Modelo de ativação multi-domínios de papéis RBAC usando controle de acesso baseado em atributos

Vilmar Abreu Junior, Altair Olivo Santin

Pontifícia Universidade Católica do Paraná (PUCPR)
Programa de Pós-Graduação em Informática (PPGIa)
Curitiba – Paraná – Brasil

{Vilmar.abreu,santin}@ppgia.pucpr.br

Resumo. *Este artigo apresenta um modelo de controle de acesso baseado em atributos (ABAC) com ativação única (SRA) de papéis RBAC em múltiplos domínios. A ativação de papéis entre os domínios é transparente para o usuário, análoga ao Single Sign-On da autenticação. A autonomia do administrador na definição das permissões de cada papel é mantida em cada domínio. Foi implementado um protótipo integrando XACML com papéis, separações de tarefas e sessões do RBAC, utilizando autorizações OAuth em ambiente de web services RESTFul. As avaliações do protótipo mostraram o mínimo de impacto no tempo de resposta do SRA em relação a um domínio único, observado em requisições simultâneas de acesso.*

Abstract. *This paper presents an attribute based access control (ABAC) model with single role activation (SRA) on a multi-domain RBAC. The role activation among domains is transparent to the user, similar to Single Sign-On authentication. The administrator autonomy on defining each role permissions is kept on each domain. We implemented a prototype integrating XACML with roles, separation of duties and sessions from RBAC, using OAuth authorizations in a RESTFul web services environment. The prototype evaluation showed a minimal impact on SRA response time, compared against a single domain, observed in simultaneous access requests.*

1. Introdução

O acesso não autorizado, principalmente por pessoas internas a um domínio, representa um problema potencial para aplicações [Power 2000]. Esse problema é agravado, devido a exposição da aplicação, quando há operações multi-domínios. O controle de acesso é um mecanismo de segurança que coíbe o acesso de usuários não autorizados. A autenticação precede a autorização, que usa políticas para limitar o acesso a usuários autorizados e legítimos [Sandhu e Samarati 1994]. Os controles de acesso mais recentes são baseados em papéis ou em atributos.

O controle de acesso baseado em papéis (RBAC, *Role-Based Access Control*) agrega vantagens ao controle de acesso tradicional (discricionário e mandatório) [Osborn e Sandhu 2000], porque 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, *Atribute-Based Access Control*) não há associação de permissões à sujeitos ou papéis, mas associação de

permissões à atributos. A avaliação de políticas efetuada em tempo real, é baseada nos atributos, constituindo-se uma vantagem em relação ao RBAC.

O XACML (*eXtensible Access Control Markup Language*), padrão criado pela OASIS, define uma arquitetura e uma linguagem declarativa de controle de acesso [Sinnema e Wilde 2013]. O *profile* RBAC para o XACML define como usar políticas baseadas em papéis no XACML. Por ser *stateless,* o XACML não consegue contemplar todas as características do RBAC, e.g. ativação e conflitos de papéis, consideradas muito importantes para o RBAC.

Tradicionalmente o RBAC foi concebido para operar em um único domínio, alguns autores [Freudenthal 2002][Li 2006][Shafiq e Bertino 2005] apresentam propostas para utilizar o RBAC em ambientes multi-domínios. A principal dificuldade nas operações multi-domínios é a semântica dos papéis, pois um papel terá permissões com diferentes significados em diferentes domínios, mesmo sendo designada com o mesmo nome. Por exemplo, o papel médico no domínio de um hospital A não possuirá necessariamente as mesmas permissões que o papel médico terá no domínio do hospital B. Além disso, os recursos e operações entre domínios podem não ser compatíveis, necessitando uma taxonomia para fornecer-lhes interoperabilidade [Li 2006][Shafiq e Bertino 2005].

Modelos de autenticação e autorização multi-domínios para web oferecem autenticação única (SSO, *Single Sign-On*), mas não oferecem controle de acesso fino, como o OAuth [Hardt 2012]. OAuth limita o acesso a um *path* (URL) protegido, mas não suporta políticas que regrem as operações permitidas nos recursos daquele *path*.

Nossa proposta é criar um controle de acesso multi-domínios que considere as diferentes semânticas de um papel, permitindo a ativação única de papéis (SRA, *Single Role Activation*). Trabalhamos com a hipótese que a arquitetura XACML e o RBAC, integrados as restrições de autorização de acesso do OAuth em um ambiente multi-domínios, oferecerão o suporte para o provimento de SRA. O resultado final é que um usuário acessará diferentes domínios sem precisar ativar papéis, não conflitantes, que já tenha ativado em seu domínio de origem.

O restante do trabalho está organizado da seguinte forma: a seção 2 apresenta a fundamentação; a seção 3 aborda os trabalhos relacionados; a seção 4 descreve a arquitetura proposta; a seção 5 apresenta o protótipo e testes e a seção 6 conclui o trabalho.

2. Fundamentação

Nesta seção serão apresentados o OAuth/OpenID Connect, *Role-Based Access Control*, *Attribute-Based Access Control* e *eXtensible Access Control Markup Language*.

