MARCOS TORK SOUZA

CONTROLE DE ACESSO PARA SISTEMAS DISTRIBUÍDOS

São Paulo
2010

MARCOS TORK SOUZA

CONTROLE DE ACESSO PARA SISTEMAS DISTRIBUÍDOS

Dissertação apresentada à Escola
Politécnica da Universidade de São
Paulo para obtenção do título de
Mestre em Engenharia

Área de Concentração:
Engenharia de Computação

Orientadora:
Profa. Dra. Tereza Cristina Melo de
Brito Carvalho

São Paulo
2010

FICHA CATALOGRÁFICA

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 22 de dezembro de 2010.

Assinatura do autor _____

Assinatura do orientador _____

Souza, Marcos Tork

**Controle de Acesso para Sistemas Distribuídos / M. T. Souza – ed.rev. – São Paulo, 2010.
95 p.**

Dissertação (Mestrado) – Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1. Sistemas Distribuídos 2. Redes de Computadores 3. Segurança de Computadores I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II. t.

DEDICATÓRIA

Dedico este trabalho aos meus pais.

AGRADECIMENTOS

Primeiramente agradeço a Deus pela graça concedida a mim na forma de pessoas excepcionais que fazem parte da minha vida. Em especial minha esposa, Roberta Lia, a qual mesmo distante esteve sempre ao meu lado, me incentivando e participando dos sacrifícios necessários para a conclusão deste trabalho.

Minha orientadora, Profa. Dra. Tereza Cristina, pela confiança no meu trabalho, inestimável orientação durante as minhas pesquisas e apresentação de resultados, assim como em todas as questões acadêmicas surgidas durante a execução deste mestrado.

Meus pais, Edhisa Souza e Marcos Souza, que acreditaram no meu potencial e sempre me incentivaram a encontrar o meu próprio caminho, apoiando as minhas escolhas independente dos mais difíceis obstáculos.

Meus amigos Joelle Quaini, Diego Gallo e Flávio Urschei que trabalharam comigo no Laboratório de Arquitetura e Redes de Computadores da Universidade de São Paulo, e me ajudaram em todas as pesquisas e tarefas desempenhadas durante o mestrado.

Em especial aos amigos Marcio Silva, Fernanda Silva e Jorge Fragoso pelas imprescindíveis sugestões e correções que tanto enriqueceram este trabalho.

RESUMO

A implementação de arcabouços de controle de acesso para sistemas distribuídos é dificultada pelas características próprias desta classe de ambientes, demandando modificações tanto na arquitetura do arcabouço quanto no modelo de política de controle de acesso usualmente empregados em arquiteturas não distribuídas.

Este trabalho tenciona sanar ou mitigar estas dificuldades formalizando os requisitos desta classe de ambientes em duas frentes distintas (arquitetura e modelo de política de acesso) e analisando o impacto que uma exerce sobre a outra. Duas conclusões fundamentais são suportadas por esta análise: a necessidade do arcabouço ser construído na forma de um sistema distribuído, e que embora um modelo de política de fato possa ser escolhido, a especificação deste precisará ser modificada de forma a se adaptar às características específicas do ambiente.

O arcabouço DRBAC (Distributed Role Based Access Control) foi desenvolvido sobre uma arquitetura distribuída e aplica o modelo de política de controle de acesso baseado em papéis. A arquitetura foi obtida a partir da expansão da arquitetura de referência de ferramentas de controle de acesso e a especificação do modelo foi desenvolvida a partir da especificação padronizada pelo NIST (*National Institute of Standards and Technology*).

A validação do trabalho é levada a termo por meio de uma série de experimentos realizados sobre a implementação de uma prova de conceito deste arcabouço.

ABSTRACT

The creation of frameworks for access control in distributed systems is made difficult by this class of systems own characteristics, demanding changes in both the architecture of the framework and in the model of access control policy usually employed on non distributed systems.

This work aims to solve or at least mitigate these problems by formalizing these requirements in two different fronts (architecture and model of access control policy) and analyzing its mutual impacts. Two fundamental conclusions are supported by this analysis: the need for the framework to be built in the form of a distributed system, and that although a policy model can indeed be chosen, the specification of this should be modified to adapt the specific features of the environment.

The DRBAC (Distributed Role Based Access Control) framework is built following a distributed architecture model that applies the Role Based Access Control policy. The DRBAC architecture was obtained from the expansion of the reference architecture for an access control tool for a generic access control system and the DRBAC access policy model was adapted from the one standardized by NIST (*National Institute of Standards and Technology*).

The validation of this work is carried out through a series of experiments conducted on a proof of concept implementation of this framework.

SUMÁRIO

1	INTRODUÇÃO	11
1.1	Objetivo	12
1.2	Motivação	13
1.3	Metodologia	14
2	SEGURANÇA EM COMPUTAÇÃO	16
2.1	Segurança: Objetivo	16
2.1.1	Integridade	17
2.1.2	Disponibilidade	18
2.1.3	Confidencialidade	19
2.2	Segurança: Modelo e Processo	20
2.3	Considerações Finais	24
3	CONTROLE DE ACESSO	25
3.1	Autenticação	27
3.2	Autorização	30
3.3	Geração de trilhas para auditoria	37
3.4	Considerações Finais	39
4	CONTROLE DE ACESSO EM SISTEMAS DISTRIBUIDOS	40
4.1	Conceituação de Sistemas Distribuídos	40
4.2	Requisitos do Arcabouço de Controle de Acesso para Sistemas Distribuídos	43
4.2.1	Requisitos da Política de Controle de Acesso	43
4.2.2	Requisitos de Sistemas Distribuídos	45
4.3	Tecnologias Habilitantes	47
4.3.1	Python	47
4.3.2	Distributed Hash Table (DHT)	48
4.3.3	Lightweight Directory Access Protocol (LDAP)	48
4.4	Soluções Existentes	49
4.4.1	Servidor de Autorização	49
4.4.2	Agentes	50
4.4.3	Proxy	51
4.5	Considerações Finais	53
5	CONTROLE DE ACESSO DISTRIBUÍDO BASEADO EM PAPÉIS (DRBAC - DISTRIBUTED ROLE BASE ACCESS CONTROL)	54
5.1	Especificação do Sistema de Controle de Acesso Distribuído Baseado em Papeis	54
5.1.1	Arquitetura	55
5.1.2	Modelo de Controle de Acesso	57
5.1.3	Blocos Fundamentais do Arcabouço	58
5.2	Desenvolvimento do arcabouço DRBAC	62
5.2.1	Funções Administrativas	64
5.2.2	Funções do Sistema	67
5.3	Testes	71
5.3.1	Casos de uso	72
5.3.2	Testes de Desempenho	82
5.4	Mapeamento entre a Especificação do Sistema e a Especificação de Requisitos	85

5.5 Considerações Finais.....	87
6 CONSIDERAÇÕES FINAIS.....	88
6.1 Comparação.....	88
6.2 Análise Crítica.....	90
6.3 Trabalhos futuros.....	92

LISTA DE ILUSTRAÇÕES

Figura 1 – Conceitos e Relações em Segurança. Adaptado de [12].	21
Figura 2 - Políticas de Controle de Acesso. Adaptado de [18].	33
Figura 3 - Modelo RBAC. Adaptado de [22].	34
Figura 4 – Arquitetura de Controle de Acesso. Adaptado de [1].	55
Figura 5 – Arquitetura de Controle de Acesso para Sistemas Distribuídos.	56
Figura 6 - Arquitetura DRBAC	62
Figura 7 – Gráfico com um cliente e um servidor	83
Figura 8 – Gráfico com um servidor e dois clientes	84

1 INTRODUÇÃO

O valor de uma organização é descrito pelos recursos que esta possui, e.g., automóveis, prédios, documentos, móveis, logomarcas, produtos, etc., motivo pelo qual estes recursos precisam ser protegidos contra quaisquer ações que possam causar danos e/ou a sua perda.

Para que estes recursos sejam protegidos as organizações estipulam um conjunto de regras e requisitos, conhecido como Política de Segurança, que possuem o intuito de definir os procedimentos que irão garantir a segurança dos recursos. A Política de Segurança tende a ser um documento conciso e de alto nível, em geral a particularização de uma norma ou padrão de segurança.

Essa política deve conter a Política de Controle de Acesso, a qual, de fato, deverá descrever a relação entre pessoas e os recursos da empresa. Essa política, por sua vez, deve ser capaz de permitir que pessoas autorizadas utilizem os recursos, garantindo o funcionamento da organização, e prevenir que acessos não autorizados, que podem originar danos e/ou a perda de recursos, aconteçam. Para garantir que a política seja respeitada, as organizações fazem uso de ferramentas que exercem coercitivamente estas regras.

Essas políticas devem ser obedecidas em toda a extensão da organização, independente do seu tamanho e da sua disposição (distribuição) geográfica ou lógica. É comum, por exemplo, que organizações multinacionais possuam inúmeras filiais caracterizadas por processos e recursos singulares à sua localização que, no entanto, precisam acatar as regras estabelecidas pelas políticas da organização.

A fim de consolidar a Política de Controle de Acesso, a qual tende a ser mais específica e, por conseguinte, mais afetada pela multiplicidade inerente a ambientes distribuídos, será proposta um arcabouço de controle de acesso que permite a centralização da política, mas que ainda assim garante a sua aplicação coercitiva sobre qualquer componente do sistema computacional distribuído.

1.1 Objetivo

O objetivo desta dissertação é apresentar a especificação, implementação e os resultados de testes de um arcabouço de controle de acesso a recursos para sistemas computacionais distribuídos que utilizará a Política de Controle de Acesso baseada em Papeis. Este arcabouço será denominado *Distributed Role Based Access Control* (DRBAC).

O DRBAC deve ser capaz de atender a requisições que intentem averiguar a legalidade de operações independente da sua localização no sistema computacional distribuído. Além disso, deve ser capaz de responder de maneira adequada ao aumento da demanda imposta ao sistema devido ao crescimento de seus usuários e serviços providos.

Tendo em vista o objetivo de atender a um sistema distribuído, é imperativo ao DRBAC ser capaz de representar o maior número possível de operações, objetos, usuários, e suas relações. Deste modo, a política de controle de acesso pode ser estipulada para todas as entidades independentemente de suas peculiaridades. A base de dados que armazena as políticas de controle de acesso deve permitir a consolidação desta política por toda a extensão do sistema distribuído. Contudo, esta base deve ser implementada de forma a atender aos requisitos de um sistema distribuído.

O DRBAC não pretende atender a requisitos de sistemas distribuídos formados por coalizão (mais de uma entidade administradora), onde o nível de confiança dos usuários deve variar de acordo com a organização a qual ele pertence, ou responder a problemas oriundos de conflito de políticas de segurança entre diferentes organizações ou qualquer outro problema originado deste tipo de ambiente.

Embora algumas premissas sejam tomadas como verdadeiras em relação aos serviços de autenticação (*Authentication*) e geração de trilhas para auditorias (*Auditing*), este trabalho não tenciona especificar uma solução para estes serviços.

Contudo, durante a fase de desenvolvimento, uma solução será apontada para a base de dados compartilhada entre o serviço de autorização e o serviço de autenticação.

Da mesma forma, a segurança na comunicação entre os elementos da arquitetura não é objeto deste trabalho, mas pressupõe-se que possa ser atingida, em grande parte, por meio da utilização de mensagens criptografadas.

1.2 Motivação

Ferramentas de controle de acesso foram implementadas de inúmeras formas, e.g., *firewalls* e serviços de autorização, e em diferentes lugares, e.g., redes de computadores, base de dados, sistemas operacionais e aplicações em geral. Essas ferramentas usualmente funcionam de forma eficaz e possibilitam que os recursos sejam acessados segundo a política especificada.

Porém, o mesmo não acontece em ambientes distribuídos devido às peculiaridades inerentes a estes ambientes. Como exemplo é possível citar o requisito de alta granularidade das autorizações, o qual é imperativo para que as políticas de controle de acesso sejam específicas o suficiente para proteger todos os recursos [01].

Há, ainda, as necessidades estruturais como: a centralização de políticas, com o intuito de possibilitar a consolidação das regras utilizadas por todos os elementos do sistema; o armazenamento e a administração de um grande número de usuários e recursos, característica usualmente encontrada em sistemas distribuídos; e o atendimento de requisições de elementos diversos (serviços e aplicações que compõem o sistema).

A dificuldade de atender todos estes requisitos adotando-se uma arquitetura distribuída teve como consequência um número reduzido de implementações de ferramentas de controle de acesso com essas características, em especial devido aos requisitos de alta capacidade de crescimento e granularidade de autorizações.

Esta carência de ferramentas de controle de acesso que atenda as necessidades de um sistema distribuídos originou o trabalho aqui apresentado.

1.3 Metodologia

Esta dissertação tem o intento de apresentar a arquitetura e implementação de um arcabouço de controle de acesso para ambientes distribuídos gerenciado por uma única entidade (administração). Como previamente assentado, não se pretende construir ou até mesmo especificar uma solução completa para os serviços de autenticação e geração de trilhas para auditoria apesar de se apontar parte da interface entre esses serviços.

Para que estes objetivos sejam atingidos, inicialmente propõe-se um estudo sobre o processo de segurança e a conseqüente contextualização do arcabouço de controle de acesso neste processo. Este estudo permite o entendimento da relevância do serviço de autorização e da performance esperada por este dentro do processo de segurança.

O segundo passo é a escrutinação dos elementos que compõem o serviço de autorização, em especial dos modelos de controle de acesso existentes. Esta análise permitirá, em um próximo passo, a escolha do modelo que melhor se adapta às necessidades de um sistema distribuído.

Uma vez especificada a composição do arcabouço de controle de acesso, o passo seguinte é o estudo do ambiente no qual este será implementado, possibilitando o levantamento de todos os requisitos que devem ser atendidos pelo arcabouço.

O quarto passo é a especificação da arquitetura do arcabouço e o modelo de política de controle de acesso a ser adotado, os quais devem atender a todos os requisitos levantados durante a etapa anterior.

O passo cinco será responsável por averiguar o desenvolvimento e correto funcionamento da solução especificada. Será construído um conjunto de ambientes (casos de uso) sobre os quais serão realizados testes para certificar que o modelo de controle de acesso escolhido foi totalmente implementado.

O último passo será a comparação do arcabouço construído com soluções existentes e, por fim, as conclusões sobre o trabalho apresentado e levantamento de possíveis trabalhos futuros.

2 SEGURANÇA EM COMPUTAÇÃO

Aristóteles argumenta que a definição de algo deve conter a definição das partes que o compõem. Sendo este princípio seguido, uma definição, ao ser finalizada, será composta por termos simples que não mais precisam de uma definição. [02]

Este princípio será utilizado no estudo de segurança proposto neste trabalho, o qual interpreta segurança como um processo, que deve ser definido segundo o seu objetivo e as ações necessárias para que este seja atingido. As seções a seguir possuem exatamente este intento, com a única ressalva da discordância encontrada na literatura quando da definição de determinados conceitos de segurança, como será explicitado no decorrer destas seções.

2.1 Segurança: Objetivo

O conceito de segurança está relacionado à proteção de bens, i.e., entidades que possuem valor. Como exemplo dessas entidades é possível citar: pessoas, objetos, características, relações, informações, etc. A necessidade de proteção advém dos riscos aos quais estas entidades estão sujeitas, i.e., deve-se proteger as entidades em questão de qualquer evento que possa causar danos e/ou a sua perda.

Antes de explicitar o conceito de segurança na área de computação é imperativo expor a distinção reconhecida pela literatura de duas frentes de estudo: Segurança da Informação e Segurança de Sistemas Críticos [03]. Este trabalho tende a se restringir apenas a Segurança da Informação, sem ter pretensões de adentrar o estudo de Sistemas Críticos. Essa reserva é relevante não apenas na definição do escopo deste trabalho, mas também para entender que o termo “segurança em computação”, neste texto, se restringe apenas à segurança da informação.

A segurança em computação tem o objetivo de “proteger os recursos de um sistema computacional, ou seja, preservar a integridade, disponibilidade e confidencialidade dos recursos (*hardware*, *software*, *firmware*, informação e telecomunicação) que fazem parte do sistema” [04]. Por conseguinte, o entendimento destes conceitos é condição *sine qua non* para o estudo de segurança em computação, mesmo que sua interpretação seja fortemente influenciada pelo ambiente no qual estes conceitos são referenciados [05].

As três seções a seguir possuem o intento de explorar estes conceitos, não necessariamente propondo uma definição formal, mas, com a ajuda da literatura existente, proporcionando o seu entendimento.

2.1.1 Integridade

A **integridade** é uma característica retida pela entidade que se mantém inteira, perfeita, exata. Em computação o conceito de integridade está ligado à confiabilidade de recursos e dados [05], sobre os quais não devem recair ações que possam promover danos ou adulterações, independente da origem destas ações: usuários (ataques ou erros) ou sistema (inadequações ou falhas).

Embora não se encontre um consenso sobre a definição formal para integridade [06, 07, 08, 09], o *National Institute of Standards and Technology* (NIST) entende que uma informação possui integridade quando esta é acurada, válida em relação ao tempo, completa e consistente. Além disso, o NIST reconhece que computadores não podem garantir todas essas qualidades e que, por este motivo, a segurança em computação trata integridade em uma forma mais restrita, a qual é abordada sobre duas frentes: a integridade de dados e a integridade de sistemas.

