecanismos de integridade e privacidade c, tam e f para serem utilizados pelo usuário (u) em um tipo de conexão (t).

Tabela 2 Exemplo de LEPS para representar resoluções de conflitos

Modelo	Expressão
Sobreposição do sujeito	*aceito_c*(s, t, c, tam, f) ← (*permite_m*(g, t, c, tam, f) & *pertence*(s, g) & (∅=*permite_m*(s, t, c, tam, f)) U (*permite_m*(s, t, c, tam, f))
Sobreposição do grupo	*aceito_m*(s,m) ←(*permite_m*(s,m) & (∅ = (*permite_m*(g,m) & *pertence*(s,g))) U (*permite_m*(g,m) & *pertence*(s,g))

Como já mencionado, as regras de derivação têm como objetivo impedir o surgimento de conflitos, onde um usuário possuiria mais de uma regra para utilização de um algoritmo de criptografia em um dado tipo de conexão. A tabela 2 representa os dois modelos de derivação, no primeiro, em caso de conflito, as regras do sujeito prevalecem sobre as do grupo, no segundo as regras do grupo prevalecem sobre as do sujeito.

4.4.Controle de Acesso

O serviço de controle de acesso é responsável por permitir ou negar a execução de determinadas ações do sistema por uma entidade. Sua funcionalidade é baseada em modelos de políticas, onde se definem privilégios e permissões para que usuários possam acessar arquivos, diretórios, serviços e outros. Entre a diversidade de modelos de controle de acesso existentes, destacam-se os baseados em papéis RBAC – *Role Based Access Control* [Sandhu 94] e os baseados na descrição de direitos para entidades DAC - *Discretionary Access Control* [Sandhu 94].

Para que se possa oferecer um controle de acesso integrado em ambientes heterogêneos, é necessário proporcionar uma interoperabilidade entre os diferentes modelos de políticas de controle de acesso existentes [CORBASEC 2001].

A funcionalidade da linguagem LEPS tem como objetivo oferecer suporte à integração dos diversos modelos de controle de acesso existentes, permitindo a representação de suas regras e parâmetros. Sua realização teve como base o LEA [Jajodia 97] a qual adicionou-se a capacidade de representar as regras para usuários ativarem papéis. LEPS se destaca também por trabalhar de forma integrada com outros serviços de segurança, permitindo assim condicionar privilégios de controle de acesso a parâmetros dos mecanismos de autenticação, privacidade e integridade utilizados.

Definição: A política de controle de acesso mapeia tuplas o (objeto), u (usuário) e a (ação) para o conjunto {+a (autorizado), -a (negado)}, como a seguinte representação: f{u,s,a}→{+a,-a}.

Serão adicionados seis novos predicados: *ative*, representa uma regra de ligação entre um papel e um sujeito; *ativo*, verifica se um sujeito pertence a um grupo ou está ativo em um papel; *pode,* representa uma autorização explícita atribuída a um sujeito para realizar uma determinada ação em um objeto, *derive_pode,* representa uma regra para derivação de autorização (*pode*); *faça*, representa a regra para a resolução de autorização de sujeitos, em caso de conflitos provenientes da derivação de autorização, *permite*, representa a regra responsável em autorizar ou negar requisições de uma ação.

Definição: *ative* - representa uma regra (L) para a ativação de papéis (p) por usuários (u), possuindo a seguinte sintaxe: *ative* (u,p) ← {L}.

Exemplos: As seguintes expressões são tomadas como exemplos:

1. *ative*(Pedro,Gerente) ←(Host <> Internet & m_autenticação = X.509);
2. *ative*(João, Técnico) ← (tempo < 20:00)

No primeiro exemplo “Pedro” ativa o papel de “Gerente” somente quando não obtém acesso da rede via Internet e utiliza o método “x.509” de autenticação No segundo exemplo “João” ativa o papel de “Técnico” em horários inferiores à 20:00.

Definição: *pode* **-** representa uma regra para autorização explícita para que sujeitos “s” possam realizar ações “a” sobre objetos “o” e possui a seguinte sintaxe: *pode*(o, s, <sinal>a) ← {L}., onde as expressão L representa as condições a serem respeitadas para a validade da regra.

Exemplos: as seguintes expressões são tomadas como exemplos:

1. *pode*(arquivo_1, Pedro, +ler) ←{t_conexão = segura);
2. *pode*(arquivo_2, secretaria, -escrever).