2.1 *OAuth e OpenID Connect*

OAuth é um *framework open source* para autorização de acesso em ambiente web, seu objetivo é permitir que uma aplicação, como terceira parte confiável (*relying party* - RP), tenha acesso restrito a um serviço, sem a necessidade de compartilhamento de credenciais [Hardt 2012]. O OAuth usa um *token* de acesso que define parâmetros para restringir as ações que o usuário pode realizar no domínio do *token*, por exemplo: escopo de acesso, tipo do *token*, tempo de expiração, *token* de *refresh* (referencia um novo *token* quando o atual expirar).

OpenID Connect (OIDC) é um protocolo de autenticação e identidade construído sobre o OAuth 2.0, ou seja, é uma extensão que permite ao RP verificar a identidade de um usuário [Sakimura 2014]. O OIDC usa um *token* de identidade (ID Token) que contém informações sobre a autenticação e atributos do usuário.

O processo de obtenção dos *tokens* pelo OIDC é ilustrado na Figura 1. Primeiramente a RP solicita a autenticação do usuário no provedor OIDC (*evento i*). O usuário fornece suas credenciais, caso sejam válidas o usuário é questionado se autoriza que a RP acesse suas informações definidas no escopo do *token* (*evento ii*). O OIDC responde ao RP sobre o processo de autenticação e autorização de acesso do usuário (*evento iii*). Finalmente, o RP pode obter do OIDC o *token* de autorização de acesso e o *token* de identidade do usuário (*eventos iv e v*, respectivamente).

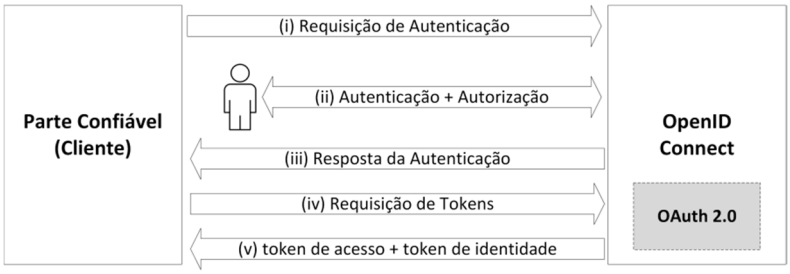

Figura 1. Visão geral das trocas do protocolo OIDC

2.2 RBAC *(Role-based Access Control)*

RBAC é um modelo de controle de acesso baseado em papéis, que permite limitar ações ou operações que determinado usuário (humano ou máquina) pode realizar sobre um recurso [Ferraiolo e Kuhn 2003]. O Modelo de Referência do RBAC é composto por quatro componentes: (i) *Core* RBAC, (ii) *Hierarchical* RBAC, (iii) *Static Separation of Duty* e (iv) *Dynamic Separation of Duty*. O *Core* RBAC define os elementos básicos do RBAC (usuário, papel, recurso, permissão e operação) e o relacionamento entre os mesmos (Figura 2).

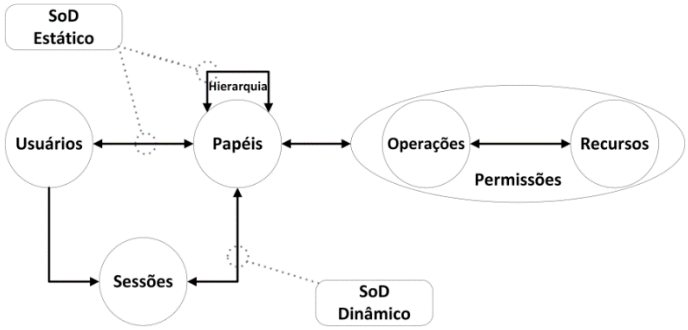

Figura 2. Modelo do RBAC adaptado de [Ferraiolo e Kuhn 2003]

O *Hierarchical* RBAC define um meio natural de estruturar os papéis para refletir as linhas de autoridade e responsabilidade de uma organização. A separação de deveres (tarefas) determina que uma operação crítica deve ser realizada por dois ou mais usuários (diferentes pessoas) para que um usuário individualmente não possa comprometer o sistema. A separação de deveres pode ser efetuada de forma estática ou dinâmica. A abordagem estática (*Static Separation of Duty Relations* - SSoD) é implementada na associação de usuário a papéis e tem como objetivo limitar a ação conflitante de dois

papéis controlados pelo mesmo usuário. Por outro lado, a abordagem dinâmica (*Dynamic Separation of Duty* - DSoD) se aplica a ativação de papel, onde um usuário pode ter dois papéis possivelmente conflitantes associados a si, desde que não ative ambos ao mesmo tempo.

2.3 ABAC *(Attribute-Based Access Control)*

ABAC é um modelo de controle de acesso que adota políticas expressas em forma de predicados, utilizando atributos [Ferraiolo e Kuhn 2014]. O ABAC evita a necessidade de que as permissões sejam ligadas diretamente a um usuário ou papel. Assim, quando uma requisição de acesso é solicitada, uma *engine* realiza a decisão baseado nos atributos do usuário, recurso, ambiente, entre outros. É possível utilizar papéis no ABAC, se for considerado que os papéis são atributos do sujeito.

