VILMAR ABREU JUNIOR

MODELO DE ATIVAÇÃO MULTIDOMÍNIO DE PAPÉIS RBAC

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná, como requisito parcial à obtenção do título de mestre em Informática.

CURITIBA

2016

VILMAR ABREU JUNIOR

MODELO DE ATIVAÇÃO MULTIDOMÍNIO DE PAPÉIS RBAC

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná, como requisito parcial à obtenção do título de mestre em Informática.

Área de Concentração: Ciência da Computação

Orientador: Prof. Dr. Altair Olivo Santin

CURITIBA

2016

Abreu Júnior, Vilmar

Modelo de ativação multidomínio de papéis RBAC; Vilmar Abreu Júnior; orientador: Altair Olivo Santin. – 2016.

xii, 55 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2016

Bibliografia: f. 53-55

1. Redes de computação – Medidas de segurança. 2. Arquitetura de computadores. 3. Cliente/servidor (Computadores). Computadores – Controle de acesso. I. Santin, Altair Olivo. II. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. III. Título.

CDD 21. ed. – 005.8

Pontifícia Universidade Católica do Paraná

PUCPR
GRUPO MARISTA

ATA DE DEFESA DE DISSERTAÇÃO DE MESTRADO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

DEFESA DE DISSERTAÇÃO DE MESTRADO Nº 17/2016

Aos 29 dias do mês de Fevereiro de 2016 realizou-se a sessão pública de Defesa da Dissertação **"Modelo de Ativação Multidomínio de Papéis RBAC"** apresentado pelo aluno **Vilmar Abreu Junior**, como requisito parcial para a obtenção do título de Mestre em Informática, perante uma Banca Examinadora composta pelos seguintes membros:

Prof. Dr. Altair Olivo Santin
PUCPR (Orientador) _____ (assinatura) Aprov (Aprov/Reprov)

Prof. Dr. Carlos Maziero
UFPR _____ (assinatura) APROV (Aprov/Reprov)

Prof. Dr. Rafael Rodrigues Obelheiro
UDESC _____ (assinatura) APROV (Aprov/Reprov)

Conforme as normas regimentais do PPGIa e da PUCPR, o trabalho apresentado foi considerado Aprovado (*aprovado/reprovado*), segundo avaliação da maioria dos membros desta Banca Examinadora. Este resultado está condicionado ao cumprimento integral das solicitações da Banca Examinadora registradas no Livro de Defesas do programa.

Programa de Pós-Graduação em Informática PPGIa PUCPR

Prof.ª Dr.ª Andreia Malucelli
Coordenadora do Programa de Pós-Graduação em Informática.

Dedico este trabalho aos meus familiares e amigos que sempre me apoiaram e aconselharam ao longo de minhas conquistas.

Agradecimentos

Agradeço ao meu orientador, Altair Olivo Santin, pelo apoio, não somente relativo à orientação desse trabalho, mas por toda a ajuda nesse período de formação acadêmica que é o mestrado. Agradeço também aos meus prezados amigos Cleverton Vicentini, Eduardo Viegas e Rafael Ribeiro pela amizade e companhia durante essa jornada. Finalmente, agradeço ao apoio dos meus pais Vilmar Abreu e Sueli do Rocio Abreu, e das minhas irmãs, Andreia Abreu e Sandra Abreu, por sempre me apoiarem incentivarem.

Resumo

Organizações estabelecem parcerias a fim de atingir determinados objetivos estratégicos. Em diversos casos, um domínio administrativo de uma organização realiza o acesso aos recursos no domínio de outra organização, caracterizando uma operação multidomínios. Domínios possuem recursos protegidos que necessariamente devem ser acessados apenas por usuários autorizados. O controle de acesso baseado em papéis (RBAC, *Role-based Access Control*) é amplamente utilizado comercialmente, devido a sua simplicidade na gestão de permissões. Este trabalho apresenta um modelo de ativação de papéis RBAC para ambientes multidomínios. Os papéis ativos de um usuário são importados de seu domínio de origem para outros domínios, realizando a ativação única de papéis (SRA, *Single Role Activation*), similarmente à autenticação única (SSO, *Single Sign-On*). O administrador tem a autonomia para definir as permissões que cada papel poderá ter em seu domínio. Para avaliar a proposta, foi implementado um protótipo baseado em serviços RESTFul que suporta o SRA, utilizando tecnologias consolidadas como o XACML e OpenID Connect. A avaliação de performance do protótipo mostrou um impacto mínimo no tempo de resposta do SRA em relação à ativação tradicional de papéis do RBAC, enquanto provê segurança.

Palavras-chave: *Ativação multidomínio de papéis; RBAC; XACML; OpenID Connect*

Abstract

Organizations establish partnerships in order to reach strategic goals. In many cases, a domain of an organization accesses resources in the domain of another organization, characterizing multi-domain operations. Domains have protected resources that necessarily should be accessed only by authorized users. Role-based Access Control (RBAC) is widely used commercially, due to its simplicity in authorization management. This work presents an approach to perform a RBAC role activation in a multi-domain environment. The active roles are imported from a user home domain to other domains, yielding a Single Role Activation (SRA), similar to Single Sign-On (SSO) authentication. The administrator has the autonomy to define which role permissions are kept on his or her local domain. To evaluate the proposal, it was implemented a prototype based on RESTFul web services to provide support to SRA, using standard specifications such as XACML and OpenID Connect. The performance evaluation showed a minimal impact on SRA response time when compared to a traditional role activation model, while providing security.

Keywords: *Multi-domain Role Activation; RBAC; XACML; OpenID Connect*

Sumário

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ABAC	Attribute-based Access Control
App	Aplicação
CH	Context Handler
DAC	Discricionary Access Control
DSoD	Dynamic Separation of Duty
IdM	Identity Management
IdP	Identity Provider
MAC	Mandatory Access Control
OIDC	OpenID Connect
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
RBAC	Role-based Access Control
RP	Relying Party
SoD	Separation of Duty
SP	Service Provider
SRA	Single Role Activation
SSO	Single Sign-On
SSoD	Static Separation of Duty
XACML	eXtensible Access Control Markup Language

Capítulo 1

Introdução

Este capítulo apresenta a contextualização do trabalho, seguida da motivação, hipótese, objetivos e contribuição. Por fim, será apresentada a organização desse documento.

1.1. Contextualização

Atualmente diversas organizações estabelecem parcerias a fim de atingir seus objetivos estratégicos. Em diversos casos, um determinado domínio administrativo de uma organização necessita acessar o sistema de outra organização, caracterizando uma operação multidomínio. Isso implica em uma flexibilidade no compartilhamento de informações e recursos. Por exemplo, um sistema de gestão de energia nacional poderia acessar diversos sistemas de distribuição e geração de energia regionais para dispor de uma visão macro da situação energética do país.

Um domínio pode possuir inúmeros recursos privados que não devem ser acessados via operações multidomínios. Neste sentido, expor o acesso à aplicação por um outro domínio representa um potencial problema para os sistemas (POWER, 2000).

O controle de acesso é um mecanismo de segurança que coíbe o acesso não autorizado de usuários. A autorização ocorre após à autenticação, e utiliza políticas para limitar o acesso a usuários autorizados e legítimos (SANDHU e SAMARATI, 1994). Os controles de acesso mais populares são: discricionários, obrigatórios, baseados em papéis ou em atributos.

O controle de acesso baseado em papéis (RBAC, *Role-Based Access Control*) é amplamente utilizado em aplicações comerciais e dispõe de diversas vantagens em relação aos controles de acesso tradicionais (discricionário e obrigatório) (OSBORN e SANDHU, 2000). Os papéis RBAC intermedeiam a associação entre o usuário e as permissões, onde as restrições de acesso estão definidas.

No controle de acesso baseado em atributo (ABAC, *Attribute-Based Access Control*) não ocorre a associação de permissões a sujeitos ou papéis no sistema, porém ocorre a associação de permissões a atributos. A avaliação de políticas é efetuada baseada nos atributos, em tempo real, constituindo-se como uma vantagem em relação ao RBAC. Entretanto, o ABAC pode adotar os papéis do RBAC, considerando que os papéis são atributos de um usuário, ou seja, como não são modelos conflitantes, podem ser utilizados juntos.

Tradicionalmente, o RBAC foi concebido para controlar um único domínio. Alguns autores (FREUDENTHAL et al., 2002; LI, 2006; SHAFIQ et al., 2005) apresentaram propostas para aplicação do RBAC em ambientes multidomínios. O principal desafio nas operações multidomínios está relacionado à semântica dos papéis, pois um papel possui diferentes permissões em domínios distintos, ainda que os papéis possuam o mesmo nome. Por exemplo: um papel médico no domínio de um determinado hospital não possui as mesmas permissões que o papel médico possui no domínio de uma clínica médica. Adicionalmente, os recursos e operações entre os domínios podem não ser compatíveis, demandando uma taxonomia para interoperá-los (LI, 2006; SHAFIQ et al., 2005).

Alguns modelos de autorização de acesso para *web* oferecem suporte para operações multidomínios, como, por exemplo, o OAuth (HARDT, 2012). O OAuth limita o acesso a um *path* (URL) protegido, porém não suporta políticas que regrem as operações permitidas nos recursos daquele *path*. Para tal, é necessário um controle de acesso.

Um controle de acesso necessita de uma linguagem para elaborar políticas e também de uma arquitetura (*framework*) responsável pela avaliação destas políticas. O XACML (*eXtensible Access Control Markup Language*) opera como arquitetura e linguagem declarativa de controle de acesso, padronizadas pela OASIS. O XACML utiliza políticas baseadas em atributos, ou seja, adota o modelo ABAC. Entretanto, existe um *profile* (extensão) que permite definir políticas baseadas em papéis RBAC no XACML (ANDERSON, 2004). Como o XACML é *stateless*, mesmo com o *profile* do RBAC, não consegue contemplar todas as características do RBAC, como a ativação e conflitos de papéis, que são características consideradas importantes do RBAC.

1.2. Motivação e Hipótese

Realizar o controle de acesso a recursos protegidos em ambientes multidomínios tornou-se um desafio amplamente explorado na literatura. O RBAC é um controle de acesso amplamente empregado em aplicações comerciais. Isso ocorre porque os papéis determinam um meio natural de estruturar as linhas de autoridade e responsabilidades de uma organização. Inúmeros autores sugerem a utilização do RBAC em ambientes multidomínios, conforme é discutido nos trabalhos relacionados, seção 3.2. Entretanto, até o momento nenhum trabalho na literatura tem o foco na ativação dos papéis em um ambiente multidomínios. Além disso, os trabalhos na literatura não apresentam soluções adequadas de RBAC em ambientes multidomínios.

A proposta deste trabalho é desenvolver um modelo de ativação de papéis RBAC multidomínios com um mecanismo de ativação única de papéis (SRA, *Single Role Activation*), para ambientes multidomínios. Para isso, é necessário um controle de acesso baseado em papéis que dê suporte ao modelo de ativação de papéis multidomínios.

Este trabalho considera a hipótese de que a arquitetura XACML, ao adotar os papéis RBAC como atributos do usuário, integrados às restrições de acesso do OAuth em um ambiente multidomínios, possibilita o suporte para provimento de SRA. O resultado final proporciona a um usuário acesso a diferentes domínios sem necessidade de ativação de papéis que já estejam ativos em seu domínio de origem.

1.3. Objetivos

O objetivo geral do trabalho é o desenvolvimento de um modelo de ativação multidomínios de papéis, utilizando um controle de acesso baseado em atributos. A proposta pretende manter a autonomia do administrador de cada domínio, no sentido de permitir a definição dos direitos associados a cada papel. Ao mesmo tempo, deve facilitar a vida do usuário, evitando a necessidade da ativação de papéis para cada domínio visitado.

Objetivos específicos:

- Projetar e desenvolver um modelo de controle de acesso baseado em atributos que possibilite a utilização de papéis ativos para avaliar a autorização de cada usuário;

- Especificar o modelo de controle de acesso;
- Implementar um mecanismo para importar a ativação de papéis de outros domínios, considerando os possíveis conflitos de interesse relacionados a esses papéis;
- Definir uma sintaxe para escrita de políticas utilizando papéis ativos locais e de outros domínios;
- Avaliar o protótipo desenvolvido que implementa a proposta.

1.4. Organização

Este documento encontra-se organizado da seguinte forma: O capítulo 2 aborda a fundamentação. O capítulo 3 apresenta os trabalhos relacionados com a proposta. O capítulo 4 detalha a proposta. O capítulo 5 exibe os resultados. E, por fim, o capítulo 6 apresenta a conclusão do trabalho.

Capítulo 2

Fundamentação

Esse capítulo apresenta conceitos relacionados à gestão de identidade, controle de acesso tradicional, e controle de acesso baseado em papéis e atributos.

2.1. Gestão de Identidade (IdM, *Identity Management*)

Uma identidade representa uma entidade em um contexto particular (JOSANG e ZOMAI, 2007). Tipicamente, uma identidade é formada por um identificador com credenciais e atributos que representam características da entidade. Por exemplo, ao considerar que a entidade é uma pessoa, seu identificador pode ser seu CPF e seus atributos são informações pessoais relevantes.

A relevância das informações depende do contexto da aplicação, por exemplo, os atributos exigidos em uma transação bancária são diferentes dos atributos exigidos para ouvir música online. Assim, é necessário considerar a privacidade das informações. Um sistema de gestão de identidade (IdM, *Identity Management*) tem a função de controlar identidades através da autenticação, transportar de maneira segura os atributos de identidade, autorizar o acesso a recursos protegidos, delegar as identidades entre domínios de maneira segura (CAO e YANG, 2010).

Um IdM é formado por três componentes (CLAU e KESDOGAN, 2005):

- **Provedor de Serviço (SP, *Service Provider*):** provê serviços ao usuário, como: serviço bancário, e-commerce, rede social;
- **Provedor de Identidade (IdP, *Identity Provider*):** provê identidade ao usuário para utilização dos serviços (SP). É responsável pela autenticação do usuário e pelo processamento das requisições dos SP;

- **Usuário**: é um cliente do SP e IdP que necessariamente precisa de uma identidade para utilizar os serviços. Na prática, um usuário pode ser uma pessoa, organização, entidade virtual etc.

Dessa forma, a utilização de um IdM possui quatro objetivos dominantes (CLAU e KESDOGAN, 2005), são estes: (i) Fornecimento de Identidade – baseado em um único registro no IdP, diversos SPs podem criar diferentes identidades para o mesmo usuário; (ii) Autenticação única (SSO, *Single Sign-On*) – baseado em uma única autenticação em um SP, é possível acessar diversos SPs que partilham do mesmo IdP; (iii) Compartilhamento de atributos – os atributos de identidade especificados em determinado SP podem ser reutilizados em outros SPs; (iv) Autorização de acesso – Restringe o acesso de um SP a um recurso sem precisar acessar as credenciais.

Existem três modelos de IdM que determinam a relação entre SP e IdP, sendo eles o modelo tradicional, centralizado e federado. As subseções seguintes detalham cada modelo.

2.1.1. Modelo Tradicional

No modelo tradicional, chamado também de isolado, cada SP tem o papel de provedor de serviço e de identidade. Dessa forma, toda a gestão de identidade e usuários é gerida pelo mesmo servidor. As operações do IdM, como autenticação e autorização de acesso, são implementadas pelo SP. A Figura 2.1 ilustra o modelo tradicional.

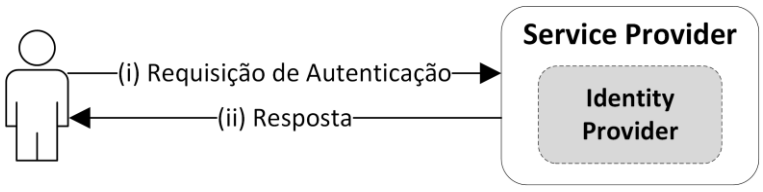

Figura 2.1. Modelo tradicional de IdM. Adaptado de (CAO e YANG, 2010).

Por ser um modelo elementar, traz consigo inúmeras desvantagens. O usuário necessita gerenciar várias identidades, uma para cada SP que precise utilizar. Além disso, todas as funcionalidades administrativas da gestão de usuários, como cadastro de usuários e recuperação de senha, devem ser implementadas em cada SP.

2.1.2. Modelo Centralizado

O modelo centralizado é baseado na arquitetura cliente/servidor. A implementação das funções de autenticação e a gestão de identidades são realizadas em servidores distintos. Dessa forma, o SP não armazena as credenciais do usuário localmente. Todas as identidades são enviadas para um servidor IdP centralizado, permitindo o armazenamento de todos os usuários, em um único domínio. A Figura 2.2 ilustra o modelo centralizado.

