Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Controle de Acesso Baseado em Papéis na Informatização de Processos Judiciais

André Thiago Souza da Silva
Hugo Vasconcelos Saldanha

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Pedro A. D. Rezende

Brasília
2006

Universidade de Brasília – UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Curso de Bacharelado em Ciência da Computação

Coordenador: Profª. Cláudia Nalon

Banca examinadora composta por:

Prof. Pedro A. D. Rezende (Orientador) – CIC/UnB
Prof. Luiz Antônio da Frota Mattos – CIC/UnB
Sr. Paulo Roberto da Silva Pinto – Secretário de TI/STF

CIP – Catalogação Internacional na Publicação

André Thiago Souza da Silva.

Controle de Acesso Baseado em Papéis na Informatização de Processos Judiciais/ Hugo Vasconcelos Saldanha, André Thiago Souza da Silva. Brasília : UnB, 2006.
66 p. : il. ; 29,5 cm.

Monografia (Graduação) – Universidade de Brasília, Brasília, 2006.

1. controle de acesso, 2. RBAC, 3. certificado, 4. LDAP, 5. processo, 6. segurança

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro – Asa Norte
CEP 70910–900
Brasília – DF – Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Controle de Acesso Baseado em Papéis na Informatização de Processos Judiciais

André Thiago Souza da Silva
Hugo Vasconcelos Saldanha

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Pedro A. D. Rezende (Orientador)
CIC/UnB

Prof. Luiz Antônio da Frota Mattos
CIC/UnB

Sr. Paulo Roberto da Silva Pinto
Secretário de TI/STF

Prof.ª Cláudia Nalon
Coordenador do Bacharelado em Ciência da Computação

Brasília, 11 de agosto de 2006

Agradecimentos

Ao professor Pedro Rezende, por suas valiosas contribuições, sugestões e orientações para a realização desse trabalho.

Ao Renato Bigliazzi, por sua inestimável colaboração na elicitação dos requisitos.

Ao professor Luis Antônio da Frota Mattos, por ter se mostrado sempre prestativo, até mesmo em situações difíceis.

Aos meus pais, José Eusébio e Bernadete, pela educação e incentivo que me proporcionaram.

A Ana Cláudia, pelo amor e compreensão que demonstrou durante os meus momentos ausentes.

Aos meus pais, Saldanha e Fátima, que me ensinaram o valor do estudo, do trabalho e da honestidade, e por sempre me darem o suporte necessário.

Aos meus irmãos, Igor e Vitor, que conviveram comigo durante esse trabalho.

A Elisa, pela paciência e compreensão durante os momentos difíceis.

Resumo

O acesso à justiça por meio eletrônico tem aumentado bastante nos últimos anos. Além de permitir um maior acesso, a informatização possibilita maior celeridade no andamento da mesma. Com a disponibilização das informações pertencentes a processos por meio eletrônico, é necessário implementar um controle de acesso que reflita a legislação vigente correspondente. Este trabalho modela o controle de acesso a processos digitais de acordo com o modelo RBAC, utilizando a ferramenta livre MACA, com uma modificação para prover autenticação via certificados digitais.

O modelo RBAC foi escolhido por proporcionar maior facilidade na administração das permissões a recursos em sistemas de grande porte, além de refletir, na sua implementação, os papéis existentes na organização, facilitando seu entendimento. A autenticação via certificados digitais é proposta por utilizar criptografia forte, além de haver uma tendência pela adoção, por parte de órgãos públicos, do uso de certificados digitais, graças à instituição da Infraestrutura de Chaves Públicas Brasileira (ICP-Brasil).

Palavras-chave: controle de acesso, RBAC, certificado, LDAP, processo, segurança

Abstract

Access to justice by eletronic means has increased greatly in the past few years. Besides allowing a greater access, it makes possible a greater agility in its course. With information from legal proceedings more accessible, it's important to implement an access control that implements the existing legislation concerning the legal proceedings. The present work models an access control to eletronic legal proceedings according to RBAC proposed stardard, making use of MACA, with modifications to provide authentication with digital certificates.

RBAC was chosen because it makes easy to administrate permissions to resources in large scale systems. It also reflects the existings roles in the organization, making its understanding easier. Authentication with digital certificates is proposed because it makes use of strong cryptography, besides the existence of a tendency for the use of digital certificates by the public organizations, thanks to the creation of Infraestrutura de Chaves Públicas Brasileiras (ICP-Brasil).

Keywords: access control, RBAC, certificate, LDAP, legal proceedings, security

Sumário

Lista de Figuras

Capítulo 1

Introdução

O acesso à justiça por meio eletrônico tem aumentado bastante nos últimos anos. O uso de meio digitais no Judiciário, além de possibilitar mais um meio de acesso à justiça, provê uma maior celeridade na prestação jurisdicional. Não faltam iniciativas, que se espalham por todo o país. As iniciativas locais que vem sendo tomadas por tribunais brasileiros, individual e separadamente, têm o intuito de agregar uma série de valores à sua prestação jurisdicional. Entre estes valores, estão a celeridade, a publicidade, a operosidade, a praticidade e a universalidade. Há também uma iniciativa de âmbito nacional para padrozinar a forma como essa informatização é disponibilizada, principalmente com relação ao sigilo e à autenticidade dos dados produzidos nessas aplicações.

Neste contexto, vários tribunais já disponibilizam seus sites na Internet com informações gerais sobre o andamento de processos e sentenças. É possível também cadastrar-se em algum desses sites e ser notificado automaticamente sempre que um processo de interesse seja movimentado.

Entretanto, essas iniciativas só contemplam funcionalidades de informação e documentação. A digitalização de processos ainda é um caminho a ser percorrido, porém já existem muitos esforços neste sentido:

- O *Supremo Tribunal Federal* – STF implementou o projeto *e*-STF que permitiu a petição por meio eletrônico, fax ou similar; e o Projeto de Inteligação Informatizada do Poder Judiciário – o *Infojus*, que oferece uma infra-estrutura em comum de redes de comunicação para os órgãos do Poder Judiciário. Além disso, sua página na internet se coloca como um dos dez melhores sistemas do País.

- O *Superior Tribunal de Justiça* – STJ desenvolve estudos na tentativa de eliminar o papel e utilizar o sistema *on-line* até mesmo para petições.

- O Projeto de Lei n. 5.828/2001, aprovado pela Câmara dos Deputados, dispõe sobre a informatização do processo judicial, admitindo o uso de meio eletrônico na comunicação de atos e a transmissão de peças processuais.

Todo esse cenário de informatização citado deve respeitar, além das boas práticas de segurança, a jurisprudência existente. Ou seja, a "justiça informatizada" deve ser um espelho, na medida do possível, da justiça do "mundo da vida". No andamento de um processo várias regras processuais devem ser respeitadas: há todo um caminho por onde o processo deve passar, prazos a serem respeitados, além de um conjunto determinado de atores jurídicos responsáveis pelo andamento do processo, alguns habilitados a decidir, sob determinadas condições, sobre os rumos que o mesmo pode tomar.

De forma simplificada, o caminho do processo é formado por um conjunto de estados pelo qual ele passa: Petição Inicial, Deferimento, Contestação, etc., com prazos a serem respeitados. Para fazer o processo caminhar por entre estados, há um conjunto de atores que devem atuar sobre o processo. Além disso, a administração de um processo é extremamente afunilada ao redor da figura do Juiz, sendo que a maioria dos atos levam a sua assinatura. É o rito processual.

A informatização do rito processual deve seguir as mesmas regras. Nesse sentido, surge o problema de como controlar e administrar o acesso dos atores jurídicos aos processos. O controle de acesso é importante não só para a manutenção do sigilo de informações constantes no processo, quando aplicável, mas também para definir, dentre os atores jurídicos que possam vir a cena, quais estão legal ou legitimamente autorizados, em determinado momento, a realizar determinada ação sobre determinado processo.

Esse controle envolve não só aos atores jurídicos que atuam no trâmite do processo, mas também a intermediação de sistemas computacionais que possibilitam o acesso de tais atores ao processo, para cumprimento do rito. Sistemas computacionais que, por sua vez, também têm suas regras e mecanismos de acesso, seus atores (usuários, administradores), etc., esses via de regra representados em várias camadas de codificação.

Assim, o uso de tecnologia de informação em processos judiciais deve ter como requisito básico a definição de um modelo para o controle de acesso a processos que respeite a autonomia e as prerrogativas dos distintos atores em diferentes sistemas de forma que, pelo menos idealmente, as relações e interdependências internas dos sistemas – o jurídico-fim e o tecnológico-meio – não interfiram de forma despropositada, indevida, subreptícia ou descontrolada um no outro.

Surgem, então, questões relativas à concepção de um tal modelo de controle de acesso, que possa satisfazer os requisitos citados. Podemos separar esses requisitos em dois grupos, ou instâncias. A primeira refere-se à automação de decisões sobre autorizar ou não o acesso em determinado modo a determinado processo ou peça processual, e por qual lógica autorizativa. A segunda é referente ao processo de se administrar as políticas de controle de acesso, a partir das quais tais decisões são executadas.

Sobre a lógica de autorização, estratégias e diretrizes conflitantes podem ser aplicadas. Ademais, no domínio de aplicação em tela, ou seja, na informatização processual judiciária, tais conflitos refletem uma tensão natural entre o legal e o legítimo. Por um lado, o acesso ao processo deve ser

restrito às partes interessadas, respeitando os aspectos legais existentes (caminho do processo, restrições de prazo) e definindo quais usuários estão aptos a realizar ações sobre determinados processos. Por outro lado, o controle de acesso não pode prejudicar a prestação do serviço aos interessados, devendo garantir a celeridade da justiça, que talvez seja o principal objetivo da informatização de processos, e que respeita o princípio constitucional da eficiência[1].

Desta forma, impor uma política muito proibitiva, ou "engessada", quer pela aplicação, pelo sistema ou por suas arquiteturas, acaba por prejudicar não só o acesso ao serviço informatizado, mas também, a médio e longo prazos, o bom desempenho de quem responde pela informatização; já uma política demasiadamente permissiva pode violar os aspectos legais envolvidos ou, pior, abrir novas brechas para fraudes processuais, ainda mais insidiosas e potencialente difíceis de erradicar, porque invisíveis ou difíceis de rastrear. Assim, no escopo de um planejamento adequado à informatização do judiciário, deve-se buscar um modelo adequado, onde cada autorização é concedida/proibida no momento exato, de acordo com a necessidade e o direito do usuário, levando em conta a situação (contexto) existente durante o acesso. Levando-se, também, em conta a complexidade ontológica dos ritos processuais, um modelo que leve em conta a escalabilidade no espaço de regras, prerrogativas, atores e ações que compõem a política de segurança (da qual a de controle de acesso faz parte), que possa atender à demanda real sem inviabilizar a administração dessas políticas

Na administração da política de controle de acesso, a questão da racionalidade se torna importante. Normalmente, um processo jurídico informatizado é acessado em um sistema distribuído, constituído de diversas aplicações e diversas bases de dados, nas mais variadas plataformas, onde cada sistema já possui um modelo de controle de acesso. Conceber um modelo que integre tais diversidades, sem inviabilizá-las, é um grande desafio.

Seria possível conceber um mecanismo de administração de políticas de controle unificada, que integre de forma coerente os diferentes sistemas, englobando suas funcionalidades sem inviabilizar sua eficácia no processo? É claro que a resposta depende, em grande parte, das plataformas existentes, de seus regimes de padronização e interfaces de programação e controle. Tanto melhor se possibilitem a interoperabilidade com outros sistemas e portabilidade a outras plataformas, principalmente no que se refere ao controle de acesso (Motta, 2003).

Não obstante, para que esse controle de acesso seja realmente efetivo, a autenticação perante o mecanismo que provê esse serviço unificado deve ser feita de maneira adequada, e de forma a forneçer os meios mais abrangentes possíveis de se aferir sua confiabilidade. Conceitualmente, para atingir esse requisito, o modelo deve fazer uso do mecanismo de autenticação através de criptografia assimétrica, com utilização de certificados digitais, por ser esse o mecansimo que demanda o mínimo possível em relação a premissas de sigilo (Schneier, 1996). A instituição da Infraestrutura de Chaves

[1]Constituição Federal, artigo 37

Públicas Brasileira (ICP-Brasil)[2] veio aproximar ainda mais esse cenário nacionalmente, fazendo com que a criptografia assimétrica, além da autenticação, seja utilizada para assinar documentos eletronicamente, provendo autenticidade, integridade e, quando aplicável, o não repúdio (Schneier, 1996).

O objetivo principal do trabalho é a construção de um modelo de controle de acesso baseado em papéis (RBAC - *Role-Based Access Control*) contextualizado na informatização de processos que contemple os requisitos apresentados, com o intuito de controlar o acesso a processos do Juizado Especial . Como objetivo secundários temos: adaptação da arquitetura do MACA (*Modelo de Autorização Contextual para o Controle de Acesso Baseado em Papéis*) (Motta, 2003) para conter autenticação via certificados digitais; e a implementação de um sistema protótipo que possua os requisitos citados.

A escolha pelo controle de acesso baseado em papéis é justificada pelo conjunto de características que esse modelo possui e que atendem os requisitos necessários para a construção/implementação do modelo desejado.

Primeiramente, o modelo dá suporte ao princípio do privilégio mínimo. Conforme esse princípio, para cada usuário não são atribuídas permissões além daquelas que ele precisa para a realização de sua função ou tarefa. Isso evita o problema de um indivíduo ter a capacidade de realizar funções desnecessárias; ou seja, um usuário só deve possuir as autorizações indispensáveis, mínimas para ele realizar as tarefas ou acessar os dados que ele necessita. Esta característica é fundamental no contexto de acesso aos processos informatizados, onde cada usuário, em um determinado momento, só pode ser autorizado a realizar as funções que se espera que ele realize e nada mais.

Em segundo lugar, está a questão da administração da política de segurança do sistema, conforme citamos anteriormente. O modelo de controle de acesso baseado em papéis permite que a administração da política seja centralizada (Ferraiolo et al., 2003). Uma vez que os papéis estejam definidos, e as transações que esses papéis podem realizar estejam estabelecidas dentro do sistema de acesso, estas transações tendem a permanecer relativamente constantes ao longo do tempo, evoluindo junto com o sistema de aplicações. As atividades administrativas passam a ser, basicamente, a inserção ou a revogação de membros no conjunto de papéis especificados dentro do sistema. Quando um indivíduo entra na organização, o administrador simplesmente o insere em um ou mais papéis existentes. Quando a função de uma pessoa muda dentro da organização, basta alterar os papéis ligados a essa pessoa. Quando uma pessoa deixa a organização, todos os seus usuários que são membros de papéis são removidos. Por fim, como os papéis são utilizados para categorizar autorizações de funções organizacionais espec íficas, quando as autorizações e responsabilidades das funções mudam, basta alterar as autorizações associadas aos respectivos papéis.

Ou seja, o controle de acesso baseado em papéis facilita a administração centralizada por interpor os papéis entre usuários e autorizações, o que

[2]Medida Provisória n. 2.200-2, de 24 de Agosto de 2001

reduz a complexidade e o custo administrativo do controle de acesso, especialmente quando comparado com outros modelos. Isso viabiliza a administra ção de uma política de acesso detalhada, onde há um grande número de usuários e recursos que acessam múltiplos sistemas, situação em que se enquadra o contexto de processos informatizados.

Por fim, a existência de aplicações práticas do controle de acesso baseado em papéis em grandes organizações é um outro fator determinante para a escolha do modelo baseado em papéis. A experiência do Instituto do Coração, em São Paulo, com mais de 2500 usuários fazendo uso, em um ambiente distribuído, do MACA (Motta, 2003), que é um software livre sob licença GPL, mostra que realmente é possível aplicar o modelo baseado em papéis à realidade.

Com a utilização do MACA, busca-se explorar o potencial do modelo de desenvolvimento colaborativo, sob regime de licenciamento livre, para alcançar maior eficiência no atendimento à demanda instrumental de mecanismos de segurança próprios para integração e/ou informatização de sistemas processuais na esfera judiciária, atualmente reprimida, alavancando e provendo assim a autonomia do usuário, em relação a fornecedores de componentes, para controlar sua própria política de segurança, autonomia esta que é a própria essência desta segurança.

Nos próximos capítulos, serão desenvolvidos os diversos conceitos pertinentes à modelagem do controle de acesso de processos judiciais: o conceito de controle de acesso propriamente dito, com a exposição de alguns modelos de uso consagrado e o RBAC; o padrão de RBAC conforme proposto pelo *National Institute of Science and Technology* - NIST dos EUA; uma descrição da arquitetura do MACA e as modificações realizadas; e uma exposição de uma modelagem do acesso a processos judiciais, mostrando a estrutura organizacional e possíveis papéis do negócio, relacionados aos atores jurídicos do domínio de aplicação específico para o qual se implementou um protótipo de sistema, com suas resposabilidades e autorizações.

Capítulo 2

Modelos de Controle de Acesso

2.1 Conceitos e Terminologia

O controle de acesso é a maneira pela qual o acesso (utilização, modificação, leitura, etc.) a determinado recurso é concedido ou proibido de alguma forma (Ferraiolo et al., 2001). O controle de acesso pode não apenas definir quem tem acesso a um determinado recurso, mas também qual tipo de acesso. O controle pode estar embutido dentro do sistema operacional, incorporado a aplicações ou pode ser implementado por meio de pacotes de segurança. Pode, ainda, ser implementado internamente ao sistema computacional a ser protegido ou ser implementado em dispositivos externos ao sistema.

O controle de acesso pode tomar diferentes formas. Além de determinar se um usuário tem permissão para utilizar um recurso, o controle de acesso pode determinar quando e como esse recurso pode ser utilizado. Por exemplo, um usuário pode ter acesso a uma rede apenas durante o expediente da empresa. Um processo executado tem permissão somente para ler, e não para escrever, em um arquivo. Dependendo do nível de segurança exigido por uma organização, um sistema corporativo pode exigir controles mais complexos, como requerer dois uma mais funcionários identificados no sistema ao mesmo tempo para realizar um operação de alto risco.