2.4 XACML *(eXtensible Access Control Markup Language)*

XACML define uma linguagem, baseada em XML, para escrita de políticas de controle de acesso, requisições, respostas e uma *engine* de avaliação das políticas [Parducci e Lockhart 2013]. Por padrão, o modelo de avaliação do XACML é baseado em ABAC, entretanto o XACML possui um *profile* para RBAC, onde os papéis são atributos do sujeito. As principais entidades do XACML são (Figura 3): (i) PAP (*Policy Administration Point*), que permite criar e armazenar as políticas; (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 gera a decisão de acesso, que podem ser: permitido ou negado, em nosso caso; e (v) *Context Handler*, que realiza a coordenação, adequação e interoperabilidade de atributos, requisições e credenciais entre as entidades do XACML.

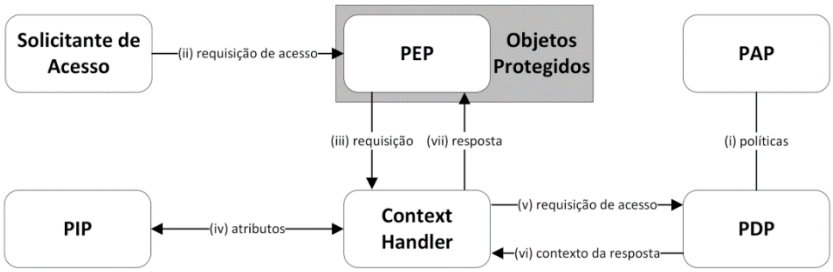

Figura 3. Arquitetura simplificada das entidades XACML

3. Trabalhos Relacionados

Diversos trabalhos objetivaram resolver o problema do RBAC distribuído em multi-domínios. Shafiq e seus colegas (2005) propõe um algoritmo que centraliza os papéis e os mapeia globalmente através de uma operação de junção (*merge* de papéis). Tal operação verifica em cada papel se existe a intersecção de permissões com algum papel de outro domínio. Esta operação de junção centralizada dos papéis impõe um custo computacional elevado, uma vez que é essencial comparar e mapear todos os papéis entre si. Além disto, no caso de atualização é necessário, na maioria das vezes, refazer todas as junções/mapeamentos. Além disto, é necessário considerar uma área de compartilhamento dos recursos, para que esses recursos sejam visíveis a todos os domínios. Por fim, há um problema ligado a 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 e seus colegas (2006) foi adotado o conceito de virtualização sob demanda de papéis. O administrador RBAC define os papéis que deseja utilizar em múltiplos domínios, então é criado um link de referência desses papéis em um domínio global. Essa abordagem é uma evolução do trabalho de Shafiq (2005), trazendo como principal vantagem a escolha de papéis sob demanda, não sendo necessário incluir todos os papéis no domínio global. Porém, ainda se mantém como problema a centralização dos papéis em um domínio global, a necessidade de compartilhamento de recursos, e a lida com a semântica das permissões.

O trabalho de Freudenthal e outros (2002) adotou um repositório de credenciais denominado *wallet* que armazena as delegações de autorizações usando papéis. A delegação é composta de três elementos no formato "[Usuário -> Papel] Domínio". Cada domínio possui uma *wallet* e na medida do possível todas as *wallets* estão sincronizadas entre si usando um serviço *publish*/*subscribe*. Caso uma delegação não exista na *wallet* local, é utilizado um serviço de descoberta para localizar a *wallet* de origem do recurso envolvido na delegação; 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 estar em várias *wallets*.

Alguns trabalhos tentaram resolver os problemas existentes no *profile* do RBAC para XACML, o trabalho de Helil e seus colegas (2010) tem como objetivo estender o *profile* para suportar tanto a separação de tarefas estáticas quanto a dinâmica. Para isto, é desenvolvida uma biblioteca de avaliação de XACML baseada na sun-xacml (sunxacml.sourceforge.net), acrescentando novas entidades responsáveis pela gestão dos papéis ativos e separações de deveres. A proposta tem a limitação de necessitar a alteração da arquitetura XACML e a criação de uma biblioteca própria.

Ferrini e outros (2009) propõe um *framework* que integra ontologias as políticas XACML para suportar o RBAC, a ideia é desacoplar os conflitos e hierarquias que não são tratados no *profile* do RBAC da arquitetura XACML. Os aspectos negativos da proposta são a alteração do fluxo tradicional de funcionamento e da linguagem do XACML, tornando-o incompatível com a padronização.

Zhang e seus colegas (2012) propõe uma solução de autorização para redes sociais utilizando OAuth e XACML. O trabalho consiste em utilizar o OAuth em nível de aplicação para autorizar acesso, e o XACML para realizar o controle de acesso de granularidade fina. A principal limitação do trabalho é que arquitetura e a integração não foram explicadas no trabalho.

4. Proposta

Esta seção apresenta inicialmente o controle de acesso proposto em um domínio único e posteriormente seu uso em ambiente multi-domínios.

4.1 Controle de Acesso

Nossa proposta de controle de acesso (Figura 4) é baseada na integração de (i) Controle de Autenticação, responsável pela autenticação e identidade do usuário; (ii) Controle de Autorização de Acesso, responsável pela emissão de *tokens* que autorizam o acesso do usuário a serviços protegidos; (iii) Controle de Acesso baseado em papéis, que controla

os papéis, hierarquias e conflitos de interesse e (iv) Controle de Acesso baseado em atributos, que controla o acesso do usuário baseado nos seus atributos.