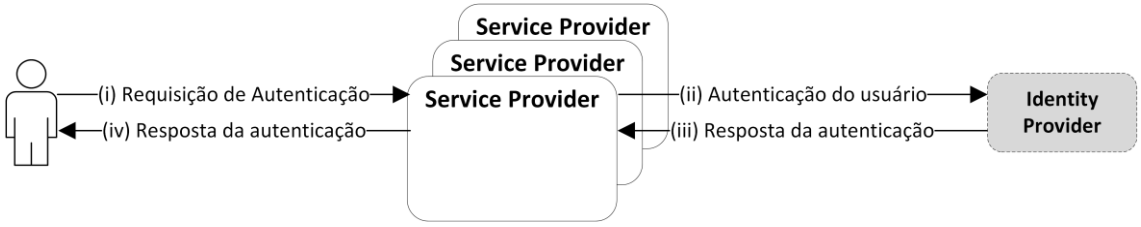

Figura 2.2. Modelo centralizado de IdM. Adaptado de (CAO e YANG, 2010).

O modelo centralizado é o mais popular atualmente por aplicações comerciais, porém possui algumas desvantagens por utilizar um único servidor. Esse modelo, em sua essência, não suporta um número elevado de usuários devido à complexidade da gestão de identidades para cada SP. Outro ponto negativo na disposição dos elementos dessa abordagem é tornar o provedor de identidades centralizado um ponto único de falhas.

2.1.3. Modelo Federado

O modelo federado integra diversos domínios, originando um domínio global virtualizado. Isso permite que diversos IdPs partilhem da identidade dos usuários, desde que exista uma relação de confiança pré-estabelecida entre os domínios. A Figura 2.3 ilustra o modelo federado.

Existe um mapeamento das identidades de um determinado usuário entre os diversos IdP. Dessa forma, quando um usuário está autenticado em um IdP, considera-se que ele está autenticado em todos, não necessitando uma segunda autenticação.

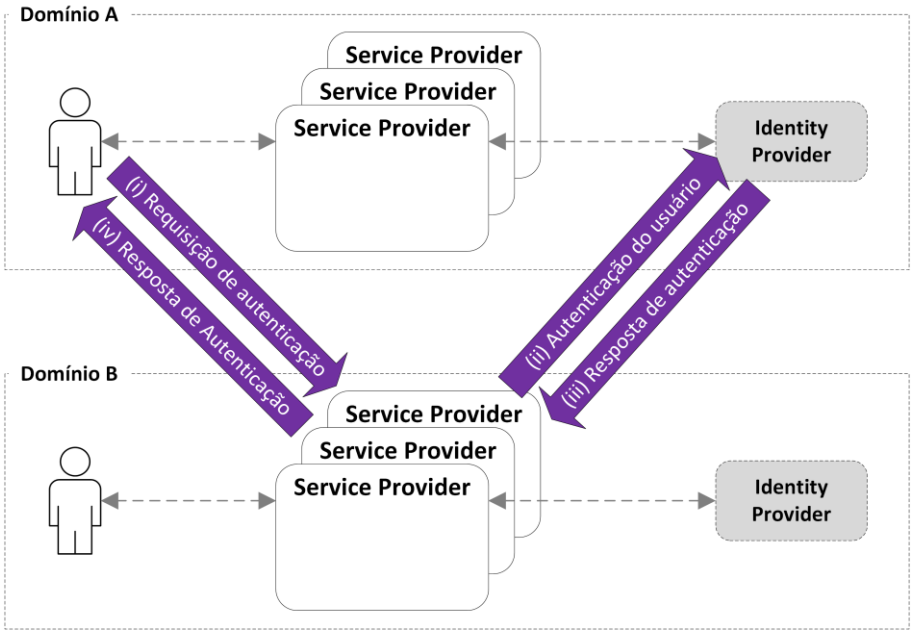

Figura 2.3. Modelo federado de IdM. Adaptado de (CAO e YANG, 2010).

2.1.4. Autorização de acesso

Um IdM pode utilizar o OAuth para realizar a autorização de acesso. OAuth é um *framework open source* para autorização de acesso para ambiente *web*. Seu objetivo é permitir que uma aplicação terceira confiável (RP, *Relying Party*) tenha acesso restrito a um serviço, sem a necessidade de compartilhamento de credenciais (HARDT, 2012). O OAuth utiliza um *token* de acesso que define parâmetros para restrição das ações que o usuário é capaz de realizar no domínio do *token* de acesso, por exemplo: escopo de acesso, tipo do *token*, tempo de expiração e *token* de *refresh* (referência um novo *token* quando o atual expirar).

A Figura 2.4 representa o fluxo do protocolo OAuth. Inicialmente a aplicação cliente (RP) solicita ao proprietário do recurso a autorização para acessá-lo *(evento i e ii)*. Em seguida, com posse da autorização emitida pelo proprietário do recurso, a aplicação cliente troca a autorização emitida, que normalmente é um código temporário (*ticket*), por um *token* de acesso *(evento iii e iv)*. Ao apresentar o *token* de acesso ao servidor (guardião) do recurso, é possível acessar o recurso, porém o *token* de acesso deve ser válido e possuir o escopo de acesso necessário *(evento v e vi)*.

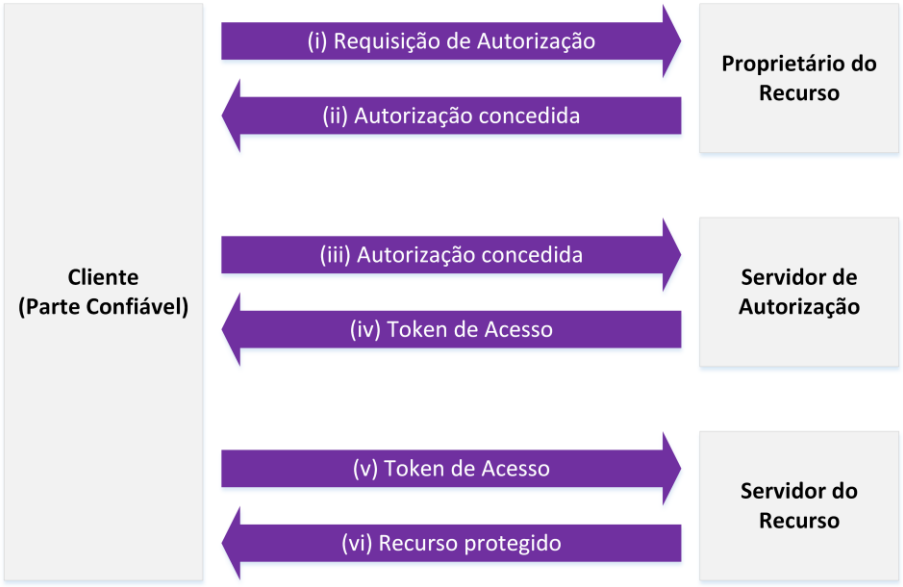

Figura 2.4. Fluxo do protocolo OAuth. Adaptado de (HARDT, 2012).

A autorização de acesso provida pelo OAuth limita o acesso a um recurso protegido, mas não suporta políticas que determinam as operações nos recursos. Dessa forma, não é possível obter um controle de granularidade fina, sendo necessário um controle de acesso adicional.

2.2. Controle de Acesso

Um controle de acesso é um mecanismo de segurança que permite limitar ações ou operações que determinado sujeito (humano ou máquina) pode realizar sobre um recurso (FERRAIOLO e KUHN, 2003). Um controle de acesso coexiste com outros serviços de segurança, como um serviço de autenticação. O controle de acesso é tipicamente utilizado após um usuário estar devidamente autenticado, ou seja, ele limitará o acesso a usuários legítimos. Entretanto, deve existir um serviço de autenticação responsável por validar a identidade de um sujeito.

A arquitetura de um controle de acesso geralmente é formada por um monitor de referências, que é responsável por intermediar as tentativas de acesso a um determinado recurso, consultando uma base de autorização que é mantida pelo administrador do sistema. Assim, o monitor de referências encaminha sua decisão ao guardião do recurso (*enforcement*), que executa a decisão permitindo ou negando o acesso do usuário ao recurso protegido.

Uma base de autorização, em sua forma mais primitiva, pode ser representada conceitualmente como uma matriz de acesso, onde cada linha da matriz é um sujeito (usuário) e cada coluna é um objeto (recurso). Cada célula dessa matriz representa a autorização que o usuário tem sobre o recurso. Por exemplo, se o recurso for um arquivo, normalmente as autorizações são para leitura, escrita e execução. O objetivo do controle de acesso é permitir que o usuário realize apenas as permissões contidas nessa matriz de acesso.

Para uma solução de segurança mais completa, é recomendada a utilização de um serviço de auditoria para registrar todas as requisições de acesso e atividades do sistema, permitindo realizar uma análise a *posteriori* de tentativas de invasão e possíveis violações no sistema. A Figura 2.5 ilustra um controle de acesso trabalhando em conjunto com outros serviços de segurança.

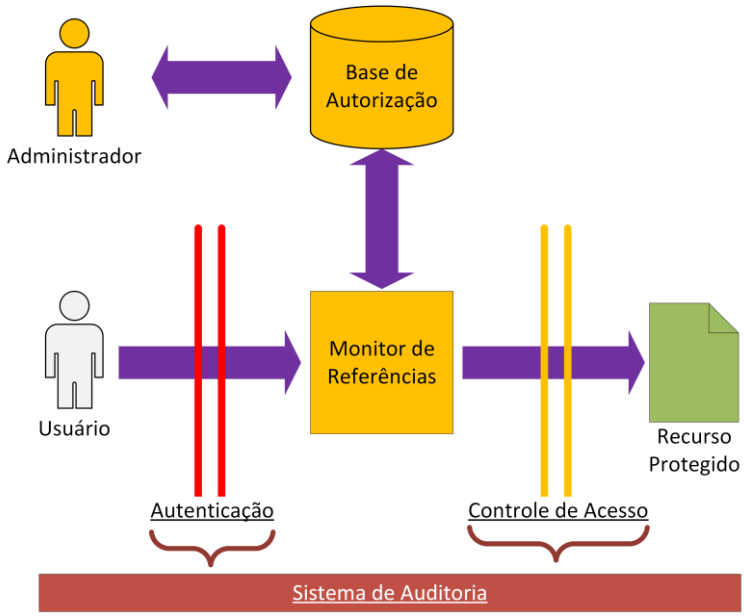

Figura 2.5. Controle de Acesso e outros serviços de segurança. Adaptado de (SANDHU e SAMARATI, 1994).

As políticas de um controle de acesso determinam o resultado da decisão de acesso. As políticas mais tradicionais são as discricionárias, mandatórias, baseadas em papéis ou em atributos. É importante ressaltar que as políticas não são exclusivas, ou seja, podem ser combinadas. A Figura 2.6 ilustra a combinação das políticas de controle de acesso. As subseções seguintes apresentam os modelos de controle de acesso mais populares.

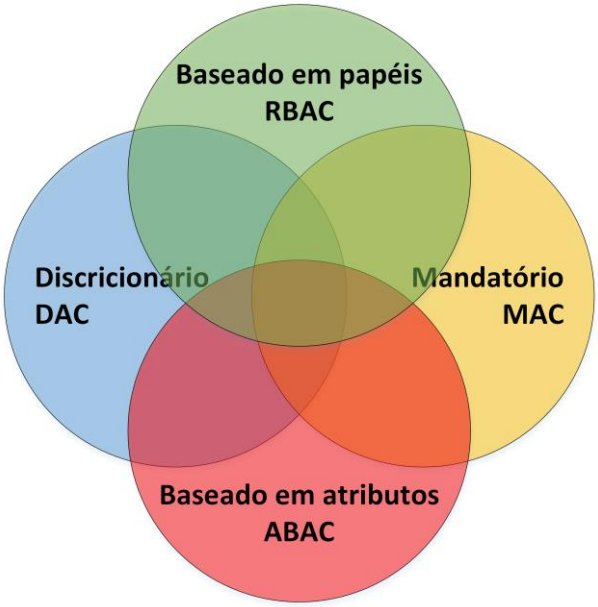

Figura 2.6. Combinação de políticas de controle de acesso. Adaptado de (SANDHU e SAMARATI, 1994)

2.2.1. Controle de Acesso Discricionário (DAC)

No Controle de Acesso Discricionário (DAC, *Discretionary Access Control*) cada requisição de acesso implica em uma consulta na base de autorizações (SANDHU e SAMARATI, 1994). Esse procedimento verifica a existência da permissão requisitada pelo usuário, o que permite a liberação da operação especificada sobre o recurso e a garantia do acesso. Caso contrário o acesso é negado.

Devido à facilidade de implementação e à alta flexibilidade e simplicidade do DAC, esse é utilizado em vários sistemas. Entretanto, a principal desvantagem desse modelo está relacionada à manutenção da base de autorização, principalmente quando é utilizado para um grande número de usuários e recursos. Por exemplo: se um usuário é excluído do sistema, todos os recursos que se relacionam com esse usuário devem remover a associação. Outra desvantagem do modelo está na ausência de controle do fluxo da informação. A partir do momento que um usuário autorizado tem acesso à leitura de um recurso, ele pode passar o conteúdo desse recurso a usuários não autorizados, sem que o proprietário do recurso tenha ciência.

2.2.2. Controle de Acesso Mandatório (MAC)

Motivado pela lacuna presente no fluxo de informações do DAC, foi desenvolvido o Controle de Acesso Mandatório (MAC, *Mandatory Access Control*). O MAC é baseado no modelo militar e define níveis de confidencialidade baseados em dois princípios básicos que controlam a escrita e a leitura das informações (SANDHU e SAMARATI, 1994). Devido a sua baixa flexibilidade, o modelo MAC é pouco empregado. Mais informações sobre o MAC podem ser encontradas em (LINDQVIST, 1987).

2.3. Controle de Acesso Baseado em Papéis (RBAC)

O Controle de Acesso Baseado em Papéis (RBAC, *Role-based Access Control*) utiliza papéis para intermediar a ligação entre usuários e permissões. O conceito de papéis vem sendo utilizado em sistemas há várias décadas, porém apenas nas últimas duas décadas se consolidou como um modelo de controle de acesso maduro (FERRAIOLO e KUHN, 1992). O início desse processo foi realizado por Ferraiolo e Kuhn (1992), quando desenvolveram as primeiras especificações técnicas e formais do RBAC. Três anos mais tarde, os mesmos autores criaram um modelo expandido de RBAC, que além de suportar hierarquia de papéis, permitiu o tratamento de conflitos de interesses. Em meados de 2000, o NIST, liderado por Ravi e Sandhu, propôs que o RBAC se tornasse um padrão, o que foi aceito apenas em 2004.

O modelo do RBAC é formado a partir dos seguintes componentes: principal, hierárquico e conflito de interesses. As subseções seguintes detalham cada componente.

2.3.1. Principal

A primeira parte do modelo do RBAC, denominada *Core* RBAC, define os elementos e princípios básicos do RBAC. A Figura 2.7 ilustra os elementos e seus relacionamentos.

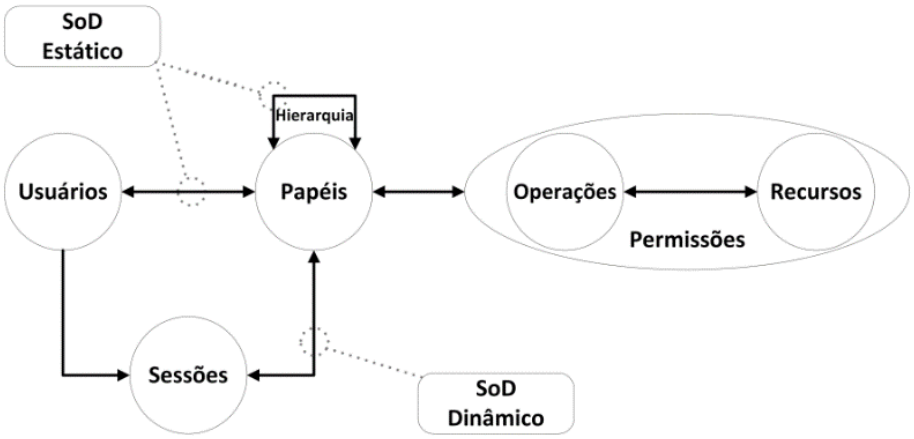

Figura 2.7. Modelo do RBAC. Adaptado de (FERRAIOLO e KUHN, 2003)

Os elementos do RBAC são definidos formalmente como (FERRAIOLO e KUHN, 2003):

- **Usuário**: pode ser definido como um ser humano, máquina, rede, agente autônomo inteligente, dentre outros;
- **Papel**: uma função de trabalho no contexto de uma organização, com uma semântica associada a uma autoridade e responsabilidade;
- **Sessão**: mapeamento entre um usuário e um subconjunto de papéis ativos;
- **Permissão**: aprovação de uma operação em um ou mais recursos protegidos;
- **Operação**: executa uma função para o usuário;
- **Recurso**: qualquer recurso/objeto do sistema.