A primeira expressão representa uma permissão para o usuário “Pedro” “ler” o objeto “arquivo_1” somente quando estiver utilizando conexões do tipo “segura”. A segunda expressão representa uma negativa de “escrever” no “arquivo_2” para o grupo “secretaria”.

Definição: *derive_pode* **-** representa uma regra L para derivação de parâmetros de autorização, possuindo a seguinte sintaxe: *derive_pode*(o,s, <sinal>a) ←{L}.

Exemplos: As seguintes expressões são tomadas como exemplos:

1.*derive_pode*(o,s,+a) ←{ *pode*(0, s', +a) & *ativo*(s, s');

2.*derive_pode*(o,s,+ler)←(*pode*(0,s',+ler) & *ativo*(s, s')) & (pode(o, s, +Escrever)=ϕ).

A primeira expressão realiza a derivação de autorizações positivas "+a" sobre objetos "o" de grupos e papéis "s`" para todos seus membros "s". De acordo com a segunda expressão uma derivação positiva de "ler" só pode ocorrer se o sujeito não possui uma permissão explícita positiva para "escrever".

Definição: *faça* **-** representa uma regra para resolução de autorização para sujeitos em caso de conflitos provenientes da derivação de privilégios, possuindo a seguinte sintaxe: *faça*(o,s, <sinal>a) ←{L}, onde a expressão L representa a regra para resolução de autorização

Exemplos: As seguintes expressões são tomadas como exemplos:

1. *faça*(o,s, +a) ←((*derive_pode(*o,s, +a) & (pode(o,s, -a)= ϕ)∪(pode(o,s, +a);
2. *faça*(o,s, -a) ←((*derive_pode(*o,s, -a) ∪ (pode(o,s, -a)).

No primeiro exemplo uma autorização positiva de uma ação "+a" é relacionada a um sujeito, caso possua uma positiva direta *pode* ou possua uma derivação positiva e não possua uma negativa direta *pode*. No segundo exemplo uma negativa é relacionada ao sujeito se ele possui uma negativa direta ou derivada.

Definição : *permite* **-** representa uma regra L para negar ou autorizar requisições de uma ação "a" por um usuário "u" em objetos "o", possuindo a seguinte sintaxe: *permitida* (o,u, +a) ←{L}.

Exemplos: As seguintes expressões são tomadas como exemplos:

1. *permite* (o,u, +a) ←*faça*(o,u, +a);
2. *permite* (o,u, +a) ←(*faça*(o,u, -a) = ϕ).

No primeiro exemplo, uma autorização é fornecida somente quando existe uma expressão positiva *faça* para sua realização. No segundo exemplo, uma autorização é fornecida somente se não possui uma expressão *faça* negativa para sua realização.

4.4.1 Regras de derivação e resolução de conflitos

As regras e modelos de derivação de previlegios já descritos evitam conflitos, onde um sujeito passa a ter regras contraditórias, porém, como o modelo hierárquico de controle de acesso, ao contrario dos anteriormente estudados, permite com que sujeitos pertencentes a um ou mais supersujeitos (grupos ou papeis), essa seção descreve com maior detalhes o processo de derivação e resolução de conflitos. Tomemos como exemplos os seguintes conflitos:

- um usuário que possui uma autorização direta para de ler um arquivo e é membro de um grupo que possui uma negativa para ler o mesmo arquivo. Em uma derivação direta de privilégio tal usuário possuiria duas permissões contrárias;
- um usuário é membro de dois grupos, onde um deles possui uma autorização positiva para uma ação sobre um arquivo, já o outro possui uma autorização negativa para mesma ação. Nesse exemplo a derivação direta geraria conflito.

O processo de delegação de privilégios tem o objetivo de impedir ambigüidade de privilégios, resolvendo conflitos de propagação, ponderando sobre dois aspectos da política de autorização, descritos a seguir:

- derivação: regula o processo de transferência de autorização ao longo da hierarquia;
- resolução de conflito: determina qual autorização deve prevalecer em caso de conflitos.

A baixo serão discutidas as diferentes escolhas a serem feitas sobre tais aspectos.

