Controle de Acesso Baseado em Papéis para o Modelo CORBA de Segurança

Rafael Rodrigues Obelheiro[†], Joni da Silva Fraga e Carla Merkle Westphall
Email: {obelix, fraga, merkle}@lcmi.ufsc.br
Laboratório de Controle e Microinformática
Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina
Caixa Postal 476 – 88040-900 – Florianópolis – SC – Brasil

Resumo

Este artigo mostra como modelos de controle de acesso baseado em papéis (*role-based access control*—RBAC) podem ser implementados em sistemas de objetos distribuídos que seguem os padrões OMG/CORBA. Nós apresentamos uma nova abordagem que permite a ativação automática de papéis pelos componentes de segurança do *middleware*, o que traz o controle de acesso baseado em papéis para aplicações não cientes da segurança.
Palavras-chave: segurança, controle de acesso, RBAC, CORBA

Abstract

This paper shows how role-based access control (RBAC) models can be implemented in distributed object-based systems that follow OMG/CORBA standards. We introduce a novel approach that allows automatic role activation by the security components of the middleware, which brings role-based access control to security-unaware applications.
Key words: security, access control, RBAC, CORBA

1 Introdução

Hoje em dia, as organizações dependem cada vez mais de seus sistemas de informação. Essa dependência faz aumentar também a importância de manter a segurança da informação armazenada e gerenciada por estes sistemas. A crescente utilização de sistemas distribuídos de larga escala, notadamente aqueles baseados na Internet, introduz novos desafios à tarefa de garantir propriedades como confidencialidade, integridade e disponibilidade dos dados. Isto se deve a vários fatores, como a impossibilidade de prover segurança física a todos os componentes do sistema e a facilidade de acesso remoto, que tornam irrelevantes as fronteiras geográficas e também as leis que regulam o acesso e utilização destas informações.

Paralelamente, observa-se também uma disseminação de sistemas baseados na tecnologia OMG/CORBA (*Object Management Group/Common Object Request Broker Architecture*),[1] um padrão para sistemas abertos e distribuídos baseados em objetos que vem sendo adotado

[†]Bolsista CNPq.

[1]O OMG, que é um consórcio formado por mais de 800 empresas, é o organismo responsável pela formulação dos padrões da arquitetura CORBA.

mundialmente pela indústria de *software* [BD99]. O modelo de sistemas abertos no qual o CORBA está inserido é denominado OMA (*Object Management Architecture*).

Dentre os diversos serviços especificados pelo OMG para o ambiente CORBA destaca-se o serviço de segurança CORBA, também conhecido como CORBAsec. O modelo CORBA de segurança foi desenvolvido com o propósito de incorporar características e funcionalidades de segurança a sistemas de objetos distribuídos de pequena e larga escala, sem deixar de lado características-chave do padrão CORBA como transparência, interoperabilidade e portabilidade.

O controle de acesso baseado em papéis vem sendo reconhecido como uma alternativa aos tradicionais modelos de controle de acesso discricionário (baseado em matriz de acesso) e obrigatório (baseado em rótulos de segurança). Estudos conduzidos nos Estados Unidos pelo NIST (*National Institute of Standards and Technology*) no início da década de 90 [FGL92] mostram que boa parte das organizações deseja que o controle de acesso à informação seja feito segundo uma política centralizada e de forma flexível, possibilitando fácil adaptação a novos requisitos que surgem naturalmente com a evolução organizacional. Claramente, estes objetivos são difíceis de serem atingidos simultaneamente usando controle de acesso discricionário (flexível mas descentralizado) ou obrigatório baseado em rótulos (centralizado porém inflexível). O controle de acesso baseado em papéis, por sua vez, é capaz de satisfazer estes requisitos de maneira efetiva [SCFY96].

O presente trabalho mostra como integrar o controle de acesso baseado em papéis em sistemas distribuídos abertos que seguem os padrões OMG/CORBA. Para isso, foi desenvolvida uma estratégia que, a nosso conhecimento, é inédita na literatura: a ativação automática de papéis pelos componentes de segurança do *middleware*. As propostas conhecidas de implementação de RBAC exigem que os usuários ou as aplicações interajam com o subsistema de segurança para selecionar quais os papéis que deverão ser ativados no sistema. A nossa abordagem, porém, permite isolar usuários e aplicações destes pormenores, possibilitando que políticas de segurança sejam implantadas sem a necessidade de modificações nas aplicações existentes ou na interação entre usuários e aplicações.

Este artigo está organizado da seguinte maneira. A seção 2 descreve o modelo RBAC utilizado como referência. A seção 3 apresenta o modelo de segurança do CORBA. A seção 4 discute como o RBAC pode ser implementado dentro do modelo CORBAsec, e a seção 5 apresenta resultados obtidos com a implementação de um protótipo. Finalmente, a seção 6 mostra alguns trabalhos relacionados e a seção 7 apresenta nossas conclusões.

2 Controle de Acesso Baseado em Papéis—RBAC

O conceito de **controle de acesso baseado em papéis** (*role-based access control*—RBAC) surgiu com os primeiros sistemas computacionais multiusuários interativos, no início da década de 70. A idéia central do RBAC é que permissões de acesso são associadas a papéis, e estes papéis são associados a usuários. Papéis são criados de acordo com os diferentes cargos ou funções em uma organização, e os usuários são associados a papéis de acordo com as suas responsabilidades e qualificações. Os usuários podem ser facilmente remanejados de um papel para outro. Mudanças no ambiente computacional, como instalação de novos sistemas

e remoção de aplicações antigas, modificam apenas o conjunto de permissões atribuídas aos diferentes papéis, sem envolver diretamente o conjunto de usuários.

O modelo de referência utilizado neste trabalho é mostrado na figura 1. Este modelo corresponde ao RBAC Simétrico da família de modelos RBAC-NIST [SFK00].

