FERNANDO BERGAMOS MARIANI

AVALIAÇÃO DOS MODELOS DE CONTROLE DE ACESSO ABAC E RBAC EM AMBIENTES DE COMPUTAÇÃO EM NUVEM

São Paulo

2013

FERNANDO BERGAMOS MARIANI

AVALIAÇÃO DOS MODELOS DE CONTROLE DE ACESSO ABAC E RBAC EM AMBIENTES DE COMPUTAÇÃO EM NUVEM

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para conclusão do curso MBA em Tecnologia da Informação

Área de Concentração:
Segurança da Informação

Orientador:
Prof. Dr. Stephan Kovach

São Paulo

2013

FICHA CATALOGRÁFICA

Mariani, Fernando Bergamos
Avaliação dos modelos de controle de acesso ABAC e RBAC em ambientes de computação em nuvem / F.B. Mariani. -- São Paulo, 2013.
52 p.

Monografia (MBA em Tecnologia da Informação) - Escola Politécnica da Universidade de São Paulo. Programa de Educação Continuada em Engenharia.

1.Computação em nuvem 2.Controles de acesso 3.Tecnologia da informação I.Universidade de São Paulo. Escola Politécnica. Programa de Educação Continuada em Engenharia II.t.

AGRADECIMENTOS

A todos que me ajudaram a concluir este trabalho, sou muito grato pelo apoio e compreensão.

Aos meus pais e familiares pelo apoio e incentivo na conclusão desse trabalho.

Aos professores do PECE e colegas de sala pelo conhecimento e experiência passados durante as aulas.

Ao meu orientador, professor Stephan Kovach pela oportunidade, confiança e paciência durante o desenvolvimento desse trabalho.

RESUMO

Computação em Nuvem *(Cloud Computing)* é um paradigma em crescimento nos últimos anos, onde os recursos computacionais são fornecidos como serviços através da rede. Esse paradigma traz novos desafios em diversas áreas, sendo uma delas a de controle de acesso. Esse trabalho pretende analisar os modelos de controle de acesso baseado em papéis (RBAC) e controle de acesso baseado em atributos (ABAC) aplicados em *Cloud Computing*, para determinar qual desses modelos melhor se aplica para este tipo de ambiente. Serão usados os critérios para avaliação de sistemas de controle de acesso definidos pelo *National Institute of Standards and Technology* (NIST) no informe NIST 7316 (2006).

ABSTRACT

Cloud Computing is a new paradigm that is emerging in the last years, where the computational resources are provided as services through the Internet. This new paradigm creates challenges in various areas, including security and access control.

This work aims to analyze two access control models applied to Cloud Computing to determine which one is more appropriate to this kind of environment taking into account their characteristics. Role Based Access Control (RBAC) and Attribute Based Access Control (ABAC) are the models analyzed.

The metrics defined by National Institute of Standards and Technology (NIST) in report NIST 7316, Assessment of Access Control Systems (2006) are used to evaluate these models.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABAC	Attribute Based Access Control
API	Application Programming Interface
CSA	Cloud Security Alliance
DAC	Discretionary Access Control
IAAS	Infrastructure as a Service
IP	Internet Protocol
MAC	Mandatory Access Control
NIST	National Institute of Standards and Technology
OSSIM	Open Group Services Integration Maturity Model
PAAS	Platform as a Service
PEP	Policy Enforcement Point
RBAC	Role Based Access Control
ROI	Return on Investment
SAAS	Software as a Service
SOA	Software Oriented Architecture
SSD	Static Separation of Duties
TI	Tecnologia da Informação
XACML	eXtensible Access Control Markup Language

SUMÁRIO

1 INTRODUÇÃO

1.1 CONSIDERAÇÕES INICIAIS

Uma das características mais importantes em sistemas corporativos atualmente é a necessidade de colaboração e integração entre diferentes usuários e sistemas para realizar determinadas tarefas (GUTIERREZ VELA *et al.*, 2006). Esses sistemas também devem ser capazes de evoluir e se adaptar rapidamente às constantes mudanças nas estratégias de negócio das empresas, que também afetam os processos de negócio.

A arquitetura orientada a serviços (SOA) é um paradigma de arquitetura de software que utiliza serviços como elemento básico para o desenvolvimento de aplicações (PAPAZOGLOU, 2003). O serviço é a unidade fundamental que provê uma funcionalidade, independente de plataforma, e com uma interface bem definida. Com isso permite a criação de aplicações distribuídas de forma rápida, flexível e com baixo acoplamento.

A computação em nuvem (*Cloud Computing*) permite uma realização da arquitetura SOA onde as funções de negócio são oferecidas como serviços, que podem ser terceirizados em provedores externos (*Public Cloud*), armazenados em uma estrutura própria da empresa (*Private Cloud*) ou em uma mistura de publica e privada (híbrida). Em todos os casos o serviço é disponibilizado para acesso através da rede (seja internet ou intranet) (MOTAHARI-NEZHAD *et al.*, 2009).

Esse novo paradigma gera novas oportunidades de negócio, mas a questão da segurança, ainda é um problema, em particular o controle de acesso, onde os modelos existentes são pouco flexíveis, não levando em conta a composição de serviços entre domínios.

1.2 OBJETIVO

O objetivo desse trabalho é avaliar os modelos de controle de acesso existentes, e analisar sua aplicação em ambientes de *Cloud Computing*, considerando as dificuldades que surgem nesses ambientes devido à troca de dados entre diferentes domínios. Serão analisados modelos conhecidos, como o *Role Based Access Control* (RBAC) e *Attribute Based Access Control* (ABAC) utilizando os critérios definidos pelo *National Institute of Standards and Technology* (NIST) para avaliar sistemas de controle de acesso. Baseado na analise, será indicado qual modelo melhor se adapta a um ambiente de *Cloud Computing*.

1.3 MOTIVAÇÃO

Cloud Computing é um paradigma em crescimento nos últimos anos, onde os recursos computacionais são fornecidos como serviços através da internet, utilizando entre outras, técnicas como SOA, e virtualização. Esse paradigma traz novos desafios em diversas áreas, sendo uma delas o controle de acesso. De acordo com uma pesquisa realizada pela Intel em Maio de 2012 com 800 profissionais de TI sobre os empecilhos para a adoção de *Cloud Computing*, a maior preocupação com relação à segurança é o controle de acesso (INTEL IT CENTER, 2012). Nessa pesquisa, 63% dos entrevistados mencionaram o controle de acesso como uma das principais preocupações, sendo que desses, 24% a escolheram como a maior preocupação dentre as outras listadas.

Essa pesquisa mostra que a área de controle de acesso em ambientes de *Cloud Computing* ainda tem muito a evoluir, e que as empresas ainda não confiam nos mecanismos existentes atualmente, o que motivou esse trabalho.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está organizado em quatro capítulos, além das referências bibliográficas.

No Capítulo 2 são apresentados os conceitos sobre ambientes distribuídos, arquitetura orientada a serviços, C*loud Computing* e controle de acesso.

O Capítulo 3 aborda os modelos de controle de acesso ABAC e RBAC, a análise desses modelos, e a recomendação de qual melhor se aplica a um ambiente de *Cloud Computing*.

No Capítulo 4 são apresentadas as conclusões desse trabalho, bem como assuntos relevantes para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 SISTEMAS DISTRIBUÍDOS

Um sistema distribuído consiste de uma coleção de diversos computadores autônomos, conectados através de uma rede, que permita a coordenação e distribuição de tarefas para a realização de uma atividade e compartilhamento de recursos do sistema, de forma que o usuário o veja como um sistema integrado. Um sistema distribuído pode ter um único objetivo comum, como por exemplo, resolver um grande problema computacional (ANDREWS, 1999).
As principais características de um sistema distribuído são:

- Compartilhamento de recursos: é a habilidade de usar um hardware, software ou dado de qualquer lugar do sistema. O recurso está fisicamente encapsulado em um dos computadores, mas pode ser acessado por qualquer outro através da comunicação.
- Transparência: é a preocupação com extensões e melhorias do sistema distribuído. Novos componentes devem ser integrados com componentes já existentes de forma que fique disponível para o sistema como um todo.
- Concorrência: diversas tarefas sendo realizadas ao mesmo tempo para realizar uma atividade. Essas tarefas podem acessar recursos concorrentemente. Deve haver uma coordenação para garantir a integridade do sistema e dos dados.
- Escalabilidade: Capacidade de aumentar a escala do sistema para acomodar mais usuários no sistema, sem necessitar alterações nos componentes existentes.
- Tolerância a falhas: capacidade de o sistema continuar operando em caso de falhas, sem degradar o seu desempenho.
- Transparência: a complexidade do sistema é escondida do usuário e dos programadores, de forma que o sistema seja visto como um só ao invés de vários componentes cooperando entre si.

2.2 SERVICE ORIENTED ARCHITECTURE (SOA)

De acordo com Slama *et al.* (2004), a definição de Arquitetura Orientada a Serviços é um tipo de arquitetura de software baseada em alguns conceitos chaves: *frontend* de aplicação, serviço, repositório de serviços, e barramento de serviços.

A Figura 1 mostra como esses conceitos formam a arquitetura SOA.