- **SoD Estático/Dinâmica**: Separação de Deveres define restrições para associação e ativação de papéis, maiores informações são encontradas na seção 2.3.3.

Não existe uma relação direta entre o usuário e permissão, os usuários são associados aos papéis que são associados as permissões. Isso acarreta vantagens para a manutenção e também para o ciclo de vida do sistema. Caso um usuário seja desligado do sistema, basta desassociá-lo do papel em que está conectado.

O modelo adota o conceito de privilégio mínimo, em que o usuário não deve possuir privilégios superiores, mas apenas o necessário para realizar suas funções de trabalho. Isso minimiza o problema de um indivíduo ter a capacidade de realizar funções indesejadas. Logo, mesmo que um usuário possua diversos papéis, apenas os papéis realmente necessários estarão ativos no sistema (através da sessão).

2.3.2. Hierárquico

A segunda parte do modelo, denominada *Hierarchical* RBAC, define um meio natural para estruturação dos papéis com objetivo de refletir as linhas de autoridade e de responsabilidade de uma organização. Para que isso ocorra, existe uma ordem hierárquica entre os papéis, em que um papel sênior (pai) herda as permissões do papel júnior (filho) e o papel júnior herda os usuários do papel sênior. Exemplificando: um papel de diretor herda as permissões de um papel analista, e o papel analista herda os usuários do papel diretor.

As hierarquias estão divididas em dois tipos: Geral e Limitada. A hierarquia geral permite herança múltipla de papéis, onde um determinado papel pode ter mais que um papel sênior. Por outro lado, a herança limitada permite apenas uma herança simples, estruturando os papéis como uma árvore.

2.3.3. Conflito de Interesses

A terceira parte do modelo, denominada de *Constrained* RBAC, determina que uma operação crítica deve ser realizada por duas ou mais pessoas. Esse procedimento evita que uma pessoa individualmente comprometa o sistema. Para reduzir a possibilidade de conluio, indivíduos de diferentes habilidades ou interesses divergentes são atribuídos a tarefas individuais necessárias na execução de uma determinada função. A motivação é garantir que fraudes e erros graves não ocorram sem conluio deliberado de múltiplos usuários. Existem dois tipos de separação de deveres (SoD, *Separation of Duty*): estático e dinâmico.

A separação de deveres estática (SSoD, *Static Separation of Duty*) é implementada na associação de usuários com papéis e tem como objetivo limitar que um usuário detenha dois papéis conflitantes. Por exemplo, um sistema pode estabelecer que para acessar determinado cofre são necessários dois papéis: diretor administrativo e diretor financeiro. Nesse cenário, o usuário não pode ser associado ao papel de diretor administrativo e de diretor financeiro.

A separação de deveres dinâmica (DSoD, *Dynamic Separation of Duty*) tem o mesmo propósito da separação de deveres estática, porém é aplicada em local diferente. A separação de deveres dinâmica é aplicada na ativação do papel, ou seja, um usuário pode ter dois papéis possivelmente conflitantes desde que ele não ative ambos ao mesmo tempo. Por exemplo, um usuário pode ser autorizado a ter o papel de caixa e supervisor de caixa, sendo que o supervisor

pode efetuar correções feitas pelo caixa. Se um usuário desejar ativar o papel de supervisor de caixa, ele precisará desativar o papel de caixa para ativar o papel de supervisor. Essa propriedade da separação de deveres dinâmica reforça o princípio do privilégio mínimo, onde cada usuário possui diferentes níveis de permissão em momentos distintos, dependendo do papel ativo. Isso assegura que as permissões não persistam além do tempo necessário para cumprimento do trabalho.

2.4. Controle de Acesso Baseado em Atributos (ABAC)

O Controle de Acesso Baseado em Atributos (ABAC, *Attribute-Based Access Control*) é um modelo para controle de acesso que adota políticas que expressam regras *booleanas* utilizando atributos (FERRAIOLO e KUHN, 2014). O usuário possui um conjunto de atributos que são utilizados para avaliar sua autorização. O ABAC evita que as permissões estejam ligadas diretamente a um usuário ou a um papel. Logo, quando uma requisição de acesso é solicitada, um mecanismo realiza a decisão baseada nos atributos do usuário, recursos, ambientes, entre outros. É possível utilizar papéis no ABAC, considerando que estes são atributos do sujeito.

2.4.1. eXtensible Access Control Markup Language (XACML)

O XACML é um popular *framework* do ABAC, define uma linguagem baseada em XML para escrita de políticas de controle de acesso, requisições e respostas. Adicionalmente, provê um mecanismo de avaliação das políticas de controle de acesso (PARDUCCI e LOCKHART, 2013). Por padrão, o modelo de avaliação do XACML é baseado em ABAC. Entretanto, o XACML dispõe de um *profile* (extensão) para RBAC, onde os papéis são atributos do sujeito.

As entidades dominantes do XACML são: (i) PAP (*Policy Administration Point*), permite criar e armazenar as políticas de controle de acesso; (ii) PEP (*Policy Enforcement Point*), responsável por encaminhar as requisições de acesso dos usuários ao *Context Handler*, atuando como guardião dos recursos; (iii) PIP (*Policy Information Point*), atua como a fonte de atributos para o *Context Handler*; (iv) PDP (*Policy Decision Point*), avalia as políticas e produz a decisão de acesso, classificada como permitido ou negado; e (v) *Context Handler*, responsável

por realizar a coordenação, adequação e interoperabilidade de atributos, requisições e credenciais entre as entidades do XACML.

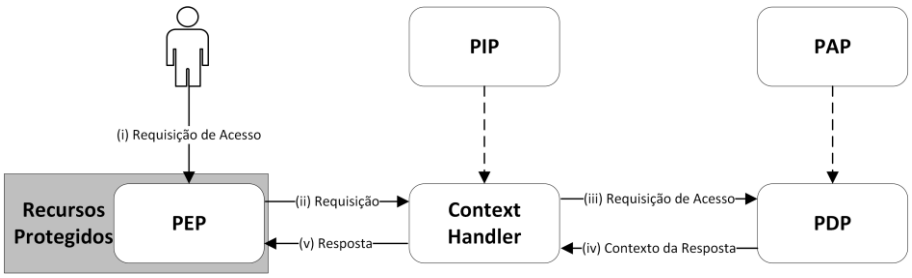

Figura 2.8. Arquitetura simplificada das entidades XACML. Adaptado de (PARDUCCI e LOCKHART, 2013).

A Figura 2.8 apresenta uma visão geral do protocolo XACML. Nela, um usuário requisita acesso a um recurso protegido *(evento i)*. O PEP intercepta a requisição de acesso e a encaminha para o *Context Handler (evento ii).* O *Context Handler* gerencia as informações (atributos) do usuário providas pelo PIP para criar uma requisição no formato XACML para o PDP *(evento iii).* O PDP requisita ao PAP as políticas relacionadas aos recursos e realiza a decisão sobre a requisição de acesso. O PDP retorna a decisão (permitido/negado) utilizando o formato XACML para o *Context Handler (evento iv).* O *Context Handler* informa o PEP a decisão *(evento v).* O PEP permite ou rejeita o acesso do usuário de acordo com a decisão do PDP.

2.5. Considerações

Ainda que o modelo federado de gestão de identidade detenha uma quantidade significativa de benefícios em relação ao modelo centralizado, é usual o modelo centralizado ser adotado em decorrência da sua simples implantação. É permissível a transformação do modelo centralizado para ser distribuído, para isso escalona-se o modelo com o aumento do número de IdPs.

É importante ressaltar que na literatura, comumente os autores consideram que o OAuth é uma espécie de controle de acesso. Porém, é recomendado considerá-lo como um controle de autorização de acesso, operando como uma espécie de controle de admissão. Pois o OAuth apenas limita o acesso a um recurso protegido, baseado no escopo de acesso contido no *token* de acesso. A partir do momento que o usuário tem acesso ao recurso protegido, o

OAuth não é capaz de limitar as operações que o usuário pode ou não realizar sobre o recurso protegido. Ou seja, existe uma distinção de granularidade.

Controle de acesso é um mecanismo de segurança essencial para o sistema. Quando o sistema é uma aplicação organizacional, a política de segurança mais adequada é baseada em papéis ou atributos. Os papéis representam uma função de trabalho no contexto de uma organização. É possível utilizar papéis no controle de acesso baseado em atributos, considerando que os papéis são atributos de um usuário. Isso possibilita a realização de um controle de acesso que realiza a avaliação dos papéis do usuário em tempo real.

Capítulo 3

Trabalhos Relacionados

Esse capítulo discorre sobre os trabalhos relacionados à pesquisa encontrados na literatura.

3.1. Autorização em Ambientes Multidomínios

Realizar autorização em ambientes multidomínios é um desafio amplamente discutido na literatura. O trabalho de Lee e Luedemann (2007) analisou uma série de trabalhos que abordam autorização multidomínios de maneira geral. Os autores identificaram algumas características que são discutidas a seguir.

- Autonomia: Os domínios são heterogêneos em termos de infraestrutura de TI, sistemas e políticas. Dessa forma, todos os domínios almejam manter sua autonomia para definir o acesso a seus recursos protegidos.
- Privacidade: As informações privadas de um usuário não podem em hipótese alguma ser acessadas por pessoas não autorizadas. Em inúmeros casos, apenas os proprietários de um determinado recurso devem ter o controle sobre o mesmo.
- Representatividade: Na maioria dos trabalhos existe de alguma forma, seja através de um usuário ou um papel global, um ponto de contato para realizar a gestão dos acessos multidomínios.
- Descentralização: É necessário um mecanismo decentralizado para gerir as políticas de segurança.
- Escalabilidade: O sistema de autorização necessita ser escalável e de simples instalação/configuração. Isso se justifica porque o objetivo é facilitar o procedimento de adesão no caso de novas organizações participantes.

- Padronização: O emprego de padrões amplia a compatibilidade e encoraja a interoperabilidade entre variados domínios.
- Regulamentação: Todos os casos estudados dispõem de algum tipo de lei ou regras conhecidas por todos os domínios participantes.

3.2. RBAC em Ambientes Multidomínios

Diversos trabalhos objetivaram resolver o desafio de utilizar o RBAC em multidomínios. Shafiq et al. (2005) propôs um algoritmo que centraliza os papéis e os mapeia globalmente através de uma operação de junção *(merging)* de papéis. Essa operação verifica em cada papel se existe a intersecção de permissões com algum papel de outro domínio.

A solução proposta por Shafiq et al. (2005) permitiu interoperar papéis entre domínios. Dessa forma, tornou viável realizar hierarquia de papéis entre domínios. A hierarquia de papéis pode causar inconsistências no modelo RBAC, como uma herança cíclica de papéis. A herança cíclica ocorre quando um papel sênior se torna filho de um papel júnior. Para resolver isso, os autores desenvolveram um método de verificação de conflitos.

Essa operação de junção centralizada dos papéis exige um custo computacional elevado, uma vez que é essencial comparar e mapear todos os papéis entre si. Além disso, num caso de atualização de papéis, é necessário na maioria das vezes, refazer todas as junções. A solução necessita de uma área de compartilhamento dos recursos, para que esses sejam visíveis a todos os domínios. Por fim, existe um problema ligado à semântica das permissões, pois as ações que uma permissão pode executar em um determinado domínio podem ser diferentes em outro domínio, ainda que tenham o mesmo nome.

Na proposta de Qi Li et al. (2006), foi adotado o conceito de virtualização de papéis sob demanda. O administrador RBAC define os papéis que deseja utilizar em multidomínio e então é criado um *link* de referência desses papéis em um domínio global. Essa abordagem é uma evolução do trabalho de Shafiq et al. (2005) e traz como principal vantagem a escolha de papéis sob demanda, sem a necessidade de incluir todos os papéis no domínio global. Entretanto, ainda se mantêm como problemas a centralização dos papéis em um domínio global, a necessidade de compartilhamento de recursos e a semântica das permissões.

O trabalho de H. K. Lee (2010) propôs uma administração autônoma e decentralizada para autorizações multidomínios. O autor se baseou no modelo RBAC e criou um *framework* que permite operações multidomínios de maneira semiautomática. Para isso, foi desenvolvida uma camada intermediária que armazena os papéis interativos (*iRoles*). Esses papéis não possuem permissões e usuários relacionados inicialmente. O modelo *iRole* é semelhante ao modelo tradicional do RBAC. Quando uma operação multidomínio se tornar necessária, um administrador do domínio em questão deve inspecionar e confirmar se determinado candidato (usuário) poderá utilizar determinado papel (*iRole*).

O trabalho de Freudenthal et al. (2002) adotou um repositório de credenciais denominado *wallet* que armazena as delegações de autorizações (através de papéis). A arquitetura utiliza-se de um monitor de provas responsável por avaliar as delegações. Cada domínio possui uma *wallet* e na medida do possível todas as *wallets* encontram-se sincronizadas entre si através de um serviço *publish*/*subscribe*. No caso de uma delegação não existir na *wallet* local, é aplicado um serviço de descoberta de localização para a *wallet* de origem do recurso envolvido na delegação, e se a delegação for encontrada, será inserida na *wallet* local para fazer *cache*. Isso pode tornar o processo de busca intenso e lento, pois um conjunto de delegações necessárias para autorizar um papel pode encontrar-se em várias *wallets*.

O trabalho de Avita et al. (2012) apresentou uma proposta de implementação do modelo RBAC com a utilização de ontologias. A complexidade da implementação do RBAC está relacionada na forma de representar os papéis. Os autores representam cada papel como uma classe OWL (*Ontology Web Language*). Dessa forma, existe uma classe principal denominada "Papel" que dispõe das características e atributos pertencentes a todos os papéis, e os demais papéis são subclasses da classe principal. Assim, é possível representar hierarquias de papéis através das heranças das classes. Os autores propõem uma delegação de papéis, onde um usuário que possui um determinando papel tem a possibilidade de delegar o papel, juntamente com as permissões relacionadas, a um outro usuário. Esse comportamento é um ponto fraco do trabalho, pois o administrador do sistema perde a autonomia da gestão de papéis. A partir do momento em que um usuário recebe a delegação de um papel, ele recebe a permissão para delegar suas permissões a outros usuários. Isso prejudica a visão gerencial do administrador sobre os papéis, que é uma das principais vantagens do RBAC.

O trabalho de Mouliswaran (2015) apresenta um modelo que utiliza análise de conceito formal (FCA, *Formal Concept Analysis*) para representar as permissões de acesso que um papel possui em diferentes domínios. Dessa forma, o autor considera que existem papéis globais

(interdomínios) que possuem permissões locais em cada um dos domínios. Essas permissões são armazenadas em uma matriz, onde os papéis globais são as linhas e as permissões são as colunas. A solução não é escalável, visto que cada operação sobre determinado recurso é uma coluna da matriz, se considerar que uma organização possui diversos recursos e operações, torna-se inviável sua aplicação.

3.3. XACML contemplando o RBAC

Alguns trabalhos buscaram resolver problemas existentes no *profile* do RBAC (ANDERSON, 2004) para contemplar todas as características no XACML. O trabalho de Lee e Luedemann (2007) é uma evolução do trabalho de Freudenthal et al. (2002), e utilizou o conceito de papéis distribuídos integrados ao XACML. Na arquitetura proposta, existem papéis e recursos externos que são visíveis a outros domínios. A delegação de papéis funciona de maneira semelhante à de Freudenthal et al. (2002). Assim, os autores desenvolveram uma área compartilhada para armazenar papéis e recursos externos. Os autores estenderam o XACML para consultar essa área compartilhada. Os problemas apontados na proposta de Freudenthal et al. (2002) persistem no trabalho de Lee e Luedemann (2007), além de adicionar o problema de alterar a arquitetura do XACML.

O trabalho de Helil et al. (2010) teve como objetivo estender o *profile* para suportar tanto a separação de deveres estática quanto a dinâmica. Para isso, foi desenvolvida uma biblioteca de avaliação de XACML baseada na *sun-xacml*[1], acrescentada de novas entidades responsáveis pela gestão dos papéis ativos e separações de deveres. Essa proposta tem a limitação de necessitar a alteração da arquitetura XACML e criar uma biblioteca própria.