O usuário requisita a autenticação para a Aplicação (Figura 4, *evento i*). A Aplicação (App) cria uma sessão para o usuário e solicita autenticação ao Controle de Autenticação (OIDC) passando a sessão do usuário (*evento ii*). O usuário informa suas credenciais, caso sejam válidas, o OIDC fornece um *ticket* para a App (*evento iii*), que solicita autorização para o Controle de Autorização de Acesso (OAuth) fornecendo o *ticket* (*evento iv*). O OAuth fornece um *token* (*evento v*) que permite que o usuário acesse o Controle de Acesso baseado em papéis (RBAC) e o Controle de Acesso baseado em atributos (ABAC). Com o *token* a App pode acessar o RBAC para requisitar e ativar papéis (*evento vi e vii*, respectivamente) e o ABAC para requisitar acesso a recursos protegidos.

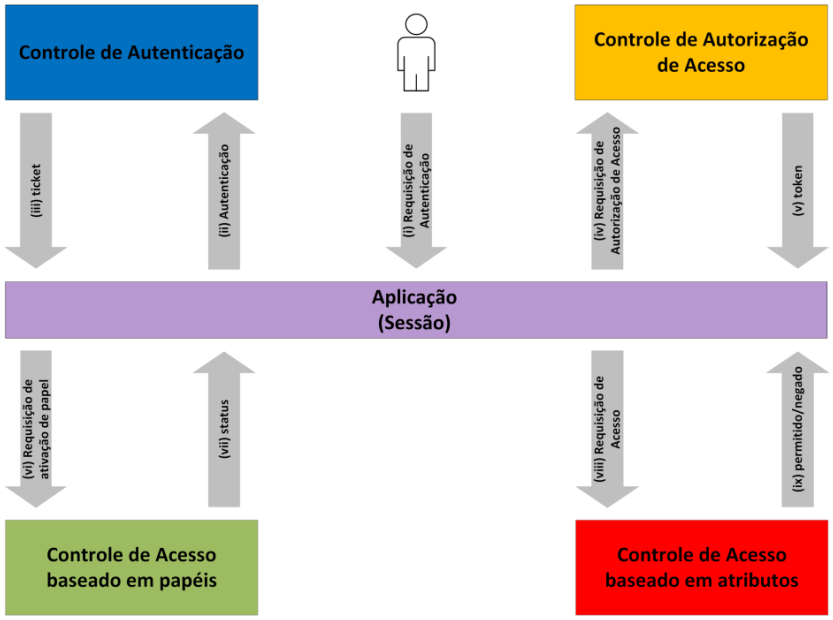

Figura 4. Modelo de controle de acesso proposto

Nesta proposta tem-se primeiramente o esquema de autorização de acesso (OAuth), funcionando como um controle de admissão para os controles de acesso. Depois o controle da sessão (RBAC) e controle de acesso de granularidade fina (XACML) baseado em políticas para avaliar os direitos dos usuários para executar operações nos objetos de cada recurso.

A arquitetura do modelo de controle de acesso foi concebida para que os componentes atuem como serviços, sendo que as funcionalidades disponíveis nos serviços exigem um *token* de acesso, cuja validação no OAuth é online. Para cada serviço são necessários dois escopos (restrição de visibilidade dos recursos) diferentes contidos no *token* de acesso, um escopo permite que o usuário realize apenas a leitura e outro permite a leitura e atualização do estado de um recurso. Esta restrição só permite que usuários autenticados com *tokens* de acesso válidos sejam admitidos no sistema. Para ativar um papel no RBAC, por exemplo, o usuário necessita ter um *token* de acesso com o escopo "rbac_origem_irrestrito". Para consultar seus papéis ativos, o usuário necessita ter um *token* com o escopo "rbac_origem_leitura" ou "rbac_origem_irrestrito".

A Figura 5 ilustra um diagrama de sequência para a ativação de um papel. O

usuário solicita a App a ativação de um papel que o mesmo tem 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* ao OAuth (*evento 1.1.1*), caso o *token* de acesso seja válido, o RBAC extrai deste o usuário vinculado (*evento 1.1.2*). Finalmente, o RBAC verifica se existe algum conflito de interesses (SoDD, *evento 1.1.3*) na ativação do papel, caso não exista é retornado o status de sucesso.