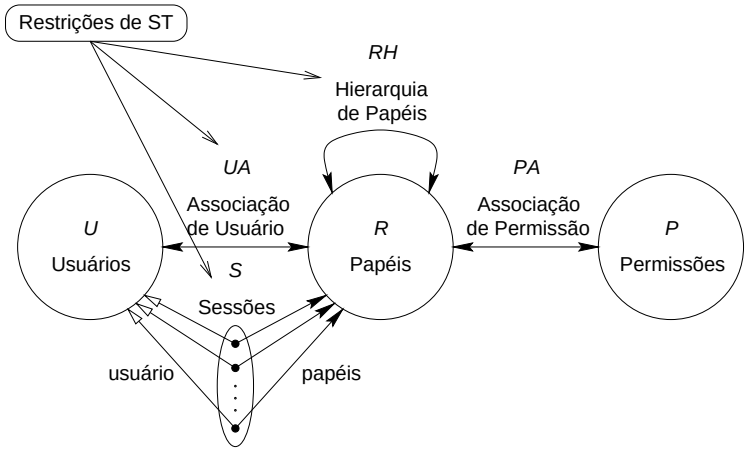

Figura 1: Modelo de Referência RBAC Simétrico

O modelo de referência mostrado na figura 1 possui quatro conjuntos de entidades: usuários (*U*), papéis (*R*, de *roles*), permissões (*P*) e sessões (*S*). Um usuário neste modelo é uma pessoa ou um processo agindo em nome dela. Um papel é uma função ou cargo dentro da organização e que possui uma semântica que representa a autoridade e a responsabilidade conferidas aos membros desse papel. Uma permissão é um direito específico de acesso a um ou mais objetos no sistema. O modelo RBAC-NIST não define permissões específicas; isso fica a critério dos implementadores [SFK00]. Uma sessão corresponde a um usuário acessando o sistema com um determinado conjunto de papéis ativos.

A associação usuário-papel (*UA*, de *user-role assignment*) e a associação permissão-papel (*PA*, de *permission-role assignment*) são relações do tipo muitos-para-muitos. O conjunto *U* de usuários tem um relacionamento um-para-muitos com o conjunto *S* de sessões (um usuário pode ter várias sessões), e o conjunto *S* tem um relacionamento muitos-para-muitos com o conjunto *P* de papéis (uma sessão tem vários papéis, e um papel pode estar ativo em várias sessões).[2]

Hierarquias de papéis, representadas na figura 1 pela relação *RH* (de *role hierarchy*), constituem uma forma natural de estruturar os papéis de forma a refletir as linhas de autoridade e responsabilidade em uma organização. Estas hierarquias são representadas matematicamente por relações de ordem parcial [SCFY96]. Um exemplo de hierarquias de papéis está mostrado na figura 2, onde os papéis superiores (*senior roles*) herdam as permissões dos papéis inferiores (*junior roles*).

Um conceito também suportado pelo modelo RBAC mostrado na figura 1 é a **separação**

[2]Relacionamentos muitos-para-muitos são representados na figura 1 por setas cheias, enquanto que relacionamentos um-para-muitos são representados por setas vazadas.

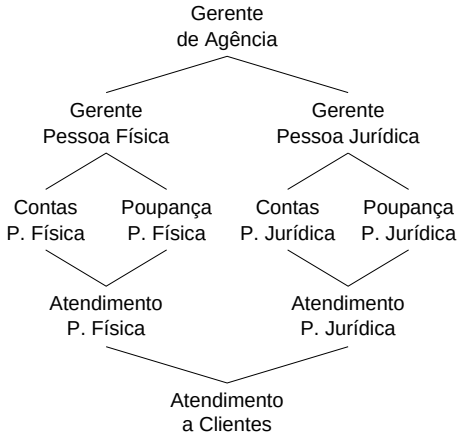

Figura 2: Exemplo de hierarquia de papéis

de tarefas (*separation of duty*), que é um princípio importante para minimizar a ocorrência de erros e fraudes na manipulação da informação. Este conceito consiste em dividir operações individuais em várias subtarefas menores que devem ser executadas por pessoas diferentes, reduzindo, desta maneira, o poder individual de cada usuário. A separação de tarefas teve a sua importância para a segurança da informação reconhecida e discutida em detalhes por Clark e Wilson [CW87]. O RBAC facilita a implantação de separação de tarefas, utilizando-se, para isso, de relações de exclusão mútua entre papéis.

Existem duas formas de separação de tarefas, estática e dinâmica. Na **separação estática de tarefas**, dois papéis R1 e R2 que são mutuamente exclusivos não podem ter usuários em comum; em outras palavras, um mesmo usuário não pode ser associado a R1 e a R2. Por outro lado, na **separação dinâmica de tarefas** uma exclusão mútua entre dois papéis R1 e R2 significa que um usuário pode ser associado a ambos, desde que apenas um deles (R1 **ou** R2) esteja ativo em um dado momento [SFK00].

3 Modelo CORBA de Segurança

O OMG elaborou um modelo de referência para segurança em sistemas de objetos distribuídos que seguem a arquitetura CORBA [OMG99]. A especificação de segurança CORBA define um conjunto de objetos e os relacionamentos entre esses objetos em um modelo capaz de fornecer funcionalidades como identificação e autenticação de principais, controle de acesso, comunicação segura entre objetos, não-repudiação, auditoria e administração de segurança.

Segundo a especificação CORBAsec, o sistema seguro de objetos distribuídos compreende quatro níveis, mostrados na figura 3. O nível de aplicação contém os objetos de aplicação (cliente e servidor). O nível de *middleware* é composto pelos serviços ORB, pelos objetos de serviço COSS (*Common Object Service Specification*) e pelo núcleo do ORB. Os serviços ORB e os objetos de serviço COSS são construídos sobre o núcleo do ORB e estendem as funções básicas com características adicionais, implementando a segurança de objetos distribuídos no nível de *middleware*. O nível de tecnologia de segurança corresponde aos serviços subjacentes

de segurança, que definem os protocolos utilizados na implementação de funcionalidades como confidencialidade e integridade na comunicação cliente-servidor. O nível inferior é o nível de proteção básica, que compreende o sistema operacional e o *hardware* subjacentes.