Sendo um dos vários aspectos pertencentes a uma solução de segurança computacional, o controle de acesso possui propósitos semelhantes aos demais mecanismos de segurança. Todos eles têm como objetivo garantir sigilo, integridade e/ou disponibilidade de recursos. No controle de acesso, o sigilo e/ou integridade da informação são críticos. Para a condição de sigilo é necessário que somente usuários autorizados tenham permissão para ler a informação. Já na condição de integridade, apenas usuários autorizados podem alterar informações de maneira também autorizada. Essas duas condições são bem claras no controle de acesso. Menos óbvia é a condição de disponibilidade, mas tambéum possui um importante papel: um usuário que adquire acesso não autorizado a um sistema tem menos dificuldade para torná-lo indisponível.

Os objetivos do controle de acesso são comumente descritos em termos

de proteção para recursos do sistema contra acesso inapropriado ou indesejado de usuários. De uma perspectiva comercial, esse objetivo também pode ser descrito em termos de otimizar o compartilhamento dos recursos e informações. Afinal, a grande motivação da tecnologia de informação é fazer com que as informações estejam disponíveis para usuários e aplicações de maneira eficiente, além de segura. Quanto maior for a capacidade de compartilhamento da informação, maior a produtividade e utilidade do sistema.

2.1.1 Autenticação e autorização

Autorização e autenticação são dois conceitos fundamentais na área de controle de acesso. São conceitos distintos mas interdependentes, de forma que a autorização a um determinado recurso é, na verdade, dependente da autenticação.

Autenticação é o processo que determina que a identidade mostrada por um usuário é legítima. A autenticação é baseada em um (ou mais de um) dos seguintes fatores:

- Algo que sabemos, como por exemplo, uma senha ou número de identificação pessoal;
- Algo que possuímos, como um cartão inteligente ou uma chave;
- Algo que somos ou uma característica física como uma impressão digital, padrão de retina ou uma característica facial.

Enquanto a autenticação é um processo de determinar quem somos para o sistema, autorização determina o que podemos a fazer. Autorização refere-se a uma decisão do tipo sim ou não que determina se um usuário tem o acesso garantido a um recurso do sistema (Ferraiolo et al., 2003). Um sistema de informação deve manter alguma relação entre identificadores de usuários e recursos do sistema, possivelmente anexando uma lista de usuários autorizados para os recursos ou armazenando uma lista de recursos acessíveis com cada identificador de usuário.

De qualquer forma, deve-se notar que a autorização necessariamente depende da realização da autenticação. Se o sistema não for capaz de ter certeza a respeito da identidade de determinado usuário, não haverá uma maneira correta de determinar se o usuário deve ou não acessar o recurso.

2.1.2 Usuários, sujeitos, objetos, operações e permissões

Praticamente qualquer modelo de controle de acesso pode ser formalmente definido utilizando as noções de usuários, sujeitos, objetos, operações e permissões, e as relações entre estas entidades. Desta forma, é importante a compreensão destes termos.

O termo usuário refere-se à pessoa que interage com o sistema computacional. Em muitos projetos, é possível para um único usuário possuir múltiplos identificadores e estes identificadores podem estar ativos simultaneamente — e são os mecanismos de autenticação que possibilitam a ocorrência dessa situação.

Um sujeito (*subject*) é um processo de computador agindo em benefício de (ou comportando-se como) um usuário. Na realidade, toda e qualquer ação de um usuário em sistema computacional é realizada por meio de algum programa executando em um computador. Um usuário pode ter múltiplos sujeitos em execução, mesmo se o usuário só possui um identificador (*login*). Por exemplo, enquanto um usuário acessa uma página na *Web* utilizando um navegador, um agente de *e-mail* pode estar executando como um *daemon*, consultando um determinado servidor periodicamente, em busca de novas mensagens. Cada um dos programas do usuário é um sujeito, e o acesso realizado por cada um dos programas será checado a fim de se garantir que o usuário que invocou os programas realmente tem acesso a eles.

Um objeto pode ser qualquer recurso acessível em um sistema computacional, o que inclui arquivos, periféricos como impressoras, banco de dados, e até mesmo entidades de granularidade fina como campos individuais em registros de banco de dados. Objetos são geralmente vistos como entidades passivas que contém ou recebem informações, embora seja possível tratar programas, impressoras e outras entidades ativas como objetos.

Uma operação é um processo ativo invocado por um sujeito. As operações podem ter diversos níveis de abstração — incluindo operações mais comuns como escrita, leitura, alteração até operações mais complexas e dependentes da natureza de cada sistema ou domínio de aplicação, como operações de movimentação de um processo judiciário, por exemplo.

Permissões (ou privilégios) são autorizações para realizar alguma ação no sistema, ou seja, refere-se a alguma forma de combinação entre um objeto e uma operação. Uma determinada operação aplicada a dois objetos diferentes representa duas permissões distintas e, de forma similar, duas operações diferentes aplicadas ao mesmo objeto representam duas permissões diferentes.

2.1.3 Princípio do menor privilégio

O princípio do menor privilégio é uma prática administrativa de controle de acesso por meio da qual as permissões são associadas aos usuários de tal forma que a cada usuário são dadas tão somente as permissões necessárias para realizar a função que o projeta como usuário do sistema. O princípio do menor privilégio evita o problema de um indivíduo possuir a capacidade de realizar ações desnecessárias e potencialmente prejudiciais ao sistema, ações que podem surgir como um efeito colateral da associação de permissões desejadas. A grande discussão presente nesta área de estudos é como associar o conjunto de permissões configuráveis no sistema ao agregado de

funções ou responsabilidades que correspondem a um papel de um usuário, ou a um sujeito que age em benefício deste usuário.

O princípio do menor privilégio provê uma maneira racional de onde instalar os limites de separação que são providos pelo mecanismo de controle de acesso.

A aderência estrita ao princípio do menor privilégio requer que um indivíduo possua diferentes níveis de permissões em momentos diferentes, dependendo da tarefa ou função sendo realizada. Deve-se reconhecer que em alguns ambientes e com algumas permissões, restringir permissões porque elas são nominalmente desnecessárias pode ser inconveniente para o usuário ou colocar uma carga adicional sobre os administradores e complexidade no sitema, aumentando a chance da ocorrencia de falhas de segurança de difícil diagnóstico. Entretanto, a permissão de privilégios excessivos que potencialmente podem ser explorados para quebrar a proteção existente, seja por meio da integridade ou da confidencialidade, que possibilite ou até estimule o desvio de função ou abuso de autoridade, sempre que possível deve ser evitada.

2.1.4 Monitor de referência

O monitor de referência é um conceito abstrato, em que as tentativas de acessos que sujeitos realizam em objetos são autorizadas baseados em informações contidas em uma base de dados de controle de acesso (Ferraiolo et al., 2003). O monitor de referência representa a porção de um sistema que é responsável pela garantia da política de segurança do sistema no que tange ao controle de acesso. O banco de dados de controle de acesso é o responsável pela manutenção dessa política, em termos de atributos ligados a sujeitos e a objetos e respectivos direitos de acesso. Quando um sujeito tenta realizar alguma operação em um determinado objeto, o monitor de referência deve realizar algum tipo de verificação, geralmente comparando os atributos do sujeito com aqueles pertencentes ao objeto.

O monitor de referência não estipula uma política específica para o sistema e muito menos uma implementação em particular. O monitor de referência, entretanto, define um *framework* — que tem sido muito utilizado — para o projeto, desenvolvimento e implementação de sistemas de segurança e que tem funcionado como base de avaliação de grau de confiança que pode ser associado a um sistema computacional.

Os requisitos abstratos de um monitor de referência são descritos em três princípios fundamentais de implementação, mostrados a seguir.

1. *Completude*: o monitor de referência deve sempre ser invocado e deve ser impossível de desviá-lo. Este princípio requer que um sujeito só possa referenciar um objeto invocando o monitor de referência.

2. *Isolamento*: o monitor de referência deve ser à prova de falsificação. Este princípio afirma que a função mediador de acesso é intransponível, ou sejam deve ser impossível para um estranho ao sistema atacar

o mecanismo de controle de acesso de forma que afete o desempenho da verificação de autorização.

3. *Verificabilidade*: deve ser possível de demonstrar que foi implementado corretamente. Este princípio pode ser alcançado por meio de critérios de projeto e práticas de engenharia de software (Ferraiolo et al., 2003). A idéia é que o monitor de referência seja tão pequeno e tão simples quanto possível, excluindo qualquer funcionalidade da qual a segurança do sistema não seja dependente e reduzindo-o a um pequeno conjunto de interfaces.

Estes princípios fornecem um guia arquitetural para o projeto e para o processo de desenvolvimento de um sistema de controle de acesso. O grau em que o sistema se relaciona com esses princípios serve como uma medida do nível de confiança e correção dos controles de segurança do sistema.

2.2 Políticas, modelos e mecanismos

2.2.1 Modelo de Bell-LaPadula

O modelo de Bell-LaPadula (Bell and LaPadula, 1973) é uma descrição formal dos caminhos permitidos para o fluxo da informação em um sistema. O objetivo do modelo é identificar as formas de comunicação permitidas, em que é importante a preservação de sigilo. O modelo tem sido utilizado para definir os requisitos de segurança para sistemas que tratam, concorrentemente, dados com diferentes níveis de sensibilidade. Esse modelo serve para formalizar políticas de segurança multinível.

Bell and LaPadula (1973) utilizam máquinas de estados finitos para a formalização de seu modelo. São definidos vários componentes da máquina de estados finitos, que definem, de maneira formal, o que significa um determinado estado ser seguro, e consideram as transições que são permitidas de tal forma que ao deixar um estado seguro, um estado inseguro nunca possa ser alcançado.

No modelo, são incluídos níveis de segurança que refletem sistemas de segurança militar: cada sujeito possui um nível de segurança máximo, e cada objeto tem uma classificação.

1. *Propriedade da Segurança Simples*: nenhum sujeito possui acesso de leitura a qualquer objeto que possua uma classificação maior que o nível de segurança do sujeito; e

2. *A Propriedade Estrela* (*): nenhum sujeito possui acesso de escrita a um objeto cujo nível de segurança for diferente ao nível corrente daquele sujeito; e nenhum sujeito possui direito de leitura em objetos cujo nível de segurança seja maior que o nível de segurança corrente daquele sujeito.

No modelo militar, a propriedade da segurança simples afirma que um usuário com um determinado nível de segurança não possui a permissão de leitura de informação cujo nível esteja acima do seu; por exemplo, um usuário com nível de segurança secreto não pode ler documentos classificados como ultra-secretos. Da mesma forma, a propriedade estrela afirma um usuário operando em determinado nível de segurança só pode escrever informação naquele nível ou acima; por exemplo, se um usuário for identificado como do nível secreto, programas e processos operados por aquele usuário não são permitidos escrever informações em níveis de classificação mais baixos, como confidencial, mas podem escrever em níveis mais altos, como ultra-secreto.

2.2.2 Modelo Clark-Wilson

Clark and Wilson (1987) documentaram uma visão generalizada da política de segurança comercial, demonstrando o quanto elas diferem das políticas de segurança militares. Eles propuseram dois princípios como os mais importantes na busca de garantia para manutenção da integridade: transações bem formadas e separação de responsabilidades.

Transações bem formadas limitam a forma que os usuários podem modificar os dados, assegurando que todos os dados que iniciem em um estado válido permanecerão válidos após a execução da transação. A unidade básica de controle de acesso do modelo é uma tripla, composta por usuário, procedimento de transformação e um item de dados confinado. Um procedimento de transformação é uma transação e um item de dados confinado é aquele cuja integridade deve ser preservada.

O princípio da separação de responsabilidades assegura a consistência de mudanças realizadas em dados críticos, o que previne que usuários autorizados realizem modificações impróprias, contribuindo, assim, para a integridade dos dados. A separação de responsabilidades é aplicada por meio da divisão de operações em diversas suboperações e estabelecendo que diferentes pessoas realizem cada uma dessas suboperações. Por exemplo, num processo de compra de uma determinada mercadoria envolvendo os passos de autorização do pedido de compra e autorização do pagamento; por meio da separação de responsabilidades, exige-se que cada uma das partes seja realizada por uma pessoa diferente e que o segundo passo só se concretize após a ocorrência do primeiro.

Diferentemente do modelo de segurança de Bell and LaPadula (1973), cuja mediação de acesso reside no núcleo do sistema, a abordagem de Clark and Wilson (1987) impõe o controle ao nível da aplicação. Esta diferença é resultante dos objetivos pretendidos por cada um dos modelos. O modelo de segurança multinível, cuja base é o modelo de Bell-LaPadula, pretende controlar o fluxo de informação, definido em termos de operações de leitura e escrita. Já o modelo comercial de integridade, como definido por Clark and Wilson (1987), deve assegurar que a informação só é modificada de forma autorizada por pessoas autorizadas.

2.3 Considerações sobre o controle de acesso compulsório

O controle de acesso compulsório é uma política de acesso suportada por sistemas que processam dados com elevada sensibilidade, como por exemplo informações de governo ou dados sigilosos de corporações.

Sistemas que provêem o controle de acesso compulsório devem associar rótulos de sensibilidade a todos os sujeitos (usuários, programas) e a todos os objetos (arquivos, diretórios, dispositivos, etc.) do sistema. O rótulo de sensibilidade do usuário especifica o nível de confiança associado àquele usuário. O rótulo de sensibilidade de um objeto especifica o nível de confiança que o usuário deve possuir para ser capaz de acessar aquele objeto. Desta forma, o controle de acesso compulsório utiliza rótulos de sensibilidade para determinar quem pode acessar qual informação no sistema.

Os rótulos em conjunto com o controle de acesso compulsório implementam uma política de segurança multinível, como aquela formalizada por Bell and LaPadula (1973).

Em um sistema de controle de acesso compulsório, todas as decisões de acesso são realizadas pelo sistema. A decisão para negar ou permitir o acesso a um objeto, como por exemplo um arquivo, envolve uma interação entre as seguintes entidades:

- O rótulo do sujeito:

  SUPER SECRETO;

- O rótulo do objeto — por exemplo um arquivo A:

  SECRETO;

- Uma solicitação de acesso — por exemplo, uma tentativa de leitura no arquivo.

Quando é realizada a tentativa de leitura do arquivo A, o sistema compara o rótulo de sensibilidade do sujeito com o rótulo do arquivo para determinar se o sujeito tem permissão de leitura do arquivo.

Para ser possível a leitura de um objeto, o nível de sensibilidade do sujeito deve dominar o nível de sensibilidade do objeto, ou seja, o rótulo do sujeito deve ser igual ou superar o rótulo do objeto. Por exemplo, se o rótulo do arquivo é SECRETO, para ser possível a leitura, o rótulo do sujeito deve ser SECRETO ou maior do que SECRETO.

Para ser possível a escrita, o nível de sensibilidade do objeto deve dominar o nível de sensibilidade do sujeito, ou seja, o rótulo do sujeito deve ser igual ou inferior ao rótulo do objeto. Assim, se o rótulo do sujeito for SUPER SECRETO e o rótulo do objeto for SECRETO, o sujeito não pode escrever no objeto.

A figura 2.1 ilustra esse princípio.

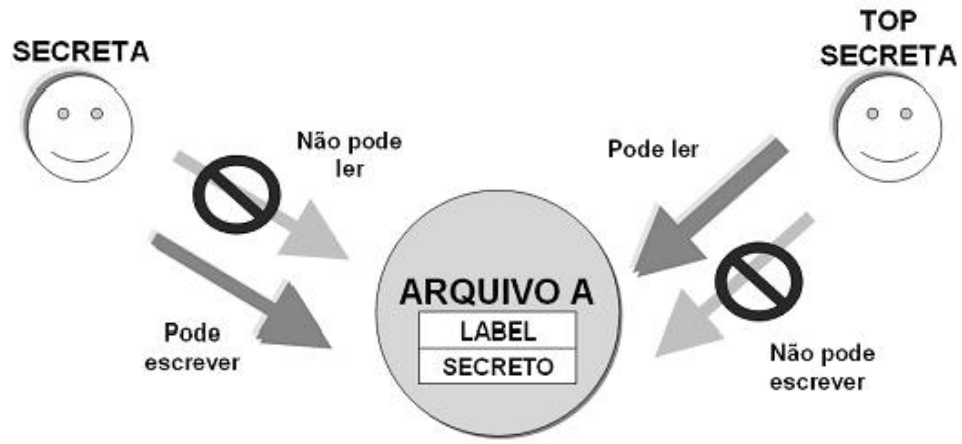

Figura 2.1: Visão do controle de acesso compulsório

2.4 Considerações sobre o controle de acesso discricionário

O controle de acesso discricionário é uma política de acesso a objetos baseado na identidade de usuário e/ou grupos aos quais eles pertencem, possivelmente configuráveis através de permissões.

Em contraste com o controle de acesso compulsório, onde o controle de acesso é imposto pelo sistema, o controle de acesso discricionário é aplicado conforme a própria discrição do usuário que tenha direitos para tal, como por exemplo, ao gravar um arquivo em seu diretório de trabalho; com o controle de acesso compulsório, pode-se escolher sobre o que fazer com os dados de sua propriedade; com o acesso compulsório, isso não é possível.

O controle de acesso discricionário não apenas permite ao usuário informar ao sistema quem pode acessar os seus dados, mas também permite especificar o tipo de acesso permitido. Por exemplo, o usuário pode desejar que todos do sistema sejam capazes de ler um arquivo em particular, mas somente ele deve ser capaz de modificar aquele arquivo.

Capítulo 3

O modelo RBAC — *Role-Based Access Control*

3.1 Visão Geral

No controle de acesso baseado em papéis (RBAC), o acesso a objetos de um sistema computacional é baseado no papel que o usuário escolhe para exercer numa determinada sessão (Ferraiolo et al., 2003). O modelo RBAC é, ao mesmo tempo, uma abstração e uma generalização. Uma abstração porque não inclui propriedades que não são relevantes para a segurança. Uma generalização, pois diversos projetos ou modelos de controle de acesso podem ser considerados uma interpretação, ou caso particular, do modelo baseado em papéis. Logo, o modelo RBAC é útil como uma base para o projeto de diversos sistemas de TI. Seu principal propósito consiste em facilitar a gerência e manutenção das autorizações de acesso no sistema, e a análise de vulnerabilidades necessária à sua evolução.