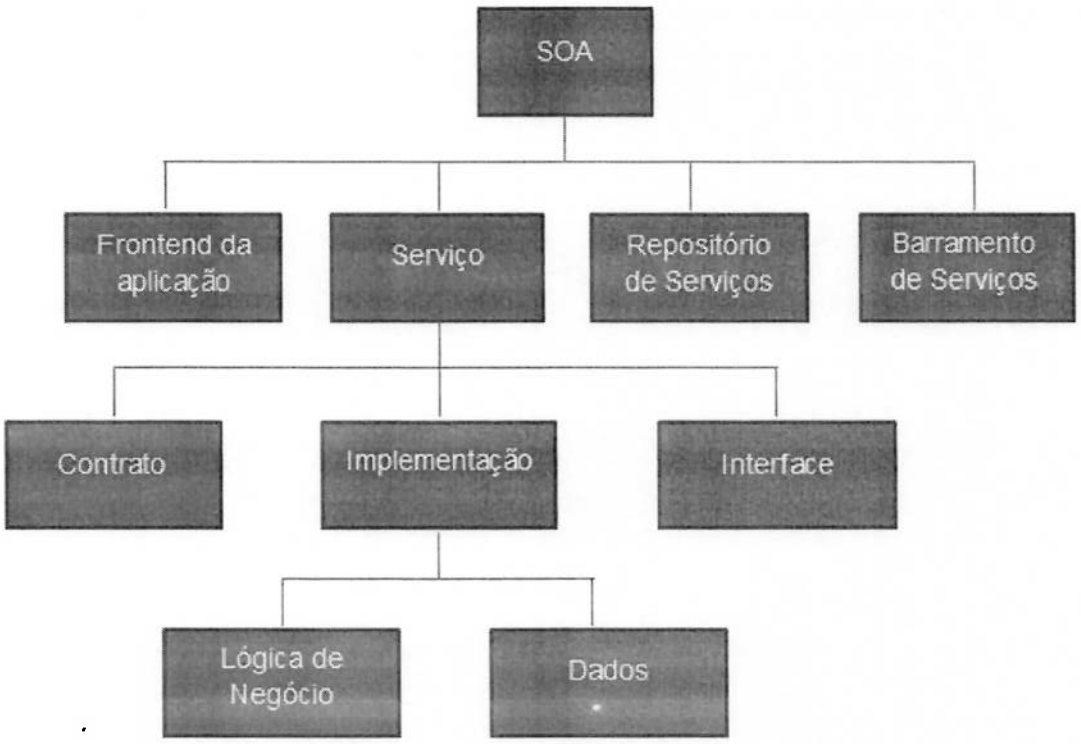

Figura 1 - Desmembramento da arquitetura SOA. [Adaptado de Slama et al. (2004)]

- O *frontend* da aplicação é a parte inicial do processo de negócio. Ele inicia e controla a atividade do sistema corporativo. Um exemplo é uma interface gráfica, como um navegador *web*. Mas o *frontend* não precisa necessariamente interagir com o usuário final. Pode ser, por exemplo, uma rotina batch que periodicamente executa uma funcionalidade.

- O serviço fornece a funcionalidade de negócio que o *frontend* da aplicação e outros serviços podem usar, e consiste de: uma implementação que fornece a lógica de negócio e os dados, um contrato e uma interface de serviço.
    - O contrato de serviço especifica o propósito, funcionalidade, restrições e usos do serviço.
    - A implementação do serviço prove a lógica de negócio e os dados. É a realização do contrato de negócio
    - A interface expõe a funcionalidade do serviço aos clientes conectados na rede.
- O repositório de serviços armazena os contratos de serviço individuais e proporciona facilidades para descobrir os serviços e obter informações de como usá-los.
- O barramento de serviços conecta os *frontends* de aplicação com os serviços. O seu propósito é promover a interconexão entre os participantes garantindo a heterogeneidade de tecnologia.

Segundo Erl (2009), existem sete motivos que levam as empresas a adotarem uma arquitetura SOA.

- Maior interoperabilidade intrínseca: Um dos objetivos é estabelecer a interoperabilidade entre serviços para reduzir a necessidade de integração
- Maior federação: um ambiente onde os recursos e aplicações estão unidos, mas mantendo sua própria autonomia e governança.
- Mais opções de diversificação de fornecedores: Possibilita que a empresa tenha a liberdade de alterar, estender ou substituir soluções implementadas.
- Maior alinhamento do negócio e do domínio tecnológico: Encapsulamento das regras de negócio nos serviços.
- Maior Retorno sobre investimento (ROI): Os serviços devem ser criados de forma que possam ser reutilizados em diferentes composições, podendo se adaptar inúmeras vezes em diferentes composições, reduzindo despesas e esforço.

- Maior agilidade organizacional: Com um inventário de serviços altamente padronizados e reutilizáveis, as mudanças se tornam mais rápidas.
- Menor carga de Trabalho de TI: Redução de tempo com trabalhos redundantes, e aumento de eficiência através do reuso.

2.3 COMPUTAÇÃO EM NUVEM (*CLOUD COMPUTING*)

Cloud Computing surgiu como uma evolução natural e integração de diversos campos de pesquisa, incluindo computação distribuída, computação em *grid*, *webservices* e arquitetura orientada a serviços (WEISS, 2007).

De acordo com Claunch e Cearley (2008), *Cloud Computing* é um estilo de computação onde os recursos de TI são enormemente escaláveis e os recursos são fornecidos aos clientes como serviços através da Internet. É uma solução onde todos os recursos de TI (*hardware*, *software*, redes, armazenamento dentre outros) são fornecidos aos usuários rapidamente e sob demanda. Esses recursos ou serviços são gerenciados de forma a garantir alta disponibilidade, segurança e qualidade de serviço.

O *National Institute of Standards and Technology* (NIST) propõe um modelo de referência para a computação em nuvem que considera a existência de três dimensões (NIST, 2009).

a) Características especiais

- Acesso Amplo à rede: Disponibilidade pela rede e acesso através de mecanismos padronizados que possibilitam o uso a partir de diversos tipos de plataforma (ex: celulares, *tablets*, *laptops*, *desktops*).
- Rápida elasticidade: permite aumentar e reduzir dinamicamente as quantidades contratadas de forma automática. Para o consumidor a capacidade disponível parece ilimitada e pode ser comprada a qualquer momento.

- Serviços mensuráveis: sistemas de *Cloud* automaticamente controlam e aperfeiçoam recursos aproveitando a capacidade de medição. O uso dos recursos pode ser monitorado de forma transparente para o consumidor do serviço.
- Auto-serviço sob demanda: Um consumidor pode, unilateralmente, provisionar capacidade computacional à medida da necessidade sem necessitar interação com o provedor de serviços.
- Agrupamento de Recursos: Os recursos computacionais do provedor de serviços são divididos e agrupados para servir múltiplos clientes simultaneamente, com diferentes recursos físicos e virtuais alocados e realocados dinamicamente de acordo com a demanda do consumidor. O consumidor não consegue determinar exatamente a localização dos recursos providos, porém é capaz de determinar a localização em um nível maior de abstração, como datacenter, estado ou país.

b) Ofertas de serviços

- *Software as a Service* (SaaS): Os serviços disponibilizados ao consumidor são aplicativos executados em uma infraestrutura de *Cloud*. Esses aplicativos podem ser acessados por diversos tipos de dispositivos, como um navegador *web*. O consumidor não gerencia ou controla a infraestrutura da nuvem, incluindo rede, servidores, sistemas operacionais, ou mesmo configurações individuais da aplicação, com exceção das configurações de usuário.
- *Platform as a Service* (PaaS): Os serviços disponibilizados ao consumidor possibilitam executar aplicativos desenvolvidos pelo próprio consumidor ou adquiridos, utilizando linguagem de programação, bibliotecas, serviços e ferramentas disponibilizados pelo provedor. O consumidor não gerencia ou controla a infraestrutura da nuvem.
- *Infrastructure as a Service* (IaaS): Os serviços disponibilizados ao consumidor possibilitam o uso de recursos da infraestrutura: processamento, armazenamento, rede e outros recursos

computacionais, onde o consumidor é capaz de executar aplicativos e sistemas operacionais. O consumidor não gerencia essa infraestrutura, mas tem controle sobre o sistema operacional, armazenamento, aplicativos, e em alguns casos podem até controlar alguns limitados componentes de segurança, como por exemplo, um *firewall.*

c) Modelos de Implementação

- Pública: A infraestrutura da nuvem é de propriedade de uma organização que comercializa os recursos e os disponibiliza para os consumidores em geral.
- Privada: A infraestrutura da nuvem é utilizada por uma única organização. Pode ser gerida por ela ou por terceiro, e pode ou não estar localizada dentro da própria organização.
- Comunitária: A infraestrutura da nuvem é compartilhada por diversas organizações que compõe uma comunidade específica, com preocupações em comum. Pode ser gerida por membros dessa comunidade ou por terceiros, e pode estar instalada em local próprio dessa comunidade.
- Híbrida: A infraestrutura da nuvem é uma combinação de dois dos três modelos acima. Continuam sendo entidades distintas, mas que estão conectadas por meio de tecnologia proprietária ou aberta, que viabiliza a portabilidade de dados e aplicativos.

A *Cloud Security Alliance* (CSA, 2009) complementa o modelo com uma representação gráfica, conforme Figura 2.

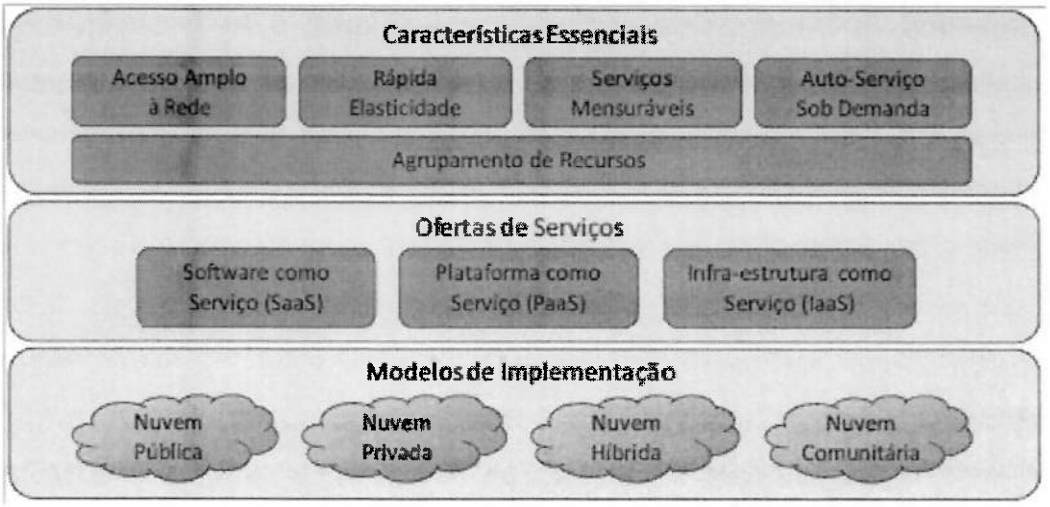

Figura 2 - Modelo de referência para computação em nuvem. [Adaptado de Cloud Security Alliance, (2009)].