A distinção entre essas duas frentes é obtida ao considerar que o objetivo da integridade de dados é garantir que as informações e os programas (*softwares*) sejam modificados apenas com autorização e pelas formas especificadas pela política de segurança, enquanto que a integridade de sistemas deve garantir que

estes realizem as suas funções de forma correta, livre de manipulações não autorizadas (intencionais ou não) [04].

Este trabalho adotará a interpretação de integridade fornecida pelo NIST, apresentada a cima, uma vez que esta definição leva em conta a proteção da informação como um todo, explicitando a diferença entre integridade de dados e integridade de sistemas de forma concisa, e prevendo a necessidade de métodos externos ao sistema computacional para garanti-la.

2.1.2 Disponibilidade

No estudo de **segurança em sistemas críticos**, a **disponibilidade** é definida como a medida da probabilidade de um sistema não possuir falhas em um determinado instante de tempo [03]. A **segurança em computação** redefine **disponibilidade** preocupando-se apenas em garantir que usuários que possuem autorização sempre sejam capazes de interagir com os recursos do sistema.

A definição de disponibilidade provida por [10] e [04], e também adotada por este trabalho, descreve disponibilidade como a garantia de que o sistema funcione prontamente e que serviços não sejam negados a usuários que possuam autorização.

A partir da análise das duas definições de disponibilidade fornecidas é possível entender a diferença deste conceito nas duas áreas. Embora ambos os estudos intentem proteger o sistema contra eventos que interrompam ou interfiram no funcionamento do sistema, o estudo de segurança em sistemas críticos concentra-se em problemas oriundos de acidentes e a **segurança em computação em problemas oriundos da interação entre usuários e o sistema**.

2.1.3 Confidencialidade

O intuito da **confidencialidade** é garantir que recursos não sejam disponibilizados sem autorização [05]. Embora esta definição esteja alinhada com o objetivo da segurança em computação, de proteger os recursos do sistema, é comum definir **confidencialidade** apenas em função da informação. Essa restrição leva à definição fornecida pela ISO/IEC-17799, referenciada por [11], a qual afirma que o intuito da confidencialidade é garantir que informações sejam acessíveis apenas por aqueles que possuem autorização.

Além de proteger a forma como a informação é acessada, Bishop [05] argumenta que existem casos em que a própria existência da informação é mais importante do que a informação em si, inferindo que esta existência precisa ser protegida e que seria responsabilidade do estudo de confidencialidade garantir esta proteção.

Como apontado por Stallings [10], o conceito de **confidencialidade** ainda deve conceber a proteção da privacidade, a qual garante o controle ou influência de indivíduos sobre quais informações relacionadas a eles podem ser coletadas e armazenadas e por quem e para quem estas podem ser providas.

Embora o conceito de **confidencialidade** seja abordado de uma forma diferente por cada autor, ainda assim não é possível afirmar que exista um conflito entre essas definições. A definição que Bishop apresenta é talvez a mais ampla por considerar todos os recursos do sistema (e não apenas a informação), além de atentar sobre a proteção da existência da informação, no entanto, ela pode ser complementada pela fornecida por Stallings, ao ponderar também sobre a proteção da privacidade.

Contudo, a definição fornecida por Bishop é a que melhor se adapta aos requisitos de uma ferramenta de controle de acesso, que deve proteger, de fato, todos os recursos de um sistema e não apenas as informações, motivo pela qual esta será a definição adotada por este trabalho.

2.2 Segurança: Modelo e Processo

Alguns conceitos foram expostos a fim de se demonstrar os objetivos do processo de segurança, como riscos e bens, contudo, outros ainda precisam ser apresentados para que o processo em si seja explicitado. Para tanto será utilizado o modelo de segurança provido por [12], o qual pode ser visualizado na Figura 1 sob uma tradução livre, e que apresenta o processo de segurança de forma genérica, possibilitando a contemplação destes conceitos e suas relações.

O modelo é apresentado na sua forma genérica com o intuito de preservar a sua origem, o estudo de segurança, para apenas posteriormente ser inserido no contexto da segurança em computação. Embora a descrição do modelo oferte uma definição dos termos observados na figura, este estudo deve exceder esta definição a fim de transpô-la para a área de segurança.

Como previamente estabelecido, bens são entidades que possuem valor para alguém. A pessoa que dá valor a este bem é o seu proprietário, o qual tem a responsabilidade de protegê-lo dos riscos aos quais o bem está sujeito. O proprietário materializa a proteção do bem através de contramedidas que visam diminuir os riscos.

Todavia, além dos riscos oriundos do ambiente no qual o bem está inserido, existem aqueles provenientes de ameaças criadas por atacantes. Os atacantes são entidades que de certa forma podem ter acesso ao bem por meio de ações dolosas, ou não, podem infringir algum dano.

A influência do ambiente e das pessoas que participam deste meio é responsável pela dificuldade do estudo de segurança, uma vez que a relação destes para com o bem não pode ser facilmente estabelecida, quantificada ou definida, além de torná-la altamente dependente do contexto onde a segurança é abordada.

Figura 1 – Conceitos e Relações em Segurança. Adaptado de [12].

Em computação, os bens, em geral referenciados como recursos, são enquadrados em uma das seguintes categorias: *Hardware*, *Software*, Informação/Dados e Telecomunicação [5]. Não obstante o intuito da segurança em computação ser o mesmo para todos os recursos, essa categorização é importante para se determinar os métodos que serão utilizados para tal finalidade.

Salvaguardar os recursos do sistema é uma função do proprietário. Contudo, em computação, esta denominação carece de cuidados uma vez que o proprietário do recurso não é necessariamente o único a utilizá-lo, e, por conseguinte, não é o único a valorizar este recurso. O proprietário pode ser personificado por um usuário do recurso independente da sua concretização, uma pessoa ou outro sistema, e do seu caráter, físico ou jurídico.

A segurança em computação possui o intuito principal de minimizar os riscos representados por ameaças oriundas de atacantes. A ação do atacante pode ser dolosa (usuários maliciosos) ou culposa (usuário que cometeu um erro), todavia o seu resultado é o mesmo: perda da integridade, confidencialidade e/ou disponibilidade do recurso.

Para atingir o seu desígnio, a segurança em computação propõe uma série de contramedidas que devem se contrapor aos riscos, e que, por subsequente, se tornam o seu principal produto.

Embora não seja o único recurso, a informação pode ser considerada o principal bem a ser protegido, e é possível argumentar que até mesmo a relevância dos outros recursos é derivada do seu poder de armazenar, processar e transmitir a informação, motivo pelo qual a segurança em computação é muitas vezes cognominada segurança da informação, como aludido no início deste capítulo.

Admitido que a informação deve ser protegida pela segurança da computação, é fácil anuir que as contramedidas geradas devem permear todas as atividades que de alguma forma interagem com a informação, e, conseqüentemente, impõe ao processo de segurança da informação uma realidade mais abrangente que deve contemplar mais do que sistemas computacionais.

Embora a escrutinação do processo de segurança da informação não faça parte do escopo deste trabalho, [13] explicita que este processo é composto basicamente por seis sub-processos a saber: Política, Divulgação e Treinamento, Tecnologia e Implementação, Monitoração e Reação, Auditoria e Conformidade e Análise de Risco e Planejamento.

A explanação destes, também provida por [13], mostra que estes sub-processos na verdade formam um ciclo virtuoso, onde a Política de Segurança deve ditar os regulamentos a serem seguidos, e, portanto, precisa ser conhecida e seguida por todos que de alguma forma interagem com a informação. O segundo sub-processo do ciclo, Divulgação e Treinamento, possui exatamente esta finalidade, garantir que todos possam conhecer e se comprometer em seguir as políticas de segurança.

Se a política de segurança é responsável por estabelecer como as atividades devem ser realizadas, e, em última instância, o que pode ou não ser feito, é responsabilidade dos mecanismos de controle de acesso e monitoração de controles a aplicação coercitiva desta política e a determinação se esta é respeitada ou não. Ao adir a reação aos eventos detectados pela monitoração, os dois sub-processos seguintes do ciclo, Tecnologia e Implementação e Monitoração e Reação são compostos.

Além da monitoração, o processo de auditoragem também possui o intuito de identificar atividades que não se encontrem em conformidade com a política de segurança. Todavia, diferente da monitoração, que se propõe a identificar violações da política de segurança com aspecto tecnológico, o processo de Auditoria e Conformidade tenciona encontrar violações em toda e qualquer atividade relacionada com a informação, e não apenas aquelas ligadas aos sistemas computacionais.

O último processo a ser descrito deste ciclo é o Planejamento de Segurança e Análise de Risco, o qual deve avaliar as falhas de segurança que foram identificadas nos processos anteriores e determinar que ações devem ser tomadas, e.g. mudança da política de segurança, de forma que todas as não-conformidades sejam retificadas.

A exposição destes sub-processos, embora de maneira simplista, possibilita o entendimento do processo de segurança e da relevância de cada sub-processo, assim como do ciclo que eles formam e que assegura o aperfeiçoamento e readaptação do processo de segurança a novas necessidades oriundas da evolução do ambiente.

2.3 Considerações Finais

Estes conceitos servirão de base para entender as responsabilidades e limitações das ferramentas de controle de acesso, possibilitando a sua definição e permitindo a compreensão das limitações deste arcabouço dentro do processo (de segurança) que visa proteger os bens do sistema.

3 CONTROLE DE ACESSO

A recomendação X.800 feita pela *International Telecommunication Union* (ITU) estipula como intuito do controle de acesso “a prevenção do uso não autorizado de recursos, incluindo a prevenção do uso de recursos de forma não autorizada” [14]. A [15] procura aprimorar esta definição ao explicitar que o controle de acesso deve proteger o sistema contra acessos não autorizados e que o acesso deve ser regulado segundo uma política de segurança e provido apenas a entidades (usuários, programas, processos ou outros sistemas) autorizadas.

Mais relevante do que a especificação das entidades que acessam os recursos, é explicitar a observância do controle de acesso à política de segurança. Como previamente assentado, o **controle de acesso** possui um papel significativo dentro do processo de segurança, ainda que restrito ao aspecto tecnológico do processo, ele é um dos serviços responsáveis por impor a política de segurança às atividades computacionais.

O **controle de acesso** pode ter impacto sobre todos os objetivos da segurança em computação. Contudo a sua implementação restringe-se em geral a apenas dois: integridade e confidencialidade. Mas, antes de explicitar a sua relação com esses objetivos, é importante ressaltar que de forma alguma ele exauri o problema de garantir a segurança aos recursos, sendo somente mais um mecanismo utilizado com este intento.

A capacidade de limitar o acesso aos recursos do sistema proporciona um relacionamento direto entre a confidencialidade e o controle de acesso. Atividades como leitura e cópia apenas serão permitidas se o usuário possuir autorização para tanto, possibilitando que a confidencialidade das informações seja preservada.

Da mesma forma, as atividades de criação e modificação da informação passam a ser atividades restritas, ou seja, só podem ser executadas por usuários que, segundo a política de segurança, possuem essa autorização. Essa restrição contribui para que a integridade da informação seja mantida.

Embora esta relação não seja tão explorada pelos serviços de controle de acesso comuns, o número de vezes que um recurso é acessado também pode ser controlado. Essa restrição é particularmente útil para recursos com número limitado de acessos simultâneos, pois ela é capaz de impedir que um usuário (ou conjunto de) esgote o recurso, ocasionando perda de disponibilidade.

A composição do serviço de controle de acesso pode ser compelida a conter todo e qualquer meio que de fato controle o acesso a um recurso do sistema (e.g., criptografia, *firewall*). Contudo, mesmo que esta propositura seja aceita é possível divisar três serviços essenciais para o controle de acesso: **autenticação**, **autorização** e **geração de trilhas para auditoria**.

Estes três serviços serão explanados nos tópicos a seguir. Entretanto existem três conceitos que precisam ser esclarecidos antes: sujeito, operação e objeto. O sujeito é a entidade (usuário, programa, processo ou outro sistema) que deseja realizar uma ação no sistema, seja esta legal ou ilegal. A ação a ser realizada (e.g., escrita, leitura, execução, criação, etc.) é chamada operação, independente do êxito na sua realização. O objeto é o recurso sobre o qual a operação, que foi iniciada pelo sujeito, tenciona ser realizada.

Embora mais informações possam ser utilizadas para se determinar se uma ação pode ser realizada, a tríade formada por **sujeito**, **operação** e **objeto** é considerada essencial para o processo de **controle de acesso**.

O serviço de **autenticação** tem o objetivo de identificar o sujeito, possibilitando ao serviço de **autorização** determinar se as operações impetradas por ele podem, ou não, ser realizadas. O serviço de **geração de trilhas** para auditoria é responsável por armazenar as operações demandadas pelo sujeito com o intento de se averiguar o comprimento da política de segurança. Em geral esta averiguação é realizada posteriormente a execução de operações, contudo a averiguação simultânea pode trazer benefícios principalmente na observância de operações complexas, compostas por diversas sub-operações.

3.1 Autenticação

O primeiro passo para que uma entidade externa passe a fazer parte de um sistema computacional é em geral a sua **autenticação**. Inúmeras são as formas existentes para que esta entidade seja identificada (e.g., biometria, usuário-senha, certificados), contudo o seu objetivo é sempre o mesmo, i.e., identificar a entidade como um sujeito legítimo (entidade conhecida pelo sistema) e permitir o seu acesso ao sistema.

É importante neste ponto diferenciar a identificação da autenticação de um usuário. Ao identificar um usuário, o sistema descobre quem é a entidade que deseja interagir com o sistema, e ao autenticá-lo, o sistema o reconhece como um usuário válido e que pode, a partir de então, realizar suas atividades.

A autenticação de uma entidade como um sujeito é de suma importância para o serviço de **controle de acesso**, uma vez que as operações tencionadas por este serão avaliadas segundo esse sujeito. Independente da forma de autenticação escolhida, o serviço de **autenticação** dependerá de duas atividades: a criação e a identificação do sujeito.

A criação de sujeitos é por definição uma atividade mais contida, uma vez que ela deve ser restrita a um sujeito já conhecido e/ou a um processo intrínseco ao sistema. Embora esta restrição limite a capacidade de ataques a essa atividade, ela não pode ser subestimada, pois o resultado oriundo de erros durante a sua realização pode fornecer a um atacante acesso aos recursos.

Como mencionado anteriormente, o objetivo do serviço de **autenticação** é garantir que, quando aplicável, as entidades sejam identificadas como um sujeito do sistema. A dificuldade de se proteger esta atividade advém do seu intento nato, i.e., mesmo que seja possível limitar a origem das entidades que podem ter acesso ao serviço de **autenticação** (e.g., pertencer a uma determinada rede), ele ainda precisa ser ofertado a qualquer entidade, sendo esta um sujeito legal ainda não identificado ou um atacante.

Existem diversas formas de **identificação**, mas todas elas se baseiam em uma informação que a entidade que deseja se identificar sabe (e.g. senha), possui (e.g. certificado) ou é inerente a ela (e.g. biometria), e que é confrontada com a informação conhecida pelo sistema. Além da informação, os serviços de **autenticação** divergem sobre o seu armazenamento: local ou remoto.

Um dos serviços de **autenticação** mais difundidos é a utilização de senhas. A entidade que deseja se autenticar precisa fornecer um nome de usuário e uma senha, ou seja, precisa especificar o sujeito que deseja se identificar, e, ao fornecer a senha, provar para o sistema que ela de fato corresponde àquele sujeito. Em geral, esta dupla é confrontada com um arquivo de senhas que se encontra no sistema de arquivos da estação. O sistema operacional UNIX adotou esta forma de identificação e, embora suporte outras, o mantém até hoje.

Embora **serviços de autenticação** baseados na dupla usuário/senha respondam satisfatoriamente para uma única estação, em um ambiente distribuído, onde os recursos estão espalhados pela rede, a sua atuação deixa de ser eficaz ao impor que todas as estações confiem na identificação realizada por uma única estação, ou que o controle de acesso seja restrito a uma única estação, o que exigiria o gerenciamento de tantos serviços de controles de acesso quanto o número de estações existentes no sistema.

O **serviço de autenticação** Kerberos [16] foi desenvolvido como parte do projeto Athena no *Massachusetts Institute of Technology* (MIT) para solucionar este problema. A idéia básica por trás do Kerberos é prover **autenticação** de um usuário para qualquer serviço por ele solicitado. A identificação é realizada pela troca de mensagens criptografadas com segredos fornecidos pelo Kerberos (tanto o usuário quanto o serviço confiam no Kerberos) e possibilita não apenas a identificação do usuário frente ao serviço, mas também a do serviço frente ao usuário.

O processo de **autenticação** utilizando este serviço pode ser simplificado ao seguinte algoritmo: o individuo utiliza um programa cliente que deve exigir um usuário e senha, e em seguida, pedir ao serviço Kerberos um tíquete de

identificação. Esse tíquete será criptografado utilizando a senha do usuário (que é conhecida pelo Kerberos) e enviado ao cliente, que, em posse da senha fornecida pelo usuário, poderá recuperar o tíquete.

Este instrumento será utilizado para obter tíquetes que identifiquem o usuário frente a serviços sempre que necessário, evitando que o usuário precise digitar a sua senha a cada requisição. A diferença é que estes tíquetes possuem um tempo de vida pequeno, e são criptografados com uma chave que apenas o serviço requisitado possui, e, portanto, apenas o verdadeiro serviço pode recuperar.

Uma terceira solução é a recomendação X.509 da ITU-T [17] que define uma **plataforma de autenticação** baseada em certificados, os quais são expedidos por uma Autoridade Certificadora (CA – *Certificate Authority*). A **autenticação** é realizada por meio da troca de informações criptografadas a partir da chave pública das duas partes envolvidas, ou seja, se uma das entidades não for quem alega ser, ela não conseguirá recuperar a informação.