O modelo RBAC possui quatro sub-modelos:

- **RBAC núcleo** (*core RBAC*), que abrange o conjunto básico de funcionalidades presentes em todos os sistemas RBAC;
- **RBAC hierárquico** (*hierarchical RBAC*), que inclui o conceito de hierarquia de papéis, definida através de uma ordenação de papéis;
- **RBAC com restrições estáticas** (*static constrained RBAC*), que inclui restrições impostas na designação de papéis; e
- **RBAC com restrições dinâmicas** (*dynamic constrained RBAC*), que impõe restrições na ativação de conjunto de papéis que podem ser incluídos como atributos do sujeito de um usuário.

No RBAC núcleo, são descritos cinco elementos administrativos:

1. usuários,
2. papéis,

3. permissões, onde permissões são compostas de

4. operações aplicadas a

5. objetos.

O conceito central no modelo é o papel, onde o papel é uma construção semântica em torno da qual cada política de segurança é formulada. As mais básicas dessas relações são as designações de usuários e permissões. No RBAC, permissões são associadas a papéis, e usuários são feitos membros de papéis, no sentido de poderem exercê-los. . Assim, podem exercer as permissões , atribuídas a determinado papel, ao assumi-lo. A figura 3.1 mostra o relacionamento entre usuários, papéis e permissões. Na figura, usa-se as setas nas duas direções para indicar um relacionamento muitos-para-muitos. Por exemplo, um único usuário pode se associar com um ou vários papéis, e um único papel pode ter um ou vários membros.

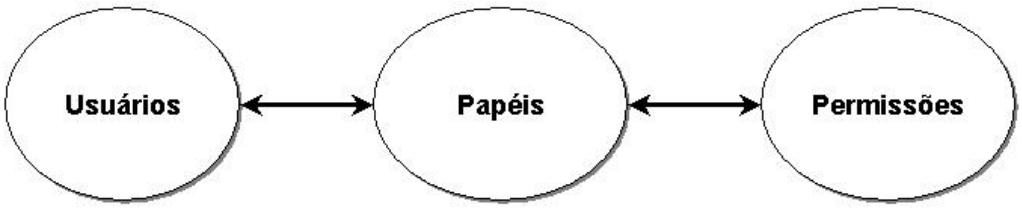

Figura 3.1: Relacionamento entre usuários, papéis e permissões

Esse arranjo possibilita grande flexibilidade e granularidade para a atribuição de permissões a papéis, e de usuários a papéis. Além disso, o aumento da flexibilidade no controle de acesso a recursos também fortifica a aplicação do *princípio do privilégio mínimo*.

Essa capacidade administrativa do modelo RBAC é uma de suas maiores virtudes. A administração das informações sobre autorizações é largamente conhecida como um processo oneroso com um enorme e recorrente gasto.

Sob o modelo RBAC, usuários são designados a papéis baseado na sua competência, autoridade e responsabilidades. Essas associações de usuários a papéis podem ser facilmente revogadas e novas podem ser estabelecidas com a mesma facilidade. Com o RBAC, as permissões não são dadas aos usuários individualmente. Ao invés disso, permissões são associadas com os papéis, um conjunto bem menor que o de usuários em grandes organizações. Novas associações entre permissões e papéis podem ser criadas enquanto antigas são removidas na medida que as funções organizacionais mudam e evoluem. Esse relacionamento entre os papéis do modelo RBAC e os papéis exercidos na organização facilita grandemente o entendimento e o gerenciamento de permissões: os administradores de sistema podem atualizar os papéis sem ter que atualizar permissões para cada usuário do sistema individualmente.

3.1.1 Permissões

Ao modelar um sistema de controle de acesso, os administradores de sistemas podem tratar as permissões como um conceito abstrato que se refere à ligação arbitrária entre operações e objetos, levando em consideração, em alguns casos, processos e valores. O conjunto de permissões designado a um papel dá o potencial para que tarefas, funções, ou outra abstração qualquer relativa a trabalho, sejam executadas. Designar um usuário a um papel confere a habilidade de executá-las.

Permissões designadas a um determinado papel refletem políticas determinadas pela organização que possui o sistema, tanto em relação ao método de acesso ao objeto como também em relação ao que é acessado no objeto. Para entender as políticas em relação ao método de acesso, basta saber que existe a possibilidade de que um papel possa só ler dados e outro só escrever dados. Outra possibilidade é um papel poder somente criar dados, mas não editá-los, enquanto outro pode somente editar depois de os dados já estarem criados. Em relação ao acesso, um determinado papel pode acessar todos os dados de um arquivo texto, porém, outro pode acessar só um dos trechos do documento.

As permissões ainda podem refletir outros aspectos como regras impostas pelas condições do ambiente, como por exemplo, expresso por metadados ou ontologias aplicáveis ao recurso em foco. Num exemplo em domínio de aplicação específico, quando um médico deve ser autorizado apenas a divulgar somente o resultado de certos exames, não podendo divulgar os exames por inteiro, pois a porosidade de um ambiente em rede e erros humanos podem violar a privacidade do paciente. Permissões pdem refletir também leis e regulamentos quando, por exemplo, uma enfermeira está autorizada apenas a adicionar novas entradas no prontuário de um paciente, mas não modificações no registro.

3.1.2 Comparação entre papéis e Listas de controle de acesso

Uma lista de controle de acesso (*access control list — ACL*) contem os nomes dos sujeitos autorizados a acessar o objeto ao qual ela se refere, e as suas respectivas permissões de acesso. Por exemplo, quando um usuário deseja acessar um determinado arquivo, o sistema faz uma busca na ACL deste arquivo por uma entrada que corresponda ao usuário. Se a encontra, verifica se a operação requisitada faz parte do conjunto de permissões que esse usuário possui para acesso ao arquivo. Em caso positivo, o acesso será concedido.

O privilégio de criação e manutenção das ACL's de um sistema depende do conceito de "proprietário da informação" (ou do recurso) do modelo de controle de acesso aplicado. Em sistemas que aplicam políticas discricionárias, o proprietário do objeto é seu criador, o qual tem o privilégio de administrar o acesso a esse objeto. Já nos sistemas que suportam políticas não discricionárias, a propriedade de todos os objetos é centralizada pelos

administradores do sistema. Para aumentar a eficiência na administração, a entrada de uma ACL pode ser, além de um único sujeito, um grupo deles, cujos membros, conseqüentemente, terão as mesmas permissões de acesso ao objeto correspondente. Isso evita a repetição na atribuição de permissões.

Basicamente, um papel pode ser considerado semelhante a um grupo. Um papel representa um ou mais usuários, e um usuário é membro de um ou mais papéis. Da mesma forma, é possível associar uma permissão a vários grupos e papéis, assim como um grupo ou papel pode possuir várias permissões. Neste nível de entendimento não se percebem diferenças entre papéis e grupos de usuário. Contudo, há uma variedade de diferenças entre seus significados nos modelos de acesso e entre seu uso nas implementações.

Uma diferença essencial está no fato de que o conceito de grupo é específico de cada implementação. Por exemplo, em alguns sistemas *UNIX*, somente um grupo pode ser associado a cada arquivo. Outros sistemas operacionais permitem a associação de vários grupos. Ainda há aqueles que não permitem a um usuário fazer parte de mais de um grupo.

Já no modelo RBAC, o papel é um elemento central e está definido em termos de um conjunto de propriedades fixas, não importando a sua implementação. Uma dessas propriedades é a relação muitos-para-muitos entre papéis e usuários e entre papéis e permissões. Neste caso, para que um grupo possa ser considerado como uma implementação do conceito de papel do modelo RBAC, sua estrutura não pode restringir o número: de grupos que poderiam ser criados; de usuários que possam ser membros de um grupo; de grupos que o usuário pode pertencer simultaneamente; de grupos incluídos na lista de controle de acesso de um objeto. Ainda, no RBAC o usuário estará, numa sessão, sempre exercendo algum papel, cujas permissões não seriam combinadas de forma fixa com as de seus outros possíveis papéis, como seria o caso com grupos, ao acessar um recurso.

Sendo o RBAC um modelo, e não um mecanismo, pode ser implementado em diversos tipos de sistema para incluir gerência de redes e empreendimentos com um escopo que vai muito além de apenas um sistema operacional ou aplicação. Papéis e usuários são tratados como entidades globais que possuem relação com permissões dos diferentes sistemas onde o modelo é aplicado, tornando a administração mais simples, eficiente e menos propensa às falhas no controle de autorização mais sutis.

Por fim, o conceito principal do modelo RBAC são os relacionamentos dos papéis. Além de possuir os relacionamentos com usuários e permissões, que servem como uma construção semântica na qual as políticas de controle de acesso são formuladas, existe ainda as relações de hierarquia e herança entre papéis, e os relacionamentos de restrição estática e dinâmica, características não existentes entre as implementações de grupos.

3.1.3 Ativação de papéis

Da mesma maneira que vários outros tipos de modelos, o RBAC inclui o conceito de sujeitos e objetos. De modo geral, as propriedades e mapeamentos definidos pelo modelo RBAC podem ser divididos em duas componentes separadas, mas dependentes: uma estática e outra dinâmica. A componente estática, discutida resumidamente até aqui, é aquela que não possui relacionamentos que envolvem o conceito de sujeito. Quando aplicamos políticas de segurança dinamicamente em um sistema, usamos a noção de sujeitos, que são entidades ativas cujo acesso a papéis, operações e objetos, deve ser controlado. Um sujeito agindo em nome de um usuário efetua todas as suas requisições. É referenciado unicamente por um identificador, usado para determinar se o referente está autorizado a exercer um papel e, se estiver, poder ativá-lo. Um usuário pode estar associado com inúmeros sujeitos a qualquer momento. Cada sujeito pode ter uma combinação diferente de papéis ativados. Essa funcionalidade mantém o *princípio do privilégio mínimo*, já que um usuário que está associado com vários papéis pode ativar um conjunto de papéis mínimo para realizar as atividades de cada sujeito.

3.2 Definição formal

O modelo RBAC possui uma definição formal, dada por Ferraiolo and Kuhn (1992). Para cada sujeito, o papel ativo é aquele que o sujeito está usando naquele momento:

$$\texttt{PA(s : sujeito) = papel ativo do sujeito s}\}$$

Cada sujeito pode ser autorizado a agir como um ou mais papéis:

$$\texttt{PAu(s : sujeito) = \{papéis autorizados para sujeito s\}}$$

Cada papel pode ser autorizado a realizar uma ou mais transações:

$$\texttt{TA(p : papel) = \{transações autorizadas para papel p\}}$$

Sujeitos podem executar transações. O predicado `exec(s, t)` é verdadeiro se, e somente se, o sujeito `s` pode executar a transação `t` naquele momento; caso contrário, é falso:

$$\texttt{exec (s : sujeito, t : transação) = \{verdadeiro} \iff \texttt{sujeito s pode executar transação t}\}$$

Um sujeito pode executar uma transação somente se o sujeito selecionou ou foi designado a um papel:

$$\texttt{s : sujeito, t : transação} \cdot \texttt{exec(s,t)} \longrightarrow \texttt{PA(s)} \neq \emptyset$$

O papel ativo de um sujeito deve estar autorizado para aquele sujeito:

$$\texttt{PA(s)} \subset \texttt{PAu(s)}$$

Um sujeito pode executar uma transação somente se a transação está autorizada para o papel ativo do sujeito

$$\texttt{s : sujeito, t : transação} \cdot \texttt{exec(s,t)} \longrightarrow \texttt{t} \in \texttt{TA(PA(s))}$$

É importante notar que está última regra condicional permite que outras restrições possam ser colocadas na execução de transações.

3.3 Modelo RBAC do NIST

3.3.1 Visão geral

O padrão proposto pelo NIST (Ferraiolo et al., 2001) foi projetado com o objetivo de prover uma definição autoritativa de funcionalidades bem aceitas do RBAC, para o uso em sistemas de gerenciamento de autorizações. O padrão é estruturado em duas partes, descritas como se segue:

- *Um modelo de referência*, definido como uma coleção de quatro componentes do modelo RBAC: o núcleo, o hierárquico, o de restrição estática e o de restrição dinâmica. Os componentes do modelo existem para fornecer um vocabulário padrão de termos relevantes para definir um extenso conjunto de funcionalidades do RBAC.

- *Uma especificação funcional* que projeta o modelo de referência em um conjunto congruente de componentes funcionais, onde cada componente define requisitos específicos de operações administrativas para criar e manter os conjuntos e relações do RBAC, funções de revisão e funcionalidades do sistema correspondentes ao respectivo componente do modelo.

O padrão descreve uma abordagem lógica ao definir pacotes de componentes funcionais, onde cada pacote corresponde a um segmento de ambiente ou mercado diferente. O conceito básico é que cada componente possa opcionalmente ser selecionado para ser incluído em um pacote com apenas uma execução - componente núcleo do RBAC deve estar incluído em todos os pacotes.

Na definição de pacotes funcionais, o núcleo do RBAC é único pelo fato de que é fundamental e deve ser incluído em todos os pacotes. Inclusive, para alguns ambientes, a sua seleção já é suficiente. O componente hierárquico é dividido em duas partes: hierarquias de papéis gerais e limitadas. O componente de restrições estáticas também está dividido em duas partes: restrições estáticas e restrições estáticas com a presença de hierarquias. Por fim, o componente de restrições dinâmicas não possui nenhuma divisão e é escolhido por inteiro ou não.

Cada componente do modelo de referência é definido pelos seguintes subcomponentes:

- um conjunto de conjuntos básicos de elementos;
- um conjunto de relações RBAC envolvendo esses conjuntos de elementos;
- um conjunto de funções mapeadoras que geram instâncias de membros de um conjunto de elementos para uma dada instância de um outro conjunto de elementos.

A especificação funcional do RBAC esboça a semântica de várias funções que são necessárias na criação e manutenção dos componentes do modelo RBAC (conjuntos de elementos e suas relações), como também no suporte de funções do sistema. As três categorias de funções em uma especificação funcional do RBAC e seu propósito são descritas como se segue:

- *Funções administrativas*: Criação e manutenção de conjuntos de elementos e suas relações na construção dos vários componentes do modelo RBAC;
- *Funções de suporte do sistema*: Funções que são necessárias pela implementação do RBAC para auxiliar na construção do modelo RBAC (por exemplo, atributos de sessão e lógica de decisão para o acesso);
- *Funções de inspeção*: Funções que revisam os resultados das ações criadas pelas funções administrativas.

3.3.2 O núcleo do RBAC

No componente núcleo do RBAC as funções administrativas são descritas como se segue:

- *Criação e manutenção de conjuntos de elementos*: os conjuntos básicos de elementos são `USERS` (usuários), `ROLES` (papéis), `OPS` (operações) e `OBS` (objetos). Desses conjuntos, os últimos dois são considerados predefinidos pelo sistema de informação sobre o qual o RBAC é desenvolvido;
- *Criação e manutenção de relações*: as duas principais relações são a designação de usuários a papéis (*user-to-role assignment* - UA) e a designação de permissões a papéis (*permission-to-role assignment* - PA). As funções de criação e remoção de instâncias das relações UA são `AssignUser` (criação) e `DeassignUser` (remoção). Para as relações PA, as funções necessárias são `GrantPermission` (criação) e `RevokePermission` (remoção).

As funções de suporte são necessárias para a gerência de sessões e na tomada de decisões do controle de acesso. Um papel ativo é necessário na regulação do controle de acesso para um usuário durante uma sessão. A função que cria a sessão estabelece um conjunto padrão de papéis ativos

para o usuário no início da sessão. O usuário pode então alterar a composição desse conjunto padrão durante a sessão adicionando ou removendo papéis. As funções relacionadas com a adição e remoção de papéis ativos e outras funções auxiliares são as seguintes:

- `CreateSession`: Cria um sessão de usuário e fornece ao usuário um conjunto padrão de papéis;
- `AddActiveRole`: Adiciona um papel como um papel ativo para a sessão atual;
- `DropActiveRole`: Remove um papel do conjunto de papéis ativos da sessão atual;
- `CheckAccess`: Determina se um processo da sessão tem permissão para realizar a operação requerida em um objeto.

As funções de inspeção são aquelas que permitem ao administrador do sistema verificar todos os usuários relacionados com um determinado papel bem como os papéis relacionados com um determinado usuário. Além disso, também permitem que o administrador verifique os resultados das funções de suporte ao determinar os atributos de sessão (papéis ativos e permissões). Algumas das funções são opcionais (O) na implementação do RBAC, outras são mandatórias (M). Elas são descritas abaixo:

- `AssgnedUsers` (M): retorna o conjunto de usuários designados a um determinado papel;
- `AssignedRoles` (M): retorna o conjunto de papéis designados a um determinado usuário;
- `RolePermissions` (O): retorna o conjunto de permissões dadas a um determinado papel;
- `UserPermissions` (O): retorna o conjunto de permissões que um determinado usuário tem a partir dos seus papéis;
- `SessionRoles` (O): retorna o conjunto de papéis ativos associados à sessão;
- `SessionPermissions` (O): retorna o conjunto de permissões disponíveis na sessão (i. e., a união de todas as permissões designadas aos papéis ativos da sessão);
- `RoleOperationsOnObject` (O): retorna o conjunto de operações um dado papel pode realizar em um dado objeto;
- `UserOperationsOnObjects` (O): retorna o conjunto de operações um dado usuário podem realizar em um dado objeto.

3.3.3 RBAC Hierárquico

Nesse componente estão presentes todas as funções administrativas que existem no RBAC núcleo. No entanto, a função `DeassignUser` deve ser redefinida porque com a presença de hierarquias de papéis um usuário pode herdar uma autorização para um papel mesmo não estando diretamente designado para este papel. Logo, a hierarquia permite que usuários herdem permissões de papéis que estão abaixo do seu papel na hierarquia. A questão é: o usuário somente pode ser removido de uma relação com um papel diretamente designado a ele, ou pode ser removido de uma relação com um papel herdado? A resposta fica a cargo da implementação e não é descrita no padrão.

As funções administrativas adicionais do componente hierárquico são relativas a criação e manutenção de relações de hierarquia entre papéis. As operações consistem em criar ou remover uma relação de hierarquia entre dois papéis existentes em um conjunto de papéis, ou adicionar um novo papel à hierarquia, posicionando-o apropriadamente. O nome e o propósito dessas funções são descritas como se segue:

- `AddInheritance`: estabelece uma nova relação de herança imediata entre dois papéis;
- `DeleteInheritance`: remove uma relação de herança imediata entre dois papéis;
- `AddAscendant`: cria um novo papel e coloca-o como ascendente imediato de um papel;
- `AddDescendant`: cria um novo papel e coloca-o como descendente imediato de um papel.