A computação em nuvem traz alguns benefícios aos seus consumidores. De acordo com Bandyopadhyay (2009), alguns desses benefícios são:

- Redução de Custo: o custo de uso para pequenas empresas é reduzido, tornando possível a elas utilizar recursos computacionais que antes eram acessíveis apenas a empresas de grande porte.
- Redução de Investimento Inicial: Não é necessário fazer um investimento inicial para aquisição de ativos. Os gastos se dão sob a forma de custo operacional, diminuindo os valores investidos antecipadamente.
- Redução de Barreiras à Inovação: A computação em nuvem tem potencial de reduzir as barreiras à inovação. Existem diversos exemplos de empresas que utilizaram de recursos computacionais em nuvem e que em pouco tempo se tornaram bem sucedidas, como por exemplo, Facebook, Twitter e outras.
- Facilidade para escalar: Como os recursos computacionais são gerenciados por *software*, é possível aumentar a capacidade rapidamente à medida que novos equipamentos são adicionados à rede.

Apesar dos benefícios citados, ainda existem barreiras que impedem a adoção da computação em nuvem. Marks e Lozano (2010) citam algumas dessas barreiras:

- Segurança e privacidade: são quesitos obrigatórios nos ambientes de computação em nuvem. A perspectiva de encontrar falhas de segurança e violação de privacidade dos usuários ainda faz os consumidores questionarem a adoção desses ambientes.
- Nível de Qualidade de Serviço: os consumidores tem medo que devido à complexidade e pouco tempo de maturação, os provedores de serviço não estejam prontos para garantir a qualidade do serviço.
- Confiabilidade: A computação em nuvem depende da confiabilidade, e os consumidores ainda não tem plena confiança e por isso relutam a aderir a esse modelo de serviços.

2.4 CONTROLE DE ACESSO EM AMBIENTES DISTRIBUIIDOS

Controle de acesso é uma política ou procedimento que autoriza, nega ou restringe o acesso a um sistema. Pode também monitorar e registrar todas as tentativas de acesso a um sistema, bem como as tentativas de acesso não autorizados. Existem diversos mecanismos de controle de acesso, e que podem ser enquadrados em uma dos tipos abaixo:

- *Discretionary Access Control* (DAC): É uma forma de restringir o acesso a objetos baseado na identidade do usuário. Um sujeito com determinada permissão de acesso é capaz de passar essa permissão para qualquer outro sujeito.
- *Mandatory Access Control* (MAC): É uma forma de restringir o acesso a objetos baseado em um conjunto fixo de atributos de segurança, ou rótulos designados a usuários e objetos. Esse controle é garantido pelo sistema, e não pode ser alterada pelo usuário ou seus programas.

Esses tipos de controle de acesso não são mutualmente exclusivos. Muitos sistemas utilizam MAC e DAC em conjunto. (YUAN E TONG, 2005)

Esses modelos são conhecidos como controle de acesso baseado em identidade, onde um usuário (sujeito) e recurso (objeto) são identificados por nomes únicos. Essa identificação pode ser feita através de papeis designados aos sujeitos. Nesse caso o modelo é chamado de controle de acesso baseado em papeis (*roles*) (RBAC).

No contexto de ambientes distribuídos, como os ambientes em *Cloud Computing*, os sistemas são como organizações virtuais, com diversos domínios autônomos. O relacionamento entre usuários e recursos é dinâmico, e os usuários e recursos não fazem parte do mesmo domínio de segurança. Nesse caso, os usuários normalmente são identificados por seus atributos e características, e não por uma identidade pré-definida. Por isso os modelos de controle de acesso baseado em identidades não são adequados para essa situação, e foram criados modelos de controle de acesso que levam em consideração os atributos do usuário para tomar a decisão. Um desses modelos é o Modelo de Controle de Acesso Baseado em Atributos (ABAC) (KHAN, 2012).

3 MODELOS DE CONTROLE DE ACESSO APLICADOS A COMPUTAÇÃO EM NUVEM

3.1 CONTROLE DE ACESSO BASEADO EM *ROLES* (RBAC)

O controle de acesso baseado em *roles* introduz o conceito de *papéis* para determinadas funções. As permissões para realizar algumas operações são atribuídas aos papéis. O conceito básico é que as permissões são atribuídas a papeis, e não a usuários. Dessa forma um usuário possui alguns papéis, e pode executar tarefas que estejam associadas a esse papel.
Por exemplo, em um sistema bancário, um usuário com perfil de "Gerente de Agência" tem permissão para executar as operações "Consultar Saldo", "Bloquear Conta" dentre outras. Caso esse usuário mude de função basta remover esse perfil de seu usuário, e adicioná-lo ao usuário do novo responsável por essa função.

O modelo RBAC é formado por um conjunto de seis elementos básicos, de acordo com Sandhu *et al.* (1996):

- Usuário (U): A pessoa ou agente que está tentando acessar o sistema
- Papel (R): Função que define um nível de autorização
- Objeto (Obj): O recurso que está sendo acessado
- Operação (Oper): Operação que está sendo chamada
- Permissão (P): Aprovação de acesso a um recurso
- Sessão (S): Um mapeamento entre o usuário, papel e recurso

O modelo RBAC pode ser usado para implementar três princípios importantes de segurança: menor privilégio, separação de responsabilidades e abstração de dados (SANDHU *et al.*, 1996).

O princípio de menor privilégio significa que o papel só tem acesso às mínimas funções necessárias em um espaço de tempo determinado.

O princípio da separação de responsabilidades indica que as permissões de acesso são separadas entre os papeis de forma a facilitar o gerenciamento de diferentes níveis de segurança.

Por último, o princípio da abstração de dados significa que permissões mais abstratas podem ser usadas ao invés das tradicionais (leitura, escrita, execução), como por exemplo, para um objeto do tipo conta, as permissões abstratas credito e débito podem ser definidas.

A Figura 3 ilustra o modelo RBAC e o relacionamento entre seus elementos básicos.

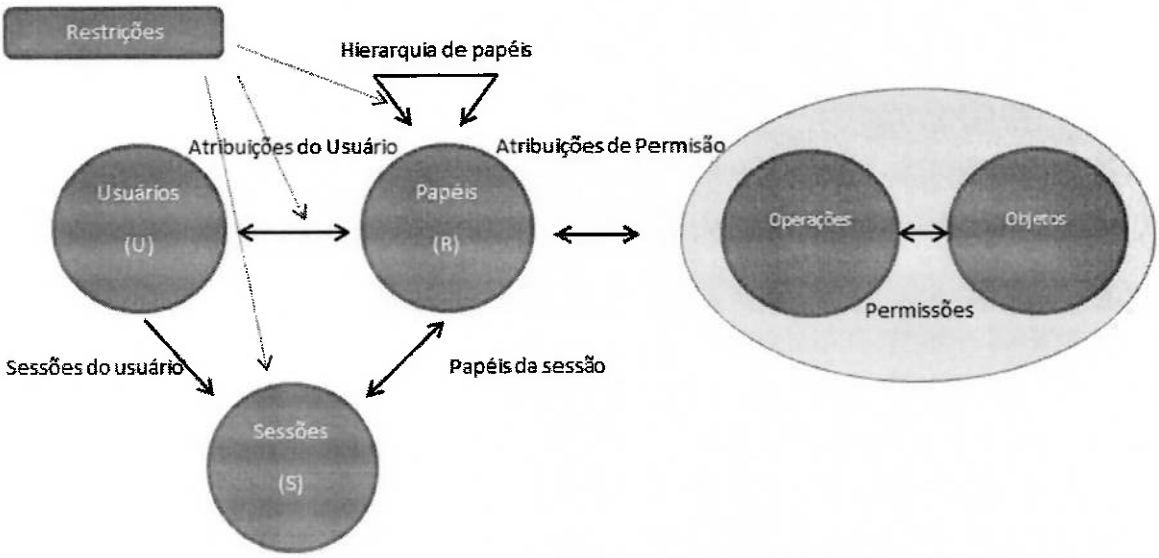

Figura 3 - Modelo de Controle de Acesso RBAC. [Adaptado de Sandu et al. (2000)].

O relacionamento entre Usuários e Papéis (chamado de Atribuições do usuário) deve ser muitos para muitos (seta bidirecional), já que um usuário pode ser membro de diversos papéis, e cada papel pode ter mais de um usuário.

O relacionamento entre Papéis e Permissões também deve ser muitos para muitos, já que um papel pode ter várias permissões, e cada permissão pode fazer parte de diversos papéis.

Cada sessão corresponde ao mapeamento de um usuário, com um ou mais papéis. Quando o usuário estabelece a sessão, ele ativa um subconjunto de papéis dos quais é membro. As permissões disponíveis para um usuário é a união de todas as permissões de todos os papéis ativos naquela sessão.

As restrições podem estar associadas com a atribuição de usuário e papel, com a ativação de papéis dentro da sessão do usuário, ou com a hierarquia de papéis. Essas restrições são usadas para prevenir o conflito de interesses que podem ocorrer de usuários que excedam o nível de autorização para suas posições. Existem dois tipos de restrição que podem ser aplicados, a separação estática de obrigação – *Static Separation of Duties* (SSD) e a separação dinâmica de obrigações – *Dynamic Separation of Duties* (DSD). Na separação estática, a restrição é aplicada no momento em que os usuários são atribuídos a um papel. Com isso pode se aplicar uma restrição que proíbe que um usuário seja membro de um papel caso ele já seja membro de outro papel conflitante. Os papéis passariam a ser mutualmente exclusivos. As restrições também são herdadas em uma hierarquia de papéis. Na separação dinâmica de papéis, as restrições são aplicadas dentro da sessão do usuário. Esse caso é usado quando o usuário pode ser membro de um conjunto de papéis quando atuarem independentemente, mas que causa um conflito de interesse quando utilizados simultaneamente.