Todavia, dado o interesse em arquiteturas distribuídas deste trabalho, mais importante do que explicar os protocolos de **autenticação** é explicitar a utilização de serviços de diretório para armazenar as informações dos usuários. Este serviço pode ser provido por uma única estação, ou por diversas, aumentando a confiabilidade do serviço, algo imprescindível para um sistema distribuído. O serviço de diretório não se restringe a certificados, ele pode armazenar qualquer tipo de informação, o que contribui para a sua disseminação inclusive entre os serviços de autenticação mencionados anteriormente (autenticação por senha e Kerberos).

3.2 Autorização

A relevância dos **serviços de autenticação e de geração de trilhas para auditorias** frente o serviço de **controle de acesso** não pode ser contestada, contudo o **serviço de autorização** é quem de fato limita as ações de usuários no sistema [18]. Esse objetivo deverá definir a política de **controle de acesso** do sistema, a qual, por mais simples que seja, deve contemplar as entidades: **sujeito, operação e objeto**.

Para que uma operação possa ser realizada, ela deve ser submetida ao julgo do **monitor de autorização**, o qual deve consultar a base de dados de autorização e decidir se a operação pode ou não ser realizada. Essa decisão será determinada pela **política de controle de acesso**, que pode ser resumida a uma relação entre o sujeito, a operação, o objeto e uma permissão, i.e., a operação será realizada se o sujeito requisitante possuir permissão para realizá-la sobre aquele determinado objeto.

A **base de dados de autorização** é gerenciada pelo(s) administrador(es) de segurança, entretanto, a permissão para sua manipulação pode ser disponibilizada para um sujeito. Essa manipulação não deve permitir que a base de dados deixe de refletir a Política de Segurança, corroborando a necessidade dos usuários de conhecê-la e a relevância do sub-processo de segurança Divulgação e Treinamento.

A **política de controle de acesso** pode ser restrita a uma relação direta entre o sujeito, a operação e o objeto, contudo outras políticas foram criadas com o intuito de reduzir o trabalho de manutenção desta e para que ela abstraísse relações reais entre usuários. Dentre as políticas criadas, três ganharam destaque e serão escrutinadas a seguir: Controle de Acesso Discricionário (DAC – *Discretionary Access Control*), Controle de Acesso Obrigatório (MAC – *Mandatory Access Control*) e Controle de Acesso baseado em Papéis (RBAC – *Role-based Access Control*).

Em agosto de 1983, o Departamento de Defesa (*Department of Defense – DoD*) dos Estados Unidos da América publicou o *Trusted Computer System Evaluation Criteria*

(TCSEC) [19], o qual possuía o intuito de estabelecer um padrão para **controles de acesso** baseados nas **políticas de controle de acesso** **discricionária** e **obrigatória**, e, ao mesmo tempo, estabelecer um critério de avaliação.

O TCSEC explicita que o **controle de acesso** discricionário deve definir e controlar o acesso de usuários a objetos, possibilitar que um usuário proveja ou revogue direito de acesso de outros usuários sobre os recursos controlados pelo primeiro, e, por fim, proteger os objetos contra acessos não autorizados. Esta política de **controle de acesso** também deve ser capaz de excluir acesso com a granularidade de um único usuário e assegurar que o acesso a um objeto seja provido a um usuário apenas por um segundo usuário que possua autorização para tal.

Uma abordagem comumente utilizada para prover essa política prevê a utilização de uma **matriz de controle de acesso**. Essa matriz tem, em uma de suas dimensões, os sujeitos que desejam acessar um objeto, ou seja, indivíduos ou grupos de indivíduos, e na segunda dimensão lista os objetos que podem ser acessados. A granularidade para esses objetos não é especificada, podendo, por exemplo, representar uma base de dados ou um único campo de informação. Por último, as entradas da matriz indicam o direito de acesso do sujeito para aquele objeto.

Na prática, essa matriz é decomposta e espalhada no sistema em uma das seguintes formas: **lista de controles de acesso** (*Access Control List – ACL*) ou **capabilities**. As **listas de controle de acesso** são obtidas ao se dividir a matriz em colunas, ou seja, para cada objeto a **lista de controle de acesso** fornece o sujeito e o seu direito de acesso. Além disso, a ACL pode possuir uma entrada padrão ou pública, permitindo que ela contenha apenas os sujeitos que de fato possuem permissões diferenciadas, e uma única entrada para todos os outros.

A decomposição da matriz em linhas da origem aos **capability tickets**, os quais especificam os objetos e direitos de acessos para cada sujeito do sistema. Estes tíquetes não são restritos a um sujeito, podendo ser emprestados ou doados a terceiros, entretanto esta funcionalidade impõe ao sistema uma nova carga: a necessidade de garantir que cada tíquete utilizado seja genuíno [10, 05].

A segunda política descrita em [19] é o **controle de acesso obrigatório**. Este **controle de acesso** não se baseia simplesmente na identidade do usuário, ou seja, não é a sua identidade que determinará se ele possui ou não acesso a determinado objeto, o controle será realizado segundo a classificação do sujeito e do objeto no sistema.

Todos os objetos que devem ser protegidos pelo sistema recebem uma classificação, ou *label*, a qual reflete a importância do recurso representado por aquele objeto. Os sujeitos também possuem uma classificação de segurança, ou *clearance*, a qual representa a confiança que o sistema possui nesse usuário, i.e., de que ele não irá repassar a informação para pessoas não autorizadas [18].

Este tipo de **controle de acesso** foi e, ainda, é muito utilizado por militares e agências governamentais, onde, em geral, os níveis de segurança em ordem decrescente de relevância são: *Top Secret (TS)*, *Secret (S)*, *Confidential (C)*, e *Unclassified (U)*. Os níveis de segurança dominam os níveis de segurança anteriores e o seu próprio, ou seja, um sujeito com *clearance S*, pode ler um documento classificado como U, C ou S, mas não um que seja classificado como TS. Esta regra é conhecida como *Read up*.

Da mesma forma, a operação de escrita é controlada pela regra *Write down*. Segundo esta regra, o mesmo sujeito que possui *clearance S*, não pode escrever um documento classificado como U ou C, pois o seu nível de segurança deve ser dominado pelo nível de segurança do documento escrito por ele, ou seja, os documentos escritos por um sujeito com nível S deve possuir nível S ou TS.

O intuito desta regra é evitar que um usuário possa copiar as informações de um documento do seu nível para um novo documento com nível menor, violando a **política de acesso**. Embora esta regra impeça a comunicação entre um usuário de nível superior com um usuário de nível inferior, este problema é solucionado pela possibilidade de um usuário entrar no sistema com qualquer nível que seja dominado pelo seu nível mais alto.

Além dos níveis hierárquicos de segurança, outras categorias podem ser criadas para controlar o acesso aos objetos. Essas categorias não hierárquicas podem ser utilizadas para segmentar os objetos para que apenas os sujeitos pertencentes àquela categoria possam ter acesso, independente do nível de segurança do sujeito.

A terceira política de acesso, o **controle de acesso baseado em papéis**, surgiu como uma resposta a anseios que as duas outras políticas não atendiam [20]. O controle de acesso obrigatório é regido por uma hierarquia de vários níveis que enquanto reflete muito bem o ambiente militar, dificulta a sua utilização em outros ambientes que não podem ser representados por esta.

Mesmo que o **controle de acesso discricionário** tenha sido desenvolvido para atuar tanto no ambiente militar quanto em organizações civis e comerciais, a sua abordagem de permitir que usuários provejam ou revoguem direito de acesso sobre objetos a terceiros sem intervenção dos administradores do sistema não abstrai a relação entre os usuários e os recursos que, de fato, pertencem à empresa, e que por esta razão deveriam estar sob a sua autoridade. [21]

Contudo, é importante ressaltar que a **política de controle de acesso baseada em papéis** também não atende a todos os anseios respondidos pelas políticas anteriores, ou seja, é preciso inferir qual política melhor atende às necessidades do sistema que se pretende proteger. Segundo [18], estas políticas não são excludentes, podendo ser utilizadas em conjunto, relação que pode ser visualizada na Figura 2.

Figura 2 - Políticas de Controle de Acesso. Adaptado de [18]

A política de controle de acesso RBAC (*Role Based Access Control*) entende que as decisões de controle de acesso são baseadas nas funções que os indivíduos desempenham dentro da empresa, incluindo especificações do seu trabalho, responsabilidade e qualificação [21]. Para abstrair esta idéia, o RBAC define uma entidade denominada papel que tem o intuito de representar estas funções. Os direitos de acesso são concedidos a esta entidade e para que um indivíduo possa realizar uma ação, o seu sujeito deve ser associado a este papel.

A proposta de padronização desta política foi aprovada em 2004 pelo *American National Standards Institute, Inc* (ANSI) [22], após um longo processo de discussões abertas, e publicações de *drafts*. Embora estes *drafts* e outros artigos sejam referenciados e, de fato, ajudem tanto na descrição do RBAC quanto no problema que o originou (dificuldade de se representar o ambientes de negócios nas políticas previamente existentes), a política descrita neste trabalho será contida pelo modelo aprovado.

O modelo de referencia do RBAC é composto por quatro componentes: **Núcleo RBAC** (*Core RBAC*), **RBAC Hierárquico** (*Hierarchical RBAC*), **Separação de Deveres Estático** (*Static Separation of Duty Relations*) e **Separação de Deveres Dinâmico** (*Dynamic Separation of Duty Relations*). O **núcleo RBAC** define uma coleção mínima de elementos, conjunto de elementos e relacionamentos para se obter um **controle de acesso baseado em papéis**, sendo, portanto, fundamental para qualquer sistema RBAC. Os outros componentes são independentes, podendo ser implementados separadamente ou totalmente omitidos.

Figura 3 - Modelo RBAC. Adaptado de [22].

O núcleo RBAC define cinco elementos, visualizados na Figura 3, que possuem a intenção de abstrair a relação entre os usuários e os recursos do sistema: usuários (USRS), papéis (PAPEIS), objetos (OBJS), operações (OPS) e permissões (PERMISSÕES) e sessões (SESSN). A entidade cognominada usuário é a mesma entidade já descrita nesta dissertação como sujeito, assim como os objetos e as operações também já o foram.

A permissão representa a concessão para um sujeito realizar uma operação sobre um objeto. As permissões são associadas aos papéis, que devem representar as funções realizadas pelos indivíduos na vida real.

Esses papéis são atribuídos aos usuários do sistema em uma relação de muitos para muitos. Esta característica pretende atender ao requisito conhecido como menor prioridade, i.e., o usuário pode acessar o sistema a partir do papel que possuir o mínimo de permissões possível para que a tarefa seja executada.

O mapeamento entre os usuários e os papéis que eles utilizam é feito pela sessão, como visualizado na Figura 3. Um usuário pode estabelecer uma ou mais sessões, e, dentro de cada uma, ativar um ou mais papéis. A implementação da sessão e de todas as entidades previamente descritas, assim como a relação entre elas, deve ser atendida por todo e qualquer sistema RBAC.

Em geral, as funções estão atreladas a uma estrutura hierárquica que pretende refletir a linha de autoridade e responsabilidade da organização. O **RBAC Hierárquico** possui o intento de representar esta estrutura, ao permitir que papéis possam herdar permissões de outros papéis, i.e., um papel r1 herda um papel r2 se todos as permissões de r2 também forem permissões de r1.

O **RBAC Hierárquico** reconhece dois tipos de herança: “**General**” e “**Limited**”. A herança “**General**” suporta o conceito de múltiplas heranças, ou seja, um papel pode herdar permissões de um ou mais papéis. A principal vantagem desta característica é a habilidade de criar papéis pela composição de muitos papéis subordinados.

A herança “**Limited**” restringe o número de ascendentes a apenas um, ou seja, um papel só pode herdar permissões de um único outro papel. Embora essa seja apenas uma limitação quando comparada à herança “**General**”, ela ainda assim provê vantagens administrativas em relação ao **Núcleo RBAC**.

Entende-se uma transação como um conjunto de operações que precisam ser concretizadas para se atingir um objetivo. Dependendo da relevância da transação, ou do impacto que ela tem para a organização, é normal exigir que esta seja acompanhada por mais de um indivíduo, evitando fraudes ou erros. O exemplo mais comum para representar esta situação é o processo de compra de equipamentos e/ou materiais, onde o funcionário responsável pode alterar os valores de compra a fim de reter as diferenças.

Em um serviço de controle de acesso a **Separação de Deveres** garante que não ocorreram fraudes ou erros ao demandar que mais de uma pessoa, de preferência que não possuam uma relação direta de trabalho, participe desta transação, aumentando a chance de erros serem percebidos e exigindo o conluio de um grupo de pessoas para que uma fraude ocorra.

O RBAC prevê duas formas de implementação da separação de deveres: **Separação de Deveres Estática** (*Static Separation of Duty Relations*) e **Separação de Deveres Dinâmica** (*Dynamic Separation of Duty Relations*). Antes de explanar as duas formas deve-se ressaltar que a política de controle de acesso tem a responsabilidade de não prover, a um mesmo papel, os direitos de acesso a todas as operações que compõem as transações que precisam ser observadas.

A primeira implementação prevê o agrupamento dos papéis que ao serem reunidos possuem o direito de acesso a todas as operações que compõem uma transação. Para evitar que um usuário obtenha todos esses direitos, ela impede que a um mesmo usuário seja atribuído mais de um dos papéis que foram agrupados.

A segunda implementação também prevê este agrupamento, contudo, ela permite que esses papéis sejam atribuídos a um mesmo usuário, mas o impede de ativar todos estes papéis em uma mesma sessão, i.e., o usuário pode acessar o sistema

com qualquer um destes papeis, contudo, ao ativar um papel ele ficará restrito a este até o final da sessão, sem a possibilidade de ativar outro que pertença ao agrupamento.

3.3 Geração de trilhas para auditoria

Conteúdo fixo é definido como um registro que tipicamente é escrito uma vez e não é modificado, e.g., ordens de compras, registro de ligações telefônicas e eventos de sistemas computacionais. Este tipo de conteúdo pode ser caracterizado pelos seus requisitos de acesso *online*, autenticidade garantida e retenção por longa data.

A sua importância deriva do seu intuito: evidenciar transações de negócios que podem suportar decisões de gerenciamento, garantir que obrigações foram cumpridas e proteger a organização de ações litigantes. [23, 24] Para o processo de segurança, o seu principal objetivo é permitir a averiguação de que todas as ações realizadas estão em conformidade com as políticas de segurança.

Se for possível garantir que todas as operações efetuadas no sistema são registradas e que esses registros são incorruptíveis, é possível não só identificar ações que violam as políticas de segurança, mas também identificar os infratores, aplicar sanções com um apoio legal (provas) e, quando plausível, corrigir as falhas de segurança que permitiram que o evento ocorresse. Este é o objetivo do **serviço de geração de trilhas**.

A **geração de trilhas** para auditoria possui dois grandes desafios, sendo o primeiro a forma de armazenamento das trilhas, uma vez que a garantia de integridade destes registros é vital para a sua utilização em ações judiciais.

O segundo é a identificação de todas as informações necessárias para a criação de registros, pois é essencial armazenar informações suficientes para se compreender os eventos, sem, no entanto, gerar um montante que dificulte o levantamento de informações relevantes.

Mesmo que algumas informações como a operação, o usuário, o horário e o(s) objeto(s) afetados sejam imprescindíveis para o processo de auditoria, observar o ambiente no qual o serviço de geração de trilhas será implantado é essencial para se determinar quais outras informações são necessárias. Essa observação pode ser concretizada pela escrutinação dos processos da organização, assim como das abstrações destes processos no sistema computacional.

Sistemas de armazenamento de dados baseados em *Content Addressed Storage* (CAS) procuram fornecer meios de atender aos requisitos do armazenamento de conteúdo fixo utilizando um novo paradigma: armazenamento de objetos. Diferentemente dos outros sistemas, os dados não são tratados como arquivos que devem ser armazenados por um sistema de arquivo.

Em um sistema baseado em CAS, a cada peça de dado é acrescido um rol de metadados, e este conjunto é tratado como um objeto. A partir deste objeto é gerado um identificador único, o qual desempenhará o papel de endereço do conteúdo. Este identificador é obtido ao se extrair o *hash* [25] do conteúdo.

O rol de metadados criados possibilita a indexação do conteúdo sem o prévio conhecimento de nomes, datas de criação ou qualquer outro designador de arquivo normalmente utilizado. O acesso a esses metadados garante a independência da implementação física dos dados, uma vez que eles podem ser utilizados para endereçar desde volumes lógicos até blocos de dados em discos sem um sistema de arquivos [26].

Devido às características inerentes ao tipo de conteúdo armazenado, como a requisição (por vezes legal) de imutabilidade e a baixa probabilidade de acesso recorrente (dados históricos); e às características inerentes à natureza do sistema de armazenamento CAS, como a independência da implementação física e lógica deste, verifica-se a necessidade da utilização de mídias onde só é possível escrever uma vez, embora seja possível ler muitas (*WORM – Write Once Read Many*), ou a utilização de sistemas de armazenamento que realizem a aplicação coercitiva das políticas de imutabilidade dos dados.

3.4 Considerações Finais

Este capítulo apresentou o serviço de controle de acesso, assim como os serviços que dão suporte para que seu objetivo seja atingido plenamente, e os principais modelos de política de controle de acesso. Estas descrições são importantes para definir a composição do arcabouço e apresentar as opções de modelo de controle de acesso que este implementará.