O trabalho de Ferrini et al. (2009) propôs um *framework* que realiza a integração de ontologias com políticas XACML para suportar o RBAC. A proposta foi desacoplar os conflitos e hierarquias que não são tratados no *profile* do RBAC da arquitetura XACML. Para isso, foi utilizado a linguagem OWL (*Ontology Web Language*), que expressa políticas que definem hierarquias e conflitos. Dessa maneira, a linguagem e o mecanismo do XACML igualmente foram estendidas para dar suporte ao OWL. O aspecto negativo da proposta se dá na alteração

[1] http://sunxacml.sourceforge.net/

do fluxo tradicional de funcionamento e da sintaxe da linguagem do XACML, tornando-a sem padronização.

3.4. Considerações

A Tabela 3.1 apresenta uma comparação entre os diversos trabalhos relacionados. Nenhum trabalho relacionado aborda o desafio de ativação de papéis multidomínios, o que representa uma lacuna na literatura.

Trabalho	Papéis multidomínios	Separação de Deveres	Hierarquia	Implementação	Desvantagens
H. K. Lee (2010)	Framework permite operações multidomínios de maneira semiautomática, utilizando papéis interativos.	Não	Sim	Não, formalização matemática.	Procedimento semiautomático, que necessita de interação humana.
Shafiq et al. (2005)	Centralização dos papéis via operação de junção (*merge*) de papéis.	Sim	Sim	Não, formalização matemática.	Centralização de papéis; Compartilhamento de recursos; Semântica das permissões.
Qi Li (2006)	Virtualização de papéis centralizada sob demanda.	Sim	Sim	Não, formalização matemática.	Centralização de papéis; Compartilhamento de recursos; Semântica das permissões.
Freudenthal et al. (2002)	Delegação de papéis através de um serviço de *publish/subscribe*.	Não	Sim	Sim, porém não disponibilizada.	Delegação de autorização; Serviço de descoberta de

					permissões pode ser profundo.
K.Avita et al. (2012)	Representação dos papéis através de ontologias (OWL).	Não	Sim	Sim, porém não disponibilizada.	Delegação de autorização;
S. Chandra Mouliswaran (2015)	Modelo utilizando FCA para representar as permissões que um papel possui em diferentes domínios.	Não	Não	Não, formalização matemática.	Solução não escalável.

Tabela 3.1. Considerações sobre os trabalhos relacionados

Capítulo 4

Proposta

Este capítulo apresenta inicialmente o modelo proposto de controle de acesso baseado em atributos utilizando papéis, seguido de sua especificação. Posteriormente, descreve o mecanismo para importar a ativação de papéis de domínios remotos e a sintaxe para escrita de políticas que utilizam papéis ativos locais e remotos.

4.1. Modelo

Para desenvolver um modelo de ativação multidomínios de papéis utilizando um controle de acesso baseado em atributos, foi necessário criar um modelo capaz de suportar a ativação única de papéis (SRA, *Single Role Activation*). Dessa forma, é necessário que o modelo possibilite a existência e integração de um sistema de gestão de identidades (IdM), um controlador de papéis e um controle de acesso de granularidade fina. O modelo proposto é formado pelos seguintes elementos: Controle de Autenticação, Controle de Autorização de Acesso, Controle de Acesso baseado em papéis e o Controle de Acesso baseado em atributos. As subseções seguintes detalham cada elemento. A Figura 4.1 ilustra o modelo, a integração dos componentes é discutida na seção 4.1.5.

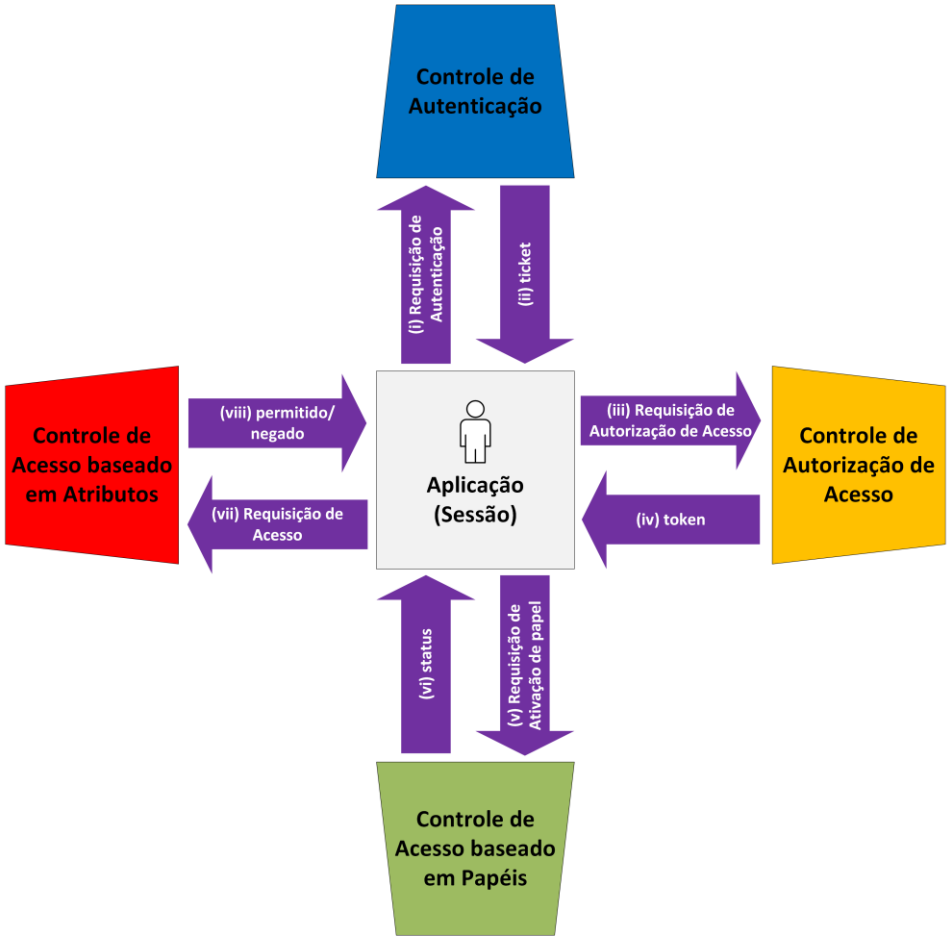

Figura 4.1. Modelo de controle de acesso proposto

4.1.1. Controle de Autenticação

O Controle de Autenticação é responsável pela autenticação e identidade do usuário. Quando um usuário é autenticado com sucesso, o controle de autenticação fornece um *ticket* de curta duração que comprovará a autenticidade do usuário junto ao Controle de Autorização de Acesso. Esse *ticket* de curta duração torna-se a garantia de que o usuário foi autenticado no Controle de Autenticação.

Adicionalmente, o Controle de Autenticação é responsável pela gestão dos usuários na arquitetura, atua como um IdP. A fim de simplificar a proposta, foi considerado o modelo centralizado de IdM, visto que o foco do trabalho está voltado à autorização. Entretanto, é possível utilizar o modelo federado de gestão de identidades sem mudança nos demais elementos da arquitetura.

4.1.2. Controle de Autorização de Acesso

O Controle de Autorização de Acesso é responsável pela emissão de *tokens* que autorizam o acesso do usuário a serviços protegidos. Dessa forma, funciona como um controle de admissão aos demais serviços (SP). Para cada serviço que o usuário necessite acessar, o controle de autorização de acesso emite um *token*, que possui um escopo de acesso.

O *token* é uma credencial para acesso a serviços protegidos. Esse é uma *string* que representa uma autorização emitida pelo Controle de Autorização de Acesso. Possui um escopo de acesso, que permite realizar a restrição de visibilidade de um recurso e um tempo de vida (duração). Quando expirado (tempo de vida esgotado), esse não é mais válido e deve ser realizado a emissão de um novo *token*.

Para cada serviço são necessários dois escopos diferentes contidos no *token*, um que permite ao usuário realizar apenas a leitura e outro permite leitura e atualização de um recurso. Essa restrição só permite que usuários autenticados proprietários de *tokens* de acesso válidos sejam admitidos no sistema. Para ativar um papel no Controle de Acesso Baseado em Papéis, o usuário necessita possuir um *token* de acesso com o escopo "*rbac_origem_irrestrito*". Para consultar seus papéis ativos, o usuário necessita possuir um *token* com o escopo "*rbac_origem_leitura*" ou "*rbac_origem_irrestrito*".

O Controle de Autorização de Acesso possui essa nomenclatura devido à popularidade do termo "Controle de Autorização" na literatura. Entretanto, ele não é um controle de autorização, porque um controle de autorização possui uma estrutura mais complexa, conforme descrito na fundamentação do trabalho, na seção 2.1.4. Este trabalho optou por utilizar o termo "Controle de Autorização de Acesso" para ser condizente com a literatura e com sua aplicação na proposta aqui tratada.

4.1.3. Controle de Acesso Baseado em Papéis (RBAC)

O controle de acesso baseado em papéis (RBAC) é responsável pela gestão dos papéis. A gestão de papéis consiste em criar, relacionar papéis com os usuários e ativá-los. Além disso, é possível definir as hierarquias de papéis e conflitos de interesse entre papéis, atuando como um provedor de serviços (SP).

Para criar o conflito de interesses, o administrador do sistema seleciona dois papéis e classifica-os como conflito de interesse estático ou dinâmico. Quando o administrador associa um papel a um usuário, o controle de acesso baseado em papéis verifica se existe algum conflito de interesses estático (SSoD). Quando o usuário ativa um papel, inicia-se o processo de verificação de um possível conflito de interesses dinâmico (DSoD) e, caso isso ocorra, o papel não é ativado.

4.1.4. Controle de Acesso Baseado em Atributos (ABAC)

O Controle de Acesso baseado em Atributos (ABAC) é responsável pelo controle de acesso com granularidade fina e utiliza políticas usando atributos do usuário. Dessa forma, torna-se o guardião dos recursos protegidos, atuando como um SP.

ABAC é responsável por deliberar as permissões que um papel terá sobre determinado recurso. Para isso, o administrador do sistema define políticas de segurança vinculando ações, recursos e papéis que estão armazenados no RBAC.

4.1.5. Integração

A integração dos componentes é ilustrada na Figura 4.1. A arquitetura do modelo de controle de acesso foi projetada para que o RBAC e o ABAC atuem como serviços, sendo que as funcionalidades disponíveis nos serviços exigem um *token*, cuja validação no Controle de Autorização de Acesso é online.

O usuário requisita a autenticação para a Aplicação (App). A App abre uma sessão para o usuário e solicita a autenticação para o Controle de Autenticação, informando a sessão do usuário (Figura 4.1, *evento i*). O usuário informa suas credenciais, caso sejam válidas, o Controle de Autenticação fornece um *ticket* para a App (*evento ii*). A App solicita o *token* para o Controle de Autorização de Acesso fornecendo o *ticket* (*evento iii*). O Controle de Autorização de Acesso fornece um *token* (*evento iv*) que permite acessar o RBAC para requisitar e ativar papéis (*evento v e vi*, respectivamente) e o ABAC para requisitar acesso aos recursos protegidos (*evento vii e viii*, respectivamente).

A Figura 4.2 apresenta o diagrama de sequência para a ativação de um papel. O usuário solicita à App a ativação de um papel que possua acesso (*evento 1*). A App repassa o pedido ao

RBAC (*evento 1.1*), fornecendo o papel em questão (*papelSelecionado*) e o *token* de acesso (*tokenAcesso*). O RBAC solicita a validação do *token* no Controle de Autorização de Acesso (*evento 1.1.1*). Se o *token* de acesso se encontrar válido, o RBAC extrai desse o usuário vinculado (*evento 1.1.2*). Finalmente, o RBAC verifica a existência de algum conflito de interesses (DSoD, *evento 1.1.3*) na ativação do papel, caso não exista é retornado o status de sucesso.

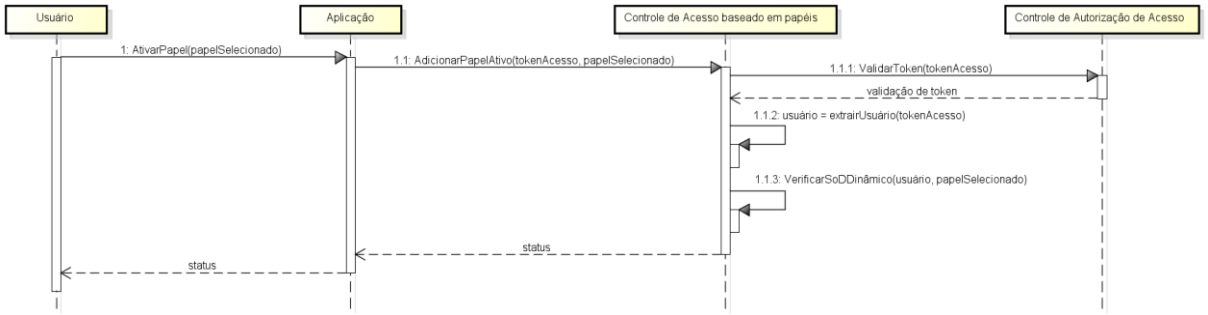

Figura 4.2. Diagrama de sequência para ativação de papéis

A Figura 4.3 ilustra um diagrama de sequência que retrata o procedimento de requisição de acesso a um recurso protegido. O usuário solicita à App a requisição de acesso a um recurso protegido, informando a ação *(evento 1).* A App, que porta o *token* de acesso do usuário, repassa o pedido ao ABAC, informando o *token* de acesso, recurso e ação (evento 1.1). O ABAC realiza a validação do *token* no Controle de Autorização de Acesso (evento 1.1.1). Caso seja válido, o Controle de Acesso baseado em atributos solicita os papéis ativos do usuário, informando o *token* de acesso (evento 1.1.2). Após possuir os papéis ativos do usuário, é avaliado se o usuário possui ou não acesso ao recurso protegido baseado nas políticas de segurança (evento 1.1.3).

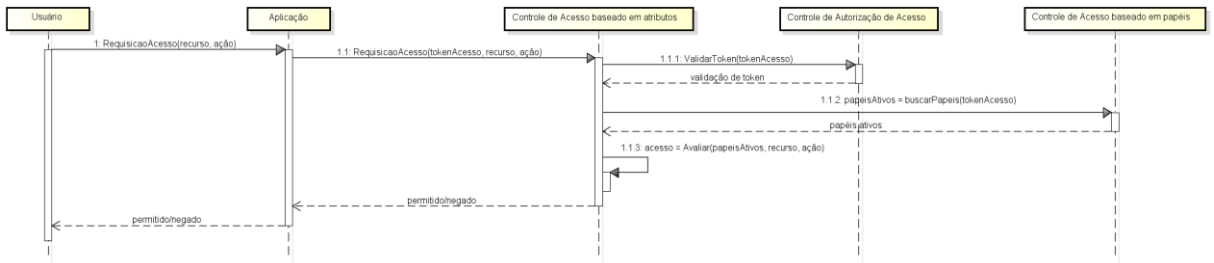

Figura 4.3. Diagrama de sequência para requisição de acesso

4.2. Especificação do Modelo

A especificação tem o propósito de utilizar especificações e padrões que tornam o modelo proposto viável. Dessa forma, para cada componente presente no modelo (Figura 4.1) foi utilizada uma especificação.

Para especificar o Controle de Autenticação foi utilizado o OpenID Connect (OIDC), que é um protocolo de autenticação e identidade (IdM) desenvolvido sobre o OAuth 2.0. Dessa forma, é uma extensão que permite ao provedor de serviço (SP/RP) a verificação da identidade de cada usuário (SAKIMURA, 2014). O OIDC dispõe um *token* de identidade (*ID Token*) que contém informações sobre a autenticação e atributos do usuário. Por se tratar de uma extensão do OAuth 2.0, além do *token* de identidade, o OIDC também fornece o *token* de acesso.

O processo de obtenção dos *tokens* através do OIDC é ilustrado na Figura 4.4. Inicialmente a RP solicita a autenticação do usuário no OIDC (*evento i*). O usuário fornece suas credenciais *(evento ii)*, caso sejam válidas, o mesmo será questionado se autoriza ou não que a RP acesse a suas informações definidas no escopo do *token* (*evento iii*). O OIDC responde ao RP sobre o processo de autenticação e consentimento de acesso do usuário (*evento iv*). Finalmente, o RP pode obter do OIDC o *token* de acesso e o *token* de identidade do usuário (*eventos v e vi*, respectivamente).