Em *Cloud Computing*, os provedores de serviço expõem um conjunto de interfaces de software (*Application Programming Interface* - APIs) para os clientes gerenciarem e monitorarem seus serviços. Essas interfaces devem possuir formas de autenticação, controle de acesso e encriptação para proteger do acesso indevido. Sirisha, A. e Kumari, G. (2010) propuseram um controle de acesso baseado em *roles* para APIs de sistemas em *Cloud Computing*. Nesse modelo, a política de controle de acesso é implementada no nível da API dos serviços da *Cloud* e é feita em duas etapas. A Figura 4 mostra as duas etapas desse modelo. Antes de acessar qualquer recurso, o usuário deve ser

autenticado por algum mecanismo (não tratado nesse texto, já que o objetivo é o controle de acesso).

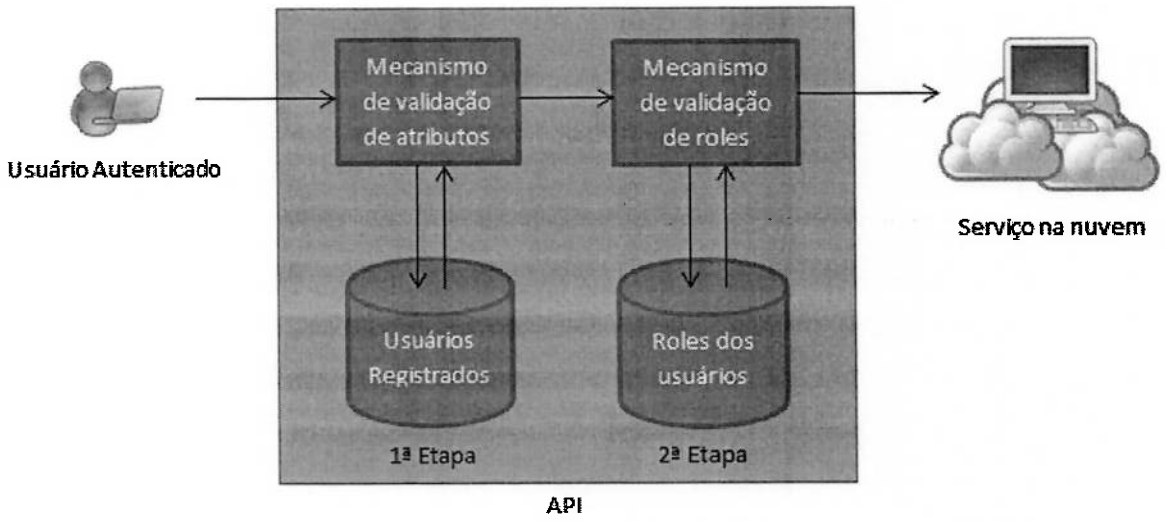

Figura 4 - RBAC no nível de API em Cloud Computing. [Adaptado de Sirisha, A. e Kurami, G. (2010)]

Quando o usuário é autenticado, junto com suas credenciais alguns atributos são informados, como por exemplo, o endereço IP de onde está acessando e o nome do domínio de rede. Esses atributos são usados para identificar a organização da qual o usuário faz parte, com ajuda de uma base de dados de usuários registrados. Essa é a primeira etapa do modelo.

No momento que uma organização é cadastrada nessa base de dados, o provedor de serviços determina os papeis (e permissões associadas) e os usuários que estão associados a cada papel de acordo com a política da organização. Qualquer mudança deve ser comunicada e atualizada imediatamente nessa base de dados para evitar falhas de segurança.

Assim que o usuário é autenticado e sua organização identificada, os papéis do usuário são carregados com ajuda da base de dados de usuários. Essa é a segunda etapa do modelo. A partir deste momento o usuário tem o acesso permitido aos recursos e operações conforme definido em seus papéis.

Em resumo, a primeira etapa garante que apenas usuários previamente registrados de domínios conhecidos acessem os serviços da *Cloud*, e ao mesmo tempo, extrai os atributos necessários para a segunda etapa, que irá obter os papeis definidos para aquela organização e usuário, e com isso as permissões à recursos e operações que aquele usuário pode acessar.

3.2 CONTROLE DE ACESSO BASEADO EM ATRIBUTOS (ABAC)

No controle de acesso baseado em atributos, a decisão de acesso é fundamentada em características relevantes de segurança, chamadas de atributos. No controle de acesso, três tipos de atributos são importantes (YUAN e TONG, 2005):

- Atributos do sujeito. O sujeito (*s*) é uma entidade (usuário, aplicação, processo) que executa uma ação em um recurso. Cada sujeito tem atributos associados a ele, que definem a sua identidade e as suas características. Esses atributos podem incluir o identificador do sujeito, nome, organização, cargo, e outros. O papel do sujeito também pode ser um de seus atributos.
- Atributos do recurso. O recurso (*r*) é a entidade (serviço, estrutura de dados, sistema) que está sendo chamada pelo sujeito. Assim como os sujeitos, os recursos também possuem atributos que podem ser levados em conta na tomada de decisão. Por exemplo, um arquivo de texto em um computador possui atributos como autor, data, título, e outros. Os atributos do recurso normalmente podem ser extraídos dos meta-dados do recurso.
- Atributos do ambiente. Os atributos do ambiente (*e*) descrevem a operação, detalhes técnicos e situações do ambiente ou contexto onde ocorre o acesso à informação. Por exemplo, data e hora corrente, nível de segurança da rede (internet ou intranet). Esses atributos não estão

associados a um sujeito ou recurso, mas são relevantes na decisão de controle de acesso.

Exemplo de uma regra de controle de acesso utilizando o ABAC, onde 's' é o sujeito, 'r' o recurso a ser acessado e 'e' os atributos de ambiente:

R1: pode_acessar(s, r, e) ← (Role(s) = 'Gerente') ^ (Nome(r) = 'Aprovar Compra') ^ (Hora(e) = 'Horário comercial')

Essa regra impõe que para acessar o recurso 'Aprovar compra', o sujeito deve possuir o *role* 'Gerente', e só pode ser acessado no horário comercial.

A Figura 5 mostra como o modelo ABAC pode ser aplicado em um ambiente de *Cloud Computing*.

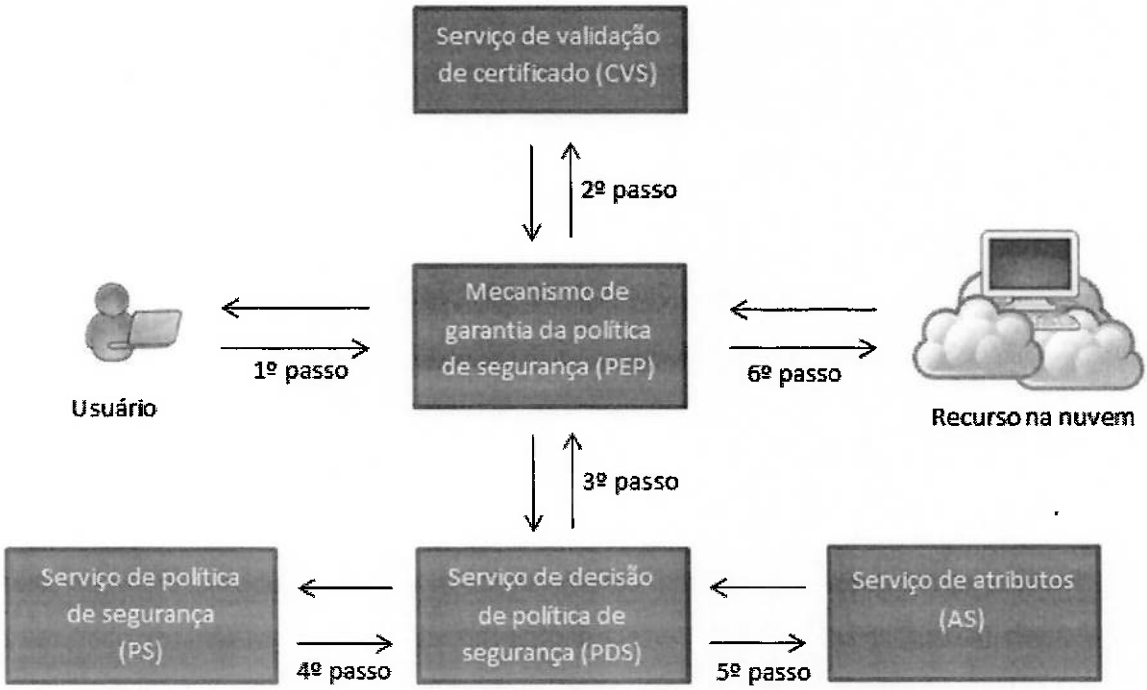

Figura 5 - Modelo ABAC aplicado em ambiente de Cloud Computing. [Adaptado de GOSAC-N (2011)]

Um usuário solicita acesso a um recurso da nuvem. O mecanismo de garantia da política de segurança (*Policy Enforcement Point* - PEP) intercepta todas as requisições aos recursos da nuvem para autorização. O PEP deve ser específico

para cada tipo de recurso. Por exemplo, para acessar um recurso do *Microsoft Share Point*, deve ser criado um PEP específico para ele.

Caso esteja sendo usado um certificado, o PEP valida as credenciais através do serviço de validação de certificado (*Certificate Validation Service* – CVS) e em seguida é feita uma requisição de autorização ao serviço de decisão de política de segurança (*Policy Decision Service* – PDS). De acordo com o recurso solicitado, a política de segurança correspondente será carregada do serviço de política de segurança (*Policy Service* – PS). O PDS também faz uma consulta ao serviço de atributos (AS) para determinar os atributos que serão necessários na tomada de decisão.