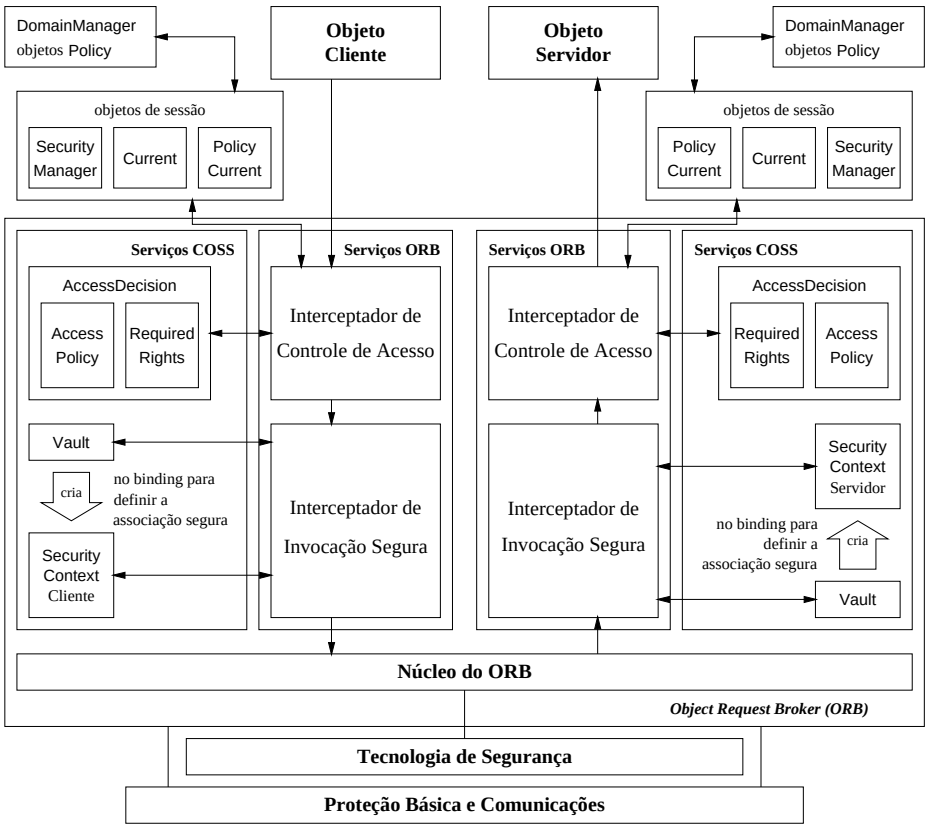

Figura 3: Modelo de Segurança CORBA

A especificação CORBAsec define um conjunto de objetos de serviço que implementam os controles de segurança em um sistema CORBA seguro: PrincipalAuthenticator, Credentials, AccessPolicy, RequiredRights, AccessDecision, SecurityManager, Current, PolicyCurrent, Vault e SecurityContext.

O modelo CORBA de segurança utiliza o conceito de **principal** para mediar as invocações de operações dos objetos. Um principal é o usuário ou entidade do sistema que é registrado e autêntico para o sistema de objetos distribuídos [OMG99]. O objeto PrincipalAuthenticator é o responsável pela autenticação dos principais na arquitetura CORBAsec. Esta autenticação tem como finalidade possibilitar a obtenção das credenciais do principal.

A **credencial** de um principal contém os seus atributos de identidade e de privilégio; estes atributos são usados para que o principal possa invocar objetos servidores no sistema. Em um sistema CORBA seguro, uma credencial é representada por um objeto Credentials. Normalmente, é utilizada uma credencial para cada mecanismo de segurança empregado por um sistema CORBA seguro. Esta abordagem é usada, por exemplo, no ORBAsec SL2, um ORB seguro que segue a especificação CORBAsec [Adi00]. O ORBAsec SL2 emprega, como mecanismos

de segurança, o Kerberos e o SSL, usando credenciais distintas para cada mecanismo.

Políticas de segurança são expressas em termos dos atributos de segurança de recursos do sistema (atributos de controle) e de principais (atributos de privilégio). As políticas de autorização são representadas na especificação CORBAsec pelo objeto AccessPolicy [OMG99]; um exemplo está mostrado na figura 4. Este objeto contém os direitos[3] concedidos aos principais para a invocação de operações no sistema CORBA seguro, com base nos seus atributos de privilégio.Apenas os direitos `g` (*get*), `s` (*set*), `m` (*manage*) e `u` (*use*)—que pertencem à família predefinida `corba`—são previstos na especificação CORBAsec, embora seja possível definir livremente outras famílias e tipos de direitos.

Atributo de Privilégio	Direitos Concedidos
`role: cliente`	`corba: gs--`
`role: cliente`	`corba: g---`
`role: gerente`	`corba: g-mu`
`role: gerente`	`corba: g--u`

Figura 4: Exemplo de AccessPolicy

Os direitos requeridos (atributos de controle) para a execução de operações em objetos servidores são armazenados no objeto RequiredRights. Conforme pode ser visto no exemplo da figura 5, os direitos requeridos são especificados por *interface* (uma classe no modelo CORBA), e não por instância (um objeto). Além disso, existe também um combinador, que indica se um principal precisa possuir todos os direitos requeridos (*All*) para executar uma operação ou se apenas um deles é suficiente (*Any*).

Direitos Requeridos	Combinador	Operação	Interface
`corba: g---`	All	`ver_saldo`	`ContaPFis`
`corba: g---`	All	`ver_saldo`	`ContaPJur`
`corba: -sm-`	Any	`abrir`	`ContaPFis`
`corba: g-m-`	All	`abrir`	`ContaPJur`

Figura 5: Exemplo de RequiredRights

O objeto AccessDecision (figura 3) é responsável por decidir se a invocação de uma operação de um determinado servidor deve ser permitida ou não. Esta decisão de acesso depende dos atributos de privilégio (representados pelo objeto AccessPolicy) e dos atributos de controle (representados pelo objeto RequiredRights). A lógica dessa decisão de acesso é deixada em aberto pela especificação CORBAsec, e é dependente do contexto do sistema CORBA seguro utilizado [BD99].