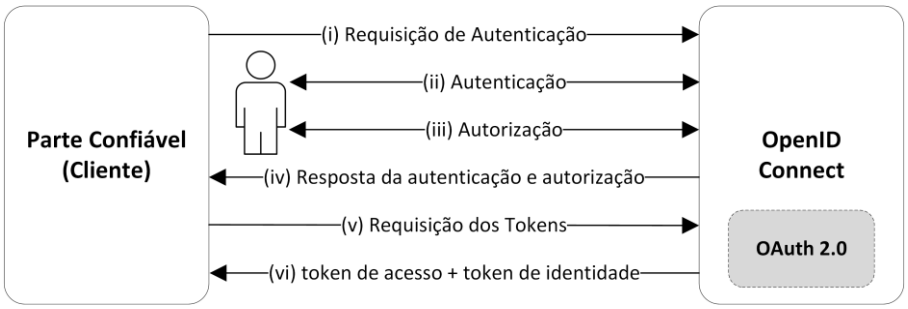

Figura 4.4. Visão geral das trocas do protocolo OIDC

Dessa forma, foi utilizado o *token* de identidade para armazenar informações do usuário tradicional e também informações sobre o domínio do usuário. O Controle de Autorização de Acesso utiliza a especificação do OAuth 2.0. Por se tratar de uma extensão do OAuth 2.0, o OIDC tem o papel de Controle de Autenticação e Controle de Autorização de Acesso na especificação da proposta.

Para o RBAC, foram utilizadas as especificações definidas no padrão de (FERRAIOLO e KUHN, 2003), conforme já mencionado na fundamentação deste trabalho, seção 2.3. Finalmente, para o ABAC, utilizou-se a especificação do XACML versão 3.0.

A integração dos elementos é representada pela Figura 4.5. A App requisita a autenticação do usuário no OIDC, informando suas credenciais (*evento i*). O OIDC valida as credenciais e retorna um *ticket* de curta duração para a App (*evento ii*). O *ticket* de curta duração é utilizado pela App para obter o *token* de acesso (*tokenAcesso*) e o *token* de identidade (*idToken*), representado no evento iii e iv, respectivamente. Portando o *token* de acesso, a App requisita ao RBAC os papéis vinculados ao usuário (*evento v*). O RBAC realiza a validação online do *token* de acesso, retornando os papéis disponíveis para este usuário que poderá escolher qual papel pretende ativar (*evento vi*). A ativação de papel é realizada ao enviar ao RBAC o *token* de acesso e o papel que deseja ativar (*evento vii*). O sucesso da ativação depende dos conflitos de interesses dinâmicos de papéis (DSoD) impostos pelo administrador do RBAC (*evento viii*).

Em nome do usuário, a App requisita acesso ao XACML, informando a ação que deseja realizar sobre determinado recurso, juntamente com o *token* de acesso (*evento ix*). Essa requisição chegará ao PEP, que encaminha a requisição ao *Context Handler* (CH). O CH constrói uma requisição XACML que é enviada ao PDP para avaliação da política, e esse retorna o estado do acesso (permitido ou negado). Para construir a requisição, o CH solicita ao PIP a coleta dos papéis ativos para o usuário corrente. A obtenção dos papéis ativos ocorre via PIP, que solicita ao serviço RBAC, informando o *token* de acesso (*evento a*). Após receber os papéis ativos para o usuário em questão (*evento b*), o CH encaminha os atributos para o PDP avaliar se o usuário tem permissão para realizar a ação sobre o recurso. O CH recebe a decisão do PDP e encaminha ao PEP, que honra a decisão do PDP, liberando ou bloqueando o acesso do usuário ao recurso (*evento x*).

Caso o usuário venha a requisitar um segundo acesso a um recurso protegido, não será necessário executar todas as etapas, pois o token de acesso e a ativação do papel mantém o estado. Apenas as etapas da requisição de acesso (evento ix e x) são necessárias.

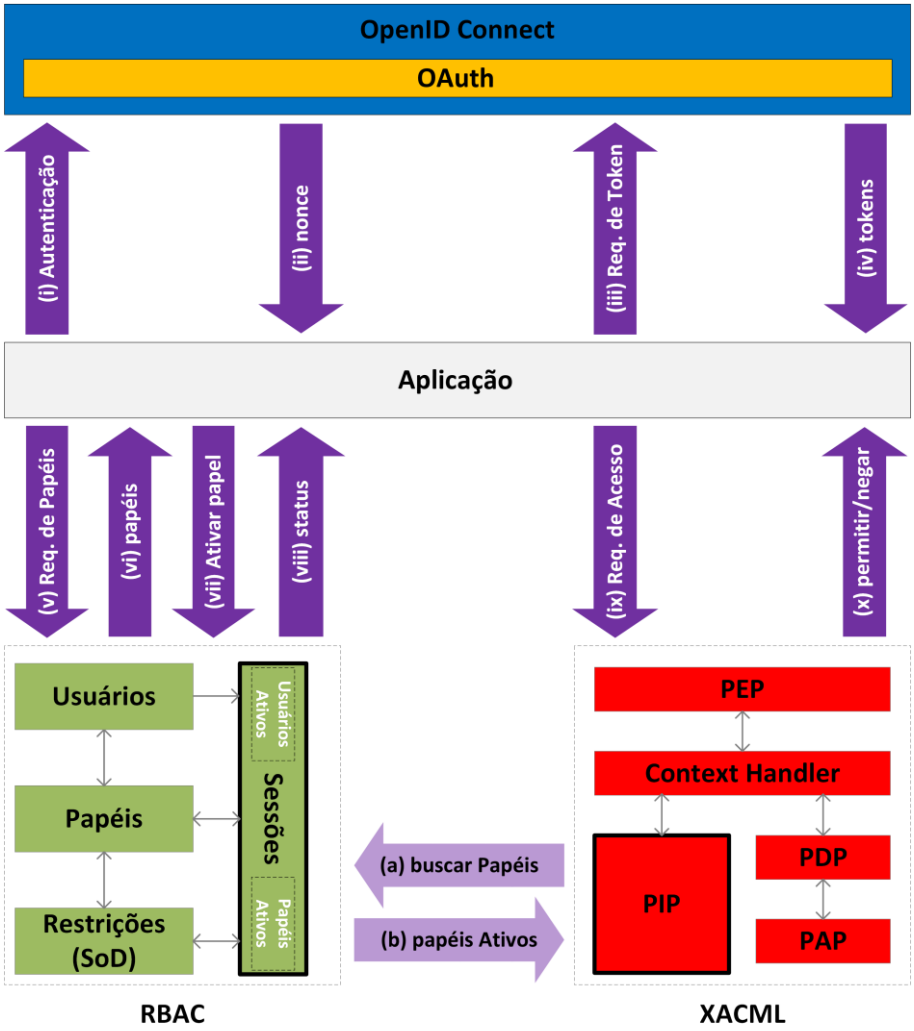

Figura 4.5. Visão detalhada do modelo proposto

Dessa maneira, as permissões dos papéis ficam armazenadas no PAP. Na prática, a cada requisição de acesso o Context Handler é responsável por realizar o *Permission Assignment* (PA) do papel. O PA é um método da especificação administrativa do RBAC (FERRAIOLO e KUHN, 2003), que determina a associação dos papéis com as permissões.

4.3. Mecanismo de ativação de papéis multidomínios

O mecanismo objetiva manter a autonomia do administrador para cada domínio no sentido de permitir o gerenciamento dos direitos associados a cada papel, e de forma adicional, simplificar a vida do usuário de forma a evitar a ativação de papéis para cada domínio que o usuário visite. Assim, é proposto um mecanismo que permite importar a ativação de um papel feita no domínio de origem do usuário para domínios remotos que o usuário pretenda visitar.

Para tornar isso viável, é necessário existir uma relação de confiança entre os domínios que não é tratada neste estudo.

Este trabalho considerou que um usuário pertence a um domínio, denominado domínio de origem, que pode acessar domínios distintos, denominados domínios remotos. Por exemplo, um médico é funcionário de uma determinada clínica médica (domínio de origem), entretanto ele pode acessar a algumas informações do domínio de um hospital (domínio remoto).

A importação da ativação, denominada *Single Role Activation* (SRA), significa que se o usuário tem um papel ativo em seu domínio de origem, poderá reutilizar a ativação para um domínio remoto. Assim, não será necessário executar as etapas de ativação do papel no domínio remoto. Entretanto, os direitos que o papel terá no domínio remoto são definidos pelo administrador do domínio remoto.

A Figura 4.6 ilustra o funcionamento do SRA. Por ser baseado em SSO, o SRA necessita que o usuário já se encontre autenticado no OIDC e esteja portando o *token* de acesso, obtido em seu domínio de origem (domínio A). Em seu domínio de origem, o usuário pode ativar os papéis que lhe pertencem, conforme a Figura 4.2. Ao considerar que o usuário possui papéis ativos em seu domínio de origem e deseja acessar outras aplicações de forma transparente através do SSO, o usuário requisita acesso a um recurso em um domínio remoto (domínio B).

Em nome do usuário, a App do domínio remoto requisita acesso ao XACML, informando a ação que pretende realizar sobre determinado recurso, juntamente com o *token* de acesso (evento ii). O escopo do *token* de acesso contém a restrição de visibilidade para que o usuário acesse o domínio remoto em modo leitura. O PEP recebe a requisição e repassa para o CH, que criará uma requisição XACML para o PDP que busca a política vinculada ao recurso no PAP, e informa ao CH a necessidade de buscar papéis ativos para o usuário em outro domínio. Assim, o CH invoca o PIP requisitando os papéis ativos do usuário no domínio de origem. Para isso, o PIP utiliza o *token* de acesso do usuário para requisitar ao OIDC o *token* de identidade do usuário, onde está uma *claim* (atributo) que identifica o domínio de origem do usuário. Através do conhecimento do domínio de origem do usuário, o PIP requisita papéis ativos do usuário no RBAC do domínio de origem (evento iii).

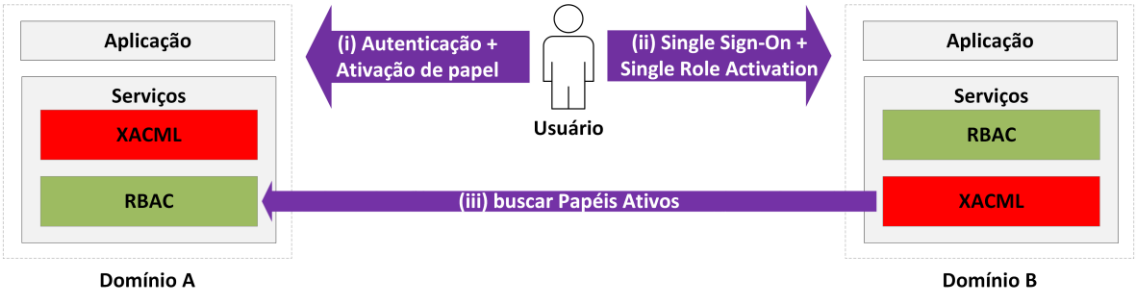

Figura 4.6. Visão detalhada do modelo multidomínios

É importante ressaltar que os papéis ativos no domínio de origem do usuário (que posteriormente são importados para um domínio remoto via SRA) têm direitos locais e esses podem conflitar com papéis locais ativos. Dessa forma, não é sempre que um determinado papel poderá ser ativado via SRA. No momento em que a ativação se faz possível (não havendo conflito dinâmico, DSoD), o PIP retorna ao CH todos os papéis ativos do usuário no domínio local, ressaltando que essa lista de papéis pode conter papéis locais e remotos (ativos pelo SRA). Por fim, o CH informa os atributos adicionais ao PDP, para que assim possa avaliar se o usuário possui permissão para executar a requisição em questão.

Este mecanismo é flexível pelo fato de manter a compatibilidade e facilidade oferecida pela especificação do RBAC vinculando direitos a papéis nos domínios remotos. Além disso, é provido o controle de acesso fino porque apenas o usuário autenticado no OIDC e com papéis ativos no seu domínio de origem obtém acesso à aplicação e importa a ativação de seus papéis. Isso só é possível pela integração do mecanismo de SSO do OIDC, com a autorização de acesso do OAuth e do controle de acesso do XACML utilizando os papéis do RBAC. Na prática, é esperado que um usuário que usa SRA acesse outros domínios de maneira transparente, sem a necessidade de ativar seu papel nestes domínios, assim como acontece com SSO para autenticação.

4.4. Escrita de políticas XACML utilizando papéis ativos

Para tornar viável que o PIP busque no serviço RBAC os papéis ativos do usuário, foi necessário adicionar um novo elemento na política XACML. A principal vantagem em alterar apenas a política, se justifica pela não alteração no fluxo e na arquitetura tradicional do XACML. Assim, foi criado um novo tipo de atributo para suportar esse processo.

Ao escrever políticas, o administrador deve utilizar o atributo "*rbac_active_role*" para referenciar os papéis ativos do usuário. Dessa forma, quando o PIP receber a solicitação dos papéis ativos para um dado usuário, buscará as informações no serviço RBAC. A Figura 4.7 apresenta um fragmento de política que utiliza o atributo "*rbac_active_role*".

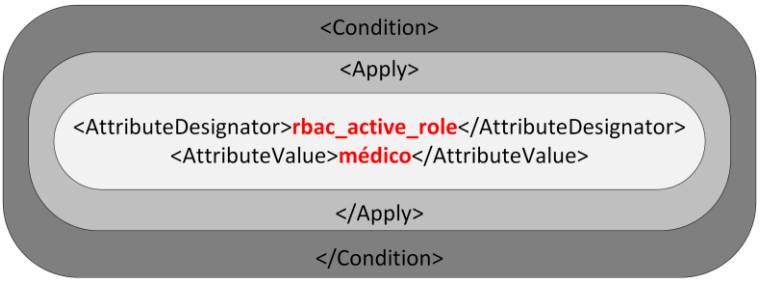

Figura 4.7. Fragmento de política XACML

No fragmento de política XACML ilustrado na Figura 4.7 existe uma condição que necessita que o papel "*médico*" esteja ativo.

Conforme descrito anteriormente, o administrador do sistema pode definir políticas XACML referenciando papéis RBAC de outros domínios. Para isso, o administrador deve inserir o prefixo do domínio seguido do nome do papel, para tirar proveito do SRA. Nesse caso, deve ser utilizado o atributo "*rbac_sra_role*" ao invés de "*rbac_active_role*" na escrita da política XACML. A Figura 4.8 ilustra um fragmento de política utilizando o atributo "*rbac_sra_role*".

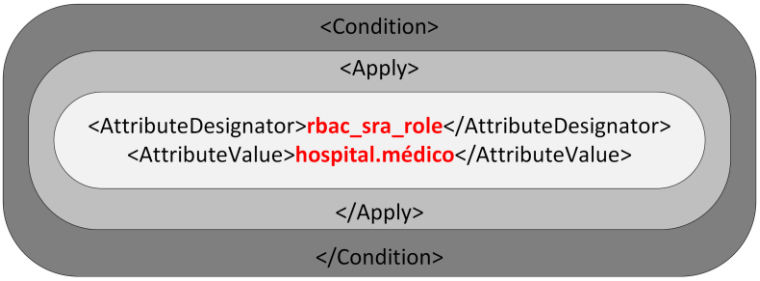

Figura 4.8. Fragmento de política XACML multidomínio com SRA

No fragmento da política XACML ilustrado na Figura 4.8, existe uma condição que necessita que o papel "*médico"* esteja ativo no RBAC do domínio "*hospital"*. Quando uma política que utiliza SRA é adicionada, os papéis relacionados são armazenados em uma área especial no RBAC. Assim, é possível usar esses papéis para definir conflitos de interesses dinâmicos (DSoD). Quando o administrador adicionar uma política, ele é questionado se deseja criar uma DSoD através do papel vinculado.

Capítulo 5

Avaliação

Este capítulo apresenta a implementação e avaliação do protótipo. O protótipo tem como objetivo implementar o modelo de controle de acesso proposto e o mecanismo que importa a ativação de papéis de outros domínios. Este capítulo detalha as bibliotecas utilizadas na implementação do protótipo, uma avaliação funcional de segurança, uma avaliação de performance da proposta e finalmente uma análise de conformidade.

5.1. Implementação do Protótipo

O protótipo foi desenvolvido com a utilização de padrões, tecnologias consolidadas e bibliotecas de código aberto. A App foi implementada em Java através do *framework* Vaadin[1], devido à sua rica experiência de usuário.