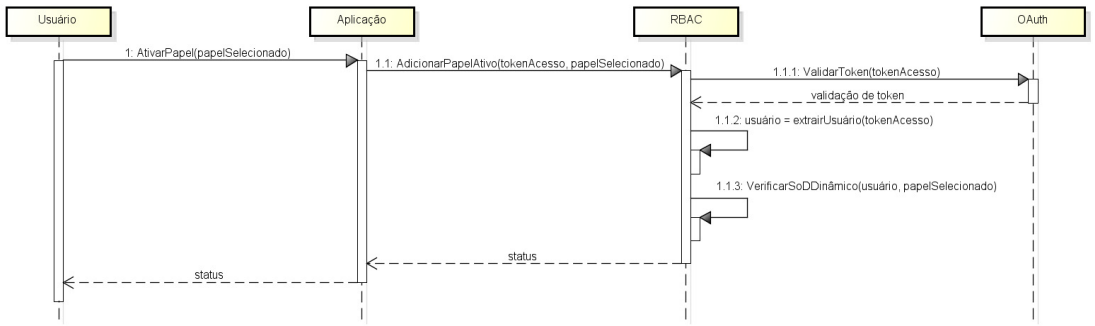

Figura 5. Diagrama de sequência para ativação de papéis

Cada entidade (ator) do XACML é concebida para ser um serviço. O PIP foi estendido para suportar o atributo do tipo "*rbac_active_role*", utilizado pelo administrador na escrita das políticas. A Figura 6 mostra um fragmento de política utilizando o atributo "*rbac_active_role*".

```
<Condition>
 <Apply>
  <AttributeDesignator>rbac_active_role</AttributeDesignator>
  <AttributeValue>engenheiro</AttributeValue>
 </Apply>
</Condition>
```

Figura 6. Fragmento de política XACML

Quando o PIP recebe a solicitação dos papéis ativos para um dado usuário, buscará as informações no serviço RBAC. A integração entre esses componentes pode ser observada na Figura 7. A App requisita a autenticação do usuário no OIDC informando suas credenciais (*evento i*). O OIDC valida as credenciais e retorna um código temporário (*nonce*) para a App (*evento ii*). O código temporário é utilizado pela App para obter o *token* de autorização de acesso (*tokenAcesso*) e o *token* de identidade (*idToken*), representado nos *eventos iii* e *iv*, respectivamente. Portando o *token* de acesso, a App requisita ao RBAC os papéis vinculados ao usuário (*evento v*). O RBAC faz a validação online do *token* de autorização de acesso, retornando os papéis disponíveis para este usuário escolher qual pretende ativar (*evento vi*). A ativação de papel é realizada enviando ao RBAC o *token* de autorização de acesso e o papel que deseja ativar (*evento vii*). O sucesso da ativação depende das restrições devidas ao conflito de interesses dinâmicos de papéis (DSoD) impostas pelo administrador do RBAC (*evento viii*).

Em nome do usuário, a App requisita acesso ao XACML, informando a ação (*action*) que deseja realizar sobre determinado recurso (*target*) juntamente com o *token* de autorização de acesso (*evento ix*). Essa requisição chegará ao PEP, que a encaminhará ao Context Handler (CH). O CH constrói uma requisição XACML para ser enviado ao

PDP, que avaliará a política e decidirá (permitido ou negado). Para construir a requisição, o CH verifica a existência de um atributo especial chamado de "*rbac_active_roles*". Assim, o CH terá que solicitar ao PIP que busque os papéis ativos para este usuário. Para obter os papéis ativos, o PIP solicita ao serviço RBAC passando o *token* autorização de 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 a encaminha ao PEP, que honra a decisão do PDP, liberando ou bloqueando o acesso do usuário ao recurso (*evento x*).

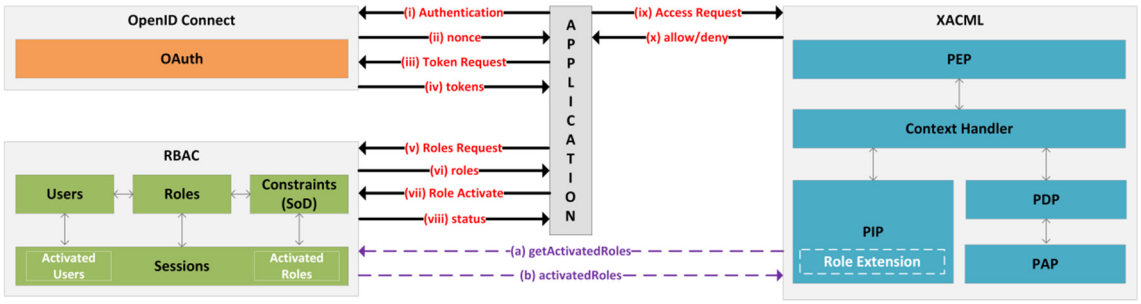

Figura 7. Visão detalhada do modelo proposto

4.2 Ativação Multi-domínio de papéis RBAC

A proposta visa manter a autonomia do administrador de cada domínio no sentido de permitir que o mesmo possa definir os direitos associados a cada papel e ao mesmo tempo facilitar a vida do usuário, evitando que o mesmo precise ativar papéis em cada domínio que visite. Assim, é proposto um mecanismo que permite importar a ativação de um papel, feita no domínio de origem do usuário, em domínios remotos que o mesmo visite. A importação da ativação, denominada *Single Role Activation* (SRA), significa que se o usuário tem um papel ativo em seu domínio de origem e poderá reusar a ativação num domínio remoto. Assim, não será necessário executar as etapas de ativação do papel no domínio remoto. Porém, os direitos que o papel, cuja a ativação foi importada, terá no domínio remoto são definidos pelo administrador do domínio remoto.