As políticas de segurança são armazenadas no formato XACML (*Extensible Access Control Markup Language*). O XACML é um padrão para declarar políticas de segurança em um arquivo no formato XML e foi criado com o objetivo de promover a interoperabilidade entre implementações de autorização de diversos fornecedores.

O arquivo XACML é estruturado em três elementos básicos (OASIS, 2010):

- <Rule> (Regra): Contém uma expressão booleana que pode ser avaliada isoladamente, mas que não é projetada para ser acessada isoladamente pelo PDS, ou seja, não deve ser usada para tomar a decisão de autorização isoladamente, mas sim para ser reutilizada por múltiplas políticas.

- <Policy> (Política): Contém um grupo de elementos de regras e um procedimento para combinar o resultado das avaliações. É a unidade básica de politica usada pelo PDS para tomar a decisão de autorização.

- <PolicySet> (Grupo de Políticas): Contem um conjunto de elementos do tipo <Policy> ou outros <PolicySet> e um procedimento para combinar o resultado das avaliações. É a forma de combinar diversas políticas de segurança separadas em uma única politica.

A Figura 6 mostra um exemplo de uma política de segurança escrita no formato XACML. Foram removidas diversas tags de formatação para facilitar o entendimento.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Policy>
3      <Target/>
4      <VariableDefinition VariableId="17590034">
5      <Apply FunctionId="string-equal">
6          <Apply FunctionId="string-one-and-only">
7              <AttributeDesignator Category="subject" AttributeId="numero-paciente" DataType="string"/>
8          </Apply>
9          <Apply FunctionId="string-one-and-only">
10             <AttributeSelector Category="resource" Path="md:registro/md:paciente/md:numero-paciente/text()"/>
11         </Apply>
12     </Apply>
13     </VariableDefinition>
14     <Rule RuleId="urn:oasis:names:tc:xacml:3.0:exemplo:regra:1" Effect="Permit">
15         <Description>Uma pessoa pode ler os registros médicos nos quais ele/ela é o paciente</Description>
16         <Target>
17             <AnyOf>
18                 <AllOf>
19                     <Match>
20                         <AttributeValue>urn:exemplo:medico:schemas:registro</AttributeValue>
21                         <AttributeDesignator Category="resource"/>
22                     </Match>
23                 </AllOf>
24             </AnyOf>
25             <AnyOf>
26                 <AllOf>
27                     <Match MatchId="string-equal">
28                         <AttributeValue>leitura</AttributeValue>
29                         <AttributeDesignator Category="action"/>
30                     </Match>
31                 </AllOf>
32             </AnyOf>
33         </Target>
34         <Condition><VariableReference VariableId="17590034"/></Condition>
35     </Rule>
36 </Policy>
```

Figura 6 - Exemplo simplificado de uma política de segurança utilizando o formato XACML versão 3

No exemplo acima, na linha cinco é definida a função que será utilizada pela regra. Essa função recebe como parâmetro dois atributos do tipo *String* e compara se os valores são iguais. Na linha sete é passado o primeiro parâmetro, que é o número do paciente, obtido do sujeito que está fazendo a requisição. Na linha dez é passado o segundo parâmetro, que é o número do paciente obtido do registro que está sendo acessado.

Na linha catorze é definida a regra, que uma pessoa só pode ler o registro médico caso ela seja o paciente daquele registro. Só será permitido o acesso se as condições forem atendidas. A linha dezenove indica que todos os recursos do tipo registro médico devem ser avaliados e a linha vinte e sete indica que quando a ação sendo executada for a de leitura, a regra deve ser avaliada. Na linha trinta e quatro é definida a condição de avaliação da regra, que é o resultado do retorno da função definida anteriormente na linha cinco.

3.3 CRITÉRIOS DE QUALIDADE PARA AVALIAR OS MODELOS DE CONTROLE DE ACESSO

O Instituto Nacional de Padrões e Tecnologia (NIST) propôs um conjunto de métricas de qualidade para avaliar mecanismos de controle de acesso. (NIST, 2006). Esses critérios serão utilizados para avaliar os modelos apresentados na seção anterior.

Os critérios definidos pelo NIST para avaliar os mecanismos de controle de acesso são:

a) Atribuição e remoção de permissões de usuários do sistema: Quais são os passos necessários para remover ou trocar as permissões de um usuário.
b) Atribuição e remoção do acesso a objetos e recursos do sistema: Quais são os passos necessários para inserir ou remover a permissão de acesso a um objeto do sistema
c) Grau que o sistema suporta o princípio de menor privilégio: O usuário só tem acesso às informações e recursos que são estritamente necessários para realizar suas atividades.
d) Grau que o sistema suporta o princípio de separação de deveres: O sistema deve restringir a quantidade de poderes para um único indivíduo
e) Passos necessários para criar uma política de controle de acesso: Qual a dificuldade de se criar uma política de controle de acesso. Quais são os passos necessários.
f) Modelo permite a evolução ou alteração das políticas de controle de acesso: Esse critério de avaliação tem como objetivo identificar se o modelo de controle de acesso permite que a política de segurança possa ser alterada. Muitos sistemas estão restritos apenas a uma única política de segurança, não permitindo a alteração do modelo usado. Por exemplo, um sistema que aplica a política DAC em que não é possível evoluir para uma política RBAC.
g) Escopo Horizontal: Dependendo da arquitetura do sistema, o mecanismo de controle de acesso pode estar limitado ao escopo de plataforma, aplicação ou a ambiente corporativo. Por exemplo, o escopo pode ser apenas um único host, uma rede distribuída ou um sistema em grid.

h) Escopo Vertical: O escopo do mecanismo de controle de acesso pode se estender desde o core do sistema operacional até as camadas mais altas do sistema. Por exemplo, um sistema gerenciador de banco de dados pode usar um mecanismo de segurança, mas ser integrado com outra política de segurança do sistema operacional para autorizar o usuário primeiro antes do seu controle de acesso ser executado. O escopo vertical é avaliado pelo número de aplicações que o mecanismo é capaz de integrar.
i) Suporte à segurança: Capacidade de garantia de cumprimento das políticas de segurança. Pode ser medido pelo diferente número de tipos de restrições que o mecanismo pode suportar
j) Grau de liberdade no gerenciamento do controle de acesso: administradores do controle de acesso conseguem visualizar as permissões dos usuários e dos recursos.
k) Desempenho na execução das políticas de segurança: Quantas operações são requeridas para o sistema permitir o acesso a um recurso.
l) Conflitos na política de segurança que o sistema pode resolver ou prevenir: Como o sistema age se duas regras entram em conflito

3.4AVALIAÇÃO DOS MODELOS DE CONTROLE DE ACESSO

Para cada um dos critérios definidos pelo NIST e listados na seção anterior, os modelos de controle de acesso RBAC e ABAC serão avaliados.

a) Atribuição e remoção de permissões de usuários do sistema

No RBAC, as permissões são atribuídas a papéis, e os papéis são atribuídos aos usuários (Conforme Figura 3). Logo é possível inserir ou remover novos papéis a usuários para atribuir ou remover permissões. Por exemplo, se um funcionário de uma empresa mudar de cargo dentro da organização, basta remover os papeis relativos ao cargo antigo e associar os novos.

No ABAC, os papéis dos usuários também fazem parte de seus atributos que são avaliados, logo para incluir ou remover permissões do usuário basta atribuir ou remover os papéis correspondentes.

b) Atribuição e remoção do acesso a objetos e recursos do sistema

No RBAC, as permissões de acesso a objetos e recursos do sistema são atribuídas a papéis, e não diretamente a usuários do sistema (Conforme Figura 3). Logo para atualizar as permissões de acesso a objetos basta atualizar as permissões dos papeis, e todos os usuários que possuírem aquele papel terão suas permissões atualizadas. Por exemplo, se o recurso do sistema "Relatório de Vendas" não puder mais ser acessado pelos gerentes, e sim pelos diretores. Para isso basta remover a permissão de acesso a esse recurso do papel "Gerente" e acrescentar no papel "Diretor".

No ABAC, as permissões de acesso a objetos e recursos do sistema são feitas por regras que usam atributos do recurso, do sujeito (incluindo seus papéis) e do ambiente para conceder ou não o acesso. Para atribuir ou remover acesso a um objeto do sistema, as regras precisariam ser atualizadas para refletir os papéis que devem ter acesso a o recurso. Na seção 3.2 a regra abaixo foi apresentada como um exemplo de regra em ABAC:

R1: pode_acessar(s, r, e) ← (Role(s) = 'Gerente') ^ (Nome(r) = 'Aprovar Compra') ^ (Hora(e) = 'Horário comercial')

Caso o perfil gerente não possa mais acessar esse recurso, que passará a ser acessado apenas por usuários com o perfil diretor, a regra deve ser alterada para:

R1: pode_acessar(s, r, e) ← (Role(s) = 'Diretor) ^ (Nome(r) = 'Aprovar Compra') ^ (Hora(e) = 'Horário comercial')

Dessa forma os usuários que possuem o papel de Gerente não conseguirão mais acessar esse recurso, e os usuários que possuem o papel de Diretor passarão a conseguir acessar esse recurso, sem precisar alterar cada usuário individualmente.

c) Grau que o sistema suporta o princípio de menor privilégio

No RBAC, o princípio de menor privilégio é suportado, pois é possível configurar apenas as permissões necessárias aos papéis para atender as tarefas realizadas por membros daquele papel. Cabe à organização analisar e atribuir corretamente as permissões que são realmente necessárias para cada papel para que esse princípio seja suportado.