As funções de suporte do componente hierárquico são aquelas do núcleo do RBAC, com a mesma funcionalidade, contudo as funções `CreateSession` e `AddActiveRole`, por causa da hierarquia de papéis, precisam ser redefinidas. Da mesma forma, as funções de inspeção do núcleo do RBAC também são válidas para o componente hierárquico. Em adição, algumas funções são redefinidas pelo fato de que os conjuntos de relações entre papéis e usuários não possuem apenas elementos diretamente associados, mas também aqueles advindos das relações de hierarquia de papéis. As funções são definidas abaixo:

- `AuthorizedUsers`: retorna o conjunto de usuários diretamente associados a um papel bem como aqueles usuários que estão associados a papéis que herdam esse papel;
- `AuthorizedRoles`: retorna o conjunto de papéis diretamente associados a um usuário bem como aqueles papéis herdados pelos papéis diretamente associados ao usuário.

Além disso, as funções de inspeção relativas às permissões associadas a usuários e papéis devem ser igualmente redefinidas considerando-se as relações de hierarquia.

3.3.4 Componente de restrições estáticas do RBAC

Para um modelo RBAC com restrições estáticas que não inclui hierarquia de papéis, as funções administrativas devem conter todas aquelas que estão no núcleo do RBAC. Contudo, a função `AssignUser` deve incorporar a funcionalidade de verificação e garantia que a designação de um papel para o usuário não viola qualquer uma das restrições estáticas associadas a ela.

As relações de restrição estática consistem em um tripé: `SSD_Set_Name`, nome que indica a transação ou processo no qual uma associação de um usuário a um determinado conjunto de papéis simultaneamente deve ser restringida; `role_set`, conjunto de papéis constituintes da relação de restrição estática; `SSD_Card`, que define a cardinalidade do subconjunto contido em `role_set` a qual a associação simultânea de papéis deve ser restringida. Portanto, as funções administrativas relacionadas a criação e manutenção de uma relação de restrição estática são operações para:

- criar e remover uma instância de uma restrição estática (`CreateSSDSet` e `DeleteSSDSet`);
- adicionar e remover papéis ao conjunto de papéis da restrição estática (`AddSSDRoleMember` e `DeleteSSDRoleMember`);
- configurar o parâmetro de cardinalidade da restrição (`SetSSDCardinality`).

Para o caso de modelos RBAC com hierarquia de papéis, as funções acima produzem o mesmo resultado com uma única exceção: restrições relacionadas à hierarquia de papéis devem ser levadas em conta ao aplicar uma dessas funções.

3.3.5 Componente de restrições dinâmicas do RBAC

A semântica para a criação de instâncias de restrições dinâmicas é idêntica àquela das restrições estáticas. Enquanto as restrições estáticas são aplicadas no momento da designação de papéis ao usuário (e também na criação de hierarquia de papéis), as restrições dinâmicas são aplicadas no momento da ativação do papel por um usuário durante uma sessão.

As funções administrativas para restrições dinâmicas são: criação e remoção de instâncias de restrições dinâmicas (`CreateDSDSet` e `DeleteDSDSet`); inclusão e remoção de papéis do conjunto de papéis da restrição dinâmica (`AddDSDRoleMember` e `DeleteDSDRoleMember`); e configuração de cardinalidade dos papéis da restrição dinâmica (`SetDSDCardinality`).

As funções de suporte para criação de sessão (`CreateSession`) e ativação de papéis (`AddActiveRole` e `DropActiveRole`) devem ser modificadas para aplicar as restrições dinâmicas no momento em que são chamadas.

Capítulo 4

O MACA - *Middleware de Autenticação e Controle de Acesso*

Este capítulo apresenta o MACA – *Middleware de Autorização e Controle de Acesso*, a ferramenta que foi utilizada como servidor de controle de acesso para o protótipo desenvolvido. O MACA é uma ferramenta com código aberto e licença livre que implementa um modelo de autorização contextual para o controle de acesso baseado em papéis. Uma autorização contextual usa informações ambientais disponíveis durante o momento de acesso para decidir se um usuário tem – ou não – o direito de acessar um recurso, e de que modo.

Este capítulo está dividido da seguinte forma: a seção 4.1 explica o que vem a ser um contexto do MACA; a seção 4.2 define o que são regras de autorização; a seção 4.3 apresenta o modelo de autorização do MACA; a seção 4.4 apresenta aspectos de arquitetura e de implementação do MACA; por fim, é apresentada uma visão geral do algoritmo de decisão de acesso do MACA.

4.1 Os contextos

Um contexto é qualquer informação que pode ser utilizada para distinguir e caracterizar uma entidade (pessoa, lugar, objeto) considerada relevante para a interação entre um usuário e uma aplicação, incluindo-se aí o próprio usuário e a própria aplicação. O contexto denota informações ambientais para decidir se um acesso será permitido ou proibido, no momento em que o usuário tenta efetuá-lo.

No MACA, há o contexto do usuário corrente (aquele que iniciou uma sessão e que solicita uma autorização de acesso); o contexto temporal, com informações sobre hora e data; o contexto de rede, com informações de redes e de comunicação; e outros contextos que podem ser livremente definidos e implementados pelos formuladores e administradores da política de acesso.

As informações disponíveis em um contexto são recuperadas de acordo

com a sintaxe <nome do contexto>.<nome>, onde o primeiro termo designa o nome do contexto e o segundo termo pode designar:

- uma variável contextual, como, por exemplo, a variável `nome` do contexto do usuário (`usuarioCtx.`**`nome`**) ou um dia da semana do contexto temporal (`dtCtx.`**`dia_semana`**);
- um conjunto contextual, como, por exemplo, o conjunto de dias úteis da semana do contexto temporal (`dtCtx.`**`dias_úteis`**); ou
- uma função contextual, como, por exemplo, a função `advCtx.`**`assiste`**`(umCodParte, usuarioCtx.registro_profissional)`, que informa se a parte identificada por `umCodParte` é assistida pelo usuário corrente identificado pela variável `usuarioCtx.registro_profissional`.

Deve-se lembrar que os contextos devem refletir, com alto nível de abstração, as entidades e os relacionamentos existentes entre elas no ambiente onde o acesso é realizado, a fim de facilitar a definição das políticas.

4.2 As regras de autorização

Uma regra de autorização relaciona informações contextuais em expressões lógicas que especificam uma política de acesso para um objeto protegido. As regras são definidas numa linguagem de expressões lógicas capaz de recuperar os valores provenientes de variáveis, conjuntos e funções contextuais para relacioná-las por meio de operadores aritméticos (`+`, `-`, `*`, `%`), relacionais (`>`, `<`, `>=`, `<=`, `=`, `!=`, `in` – pertinência) e booleanos (`&`, `|`, `!`).

Uma regra suporta os seguintes tipos de valores primitivos:

- *Booleano*: valores do tipo `true` (verdadeiro) ou `false` (falso).
- *Inteiro*: tipo que contém valores de números inteiros;
- *Texto*: tipo que contém valores que são cadeias de caracteres com tamanho variável.

Vale lembrar que uma regra de autorização, quando avaliada, sempre deve retornar um valor do tipo booleano, caso contrário, ocorrerá um erro pela impossibilidade de se decidir pela concessão ou negação de acesso.

Há casos, ainda, em que uma regra precisa ser parametrizada para permitir a passagem de informações contextuais do ambiente da aplicação (que solicita a autorização) para a regra, no momento de sua avaliação. Um exemplo dessa situação é mostrado a seguir:

```
umDia in dtCtx.dia_semana
```

Nesse caso, o valor do identificador `umDia` depende do momento em que a aplicação solicita o acesso. Para que a regra se torne válida, o identificador deve se subordinar a um contexto ou ser declarado explicitamente como um parâmetro de uma regra. A regra anterior torna-se uma regra bem formada se for definida da seguinte forma,

```
exp-abs(umDia)
{
    umDia in dtCtx.dia_semana
}
```

A palavra reservada `exp-abs` indica a definição de uma expressão parametrizada; o parâmetro formal `umDia` denota um dia qualquer, que é passado como argumento no momento de avaliação da regra.

Para uma regra parametrizada ser avaliada, é necessário que uma lista de argumentos correspondentes aos parâmetros formais seja definida. Podem ser passados como argumentos de uma regra parametrizada valores primitivos (booleanos, inteiros, texto); variáveis, conjuntos e funções contextuais; e demais expressões da linguagem, inclusive regras parametrizadas.

4.3 O modelo de autorização contextual do MACA

Conforme já dito, o MACA estende a especificação do controle de acesso baseado em papéis, com o acréscimo de autorizações contextuais, de autorizações positivas e negativas, de autorizações fortes e fracas, da possibilidade de revogar-se autorizações e da separação de responsabilidades estática e dinâmica baseadas em conflitos fortes e fracos entre autorizações.

No MACA, uma autorização positiva (+) é aquela que concede um acesso. Já uma autorização negativa (-) , por sua vez, é aquela que proíbe explicitamente o acesso. Quando o acesso é proibido para a maioria dos papéis descendentes, uma autorização negativa deve ser usada no papel ascendente. De maneira similar, quando o acesso é permitido para a maioria dos papéis descendentes, uma autorização positiva deve ser definida no papel ascendente. Por exemplo: considere que, em uma hierarquia de papéis, a maioria dos papéis tem o direito de pesquisar um processo; então, essa autorização deve ser definida como positiva no papel ascendente, a fim de que os descendentes herdem essa autorização. De forma análoga, suponha que apenas uma minoria da descendência tenha a capacidade de proferir uma sentença; assim, essa autorização pode ser definida como negativa no papel ascendente e redefinida como positiva apenas nos papéis que possuem o privilégio de executá-la.

Uma autorização pode, ainda, ser do tipo fraca ou forte. Autorizações fortes estabelecem políticas absolutas, que não podem ser revogadas, enquanto as fracas são utilizadas para definir políticas mais permissivas.

Desta forma, autorizações fortes e fracas podem ser utilizadas para resolver conflitos. As autorizações fortes devem ser utilizadas para proteger recursos considerados críticos, que envolvam conflitos de interesses. Na presença de uma autorização forte em um papel, esta não pode ser redefinida nos papéis descendentes; quando da existência de conflitos entre autorizações fortes, opta-se por negar o acesso. As autorizações fracas são utilizadas para definir políticas que podem ser revogadas, permitindo que autorizações possam ser redefinidas em papéis descendentes; quando da ocorrência de conflitos entre autorizações fracas, prevalecerá aquela que concede o acesso.

4.4 Arquitetura e implementação do MACA

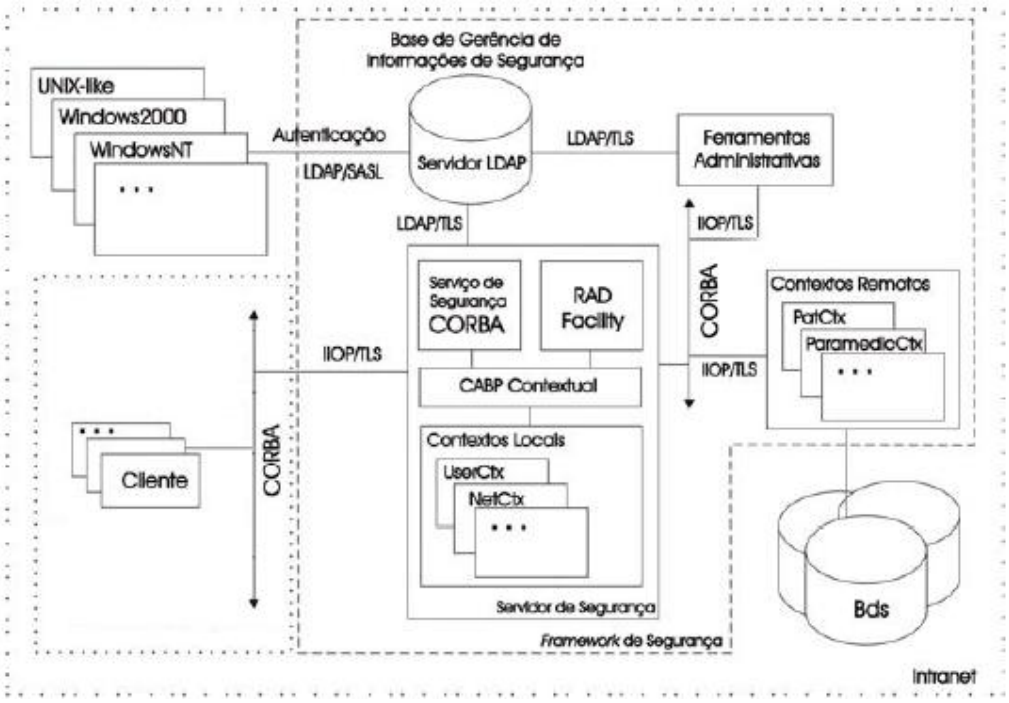

Figura 4.1: Arquitetura do MACA

A arquitetura ilustrada na figura 4.1 especifica os componentes de software e de comunicação para suportar a implementação do MACA. É um modelo cliente-servidor multicamada com os seguintes componentes principais: um servidor LDAP, encarregado de manter a base de gerência de informações de segurança (BIGS); um servidor de segurança, encarregado de oferecer serviços de autenticação de usuário, de decisão de acesso à recursos, etc.; e finalmente, as aplicações clientes que requisitam estes serviços de segurança. Foram adotados, na arquitetura, padrões de processamento aberto e distribuídos. A seguir, é detalhado cada um desses componentes.

A BIGS mantém os perfis de segurança para proteção da aplicação cliente, tais como, autorizações de acesso, papéis, representações dos recur-

sos protegidos e dos usuários, dados para autenticação, relacionamentos papéis-usuários, papéis-autorizações, etc. Essas informações são armazenadas em um serviço de diretório hierarquizado, cujo acesso e esquemas de descrição de dados são padronizados através do protocolo LDAP. Esquemas de dados preexistentes no LDAP são utilizados no armazenamento de informações sobre usuários (nome, e-mail, etc.), papéis (nome, descrição, membros, etc.) e recursos (nome, descrição, localização, etc.).

Todo acesso à BIGS é realizado através do protocolo LDAP sobre TLS (*Transport Layer Security*) para assegurar confidencialidade e integridade na comunicação. As informações persistentes do MACA foram representadas com esquemas do protocolo LDAP versão 3, sendo que o modelo de dados que representa o MACA foi implementado no software (versão 2.3.11) de código fonte aberto OpenLDAP.

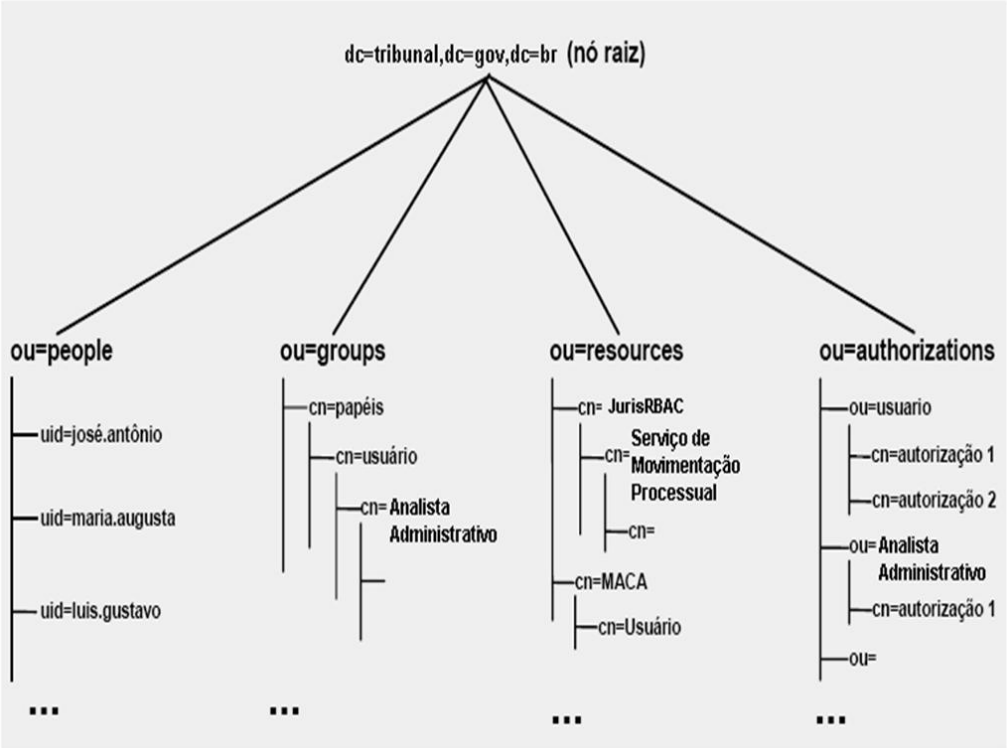

Figura 4.2: Árvore de informações de diretório

O serviço de diretórios LDAP consiste em um conjunto de serviços de diretórios interconectados que se comunicam por meio do protocolo LDAP para atender as requisições (pesquisa, comparação, inserção, remoção, etc.) de clientes. O diretório organiza as informações numa estrutura hierarquizada, denominada *Árvore de Informações de Diretório* (AID), numa versão simplificada do modelo de dados X500 (fig. 4.2). Cada entrada na AID corresponde a um registro armazenado. A estrutura de dados de uma entrada é determinada pelas classes de objeto a que ela pertence; essa classe especifica os tipos de atributos de uma entrada que pertence à classe. Cada entrada é identificada unicamente numa AID pelo seu *Distinguished Name*

(DN). O DN é similar ao caminho completo que identifica unicamente um arquivo num sistema de arquivos.

Cabe ao servidor de segurança oferecer serviços transientes do MACA como autenticação, autorização e controle de acesso às aplicações clientes, dentre outros serviços de segurança. Ele integra o *serviço de segurança do CORBA* (SSC), *o serviço de decisão para acesso a recursos* (RAD – *Facility*) e o serviço de autorização do MACA.