O administrador do sistema local (remoto para o usuário) pode definir políticas XACML referenciando papéis de outros domínios, colocando o prefixo do domínio seguido do nome do papel, para tirar proveito do SRA. Neste caso, deve ser utilizado o atributo "*rbac_sra_role*" ao invés de "*rbac_active_role*" na escrita da política XACML. Uma extensão do PIP foi desenvolvida para suportar esse tipo de atributo. A Figura 8 mostra um fragmento de política utilizando o atributo "*rbac_sra_role*".

```
<Condition>
 <Apply>
  <AttributeDesignator>rbac_sra_role</AttributeDesignator>
  <AttributeValue>dominio.engenheiro</AttributeValue>
 </Apply>
</Condition>
```

Figura 8. Fragmento de política XACML multi-domínio com SRA

A Figura 9 ilustra o funcionamento do SRA. Por ser baseado em SSO, o SRA necessita que o usuário esteja autenticado no OIDC e portando o *token* de autorização de acesso, obtido em seu domínio de origem (domínio A). Em seu domínio de origem, o usuário pode ativar os papéis que tem acesso, conforme explicado na seção 4.1.

Considerando que o usuário tenha papéis ativos em seu domínio de origem e que deseje 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 deseja realizar sobre determinado recurso juntamente com o *token* de autorização de acesso (Figura 9, *evento ii*). O escopo do *token* de autorização ode acesso contém a permissão para que o usuário acesse o domínio remoto no modo de leitura. O PEP recebe a requisição e repassa para o CH, que criará uma requisição XACML para o PDP. Após o PDP buscar a política vinculada ao recurso no PAP, e o CH verificar que é necessário o atributo "*rbac_sra_role*". O CH invocará o PIP requisitando os papéis ativos do usuário no domínio de origem.

O PIP usará o *token* de autorização 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. Então, o PIP requisita os papéis ativos do usuário ao RBAC do domínio de origem (Figura 9, *evento iii*).

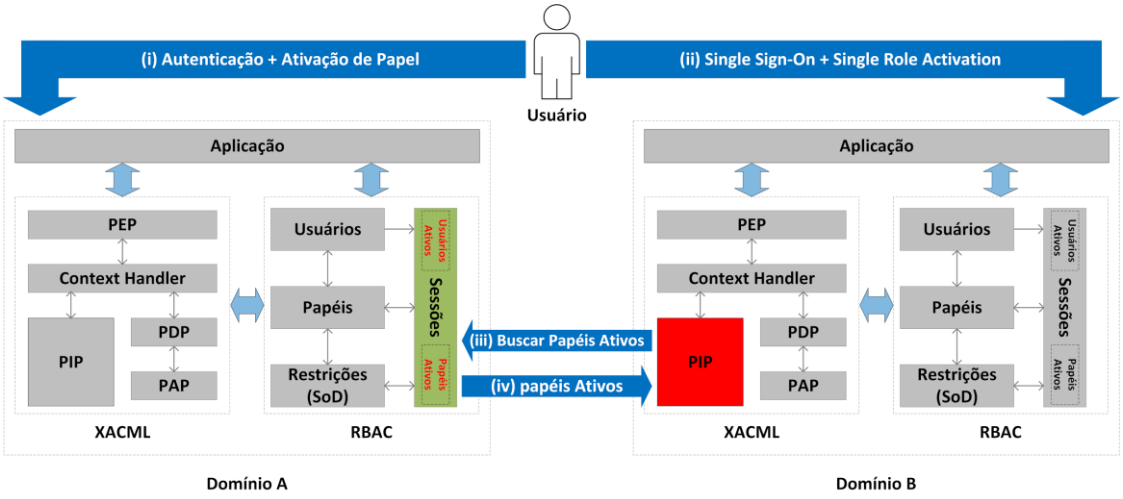

Figura 9. Visão detalhada do modelo multi-domínios

É importante ressaltar que os papéis ativados no domínio de origem do usuário e importados em domínio remoto por SRA, têm direitos locais específicos e estes podem conflitar com papéis locais ativos. Assim, nem sempre um papel pode ser ativado por SRA. Quando a ativação for 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 por SRA). Enfim, o CH informa os atributos adicionais ao PDP, para que assim possa avaliar se o usuário possui permissão para executar a requisição em questão (Figura 9).

Este mecanismo é flexível porque mantém a compatibilidade e facilidades oferecidas pela especificação do RBAC, vinculando direitos a papéis nos domínios remotos. Além disto, é provido o controle de acesso fino porque apenas o usuário autenticado no OIDC e com papéis ativos no seu domínio de origem consegue ter acesso a aplicação e importar a ativação de seus papéis. Isto só é possível pela integração do mecanismo de SSO do OIDC a autorização de acesso do OAuth e do controle de acesso do XACML, utilizando os papéis do RBAC, conforme apresentado anteriormente. Na prática se espera que um usuário que usa SRA acesse outros domínios de maneira

transparente, sem precisar ativar papel nestes domínios, assim como acontece com o SSO na autenticação.

5. Protótipo

Esta seção apresenta o protótipo que implementa a proposta e os testes de avaliação.

5.1 Implementação

O protótipo utiliza padrões e tecnologias consolidadas, e bibliotecas de código aberto. A App (aplicação) foi implementada em Java utilizando o framework Vaadin[1].

O servidor OIDC foi implementado utilizando a biblioteca Nimbus[2] – desenvolvida em Java, seguindo a especificação do OAuth/OpenID Connect. A arquitetura de avaliação do XACML foi implementada utilizando a biblioteca WSo2 Balana[3] – desenvolvida em Java, baseada na conhecida biblioteca sun-xacml[4], que suporta a versão 3.0 do XACML.