4 CONTROLE DE ACESSO EM SISTEMAS DISTRIBUIDOS

Tanto a relevância do controle de acesso para a segurança, quanto os seus objetivos, políticas e mecanismos foram descritos no capítulo 2 e 3. Todavia, posto o objetivo deste trabalho de especificar e implementar um arcabouço de **controle de acesso para sistemas distribuídos** é de suma importância o entendimento deste ambiente e do seu impacto sobre o sistema a ser desenvolvido.

Para que este objetivo seja alcançado é apresentada na seção 4.1 uma breve descrição sobre sistemas distribuídos e na seção 4.2 a enumeração dos requisitos advindos tanto deste estudo quanto da escrutinação das **políticas de controle de acesso** realizada na seção 3.2.

Este conjunto de requisitos permitirá o delineamento do **arcabouço de controle de acesso** desejado, mas antes mesmo que um esboço seja especificado é oportuno examinar trabalhos equivalentes ou que ao menos possuam objetivos em comum. A última seção deste capítulo possui este intuito e possibilitará que a solução proposta no capítulo 5 seja posteriormente diferenciada.

4.1 Conceituação de Sistemas Distribuídos

A quebra de tarefas em micro-tarefas e sua distribuição entre diferentes elementos de um conjunto (e.g., pessoas, departamentos) faz parte da civilização, como pode ser observado desde a construção das cidades estado na antiga Mesopotâmia e das pirâmides no Egito até as grandes corporações dos dias de hoje, onde inúmeros indivíduos trabalham coordenadamente em prol de um mesmo objetivo.

O início da era dos computadores foi caracterizado pelos grandes computadores, que ocupavam não apenas salas, mas andares inteiros. Além de grandes, esses

computadores custavam milhões de dólares, impedindo que a maioria das pessoas possuísse mais de um.

Entretanto, o avanço tecnológico permitiu tanto a redução de tamanho quanto de custo, popularizando o seu uso e possibilitando que a característica de distribuição de tarefas adentrasse este ambiente, levando à criação das redes de computadores e à sua utilização como ferramenta na busca de soluções para problemas que envolviam tarefas exercidas por mais de um indivíduo.

Alguns problemas apresentam características que exigem uma solução distribuída, por exemplo, o processo de vendas em uma empresa de varejo: Todos os produtos devem ser vendidos segundo uma única tabela de preço, contudo, o tamanho da loja pode tornar impraticável a utilização de um único computador (recurso) para este fim. Até mesmo a busca pela solução de problemas pode ser beneficiada pela cooperação de indivíduos, estando estes lado a lado ou dispostos geograficamente distantes.

Estes fatos em conjunto, à popularização dos computadores, o surgimento das redes de computadores e à necessidade de soluções distribuídas, culminaram com a criação dos sistemas computacionais distribuídos. Dada a sua própria natureza de atender a muitas entidades (que podem ser diferentes entre si), o que em geral implica em dificuldade ou impossibilidade de caracterização das entidades participantes e dos requisitos aos quais o sistema deve atender, é difícil propor uma definição geral e abrangente para os **sistemas computacionais distribuídos**.

Tanenbaum propõe uma definição que tende a ser abrangente o suficiente para permitir uma caracterização destes sistemas: “Um **sistema distribuído** é uma coleção de computadores independentes que se apresenta para o usuário como um sistema único e coerente.” (tradução livre) [27].

Entretanto, esta definição não releva a interação necessária entre as entidades participantes da coleção, e.g., se dentre todas as entidades apenas uma realiza o armazenamento e o processamento de informações, e todas as outras servem

apenas como terminal de consulta, esta coleção não poderá ser caracterizada como um sistema distribuído.

Se o foco da definição for alterado para a atividade que se deseja realizar, ao invés dos elementos que fazem parte do sistema, é possível obter a definição que Lamport criou: “No termo **computação distribuída**, a palavra distribuída significa espalhado através do espaço. Portanto, computação distribuída é a atividade realizada por um **sistema espacialmente distribuído**.”(tradução livre) [28]

Lamport ainda argumenta que embora geralmente seja utilizado o termo **sistema distribuído**, o mais apropriado seria se falar sobre uma visão distribuída de um sistema. Para explanar este conceito é utilizado o seguinte exemplo: um *projetista* de *hardware* enxerga um computador seqüencial como um sistema distribuído, já que as suas atividades serão realizadas por circuitos espacialmente distribuídos em uma placa de circuito, no entanto, um programador Pascal irá enxergá-lo como um sistema não distribuído.

Isto nos remete a uma das primeiras características (e problemas a serem solucionados) de sistemas computacionais distribuídos: a capacidade do sistema de oferecer ao usuário uma **visão não-distribuída** de um **sistema distribuído**, omitindo a complexidade de seu funcionamento oriunda da localização e não uniformidade dos elementos que o compõe e da comunicação existentes entre estes.

Outra característica esperada é a alta taxa de confiabilidade quando comparada a dos sistemas centralizados, uma vez que a falha em um dos computadores que compõem o sistema não deve necessariamente incapacitar o sistema como um todo, permitindo que os elementos restantes continuem a desenvolver a atividade requerida.

A capacidade de crescimento destes sistemas é uma importante característica, não apenas por permitir melhorias de desempenho na realização de uma tarefa, mas também por garantir que um aumento da carga imposta ao sistema possa ser balanceado pelo aumento do número de recursos.

4.2 Requisitos do Arcabouço de Controle de Acesso para Sistemas Distribuídos

As características dos **sistemas distribuídos** impõem novos requisitos ao **arcabouço de controle de acesso**, não apenas pela comunicação e localização (distribuída) dos recursos que devem ser protegidos, mas também pela própria natureza diversa destes. Ao se considerar essas características é fácil anuir que o **arcabouço de controle de acesso** deve, ele mesmo, se tornar um sistema distribuído.

Esta dupla natureza (sistema de segurança e sistema distribuído) leva a dois conjuntos de requisitos que o arcabouço aqui descrito deve atender. A construção destes será realizada por meio da união de conjuntos de requisitos básicos de cada natureza, e complementada com a análise da influência desta união.

4.2.1 Requisitos da Política de Controle de Acesso

O conjunto de requisitos a ser atendido pela **política de controle de acesso** é extraído das definições e referências previamente utilizadas, contudo este conjunto é principalmente delineado por [29]. A influência do ambiente (distribuído) será revelado por meio de ponderações realizadas ao longo desta enumeração.

- **RCA1 – Entradas Confiáveis:** A **política de controle de acesso** é descrita através de relações entre recursos, operações e usuários. Por este motivo é condição imprescindível que o arcabouço possa confiar na identificação destes elementos (entradas) para poder decidir se uma ação deve ou não ser realizada.
- **RCA2 – Suporte a diversos níveis de especificação:** A descrição de um recurso ou operação deve ser tão específica quanto necessária, possibilitando que permissões sejam concedidas tanto sobre um recurso quanto sobre uma parte deste.

Além disso, ao longo da vida de um sistema, o grupo de recursos e operações tende a ser alterado, motivo pelo qual estes não podem ser restringidos a grupos pré-definidos.

Essa característica de **mutabilidade da especificação de operações e recursos** é presente principalmente em **sistemas distribuídos**, que, dada a sua natureza, tendem a possuir uma maior diversidade de elementos do que os sistemas centralizados.

- **RCA3 – Autorizações condicionais:** A possibilidade de se restringir permissões a uma determinada condição pode ser de grande valia em um sistema distribuído, onde a utilização de um recurso por vários usuários é prevista e até esperada.

Uma condição especialmente interessante para **sistemas distribuídos** é a localização. Esta condição poderia ser adicionada a uma permissão, garantindo que um recurso fosse utilizado apenas por um usuário localizado em um determinado país, região, etc.

- **RCA4 – Menor privilégio:** Além de proteger o sistema contra ataques de usuários que não possuem permissões, a ferramenta de **controle de acesso** pode ser utilizada para proteger os recursos de erros (acidentes) cometidos por usuários que de fato possuem acesso.

Para se atingir este objetivo é provido ao usuário a opção de se conectar ao sistema tendo acesso a um conjunto mínimo de permissões (menor privilégio) necessárias para a realização de sua tarefa. Isto contribui para a contenção dos danos oriundos de falhas a apenas a um conjunto mínimo de recursos acessíveis ao usuário naquele instante.

Este requisito é de absoluta importância em ambientes distribuídos onde o número de recursos em geral é muito maior, o que além de dificultar a discriminação entre estes (aumentando a chance de acidentes) tende a aumentar o impacto de possíveis equívocos.

- **RCA5 – Separação de deveres:** Determinadas tarefas podem levar a sérios danos ao sistema ou recurso que se pretende proteger, contudo, estas ainda assim precisam ser executadas. Para aumentar a segurança pode-se exigir a aquiescência de dois ou mais usuários para que estas tarefas sejam executadas, dificultando que usuários mal intencionados se aproveitem ilicitamente do sistema e que erros aconteçam por descuido (ao se exigir dois usuários para validar uma operação se aumenta a chance de detecção de erros).

Um exemplo deste tipo de operação é a realização de compras em uma empresa. Sendo esta operação realizada por uma única pessoa, a compra pode ser realizada por um preço mais elevado beneficiando o vendedor (que pode repassar recursos para a pessoa responsável) em detrimento da empresa.

- **RCA6 – Administração de políticas:** O arcabouço deve permitir que apenas um conjunto de usuários possa modificar as políticas de controle de acesso, restringindo este controle apenas às pessoas autorizadas pela organização.

A análise dos requisitos enumerados mostra que o seu maior impacto se dará sobre a **política de controle de acesso**, uma vez que estes se referem principalmente às atividades que o arcabouço deve desenvolver (controle de acesso).

4.2.2 Requisitos de Sistemas Distribuídos

O segundo conjunto, **requisitos de sistemas distribuídos**, terá maior impacto sobre a arquitetura, uma vez que este levanta considerações sobre o ambiente ao qual o arcabouço deve se adaptar. Para a construção deste conjunto de requisitos será utilizado como suporte a conceituação previamente feita de **sistemas distribuídos** e as características descritas em [30]:

- **RSD1 – Capacidade de crescimento:** Um dos principais apelos de um sistema distribuído é a possibilidade de se utilizar um grande número de recursos para se

realizar tarefas mais complexas. Contudo, pode ser difícil de se determinar previamente qual será a carga máxima desta tarefa, ou se até mesmo o sistema será ou não utilizado para a execução de outras tarefas com cargas ainda maiores.

Esta situação leva ao primeiro requisito de um sistema distribuído, a capacidade de crescimento, i.e., o sistema deve ser capaz de absorver novos recursos e distribuir a carga ao longo destes.

- **RSD2 – Disponibilidade:** Entende-se que um sistema distribuído pode oferecer alta disponibilidade através da replicação de recursos e sistemas, ou seja, ainda que um recurso se torne indisponível, a tarefa imposta ao sistema poderá ser realizada pelos recursos restantes.
- **RSD3 – Confiabilidade:** O conceito de confiabilidade está ligado à probabilidade de um sistema não apresentar falhas por um determinado período de tempo. Embora o aumento no número de elementos do sistema possa aumentar a probabilidade de uma falha ocorrer, por meio de soluções de contingência, que utilizam os recursos replicados, é possível diminuir o impacto destas falhas, e até mesmo garantir a realização das tarefas propostas.
- **RSD4 – Vazão:** Dado que um sistema distribuído é composto de inúmeros elementos é esperado que a vazão total do sistema supere a vazão de sistemas monolíticos.
- **RSD5 – Abertura:** Um sistema distribuído deve suportar facilmente a sua expansão e atualização, permitindo que novas aplicações utilizem os recursos já existentes, e possibilitando a sua integração com as outras aplicações que fazem parte do sistema.

4.3 Tecnologias Habilitantes

A construção de novas ferramentas, programas e sistemas deve sempre levar em conta as tecnologias já existentes, o que pode não apenas diminuir a necessidade de desenvolvimento como prover funcionalidades que passam a ser imprescindíveis para o desempenho adequado do que se está desenvolvendo.

A partir desta idéia um conjunto de tecnologias foi pesquisado e incorporado à solução descrita neste trabalho. Para que a solução final seja apresentada de forma eficaz nos próximos capítulos se faz necessária a sua exposição prévia.

4.3.1 Python

Python é uma linguagem de programação que tem por objetivo permitir o rápido desenvolvimento de aplicações que são suportadas pela maioria dos sistemas operacionais. Além disso, esta linguagem prima por um código que seja de fácil leitura e entendimento [31].

A linguagem Python possui seu código aberto e a sua utilização é livre, o que facilita o desenvolvimento de complementos para o arcabouço e a sua integração com outros programas. Além disso ela suporta diversos paradigmas de programação (como orientado a objeto, funcional e imperativo) e acesso a uma larga biblioteca padrão que oferece interfaces a diversos serviços e operações de sistema.

4.3.2 Distributed Hash Table (DHT)

Tabela de Hash Distribuída (*Distributed Hash Table - DHT*) é um sistema de armazenamento que tem como objetivo armazenar blocos de dados em uma coleção de nós distribuídos, onde cada bloco é identificado por uma chave unívoca.

A meta de uma DHT é distribuir o custo de armazenamento e de fornecimento dos dados entre todos os nós e manter esses dados disponíveis, independentemente de quais nós serão consultados. Além disso, a DHT permite que nós entrem e saiam do sistema a qualquer momento, através da replicação de todos os dados armazenados [32].

4.3.3 Lightweight Directory Access Protocol (LDAP)

O protocolo LDAP [33] foi desenvolvido para permitir pesquisas e a atualização de informações contidas em serviços de diretório. Estes diretórios são formados por conjuntos de objetos com atributos que são arranjados de forma hierárquica.

O armazenamento de objetos desta forma permite que esses diretórios reflitam cadeias de comando, limites geográficos, estruturas organizacionais de empresas, governos e instituições entre outros. As árvores de diretórios não são limitadas a um único objeto, permitindo representar, por exemplo, países, estados, cidades, pessoas e animais em uma mesma árvore.

Esta base de informações difere-se das outras por possuir uma arquitetura pré-definida e um sistema de indexação customizado, possibilitando a criação de uma base de informações com alto desempenho e capacidade de crescimento sem onerar a confiabilidade.

4.4 Soluções Existentes

Em sistemas não distribuídos, o **controle de acesso** é regulado internamente por cada aplicação, contudo, esta abordagem mostra-se extremamente ineficiente em sistemas distribuídos, onde modificações (em recursos, usuários e regras) forçariam a realização de atualizações em cada um dos elementos (do sistema), os quais podem estar separados por milhas de distância.

A separação entre o **serviço de controle de acesso** e a aplicação possui diversas vantagens: facilita a administração e a consolidação das políticas de segurança ao centralizar o **controle de acesso** e reduz a necessidade de desenvolvimento de *software* ao permitir que todos os aplicativos utilizem um único serviço de **controle de acesso**.

Seguindo esta linha, a maioria dos estudos de controles de acesso para sistemas distribuídos podem ser categorizados em três vertentes [01]: servidores de autorização, agentes e proxies. Embora as 3 vertentes possuam as vantagens mencionadas, é possível distinguir suas virtudes quando comparadas.

4.4.1 Servidor de Autorização

O **servidor de autorização** é responsável por receber, processar e responder a requisições de acesso dos programas clientes que fazem parte do sistema distribuído, ou seja, o servidor deve ser capaz de armazenar as políticas de controle de acesso e ao receber requisições de acesso confrontar essas informações (contidas na requisição) com a sua base de políticas e decidir se esta deve ou não ser atendida.

Esta vertente, encontrada em [34] e [35], sofre de dois problemas estruturais, o primeiro é a inserção de um ponto único de falha no sistema distribuído, pois uma

falha no servidor de autorização leva inevitavelmente a apenas duas opções: todas as requisições são atendidas ou todas são negadas.

O segundo problema é a criação de um gargalo, pois todas as requisições devem ser enviadas para um único servidor, limitando o número de transações do sistema e conseqüentemente o desempenho do servidor de autorização.

Estes problemas podem ser combatidos por meio da replicação do **servidor de autorização**, contudo o problema será apenas realocado para a arquitetura de replicação. A utilização de *Web Services* [36] pode ser empregada com o intuito de criar uma arquitetura que combata esta deficiência.

4.4.2 Agentes

A segunda vertente, denominada **agentes**, funciona por meio da compartimentalização do **sistema de controle de acesso**. Cada compartimento (programa ou elemento) do sistema distribuído possui sua própria **ferramenta de controle de acesso**, a qual recebe as informações de autorização a partir de um ponto central, garantindo a concisão da política aplicada em todos os setores.

As regras são construídas de acordo com a política de segurança e de forma que os compartimentos a compreendam, e então são enviadas aos **agentes** localizados em cada setor e que são responsáveis por manter as ferramentas locais de controle de acesso atualizadas.

Um dos principais benefícios desta vertente é permitir que programas legados ou aqueles desenvolvidos fora do ambiente possam ser adequados à política vigente. Além disso, a divisão em compartimentos aumenta a tolerância a falhas do sistema como um todo, oferece uma arquitetura com grande capacidade de crescimento e ainda permite que diferentes **modelos de controle de acesso** sejam utilizados.

As desvantagens desta vertente devem-se principalmente às dificuldades de se atingir consistência nos diferentes compartimentos (diferentes programas, modelo de controle de acesso) e na tradução/mapeamento da política de segurança para as linguagens compreendidas por estes.