O SSC oferece interfaces padronizadas para autenticação de usuários, gerências de sessões e outros serviços de segurança. O RAD – *Facility* especifica um serviço padrão de decisão de acesso a recursos capaz de suportar diferentes políticas de acesso. O serviço do MACA implementa o modelo de autorização contextual para o controle de acesso baseado em papéis que decide o acesso aos recursos com base na política armazenada na BIGS e nos contextos disponíveis no momento do acesso.

A integração entre o MACA e o SSC requer implementações das interfaces
`PrincipalAuthenticator` e `Credentials`. `PrincipalAuthenticator` realiza a autenticação (por meio do método `authenticate`), verificando a validade da identidade usuário no servidor LDAP. Caso seja bem sucedida, a autenticação resulta na criação de uma instância de `Credentials`, que permite solicitar ao MACA a execução de tarefas relacionadas a sessão do usuário, como por exemplo, obter os papéis de um usuário.

Com o usuário autenticado (e com sua credencial criada) o controle de acesso é imposto pelo ORB (*Object Request Broker*) às chamadas de operações das interfaces dos objetos CORBA servidores (vide figura refmodelo). Assim, o ORB atua como um monitor de referência que impõe, obrigatoriamente, o controle de acesso. Ele utiliza uma implementação da interface `AccessDecision` do SSC para obter, do MACA, os privilégios de acesso conferidos pelos papéis do usuário.

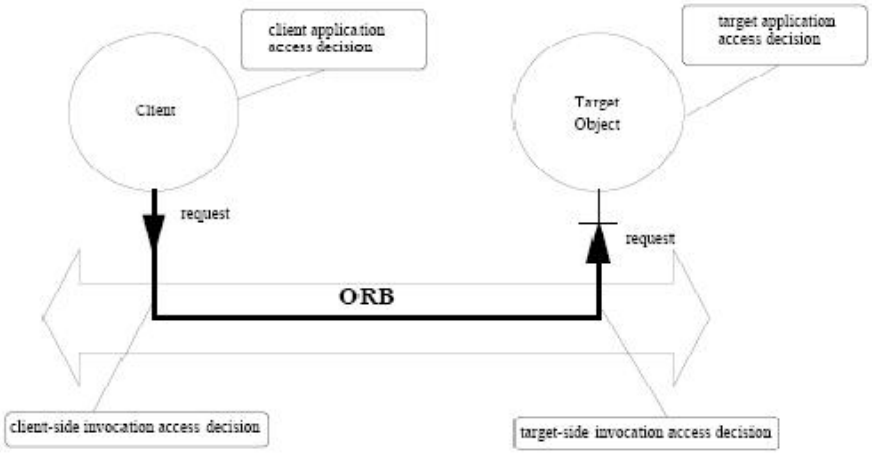

Figura 4.3: Modelo de controle de acesso do SSC

O RAD – *Facility* utiliza o serviço de autorização do MACA como política

de acesso para proteger os recursos das aplicações clientes. A interface do objeto
`AccessDecision` do RAD – *Facility* padroniza o modo como as aplicações clientes (objeto alvo) solicitam autorizações para um usuário acessar um recurso. As interações entre o cliente, o objeto CORBA servidor e o serviço de decisão do RAD – *Facility* dá-se da seguinte maneira:

1. Um cliente solicita que o objeto servidor (objeto alvo) a execução de uma operação.

2. Durante a execução da operação, o objeto servidor solicita ao objeto `AccessDecision` uma autorização para executar a operação. Podem haver parâmetros, que serão utilizados para decidir o acesso.

3. O objeto `AccessDecision` retorna um valor booleano indicando se a autorização foi concedida ou proibida.

4. Com base no retorno de `AccessDecision`, o objeto servidor responde à aplicação cliente, completando a execução da operação solicitada, se o acesso foi autorizado, ou então, interrompendo a execução e sinalizando para o cliente que o acesso foi proibido.

4.5 O algoritmo de decisão de acesso do MACA

O algoritmo de decisão de acesso do MACA recebe os seguintes parâmetros de entrada:

- a operação a executar (`opr`),
- o objeto protegido (`obj`),
- a lista de parâmetros de autorização (`params`), e
- a referência da sessão (`session_ref`).

E retorna como saída um valor booleano indicando se o acesso deve ser permitido ou proibido. Os passos a seguir são executados durante o algoritmo:

1. Caso o conjunto de papéis ativáveis do usuário seja vazio, o acesso é negado (retorna falso) e o algoritmo pára.

2. Caso contrário, verifica-se existe alguma autorização válida, forte, associada a algum papel ativo ou ativável do usuário que proíba a execução da operação `opr` no objeto `obj`. Caso existe, o acesso é negado (retorna falso) e o algoritmo pára.

3. Caso o algoritmo não tenha parado, então verifica-se a existência de alguma autorização válida, forte, associada a algum papel ativo ou ativável do usuário que permita executar a operação `opr` no objeto `obj`. Caso exista, o acesso é concedido (retorna verdadeiro), o eventual papel inativo é ativado na sessão `session_ref` e o algoritmo pára.

4. Caso o algoritmo não tenha parado, então verifica-se a existência de alguma autorização válida, fraca, associada a algum papel ativo ou para ativação do usuário que permita executar a operação `opr` no objeto `obj`. Caso exista o acesso é concedido (retorna verdadeiro), o eventual papel inativo é ativado na sessão `session_ref` e o algoritmo pára.

5. Caso o algoritmo não tenha parado, então o acesso é negado (retorna falso) e o algoritmo pára.

Note-se, pela execução dos passos algoritmo, que as autorizações fortes e negativas prevalecem sobre as demais autorizações, inclusive as eventuais autorizações equivalentes, fortes e positivas. Por sua vez, autorizações fortes prevalecem sobre as autorizações fracas equivalentes e as autorizações fracas positivas prevalecem sobre as autorizações fracas negativas equivalentes.

Capítulo 5

Modelo de acesso implementado como prova de conceito

Esse capítulo apresenta uma implementação de controle de acesso baseado em papéis para processos judiciais virtuais, restrita a uma área específica do domínio de aplicação jurídica. O objetivo desta implementação é servir de prova de conceito da racionalidade da solução em *middleware* proposta por sua arquitetura, tendo como foco imediato sua evolução para um protótipo piloto nesta área. A solução em *middleware* aqui proposta visa a racionalidade em dois sentidos.

No primeiro sentido, abstrato, busca-se explorar o potencial do modelo RBAC para alcançar maior eficiência na gerência de políticas de segurança demandadas, de um lado, por regulamentação complexa das "regras de negócio" (norma processual), e por outro, por altos níveis de adaptabilidade, esclabilidade e sensibilidade nos requisitos de controle operacional das aplicações (informatização do judiciário, sob os influxos da ŞICP-izaçãoŤ)

No segundo sentido, concreto, busca-se explorar o potencial do modelo de desenvolvimento colaborativo, sob regime de licenciamento livre, para alcançar maior eficiência no atendimento à demanda instrumental de mecanismos de segurança próprios para integração e/ou informatização de sistemas processuais na esfera judiciária, atualmente reprimida, alavancando e provendo assim a autonomia do usuário, em relação a fornecedores de componentes, para controlar sua própria política de segurança, autonomia esta que é a própria essência desta segurança.

Daí a adaptação do projeto MACA, software sob licença GPL originalmente desenvolvido para controle de acesso a sistemas de prontuários médicos, para atender aos requisitos de segurança específicos do domínio de aplicação em tela.

A seção 5.1 apresenta o modelo de acesso proposto. A seção 5.2 apresenta aspectos de implementação.

5.1 O modelo de acesso proposto

A solução apresentada cobre os processos judiciais autuáveis em Juizados Especiais Cíveis de primeira instância.

Os Juizados Especiais foram criados pela Lei 9099 de 1995 (`http://www.trf2.gov.br/juizados/9099jef.htm`). São órgãos da Justiça (Poder Judiciário) que servem para resolver as pequenas causas com rapidez, de forma simples, sem despesas e sempre buscando um acordo entre as partes. O artigo 3○ da Lei 9099 — exposto a seguir — esclarece qual a competência dos Juizados Especiais Cíveis.

> "Art. 3○ O Juizado Especial Cível tem competência para conciliação, processo e julgamento das causas cíveis de menor complexidade, assim consideradas:
>
> I - as causas cujo valor não exceda a quarenta vezes o salário mínimo;
>
> II - as enumeradas no art. 275, inciso II, do Código de Processo Civil;
>
> III - a ação de despejo para uso próprio;
>
> IV - as ações possessórias sobre bens imóveis de valor não excedente ao fixado no inciso I deste artigo.
>
> (...)"

A escolha pelo Juizado Especial Cível se justifica pela simplicidade e pela facilidade de aplicação de um procedimento virtual para seus atos processuais. Os Juizados Especiais têm dentre seus princípios processuais a informalidade e a celeridade, buscando obter a rápida solução de conflitos utilizando menos recursos processuais, e com execução efetiva e célere. Desta forma, a virtualização do processo é objetivo desejável nos Juizados Especiais, pois, além de diminuir gastos (como por exemplo, na tramitação de papelada) permite a automatização de boa parte do caminhamento do processo, repassando ao sistema computadorizado a execução de certas tarefas e a implementação do fluxo processual, o que pode resultar na aceleração de sua execução.

Para implementar o processo jurídico virtual autuável em Juizados Especiais Cíveis, visando esta prova de conceito, foi adotado um fluxo simplificado de caminhamento do processo. Este fluxo é apresentado na figura 5.1.

O fluxo, de maneira geral, mostra as etapas pelas quais um processo caminha, desde a entrada do pedido de autuação até o seu arquivamento.

Cada uma dessas etapas pode ser executada por uma pessoa diferente; por exemplo, o autor do processo ou o advogado deste pode entrar com a petição inicial; o Conciliador é o responsável por dirigir a Audiência de Conciliação; o Juiz é quem profere a sentença, etc.

A modelagem desenvolvida baseia-se no modelo de controle de acesso baseado em papéis (RBAC). Como já foi sinalizado na introdução deste capítulo, a natureza da lide processual a torna candidata natural a esse tipo

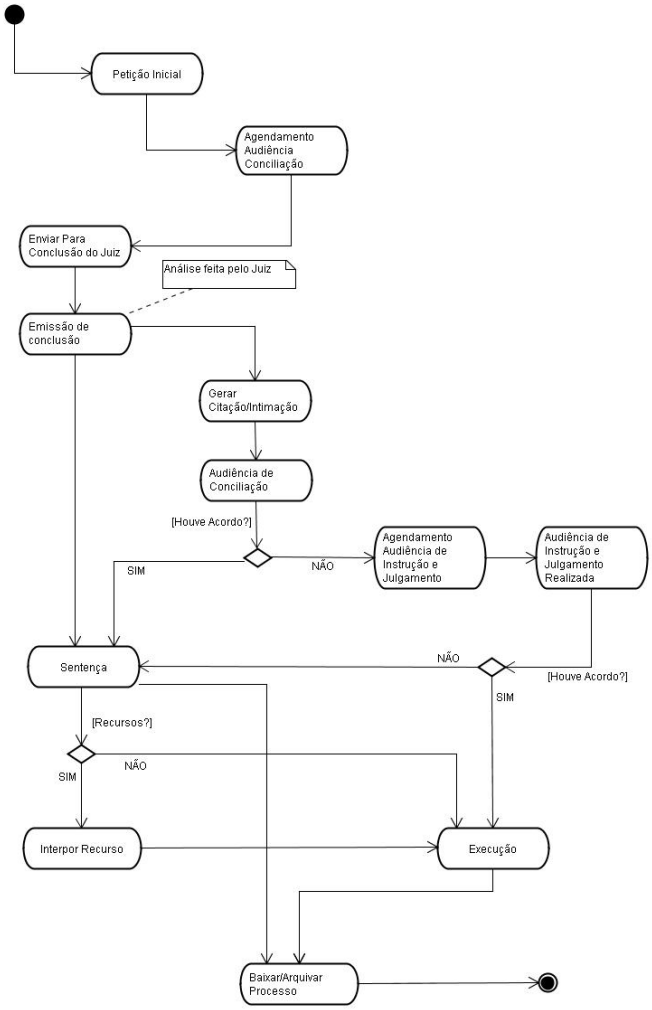

Figura 5.1: Fluxo de caminhamento do processo

de modelagem, já que a própria existência de atores jurídicos é predicada por conjuntos de prerrogativas processuais e normativas, cujo caráter dinâmico atinge não só a vida útil do sistema processual, mas muitas vezes também o próprio processo.

Para aplicação do modelo de acesso proposto, foram definidos serviços forenses que podem ser acessados pelas pessoas que lidam com os processos, no desempenho de suas funções (papéis, no RBAC). Estes serviços são descritos a seguir.

- *Serviço de Cadastro*: permite cadastrar: um processo, quando da autuação de uma petição inicial;cadastrar partes do processo; editar dados de partes do processo; remover partes do processo.

- *Serviço de Pesquisa*: permite a pesquisa por processos (segundo os filtros número do processo, nome de partes ou advogados), a pesquisa por partes e a pesquisa por advogados.

- *Serviço de Movimentação Processual*: permite movimentar um pro-

cesso, segundo o fluxo de processo apresentadoOu seja, desde a autuação da petição inicial até a baixa/arquivamento do processo.

- *Serviço de Acompanhamento e Auditoria*: permite acompanhar o processo em suas diversas fases e visualizar as ações realizadas no processo (e quem as realizou).

O acesso a esses serviços deve atender a política de acesso definida através de regras. Essa política procura respeitar os requisitos do Juizado Especial Cível, e espelhar, na medida do possível, as funções existentes em um Juizado e suas normas processuais. Ela é apresentada, em linhas gerais, a seguir.

- Somente usuários devidamente autenticados (com o uso de certificados digitais) podem acessar o sistema.
- Conforme o controle de acesso baseado em papéis, cada usuário deve estar associado a um papel, para poder executar um serviço forense permitido àquele papel. A figura 5.2 apresenta a hierarquia de papéis que foi utilizada no modelo proposto, onde procurou-se representar os cargos existentes em uma vara desse tipo de Juizado.

Nessa hierarquia, houve a tentativa de, na medida do possível, levar-se em conta as relações de responsabilidade e autoridade existentes quer nos costumes ou nas normas processuais. Com o objetivo de formar uma hierarquia inicial simples, mas passível de evolução, optou-se apenas por retratar os cargos que lidam diretamente com a movimentação do processo. Houve, ainda, a preocupação de definição de papéis genéricos, como os papéis Usuário, Analista Judiciário e Técnico Judiciário, que possuem autorizações que são herdadas pelos papéis mais especializados.

- Os atos praticados são públicos, o que implica que todos os usuários legítimos do sistema podem ver os movimentos realizados e podem pesquisar por processos.
- Somente usuários com papéis que participam em um processo (parte ou advogado) e determinados funcionários do Juizado podem ver os documentos que são peças de um processo.
- Os advogados só podem adicionar documentos, interpor recursos, etc. em processos nos quais representam partes. Além disso, dentro dos prazos processuais em que cabe à parte representada manifestar-se nos autos.
- Os juízes podem estabelecer e estender prazos, como por exemplo: prazos para a Audiência de Conciliação e prazos para interposição de recursos.
- Somente juízes podem determinar a baixa de um processo, desde que não obstado por norma processual impeditiva.

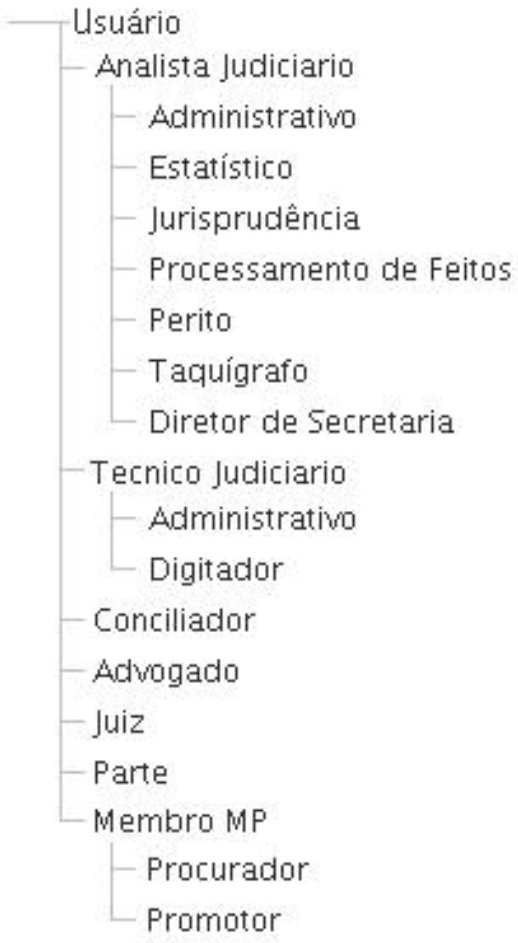

Figura 5.2: Hierarquia de papéis considerada na implementação do modelo de acesso aos processos

A figura 5.3 ilustra a política de acesso definida. São mostrados os recursos principais e as autorizações ligadas aos principais papéis.

Figura 5.3: Do lado esquerdo estão representados os recursos acessíveis, com as respectivas operações associadas.Do lado direito, as autorizações contextuais associadas a cada papel.

5.2 Aspectos de implementação

Esta seção trata de aspectos de implementação do protótipo desenvolvido. O protótipo foi implementado na plataforma Java 2 SE, versão 5. Aqui serão discutidos aspectos relativos a autenticação e aspectos relativos a autorizações.

O MACA, na versão utilizada (3.2.2c), só fornece suporte para autenticação baseada em senha e nome de usuário. Para adequar a forma de autenticação àquela que será demandada em implementações piloto que busquem testar e eventualmente atender demandas de imformatização e/ou integração com serviços existente, sob o regime normativo de uma infra-estrutura de chaves públicas, optou-se por estendê-lo para permitir a autenticação via certificados digitais e chaves privadas correspondentes.