Foram implementados dois *web services* RESTFul utilizando a API JAX-RS[5]. 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, guardião que honra as decisões de acesso do PDP (WSo2 Balana). Todas as funções disponíveis em ambos serviços necessitam de um *token* de autorização de acesso emitido pelo OAuth 2.0. Além da necessidade do *token* de autorização de acesso válido, os serviços verificam se o escopo de acesso é compatível com a função solicitada. Toda a comunicação é realizada por HTTPS, utilizando certificados auto-assinados gerados pelo keytool[6].

Para a integração e funcionamento da arquitetura, o PIP foi estendido para buscar os papéis ativos do usuário no serviço RBAC. Por padrão, o WSo2 Balana realiza *cache* dos valores dos atributos que o PIP fornece para otimizar o processo. Entretanto, como os papéis dos usuários são temporários, foi necessário limpar o *cache* do PIP toda vez que um usuário ativa/desativa um papel, para que o PIP busque e forneça sempre os papéis que estão realmente ativos no domínio de origem do usuário.

5.2 Avaliação

Para realizar a avaliação do protótipo foram utilizadas quatro máquinas em uma rede local, com o objetivo de ter um ambiente controlado para evitar interferências nas medições do tempo. Os componentes da arquitetura são baseados em serviços e a distribuição dos mesmos fica a critério do administrador do sistema.

Nosso cenário ficou organizado 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. Todas as máquinas são físicas e conectadas via rede local. As interações que exigem um humano, por exemplo, a escolha de um papel a ser ativado foram automatizadas,

[1] https://vaadin.com/home
[2] http://connect2id.com/
[3] https://github.com/wso2/balana
[4] http://sunxacml.sourceforge.net/
[5] https://jax-rs-spec.java.net/
[6] https://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html

escolhendo o primeiro papel disponível, pois não é possível imitar o comportamento do usuário nos testes.

A aplicação de teste implementa o cenário da Figura 9, composta por 11 facilidades (*facilities*), sendo que podem ser agrupadas em quatro grupos, (i) Grupo OpenID é composto das etapas de gerar sessão, autenticação local, autenticação remota e *logout*; (ii) Grupo OAuth é composto das etapas de obtenção de *tokens* locais e remotos; (iii) Grupo RBAC é composto das etapas de obtenção de papéis no domínio local, ativação de papéis locais, obtenção de papéis no domínio remoto e desativação de papéis, e (iv) Grupo XACML é composto da etapa de requisição de acesso a um recurso protegido. Todas as etapas mencionadas anteriormente 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, avaliamos o impacto do número de requisições e do número de usuários.

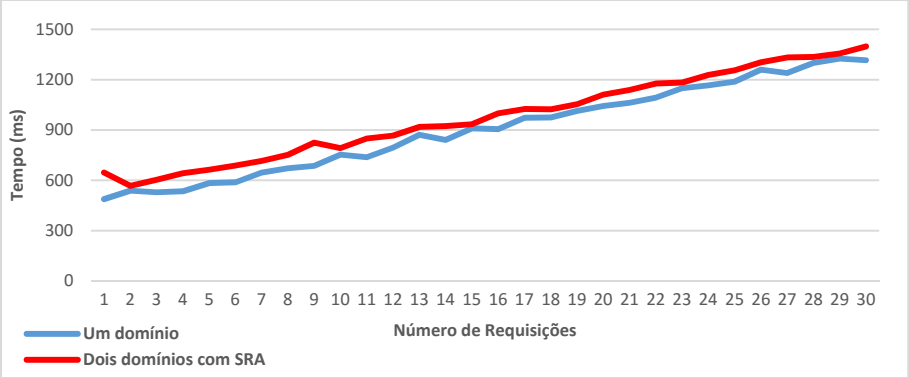

Figura 10. Avaliação do teste (i)

Primeiramente foram cadastrados 10 papéis em cada RBAC. Em seguida, foram criadas 10 políticas em cada XACML vinculadas aos papéis locais de seus controladores RBAC e 10 políticas vinculadas aos papéis do outro domínio, com o intuito de realizar o SRA. Então, foram cadastrados 1000 usuários no OIDC, sendo vinculado cada usuário a um papel de cada domínio, escolhido de forma aleatória. Dessa forma, temos um ambiente preparado para a realização dos testes.

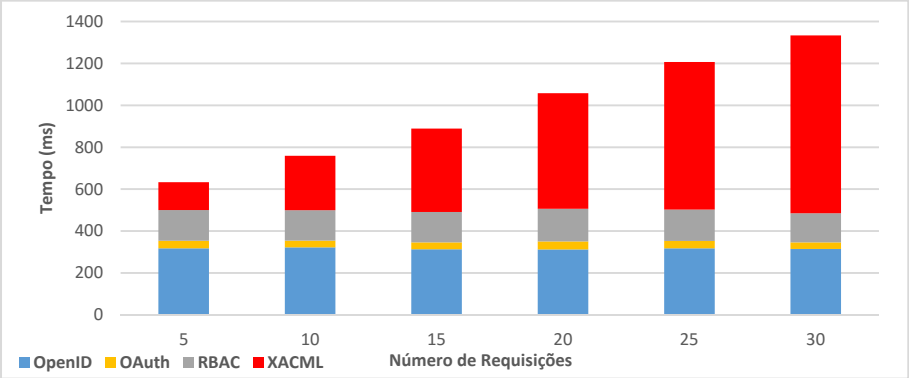

Figura 11. Discretização do tempo de resposta do teste (i)