Derivação de privilégios: a política de derivação de privilégios define regras para a derivação de autorização ao longo de uma hierarquia de grupos e papéis até as entidades finais. Três diferentes modelos de propagação são mostrados a seguir:

- sem sobreposição - nesse modelo todos privilégios são propagados deixando a cargo de *faça* resolver os problemas de conflito;
- sobreposição do subsujeito – a autorização especificada para o sujeito hierarquicamente abaixo é adotada em caso de conflito;
- sobreposição do super sujeito - a autorização especificada para o sujeito hierarquicamente acima é adotada em caso de conflito.

Utilizando a linguagem LEPS, a tabela 3 demonstra a representação dos modelos de propagação e resolução de conflito acima descrito.

Tabela 3 Exemplo de LEPS para representar derivação de privilégios.

Modelo	**Expressão**
Sem sobreposição	*Derive_pode* (o,s,a) ← *pode* (o,s,a)& *ativo*(s,s')
Sobreposição do subsujeito	*Derive_pode* (o,s,a) ← *pode* (o,s,a) & *ativo*(s,s') & (∅=(*pode* (o,s,-a))
Sobreposição do super sujeito	*Derive_pode* (o,s,a)←*pode* (o,s,a) & *ativo*(s,s') & (∅=(*pode* o,s'',-a) & *ativo*(s',s")&s'<>s'')

Resolução de conflitos: a resolução de conflitos tem como objetivo impedir que uma decisão de acesso do sistema possa ser simultaneamente autorizada e negada. Os seguintes modelos de resoluções de conflitos são propostos:

1. a recusa tem preferência: em caso de conflito a decisão é negada;
2. o acesso tem preferência: em caso de conflito o acesso é permitido;
3. nada é sobreposto: em caso de conflito o sistema age como se nada fosse especificado para a permissão requerida.

Utilizando LEPS, a tabela 4 demonstra a representação dos modelos de resolução de conflitos acima descrito.

Tabela 4 Exemplo de LEPS para representar resoluções de conflitos

Modelo	**Expressão**
Recusa tem preferência	*faça*(o,s,+a) ← *derive_pode* (o,s,+a)& (∅=*derive_pode* (o,s,-a)) *faça* (o,s,-a)) ← *derive_pode* (o,s,-a)
O acesso tem preferência	*faça* (o,s,+a) ← *derive_pode* (o,s,+a) *faça* (o,s,-a)) ← *derive_pode* (o,s,-a) & (∅=*derive_pode* (o,s,+a))
Nada é sobreposto	*faça* (o,s,+a) ← *derive_pode* (o,s,+a)& (∅=*derive_pode* (o,s,-a)) *faça* (o,s,-a)) ← *derive_pode* (o,s,-a) & (∅=*derive_pode* (o,s,+a))

Os exemplos descritos nessa seção remetem a configuração de segurança da rede *heterogenea.com*, onde as regras *pode* são armazenadas junto com as entidades responsáveis pelo recurso (um servidor de arquivo possui todas as regras *pode* que

envolvam seus arquivos) e as regras *ative* e *atribui* são armazenadas no Servidor de Credenciais e distribuídas via credenciais dos usuários. As regras para resolução de conflitos *Derive_pode e faça* são únicas em todo o sistema, sendo replicadas para permitir que todas as entidades da rede possam realizar as tarefas de controle de acesso de forma segura.

4.5.Auditoria e Irretratabilidade

O serviço de Auditoria é responsável em gerar notificação ou armazenamento de informações sobre a realização de ações no sistema. A irretratabilidade tem o objetivo de fazer os usuários responsáveis por seus atos, por meio do armazenamento de provas irrefutáveis de determinadas ações.

Em comum duas questões permeiam as políticas para irretratabilidade e auditoria em aplicações distribuídas:

- permitir especificar dinamicamente a combinação entre ações e contextos que necessitam dos serviços de auditoria e ou irretratabilidade;
- identificada uma ação que necessite de auditoria ou irretratabilidade, possibilitar que os parâmetros contidos nas mensagens ou provas envolvidas possam ser estabelecidos dinamicamente, assim como identificar qual mecanismo de auditoria ou irretratabilidade deve ser utilizado.

Definição: A linguagem lógica para irretratabilidade e auditoria tem como objetivo mapear as ações (a) do sistema no conjunto de {sim, não} para a necessidade de realização de rotinas. Em caso positivo, fornecer uma tupla contendo a rotina a ser realizada e os parâmetros do sistema que devem ser passados à mesma, conforme a seguinte expressão: $f(a) \rightarrow \{ (sim/não), (R, P_1, P_2, P_3 ... P_n)\}$.