Além disso, a **granularidade das regras de controle de acesso** ficam restritas àquela ofertada por cada compartimento, assim como o modelo de controle de acesso utilizado. A distribuição das regras pode ser um processo dispendioso e lento, o que pode levar a inconsistência no **controle de acesso** por um período de tempo.

Esta característica pode incapacitar a utilização destas soluções para ambientes com alta taxa de atualizações da base de autorização, embora a utilização de *capability* [37] e agentes inteligentes [38] possa minimizar ou solucionar este problema.

A utilização de certificados provê grandes benefícios a essa vertente, provendo não apenas autenticação, mas também adicionando segurança à comunicação (e.g. distribuição de regras) como pode ser visto em [38] e [39].

4.4.3 Proxy

A última vertente, proxy, desacopla o módulo de **controle de acesso** da aplicação, o qual passa a ser implementado como um programa à parte, permitindo que seja desenvolvido de forma a atender aos requisitos do sistema. Esta arquitetura exige que todas as requisições sejam enviadas (ou interceptadas e direcionadas) para o monitor de referência.

As requisições podem ser interceptadas enquanto são conduzidas entre serviços através de um sistema de transmissões de objetos, como CORBA [40] e WebSphere Message Queue [41], e direcionadas para o monitor de referência. Outra opção é o

direcionamento das requisições pelas próprias aplicações, as quais apenas atenderiam as requisições após o aval do sistema de controle de acesso.

Um dos benefícios desta vertente é a possibilidade de desenvolvimento de aplicações sem módulos de controle de acesso. Além de diminuir a quantidade de código desenvolvido, permite um maior controle sobre o comportamento (tamanho, desempenho, funcionalidades) do arcabouço de **controle de acesso**, uma vez que este será construído especificamente para atender as necessidades do sistema.

Outro benefício é a oportunidade de adequar aplicações já existentes à política estipulada, exigindo apenas que as requisições dos usuários sejam encaminhadas para o monitor de referências desenvolvido. Contudo, esta funcionalidade deve levar a atrasos quando comparada as soluções que se enquadram na vertente **agente**, uma vez que a comunicação com um segundo serviço passa a ser necessária.

Essa característica oferece uma vantagem em relação à utilização de agentes: As regras não precisam ser adequadas ao controle de acesso de cada setor, o arcabouço de controle de acesso pode ser construído e atualizado para atender a todas as exigências da política de segurança.

A maior dificuldade enfrentada por esta vertente é a consolidação das regras, i.e. base de autorização, uma vez que existirão tantos *proxies* quanto instanciações de programas no sistema. Esta dificuldade pode ser combatida pela solução implementada em [42], onde as permissões são espalhadas em uma rede peer-to-peer.

Outra opção é apresentada em [43] que utiliza envelopes para enviar tanto o pedido do usuário quanto as suas permissões para o monitor de referência, permitindo que a base de autorização seja centralizada. Para garantir a atualização das permissões, cada envelope é válido por apenas um determinado tempo, após o qual outro envelope precisa ser requisitado.

4.5 Considerações Finais

Este capítulo apresentou o ambiente no qual o arcabouço deve ser implementado, permitindo especificação dos requisitos que a solução deve atender, o levantamento das tecnologias que habilitam a sua construção e, por fim, a descrição das soluções já existentes, das quais o arcabouço DRBAC deve posteriormente ser diferenciado.

5 CONTROLE DE ACESSO DISTRIBUÍDO BASEADO EM PAPÉIS (DRBAC - DISTRIBUTED ROLE BASE ACCESS CONTROL)

A observação do intuito do controle de acesso e do ambiente no qual o arcabouço que o implementa estará inserido permitiu o levantamento de dois conjuntos de requisitos ao qual o arcabouço deve atender, os quais foram apresentados e discutidos no capítulo 4.

O primeiro conjunto de requisitos, referentes à política de controle de acesso, será principalmente considerado pelo modelo de controle de acesso escolhido, e o segundo conjunto relativo aos requisitos de um sistema distribuído, será observado especialmente durante a especificação da arquitetura sobre a qual o arcabouço será desenvolvido.

Este capítulo irá apresentar a especificação e desenvolvimento do arcabouço, assim como um conjunto de casos de uso que tencionam mostrar o seu funcionamento.

5.1 Especificação do Sistema de Controle de Acesso Distribuído Baseado em Papeis

Para que a premissa seja atingida, i.e., os requisitos sejam atendidos, é proposto inicialmente a observância de uma arquitetura básica e abstrata de sistemas de controle de acesso e, posteriormente, a sua expansão para uma arquitetura distribuída.

Uma vez que esta propositura se torne verdadeira, os modelos de controle de acessos serão revistos e o que melhor se adequar ao ambiente distribuído será escrutinado para se determinar o seu impacto sobre o desenvolvimento do arcabouço.

A conclusão da especificação dar-se-á com a definição das tecnologias que serão empregadas para implementar cada um dos blocos fundamentais da arquitetura proposta, possibilitando, assim, o desenvolvimento do arcabouço.

5.1.1 Arquitetura

Uma das propostas mais básicas para a arquitetura de arcabouços de controle de acesso [01] é composta por um monitor de referência e uma base de autorização. O monitor de referência é responsável por receber requisições de usuários e determinar se estas devem ou não ser atendidas pelo sistema. A base de autorização possui todas as relações de direito de acesso dos sujeitos sobre os objetos do sistema.

O fluxo de informações nesta arquitetura pode ser visualizado na Figura 4, e descrito da seguinte forma: o monitor de referência recebe um conjunto de informações que possibilitam identificar o autor, a ação e o recurso sobre o qual a ação recairá. Estas identificações permitem que o monitor contate a base de autorização e resgate a relação existente entre estas entidades. A análise desta relação permite ajuizar se a requisição deve ou não ser atendida. Tanto a forma de identificação, quanto a relação entre estas entidades será ditada pelo modelo de controle de acesso escolhido.

Figura 4 – Arquitetura de Controle de Acesso. Adaptado de [1].

Exposto o objetivo do monitor de referência, pode-se afirmar que é inequívoca a sua necessidade presencial ao longo de todo o sistema distribuído. É possível argumentar neste ponto que o mesmo se aplica à base de autorização, no entanto, a

sua ausência implica apenas na impossibilidade de se determinar se um sujeito possui ou não direito a realizar uma ação, circunstância a qual pode e deve ser prevista pela política de acesso. Nesta situação, a presença do monitor de referência pode garantir que a política seja respeitada.

Por este motivo, é recomendável que o **monitor de referência** seja desenvolvido como um módulo independente, capaz de tomar decisões sem a necessidade de se comunicar com qualquer outro módulo do arcabouço e que esteja acoplado a todos os elementos do sistema distribuído, i.e., ainda que se considere a menor porção possível do sistema distribuído, o monitor de referência deverá estar presente e apto a desempenhar o seu papel.

A **base de autorização** sofre impacto tanto do sistema distribuído quanto do modelo de controle de acesso escolhido, uma vez que é este modelo que irá ditar as estruturas de dados que esta base de dados deve suportar. Por consequência, apenas pode-se prever que esta base precisa ser distribuída, contudo sem maiores especificações até que o modelo de controle de acesso tenha sido escolhido.

Embora a proposta limite-se a estes dois componentes, para a construção deste arcabouço também será especificada uma **base de autenticação**. A identificação do usuário é de responsabilidade do **serviço de autenticação**, no entanto, a base de dados utilizada por este serviço serve de suporte ao **arcabouço de controle de acesso** ao fornecer informações sobre os usuários. Esta base também deve compartilhar das características de um sistema distribuído.

Figura 5 – Arquitetura de Controle de Acesso para Sistemas Distribuídos

Esta análise prévia leva à arquitetura visualizada na Figura 5, a qual prevê, como explanado, o desenvolvimento do **monitor de referência** em módulos e a utilização de bases de dados distribuídas para a **base de autorização** e a **base de autenticação**.

5.1.2 Modelo de Controle de Acesso

Ao longo desta dissertação três **políticas de controle de acesso** foram descritas, e embora muitas outras existam, e até mesmo tenham sido desenvolvidas com o intuito de atender aos requisitos de sistema distribuídos, apenas estas três serão analisadas.

Esta decisão é justificada pelo fato de grande parte destas novas políticas existentes serem oriundas das três aqui escrutinadas, e a utilização de uma política original poder garantir a adoção posterior da maioria destas novas políticas. Outro fator importante de decisão é a especificidade destas novas políticas, as quais tendem a focar em um ambiente ou situação em particular, em detrimento da simplicidade e do comum.

O modelo de **política de controle de acesso discricionária** é definida pela capacidade de seus usuários de prover e revogar permissões a terceiros ou grupos. Esta abordagem não é a ideal para ambientes distribuídos dado o grande número de usuários e grupos que estes sistemas tendem a possuir. Seria virtualmente impossível gerenciar e garantir que a política de acesso é respeitada por todos.

O modelo de **política de controle de acesso obrigatória** é bastante eficaz para o ambiente militar, no entanto, o ambiente civil é muito diversificado para este modelo, e mesmo que esta política possa ser incrementada, a mesma foi desenvolvida com o intuito de proteger informações, sem prever a especificidade exigida pelos elementos de um ambiente distribuído.

Não obstante a **política de controle de acesso baseada em papéis** (RBAC) não ter sido desenvolvida especificamente para sistemas distribuídos, esta política consegue responder às ineficiências encontradas nas políticas mencionadas sem possuir, a priori, qualquer outro tipo de desvantagem.

Ao se considerar que no RBAC as permissões são garantidas a papéis, e que apenas a entidade administrativa pode prover acesso a esses papéis, é possível anuir que o número de usuários do sistema não deve impactar na aplicação coercitiva da **política de controle de acesso**. Além disso, o RBAC enxerga os recursos do sistema como objetos, sem impor qualquer tipo de limitação a especificidade destes.

A **política de controle de acesso baseada em papéis** não garante, por si só, o atendimento a todos os requisitos levantados em 4.2, contudo esta também não contradiz nenhum destes. A resposta aos requisitos não atendidos será dada nas seções seguintes, que tencionam completar a especificação do arcabouço ao completar a arquitetura sob a observância da política escolhida.

5.1.3 Blocos Fundamentais do Arcabouço

Dispondo da arquitetura que deve atender aos requisitos de um **sistema distribuído** e do **modelo de controle de acesso** ao qual esta deve suportar, é possível preencher os blocos fundamentais da arquitetura com as tecnologias que irão habilitar a construção do arcabouço.

O **monitor de referência** deve oferecer uma interface simples e de fácil acesso, permitindo que as aplicações possam utilizá-la sem demandar muito trabalho de adaptação para operar com a mesma. Esta proposição exclui interfaces gráficas ou aquelas que para sua utilização exijam uma linguagem de programação específica.

Interfaces de linha de comando são exaltadas pela sua simplicidade, facilidade de integração com outros sistemas (conjunto de entradas e saídas de fácil

interpretação) e por serem suportadas pela maioria das linguagens de programação existentes. Tais características tornam a interface de linha de comando uma ótima opção para o **monitor de referência**.

Sendo a interface do **monitor de referência** de fácil acesso, poucas são as exigências que recaem sobre a linguagem de programação do **monitor de referência**, abrindo uma gama enorme de opções. A escolha da linguagem de programação Python foi baseada principalmente pela sua usabilidade.

A linguagem Python é suportada por virtualmente todos os sistemas operacionais, não exige um compilador (o código é executado após a escrita), e possui uma excelente biblioteca que permite a interação com diferentes serviços de forma simples e rápida. Além disso, o Python prima pela facilidade de leitura do código, característica fundamental em posteriores expansões por facilitar a compreensão do código já desenvolvido.

Uma análise superficial do **modelo de controle de acesso baseado em papéis** revela duas estruturas de dados básicas: a estrutura que suporta a relação de autorização, a qual será armazenada pela base de autorizações e entre usuários e seus papéis, que se encontrará na base de autenticação.

Ainda que estas bases de dados precisem armazenar outras estruturas, estas são as estruturas básicas do modelo, as que serão utilizadas para responder a todas as requisições dos usuários, por conseguinte, justifica-se a observância destas estruturas para suportar a escolha das bases de dados.

No modelo RBAC, autorizações são averiguadas segundo três informações: **papel**, **objeto** e **permissão**. O **monitor de referência** deve prover estas três informações para obter a informação ('permitido' ou 'não permitido') que determinará se uma requisição feita ao sistema deve ou não ser atendida. Estas três informações podem ser entendidas como chaves que precisam ser fornecidas para se adquirir uma informação, i.e., são informações que podem ser armazenadas na primeira forma normal (*first normal form* – 1NF).

Dada as características levantadas para a **base de autorização**, armazenamento de informações de forma distribuída e no formato de tabelas na primeira forma normal, a opção por tabelas de *hash* distribuído (DHT) se torna óbvia. As informações armazenadas em uma DHT são replicadas e depois dispersas entre os elementos que a compõe, provendo a esse serviço exatamente as características que se busca em um sistema distribuído: capacidade de crescimento, disponibilidade e confiabilidade.

Com o objetivo de diminuir a latência das requisições direcionadas a DHT é proposta a utilização do programa memcached [30], o qual é uma implementação de código aberto que armazena de forma distribuída tabelas 1NF (primeira forma normal) na memória de acesso rápido. Esta forma de armazenamento remove a latência da busca de informações no disco rígido.

O foco do arcabouço aqui descrito é prover um **serviço de controle de acesso**, limitando-se a apenas sugerir uma base de usuários que não insira um ponto único de falha no arcabouço ou crie gargalos no sistema. Além destes requisitos, a base também deve permitir o armazenamento dos usuários e dos papéis designados a este.

Embora estes requisitos possam ser atendidos por um serviço de DHT, essa escolha poderia limitar severamente as opções de **serviços de autenticação**, uma vez que este serviço armazena informações apenas na primeira forma normal. Tendo este objetivo em foco, de prover maior suporte (estruturas de dados mais robustas) ao **serviço de autenticação** que será utilizado em conjunto com o arcabouço, propõe-se a utilização de um **serviço de diretório distribuído**.

A utilização de **serviços de diretório** como base de dados para o **serviço de autenticação** não é uma novidade e possui diversas soluções como: *Active Directory da Microsoft* [32], e a união do *Pluggable Authentication Module* (PAM) com *OpenLDAP* [31].

O servidor OpenLDAP é um programa de código aberto, que possui vários utilitários e bibliotecas para diversas linguagens de programação, além de suportar inúmeras

arquiteturas de replicação que podem prover as características exigidas (e.g. um servidor principal e diversos servidores escravos ou diversos servidores principais) pelo sistema distribuído.

A Figura 6 permite a visualização destes componentes e do fluxo de informações, o qual pode ser resumido da seguinte maneira:

- Passo 0 – Antes que um usuário possa fazer uma operação ao sistema, é necessário criar uma sessão e requisitar a ativação de um ou mais papéis. Na requisição é enviada a identificação do usuário (*userid*) e dos papéis (*roleid*);
- Passos 1 e 2 – o **monitor de referência** cria uma sessão e comunica-se, então, com o **serviço de diretórios** para descobrir o conjunto de papéis ao qual o usuário pode ter acesso;
- Passo 3 – o **monitor de referência** ativa os papéis requisitados e confirma a ativação ao sistema, possibilitando que requisições de autorizações sejam executadas;
- Passo 4 – O sistema recebe uma requisição de um usuário e a envia ao **monitor de referência**. Esta requisição deve conter a identificação da sessão, do objeto (*objid*) e da operação (*opid*);
- Passo 5 e 6 – O **monitor de referência** comunica-se, então, com o **serviço DHT** e obtém as autorizações para todos os papéis ativos na sessão especificada, i.e., são solicitadas as informações armazenadas nas chaves pertencentes ao conjunto formado por <todos os papéis x operação x objeto>;
- Passo 7 – Como resultado, o **monitor de referência** dará uma de duas respostas possíveis: ‘permissão concedida’ (*granted*) se pelo menos um dos papéis possuir permissão para realizar a operação desejada sobre o objeto especificado, ou ‘permissão negada’ (*not granted*) caso contrário;

Figura 6 - Arquitetura DRBAC

Essa arquitetura garante que o número de elementos do **serviço de controle de acesso** possa crescer junto com o **sistema distribuído**, uma vez que o monitor é um cliente que pode ser acoplado a qualquer elemento, e as **bases de autorização e autenticação** podem ter seus recursos incrementados para atender a demanda exigida.

5.2 Desenvolvimento do arcabouço DRBAC

A especificação do RBAC provida pela ANSI [22] não apenas cobre o modelo, mas também define um conjunto de funções que devem ser implementadas por sistemas que utilizem o modelo. O arcabouço DRBAC seguirá esta especificação, propondo apenas pequenas modificações que tendem a facilitar a sua utilização na arquitetura definida.

As funções especificadas foram agrupadas em três grupos (tradução livre): **Funções Administrativas**, **Funções de Análise** e **Funções de Sistema**.

As **Funções Administrativas** possibilitam a criação e a manutenção do conjunto de elementos e relações que compõem o modelo, enquanto que as **Funções de Análise** permitem a realização de pesquisas administrativas sobre os mesmos. As **Funções de Sistema** tornam possível a criação e gerenciamento dos atributos RBAC nas sessões de usuários e a tomada de decisões de controle de acesso.