Os **objetos de sessão**—SecurityManager, PolicyCurrent e Current—armazenam informações sobre o contexto corrente de segurança, como referências aos objetos RequiredRights e AccessDecision (SecurityManager), referências aos objetos de política usados no estabelecimento de associações seguras (PolicyCurrent) e as credenciais do principal obtidas pelo servidor (Current). Ainda na figura 3, os objetos Vault e SecurityContext, por sua vez, são participantes no

[3]O conceito de **direitos** utilizado no modelo CORBA de segurança é equivalente ao conceito de **permissões** no modelo RBAC. Os dois termos são utilizados indistintamente neste trabalho.

estabelecimento de **associações seguras**, que garantem confidencialidade e/ou integridade às mensagens trocadas entre cliente e servidor.

A especificação CORBAsec define dois tipos de interceptadores[4] que atuam durante uma invocação de método. O primeiro é o **interceptador de controle de acesso**, que atua em nível mais alto, desviando a invocação para fazer o controle de acesso, e o segundo é o **interceptador de invocação segura**, que atua em nível mais baixo, fornecendo integridade e confidencialidade às mensagens trocadas entre cliente e servidor. Estes interceptadores são criados durante o *binding* entre cliente e servidor, e atuam de maneira diferente em momentos distintos de uma invocação de método. No *binding*, o interceptador de controle de acesso é responsável por instanciar o objeto AccessDecision e atualizar a sua referência no objeto SecurityManager. O objeto AccessDecision, por sua vez, deve disponibilizar o objeto de política AccessPolicy do domínio, inserindo sua referência no objeto PolicyCurrent, e também localizar o objeto RequiredRights, atualizando sua referência no objeto SecurityManager. Em tempo de decisão de acesso, o interceptador de controle de acesso invoca a operação `access_allowed` do objeto AccessDecision, que é responsável por autorizar ou não a execução da operação, obtendo os direitos concedidos (contidos em AccessPolicy) e comparando-os com os direitos requeridos (contidos em RequiredRights).

4 A Proposta RBAC-JACOWEB

O projeto JACOWEB, desenvolvido no LCMI–DAS–UFSC, visa investigar a problemática da autorização em sistemas distribuídos de larga escala. O foco do projeto é a definição e implementação de esquemas de autorização para aplicações distribuídas, utilizando como suporte o modelo CORBA de segurança integrado aos modelos de segurança Java e Web [WF99]. Na arquitetura JACOWEB, o controle de acesso é realizado pelo *middleware*, de maneira transparente, tanto no lado do cliente quanto no lado do servidor. Objetos de serviço localizados do lado do cliente verificam se o acesso solicitado está em conformidade com a política de segurança, e se ele deve ou não ser autorizado. Caso o acesso seja autorizado, é gerada uma *capability* que é agregada à requisição (*request*) CORBA e enviada ao servidor; objetos de serviço no lado do servidor extraem e validam a *capability* antes de liberar o acesso.

Este artigo introduz a proposta RBAC-JACOWEB, cujo objetivo é incorporar um controle de acesso baseado em papéis ao esquema de autorização JACOWEB. O modelo RBAC utilizado é o RBAC Simétrico do padrão RBAC-NIST, apresentado na seção 2. A idéia é estender o serviço de políticas PoliCap definido em [WFW00] de modo que este gerencie a configuração RBAC em um contexto CORBAsec.

[4]No modelo CORBA de segurança, os serviços ORB são implementados por **interceptadores**, que são objetos que são logicamente interpostos na seqüência de invocação de um cliente a um servidor. Cada serviço COSS relacionado com a segurança é associado a um interceptador, que tem a finalidade de desviar a chamada de maneira transparente, ativando assim um objeto de serviço associado.

4.1 O Serviço de Políticas PoliCap

O PoliCap é um serviço de políticas para objetos distribuídos cujas invocações são reguladas segundo o modelo CORBAsec [WFW00, Wes00]. O PoliCap foi desenvolvido no contexto do projeto JACOWEB e corresponde a um primeiro nível de verificação do controle de acesso no esquema de autorização.

O PoliCap possibilita o gerenciamento centralizado de objetos de política em um domínio de segurança de objetos distribuídos, suprindo a carência na especificação CORBAsec quanto ao gerenciamento de objetos de política. Segundo a especificação, as políticas de segurança são disponibilizadas através dos objetos AccessPolicy e RequiredRights. Aplicações administrativas interagem com o PoliCap para gerenciar estes objetos. Por sua vez, aplicações operacionais ou objetos COSS interagem com o serviço de política PoliCap para obter, no *binding*, as políticas e os direitos necessários aos controles em tempo de execução sobre as invocações de um método. A idéia é que, no *binding*, o serviço de política forneça versões locais dos objetos AccessPolicy e RequiredRights que contenham os atributos de privilégio e de controle que se aplicam à invocação. A decisão de acesso, então, é efetuada com base nestas instâncias locais dos objetos AccessPolicy e RequiredRights. Maiores detalhes sobre o PoliCap podem ser encontrados em [WFW00, Wes00].

4.2 Integrando Modelos RBAC ao CORBAsec

Na proposta RBAC-JACOWEB, cada principal tem apenas uma credencial (uma vez que o JACOWEB suporta apenas o SSL como tecnologia de segurança). Os papéis associados a um principal são representados, na sua credencial, por atributos de privilégio do tipo Role. Após a autenticação do principal, a credencial contém apenas o seu AccessId, que é um atributo de privilégio que representa a identidade do principal para fins de controle de acesso, e que pode ser extraído do certificado X.509 do cliente (usado no estabelecimento da associação segura SSL). Os papéis são agregados à credencial à medida em que forem ativados dentro do sistema. Portanto, a credencial de um principal representa, na proposta RBAC-JACOWEB, a sua identificação de acesso e os papéis ativos que ele possui no sistema.