No ABAC, o princípio de menor privilégio é suportado, pois é possível definir regras que permitam que o usuário acesse apenas o que for necessário para realizar suas atividades. Além de poder restringir o acesso apenas pelo papel do usuário, no ABAC ainda é possível utilizar atributos do ambiente para restringir ainda mais o acesso aos recursos, podendo limitar, por exemplo, o acesso a recursos fora do horário comercial. Assim como no RBAC, cabe à organização definir as regras corretamente para que o princípio seja suportado.

d) Grau que o sistema suporta o princípio de separação de deveres

No RBAC, o princípio de separação de deveres estático (*Static Separation of Duties* - SSD) pode ser atingido fazendo com que papéis mutualmente exclusivos sejam usados para realizar uma tarefa sensível. Por exemplo, para autorizar uma transferência bancaria, dois papeis diferentes precisam ser utilizados para completar a tarefa: assistente de conta e gerente da agência. Outro exemplo é definir que um usuário não pode ser membro dos papéis "Compra" e "Aprovação de Compra". O RBAC também suporta o princípio de separação de deveres dinâmica (*Dynamic Separation of Duties* – DSD). Na DSD, o usuário pode ser membro dos dois papéis, mas não pode usar os dois na mesma transação. Utilizando o exemplo anterior, o usuário pode ser membro dos papéis "Compra" e "Aprovação de Compra", mas não pode aprovar uma compra que ele mesmo fez.

No ABAC os princípios de separação de deveres estático e dinâmico também podem ser atingidos de acordo com as regras que são montadas. É possível definir que para acessar determinado recurso o usuário tem que ter um papel, mas não pode ter outro papel (mutualmente exclusivo). É possível também determinar que para realizar uma aprovação de compra, a compra não pode ter sido feita pelo mesmo usuário.

e) Passos necessários para criar uma política de controle de acesso.

No RBAC, para criar uma política de controle de acesso primeiro é necessário mapear todos os objetos (ou recursos) e operações que podem ser realizadas nesses objetos (chamado de permissões). Posteriormente é necessário criar os papéis existentes na empresa, e quais permissões cada papel possui (ou seja, quais operações em cada objeto podem ser feitas). O RBAC suporta também hierarquia de papéis, logo as permissões definidas para um papel pai são automaticamente herdadas pelo papel filho. Por último os usuários são mapeados com os papeis identificados pela organização.

No ABAC, assim como no RBAC, para criar uma política de controle de acesso é necessário mapear os recursos e operações que podem ser realizadas. Também é necessário fazer o levantamento dos atributos dos recursos que podem ser usados na tomada de decisão. Os atributos relevantes do sujeito também devem ser levantados, como por exemplo, papel, origem, etc. Por último devem ser levantados os atributos de ambiente que serão usados, como por exemplo, a data e hora do sistema, dia da semana, tipo de acesso (internet, intranet) e outros. Feito o levantamento de todos os atributos do recurso, sujeito e ambiente, as regras são criadas e armazenadas no serviço de política de segurança, no formato XACML.

f) Modelo permite a evolução ou alteração das políticas de controle de acesso

O modelo RBAC é uma evolução de outras políticas de segurança, o DAC e MAC, logo um sistema que utiliza o modelo de controle de acesso RBAC pode ser adaptado para utilizar apenas a identidade do usuário para controlar o acesso, ao invés de fazer o controle através dos papéis. E um sistema que utiliza o RBAC pode evoluir e passar a utilizar o modelo ABAC, acrescentando outros tipos de atributos (como atributos de ambiente) para ajudar na tomada da decisão.

Da mesma forma, o modelo ABAC é uma evolução do modelo RBAC, e um sistema que utiliza esse modelo de controle de acesso pode ser adaptado para utilizar apenas os papéis na tomada de decisão, ou utilizar apenas a identidade do usuário.

g) Escopo Horizontal

No modelo RBAC, como o controle de acesso é feito no nível da API do sistema de *Cloud Computing*, os recursos são expostos pelos provedores de serviço através de uma interface. Para isso o usuário deve se autenticar no sistema, que então determina quais os recursos estão disponíveis para os papeis disponíveis na sessão do usuário. Com isso o escopo horizontal desse modelo abrange diferentes aplicações e plataformas do sistema.

No modelo ABAC, para acessar qualquer recurso o usuário também deve ter sido autenticado, e baseado nos atributos do sujeito e do recurso a ser acessado, as regras específicas do recurso são carregadas e avaliadas. Assim como no RBAC o escopo horizontal desse modelo abrange diferentes aplicações e plataformas do sistema. O ABAC leva vantagem no sentido que também são levados em conta os atributos do ambiente, que podem oferecer informações e características relevantes sobre as propriedades do sistema, aumentando o escopo horizontal entre diferentes plataformas e aplicações.

h) Escopo Vertical

Tanto no RBAC quanto no ABAC, o escopo vertical é apenas a API disponibilizada pelos provedores de *Cloud Computing* para acessar os recursos. Não existe nenhuma integração com as outras camadas do sistema.

i) Suporte à segurança

No RBAC é possível criar restrições de segurança na atribuição entre usuário e papel, na hierarquia entre os papéis e durante a sessão do usuário (Conforme Figura 3). Na atribuição entre usuário e papel é possível definir papéis mutualmente exclusivos. Por exemplo, um usuário que tem o perfil "Caixa Bancário" não pode possuir o perfil "Supervisor de Caixa". Na atribuição de perfis para o usuário, se um dos perfis for atribuído ao usuário, o outro não poderá ser atribuído. A restrição obedece à hierarquia de papéis. Se um papel pai tiver um relacionamento mutualmente exclusivo com outro papel, todos os seus papéis filhos também herdam a mesma restrição. A restrição durante a sessão do usuário ocorre quando o usuário pode ter dois

papeis que são conflitantes, desde que os dois não estejam ativos simultaneamente na sessão do usuário. Utilizando o mesmo exemplo anterior, um usuário pode possuir os dois perfis "Caixa Bancário" e "Supervisor de Caixa", mas somente um deles pode estar ativo durante a sessão.

No ABAC é possível definir as mesmas restrições que existem no RBAC. Isso pode ser feito reescrevendo as regras de controle de acesso. Por exemplo, uma regra pode ser escrita de forma que um recurso pode ser acessado caso o usuário possua o perfil "Caixa Bancário", mas não possua o perfil "Supervisor de Caixa":

R1: pode_acessar(s, r, e) ← (Role(s) = 'Caixa Bancário) ^ (Role(s) != 'Supervisor de Caixa')

Além das restrições disponíveis no RBAC, o ABAC ainda tem a capacidade de impor restrições baseadas nos atributos do ambiente, como data, hora, dia semana, tipo de acesso (internet, intranet), que permitem uma maior flexibilidade e segurança na criação das regras.

j) Grau de liberdade no gerenciamento do controle de acesso

No RBAC, analisando sua estrutura de dados (conforme Figura 3), o administrador das políticas de controle de acesso consegue consultar para um determinado usuário quais papéis ele possui, e para um determinado papel é possível consultar quais usuários são membros daquele papel. É possível também consultar quais permissões de acesso a recursos um papel possui, e também quais papéis possuem permissão de acesso a um determinado recurso.

No ABAC, o administrador das políticas de controle de acesso não consegue consultar diretamente quais usuários tem acesso a um determinado recurso devido à forma como as regras são criadas, e devido à complexidade criada ao se adicionar os atributos de ambiente na tomada de decisão. O administrador consegue visualizar os papéis associados a um usuário, e consegue consultar as regras para acessar um recurso.

k) Desempenho na execução das políticas de segurança

Para avaliar o desempenho na execução das políticas de segurança o ideal seria ter dois ambientes similares, cada um rodando um dos modelos para poder executar e comparar os desempenhos. O que o NIST sugere na avaliação desse critério é comparar o número de operações necessárias para o sistema permitir ou negar uma solicitação de acesso. Esse será o critério utilizado.

No modelo proposto utilizando RBAC (Figura 4), para cada requisição de acesso a um recurso são realizados dois passos: consulta na base de dados de usuários registrados no sistema, e validação dos papéis que aquele usuário possui.

No modelo proposto utilizando ABAC (Figura 5), para cada requisição de acesso a um recurso são realizados cinco passos: interceptação pelo mecanismo de garantia de política de segurança (PEP), validação do certificado pelo mecanismo de validação de certificado (CVS), autorização pelo serviço de decisão de política de segurança (PDS), que faz uma consulta ao serviço de política de segurança (PS) para carregar as politicas no formato XACML, e também ao serviço de atributos (AS), que carrega os atributos necessários na tomada de decisão.

l) Conflitos na política de segurança que o sistema pode resolver ou prevenir

No RBAC não existe nenhum procedimento para resolver ou prevenir conflitos na política de segurança. Porém, o RBAC está menos suscetível a esse problema, já que o conflito na política de segurança ocorre quando uma regra *r* referencia outra regra *s*, que referencia novamente *r*, causando um *deadlock*. No RBAC, como as permissões do usuário são carregadas a partir dos papéis disponíveis na sessão do usuário, é mais difícil ocorrer conflitos nesse modelo.

No modelo ABAC não existe nenhum mecanismo para resolver ou prevenir conflitos na política de segurança, mas HUONDER (2010) propôs um modelo de detecção e resolução de conflitos para políticas definidas utilizando XACML (HUONDER, 2010). Foram propostos algoritmos para detecção de conflitos, sendo um deles específico para ambientes dinâmicos, onde a estrutura de políticas está em constante mudança, e também

algoritmos para resolução de conflitos. Um dos algoritmos para resolução de conflitos proposto pode ser usado para recombinar as políticas em novas combinações. Com esse algoritmo, um administrador pode otimizar as políticas de segurança e reduzir ou prevenir os conflitos.