A implementação do servidor OIDC utilizou a biblioteca Nimbus[2]. Nimbus é uma biblioteca Java que segue a especificação do OIDC, e de forma adicional implementa o OAuth 2.0. A arquitetura de avaliação do XACML foi implementada com a utilização da biblioteca WSo2 Balana[3], biblioteca em Java baseada na conhecida biblioteca sun-xacml[4], que suporta a versão 3.0 do XACML.

[1] https://vaadin.com/home

[2] http://connect2id.com/

[3] https://github.com/wso2/balana

[4] http://sunxacml.sourceforge.net/

Foram implementados dois *web services* RESTFul utilizando a API JAX-RS[1]. Um *web service* implementa o RBAC, tendo como principais funcionalidades a administração dos papéis, associação dos usuários com os papéis, usuários ativos, papéis ativos e a administração das separações de deveres. O outro *web service* implementa o PEP, honrando as decisões de acesso do PDP contido na biblioteca WSo2 Balana. Todas as funções disponíveis em ambos serviços necessitam de um *token* de acesso emitido pelo OAuth 2.0. Além da necessidade do *token* de acesso ser válido, os serviços verificam se o escopo de acesso é compatível com a função solicitada. Toda a comunicação ocorre via HTTPS, utilizando certificados auto assinados gerados pelo Keytool[2] que permite gerar um par de chave pública/privada e associá-lo a certificados usando assinatura digital.

Para a integração e funcionamento da arquitetura, o PIP foi estendido para coletar os papéis ativos do usuário no serviço RBAC. Por padrão, o WSo2 Balana realiza *cache* dos valores dos atributos que o PIP fornece. Entretanto, como os papéis dos usuários são voláteis, tornou-se necessário limpar o *cache* do PIP toda vez que um usuário ativa/desativa um papel, dessa forma garantindo que o PIP sempre forneça os papéis que estão realmente ativos.

5.2. Protótipo

Esta seção tem como objetivo relacionar algumas das principais funcionalidades do protótipo.

5.2.1. Token de Acesso e Identidade

A Figura 5.1 ilustra um *token* de acesso obtido pelo protótipo. O *token* de acesso dispõe de uma identificação (*ID token*) que é representada por uma *string*. Quando a App utiliza o *token* para acessar um determinado serviço (RBAC ou XACML), ela encaminha o ID do *token* para os serviços. O *token* de acesso utilizado é do tipo *Bearer*, tipo autocontido de autorização,

[1] https://jax-rs-spec.java.net/

[2] https://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html

ou seja, a pessoa que possui o *token* de acesso pode utilizar as autorizações contidas nele. O tempo de vida (lifetime) do *token* de acesso utilizado foi de 6000 segundos. Para realizar a validação do token, foi escolhido a abordagem *bearer*, onde os serviços necessitam solicitar a validação do token online no Controle de Autorização de Acesso.

O *token* possui como escopo de acesso os seguintes valores: *openid*, *profile*, *rbac-home-full*, *rbac-remote-read*. O escopo *openid* e *profile* são referentes às informações do usuário no OIDC. Já o escopo *rbac-home-full* permite que o usuário acesse a todos os recursos disponíveis no RBAC do domínio de origem. Finalmente, o *rbac-remote-read* possibilita que o usuário acesse limitadamente ao RBAC de um determinado domínio remoto. Se o *token* atual expirar, é utilizado o campo "*Refresh token*" para solicitar um novo *token*.

Figura 5.1. *Token* de Acesso

O *token* de identidade contém informações relativas à identidade e à autenticação do usuário. Alguns atributos contidos no *token* de identidade são padrões, conforme a especificação do OpenID Connect (SAKIMURA, 2014). Já os atributos presentes na Figura 5.2 representam os atributos adicionais, necessários para o provimento do SRA.

O atributo "home" representa o domínio de origem do usuário. O atributo "sub" é o ID do usuário no IdP. Finalmente, o atributo "rbac_home" representa o endereço do RBAC do domínio de origem do usuário, utilizado para consultar os papéis ativos do usuário no momento da realização do SRA.

Figura 5.2. Atributos adicionais do *token* de Identidade para prover o SRA

5.2.2. Ativação de papéis e requisições de acesso

A Figura 5.3 ilustra a interface inicial da App após o usuário realizar a autenticação. A tabela "*Avaliable roles*" indica quais papéis estão disponíveis para o usuário ativar. Já a tabela "*Activated roles*" possui os papéis ativos do usuário.

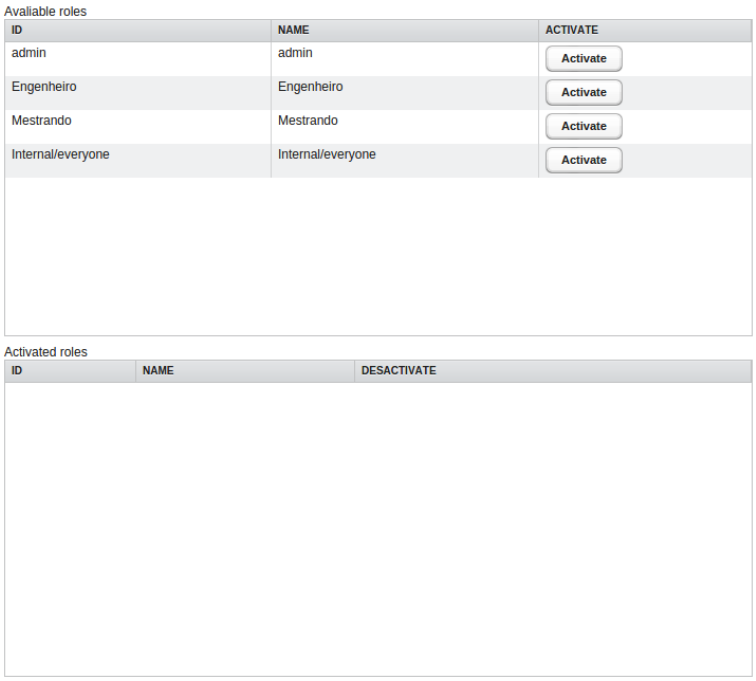

Figura 5.3. Papéis disponíveis e ativos

No momento que o usuário ativa um determinado papel, esse é movido para a tabela de papéis ativos (caso seja ativado com sucesso). A Figura 5.4 ilustra a ativação do papel "Engenheiro".

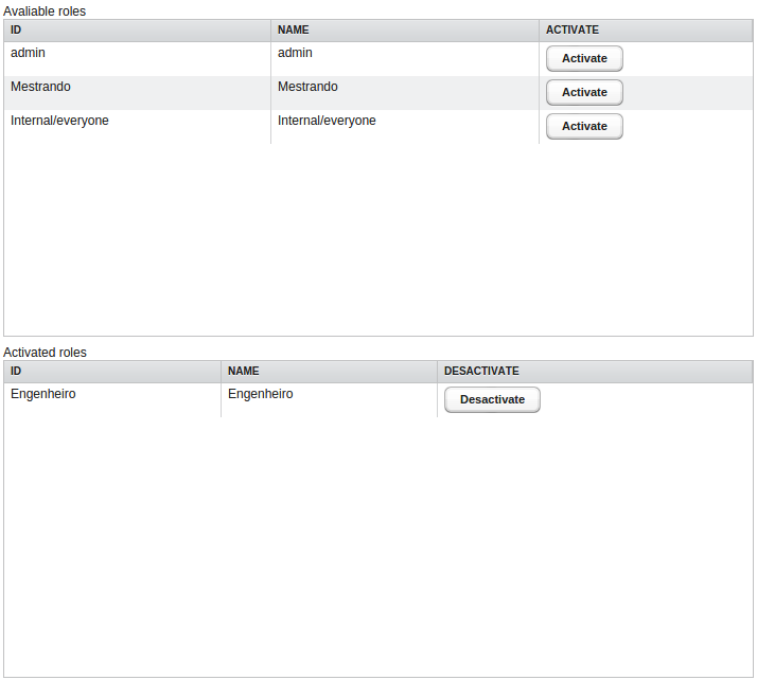

Figura 5.4. Ativação do papel "Engenheiro"

A Figura 5.5 ilustra uma política de acesso no formato XACML armazenada no PAP. Essa política permite que o recurso "Disjuntor" seja acessado pelo papel ativo "Engenheiro" para realizar as operações "read" e "write". As demais requisições de acesso para esse recurso serão negadas. Foi utilizado o tipo de atributo "rbac_active_role", ou seja, não está sendo utilizado o SRA.

```
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"  PolicyId="XACML01" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algori
   <Target>
      <AnyOf>
         <AllOf>
            <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">disjuntor</AttributeValue>
               <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" Category="urn:oasis:names:tc:xacml:3.0:attribute-cate
            </Match>
         </AllOf>
      </AnyOf>
   </Target>
   <Rule Effect="Permit" RuleId="Rule-1">
      <Condition>
         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-at-least-one-member-of">
               <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
                  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
                  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">write</AttributeValue>
               </Apply>
               <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" Category="urn:oasis:names:tc:xacml:3.0:attribute-category
            </Apply>
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of">
               <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"></Function>
               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Engenheiro</AttributeValue>
               <AttributeDesignator AttributeId="rbac_active_role" Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject" DataType="http
            </Apply>
         </Apply>
      </Condition>
   </Rule>
   <Rule Effect="Deny" RuleId="Deny-Rule"></Rule>
</Policy>
```

Figura 5.5. Política XACML

A Figura 5.6 ilustra uma interface de requisição de acesso. Na interface, o usuário preenche o recurso e ação que deseja realizar. Quando o usuário pressionar o botão "Request", a App requisita acesso ao XACML, fornecendo o *token* de acesso, o recurso e a ação (conforme detalhado na seção 4.1, Figura 4.3). Considerando os papéis ativos da Figura 5.4, o resultado da requisição de acesso é permitido.

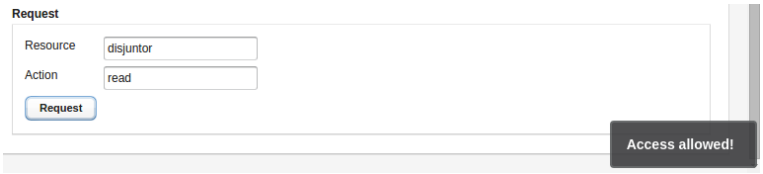

Figura 5.6. Requisição de acesso permitida

A Figura 5.7 ilustra uma requisição de acesso através da operação “execute”. Conforme ilustrado na política de acesso (Figura 5.5), apenas as operações “read” e “write” são permitidas, logo a requisição de acesso é negada.

Figura 5.7. Requisição de acesso negada

5.2.3. Separações de deveres dinâmica

A Figura 5.8 ilustra a interface para adicionar as separações de deveres dinâmica. O usuário seleciona dois papéis que tem acesso e pressiona o botão “Create”. Dessa forma, a restrição de acesso fica adicionada na tabela ao lado.

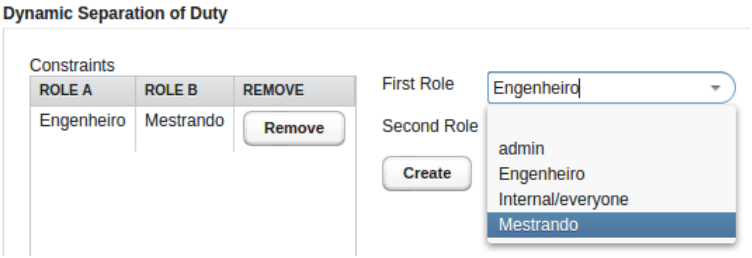

Figura 5.8. Interface de separação de deveres dinâmica

A Figura 5.9 ilustra um cenário onde o papel “Engenheiro” está ativo e o usuário realiza a tentativa de ativar o papel “Mestrando”. Considerando que existe um DSoD entre esses papéis, o usuário recebe uma notificação que informa que não é possível ativar esse papel.

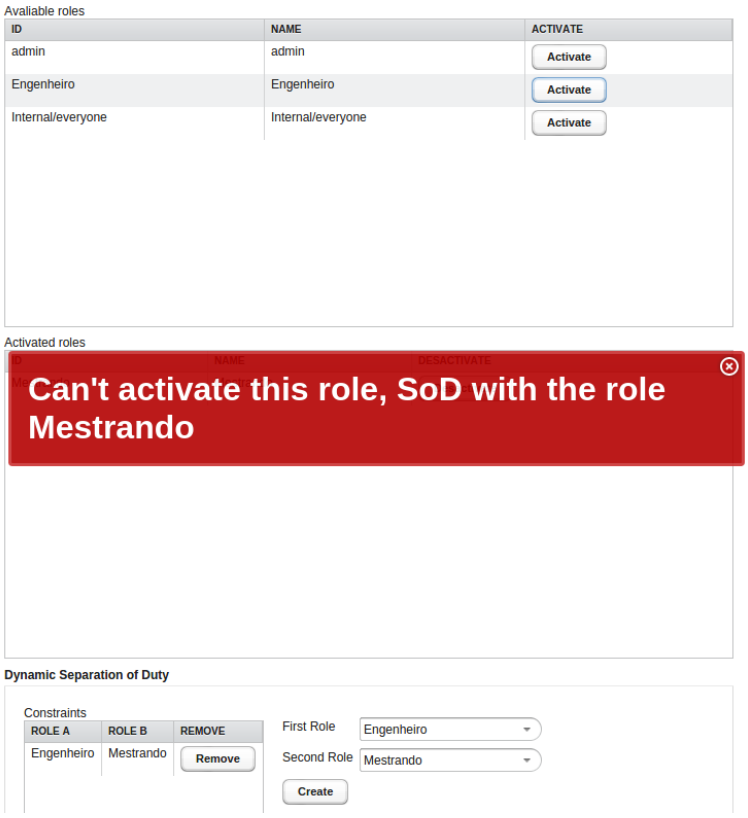

Figura 5.9. Tentativa de ativar um papel conflitante

5.3. Avaliação de Desempenho

A avaliação de desempenho do protótipo foi realizada em um ambiente controlado formado por quatro máquinas em uma rede local, a escolha desse ambiente evita interferências nas medições do tempo. Todas as máquinas físicas possuem a mesma configuração de hardware, com processadores core i7, 8 GB memória RAM conectados em uma rede Gigabit. As máquinas utilizam o sistema operacional Ubuntu na versão 14.10, com o Java 1.7 instalado e *tomcat* 7 para executar os *web services*.

Os componentes da arquitetura são baseados em serviços, sua distribuição fica a critério do administrador do sistema. O cenário ficou distribuído da seguinte forma: duas máquinas hospedam os serviços do RBAC e XACML, representando dois domínios. Uma terceira máquina hospeda o servidor OIDC e a quarta máquina hospeda uma aplicação de teste que automatiza todo o processo, desde a autenticação do usuário até a requisição de acesso. As interações que exigem a intervenção humana, por exemplo, a escolha de papel a ser ativado, foram automatizadas. Dessa forma a aplicação escolhe o primeiro papel disponível e isso se justifica por não ser possível imitar o comportamento do usuário no período de testes.

A aplicação de teste possui dois fluxos, um deles implementa o cenário definido na Figura 4.5, onde é realizado apenas operações em um único domínio e o outro o cenário definido na Figura 4.6, onde existem operações multidomínios. As operações de cada fluxo estão apresentadas na tabela 2.

Um domínio	Dois domínios
1. Gerar Sessão	1. Gerar Sessão
2. Autenticação	2. Autenticação
3. Obtenção de tokens	3. Obtenção de tokens
4. Obtenção de papéis	4. Obtenção de papéis
5. Ativação de papéis	5. Ativação de papéis
6. Requisitar acesso a um recurso N vezes	6. Autenticação no domínio remoto
	7. Obtenção de papéis remoto
	8. Obtenção de tokens remoto
	9. Requisitar acesso a um recurso remoto N vezes
7. Desativação do papel	10. Desativação do papel
8. Logout	11. Logout

Tabela 5.1. Operações da aplicação de teste

As operações da aplicação podem ser agrupadas em quatro grupos:

- **Grupo OpenID**: é composto pelas etapas de gerar sessão, autenticação local, autenticação remota e *logout*;
- **Grupo OAuth:** formado pelas etapas de obtenção de *tokens* locais e remotos;
- **Grupo RBAC:** formado pelas etapas de obtenção de papéis no domínio local, ativação de papéis locais, obtenção de papéis no domínio remoto e desativação de papéis;
- **Grupo XACML:** formado pela etapa de requisição de acesso a um recurso protegido.

Todas as etapas mencionadas comunicam-se com os diversos serviços definidos na arquitetura proposta.