A funcionalidade dos interceptadores de segurança descrita na seção 3 permanece a mesma. As modificações para incorporar o controle de acesso baseado em papéis são feitas no objeto AccessDecision, que verifica se os direitos concedidos ao principal permitem acesso à operação solicitada. Com a autorização do acesso, o interceptador de controle de acesso do cliente monta uma *capability* que será agregada à requisição CORBA.

O PoliCap contém todos os dados relativos às políticas de segurança do domínio, que incluem usuários, papéis, associações usuário-papel e papel-permissão, relações de hierarquia de papéis e restrições de separação de tarefas. O PoliCap realiza o controle de acesso baseado em papéis através da operação `role_access`, cuja *interface* IDL está mostrada na figura 6.

Com base nas credenciais do cliente e na operação invocada, o PoliCap decide se é possível autorizar o acesso ou não. Se o usuário possui um ou mais papéis que lhe conferem as permissões necessárias à invocação da operação e estes papéis podem ser ativados (i.e., esta ativação não viola nenhuma restrição), o acesso é autorizado, o que é indicado por um valor de retorno

```
boolean role_access(inout SecurityLevel2::CredentialsList cred_list,
                    in CORBA::Identifier operation_name,
                    in CORBA::Identifier target_interface_name,
                    inout SecurityAdmin::AccessPolicy local_ap);
```

Figura 6: *Interface* IDL da operação `role_access` do PoliCap

`true`. Caso não seja possível ativar papéis que confiram as permissões necessárias, a operação retorna `false`.

Quando o acesso é autorizado, as credenciais do principal são modificadas de forma a incorporar os novos papéis ativados, e o argumento `local_ap` (que representa o objeto AccessPolicy local) é modificado de forma a incorporar os novos direitos concedidos pela ativação dos papéis ao principal.

As restrições de separação estática de tarefas são verificadas pela operação do PoliCap que associa usuários a papéis: um usuário não pode ser associado a um papel que tenha uma restrição de separação estática em relação a outro com o qual ele já esteja associado. Por sua vez, as restrições de separação dinâmica são tratadas pela operação `role_access`. Um papel só pode ser ativado se não tiver restrições de separação dinâmica em relação a qualquer um dos papéis ativos (contidos na credencial). Dessa forma, garante-se que uma ativação automática de papéis não irá violar a política de segurança definida.

4.3 Dinâmica do Controle de Acesso usando Papéis

4.3.1 *Binding*

O *binding* ocorre na primeira invocação que o cliente faz para o servidor. Em primeiro lugar, o interceptador de controle de acesso utiliza a operação `role_access` do PoliCap para fazer uma verificação global da política, isto é, o PoliCap verifica se o principal possui direitos suficientes para executar a operação solicitada. Se os direitos forem suficientes, atualizam-se as credenciais do cliente e o seu objeto AccessPolicy local, e o *binding* prossegue com a obtenção dos direitos requeridos para a *interface* alvo e a subseqüente atualização do objeto RequiredRights local. Além disso, conforme mostrado na seção 3, o objeto AccessDecision é instanciado e a sua referência é atualizada no objeto SecurityManager. Em caso de insuficiência de direitos, o acesso é negado, a invocação é interrompida com o retorno de uma exceção para o cliente e o evento é registrado pelo serviço de auditoria do CORBAsec.

4.3.2 Decisão de Acesso

Em tempo de decisão de acesso, o interceptador de controle de acesso do lado do cliente invoca a operação `AccessDecision::access_allowed`, que verifica se os direitos concedidos ao principal lhe permitem executar a operação desejada. Caso o acesso seja autorizado, a seqüência de invocação prossegue normalmente, com a geração e envio da *capability*.

Por outro lado, em caso de insuficiência dos direitos concedidos, o objeto AccessDecision invoca a operação `role_access` do PoliCap para que esta verifique a possibilidade de ativação de novos papéis que confiram os direitos necessários à execução da operação solicitada. Caso o acesso seja autorizado com a ativação de um ou mais papéis, o interceptador de controle de acesso deve, então, gerar a *capability*, dando seqüência à invocação. Se a configuração do esquema RBAC não autoriza o acesso, a seqüência de invocação é interrompida com a negação do acesso e o evento é registrado pelo serviço de auditoria.

4.4 Exemplo

Consideremos um exemplo de funcionamento do controle de acesso baseado em papéis em uma aplicação bancária. O conjunto de papéis do sistema é $R = \{\mathsf{cli}, \mathsf{cxpf}, \mathsf{cxpj}, \mathsf{ger}\}$, representando, respectivamente, clientes, caixas para pessoas físicas, caixas para pessoas jurídicas e gerentes. A configuração do esquema RBAC, mantida pelo PoliCap, é mostrada na figura 7. Não há restrições de separação estática de papéis. As restrições de separação dinâmica de papéis são dadas pela figura 7(a), onde $(\surd)$ indica que a linha e a coluna representam papéis mutuamente exclusivos. A figura 7(b) representa o AccessPolicy. A figura 7(c) mostra a associação entre principais e papéis (conjunto *UA*). O RequiredRights é representado pela figura 7(d).

	cli	cxpf	cxpj	ger
cli	—	√	√	√
cxpf	√	—		√
cxpj	√		—	√
ger	√	√	√	—

(a) ST dinâmica

Atributo de Privilégio	Direitos Concedidos
`role: cli`	`corba: g---`
`role: cxpf`	`corba: gs--`
`role: cxpj`	`corba: g--u`
`role: ger`	`corba: g-m-`

(b) AccessPolicy

Principal	Papéis
`ana`	cli, cxpj, ger
`bia`	cli, cxpf, cxpj
`cris`	cli, cxpf

(c) Conjunto *UA*

Direitos Requeridos	Combinador	Operação	Interface
`corba: g---`	All	`ver_saldo`	`ContaPFis`
`corba: g---`	All	`ver_saldo`	`ContaPJur`
`corba: -s--`	All	`depositar`	`ContaPFis`
`corba: ---u`	All	`depositar`	`ContaPJur`
`corba: -sm-`	Any	`abrir`	`ContaPFis`
`corba: g-m-`	All	`abrir`	`ContaPJur`

(d) RequiredRights

Figura 7: Configuração RBAC gerenciada pelo PoliCap

A figura 8 mostra um cenário de funcionamento do controle baseado em papéis proposto, concentrando-se na evolução das ativações de papéis no sistema. No exemplo, o principal `bia` executa as operações da primeira coluna na seqüência mostrada na figura. As colunas *AR Antes*

e *AR Depois* representam o conjunto de papéis ativos (*active roles*)—armazenado na credencial do cliente—antes e depois da decisão de acesso para a operação correspondente.