3.5 RESULTADO DA AVALIAÇÃO DOS MODELOS DE CONTROLE DE ACESSO

Na seção anterior os modelos de controle de acesso foram analisados utilizando os critérios definidas pelo NIST (2006) para avaliação de sistemas de controle de acesso. Nessa seção será feito um comparativo da avaliação feita na seção anterior, destacando itens positivos e negativos de cada modelo destacando o modelo que melhor se adapta no contexto de *Cloud Computing.*

O modelo RBAC é muito simples e fácil de usar. Os papeis são atribuídos aos usuários de forma estática pelo administrador de segurança. Isso pode ser um problema em algumas situações como ambientes colaborativos e com muitas mudanças. Outro problema com os papéis é que diferentes organizações possuem diferentes nomenclaturas e estruturas para definir os papéis, o que torna difícil diferenciar papeis em diferentes contextos. Isso faz com que o número de papeis cresça muito. Em alguns casos o número de papéis supera o número de usuários, tornando a manutenção das atribuições entre papeis e usuários difícil. Outro fator que faz com que o número de papéis cresça é que o RBAC não suporta atributos dinâmicos, como por exemplo, o dia e a hora para auxiliar na tomada de decisão. Por isso, para suportar esses atributos dinâmicos, algumas organizações criam papéis diferentes para tentar simular esses atributos dinâmicos, fazendo com que o número de papeis cresça.

O modelo ABAC é mais flexível que o modelo RBAC, e funciona bem em ambientes dinâmicos e distribuídos, pois além de levar em conta as características do recurso e do sujeito na tomada de decisão, os atributos de

ambiente (dinâmicos) também podem ser usados, o que aumenta a flexibilidade do modelo. Essa maior flexibilidade também traz problemas na especificação e manutenção das políticas de segurança. O administrador de segurança deve saber do esquema de atributos usados pelas organizações. Cada organização utiliza o seu próprio esquema de atributos, mas que em alguns casos tem o mesmo significado. O administrador deve fazer uma relação entre eles. Uma solução para esse caso é definir um conjunto de atributos padrão para ser adotado por todas as organizações que fazem parte do sistema. Essa solução pode ser feita criando uma base de dados centralizada em que os atributos de usuários são armazenados utilizando um formato padrão. Uma base de dados desse tipo é benéfica, pois padroniza os atributos entre as organizações, aumenta o compartilhamento entre elas e cria um acordo de atributos que podem ser usados na tomada de decisão. Porem, utilizando essa solução, parte da flexibilidade na criação das regras utilizando os atributos dinâmicos é perdida, pois as organizações ficam restritas aos atributos previamente estabelecidos (PRIEBE et al., 2006).

No RBAC, para criar as permissões de acesso pela primeira vez devem ser levantados os recursos, papéis que tem permissão para acessar aqueles recursos, e depois associar os papéis com os usuários. Depois que a lista de usuários, perfis e permissões estiverem prontas, qualquer alteração nas permissões deve ser feita removendo ou adicionando um perfil ao usuário, ou removendo e adicionando uma permissão a um perfil. Quando a quantidade de usuários, perfis e recursos é pequena, isso não é difícil, mas em um ambiente com muitos usuários, perfis e recursos, e com constantes mudanças, passa a ser difícil manter atualizadas as relações entre usuário, perfil e recurso.

No ABAC, para criar as permissões de acesso pela primeira vez é mais trabalhoso, uma vez que devem ser definidas as regras para cada recurso utilizando os atributos do sujeito, recurso e ambiente. Para qualquer alteração nas permissões de acesso a recurso, as regras devem ser atualizadas para refletir as mudanças. O problema é que quando ocorrem muitas mudanças nas regras, podem ocorrer inconsistências, como por exemplo, duas regras

que dependem uma da outra e ocasionam uma situação de *deadlock*. Por padrão, o ABAC não possui nenhum mecanismo para identificar e corrigir situações desse tipo, mas já existem modelos para identificar e corrigir problemas desse tipo em sistemas que utilizam XACML para definir suas políticas de segurança. Em um ambiente dinâmico e com muitas alterações, como é um ambiente de *Cloud Computing*, um mecanismo desse tipo é importante.

O exemplo abaixo apresenta os pontos positivos e negativos de RBAC e ABAC.

Supondo que para disponibilizar um novo recurso no ambiente *Cloud*, as seguintes regras devem ser seguidas:

- Os serviços que podem ser disponibilizados são divididos em três categorias: SaaS, PaaS e IaaS.
- Os serviços são disponibilizados aos usuários de acordo com a experiência deles no cargo (tempo na empresa).
- Analistas com até 5 anos de empresa só podem disponibilizar recursos do tipo SaaS, analistas com até 10 anos podem disponibilizar recursos do tipo PaaS e analistas com mais de 15 anos podem disponibilizar recursos do tipo IaaS.

A tabela 1 resume as definições destas regras.

Tabela 1- Definição das regras de um sistema para modelagem utilizando RBAC e ABAC.

Tipo de serviço	Usuários permitidos
SaaS	Até 5 anos de empresa
PaaS	Até 10 anos de empresa
IaaS	15 anos ou mais de empresa

Para definir estas regras utilizando RBAC, são criados três perfis pré-definidos:

- Analista Junior (até 5 anos),
- Analista Pleno (até 10 anos), e
- Analista Sênior (15 anos ou mais).

Cada usuário tem um dos perfis associados a ele. Além disso, são criadas três permissões:

- Pode solicitar serviço SaaS,
- Pode solicitar serviço PaaS,
- Pode solicitar serviço IaaS.

Dessa forma, o perfil Analista Senior recebe as três permissões. O perfil Analista Pleno recebe as permissões de SaaS e PaaS, e o perfil de analista Junior recebe a permissão de SaaS.

As atividades de atribuição de usuário e papel e atribuição de papel e permissão são tarefas manuais que devem ser executadas pelo administrador do sistema.

No ABAC, para definir a mesma regra não é necessário definir papéis. Para um usuário *u* criar um serviço *s* em um ambiente *e*, a seguinte regra pode ser utilizada:

R1: pode_acessar(*u*, *s*, *e*) ←
(TempoServico(u) ≥ 15 ^ TipoServico(s) ∈ {SaaS, PaaS, IaaS}) ∨
(TempoServico(u) > 5 ^ TipoServico(s) ∈ {SaaS, PaaS) ∨
(TipoServico(s) ∈ {SaaS})

A vantagem do ABAC demonstrada aqui é que elimina as definições e gerenciamento estático de papéis e também a necessidade de tarefas administrativas na atribuição de usuário e papel e de papel e permissão.

Políticas de controle de acesso mais refinadas normalmente envolvem diversos usuários e objetos. Nesse caso, o ABAC se mostra mais maleável e

escalável do que o RBAC. Para demonstrar isso, pode-se considerar a expansão do exemplo anterior para incluir duas categorias dentro do tipo de serviço, baseado nos atributos do serviço: categoria Premium e categoria Básica.

Na categoria Básica existem certas limitações nos recursos, e na categoria Premium o usuário tem a liberdade para alterar a configuração de acordo com o desejado.

Além disso, os usuários também serão classificados em supervisores e não supervisores.

Para criar uma nova política de forma que usuários não supervisores só podem solicitar recursos da categoria Básica, e apenas supervisores podem solicitar recursos na categoria Premium, as seguintes alterações são necessárias:

No RBAC, seria necessário criar novos papéis e novas permissões conforme tabela 2:

Tabela 2 - Papéis e Permissões que devem ser criados para atender a nova regra

Papéis RBAC	Permissões RBAC
Analista Junior	Pode solicitar SaaS Basico
Analista Junior Supervisor	Pode solicitar SaaS Premium
Analista Pleno	Pode Solicitar PaaS Básico
Analista Pleno Supervisor	Pode Solicitar PaaS Premium
Analista Sênior	Pode solicitar IaaS Básico
Analista Sênior Supervisor	Pode Solicitar IaaS Premium

Dessa forma, o perfil Analista Junior tem apenas a permissão SaaS Básico, o Analista Junior Supervisor tem a permissão SaaS Básico e Saas Supervisor,

e assim por diante até o Analista Sênior Supervisor, que tem todas as permissões.

À medida que as políticas vão ficando mais refinadas e mais atributos são envolvidos, o número de papéis cresce, tornando as atribuições entre usuário e papel, e entre papel e permissão tarefas complicadas e demoradas.

No ABAC, para contemplar a nova política, a regra R1 definida anteriormente continua valendo, sendo necessário incluir duas novas regras (R2 e R3):

R2: pode_acessar(*u, s, e*) ←
((CategoriaServico(s) = 'Premium') ∧(TipoUsuario(u) = 'Supervisor')) ∨
(CategoriaServico(s) = 'Básico')

R3: pode_acessar(*u, s, e*) ← R1 ∧ R2
Se as regras R1 e R2 forem atendidas, o usuário *u* pode solicitar o serviço *s*.

No exemplo anterior não foram utilizadas as variáveis de ambiente. Seria difícil implementar uma regra em RBAC que permite usuários não supervisores só solicitarem recursos durante a semana e que durante o fim de semana somente usuários supervisores poderem solicitar recursos. Já utilizando o ABAC, bastaria acrescentar mais algumas regras utilizando os atributos do ambiente para verificar o dia da semana.

Os pontos fracos e fortes de cada modelo estão colocados nas tabelas 3 e 4:

Tabela 3 - Pontos positivos e negativos do modelo RBAC

Pontos Positivos	Pontos Negativos
Simplicidade	Não é possível alterar as permissões de acesso de uma entidade sem alterar seus papéis
Fácil de usar	Não permite utilizar atributos dinâmicos na tomada de decisão
Baixa complexidade, já que os papéis são estáticos	
Bom para domínios locais	

Tabela 4 - Pontos positivos e negativos do modelo ABAC

Pontos Positivos	Pontos Negativos
Flexível, principalmente em um sistema grande onde o número de usuários é alto	Alta complexidade para especificar e manter as políticas de segurança
Utiliza atributos dinâmicos, do usuário, do recurso e do ambiente na tomada de decisão	Falta de padronização dos atributos entre diferentes provedores de serviço quando não é utilizado um repositório central