Os testes têm como objetivo avaliar o desempenho do controle de acesso no domínio de origem versus o domínio remoto utilizando SRA. Para tal, foi avaliado o impacto do número de requisições e do número de usuários. Inicialmente foram cadastrados 10 papéis em cada RBAC (domínio de origem e remoto). Em seguida, criou-se 10 políticas em cada XACML vinculadas aos papéis RBAC locais e 10 políticas vinculadas aos papéis RBAC do domínio remoto, com o intuito de realizar o SRA. Foram cadastrados 1000 usuários no OIDC e vinculados a um papel de cada domínio, escolhido de forma aleatória.

No primeiro teste, o cenário foi formado por 10 clientes que realizaram de 1 a 100 requisições de maneira simultânea. A cada iteração são escolhidos 10 usuários de forma aleatória. Os testes foram executados em um cenário com um único domínio e em outro multidomínios. As requisições de acesso envolvem o papel ativo do usuário, ou seja, sempre retornam como permitido o acesso. Como foi desabilitado o *cache* do PDP/PIP, todas as requisições são avaliadas da mesma forma. É possível observar na Figura 5.10, um desempenho equivalente do controle de acesso no domínio de origem em relação ao domínio remoto, com SRA. O desempenho equivalente foi comparado através de métodos estatísticos, apresentados posteriormente.

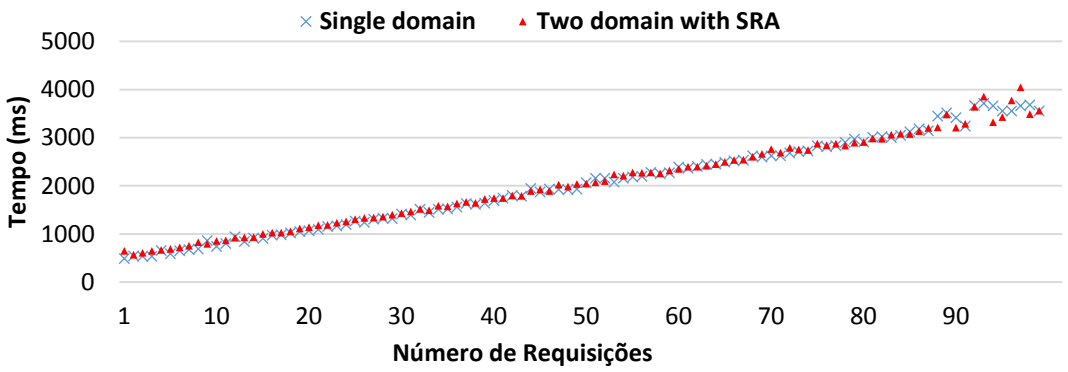

Figura 5.10. Avaliação de desempenho aumentando o número de requisições

A Figura 5.11 ilustra a discretização do tempo de avaliação do número de requisições, baseada nos valores de dois domínios com SRA. A discretização foi realizada com base nos tempos de cada etapa dos testes. É possível observar que o aumento da quantidade de requisições impacta apenas no XACML, nos demais componentes o desempenho se mantém análogo.

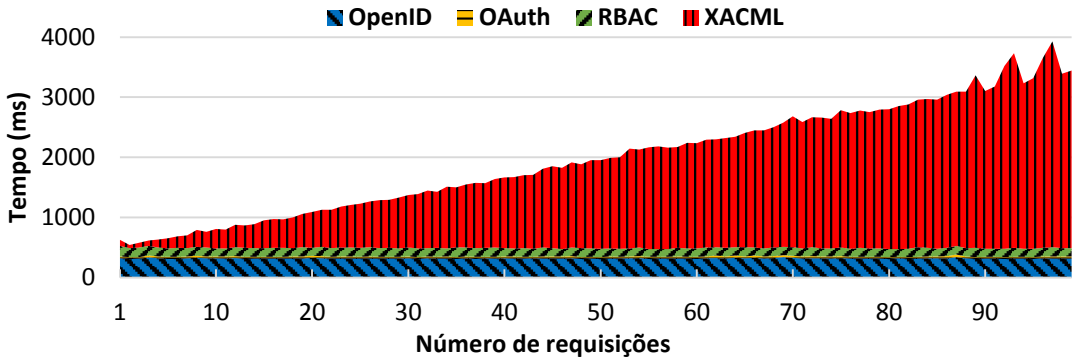

Figura 5.11. Discretização do tempo aumentando o número de requisições

No segundo teste foi definido um cenário onde o número de usuários varia de 1 a 100, cada um realizando 10 requisições de acesso. A cada iteração dos testes os usuários foram escolhidos de maneira aleatória para a execução do cenário. É possível notar uma pequena diferença no desempenho (Figura 5.12). O aumento do número de clientes impacta diretamente em todos os componentes da arquitetura (Figura 5.13).

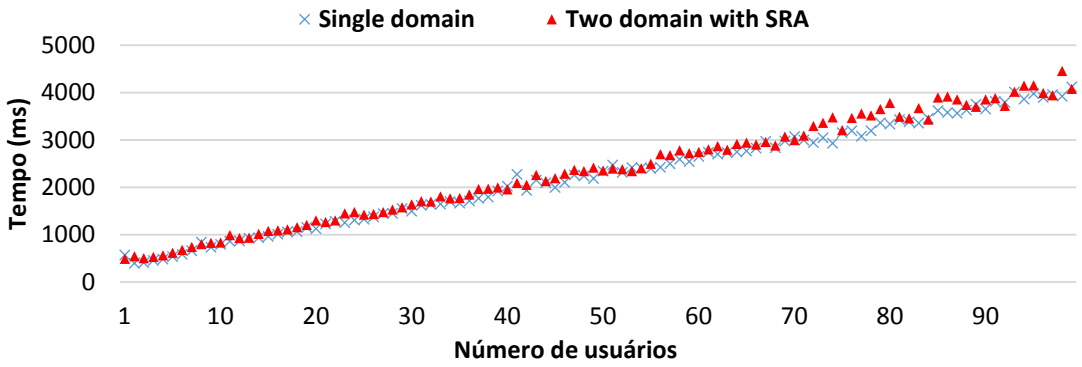

Figura 5.12. Avaliação de desempenho aumentando o número de usuários

Ambos os testes impactam significantemente no tempo de resposta do XACML. Esse comportamento era esperado no primeiro teste, devido ao crescimento do número de requisições de acesso. Porém no segundo teste, isso ocorre porque a aplicação realiza 10 requisições de acesso por usuário, aumentando assim o número de requisições de acesso quando aumenta o número de usuários.

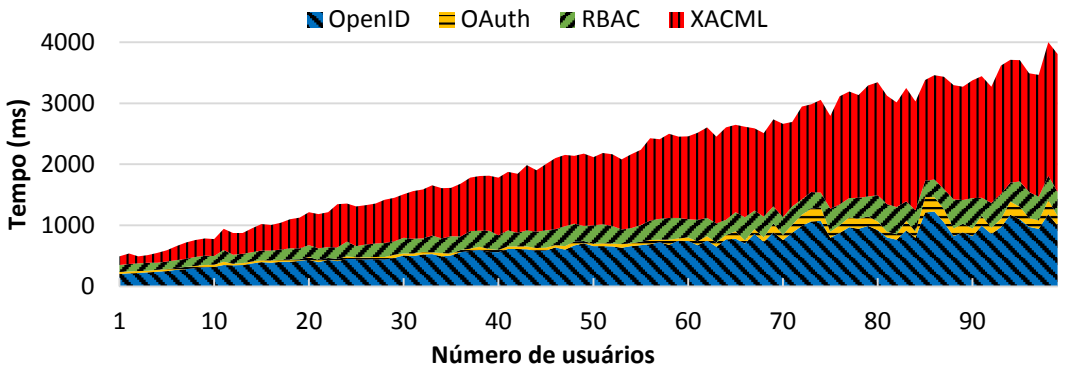

Figura 5.13. Discretização do tempo aumentando o número de usuários

É importante ressaltar que um usuário é identificado pelo seu *token* de acesso nos serviços (RBAC e XACML). A cada iteração de testes o usuário abre uma sessão e no final da iteração é realizado o *logout* da sessão. Dessa forma, cada iteração utiliza um *token* de acesso diferente.

Para realizar os testes de desempenho apresentados realizou-se o cálculo do tempo gasto para cada operação. A Figura 5.14 apresenta um *log* (utilizando outra configuração de hardware) da aplicação de teste executando em um único domínio. No console, é possível observar que a aplicação teve um tempo de execução total de 7793 milissegundos para realizar 100 requisições de acesso feitas por 10 usuários. Cada usuário realizou 7 operações, sendo que o maior tempo gasto foi para realizar as 100 requisições de acesso.

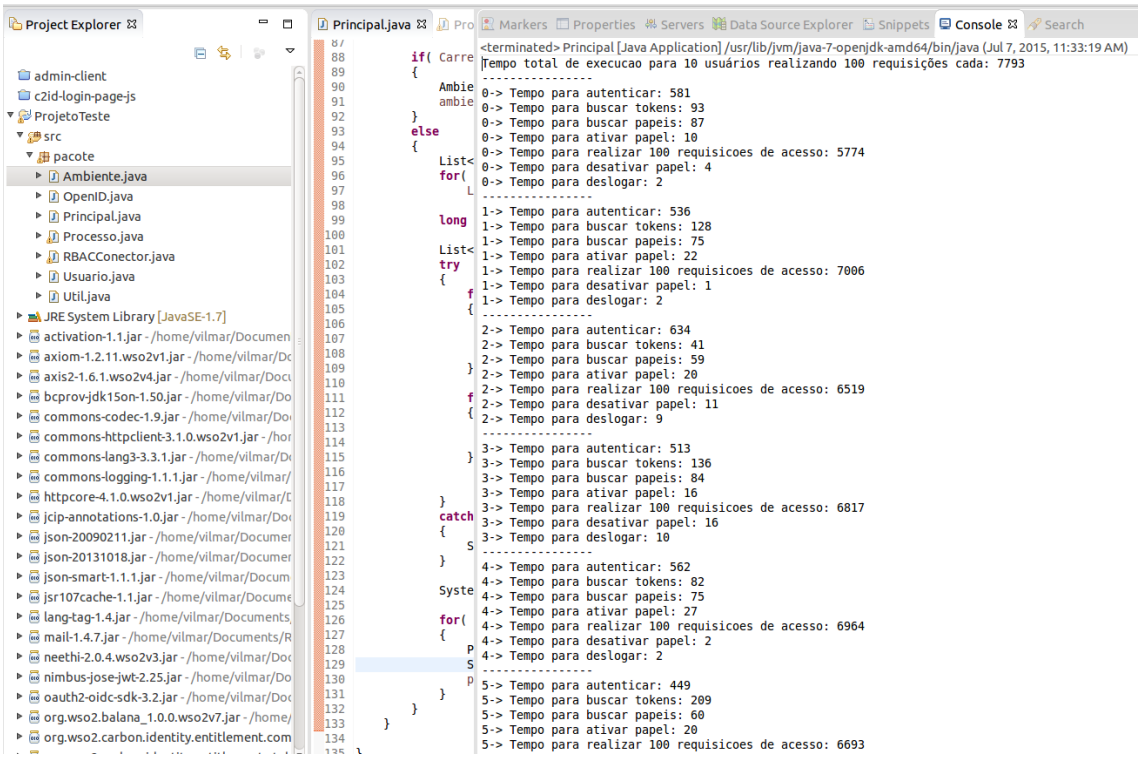

Figura 5.14. *Log* com o resultado dos testes de desempenho

Os dados utilizados para a construção dos gráficos já apresentados anteriormente foram obtidos via arquivos tabulados no formato CSV (*Comma Separated Value*), gerados pela aplicação de testes durante as execuções. Para cada teste executado foram efetuadas 20 repetições, a fim de amenizar possíveis *outliers*. Por exemplo, o teste que utilizou 10 usuários e 100 requisições como parâmetro foi executado 20 vezes e foi utilizada a sua média como valor final. Uma análise estatística foi realizada sobre os dados obtidos no cenário com um único domínio e em multidomínios, a fim de verificar se existe uma diferença significativa entre as amostras.

Para isso, inicialmente foi realizada a distribuição de frequências, conforme exibe a Tabela 5.2. As classes foram criadas baseadas no tempo de execução em milissegundos.

Classes	Um domínio	Multidomínios
0-500	1	0
500-1000	17	16
1000-1500	15	17
1500-2000	17	15
2000-2500	16	18
2500-3000	16	17
3000-3500	9	12
3500-4000	9	4
4000-4500	0	1

Tabela 5.2. Distribuição de Frequências

Com a distribuição de frequências, é possível gerar o histograma das amostras (Figura 5.15). Com base no histograma, a distribuição dos dados não aparenta ser normal, porém, para verificar com precisão se a distribuição é normal ou não, foi o utilizado o teste de Shapiro-Wilk (SHAPIRO, 1965). Conforme esperado, as amostras não têm uma distribuição normal, considerando um alpha (nível de significância) de 0.05.

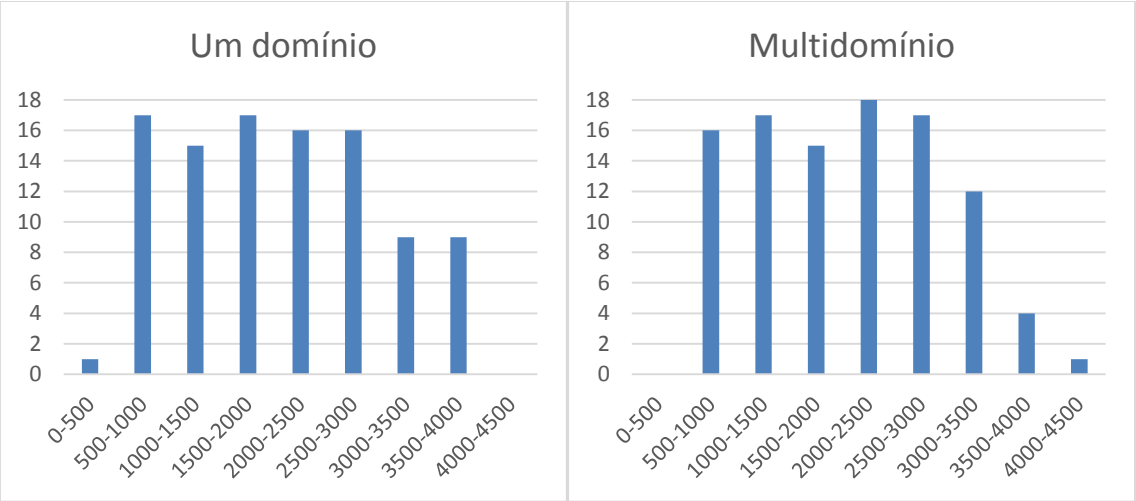

Figura 5.15. Histogramas de um domínio e multidomínios

Para verificar se existe uma diferença significativa entre as duas amostras que não possuem uma distribuição normal, o teste estatístico mais apropriado é o de Mann-Whitney (HETTMANSPERGER, 1998) que considerou como hipóteses:

- H0 (Hipótese nula): Não existe diferença entre o tempo de resposta em um domínio e multidomínios;
- H1 (Hipótese alternativa): Existe diferença entre o tempo de resposta em um domínio e multidomínios.

Dessa forma, um cálculo sobre as distribuições foi realizado e como resultado foi aceita a hipótese nula (p-value=0,389), para um nível de significância de 5%. Ou seja, não existe uma diferença significativa no tempo de execução utilizando um domínio e multidomínios com SRA.

5.4. Avaliação de Segurança

Para destacar os benefícios da utilização do Controle de Autorização de Acesso, a Figura 5.16 ilustra dois cenários. O primeiro cenário (A) não utiliza o *token* de acesso. Dessa forma, é necessário realizar a avaliação de todos elementos do XACML para concluir que um usuário não possui acesso a um determinado recurso. Esse cenário considera que um usuário não autorizado requisita acesso a um recurso protegido *(evento i)*. O PEP intercepta a requisição e encaminha ao *Context Handler*, que solicita os atributos do usuário ao PIP *(evento ii e iii)*. O PIP consulta os papéis ativos no RBAC *(evento iv)*. O RBAC não retorna nenhum papel, pois

o usuário não é autorizado *(evento v)*. Dessa forma, o PIP informa ao *Context Handler* que o usuário não possui nenhum papel *(evento vi)*. O *Context Handler* cria uma requisição XACML ao PDP *(evento vii)*. O PDP avalia que o usuário não tem autorização de acesso e responde ao *Context Handler (evento viii)*, que informará ao PEP *(evento ix)*. O PEP rejeita o acesso ao recurso protegido *(evento x)*.