Além de buscar cobrir o potencial de demanda deste importante momento da Justiça Brasileira, a utilização de certificados para autenticação, nesse tipo de arquitetura, abre a possibilidade para utilização de campos específicos (extensões X.509) na alocação de papéis, na decisão de se permitir ou se negar acesso de um usuário autenticado a um determinado serviço (Seção 6.3). Ademais, implementações-piloto de protótipos desta arquitetura podem também servir de base para prova de conceito ou para homologação de normas em considração por uma ICP à qual esteja subordinada o Juizado correspondente. Desta forma, fez-se necessária a modificação no código do MACA para satisfazer essa necessidade de projeto.

Assim, foi-lhe acrescentada mais uma forma de autenticação, chamada `X509_CERTIFICATE`.

Quando `X509_CERTIFICATE` é repassado ao MACA, durante a execução do protocolo de autenticação por uma aplicação-cliente, isso indica que a autenticação será realizada via certificado digital . No protótipo implementado, assume-se que o certificado, (e a respectiva chave privada) estão armazenados em um *keystore*.

Para a criação do *keystore*, a ser utlizado nos testes do protótipo implementado, foi utilizada a ferramenta `keytool` do Java, que é um gerenciador de *keystores* de chaves privadas e certificados, incluindo bibliotecas de geração de chaves.

5.2.1 Geração de chaves

A seguir, é apresentada o código para a criação de um certificado com essa ferramenta.

```
keytool -genkey -alias usuario -keyalg RSA -keystore
user_keystore
 Enter keystore password:  password
 What is your first and last name?
 [Unknown]:  Usuario
 What is the name of your organizational unit?
```

```
[Unknown]:  Seção XYZ
What is the name of your organization?
[Unknown]:  Tribunal de Pequenas Causas
What is the name of your City or Locality?
[Unknown]:  Brasília
What is the name of your State or Province?
[Unknown]:  DF
What is the two-letter country code for this unit?
[Unknown]:  BR
Is CN=Usuario, OU=Seção XYZ, O="Tribunal de Pequenas
Causas",
L=Brasília, ST=DF, C=BR correct?
[no]:  yes
Enter key password for <usuario>
(RETURN if same as keystore password):  <CR>
```

Isso cria um certificado auto-assinado com as correspondentes chaves públicas e privadas e os armazena no *keystore* `user_keystore`.

5.2.2 Autenticação bidirecional usuário-servidor MACA

O protótipo, durante a autenticação, pede que o usuário informe o seu *keystore* (veja figura 5.4).

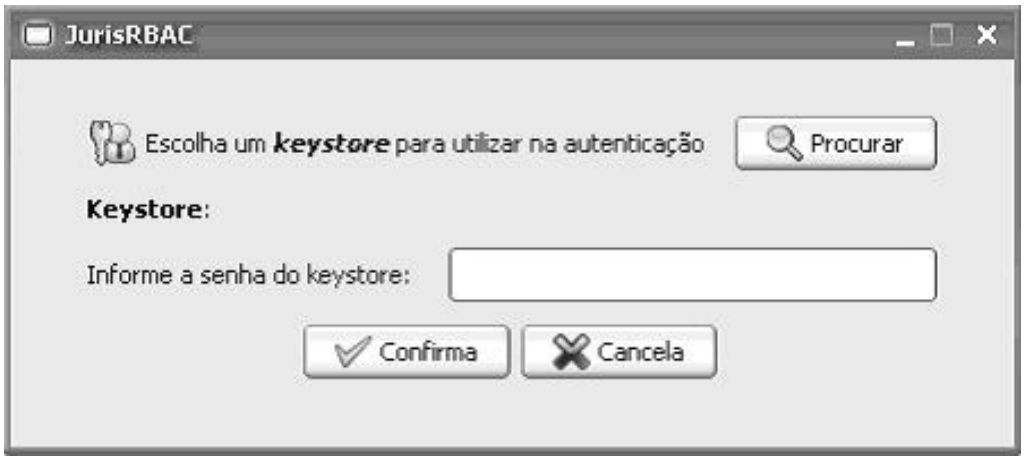

Figura 5.4: Tela de Autenticação

A figura 5.5 ilustra como a aplicação utiliza o *keystore* para realizar a autenticação.

```
ByteArrayOutputStream byteStream = new ByteArrayOutputStream(4096);
try
{
  keystore = KeyStore.getInstance(KeyStore.getDefaultType());
  FileInputStream file = new FileInputStream(ksPath);
  keystore.load(file,senha);
  file.close();

  Enumeration en = keystore.aliases();
  String alias = (String) en.nextElement();

  X509Certificate cert = (X509Certificate) keystore.getCertificate(alias);

  ObjectOutputStream outputStream = new ObjectOutputStream(
          new BufferedOutputStream(byteStream));
  outputStream.flush();
  outputStream.writeObject(cert);
  outputStream.flush();
}
catch(Exception e)
{
  e.printStackTrace();
}

gerenciadorAutorizacao = GerenciadorAutorizacaoSingleton.getInstancia();
PrincipalAuthenticator pa = gerenciadorAutorizacao.getPrincipalAuthenticator();

SecAttribute[] attrs = {};
//Holder que retornará a credencial criada para o usuário
CredentialsHolder credHolder = new CredentialsHolder();
//Parametros de retorno para continuação da autenticação - não usados
OpaqueHolder contData = new OpaqueHolder();
OpaqueHolder authEspec = new OpaqueHolder();
//Faz a autenticação
AuthenticationStatus authStatus = pa.authenticate(X509_CERTIFICATE,          // Método de autenticação
                                                  "",                        // campo não utlizado
                                                  byteStream.toByteArray(),  // certificado
                                                  attrs,                     // Atributos de segurança fornecidos
                                                  credHolder,                // Retorna a credencial criada para
                                                                             // usuário em caso de sucesso
                                                  contData,                  // Valor de retorno não usado
                                                  authEspec);                // Valor de retorno não usado
```

Figura 5.5: Utilização do *keystore* para realizar a autenticação

De posse do caminho do arquivo que representa o *keystore* (variável `ksPath`), o *keystore* é carregado. A seguir, o certificado é obtido (`keystore.getCertificate()`), a partir do *alias* encontrado no *keystore* (assume-se que o *keystore* só contenha uma entrada). O objeto que representa o certificado é, então, convertido para um *stream* de bytes; essa conversão é necessária, pois, o método de autenticação `authenticate` da classe `PrincipalAuthenticator` especifica, em sua assinatura, que o valor utilizado para a autenticação seja passado como um vetor de bytes (veja o terceiro argumento da chamada `pa.authenticate()`).

Após a chamada à `authenticate()`, deve-se verificar o retorno do mesmo. Os retornos verificados são `AuthenticationStatus._SecAuthSuccess` e `AuthenticationStatus._SecAuthFailure`. O primeiro valor indica que a autenticação foi realizada com sucesso, enquanto o segundo indica a falha na autenticação. A figura 5.6 apresenta essa verificação.

Se a autenticação falhar, é lançada uma exceção indicando a falha. Se o MACA retornar que a autenticação foi realizada com sucesso, é realizado um segundo passo de autenticação, através da chamada ao método `performTwoWaySSL()`. Essa chamada estabelece uma espécie de desafio, para provar que o certificado foi apresentado por quem se encontra de posse da chave privada correspondente (ou seja, como sempre, o protocolo presume que a chave privada associada ao certificado é de controle exclusivo do seu titular). Esse desafio é realizado por meio do estabelecimento de uma conexão SSL (*Secure Socket Layer*) bidirecional, isto é, onde as duas partes se autenticam entre si; assim, além da forma comumente utilizada na autenticação SSL, onde o servidor (neste caso o MACA) apresenta o seu certificado, exige-se que o cliente também apresente um certificado, como forma de se identificar perante o servidor, além da usual autenticação do

```
// Verifica o status da autenticação
switch(authStatus.value())
{
  case AuthenticationStatus._SecAuthSuccess :
  {
    try
    {
      // realiza um desafio
      this.performTwoWaySSL();

      // Recupera mensagem sobre a autenticação, por exemplo,
      // a que avisa que a senha irá expirar numa determinada data.
      // Recomenda-se sempre mostrar esta mensagem para o usuário
      String authMessage = new String(authEspec.value);
      System.out.println(authMessage);

      // Recupera credencial criada para esta sessão do usuário e
      // registra a mesma junto ao gerenciador de autorizacao
      creds = credHolder.value;
      gerenciadorAutorizacao.setCredentials(credHolder.value);
    }
    catch(Exception e)
    {
      throw new JurisRBACException(e.getMessage());
    }

    break;
  }

  case AuthenticationStatus._SecAuthFailure :
  {
    throw new JurisRBACException("Falha na autenticação!");
  }

}
```

Figura 5.6: Verificação do retorno do método `authenticate()`

servidor perante o cliente.

O certificado utilizado neste passo pelo usuário é o mesmo que foi apresentado anteriormente na chamada à `authenticate` de `PrincipalAuthenticator`. Para implementação da comunicação SSL, foi utilizada a API JSSE — *Java Secure Socket Extension* (`http://java.sun.com/j2se/1.5.0/docs/guide/security/jsse/JSSERefGuide.html`). A figura 5.7 ilustra o lado do cliente nesse desafio.

Se nenhuma exceção for lançada durante a execução do código acima, então a autenticação (bidirecional) foi completada com sucesso.

Do lado do MACA, conforme já dito, foi criado mais um *flag* de autenticação (`X509_CERTIFICATE`), para poder realizar a autenticação via certificados digitais. Foram, então, realizadas modificações nas seguintes classes do MACA:

- `PrincipalAuthenticator` : classe responsável por realizar a autenticação; foi nessa classe que acrescentou o código de autenticação via certificados digitais.
- `User` : essa classe mantém os dados de um usuário autenticado no MACA; é nela que se realiza o `BIND` junto ao OpenLDAP; ela foi modificada para que se utilizasse o mecanismo SASL-EXTERNAL de autenticação junto ao LDAP; o método SASL-EXTERNAL, dentre outros

```
String trustStore = System.getProperty("jurisrbac.ssl.trustStore");
System.setProperty("javax.net.ssl.trustStore", trustStore);
System.setProperty("javax.net.ssl.trustStorePassword", System.getProperty("jurisrbac.ssl.trustStorePwd"));

KeyManagerFactory kmf = KeyManagerFactory.getInstance("SunX509");
kmf.init(keystore, senha);

SSLContext sslCtx = SSLContext.getInstance("TLS");
sslCtx.init(kmf.getKeyManagers(), null, null);
SSLSocketFactory ssf = sslCtx.getSocketFactory();

int porta = Integer.parseInt(System.getProperty("jurisrbac.maca_sa.sslPort"));
SSLSocket socket = (SSLSocket) ssf.createSocket(gerenciadorAutorizacao.getLDAPServer(), porta);

BufferedWriter out = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()));
BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

out.write("PERFORM 2-WAY SSL\n");
out.flush();


String line;
StringBuffer sb = new StringBuffer();

while((line = in.readLine()) != null)
  sb.append(line);
```

Figura 5.7: Implementação do desafio do protocolo SSL no cliente

mecanismos e funcionalidades, permite que seja utilizada a autenticação via certificado digital entre um servidor e um cliente LDAP.

Além disso, foi adicionada a classe `SSLSimpleServer`, que permite a realização do desafio da segunda etapa da autenticação, conforme já explicado anteriormente.

A figura 5.8 apresenta o trecho de código que foi acrescentado à classe `PrincipalAuthenticator` do MACA, onde é recebido o certificado para realização da autenticação.

```
// le o certificado recebido
ByteArrayInputStream byteStream = new ByteArrayInputStream(auth_data);
ObjectInputStream ois = new ObjectInputStream(new BufferedInputStream(byteStream));
X509Certificate certificate = (X509Certificate) ois.readObject();
ois.close();

String userRole = this.getUserRole(privileges);
UserHolder _userHolder = null;
User       _user = null;
synchronized (this) {
    _userHolder = (UserHolder)maca_cs.AuthenticatorServer.users.get(certificate.getSubjectX500Principal().getName());
    if (_userHolder != null)        //Usuário já possui sessões abertas, logo basta criar uma nova sessão (credencial)
       _user = _userHolder.getUser();
    else {  //Cria um novo usuário
        if ((AuthenticatorServer.MAX_USERS > 0) &&
          (AuthenticatorServer.users.size() >= AuthenticatorServer.MAX_USERS))
          throw new Exception("MACA_CS: ["+AuthenticatorServer.MAX_USERS+
              "] - número máximo de usuários simultâneos alcançado.");
        else {
          contexts.ContextExpressionInterpreter inter = null;
          try {
            inter = new contexts.ContextExpressionInterpreter(false);
          }
          catch (Exception except) {
            _log.println(java.util.logging.Level.SEVERE, security_name,
                         "maca_cs.PrincipalAuthenticator", "authenticate",
                         "Falha na criação do interpretador de contextos do usuário: "+except.getMessage());
          }
          _user       = this.createUser(_log,
                                        maca_cs.AuthenticatorServer.SERVER_AND_PORT,
                                        maca_cs.AuthenticatorServer.ROOT,
                                        certificate,
                                        inter);
          _userHolder = new UserHolder(_user);

          maca_cs.AuthenticatorServer.users.put(_user.getUserID(), _userHolder);
        }
    }
```

Figura 5.8: Modificações na classe `PrincipalAuthenticator`

Primeiramente, obtém-se o certificado; como o certificado é recebido como um vetor de bytes (conforme já explicado anteriormente), devemos ler esses bytes para obter o certificado Ú isso é feito nas quatro primeiras linhas mostradas. A seguir, foi seguido a mesma lógica que o MACA impõe quando da autenticação via nome de usuário senha:

1. verifica se o usuário já possui sessões abertas;
2. se tiver, o usuário é obtido (via `_userHolder.getUser()`);
3. caso contrário, é chamado o método `createUser()`, que criará um novo usuário.

5.2.3 Autenticação bidirecional MACA - LDAP

A figura 5.9 mostra o código da classe `User` que foi modificado; este trecho realiza o `BIND` no OpenLDAP, faz a pesquisa pelo usuário com base no *Distinguished Name* (DN) do certificado e, se possível (e necessário) for, cria um usuário.

```
// keystore do LDAP
String trustStore = "/usr/local/maca_cs/bin/ca_keystore";
System.setProperty("javax.net.ssl.trustStore", trustStore);
System.setProperty("javax.net.ssl.trustStorePassword", senha);

// acesso ao certificado do maca
System.setProperty("javax.net.ssl.keyStore", "/usr/local/maca_cs/bin/certificados/maca_cs_keystore");
System.setProperty("javax.net.ssl.keyStorePassword", senha_maca);

Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
env.put(Context.PROVIDER_URL,"ldap://"+serverAndPort+"/");

try
{
    // Cria o contexto inicial
    LdapContext ldapCtx = new InitialLdapContext(env, null);

    // Inicia o TLS
    StartTlsResponse tls = (StartTlsResponse) ldapCtx.extendedOperation(new StartTlsRequest());
    tls.negotiate();

    // realiza a autenticacao utilizando a credencial TLS
    ldapCtx.addToEnvironment(Context.SECURITY_AUTHENTICATION, "EXTERNAL");
    this.ctx = new DirContextHolder(ldapCtx);
    newUser = false;

    String subjectDN = certificate.getSubjectX500Principal().getName();

    //Cria o search controls
    SearchControls searchCtls = new SearchControls();

    // Define os atributos a serem retornados
    searchCtls.setReturningAttributes(UserSchema.getAttributesNames());

    // Define o escopo da pesquisa
    searchCtls.setSearchScope(SearchControls.SUBTREE_SCOPE);

    String searchFilter = "(&";
    StringTokenizer tokensDN = new StringTokenizer(subjectDN, ",");
    while(tokensDN.hasMoreElements())
    {
        searchFilter += "("+tokensDN.nextToken()+")";
    }

    searchFilter += ")";
    // Define a base para a pesquisa; neste caso, o usuario sobre localhost
    String searchBase = userRoot;

    // Search for objects using the filter
    NamingEnumeration answer = this.ctx.value.search(searchBase, searchFilter, searchCtls);

    if(answer.hasMoreElements())
    {
            SearchResult sr = (SearchResult)answer.next();
        attrs = sr.getAttributes();
        if(attrs != null)
          this._userRDN = "uid="+attrs.get("uid").get();
    }
    else // nenhum usuario encontrado
        throw new NamingException("Usuario "+subjectDN+" nao encontrado");


    ActivableRolesSetup();
}
catch(IOException e)
{
  e.printStackTrace();
}
```

Figura 5.9: Modificações na classe `User`

Observe como é realizada uma autenticação mútua (bidirecional) entre o servidor OpenLDAP e o MACA — as cinco primeiras linhas mostram o acesso ao certificado do OpenLDAP e ao certificado do MACA; essa autenticação é realizada via SASL-EXTERNAL via TLS (ambos gerenciam

internamente seus próprios *keystores*, com suas chaves privadas correspondentes). Os comandos

```
StartTlsResponse tls = (StartTlsResponse)
ldapCtx.extendedOperation(new StartTlsRequest());
tls.negotiate();
```

são os responsáveis por negociar a sessão TLS/SSL para a realização da autenticação. O comando

```
ldapCtx.addToEnviroment(Context.SECURITY_AUTHENTICATION,
“EXTERNAL”);
```

define que o método de autenticação a ser utilizado é o EXTERNAL. A seguir, por meio da instrução

```
String subjectDN =
certificate.getSubjectX500Principal().getName();,
```

é obtido o *Distinguished Name* (DN) do titular do certificado do usuário que se autenticou no MACA, e que precisa assumir papéis para sua sessão no sistema de aplicações; esse DN será utilizado como filtro de pesquisa junto ao OpenLDAP.

Se houver resultados na pesquisa realizada, existe uma entrada de usuário correspondente àquele certificado de usuário apresentado; caso contrário, é lançada uma exceção do tipo `NamingException`, indicando que não há usuário correspondente para aquele certificado.