Não usa os atributos para definir a permissão diretamente entre o usuário e o recurso	Conflitos entre as políticas de segurança definidas.
Utilizando um repositório central de atributos, as organizações podem padronizar e compartilhar os tipos de atributos usados na tomada de decisão.	

Através do exemplo acima, pode-se observar que ambos os modelos de controle de acesso possuem vantagens e desvantagens, e podem se adaptar melhor em determinados ambientes. O controle de acesso baseado em atributos (ABAC) mostra ser mais adequado para um ambiente de *Cloud Computing* devido à dinamicidade de um ambiente desse tipo, além das constantes mudanças que ele sofre. Os atributos de ambiente também podem ser considerados um fator importante, pois aumenta a flexibilidade na criação das regras. Além disso, utilizando o ABAC é possível simular o RBAC, utilizando apenas o papel dos usuários nas regras definidas.

O RBAC seria mais adequado para ambientes não distribuídos, onde o número de usuários não é tão grande, e as alterações nos relacionamentos entre usuários e papéis e papéis e permissões não são tão frequentes. Nesses casos a simplicidade do RBAC seria mais recomendada do que utilizar o ABAC.

A Tabela 5 resume a análise dos modelos RBAC e ABAC de acordo com os critérios definidos pelo NIST

Tabela 5 - Resumo da análise dos modelos utilizando os critérios do NIST

Critério	RBAC	ABAC
Atribuição e remoção de permissões de usuários	Permissões são atribuídas a papéis e não diretamente aos usuários	Pode utilizar papéis ou outros atributos do sujeito na tomada de decisão.
Atribuição e remoção de permissões de acesso a recursos	Permissões são atribuídas aos papéis e não diretamente aos usuários	As regras de acesso ao recurso devem ser alteradas para refletir as mudanças
Suporta princípio de menor privilégio	Suporta	Suporta
Suporta princípio de separação de deveres	Suporta separação estática (SSD) e dinâmica (DSD)	Suporta separação estática (SSD) e dinâmica (DSD)

Passos para criar política de segurança	Mais simples. Basta levantar os objetos, mapear os papéis e associar aos usuários	Mais complexo. Deve levar em conta atributos do sujeito, recurso e ambiente. As regras devem ser escritas utilizando esses atributos
Permite evolução/alteração das políticas de segurança	Permite alteração para utilizar apenas a identidade	Permite utilizar apenas a identidade do usuário ou apenas papeis
Escopo Horizontal	Abrange diferentes aplicações e plataformas do sistema	Abrange diferentes aplicações e plataformas do sistema
Escopo Vertical	Apenas API disponibilizada pelo provedor	Apenas API disponibilizada pelo provedor
Suporte à segurança	Permite restrições entre usuário e papel, na hierarquia de papéis e na sessão do usuário	Permite as mesmas restrições suportadas no RBAC. Permite restrições baseadas nos atributos de ambiente e de recurso
Liberdade no gerenciamento do controle de acesso	Maior liberdade. Consegue consultar diretamente os usuários que possuem acesso a um recurso	Menor liberdade. Não consegue consultar diretamente os usuários que possuem acesso a um recurso. Depende das regras e dos atributos utilizados
Desempenho na execução das políticas de segurança	Menos passos para tomar a decisão (2 passos)	Mais passos para tomar a decisão (5 passos)

Conflitos que o sistema pode resolver ou prevenir	Nenhum mecanismo definido, porém está menos sujeito a conflitos	Nenhum mecanismo definido, mas já existem modelos para detectar e resolver conflitos em políticas definidas utilizando XACML

4 CONCLUSÃO

Nesse trabalho foram apresentados os modelos de controle de acesso baseado em papéis (RBAC) e baseado em atributos (ABAC), e uma aplicação desses modelos em ambientes de *Cloud Computing*. Posteriormente foram apresentadas os critérios de qualidade para sistemas de controle de acesso definidas pelo *National Institute of Standards and Technology* (NIST, 2006). Os dois modelos foram analisados utilizando todos os itens dessa métrica, e por ultimo foi feito um comparativo entre os dois modelos, levantando pontos fortes e fracos de cada um deles. Além disso, foi dado um exemplo real de como seria para modelar as regras de um sistema utilizando cada um dos modelos. Para o RBAC, quais papéis e permissões seriam criados, e para o ABAC quais regras seriam necessárias para atender os requisitos. Depois foram acrescentadas mais algumas regras para mostrar como em algumas situações fica difícil mapear as regras utilizando apenas papéis, e como os atributos ajudam na hora de criar as regras de tomada de decisão.

Após a análise, a conclusão obtida é que o modelo RBAC seria mais recomendado para sistemas não distribuídos, onde o número de usuários não é muito grande, e as alterações não são constantes, já que à medida que aumenta o número de usuários e alterações nos relacionamentos entre usuários e papéis, e papéis e permissões, a administração das políticas de segurança fica complexa e demorada.

O modelo ABAC seria mais recomendado para ambientes de *Cloud Computing*, pois atende melhor as necessidades de um ambiente dinâmico, com alterações constantes. Os atributos de recurso, sujeito e ambiente facilitam a formar regras dinâmicas, que não dependem apenas do papel atribuído a um usuário. Além disso, o ABAC pode simular o comportamento do modelo RBAC, utilizando apenas o papel do sujeito em suas regras. Como ponto negativo do ABAC é possível destacar a alta complexidade para especificar e manter as políticas de segurança, já que a modificação de alguma regra pode causar um conflito na política de segurança, e a falta de padronização de atributos entre diferentes provedores de serviço.

4.1 TRABALHOS FUTUROS

Como sugestão para trabalhos futuros, seria interessante estudar outros modelos de controle de acesso não tão conhecidos, como por exemplo, o *Task-Based Access Control* (TBAC), que é uma nova perspectiva de controle de acesso baseado em tarefas, ou o *Hierarchical Attribute-Based Encryption Acces* (HABE). Além disso, outra sugestão de trabalho futuro é a de analisar as ferramentas de controle de acesso oferecidas pelos principais fabricantes de *Cloud Computing*, e fazer uma analise dentre as existentes qual a melhor opção.

REFERÊNCIAS

ANDREWS, G. R.; "Foundations of Multithreaded, Parallel, and Distributed Programming", Addison-Wesley, 1999

BANDYOPADHYAY, S.; "Cloud Computing: the business perspective", 2009, Disponível em: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1413545. Acessado em 22 Nov. 2012.

CLAUNCH, C.; CEARLEY, D.; "Top 10 Strategic Technology Areas for 2009", ITxp Synopsium Orlando, 2008.

CLOUD SECURITY ALLIANCE (CSA), "Security Guidance for Critical Areas of Focus in Cloud Computing", 2009

ERL, T.; "SOA: Principles of Service Design", Prentice Hall, 2009

GOSAC-N, "Technical PBAC Reference Implementation", Government Open Source Access Control - Next Generation, 2011. Disponível em: <http://www.gosac-n.org/solution>. Acesso em: 05 Jan. 2013.

GUTIERREZ VELA, F.L.; ISLA MONTES, J.L.; PADEREWSKI RODRIGUEZ, P.; SANCHEZ ROMAN, M.; JIMENEZ VALVERDE, B.; "An architecture for access control management in collaborative enterprise systems based on organization models", Science of Computer Programming 66, 2007, pp. 44–59

HUONDER, F.; "Conflict Detection and Resolution of XACML Policies", 2010, Tese (Mestrado),University of Applied Sciences Rapperswil, Suiça.

INTEL IT CENTER, "What's holding back the Cloud? Intel Survey on Increasing IT Professional's Confidence in Cloud Security". Disponível em: <http://www.intel.com/content/www/us/en/cloud-computing/whats-holding-back-the-cloud-peer-research-report.html >. Acesso em: 08 Nov. 2012.

KHAN, A. R.; "Access Control in Cloud Computing Environment", ARPN Journal of Engineering, Vol. 7, No. 5, 2012

MARKS, E. A.; Lozano, R. R.; "Executive's guide to cloud computing", Hoboken, 2010

MOTAHARI-NEZHAD, H. R.; Stephenson, B.; Singhal, S.; "Outsourcing Business to Cloud Computing Services: Opportunities and Challenges", 2009

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST), "Assessment of Access Control Systems", Interagency Report 7316, 2006

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST), "The NIST definition of Cloud Computing", 2009

ADVANCING OPEN STANDARDS FOR THE INFORMATION SOCIETY (OASIS), "Extensible Access Control Markup Language (XACML) Version 3.0", 2010. Disponível em http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.pdf. Acessado em 02/02/2013.

OPEN GROUP, "The Open Group Service Integration Maturity Model (OSIMM)", 2011, Disponível em: http:// www.opengroup.org/soa/source-book/osimmv2/index.htm . Acessado em 20 Nov. 2012.

PAPAZOGLOU, M.P.; "Service-oriented computing: concepts, characteristics and directions", International Conference on Web Information Systems Engineering (WISE 2003), pp. 3-12, 2003.

PRIEBE, T.; DOBMEIER, W.; SCHLAGER, C.; KAMPRATH, N.; "Supporting Attribute-based Access Control in Authorization and Authentication Infrastructures with Ontologies", First International Conference on Availability, Reliability and Security (ARES 2006), Vienna, Austria, April 2006.

SANDHU, R.; COYNE, E.; FEINSTEIN, H.; YOUMAN, C.; "Role-Based Access Control Models", IEEE Computer, Vol. 29, No. 2, pp. 38-47, 1996

SANDHU, R.; FERRAIOLO D.; KUHN, R.; "The NIST Model for Role-Based Access Control: Towards a Unified Standard", Proceedings of the fifth ACM workshop on Role-based access control, pp. 47-64, 2000

SIRISHA, A.; KUMARI, G.; "API Access Control in Cloud Using the Role Based Access Control Model", Trendz in Information Sciences & Computing (TISC), pp. 135-137, 2010.

SLAMA, D.; BANKE, K.; KRAFZIG, D.; "Enterprise SOA: Service Oriented Architecture Best Practices", Prentice Hall, 2004

WEISS, A.; "Computing in the Clouds", ACM networker 11(4), pp.16-25, 2007

YUAN, E.; TONG, J.; "Attributed Based Access Control (ABAC) for Web Services", Proceedings of the IEEE International Conference on Web Services, IEEE Computer Society, 2005