No segundo cenário (B), o usuário não autorizado necessita informar um *token* de acesso. Como ele não possui um *token* de acesso válido, ele encaminha um conteúdo falso. O usuário não autorizado requisita acesso a um recurso protegido *(evento i)*. O PEP intercepta a requisição e encaminha ao OAuth, que irá validar o *token* de acesso *(evento ii)*. O OAuth rejeita o *token* de acesso e informa ao PEP *(evento iii)*. O PEP rejeita o acesso ao recurso protegido *(evento iv)*.

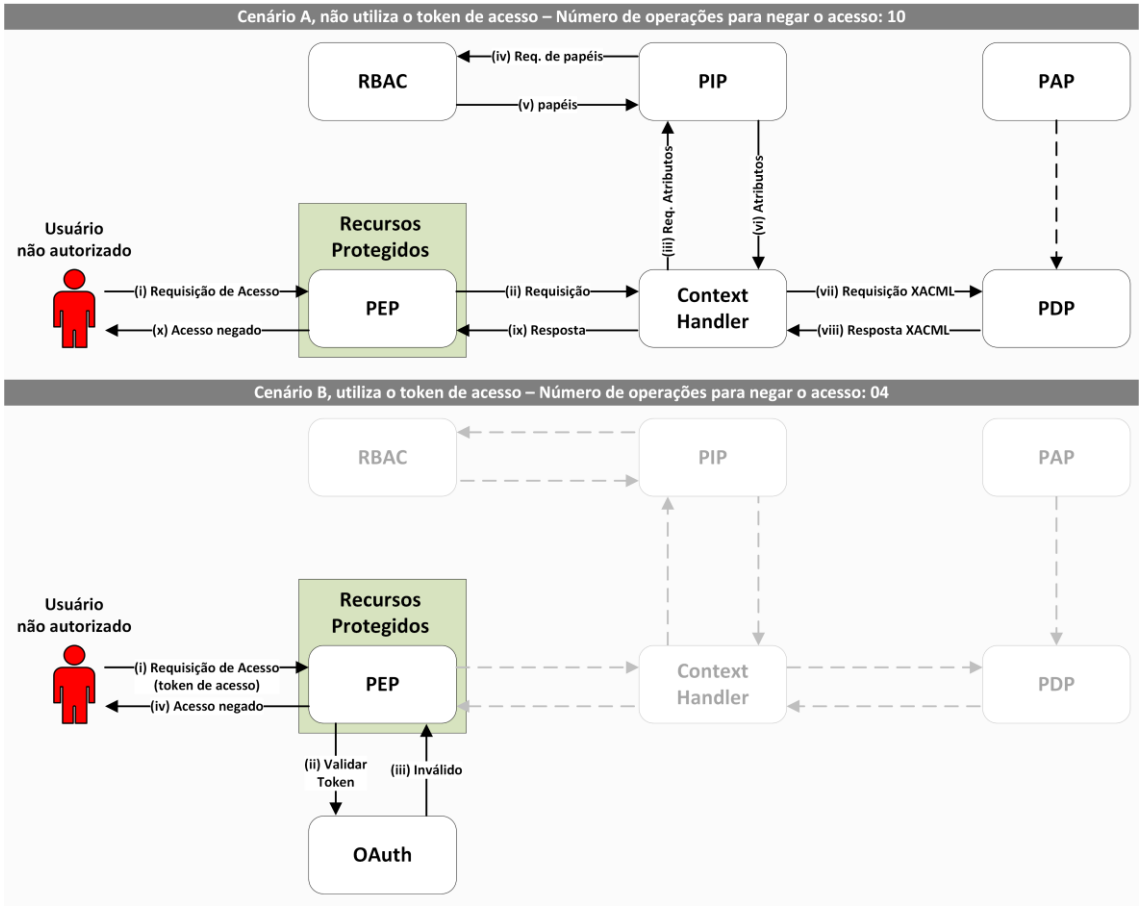

Figura 5.16. Avaliação da utilização do OAuth

Para avaliar os cenários A e B, foi utilizado uma distribuição de máquinas igual ao da subseção 5.3. A avaliação realizada tem o intuito de avaliar o tempo de resposta para a execução de ambos cenários. O resultado da avaliação mostrou que o cenário B, que utiliza o OAuth, foi aproximadamente 15 vezes mais rápido do que o cenário A. Essa vantagem ocorre porque o

número de operações é reduzido no cenário B quando um usuário não possui autorizações necessárias. As operações mais custosas computacionalmente envolvem o PIP e o PDP. Com o emprego do OAuth essas operações não são invocadas para o usuário que não detenha as devidas autorizações. Em um possível caso de negação de serviço (DoS, *Denial of Service*), o cenário A, que não utiliza o OAuth, atinge a exaustão de recursos mais rapidamente.

5.5. Análise de Conformidade

A análise de conformidade tem foco no tratamento dos papéis RBAC no XACML. Este trabalho considerou que os papéis são atributos do usuário, dessa forma as permissões de um papel são armazenadas no PAP. A análise de conformidade tem como principal objetivo verificar se a utilização dos papéis está de acordo a especificação administrativa do RBAC (FERRAIOLO e KUHN, 1992).

Considerando que o XACML define três níveis de elementos: *Rule, Policy* e *PolicySet*. De maneira simplificada, a *Rule* é expressada com um predicado (utilizando papéis) que é avaliada de maneira individual. O elemento *Policy* contém um ou mais *Rules*. Finalmente, o elemento *PolicySet* contém uma ou mais *Policies*. Os elementos *Policy* e *PolicySet* adotam um algoritmo de combinação que é utilizado quando existe conflito de *Rules*, por exemplo: Uma *Policy* contém duas *Rules*, se uma retornar verdadeiro e a outra falsa, é necessário utilizar um algoritmo de combinação para decidir o resultado da avaliação. Os algoritmos de combinação mais conhecidos são: Deny-overrides, Permit-overrides e First-applicable [13].

O modelo do RBAC (FERRAIOLO e KUHN, 1992) possui uma especificação administrativa/funcional que define diretrizes para implementar o RBAC. Dentre elas, a especificação propõe a função *Permission Assignemnt* (PA) que vincula os papéis com todas suas permissões. Esse trabalho considerou que os papéis do usuário são atributos do mesmo. Assim, os direitos que um papel possui estão definidos nas políticas armazenadas no PAP. Dessa forma, a função PA é realizada toda vez que o PDP avalia a decisão de acesso de uma requisição (Figura 2.8, *evento iii*).

Foi necessário adotar uma estratégia para a escrita das políticas XACML utilizando papéis ativos do usuário para evitar uma possível inconsistência de papéis relacionadas a função PA. Por exemplo, um papel Engenheiro necessariamente deve ter a permissão de ler um projeto e escrever um laudo técnico. Se esses direitos forem escritos em *policies* diferentes, pode

ocorrer que o papel Engenheiro só tenha direito a utilizar uma das permissões, ocorrendo uma inconsistência semântica do papel.

A estratégia adotada para a escrita de políticas vincula todos os direitos do papel (PA) na mesma *Policy*. Assim, quando o papel for ativado, o usuário terá direito a todas as permissões do papel ou nenhuma permissão (caso ocorra conflito). Esta estratégia facilita visualização das políticas de segurança além de ser compatível com o modelo RBAC. Os autores (Ferrini et al., 2009; Helil et al., 2010) e o profile do RBAC para XACML (ANDERSON, 2004) não tratam esse problema.

Capítulo 6

Conclusão

Este trabalho desenvolveu um modelo de ativação de papéis multidomínios que considera diferentes semânticas de um papel, permitindo a ativação única de papel (SRA, *Single Role Activation*). Foi utilizado o OpenID Connect para prover a autenticação única (SSO, Single Sign-On) entre os domínios e o OAuth foi empregado para um controle de autorização de acesso para os serviços. O escopo de acesso do OAuth permite restringir a visibilidade dos recursos. O controlador RBAC gerencia a sessão dos papéis, usuários, hierarquia de papéis e as separações de deveres. O XACML é responsável pelo controle de acesso com granularidade fina utilizando os papéis do usuário como atributos.

A proposta apresentada mantém a autonomia do administrador de cada domínio, permitindo que ele mesmo defina as permissões associados a cada papel. Isso permite ao administrador do domínio local definir políticas XACML referenciando papéis de outros domínios. Quando um usuário acessa a um domínio remoto e requisita acesso a um recurso protegido, são considerados os papéis ativos em seu domínio de origem. Se os papéis ativos no domínio de origem não forem conflitantes com os papéis do domínio remoto, o RBAC importa a ativação do papel em questão e o ativa no domínio remoto. Como resultado, um usuário acessa domínios remotos de maneira transparente, sem precisar ativar papéis que estão em uso em seu domínio de origem, assim como acontece com SSO para autenticação.

Portanto, o protótipo revelou a viabilidade da proposta. Os testes relacionados a multidomínios com SRA evidenciaram que o controle de acesso tem um desempenho semelhante a domínio único, com aumento não significativo em relação ao tempo das respostas. Considerando as vantagens qualitativas que o SRA fornece ao usuário e o bom desempenho, a proposta é avaliada como viável e promissora.

6.1. Limitações e Trabalhos Futuros

Esta seção tem como objetivo discutir algumas limitações do trabalho. O trabalho adotou o modelo centralizado de gestão de identidades. Entretanto, o modelo federado apresenta maiores vantagens (CLAU e KESDOGAN, 2005), por naturalmente distribuir os provedores de identidade. Adotar o modelo federado não impacta nos demais elementos da proposta, podendo aproveitar a relação de confiança que existe entre os domínios para estabelecer a relação de confianças dos IdPs.

A validação do token no Controle de Autorização de Acesso foi realizada de forma Online. A principal vantagem dessa abordagem é que o serviço que utiliza o token não necessita verificar a validade do token, apenas delega a validação ao Controle de Autorização de Acesso. O ponto fraco dessa abordagem está relacionado ao tempo gasto para validação do token no Controle de Autorização de Acesso, utilizar uma metodologia off-line para validação de token pode acarretar em ganhos de desempenho.

Conforme descrito na seção 5.1, foi necessário limpar o cache do PIP para garantir que o mesmo sempre forneça os papéis que estão realmente ativos do usuário. Entretanto, isso prejudica o desempenho da proposta. Uma alternativa para resolver esse problema, seria estabelecer um tempo de vida da ativação do papel, durante esse tempo é considerado que o papel está ativo. Porém, caso o usuário venha a desativar o papel, seja no domínio de origem ou no remoto, seria necessário notificar o XACML de cada domínio. Utilizar uma estrutura de *publish/subscribe* pode ser benéfica.

6.2. Publicação

No desenvolvimento desse trabalho de pesquisa, foi publicado o seguinte artigo:

- **V. Abreu**, A. Santin. (2015). Modelo de ativação multi-domínios de papéis RBAC usando controle de acesso baseado em atributos. Em XV Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg).

Referências Bibliográficas

Anderson, A. (2004). XACML profile for role based access control (RBAC). OASIS Access Control TC committee draft, v. 1, n. February, p. 13.

Avita, K., Pranjal, G., Mohammad, W., et al. (2012). Authentication and Authorization Domain Specific Role Based Access Control Using Ontology. *Intelligent Systems and Control (ISCO)*, p. 439–444.

Cao, Y. C. Y. and Yang, L. Y. L. (2010). A survey of Identity Management technology. *2010 IEEE International Conference on Information Theory and Information Security*, p. 287–293.

Clauβ, S., Kesdogan, D. and Kölsch, T. (2005). Privacy enhancing identity management. *Proceedings of the 2005 workshop on Digital identity management - DIM '05*, v. 9, n. 1, p. 35–44.

Ferraiolo, D. F., Kuhn, D. R. and Chandramouli, R. (2003). Role-Based Access Control. In *ACM Transactions on Information and System Security (TISSEC) TISSEC Homepage archive Volume 4 Issue 3, August 2001*. . http://books.google.com/books?hl=en&lr=&id=48AeIhQLWckC&pgis=1.

Ferrini, R. and Bertino, E. (2009). Supporting RBAC with XACML+OWL. In *Proceedings of the 14th ACM symposium on Access control models and technologies*. . http://doi.acm.org/10.1145/1542207.1542231.

Freudenthal, E., Pesin, T., Port, L., Keenan, E. and Karamcheti, V. (2002). dRBAC: Distributed Role-based Access Control for Dynamic Coalition Environments. In *Proceedings 22nd International Conference on Distributed Computing Systems*.

Hardt, D. (2012). The OAuth 2.0 Authorization Framework. In *Internet Engineering Task Force (IETF)*.

Helil, N. and Rahman, K. (2010). RBAC Constraints Specification and Enforcement in Extended XACML. In *Proceedings - 2010 2nd International Conference on Multimedia Information Networking and Security, MINES 2010*.

Hettmansperger, T. P. (1998). Robust Nonparametric Statistical Methods.

Hu, V., Ferraiolo, D. and Kuhn, R. (2014). Guide to Attribute Based Access Control (ABAC) Definition and Considerations. In *NIST Special Publication 800-162*. . http://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.sp.800-162.pdf.

Joshi, J. B. D., Bhatti, R., Bertino, E. and Ghafoor, A. (2004). Access-Control Language for Multidomain Environments. In *IEEE Internet Computing*.

Lee, H. K. (2007). Unraveling decentralized authorization for multi-domain collaborations. *2007 International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2007)*, p. 33–40.

Lee, H. K. (2010). Towards Autonomous Administrations of Decentralized Authorization for Inter-domain Collaborations. *2010 IEEE International Symposium on Policies for Distributed Systems and Networks*, p. 141–145.

Lee, H. K. and Luedemann, H. (2007). Lightweight Decentralized Authorization Model for Inter-Domain Collaborations. In *Proceedings of the 2007 ACM workshop on Secure web services*. . http://dl.acm.org/citation.cfm?id=1314431\npapers3://publication/uuid/2037AD69-7D28-422B-A693-882C9EDB0F97.

Li, Q., Zhangt, X., Qing, S. and Xut, M. (2006). Supporting Ad-hoc Collaboration with Group-based RBAC Model. In *Proceedings of the 2nd International Conference on Collaborative Computing*.

H. Lindqvist, "Mandatory Access Control", Master's Thesis in Computing Science, Umea University, Department of Computing Science, SE-901, 87.

Mouliswaran, S. C. (2015). Representation of Multiple Domain Role Based Access Control Using FCA. *Electrical, Computer and Communication Technologies (ICECCT)*, p. 1–6.

Mouliswaran, S. C. and Kumar, C. A. (2015). Inter-domain Role Based Access Control using Ontology. *Advances in Computing, Communications and Informatics (ICACCI)*, p. 2027–2032.

Osborn, S., Sandhu, R. and Munawer, Q. (2000). Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies. *ACM Transactions on Information and System Security*, v. 3, n. 2, p. 85–106.

Parducci, B. and Lockhart, H. (2013). eXtensible Access Control Markup Language (XACML) Version 3.0. In *OASIS Standard*. . https://www.oasis-open.org/committees/download.php/4412/oasis-xacml-2_0-core-spec-wd-01.pdf.

Power, R. (2000). *Tangled Web: Tales of Digital Crime from the Shadows of Cyberspace*. Macmillan Press.

Sakimura, N., Bradley, J., Jones, M., Medeiros, B. De and Mortimore, C. (2014). OpenID Connect Core 1.0. http://openid.net/specs/openid-connect-core-1_0.html, [acessado em outubro de 2015].

Sandhu, R. S., Coyne, E. J., Feinstein, H. L. and Youman, C. E. (1996). Role-Based Access Control Models. *IEEE Computer*, v. 29, n. 2, p. 38–47.

Sandhu, R. S. and Samarati, P. (1994). Access Control : Principles and Practice. In *Communications Magazine, IEEE*.

Shafiq, B., Joshi, J. B. D., Bertino, E. and Ghafoor, A. (2005). Secure Interoperation in a Multidomain Environment Employing RBAC Policies. In *IEEE Transactions on Knowledge and Data Engineering*.

Shapiro, S. S. and Wilk, M. B. (1965). An Analysis of Variance Test for Normality (Complete Samples). Biometrika, v. 52, n. 3/4, p. 591–611.

Sinnema, R. and Wilde, E. (2013). eXtensible Access Control Markup Language (XACML) XML Media Type.

Zhang, H., Li, Z. and Wu, W. (2012). Open Social and XACML based Group Authorization Framework. In *Proceedings - 2nd International Conference on Cloud and Green Computing and 2nd International Conference on Social Computing and Its Applications, CGC/SCA 2012*.