Para que a autenticação mútua entre o OpenLDAP e o MACA funcionasse, foi necessária a adição de algumas diretivas no arquivo de configuração do OpenLDAP — `slapd.conf`. Essas diretivas são:

```
TLSCACertificateFile /etc/openldap/certificados/trust.pem
TLSCertificateFile /etc/openldap/certificados/ldap.cert
TLSCertificateKeyFile /etc/openldap/certificados/privkey.pem
TLSVerifyClient try
```

A primeira diretiva especifica o caminho do arquivo que contém os certificados confiáveis do OpenLDAP. A segunda especifica o caminho do certificado o OpenLDAP. A terceira é o caminho do arquivo com a chave privada do OpenLDAP (*keystore*). A última diretiva especifica se há a necessidade ou não de requisitar o certificado do cliente (com o valor `try`, especificamente, o OpenLDAP requisita um certificado, mas se nenhum for apresentado a sessão prossegue normalmente; no entanto, se o certificado apresentado for inválido, a sessão é imediatamente terminada). Vale ressaltar que, por conter valores sensíveis como o caminho do arquivo da chave privada, é altamente recomendável que a máquina que hospeda o servidor OpenLDAP esteja fisicamente protegida.

Após a realização desses passos, conforme explicado anteriormente, é realizada uma espécie de desafio. Nos passos apresentados até agora, foi realizada a autenticação do MACA junto ao OpenLDAP e a pesquisa do DN do usuário na base LDAP.

Esse desafio é um segundo passo da autenticação, em que há a autenticação mútua entre o usuário que deseja acessar o sistema de aplicações e o servidor de controle de acesso — o MACA. Para a realização desse passo, foi acrescentada, ao código do MACA, a classe `SSLSimpleServer`, cujas partes principais são mostradas na figura 5.10.

```
System.setProperty("javax.net.ssl.trustStore", trustStore);
System.setProperty("javax.net.ssl.trustStorePassword", trustStorePwd);

try
{
    KeyStore ks = KeyStore.getInstance("JKS");
    ks.load(new FileInputStream(ksPath), password);
    KeyManagerFactory kmf = KeyManagerFactory.getInstance("SunX509");
    kmf.init(ks, password);

    sslCtx = SSLContext.getInstance("TLS");
    sslCtx.init(kmf.getKeyManagers(), null, null);

    ServerSocketFactory ssf = sslCtx.getServerSocketFactory();
    serverSocket = (SSLServerSocket) ssf.createServerSocket(HTTPS_PORT);
    serverSocket.setNeedClientAuth(true);
}
catch(Exception e)
{
    e.printStackTrace();
}
```

Figura 5.10: Modificações no código do MACA para autenticação

A implementação foi realizada utilizando-se a API JSSE. Nesse trecho, é definido qual o *keystore* com os certificados confiáveis (duas primeiras linhas), o certificado que o MACA irá apresentar durante a autenticação, a criação do contexto SSL e do socket SSL.

Vale atentar para a seguinte chamada:

```
serverSocket.setNeedClientAuth(true).
```

Esta é a chamada que define que a autenticação será bidirecional, pois, obrigando que o cliente apresente o seu certificado. Se o certificado apresentado não for validado por uma cadeia ancorada em um certificado confiável ao LDAP, será lançada uma exceção indicando o erro. Nesse caso, o desafio irá falhar.

5.3 Autorizações

Após a realização com sucesso da autenticação, o usuário está apto para acessar o sistema de aplicações. A partir desse momento, entra em ação o mecanismo de controle de acesso baseado em papéis; toda e qualquer tentativa de acesso ao sistema é intermediada pelo MACA. Ao tentar realizar uma operação, como por exemplo a pesquisa por um processo, o sistema,

primeiramente, verifica se o usuário pode realizar aquela ação. Isto é, se nesse momento da sessão ele desempenha algum papel que lhe autoriza a executar tal ação. Para isso, são levados em conta o papel (ou papéis) que o usuário está desempenhando no momento, o recurso que ele deseja acessar e o tipo de acesso desejado (escrita, leitura, execução, etc). A figura 5.11 apresenta um exemplo típico de acesso no sistema, onde o usuário tenta acessar o serviço de pesquisa de processos.

```
if(cntrAutorizacao.autorizaAcesso("Serviço de Pesquisa;jurisrbac", "execução"))
{
  iFramePesquisaProcesso = new JInternalFrame("Pesquisa de Processos", true,
      true, true, true);
  cntrApresentacaoPesquisa.mostraTelaPesquisaProcesso(iFramePesquisaProcesso);
  iFramePesquisaProcesso.setSize(iFramePesquisaProcesso.getPreferredSize());
  desktopPrincipal.add(iFramePesquisaProcesso);
  iFramePesquisaProcesso.setVisible(true);
}
else
{
  JOptionPane.showMessageDialog(null, "Você não possui permissão para acessar o serviço de pesquisa",
      "Acesso Proibido", JOptionPane.ERROR_MESSAGE);
}
```

Figura 5.11: Exemplo de acesso no sistema

Antes de acessar o serviço, é necessária a verificação de acesso, para decidir se o acesso deve ou não ser liberado; isso é feito no teste da primeira linha do código mostrado, na qual é chamado o método `autorizaAcesso()` da controladora de autorização. Esse método retorna um valor booleano que indica a liberação ou a proibição do acesso; como parâmetros são passados o recurso a ser acessado – neste caso, ŞServiço de Pesquisa;jurisrbacŤ – e a forma de acesso – "execução".

A figura 5.12 mostra o método autorizaAcesso().

```
public boolean autorizaAcesso(String recurso, String operacao)
{
  boolean acesso_autorizado = false;

  gerenciadorAutorizacao = GerenciadorAutorizacaoSingleton.getInstancia();
  AccessDecision ado = gerenciadorAutorizacao.getAccessDecisionObject();

  // Verifica se a credencial e 'ado' foram criados
  if((creds != null) && (ado != null))
  {
    //Define a familia dos atributos de segurança
    ExtensibleFamily extFamily = new ExtensibleFamily((short) 0, (short) 1);
    //Define os tipos de atributos a recuperar. Role e AccessId neste caso
    AttributeType[] attrTypes = {new AttributeType(extFamily, Role.value),
                                 new AttributeType(extFamily, AccessId.value)};
    //Recupera os atributos de segurança das credenciais do usuário
    SecAttribute[] attrs = creds.get_attributes(attrTypes);
    //Define o componente com o nome do recurso e a sua categoria - 'Aplicacoes'
    ResourceNameComponent[] resNameCompts = {new ResourceNameComponent("Aplicacoes", recurso)};
    //Define o nome do recurso para o 'ado', composto pelo nome do servidor LDAP e pela lista
    //dos componentes do nome
    ResourceName resName = new ResourceName(gerenciadorAutorizacao.getLDAPServer(), resNameCompts);
    try
    {
      //Invoca o método de solicitaçao de autorizaçao de acesso
      //Passa como parâmetros o nome do recurso, a operação a realizar e os atributos
      //de segurança do usuário. Retorna um booleano indicando se o acesso está autorizado ou não
      acesso_autorizado = ado.access_allowed(resName, operacao, attrs);
    }
    //O método 'access_allowed' pode levantar esta exceçao quando da ocorrência de erros
    catch (org.omg.maca_cs.DfResourceAccessDecision.InternalError exc)
    {
      System.out.println(exc);
    }
  }
  else
  {
    System.out.println("Objeto credentials ou access decision nao foi criado; credentials "+creds+", ado "+ado);
    throw new RuntimeException();
  }

  return acesso_autorizado;
}
```

Figura 5.12: Código do método `autorizaAcesso()`

Para requisitar o acesso, utiliza-se um objeto `AccessDecision`; a seguir, recupera-se o papel atual do usuário por meio de `get_attributes()`

de `Credentials`. Por fim, a linha

```
acesso_autorizado = ado.access_allowed(resName, operacao,
                                       attrs)
```

verifica se o acesso ao recurso é permitido. São passados o recurso a ser acessado – `resName` –, a operação – nesse caso "execução" – e os atributos do usuário, dentre os quais se encontra o seu papel. Essa operação retorna um booleano que indica a permissão ou proibição do acesso e que é retornado para o chamador de `autorizaAcesso()`.

A figura 5.13 ilustra, de maneira geral, como é realizado uma operação dentro do sistema.

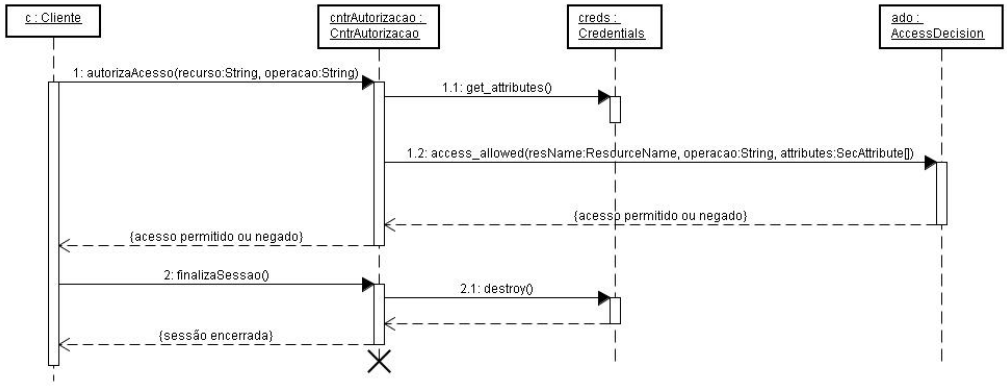

Figura 5.13: Diagrama de seqüência com as interações entre os objetos durante a requisição de acesso a um determinado recurso

Um cliente solicita a realização de uma operação sobre um determinado recurso. Para que a operação seja realizada, deve ser chamada a operação `autorizaAcesso()` da controladora de autorização, que decidirá se o acesso será permitido ou proibido.

A controladora de autorização acessa o objeto `Credentials` — que representa a sessão corrente do usuário — para obter os atributos — `get_attributes()` — do usuário (o principal atributo a ser considerado para a decisão de acesso é o papel do usuário).

De posse dos atributos, a controladora de autorização chama a operação `access_allowed()` do objeto `AccessDecision` (implementado pelo MACA) para decidir pela liberação de acesso; nesta chamada são passados o nome do recurso a ser acessado, a operação que o cliente deseja realizar e os atributos do usuário.

O retorno de `access_allowed()` é, então, repassado para o cliente, permitindo ou negando o acesso ao recurso. Quando o cliente deseja sair do programa, ele acessa o método `finalizaSessao()` da controladora de autorização; essa, por sua vez, realiza uma chamada ao método `destroy()` de `Credentials` que, efetivamente, finalizará a sessão corrente.

Capítulo 6

Desdobramentos

Aqui são apresentadas uma visão geral da aplicação desenvolvida e sugestão de trabalhos futuros.

6.1 JuRisBAC

Para testar o modelo sugerido, foi desenvolvida uma aplicação piloto — JuRisBAC — em código aberto que permite a movimentação processual segundo o fluxo de caminhamento adotado (figura 5.1). A aplicação foi desenvolvida utilizando a linguagem de programação Java, versão 1.5. Foram utilizadas, ainda, a biblioteca cliente do MACA (`http://maca.sourceforge.net/`) para a implementação do controle de acesso e a API *Log4j* (`http://logging.apache.org/log4j/docs/`) para a geração de *logs* de auditoria e para o acompanhamento das fases do processo judiciário. A aplicação foi desenvolvida segundo os termos da Licença Pública Geral Gnu (GPL).

A aplicação permite, basicamente, acompanhar todo o rito processual, desde a entrada de petição inicial (autuação) até o arquivamento do processo (conforme o fluxo da figura 5.1). Permite a entrada de documentos (petição/citação), a movimentação processual, a pesquisa por processos, o agendamento de audiências, a visualização de documentos, a criação de *logs* de auditoria e o acompanhamento dos atos processuais.

A seguir, encontram-se algumas telas da aplicação desenvolvida.

6.2 Documentações

Esta seção traz as referências para a documentação, binários e códigos desenvolvidos em nossa aplicação.

Os fontes e os binários do JuRisBAC encontram-se em `http://jurisrbac.codigolivre.org.br`.

Os fontes e os binários do MACA_CS, com as modificações que realizamos (introdução da autenticação via certificado digital), encontram-se `http://jurisrbac.codigolivre.org.br/`. No momento, o projeto de