Contudo, dada a arquitetura construída, será proposta uma nova divisão: **Funções Administrativas** e **Funções de Sistema**. Todas as funções previamente enquadradas nas Funções Administrativas, Funções de Análise e Funções de Sistema farão parte do conjunto formado pela união dos dois novos conjuntos (**Funções Administrativas** e **Funções de Sistema**). Estes dois conjuntos de funções serão implementados em dois módulos diferentes.

Essa separação tem o intuito de permitir que cada módulo seja utilizado apenas quando de sua necessidade, ou seja, o módulo com as **Funções de Sistema** será utilizado sempre que um usuário desejar interagir com o sistema, e o módulo com as **Funções Administrativas** será ativado apenas para se realizar modificações na **política de controle de acesso**.

A seguir será apresentada uma lista com os dois grupos de funções. Vale observar que algumas funções sofrerão pequenas modificações com o intuito de melhor adaptar o conjunto a arquitetura definida. Outras funções serão adicionadas aos dois grupos (sofrendo pequenas modificações para isso) para acomodar a divisão do **monitor de referência** em dois módulos; além de funções adicionadas para permitir a criação de elementos do sistema (objetos) ou sanar possíveis problemas de consistência devido a falhas de comunicação. Na descrição de cada função modificada ou adicionada será explanado em maiores detalhes o que motivou tais intervenções.

Com o intento de facilitar a descrição das funções será utilizada a notação apresentada a seguir:

- [usuários], [papéis], [objetos] e [operações]: conjunto de usuários, papéis, objetos e operações existentes no sistema.
- [sessões]: conjunto de sessões ativas no **monitor de referência**.
- [usuário: papéis]: conjunto de papéis ao qual o usuário tem acesso.
- [objeto: operações]: conjunto de operações que podem ser realizadas sobre este objeto.
- <papel, objeto, operação>: permissão para que o papel realize a operação sobre o objeto.

- [papel: permissões]: conjunto de permissões garantidas ao papel.

5.2.1 Funções Administrativas

Como **Funções Administrativas** tem-se:

- **AddUser**(usuário, senha, primeiro nome, último nome): Adiciona um usuário à **base de autenticação**, estabelecendo um nome de usuário, senha, primeiro e último nome e a sua localização.

Condições:

$\text{usuário} \notin [\text{usuários}]$

- **DeleteUser**(usuário): Apaga o usuário identificado pelo nome de usuário fornecido da **base de autenticação**.

Condições:

$\text{usuário} \in [\text{usuários}]$

- **AddRole**(papel): Cria um novo papel com o nome indicado no argumento da operação. Não é possível criar um papel e adicionar permissões a este em uma única ação.

Condições:

$\text{papel} \notin [\text{papéis}]$

- **DeleteRole**(papel): Apaga o papel identificado no argumento da operação. Como uma permissão é identificada pela tríade <papel x operação x objeto> não é necessário apagar todas as permissões que o papel possui, pois uma permissão só é avaliada se todos os três identificadores forem válidos.

Condições:

$\text{papel} \in [\text{papéis}]$

- **AssignUser**(usuário, papel): Esta função tem o objetivo de adicionar o papel indicado ao conjunto de papéis do usuário.

Condições:

$\text{usuário} \in [\text{usuários}]$

$\text{papel} \in [\text{papéis}]$

$\text{papel} \notin [\text{usuário: papéis}]$

- **DeassignUser**(usuário, papel): Esta função tem o intuito contrário da função anterior, ou seja, tenciona retirar um papel do conjunto de papéis de um usuário.

Condições:

usuário $\in$ [usuários]

papel $\in$ [papéis]

papel $\in$ [usuário: papéis]

- **GrantPermission**(papel, objeto, operação): Esta função intenta conceder a permissão para que um papel execute uma determinada operação sobre um objeto.

Condições:

papel $\in$ [papéis]

objeto $\in$ [objetos]

operação $\in$ [objeto: operações]

$\langle$ papel, objeto, operação $\rangle \notin$ [papel: permissões]

- **GrantPermissionConditional**(papel, objeto, operação, condição): Esta função intenta conceder a permissão para que um papel execute uma determinada operação sobre um objeto desde que uma determinada condição seja alcançada.

Condições:

papel $\in$ [papéis]

objeto $\in$ [objetos]

operação $\in$ [objeto: operações]

$\langle$ papel, objeto, operação $\rangle \notin$ [papel: permissões]

- **RevokePermission**(papel, objeto, operação): Esta função objetiva revogar a permissão de um papel executar uma determinada operação sobre um objeto.

Condições:

papel $\in$ [papéis]

objeto $\in$ [objetos]

operação $\in$ [objeto: operações]

$\langle$ papel, objeto, operação $\rangle \in$ [papel: permissões]

- **AddObject**(objeto): Esta função foi adicionada com o objetivo de suportar a criação de identificadores de objetos e permitir a criação de um ambiente para testes.

Condições:

$\text{objeto} \notin [\text{objetos}]$

- **DeleteObject** (objeto): Esta função foi adicionada com o objetivo de suportar a remoção de identificadores de objetos e permitir a criação de um ambiente para testes.

Condições:

$\text{objeto} \in [\text{objetos}]$

- **RoleOperationsOnObject**(papel, objeto): Esta função retorna o conjunto de operações que o papel pode executar sobre o objeto.

Condições:

$\text{objeto} \in [\text{objetos}]$

$\text{papel} \in [\text{papéis}]$

- **UserOperationsOnObject**(usuário, objeto): Esta função retorna o conjunto de operações que um usuário, através do seu conjunto de papéis, pode executar sobre o objeto.

Condições:

$\text{objeto} \in [\text{objetos}]$

$\text{usuário} \in [\text{usuários}]$

- **AssignedUsers**(papel): Esta função retorna o conjunto de usuários que podem assumir o papel.

Condições:

$\text{papel} \in [\text{papéis}]$

- **AssignedRoles**(usuário): Esta função retorna o conjunto de papéis que o usuário pode assumir.

Condições:

$\text{usuário} \in [\text{usuários}]$

- **ListRoles**(): Esta função foi adicionada com o objetivo de visualizar todo o conjunto papéis e facilitar a monitoração do ambiente. Esta função não aceita parâmetros e não possui condições a serem atendidas.

Condições:

Não há.

- **RolePermissions**(papel): Esta função retorna o conjunto de permissões que o papel possui, ou seja, todos os objetos e operações que podem ser realizada sobre esses.

Condições:

$\text{papel} \in [\text{papéis}]$

- **UserPermissions**(usuário): Esta função retorna o conjunto de permissões que o usuário possui, as quais serão mapeadas a partir do conjunto de papéis que o usuário pode assumir.

Condições:

$\text{usuário} \in [\text{usuários}]$

- **listCommans**(): Esta função foi adicionada apenas para listar as funções aqui descritas facilitando a utilização do arcabouço. Não existem condições a serem atendidas.

Condições:

Não há.

5.2.2 Funções do Sistema

Como **Funções do Sistema** tem-se:

- **Identify**(usuário, senha): Esta função foi adicionada com o objetivo de criar uma forma de identificar usuários e permitir a criação de um ambiente para testes.

Condições:

$\text{usuário} \in [\text{usuários}]$

- **CreateSession**(sessão): Esta função tem o objetivo de iniciar uma sessão para o usuário.

Condições:

$\text{sessão} \notin [\text{sessões}]$

- **DeleteSession**(sessão): Esta função tem o objetivo de encerrar uma sessão para o usuário.

Condições:

$\text{sessão} \in [\text{sessões}]$

- **AddActiveRole**(sessão, papel): Esta função tem o objetivo de ativar um papel naquela sessão.

Condições:

$\text{sess\~ao} \in [\text{sess\~oes}]$

$\text{papel} \in [\text{papeis}]$

- **DropActiveRole**(sess\~ao, papel): Esta fun\~cao tem o objetivo de desativar um papel da sess\~ao.

Condições:

$\text{sess\~ao} \in [\text{sess\~oes}]$

$\text{papel} \in [\text{papeis}]$

- **CheckAccess**(sess\~ao, objeto, opera\~cao): Esta fun\~cao retorna o conjunto de opera\~oes que a sess\~ao, através do conjunto de papéis ativos, permite realizar sobre o objeto.

Condições:

$\text{sess\~ao} \in [\text{sess\~oes}]$

$\text{objeto} \in [\text{objetos}]$

$\text{opera\~cao} \in [\text{opera\~oes}]$

- **SessionRoles**(sess\~ao): Esta fun\~cao retorna o conjunto de papéis ativos na sess\~ao.

Condições:

$\text{sess\~ao} \in [\text{sess\~oes}]$

- **AssignedRoles**(): Esta fun\~cao foi modificada para também fazer parte das fun\~coes do sistema: retorna apenas o conjunto de papéis que podem ser assumidos pelo usuáριο identificado. Como o usuáριο já deve estar identificado para utilizar a fun\~cao, não existem condi\~oes a serem atendidas.

Condições:

Não há.

- **SessionPermissions**(sess\~ao): Esta fun\~cao retorna o conjunto de permissões que a sess\~ao, por meio do conjunto de papéis ativos, permite realizar.

Condições:

$\text{sess\~ao} \in [\text{sess\~oes}]$

- **listCommans**(): Esta fun\~cao foi adicionada apenas para listar as fun\~coes aqui descritas facilitando a utiliza\~cao do arcabou\~co. Não existem condi\~oes a serem atendidas.

Condições:

Não há.

A especificação das funções que o arcabouço deve possuir permite o levantamento das **estruturas de dados** que irão consolidar as informações manipuladas por estas funções.

A escolha da **base de dados** que irá implementar a base de autorização foi tomada considerando-se apenas a estrutura que armazena a autorização em si, a qual exigia apenas estruturas de dados na primeira forma normal.

Um dos principais benefícios desta forma é a sua simplicidade, o que levou ao desenvolvimento de diversas implementações com arquiteturas distribuídas (DHT), contudo esta estrutura impõe um ônus que pode levar a problemas de desempenho: a falta de suporte à criação de relações entre as informações.

Além disso, o funcionamento da DHT dita que uma informação só pode ser fornecida frente à apresentação da sua chave (índice), e ao se considerar o tamanho do espaço de chaves, é razoável afirmar que é virtualmente impossível procurar por uma informação tentando utilizar como índice da informação cada elemento deste espaço.

Essa característica exige do sistema, que utiliza o arcabouço, a conservação de listas com: todos os objetos, operações e papéis existentes, os quais são necessários para permitir a listagem de todas as permissões do sistema, algo indispensável para a execução de determinadas funções (e.g., SessionPermissions, UserPermissions e RoleOperationsOnObject).

Mesmo esta solução iria impor uma grande carga ao sistema, uma vez que todas as permissões possíveis precisariam ser resgatadas para se encontrar apenas aquelas que de fato seriam úteis. Para solucionar este problema foram criadas mais duas estruturas de dados, as quais permitem **relacionar os papéis com os objetos e suas operações**.

A primeira estrutura utiliza o **identificador do objeto** como chave para a obtenção de todas as operações que podem ser executadas sobre o objeto. A segunda

estrutura utiliza o **nome do papel** como chave e armazena o identificador de todos os objetos sobre os quais o papel possui algum tipo de permissão.

Estas estruturas são suficientes para reduzir o espaço de chaves de todas as possibilidades possíveis, para apenas o espaço produzido por todas as operações que podem ser realizadas sobre os objetos ao qual o papel de fato possui acesso, ao ônus de apenas duas consultas ao **banco de autorizações**.

As estruturas de dados do banco de autorização podem ser vistas na Tabela 1:

Chave	Informação
papel x objeto x operação	Permissão concedida ou não.
papel	Lista de todos os objetos (LOB) sobre o qual este papel provê algum tipo de acesso.
objeto	Lista de todas as operações (LOP) que podem ser executadas sobre este papel.

Tabela 1 – Estruturas de dados da base de autorização.

A utilização destas estruturas não apresenta nenhum ônus à operação normal do **monitor de referência**, com exceção da alteração de permissões de um papel. Sempre que uma permissão for concedida a um novo objeto, este precisa ser adicionado à **lista de objetos do papel** (LOB), e ao remover a última permissão de acesso de um papel a um objeto, este também precisa ser removido da lista.

A **lista de objetos do papel** (LOB) apenas sofrerá modificações quando um novo objeto for adicionado ou um objeto for removido. No caso de um novo objeto ser adicionado ao sistema, além do seu identificador, o administrador também precisará adicionar as operações suportadas por este, as quais irão compor a **lista de operações do objeto** (LOP).

Uma vez especificada as estruturas de dados da **base de autorização**, o mesmo deve ser realizado para a **base de autenticação**, a qual é responsável por armazenar todas as informações necessárias ao **serviço de autenticação**. Embora este serviço não seja oferecido pelo arcabouço DRBAC, é válido salientar que a estrutura de dados suportada pelo serviço LDAP é extremamente rica e pode ser

explorada para refletir até mesmo a estrutura organizacional da empresa proprietária do sistema distribuído.

O arcabouço, no entanto, exige que cada cadastro de usuário também possua uma lista com todos os papéis que o usuário pode personificar. A lista foi implementada através de um diretório de *strings* que foi adicionado ao **banco de dados do serviço LDAP** através do *schema* apresentado abaixo:

```
attributetype ( 1.3.6.1.4.1.4203.666.1.90
    NAME 'role'
    DESC 'Role associated with this individual'
    EQUALITY caseIgnoreMatch
    SUBSTR caseIgnoreSubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

Este *schema* adiciona um novo atributo aos **diretórios do serviço LDAP** e especifica um identificador para o atributo, o nome (NAME), uma descrição (DESC), o tipo do atributo (SYNTAX) e as regras de comportamento (EQUALITY E SUBSTR) deste atributo quando utilizado em pesquisas (e.g. pesquisa por todos os papéis de um usuário, ou por todos os usuários que possuem acesso a um determinado papel) na **base de autenticação**.

A descrição da arquitetura na seção 5.1.1, do **modelo de controle de acesso** em 5.1.2, dos programas que compõem os blocos fundamentais do arcabouço em 5.1.3 e da concepção do **monitor de referência** e das **bases de autorização e autenticação** compõem a especificação do **arcabouço de controle de acesso** para sistemas digitais, o DRBAC.

5.3 Testes

A realização de testes tende a ser um processo longo e necessariamente exaustivo, motivo pelo qual a utilização de métodos que facilitem a sua realização e análise são incentivados.

Com este objetivo, primeiramente será utilizada uma série de casos de uso que intentam testar o arcabouço em diferentes situações e ao mesmo tempo mostrar o seu funcionamento. Posteriormente será apresentado dois testes de desempenho que buscam observar o arcabouço frente a diferentes cargas e o seu comportamento ao atingir seus limites.

5.3.1 Casos de uso

O primeiro caso de uso será a criação de um ambiente com usuários, objetos e suas respectivas operações, papéis e permissões. O ambiente final pode ser visualizado através das estruturas de dados que estarão representadas nas tabelas a seguir, as quais servirão como guias para os casos de uso seguintes.

O ambiente possuirá o cadastro de três usuários: Usuário A, Usuário B e Usuário C (vide Tabela 2).

Usuário	Papéis
UsuárioA	Suporte de Redes; Suporte de Armazenamento
UsuárioB	Administrador Web; Suporte de Armazenamento; Administrador de Armazenamento
UsuárioC	Administrador de Armazenamento

Tabela 2 - Usuários e Papéis do Ambiente de Casos de Uso

A Tabela 3 apresenta os objetos e as palavras chaves através das quais os objetos são referenciados no arcabouço.

Objeto	Identificador
Disco Rígido 0	hd0
Disco Rígido 1	hd1
Diretório Web	dirweb
Diretório Backup	dirbcp

Servidor Web N	webservern
Servidor de Armazenamento 0	datapool0
Link0	link0
Link1	link1
Serviço iSCSI 0	ldatapool0
Roteador A	roteadora

Tabela 3 - Identificadores de Objetos

A Tabela 4 apresenta as operações e os identificadores através dos quais estas operações são referenciadas no arcabouço.

Operação	Identificador
Formatar	formatar
Desativar	desativar
Ativar	ativar
Escrever	escrever
Ler	ler
Configurar	configurar
Configurar protocolos	confproto
Adicionar diretórios	adicionadir
Configurar rotas	confrotas
Disponibilizar para uso	particionar
Realizar backups	backup

Tabela 4 - Identificadores de Operações

A Tabela 5 informa quais operações podem ser realizadas sobre os objetos existentes:

Objeto	Operações
hd0	ativar;desativar;formatar
hd1	ativar;desativar;formatar
dirweb	ler;escrever;backup
dirbkp	ler;escrever;backup

webservern	ativar;desativar;configurar
datapool0	ativar;desativar;particionar
link0	ativar;desativar
link1	ativar;desativar
idatapool0	ativar;desativar;particionar
roteadora	ativar;desativar;confproto;confrotas;backup

Tabela 5 - Relação entre objetos e operações

A tabela de permissões (Tabela 6) irá mostrar apenas as permissões concedidas, qualquer outra permissão deve ser negada pelo arcabouço.

Identificador de Permissões	Permissões
Suporte_de_Redes x link0 x ativar	Concedida
Suporte_de_Redes x link0 x desativar	Concedida
Suporte_de_Redes x roteadora x confrotas	Concedida
Suporte_de_Armazenamento x roteadora x backup	Concedida
Suporte_de_Armazenamento x dirweb x backup	Concedida
Suporte_de_Armazenamento x hd0 x formatar	Concedida
Suporte_de_Armazenamento x hd1 x formatar	Concedida
Suporte_de_Armazenamento x datapool0 x particionar	Concedida
Suporte_de_Armazenamento x idatapool0 x particionar	Concedida
Administrador_de_Armazenamento x dirbcp x escrever	Concedida
Administrador_de_Armazenamento x dirbcp x ler	Concedida
Administrador_de_Armazenamento x datapool0 x ativar	Concedida
Administrador_de_Armazenamento x datapool0 x desativar	Concedida
Administrador_de_Armazenamento x idatapool0 x ativar	Concedida
Administrador_de_Armazenamento x idatapool0 x desativar	SeparacaoDeveres
Administrador_Web x dirweb x ler	Concedida
Administrador_Web x dirweb x escrever	Concedida
Administrador_Web x webservern x ativar	Concedida
Administrador_Web x webservern x desativar	Concedida
Administrador_Web x webservern x configurar	Concedida

Tabela 6 - Permissões do Ambiente de Casos de Uso

5.3.1.1 Caso de uso de criação do Ambiente

A criação do ambiente será realizada por meio da manipulação das funções do módulo que executa as funções administrativas, a partir da interface de linha de comando. Os comandos utilizados e suas respostas seguem abaixo:

Welcome to dBAC Manager, your access control manager

Queries format: addUser(userid, password, fistname, surname)

listCommands()

addUser(usuarioa,senhaa)

User 'usuarioa' added.

addUser(usuariob,senhac)

User 'usuariob' added.

addUser(usuarioc,senhac)

User 'usuarioc' added.

addRole(Suporte_de_Redes)

Role 'Suporte_de_Redes' created successfully

addRole(Suporte_de_Armazenamento)

Role 'Suporte_de_Armazenamento' created successfully

addRole(Administrador_Web)

Role 'Administrador_Web' created successfully

addRole(Administrador_de_Armazenamento)

Role 'Administrador_de_Armazenamento' created successfully

addObject(hd0,ativar;desativar;desativar;formatar)

Object 'hd0' created successfully with the following operations:

['ativar', 'desativar', 'desativar', 'formatar']

deleteObject(hd0)

Object 'hd0' deleted

addObject(hd0,ativar;desativar;formatar)

Object 'hd0' created successfully with the following operations:
 ['ativar', 'desativar', 'formatar']
addObject(hd0,ativar;desativar)
 Object already exists
addObject(hd1,ativar;desativar;formatar)
 Object 'hd1' created successfully with the following operations:
 ['ativar', 'desativar', 'formatar']
addObject(dirweb,ler;escrever;backup)
 Object 'dirweb' created successfully with the following operations:
 ['ler', 'escrever', 'backup']
addObject(dirbkp,ler;escrever;backup)
 Object 'dirbkp' created successfully with the following operations:
 ['ler', 'escrever', 'backup']
addObject(webservern,ativar;desativar;configurar)
 Object 'webservern' created successfully with the following operations:
 ['ativar', 'desativar', 'configurar']
addObject(datapool0,ativar;desativar;particionar)
 Object 'datapool0' created successfully with the following operations:
 ['ativar', 'desativar', 'particionar']
addObject(link0,ativar;desativar)
 Object 'link0' created successfully with the following operations:
 ['ativar', 'desativar']
addObject(link1,ativar;desativar)
 Object 'link1' created successfully with the following operations:
 ['ativar', 'desativar']
addObject(idatapool0,ativar;desativar;particionar)
 Object 'idatapool0' created successfully with the following operations:
 ['ativar', 'desativar', 'particionar']
addObject(roteadora,ativar;desativar;confproto;confrotas;backup)
 Object 'roteadora' created successfully with the following operations:
 ['ativar', 'desativar', 'confproto', 'confrotas', 'backup']

Além de mostrar interação entre um usuário com o módulo de administração, esta cadeia de eventos permite a visualização do comportamento do arcabouço a introdução de erros (tentativa de criação de cadastro duplo de papéis e objetos), e da sintaxe esperada pelo módulo na utilização de funções.

Esta etapa do caso de uso exigiu tanto a interação do módulo com a DHT (criação de objetos e papéis) quanto com o serviço LDAP (criação de usuários), proporcionando a oportunidade de se testar a comunicação entre todos os elementos da arquitetura.

A segunda e última etapa deste caso de uso é a criação das permissões, designação dos papéis e posterior verificações, visualizada novamente pela interação do usuário com a interface de linha de comando:

```
# assignUser(usuarioa,Suporte_de_Redes)
Role 'Suporte_de_Redes' already assigned to subject 'usuarioa'
# assignUser(usuarioa,Suporte_de_Armazenamento)
Role 'Suporte_de_Armazenamento' already assigned to subject 'usuarioa'
# assignUser(usuariob,Administrador_Web)
Role 'Administrador_Web' already assigned to subject 'usuariob'
# assignUser(usuariob,Administrador_de_Armazenamento)
Role 'Administrador_de_Armazenamento' already assigned to subject 'usuariob'
# assignUser(usuariob,Suporte_de_Armazenamento)
Role 'Suporte_de_Armazenamento' already assigned to subject 'usuariob'
# assignUser(usuarioc,Administrador_de_Armazenamento)
Role Administrador_de_Armazenamento assigned to subject usuarioc
# grantPermission(Suporte_de_Redes,link0,ativar)
Permission for 'Suporte_de_Redes' to execute 'ativar' over 'link0' granted.
# grantPermission(Suporte_de_Redes,link0,desativar)
Permission for 'Suporte_de_Redes' to execute 'desativar' over 'link0' granted.
...
# grantPermissionConditional(Administrador_de_Armazenamento,idatapool0,
desativar,SeparacaoDeveres)
```

Permission for 'Administrador_de_Armazenamento' to execute 'desativar' over 'idatapool0' granted over condition.

grantPermission(Administrador_Web,webservern,desativar)

Permission for 'Administrador_Web' to execute 'desativar' over 'webservern' granted.

grantPermission(Administrador_Web,webservern,configurar)

Permission for 'Administrador_Web' to execute 'configurar' over 'webservern' granted.

assignedUsers(Suporte_de_Armazenamento)

Subjects assigned to role 'Suporte_de_Armazenamento':

['usuarioa', 'usuariob']

assignedUsers(Administrador_Web)

Subjects assigned to role 'Administrador_Web':

['usuariob']

assignedRoles(usuarioa)

Subject 'usuarioa' possible roles are:

['Suporte_de_Redes', 'Suporte_de_Armazenamento']

roleOperationsOnObject(Suporte_de_Armazenamento,roteadora)

Role 'Suporte_de_Armazenamento' has the following permissions over object 'roteadora':

['backup']

listRoles

Existing roles:

['Suporte_de_Redes', 'Suporte_de_Armazenamento', 'Administrador_Web', 'Administrador_de_Armazenamento']

userPermissions(usuarioa)

User 'usuarioa' may execute the following operations:

Through role 'Suporte_de_Redes' over object 'roteadora':

['confrotas']

Through role 'Suporte_de_Redes' over object 'link0':

['ativar', 'desativar']

Through role 'Suporte_de_Armazenamento' over object 'roteadora':

['backup']

Through role 'Suporte_de_Armazenamento' over object 'hd1':

['formatar']

Through role 'Suporte_de_Armazenamento' over object 'hd0':

['formatar']

Through role 'Suporte_de_Armazenamento' over object 'dirweb':

['backup']

Through role 'Suporte_de_Armazenamento' over object 'idatapool0':

['particionar']

Through role 'Suporte_de_Armazenamento' over object 'datapool0':

['particionar']

Algumas interações foram omitidas com o intuito de não poluir o texto, uma vez que consistem da execução da mesma função e a averiguação do correto funcionamento acontece nos passos seguintes.

Esta última etapa mostrou não apenas o fim da construção do ambiente como também a verificação da maioria das ações tomadas para que isso ocorresse, o próximo caso de uso permitirá que as outras ações também sejam verificadas, além de mostrar o funcionamento do módulo de funções do sistema.

5.3.1.2 Caso de uso de utilização do arcabouço por um usuário

Uma vez que o ambiente já foi construído, será mostrado um conjunto de tentativas do usuário A de executar funções sobre os objetos do sistema. Além das operações normais, serão inseridas operações com erros lógicos (e.g. operações que não existem) e de sintaxe (e.g. erro no formato dos parâmetros das funções).

Este caso de uso mostra o usuário B tentando realizar um conjunto de operações web, de redes e armazenamento, e, em caso de erro, as ações que o usuário executaria para tentar descobrir o motivo destes.

Welcome to dRBAC, your access control framework

Queries format: identify(user, password)

listCommands()

identify(usuariob, senhaa)

Wrong password for subject usuariob

identify(usuariob, senhab)

Subject usuariob recognized.

assignedRoles()

Your possible roles are:

['Administrador_Web', 'Administrador_de_Armazenamento',
'Suporte_de_Armazenamento']

addActiveRole(sessiona,Administrador_Web)

The subject should create a session before activating a role

createSession(sessiona)

Session sessiona created successfully

addActiveRole(sessiona,Administrador_Web)

Active roles:

['Administrador_Web']

checkAccess(sessiona,dirweb,ler)

Permission granted.

checkAccess(sessiona,webservern,desativar)

Permission granted.

checkAccess(sessiona,dirweb,escrever)

Permission granted.

checkAccess(sessiona,webservern,ativar)

Permission granted.

checkAccess(sessiona,dirweb,backup)

Permission not granted.

sessionRoles(sessiona)

Active roles:

['Administrador_Web']

addActiveRole(sessiona,Suporte_de_Armazenamento)

Active roles:

['Administrador_Web', 'Suporte_de_Armazenamento']

checkAccess(sessiona,dirweb,backup)

Permission granted.

createSession(sessionb,Administrador_de_Armazenamento)

Wrong number of arguments for query: createSession

createSession(sessionb)

Session sessionb created successfully

addActiveRole(sessionb,Administrador_de_Armazenamento)

Active roles:

['Administrador_de_Armazenamento']

checkAccess(sessiona,idatapool0,ativar)

Permission not granted.

sessionRoles(sessiona)

Active roles:

['Administrador_Web', 'Suporte_de_Armazenamento']

sessionRoles(sessionb)

Active roles:

['Administrador_de_Armazenamento']

checkAccess(sessionb,idatapool0,ativar)

Permission granted.

checkAccess(sessiona,dirweb,ler)

Permission granted.

checkAccess(sessiona,dirweb,backup)

Permission granted.

checkAccess(sessionb,idatapool0,desativar)

This action requires a second user, please identify (user, password): (**usuarioc, senhac**)

Subject usuarioc recognized.

Permission granted.

deleteSession(sessiona)

Session sessiona was successfully deleted

deleteSession(sessionb)

Session sessionb was successfully deleted

Todas as respostas estão de acordo com o ambiente construído e os erros lógicos e de sintaxe foram corretamente encontrados e notificados. Este caso de uso mostra o

funcionamento das sessões, e da ativação de vários papéis possibilitando a realização de tarefas com o papel de menor privilégio.

5.3.2 Testes de Desempenho

O primeiro desafio a ser enfrentado na realização de testes em ambientes distribuído é exigência dos recursos computacionais que devem compor o ambiente. Para superar este primeiro desafio é proposta a utilização de máquinas virtuais, solução que além de minimizar a quantidade de recursos reais exigido, diminui a complexidade do sistema a ser observado.

Ao se adotar um ambiente virtual o numero de participantes diminui, uma vez que deixa de ser necessário a utilização de cabos de rede, switchs, roteadores, etc., além de permitir uma maior homogeneidade nos elementos que de fato compõem o sistema distribuído. Essa redução de recursos reais além de tornar o ambiente mais barato, diminui a chance de interferências por recursos defeituosos, facilitando a análise de resultados.

O ambiente de testes será composto por três máquinas virtuais, todas instanciadas pelo mesmo monitor de maquina virtuais, utilizando recursos reais de uma única maquina hospedeira. Uma máquina hospeda os serviços de DHT e OpenLDAP, base de autorização e autenticação respectivamente, a qual possui um único processador e 512MB de memória RAM, e duas máquinas que possuem o monitor de referência DRBAC, as quais possuem apenas 256MB de memória RAM e um único processador. Todas as máquinas virtuais utilizam o sistema operacional Fedora 13, Kernel 2.6.33.3-85.fc13.i686.

A quantidade de recursos limitado que cada máquina virtual (elemento do sistema) possui não se mostra um problema uma vez que a principal informação que se espera obter dos testes de desempenho do sistema são exatamente sobre o seu comportamento quando submetido a uma carga maior do que os seus recursos atuais podem lidar, e se o incremento de recursos permitem o processamento de uma carga ainda maior.

O primeiro teste realizado, apresentado no gráfico da Figura 7, foi realizado utilizando uma máquina cliente (com o monitor de referência DRBAC) e uma máquina servidor (com o serviço DHT e LDAP). Este teste apresenta a instanciação de diversos processos simultaneamente, que realizam 250 operações de requisição de autorização.

Figura 7 – Gráfico com um cliente e um servidor

Este teste fornece duas informações importantes, primeiro que a instanciação de diversos monitores de referência é possível, uma vez que todas as respostas foram recebidas com sucesso, e mais importante que o número de clientes (mais especificamente, o número de instancias do monitor de referência) não influi na quantidade de operações realizadas, ou seja, mesmo que o incremento no número de requisições não leva a um comportamento inadequado do sistema (e.g. um gráfico exponencial).

O segundo teste apresentado utiliza duas máquinas clientes e uma máquina servidor. O intuito deste teste é incrementar gradativamente o número de operações de um cliente e depois comparar esse resultado com o obtido ao impor a mesma carga a partir de dois clientes sobre o mesmo servidor. O resultado destes testes pode ser observado na Figura 8.

Figura 8 – Gráfico com um servidor e dois clientes

O primeiro teste realizado (um único cliente) forneceu uma velocidade máxima e constante de 1520 operações por segundo. A informação mais importante fornecida por este gráfico é o comportamento dos pontos encontrados, o qual se aproxima de uma reta, mostrando que mesmo com o aumento do número de requisições o sistema se comporta normalmente (mesma velocidade de resposta).

O segundo conjunto de pontos encontrados é apresentado na reta A – 2 Clientes. Este conjunto foi obtido ao se utilizar a mesma carga de testes em dois clientes simultaneamente, ou seja, ao invés de apenas um cliente requisitar autorizações ao servidor, a mesma quantidade de requisições são iniciadas ao mesmo tempo em dois clientes diferentes, dobrando a quantidade de requisições imposta ao servidor.

O comportamento dos pontos (resultados) encontrados é importante para mostrar a velocidade constante de operações mesmo na presença de dois clientes requisitantes, contudo, a análise mais importante advém da comparação entre os dois gráficos, que permite a observação do comportamento do sistema quando todos os seus recursos são exauridos.

No primeiro caso, com apenas um cliente, encontra-se a velocidade de 1520 operações por segundo, contudo, não se pode determinar se os recursos do cliente ou do servidor foram exauridos e não permitiram o aumento no número de operações realizadas por segundo. Contudo, a análise do segundo gráfico mostra outra velocidade, de 1100 operações por segundo, ou seja, o número de operações foi menor.

Essa informação precisa ser analisada levando-se em conta a carga total imposta ao sistema, ou seja, foram 1100 operações por segundo em cada cliente, e, por conseguinte, 2200 operações por segundo considerando-se o sistema como um todo. Pode-se concluir então que no primeiro teste foi atingido o limite de requisições do cliente (já que o número de operações que o servidor pode realizar é maior) e que no segundo foi atingido o limite de requisições que podem ser atingidas pelo servidor (já que o número de operações que cada cliente pode realizar é maior que 1100 operações por segundo, na verdade é de 1520).

O fato dos recursos terem sido exauridos, em momentos diferentes, tanto no cliente quanto no servidor trás o resultado mais importante desta análise: mesmo que o número de requisições realizadas ultrapasse o limite do servidor ou do cliente o sistema continuará se comportando normalmente, ou seja, o tempo de resposta não se altera.

Essa informação é extremamente importante pois permite o levantamento dos limites de um sistema real e, uma vez provada a capacidade de crescimento do DRBAC, a determinação da taxa de crescimento dos recursos para que todas as requisições realizadas ao arcabouço DRBAC sejam atendidas de forma satisfatória.

5.4 Mapeamento entre a Especificação do Sistema e a Especificação de Requisitos

Finda esta apresentação sobre o desenvolvimento do **arcabouço de controle de acesso** para sistemas digitais, o DRBAC, é possível responder aos requisitos

levantados anteriormente. Embora o requisito de **entradas confiáveis (RCA1)** não tenha sido especificamente tratado, o próprio sistema que deve ser protegido é responsável por instanciar o **monitor de referências**, e por conseguinte, repassar ao **arcabouço de controle de acesso** todas as entradas necessárias.

O **requisito RCA2 (Suporte a Diversos Níveis de Especificação)** é atendido por meio da implementação da função de adição de objetos sem qualquer tipo de restrição em relação às operações a serem realizadas sobre esses, e ao modelo de relacionamentos providos pelo RBAC.

Os requisitos **RCA4 (Menor Privilégio)** e **RCA6 (Administração de Políticas)** são garantidos pelo simples atendimento à especificação do modelo RBAC definido pelo NIST, mais especificamente através da utilização de sessões e ativação de papéis (RCA4) e pela impossibilidade de um usuário que não seja administrador prover qualquer tipo de permissão a outro (RCA6).

O atendimento ao requisito **RCA3 (Autorizações Condicionais)** exigiu a modificação do modelo RBAC definido pelo NIST, contudo esta modificação permitiu que o requisito **RCA5 (Separação de Deveres)** também fosse atendido. Embora a exigência de uma segunda autenticação tenha sido a única condição implementada, o desenvolvimento de outras (que atenda a sistemas distribuídos mais específicos) pode ser obtido através da mesma estrutura de dados da permissão, exigindo apenas o complemento do código do **modelo de referência** e a **criação de novas estruturas** se necessário, não impactando as outras funções.

Os requisitos de **sistema distribuído** foram analisados durante a criação e implementação de cada elemento do arcabouço. O **monitor de referência** foi desenvolvido como uma simples biblioteca de processamento de dados, permitindo a instanciação de tantos monitores quanto necessários. Dado a capacidade de crescimento (inúmeras instanciações) e o fato destes módulos serem independentes entre si, é possível garantir tanto à **capacidade de crescimento (RSD1)** quanto à **confiabilidade (RSD3)** e **disponibilidade (RSD2)**.

A **vazão (RSD4)** não é onerada por qualquer tipo de envio de mensagem para inúmeros nós (*broadcast*), ou qualquer tipo de busca que pudesse impor uma enorme troca de mensagens entre os elementos. O requisito de **abertura (RSD5)** é atendido pela construção de uma interface simples e que permite a manipulação do **monitor de referência** por qualquer aplicativo.

As bases de dados utilizadas também são sistemas distribuídos, caracterizados tanto pela **sua capacidade de crescimento (RSD1)** quanto pela **disponibilidade (RSD2)**. Embora a **confiabilidade (RSD3)** destes serviços não seja alta, os efeitos são minimizados através da replicação de servidores. Tanto a distribuição quanto a replicação ajudam a garantir a **vazão (RSD4)** esperada por estes sistemas.

O último requisito, **abertura (RSD5)**, é oferecido por estas bases de dados através de interfaces padronizadas e pelo grande número de bibliotecas que garantem fácil acesso através de diversas ferramentas e linguagens de programação.

5.5 Considerações Finais

Este capítulo apresentou a especificação e desenvolvimento do arcabouço DRBAC e a criação de um ambiente e dois casos de uso que possibilitaram mostrar o funcionamento do sistema. Embora não tenham sido realizados testes exaustivos, os casos de uso mostram que a prova de conceito implementa as principais funções esperadas: manipulação de usuários, objetos, sessões e permissões, além da análise de permissões segundo o modelo de controle de acesso baseado em papéis.