Operação	***AR* Antes**	***AR* Depois**	**Comentários**
`ContaPFis::abrir`	$\varnothing$	{cxpf}	A operação requer um dos direitos `-sm-`, sendo que o papel cxpf confere os direitos `gs--`. Acesso permitido, com a ativação do papel cxpf.
`ContaPFis::depositar`	{cxpf}	{cxpf}	A operação requer o direito `-s--`, já conferido pelo papel cxpf. Acesso permitido.
`ContaPJur::depositar`	{cxpf}	{cxpf, cxpj}	A operação requer o direito `---u`, conferido pelo papel cxpj. Acesso permitido, com a ativação do papel cxpj.
`ContaPJur::abrir`	{cxpf, cxpj}	{cxpf, cxpj}	A operação requer os direitos `g-m-`, não conferido por nenhum papel do principal `bia`. **Acesso negado**.

Figura 8: Cenário de funcionamento para o principal `bia`

A seguir, é mostrado em detalhes como os objetos do modelo CORBA de segurança e o PoliCap interagem para realizar o controle de acesso, baseando-se no cenário apresentado na figura 8.

4.4.1 Execução do cenário da figura 8

Quando o sistema CORBA seguro é inicializado, o principal é autenticado através do objeto PrincipalAuthenticator. A sua credencial (objeto Credentials) contém apenas o seu AccessId, extraído de seu certificado SSL:

Tipo de Atributo	**Valor do Atributo**
AccessId	`bia`

▷ **Invocação da operação `ContaPFis::abrir`:**

Para que esta operação seja invocada, é necessário fazer o *binding* entre o cliente e o objeto servidor ContaPFis. De acordo com a seção 4.3.1, neste momento são obtidos, através do PoliCap, os direitos requeridos para a *interface* ContaPFis e os direitos concedidos para o principal que autorizam a invocação da operação `abrir`. Neste instante, também é instanciado o objeto AccessDecision, atualizando-se sua referência no SecurityManager.

As permissões necessárias para a invocação de `ContaPFis::abrir` são `corba: -s--` ou `corba: --m-`, já que o combinador é *Any*. Neste caso, o papel cxpf, por conceder as permissões `corba: gs--`, é então ativado no próprio *binding*, e o acesso é autorizado. O objeto Credentials do principal passa a ter o seguinte conteúdo:

Tipo de Atributo	**Valor do Atributo**
AccessId	`bia`
Role	cxpf

O objeto AccessPolicy local passa a ter o conteúdo abaixo:

Atributo de Privilégio	Direitos Concedidos
role: cxpf	corba: gs--

O objeto RequiredRights local, após o *binding*, é constituído por:

Direitos Requeridos	Combinador	Operação	Interface
corba: g---	All	ver_saldo	ContaPFis
corba: -s--	All	depositar	ContaPFis
corba: -sm-	Any	abrir	ContaPFis

▷ **Invocação da operação `ContaPFis::depositar`:**

Como o *binding* entre o cliente e o objeto servidor ContaPFis já está estabelecido, as verificações em relação à operação `ContaPFis::depositar` se resumem a efetuar o procedimento de decisão de acesso mostrado na seção 4.3.2. Para isso, o interceptador de controle de acesso invoca a operação `AccessDecision::access_allowed`. O direito requerido para a operação é `corba: -s--`, contido no objeto AccessPolicy local; desta forma, o acesso é autorizado, sem modificações nos objetos Credentials e AccessPolicy local.

▷ **Invocação da operação `ContaPJur::depositar`:**

Para a invocação desta operação é necessário estabelecer outro *binding*, desta vez entre o cliente e o objeto servidor ContaPJur. Novamente são obtidos os direitos requeridos (desta vez para a *interface* ContaPJur) e os direitos concedidos que autorizam a invocação da operação `depositar`.

O direito necessário à invocação de `ContaPJur::depositar` é `corba: ---u`, conferido pelo papel cxpj, que é ativado. O acesso é autorizado, e o objeto Credentials é modificado, passando a ter o seguinte conteúdo:

Tipo de Atributo	Valor do Atributo
AccessId	bia
Role	cxpf
Role	cxpj

O objeto AccessPolicy local também é atualizado:

Atributo de Privilégio	Direitos Concedidos
role: cxpf	corba: gs--
role: cxpj	corba: ---u

O objeto RequiredRights local passa a ser constituído por:

Direitos Requeridos	Combinador	Operação	Interface
corba: g---	All	ver_saldo	ContaPFis
corba: -s--	All	depositar	ContaPFis
corba: -sm-	Any	abrir	ContaPFis
corba: g---	All	ver_saldo	ContaPJur
corba: ---u	All	depositar	ContaPJur
corba: g-m-	All	abrir	ContaPJur

▷ **Invocação da operação `ContaPJur::abrir`**:

O *binding* entre o cliente e o objeto servidor ContaPJur já está estabelecido, passando-se direto para o procedimento de decisão de acesso. A operação `ContaPJur::abrir` requer os direitos `corba: g-m-`, que não estão presentes no objeto AccessPolicy local. Desta forma, a operação de decisão de acesso `AccessDecision::access_allowed` invoca a operação `role_access` para verificar se os direitos requeridos podem ser conferidos através da ativação de um papel. Todavia, o principal `bia` não possui nenhum papel que confira os direitos necessários, de modo que o acesso é negado, sem modificação das credenciais do cliente nem do objeto AccessPolicy local.