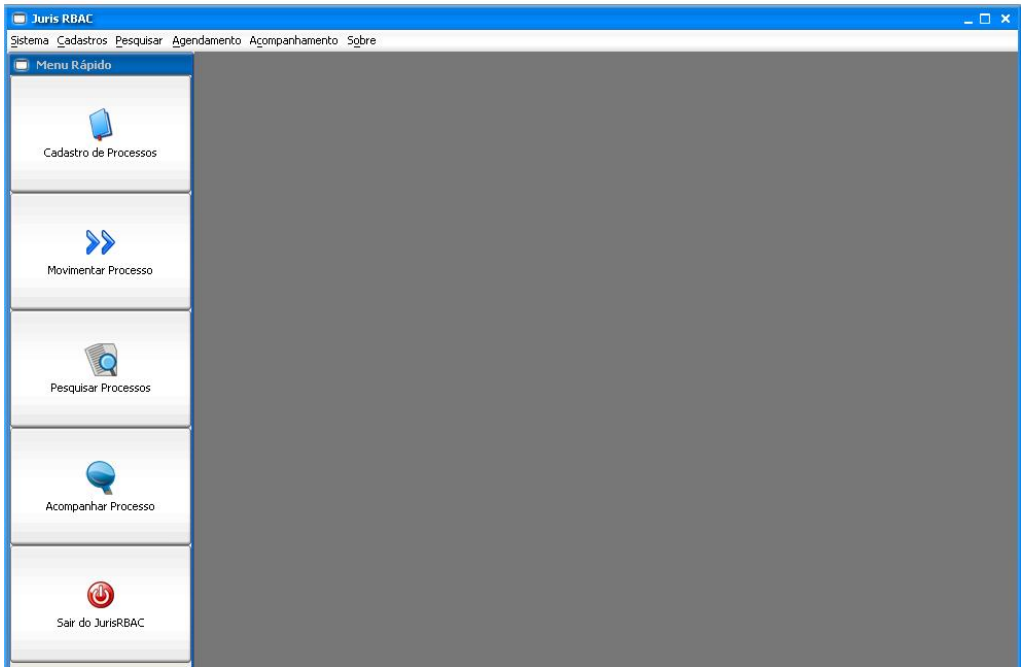

Figura 6.1: Tela Principal da Aplicação

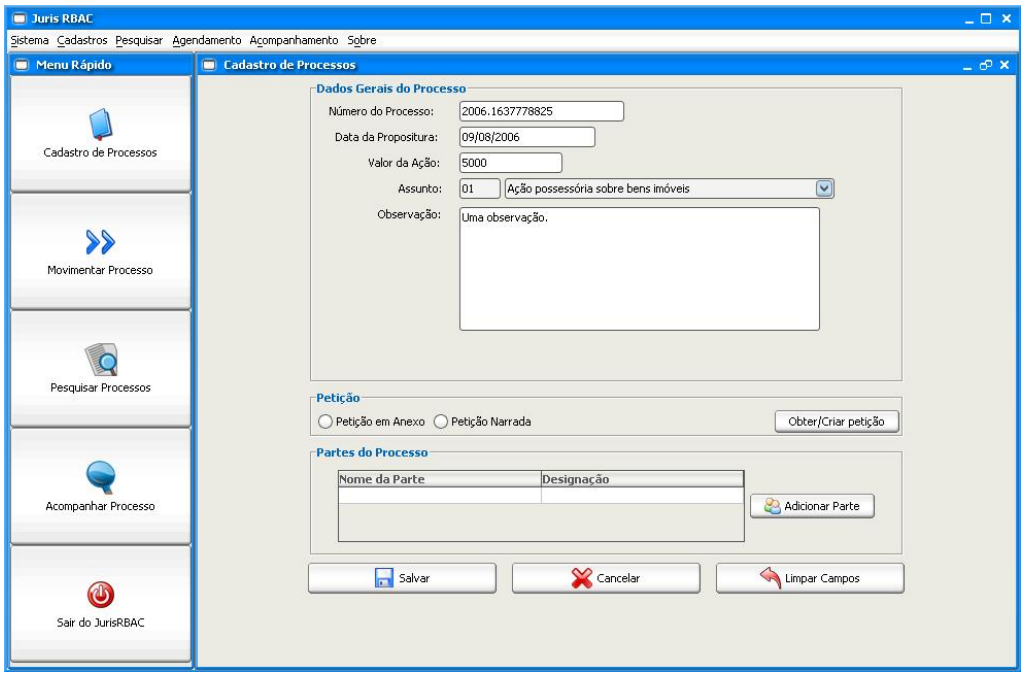

Figura 6.2: Tela de Cadastro de Processo

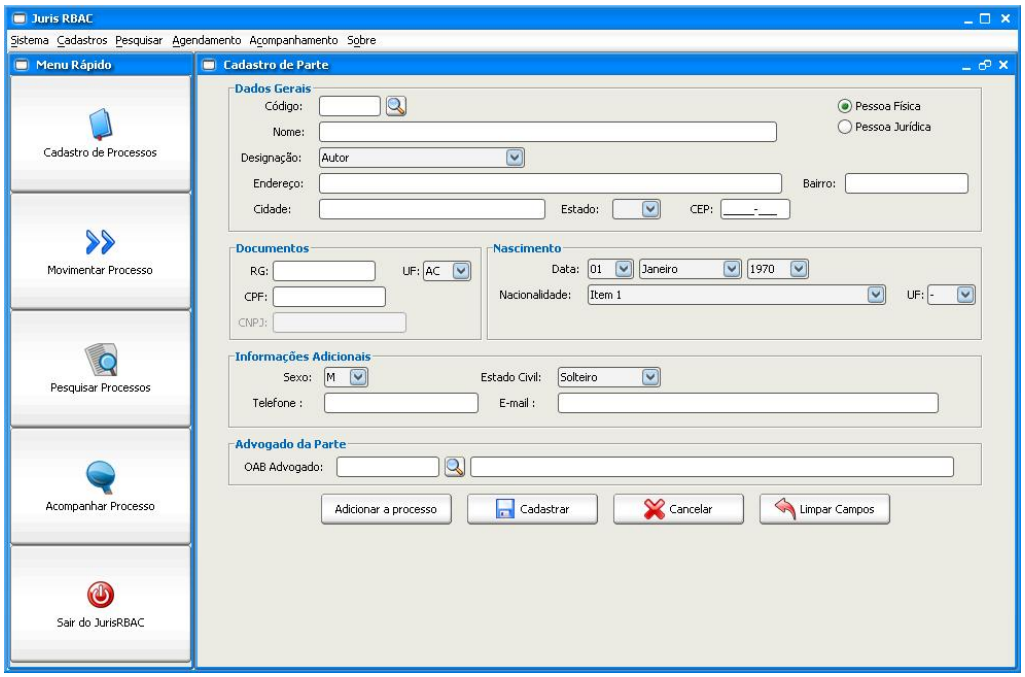

Figura 6.3: Tela de Cadastro de Parte

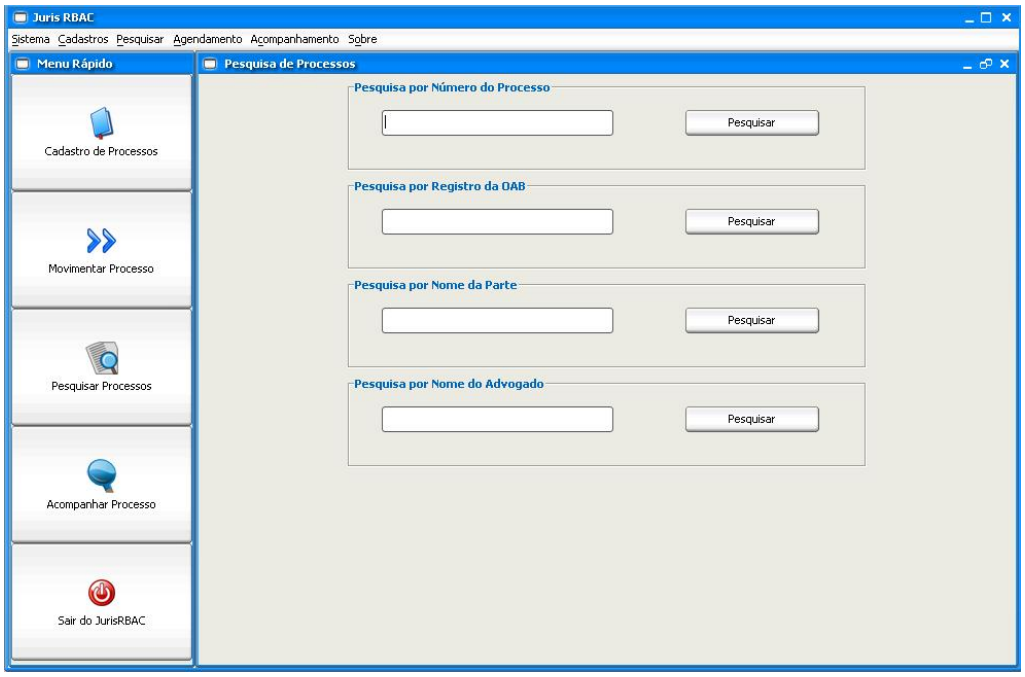

Figura 6.4: Tela de Pesquisa de Processo

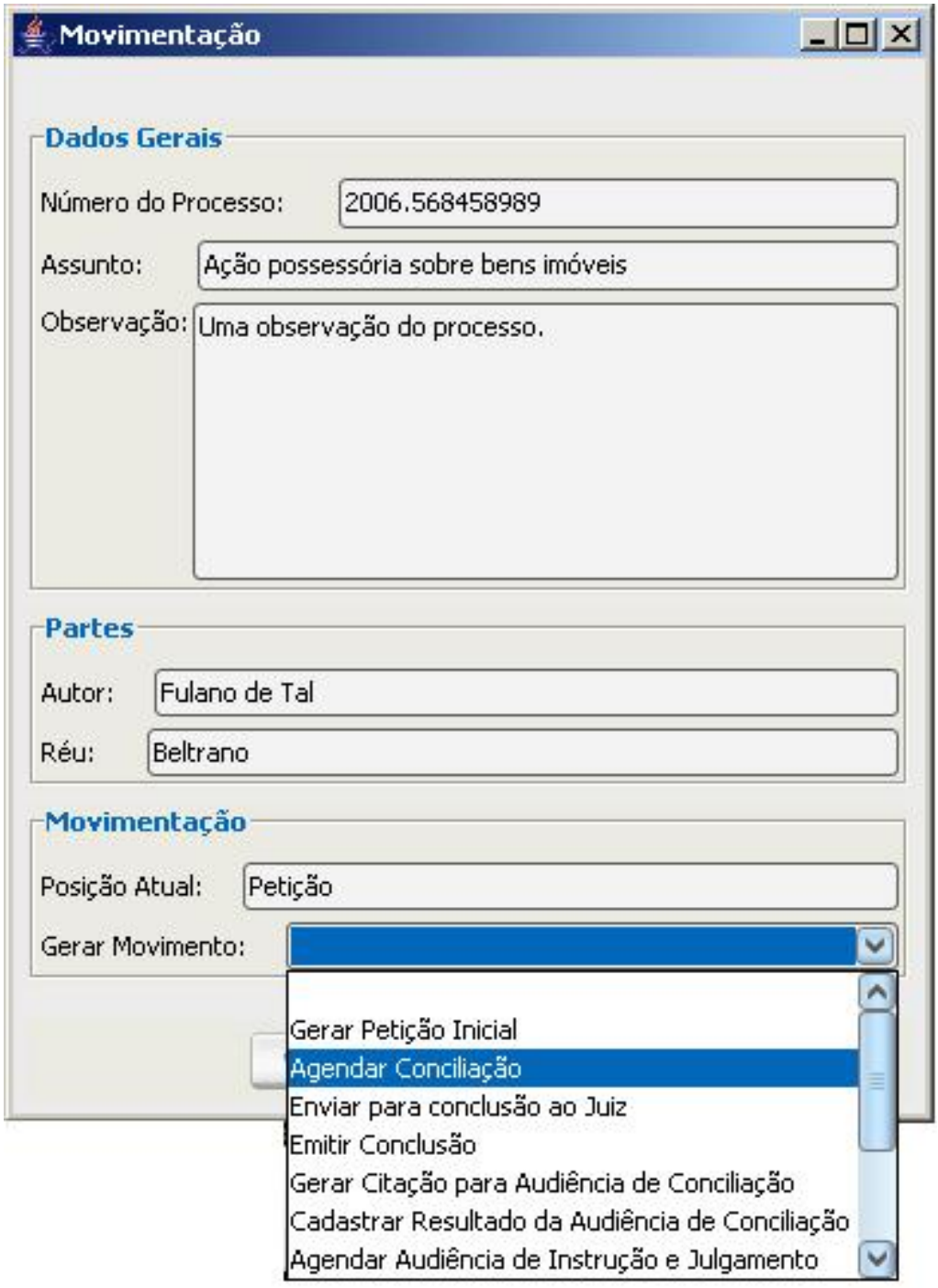

Figura 6.5: Tela de Movimentação de Processo

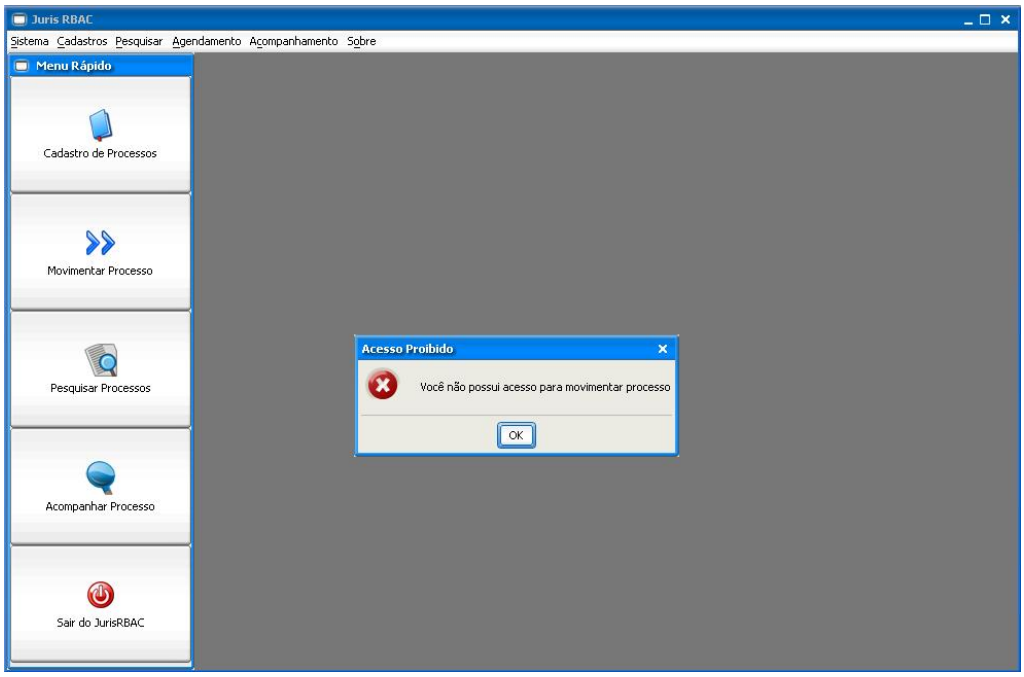

Figura 6.6: Tela proibindo acesso ao módulo de movimentação de processo

modificação do MACA_CS é um *fork* do projeto original do MACA, considerado a versão 3.2.2cx do MACA_CS, necessário para o funcionamento do JuRisBAC. Ambos projetos são gerenciados pelos autores deste trabalho.

A versão digital desse documento também se encontram em `http://jurisrbac.codigolivre.org.br/`.

Todos esses arquivos estão disponíveis na seção Downloads do *site*.

6.3 Trabalhos futuros

Na pesquisa realizada com este protótipo, surgem inúmeras possibilidades para trabalhos futuros que poderão ampliar as funcionalidades da arquitetura proposta, aumentar sua robustez e melhor validar os objetivos deste projeto.

Uma primeira necessidade seria no âmbito da Engenharia de Software, buscando uma forma para que a aplicação desenvolvida sobre a camada de controle de acesso se configure dinamicamente, evitando problemas posteriores de inconsistência entre regras existentes no servidor de controle de acesso e na aplicação. Uma proposta seria fazer com que o servidor de controle de acesso gerasse e atualizasse periodicamente — por exemplo, sempre que inicializado ou alterado — um arquivo de configurações. Este seria lido pela aplicação, que permitiria ao usuário autenticado acessar dinamicamente apenas aos recursos que os papéis assumidos naquela sessão no sistema lhe dariam acesso. Por exemplo, criando menus e botões em tempo de execução a partir das informações obtidas do arquivo de configurações e das credenciais apresentadas pelo servidor de autenticação, para aquela

sessão daquele usuário. Essa solução evitaria problemas de inconsistência entre a aplicação o e servidor de autenticação, evitando gargalos durante a implementação e manutenção de aplicações, bem como na gerência de políticas de segurança do sistema.

Ainda com relação à Engenharia de Software, considerando o modelo de desenvolvimento colaborativo para softwares abertos com licenças livres, seria necessário desenvolver uma metodologia para testes da aplicação, criar um padrão de codificação e documentar as interfaces de programação existentes.

Levando-se em conta os aspectos de segurança do sistema, seria preciso ainda pesquisar a melhor forma de proteger o banco de dados, tanto de acessos remotos como de acessos físicos. Não nos aprofundamos sobre o assunto neste protótipo pois o foco era a modelagem do controle de acesso na aplicação, usando criptografia forte (*2-way* SSL entre os servidores de aplicação, de controle de acesso e de diretórios LDAP). Como primeiro passo, propomos uma arquitetura onde a aplicação residiria em um servidor de aplicações, sendo possível o acesso remoto ao banco de dados somente por meio desse servidor, através de uma VPN, protegido contra vazamento e interceptação. Os clientes teriam acesso aos dados pela aplicação, que seria controlada pelo servidor de controle de acesso, não podendo acessar diretamente o banco de dados.

Além disso, seria interessante, ou talvez necessária, conforme a escala e/ou requisitos de segurança da aplicação, explorar as possibilidades de haver um mapeamento entre as permissões existentes no SGBD e as permissões criadas no servidor de controle de acesso, integrando a gerência do banco de dados à da política integrada de segurança para aquela implementação. Qual a forma mais elegante e segura de fazê-lo, dependeria do legado da arquitetura e da cultura gerencial do banco de dados a ser integrado. Não obstante, e talvez mais importante, essa abordagem oferece uma oportunidade para se considerar, de forma racional, a migração de banco de dados legado para, por exemplo, um regime de licenciamento condizente com a autonomia proporcionada pela solução de gerência de política de segurança através de *middleware* livre, aqui proposta.

Por fim, um trabalho futuro importante, no contexto atual da informatização do Judiciário em curso no Brasil, seria a utilização de campos no certificado apresentado pelo usuário no momento da autenticação para a filtragem dos papéis e permissões provenientes do servidor de controle de acesso. Durante a autenticação, a aplicação verificaria a existência de campos definidos pela autoridade certificadora que permitiria ou bloquearia a ativação de papéis vindos da camada de controle de acesso. Um exemplo seria, no caso de um usuário ser um advogado, o seu certificado conter o número de registro na OAB. No momento da autenticação, seria verificada a existência deste campo no certificado e, caso o servidor de controle de acesso contivesse uma ligação entre este usuário e o papel Juiz, a aplicação bloquearia a ativação deste papel. Ou vice-versa. A arquitetura aqui proposta possibilita esta funcionalidade, que seria de grande valia. Capaz, ao mesmo tempo, de diminuir sensivelmente o impacto dos riscos gêmeos,

num prestador jurisdicional informatizado, da excessiva responsabilidade na gerência da política de segurança e do abuso de poder concentrado na função administrativa de permissões e papéis, enquanto instrumenta tecnicamente esses prestadores para participarem ativamente na evolução da ICP a que estão subordinados, oferecendo-lhes uma plataforma adequada para implementarem, proporem, testarem e homologarem propostas normativas que incluem o conteúdo de certificados digitais ao negócio da prestação jurisdicional.

Capítulo 7

Conclusão

O presente trabalho preocupou-se em modelar o controle de acesso a um sistema de processos judiciais representados em meios digitais, baseado no rito processual do Juizado Especial Cível, utilizando o modelo RBAC. A adoção deste modelo em sistemas de grande porte procura facilitar e agilizar a administração de permissões de acesso a recursos, bem como a gerência de políticas de segurança do mesmo. Isto é possível através da racionalização das atribuições de permissões conjugado ao mapeamento direto dos papéis reais da organização, com suas políticas de acesso e de segurança correspondentes, para papéis da camada de controle de acesso, facilitando seu entendimento e sua gestão.

Como a autenticação tem papel primordial na aplicação do controle de acesso, houve a iniciativa de prover um mecanismo de autenticação com criptografia forte. A escolha recaiu sobre a autenticação via certificados digitais, por esta prova de conceito inserir-se no contexto de uma iniciativa, de âmbito nacional , para institucionalizar e popularizar o uso de uma Infraestrutura de Chaves Públicas — a ICP-Brasil. Esta seria responsável por fazer a ligação entre identidades de usuários com seus pares de chave, pública e privada, proporcionando assim a autenticação eletrônica em sistemas governamentais, bem como a assinatura digital em documentos oficiais. Ainda, o emprego deste modelo proporcionaria o mapeamento direto de regras no regime da ICP, relativas a prerrogativas neste domínio de aplicação, para regras de acesso a sistemas processuais por ele controlado. Como conseqüência, o uso de meios eletrônicos para a produção, armazenamento e disponibilização de documentos públicos, incluindo-se processos judiciais, poderia ser adotado amplamente, de forma eficiente e ágil em consonância com a evolução da ICP-Brasil.

Com esses objetivos, foi utilizado o MACA para a modelagem do controle de acesso RBAC. Como a versão atual desse *middleware* não possui um mecanismo de autenticação via certificados, e como o seu regime de licenciamento é livre, foi realizado uma modificação em sua estrutura a fim de possibilitar a autenticação bidirecional via criptografia assimétrica entre aplicações clientes e o MACA, e entre o MACA e o servidor LDAP (BIGS). Uniu-se, assim, as facilidades advindas da utilização do modelo RBAC com a robustez da autenticação de usuários perante o sistema via respectivos

certificados digitais.

Este trabalho de modificação de código já desenvolvido validou a versatilidade e utilidade do desenvolvimento colaborativo, possibilitado no regime de desenvolvimento de softwares de código aberto e com licenciamento livre, a exemplo do MACA. Este regime permite a adoção de mecanismos de segurança próprios, aumentando a autonomia de seus usuários em relação a fornecedores de componentes, qualidade essencial no âmbito da segurança. Verifica-se igualmente a tendência para que o MACA incorpore uma ampla gama de funcionalidades de segurança e para adoção e uso em diversos domínios de aplicação, com diferentes perfis de necessidades relativas a funções de segurança.

Finalmente, houve a implementação de um protótipo que contemplasse essas realidades, como prova de conceito e ponto de partida para novas pesquisas, que teriam como objetivo alcançar uma arquitetura robusta no domínio da segurança, já demonstrada no seu uso pelo Incor. No caso do domínio de aplicação para o qual foi implementada a prova de conceito, que foi objeto desse trabalho, no domínio da informatização do judiciário. Procuramos dar os primeiros passos para que, no futuro, esta arquitetura possa ser adotada no processo de informatização do Poder Judiciário no Brasil.

Referências Bibliográficas

Bell, D. E. and LaPadula, L. (1973). *Secure Computer Systems: Mathematical Foundations and Model*. The Mitre Corporation, Bedford, MA.

Clark, D. D. and Wilson, D. R. (1987). A comparison of commercial and military computer security policies. In *IEEE Symposium on Computer Security and Privacy*.

Ferraiolo, D. and Kuhn, D. R. (1992). Role-based access control. In *Proceedings of the NIST-NSA National (USA) Computer Security Conference*.

Ferraiolo, D., Sandhu, R., Gavrila, S., Kuhn, D., and Chandramouli, R. (2001). A proposed standard for role based access control. *ACM Transactions on Information and System Security*, 4(3).

Ferraiolo, D. F., Kuhn, D. R., and Chandramouli, R. (2003). *Role-Based Access Control*. Artech House, Norwood, MA.

Motta, G. H. M. B. (2003). Um modelo de autorização contextual para o controle de acesso ao prontuário eletrônico do paciente em ambientes abertos e distribuídos. Programa de Pós-graduação em Engenharia Elétrica.

Schneier, B. (1996). *Applied Criptography*. John Willey & Sons, Hoboken, NJ.