LEPS faz uso do predicado *Executa* para representar políticas de autenticação:

Definição *: executa*(a, R) $\leftarrow\{L\}\leftarrow\{P_1, P_2, P_3, \ldots, P_n\}$, representa a necessidade de execução de uma rotina, quando realizada determinada ação, sendo a expressão L condição a ser obedecida para validade de *executa*. Os parâmetros do sistema Pn correspondem às variáveis a serem passadas para a rotina.

Como exemplo tem-se, uma aplicação de vídeo sob demanda na rede *heterogenea.com* que necessita do serviço de irretratabilidade para pedidos de execução (exe_film) de filmes, onde os parâmetros de data do pedido e nome do filme devem estar contidos nas provas de irretratabilidade:

- *executa*(exe_film, Irretratabilidade) $\leftarrow\{\}\leftarrow$ {data, nome do filme}

5. Conclusões e Trabalhos Futuros

Ambientes pervasivos - devido à utilização de dispositivos com capacidades distintas e o oferecimento de serviços com diferentes necessidades de segurança - carecem de sistemas mais flexíveis que permitam a definição dinâmica dos mecanismos de segurança.

A linguagem LEPS apresentada nesse trabalho tem como objetivo fornecer justamente a flexibilidade de segurança necessária aos sistemas pervasivos, permitindo a integração dinâmica de mecanismos e a representação formal de regras e parâmetros para diferentes modelos de políticas, abrangendo os serviços de controle de acesso, privacidade, integridade, autenticação, auditoria e irretratabilidade.

A utilização de uma rede teste *heterogenea.com* apresentou de maneira prática, como a utilização da LEPS pode fornecer um sistema de segurança mais flexível e adaptativo a

uma aplicação real, provendo a robustez necessária sem comprometer o desempenho do sistema.

Em comparação com as outras linguagens estudadas, verificou-se que, por tratar políticas dos serviços de autenticação, integridade, privacidade, controle de acesso, auditoria e irretratabilidade de forma unificada, LEPS permite políticas de segurança integradas, onde as regras de oferecimento de um serviço podem depender explicitamente de parâmetros de outro.

Futuros trabalhos poderão aperfeiçoar a LEPS oferecendo APIs e ferramentas administrativas com interface gráfica para a sua utilização.

Referências

Corbasec - Corba Security Specification, Mar 2001. disponível em: www.corba.com.br

Damianou, N.; Dulay N, Lupu The Ponder Specification Language Workshop on Policies for Distributed Systems and Networks, HP Labs Bristol, 2001.

Horst F. Wedde, M., Modular Authorization, ACM Symposiumon Access Control Modelsand Technologies(SACMAT2001), pages97–105.ACMPress,2001.

Guelfi, A.; Meylan, F. Middleware to Distributed Multimedia Applications on High Speed Multi-Service Networks", Proceedings of the International Conference on Software Engineering (CSITeA 2002), Foz do Iguazu, Jun 2002.

Jajodia, S;. Samarati, P. A Logical Language for Expressing Authorizations_1997. IEEE Symposium on Security and Privacy (S&P'97), 1997.

Kagal, L.; Finin, T. Joshi, A.. A Policy Language for a Pervasive Computing Environment, 2002 IEEE 4th International Workshop on Policies for Distributed Systems and Networks, Lake Como, June 4-6 2003.

Lymberopoulos, L.; Lupu, E. ; Sloman M. Using CIM to Realize Policy Validation within the Ponder Framework. DMTF Global Management Conference, San-Jose, California, June 2003.

Mattes, L., Meylan, F., Guelfi, A., Zuffo J. A Performance Analysis of Security Manager for Communication Middlewares. The 2003 International Conference on Security and Management (SAM'03), Las Vegas 2003.

Peikari, C., Maximum Wireless Security. SAMS, 2003.

Sandhu, R.; Samarati. Access Control: Principles and Practice, 1994.

Stallings, W. Cryptography and Network Security: Principles and Practice. 2 ed. Prentice Hall, 1998.

Steve, L., Carlisle, A. Understanding PKI: Concepts, Standards, And Deployment Considerations, 2002.

Undercoffer, J.et all A Secure Infrastructure for Service Discovery and Access in Pervasive Computing. ACM Monet: Special Issue on Security in Mobile Computing Environments, October 2003.