Cabe notar que os objetos AccessPolicy e RequiredRights locais são acessados a partir de referências armazenadas no objeto de sessão SecurityManager; estas referências são válidas para todas as ligações estabelecidas entre o cliente e os servidores durante a sessão, o que explica o fato dos objetos AccessPolicy e RequiredRights serem atualizados (e não instanciados novamente) durante a evolução do sistema.

5 Resultados de Implementação

Um protótipo de implementação da proposta RBAC-JACOWEB foi desenvolvido em nossos laboratórios. Para fins de teste e refinamento deste protótipo, usamos como exemplo um contexto de aplicações bancárias, constituída por dois objetos servidores CORBA e um *applet* cliente Java. Os servidores CORBA utilizam o JacORB [Bro97], um ORB Java livre, e o *applet* foi desenvolvido com o Java 2 SDK, versão 1.2.1. Os testes são realizados usando como navegador o Netscape Communicator 4.76. Essa implementação teve por objetivo investigar a eficácia do controle de acesso baseado em papéis proposto. A mesma arquitetura de sistema foi previamente utilizada para implementar, com sucesso, um esquema de autorização discricionário [WFW00]. As estruturas já desenvolvidas foram revisadas e adaptadas para servirem de base para a construção do controle de acesso baseado em papéis.

Os elementos-chave do protótipo são os interceptadores de controle de acesso apresentados na seção 3 e o serviço de políticas PoliCap aumentado com o suporte ao RBAC descrito na seção 4.2. Uma versão do objeto AccessDecision que interage com o PoliCap, crucial na proposta RBAC-JACOWEB, foi desenvolvida. As credenciais do cliente—contendo apenas o seu atributo de privilégio AccessId—são geradas na inicialização do sistema seguro com base no seu certificado SSL. Uma *interface* gráfica para a administração da configuração (figura 9) foi também implementada. Ela permite a gerência de usuários, papéis e permissões, além de hierarquias de papéis e restrições de separação de tarefas.

O cenário descrito na figura 8 foi usado como caso de teste. Foram implementados os objetos servidores ContaPFis e ContaPJur, assim como um *applet* que implementa a seqüência de operações mostrada na figura. O resultado do esquema de autorização baseado em papéis implementado concretizou o cenário da figura 8, demonstrando a viabilidade e a adequação da proposta RBAC-JACOWEB.

A versão atual do protótipo não conta, ainda, com autenticação de principais através do

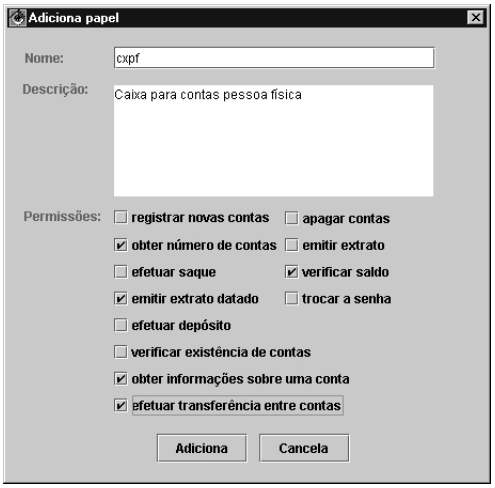

Figura 9: *Interface* gráfica de administração do PoliCap

objeto PrincipalAuthenticator. A credencial do cliente é inicializada com o AccessId extraído do seu certificado SSL, sem que este cliente passe por um processo de autenticação. A *interface* de gerência das políticas de segurança pode ser melhorada, incorporando elementos como um módulo de visualização gráfica da hierarquia de papéis do sistema (atualmente, é necessário manipular a lista de papéis inferiores a um determinado papel para modificar a hierarquia). O mecanismo de seleção de papéis embutido na operação `role_access` do PoliCap não é, ainda, o mais sofisticado possível; o algoritmo está sendo refinado para escolher melhor os papéis a serem ativados para uma dada operação, com vistas a minimizar as permissões adquiridas pelo usuário através da ativação.

6 Trabalhos Relacionados

Embora o conceito básico de papel venha sendo utilizado há décadas como um mecanismo para a gerência de permissões, modelos RBAC formais começaram a surgir há pouco tempo. O primeiro modelo RBAC formalizado na literatura foi o de Ferraiolo e Kuhn [FK92], do NIST. O trabalho de Sandhu *et. al.* [SCFY96] foi o primeiro a reconhecer a impossibilidade de capturar todas as nuances do RBAC em um único modelo, o que levou à definição da família RBAC96 de modelos. O modelo original do NIST também passou por revisões, e alguns de seus conceitos foram atualizados ao longo do tempo [FCK95, FBK99]. Todos estes modelos compartilham um núcleo comum de conceitos, mas cada um deles possui particularidades que os diferenciam. Entretanto, há visivelmente bem mais semelhanças do que diferenças. Isto levou o NIST e o grupo liderado por Sandhu a propor um modelo unificado que padronizasse os conceitos do RBAC, numa tentativa de estabelecer um consenso e um ponto de partida para novos desenvolvimentos. Este modelo, baseado largamente na família RBAC96 e no modelo NIST, ficou conhecido como modelo unificado RBAC-NIST [SFK00]. Embora alguns aspectos do modelo unificado tenham sido contestados [Jae00], a proposta foi bem recebida pela comunidade, e uma revisão

do modelo está atualmente sendo elaborada [Fer00].

O RBAC/Web [FBK99] é uma aplicação *intranet* em que o RBAC é utilizado como esquema de autorização para controlar o acesso a páginas de um servidor Web. O modelo RBAC utilizado como referência é o modelo NIST. O SSL não faz parte da aplicação em si, mas é considerado parte do seu contexto operacional. Os usuários no RBAC/Web correspondem a *logins* no servidor Web, e as transações HTTP que podem ser executadas pelos usuários (através dos seus papéis) nas páginas sujeitas ao RBAC representam as permissões. O próprio usuário é quem escolhe quais os papéis que serão ativados dentre aqueles aos quais ele está associado.