6 CONSIDERAÇÕES FINAIS

A última etapa da apresentação da solução DRBAC é constituído pela comparação entre a solução proposta e aquelas previamente descritas, uma breve conclusão a respeito do arcabouço construído e, por fim, a apresentação de propostas para trabalhos futuros.

6.1 Comparação

Como explicitado na seção 4.4 existem três vertentes sobre a qual a maioria das **soluções de controle de acesso** para **sistemas distribuídos** podem ser enquadradas. Tanto a explanação destas vertentes, como também suas vantagens e desvantagens, foram abordadas nesta seção.

A vertente utilizada no desenvolvimento deste trabalho, **proxies**, diferencia-se das soluções que utilizam **servidores de autorização** por realizar as **decisões de controle de acesso** em cada elemento do sistema distribuído. Embora esta característica aproxime esta vertente das soluções que utilizam **agentes**, a primeira (**proxie**) desenvolve um serviço de autorização que deve ser utilizado por todas as aplicações, enquanto que a segunda (**agentes**) simplesmente tenta garantir que todos os serviços de autorizações apliquem as mesmas regras.

Dentre as soluções apresentadas que também se enquadram nesta vertente, o trabalho exposto se diferencia de [43] principalmente pela **forma de armazenamento** da **base de autorizações**. A solução proposta utiliza uma **tabela de hash distribuída**, enquanto que a segunda solução utiliza um serviço de catálogos.

Esta diferença causa grande impacto na **forma de obtenção da permissão**. Enquanto o dRBAC requisita informações da **base de autorizações** sempre que

uma autorização precisa ser avaliada, a segunda solução provê ao usuário um envelope que pode ser utilizado por um determinado período de tempo. Este espaço de tempo trás o benefício de diminuir o número de requisições de autorização, além de permitir que um usuário continue seguindo a política de segurança mesmo no evento da **base de autorizações** não estar disponível.

Contudo, esta solução pode se tornar inviável para um sistema que possui uma alta taxa de atualização de permissão, requisito atendido pelo sistema dRBAC por utilizar um serviço de tabela de *hash* distribuída, que é limitado apenas pela taxa de atualização da DHT.

A solução apresentada em [42], o protocolo chamado **Maille**, expõe uma solução muito semelhante ao arcabouço DRBAC, ao distribuir a **base de autorizações** em uma rede *peer-to-peer*. Cada elemento da rede armazena um conjunto de partes de permissões, os quais precisam ser resgatados a fim de reconstruir a permissão requisitada.

O **resgate de permissões** é realizado através de uma solicitação que é enviada para todos os elementos da rede, os quais devem responder caso possuam uma parte da permissão requisitada. O nó requisitante espera por um determinado período de tempo antes de analisar todas as partes recebidas e tentar reconstruir a permissão.

Esta solução se diferencia da ofertada por DRBAC exatamente por desconstruir a permissão, o que exige a presença de diversos nós (pelo menos tantos nós quanto partes necessárias para a construção da permissão) para garantir que a permissão seja reconstruída.

O DRBAC distribui as permissões na sua forma completa, exigindo que apenas o elemento da DHT que armazena aquela permissão esteja presente no momento da sua requisição. Além disso, o protocolo Maille também não especifica um modelo de controle de acesso, apenas o protocolo utilizado para o seu transporte, enquanto que a solução DRBAC se beneficia das vantagens oferecidas pelo modelo de controle de acesso baseado em papéis.

6.2 Análise Crítica

O **arcabouço DRBAC** foi construído para responder a dois problemas: encontrar uma **arquitetura e um modelo de controle de acesso** que permitisse a construção de uma ferramenta que aplicasse coercitivamente uma política de controle de acesso adequada a um sistema distribuído.

Estes dois problemas foram formalizados através da criação de dois conjuntos de requisitos e respondidos através da expansão de uma arquitetura mínima para ferramentas de controle de acesso e da escrutinação dos modelos de controle de acesso mais conhecidos.

O modelo RBAC foi escolhido principalmente pela **sua capacidade de lidar com um número ilimitado de usuários**, característica comum em ambientes distribuídos, pela facilidade de gerenciamento das permissões concedidas (não existe propagação de permissões) e pela facilidade de se implantar políticas de segurança de alta granularidade.

O arcabouço oferece todos os benefícios previstos por soluções baseadas em *proxies* e combate a deficiência desta vertente ao desacoplar o **monitor de referências** das bases de informações e implementando-as de forma distribuída.

A utilização de tabelas de *hash* distribuídas foi imprescindível para o sucesso da arquitetura do arcabouço, uma vez que esta tecnologia não apenas efetiva a **distribuição da base de autorizações** como também permite o **crescimento do sistema distribuído** e ainda tira proveito disto, possibilitando que cada nó do sistema possua não apenas o monitor de referência, mas também hospede uma instância da DHT.

A utilização da tríade (papel, objeto, operação) como chaves do serviço DHT permite que as informações sejam distribuídas no maior número possível de nós participantes da DHT, contribuindo para a dispersão da carga e garantindo um maior aproveitamento dos recursos do sistema distribuído.

Se for considerado que objetos recentemente acessados possuem maior probabilidade de serem novamente requisitados, a união desta arquitetura com uma política de priorização das informações armazenadas em cada instância da DHT permitiria que até mesmo um nó que perdesse comunicação com o restante do sistema distribuído ainda pudesse responder adequadamente a diversas requisições.

A construção do **monitor de referência** como apenas uma biblioteca de processamento de informações possibilita que cada aplicação instancie o seu monitor e não precise disputar o acesso a uma única instância. De fato, a aplicação pode até mesmo controlar o número de instâncias do monitor de forma a atender às suas necessidades.

Embora o serviço de autenticação não faça parte da solução construída, a **base de informações** escolhida permite o armazenamento dos dados necessários pelo arcabouço de forma distribuída e ainda oferece uma gama de funcionalidades para o serviço de autenticação, sendo a mais relevante para a criação de árvores de diretórios que refletem a estrutura organizacional da entidade proprietária do sistema.

O arcabouço especificado pôde atender aos dois conjuntos de requisitos previamente mencionados e a realização de testes pode corroborar a correta implementação da especificação desenvolvida, além de permitir a análise do comportamento do arcabouço quando submetido a diferentes cargas.

O resultado mais importante advindo dos testes realizados foi **a determinação do comportamento do arcabouço** quando submetido a cargas que **ultrapassavam o limite de requisições** que cada elemento poderia processar, permitindo anuir que o **arcabouço não apresenta anormalidades frente ao incremento da carga imposta**, e ainda permite a determinação da quantidade de recursos necessário para se atender a uma determinada demanda.

Embora os conjuntos de requisitos possam não atender perfeitamente a um determinado sistema distribuído, o arcabouço pode ser estendido para responder a

um ambiente mais específico. Esta proposta seria facilitada pela arquitetura altamente desacoplada da solução construída.

6.3 Trabalhos futuros

Este trabalho focou em estabelecer um **arcabouço de controle de acesso** básico para sistemas distribuídos, comprovando o funcionamento da arquitetura especificada tanto frente aos requisitos de segurança quanto aos de arquitetura de sistemas distribuídos.

Muitas são as possibilidades de expansão. A frente de segurança pode ser beneficiada pela especificação de uma linguagem e adição de um interpretador ao **monitor de referência**, possibilitando que o **modelo de controle de acesso** possa ser incrementado sem a exigência do desenvolvimento de *software*.

Ainda na frente de segurança, é possível vislumbrar um levantamento das soluções existentes dos serviços de autenticação e de auditoria, possibilitando a especificação, ou criação, de uma solução que atenda a todos os requisitos.

Uma possibilidade de trabalho futuro para a frente de arquitetura distribuída é a criação de políticas que permitam a priorização no armazenamento de dados da DHT, permitindo que as permissões mais utilizadas por cada elemento sejam mantidas em instancias nos próprios elementos. Esta abordagem aumentaria a probabilidade do arcabouço de fato responder às requisições de permissão conforme a política de segurança quando o elemento perdesse a conexão com o restante do sistema distribuído.

REFERÊNCIAS

- [01] Beznosov, K.; Deng, Y. **Engineering Access Control in Distributed Applications**. [S.l.: s.n.], 2007
- [02] **Aristotle's Metaphysics (Stanford Encyclopedia of Philosophy)**. Disponível em: <http://plato.stanford.edu/entries/aristotle-metaphysics/>. Acesso em: 24 ago. 2008.
- [03] Rady, J. A. J. **Segurança em Sistemas Críticos e em Sistemas de Informação - Um Estudo Comparativo**. 2003. 191p. Tese (obtenção do Título de Professor Livre Docente) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2003.
- [04] ESTADOS UNIDOS. National Institute of Standards and Technology. **An Introduction to Computer Security: The NIST Handbook**. Special Publication 800-12. New York, 1995. 290p.
- [05] Bishop, M. **Introduction to Computer Security**. Addison Wesley Professional, 2004. 784p.
- [06] Sandhu, R. S. **On Five Definitions of Data Integrity**. Proceedings of the IFIP WG11.3 Working Conference on Database Security VI, 1993. p. 257-267.
- [07] Akeson, J., 1989. **Assuring system data integrity-an overview**. In Global Telecommunications Conference, 1989, and Exhibition. Communications Technology for the 1990s and Beyond. GLOBECOM '89., IEEE. p. 217-221 vol.1
- [08] Byun, J., Sohn, Y., Bertino, E. **Systematic control and management of data integrity**. Proceedings of the eleventh ACM symposium on Access control models and Technologies, 2006. p. 101-110.
- [09] Foley, S. N. **A nonfunctional approach to system integrity**. In Selected Areas in Communications, IEEE Journal on. Vol. 21, 2003. p. 36-43.
- [10] Stallings, W.; Brown, L. **Computer Security: Principles and Practice**. Prentice Hall, 2007. 880p.

[11] Hammer, J., Schneider, G. **On the Definition and Policies of Confidentiality**. In Information Assurance and Security, 2007. IAS 2007. Third International Symposium on. p. 337-342.

[12] ESTADOS UNIDOS. Common Criteria. **Common Criteria for Information Technology Security Evaluation**, version 3.1, (CCMB-2006-09-001) Set. 2006.

[13] Lopes, M. **Um sistema de Apoio ao Processo de Gerenciamento de Segurança**. 2003. 112p. Dissertação (Mestrado) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2003.

[14] SUIÇA. International Telecommunication Union. **Security Architecture for Open Systems Interconnection for CCITT Applications**. Recommendation X.800. Geneva: 1991.

[15] Shirey, R. **Internet Security Glossary**. Disponível em: <<http://www.ietf.org/rfc/rfc2828.txt>> Acesso em: 05 Jul. 2008

[16] **Kerberos: The Network Authentication Protocol**. Disponível em <<http://web.mit.edu/Kerberos/>> Acesso em: 25 Ago. 2008

[17] SUIÇA. International Telecommunication Union. **Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks**. Recommendation X.509. Geneva: 2005.

[18] Sandhu, R.; Samarati, P. **Access Control: Principles and Practice**. In Communications Magazine, IEEE Journal on. Vol. 32, 1994. p. 40-48.

[19] ESTADOS UNIDOS. Department of Defense. **Department of Defense Trusted Computer System Evaluation Criteria**. DoD 5200.28-STD, Library No. S225,711, 1985.

[20] Clark, D.; Wilson, D. R. **A comparison of Commercial and Military Computer Security Policies**. Proceedings of IEEE Symposium on Security and Privacy, 1987. p. 184.

[21] Ferraiolo, D.; Kuhn, R. **Role-Based Access Control**. Proceedings of 15th National Computer Security, 1992. p. 554-563.

[22] National Institute of Standards and Technology. **ANSI INCITS 359-2004**: for Information Technology – Role Based Access Control. New York, 2004. 56p.

[23] **MDI, Fixed Content Storage.** Disponível em:
<<http://www.mdi.com/SoftwareAndHardware/FixedContentStorage.asp>> Acesso em:
04 Fev. 2009

[24] ESTADOS UNIDOS. EMC Corporation. **Centera Compliance Edition: Understanding and Addressing the Challenges of Compliance and Discovery.** Abr. 2003. 10 p.

[25] Stallings, W. **Cryptography and Network Security.** Prentice Hall, 4Ed, 2005. 592p.

[26] Farley, M. **Building Storage Networks.** Osborne / McGraw-Hill, 1 Ed, 2001. 656p.

[27] Tanenbaum, A. S.; Steen, M. V. **Distributed Systems: Principles and Paradigms.** Prentice Hall, 2 Ed, 2006. 704p.

[28] Lamport, L.; Lynch, N. **Chapter on Distributed Computing.** 1989

[29] Vimercati, S. C. et al. **Access control: principles and solutions.** In Security Software, Special issue. John Wiley & Sons, Inc. New York, NY, 2003. Vol 22, p. 397-421.

[30] Emmerich, W. **Requirements for distributed systems.** London, UK. London University College. Material eletrônico para acompanhamento do curso.

[31] Chun, W. **Core Python Programming.** Prentice Hall. 2nd Edition, 2006. 1136p.

[32] H. Balakrishnan; et al. **Looking Up Data in P2P Systems.** Communications of the ACM, vol. 46, Feb. 2003, pp. 43-48.

[33] W. Yeong; T. Howes; S. Kille **Lightweight Directory Access Protocol**, 1995. Disponível em: <<http://tools.ietf.org/html/rfc1777>>. Acesso em: 03 Abr. 2009.

[34] Woo, T. Y. C.; Lam, S. S. **Designing a distributed authorization service.** Proceedings of INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, 1998. Vol 2, p. 419-429.

- [35] Varadharajan, V.; Crall, C.; Pato, J. **Authorization in Enterprise-wide Distributed System A Practical Design and Application**. In Computer Security Applications Conference, 1998. p. 7-11.
- [36] Khider, H.; Osman, T.; Sherkat, N. **Attribute-Based Authorization for Grid Computing**. In 2010 International Conference on Intelligent Systems, Modelling and Simulation (ISMS), 2010. p. 71-74.
- [37] Pearlman, L.; et al. **A community authorization service for group collaboration**. Proceedings of Third International Workshop on Policies for Distributed Systems and Networks, 2002. p. 50-59.
- [38] Zhi-Hui Wang; et al. **A New Intelligent Authorization Agent Model in Grid**. In Ninth International Conference on Hybrid Intelligent Systems, 2009. p. 394-398,
- [39] Chen, J.; Feng, D.; Liu, J. **An Implementation of Object-based Storage System Access Control Based on IBE**. In Third International Symposium on Intelligent Information Technology and Security Informatics, 2010. p. 635-639.
- [40] **Welcome To The OMG's CORBA Website**. Disponível em <<http://www.corba.org/>>. Acessado em: 02/03/2010.
- [41] **IBM – WebSphere MQ – Software**. Disponível em <<http://www.ibm.com/software/integration/wmq/>>. Acessado em: 02/03/2010.
- [42] Fritz, A.; Paris, J.-F.; **Maille authorization - a distributed, redundant authorization protocol**. In 25th IEEE International Performance, Computing, and Communications Conference, 2006. 8 p.
- [43] Feichtinger, D.; Peters, A.J. **Authorization of data access in distributed storage systems**. In 6th IEEE/ACM International Workshop on Grid Computing, 2005. 7 p.