A proposta JRBAC-Web [Giu99] também tem por objetivo a segurança de servidores Web, mas difere do RBAC/Web por centrar-se no modelo de segurança Java. O RBAC é implementado como uma extensão do JAAS (*Java Authorization and Authentication Service*), e baseia-se em um modelo de referência próprio. Assim como no RBAC/Web, as permissões são representadas pelas transações HTTP, mas não há uma vinculação direta entre os usuários do modelo RBAC e entidades externas (como um *login*, por exemplo). Nesta proposta, as aplicações (que são *servlets* Java) é que são responsáveis pela ativação de papéis. As principais vantagens da proposta RBAC-JACOWEB em relação a estas experiências são a flexibilidade (é utilizável em qualquer aplicação CORBA), a transparência (os papéis são ativados automaticamente pelo sistema) e a conformidade com padrões (modelo RBAC-NIST e modelo de segurança CORBA).

Beznosov e Deng [BD99] definem um *framework* para implementar RBAC usando o serviço de segurança CORBA. As principais diferenças entre a nossa proposta e este *framework* são a transparência para as aplicações e usuários e o modelo RBAC utilizado como referência. Na proposta de Beznosov e Deng, o usuário interage, através de um UserSponsor, com o objeto PrincipalAuthenticator para selecionar os papéis ativados em uma sessão, e o modelo RBAC utilizado é a família RBAC96 [SCFY96]. Na proposta RBAC-JACOWEB, por outro lado, os papéis são ativados automaticamente pelo PoliCap quando necessários, de maneira transparente para o usuário, e o modelo RBAC de referência é o modelo RBAC-NIST [SFK00].

7 Conclusões

Este trabalho apresentou a proposta RBAC-JACOWEB, que mostra como o controle de acesso basedo em papéis pode ser incorporado a sistemas de objetos distribuídos que utilizam a tecnologia CORBA, valendo-se de padrões como o modelo de segurança CORBA e o modelo de referência RBAC-NIST. A nossa principal contribuição, contudo, é a introdução da ativação automática de papéis pelo subsistema de segurança, uma abordagem inovadora na literatura conhecida sobre RBAC.

O protótipo implementado, desenvolvido no contexto do projeto JACOWEB, enfatiza a viabilidade da proposta e a adequação do RBAC como um modelo de controle de acesso ao mesmo tempo flexível (na definição da política de segurança) e rigoroso (na implantação da política definida).

Um aspecto que será futuramente abordado dentro da proposta RBAC-JACOWEB é o uso do RBAC para a administração de segurança através da utilização de modelos RBAC administrativos como o descrito em [SM99]. Outras perspectivas incluem o acompanhamento da evolução

do modelo unificado RBAC-NIST e a adequação da proposta RBAC-JACOWEB às suas futuras revisões e a incorporação de melhorias à ferramenta de administração da configuração RBAC.

Referências

[Adi00] Adiron. *ORBAsec SL2 User Guide, version 2.1.4*. Syracuse, NY, July 2000.

[BD99] K. Beznosov and Y. Deng. A Framework for Implementing Role-Based Access Control Using CORBA Security Service. In *Proc. 4th ACM Workshop on RBAC*, pages 19–30, 1999.

[Bro97] G. Brose. JacORB—Design and Implementation of a Java ORB. In *Proc. DAIS'97*, pages 143–154, Sep. 1997.

[CW87] D. Clark and D. Wilson. A Comparison of Commercial and Military Computer Security Policies. In *Proc. IEEE Symp. Security and Privacy*, pages 184–194, 1987.

[FBK99] D. F. Ferraiolo, J. F. Barkley, and D. R. Kuhn. A Role-Based Access Control Model and Reference Implementation Within a Corporate Intranet. *ACM Trans. Information and Systems Security*, 2(1):34–64, Feb. 1999.

[FCK95] D. F. Ferraiolo, J. A. Cugini, and D. R. Kuhn. Role-Based Access Control (RBAC): Features and Motivations. In *Proc. 11th Annual Computer Security Conf.*, pages 241–248, Dec. 1995.

[Fer00] D. F. Ferraiolo. Re: Status of the NIST RBAC model. Comunicação privada, agosto de 2000.

[FGL92] D. F. Ferraiolo, D. M. Gilbert, and N. Lynch. Assessing Federal and Commercial Information Security Needs. NISTIR 4976, NIST, Nov. 1992.

[FK92] D. F. Ferraiolo and D. R. Kuhn. Role-Based Access Controls. In *Proc. 15th NIST-NCSC National Computer Security Conference*, pages 554–563, 1992.

[Giu99] L. Giuri. Role-Based Access Control on the Web Using Java. In *Proc. 4th ACM Workshop on RBAC*, pages 11–18, 1999.

[Jae00] T. Jaeger. Rebuttal to the NIST RBAC Model Proposal. In *Proc. 5th ACM Workshop on RBAC*, pages 65–66, 2000.

[OMG99] OMG. Security Service Specification, v1.7. OMG Doc. 99-12-02, Dec. 1999.

[SCFY96] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2):38–47, Feb. 1996.

[SFK00] R. S. Sandhu, D. F. Ferraiolo, and D. R. Kuhn. The NIST Model for Role-Based Access Control: Towards a Unified Standard. In *Proc. 5th ACM Workshop on RBAC*, pages 47–63, 2000.

[SM99] R. S. Sandhu and Q. Munawer. The ARBAC99 Model for Administration of Roles. In *Proc. 15th Annual Computer Security Application Conf.*, Dec. 1999.

[Wes00] C. M. Westphall. *Um Esquema de Autorização para a Segurança em Sistemas Distribuídos de Larga Escala*. Tese de doutorado, PGEEL–UFSC, dezembro de 2000.

[WF99] C. M. Westphall and J. S. Fraga. A Large-Scale System Authorization Scheme Proposal Integrating Java, CORBA and Web Security Models and a Discretionary Prototype. In *Proc. IEEE LANOMS'99*, pages 14–25, dezembro de 1999.

[WFW00] C. M. Westphall, J. S. Fraga, and M. S. Wangham. PoliCap—Um Serviço de Política para o Modelo CORBA de Segurança. In *Anais do 18º SBRC*, pages 355–370, maio de 2000.