No primeiro teste (i), definimos um cenário com 10 usuários realizando de 1 a 30 requisições de maneira simultânea. Assim, escolhemos os usuários de forma aleatória em cada iteração do número de requisições e executamos o cenário realizando requisições em um único domínio e em múltiplos domínios. As requisições de acesso envolvem o papel ativo do usuário, ou seja, sempre retornam o acesso permitido. Como desabilitamos o *cache* do PDP/PIP, todas as requisições são avaliadas da mesma forma. É possível observar na Figura 10, um desempenho equivalente do controle de acesso no domínio de origem em relação ao domínio remoto, com SRA.

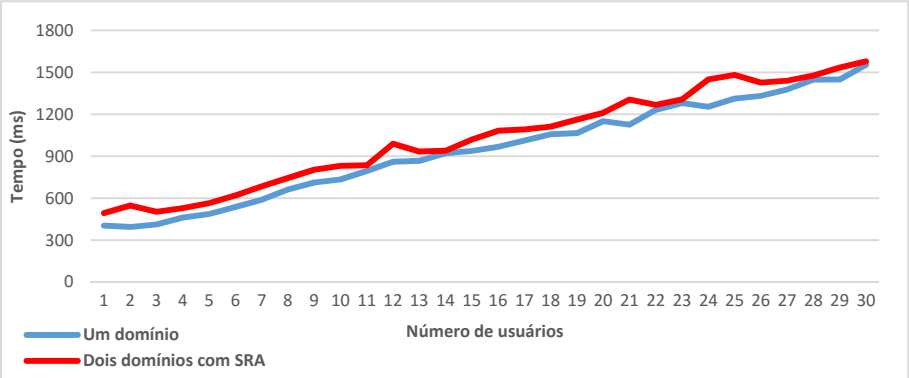

Figura 12. Avaliação do teste (ii)

A Figura 11 ilustra a discretização do tempo de avaliação do número de requisições, baseado 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.

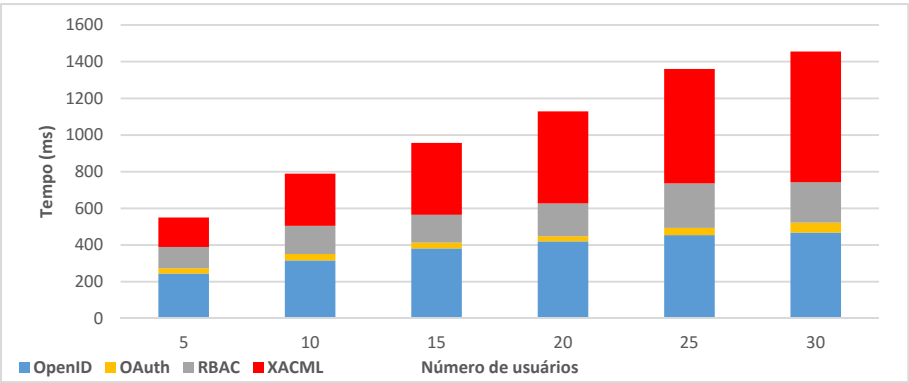

Figura 13. Discretização do tempo de resposta do teste (ii)

No segundo teste (ii), variamos o número de usuários de 1 a 30, cada um realizando10 requisições de acesso. Da mesma forma, a cada iteração dos testes escolhemos os usuários de maneira aleatória para a execução do cenário. Podemos notar uma pequena diferença no desempenho (Figura 12). O aumento do número de usuários impacta diretamente em todos os componentes da arquitetura (Figura 13).

Ambos os testes impactam significantemente no tempo de resposta do XACML. Este 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, então aumentando o número de usuários aumenta o número de requisições de acesso.

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

Todas as máquinas físicas possuem a mesma configuração, cada máquina possui um processador core i7 com 4 cores e 16 GB memória RAM. Foi utilizado o sistema operacional Ubuntu na versão 14.10 com o Java 1.7 e servidor tomcat 7 para executar os *web services*.

6. Conclusão

Neste trabalho foram utilizados o OIDC para prover a autenticação única entre os domínios e o OAuth para controle de admissão nos serviços, pois cada serviço tem visibilidade diferente. Além disto, foi utilizado o RBAC para controlar a sessão dos papéis, usuários e separações de tarefas e o XACML para realiza o controle de acesso baseado em atributos, com granularidade fina, a partir de papéis que são encarados atributos do usuário. Deste modo, alcançamos a meta de desenvolvimento de um modelo de controle de acesso baseado em atributos, que considera as diferentes semânticas de um papel, permitindo a ativação única de papéis (SRA) em múltiplos domínios.

A proposta apresentada mantém a autonomia do administrador de cada domínio, permitindo que o mesmo possa definir as permissões associados a cada papel. Assim, o administrador do domínio local pode definir políticas XACML referenciando papéis de outros domínios. Quando um usuário acessar um domínio remoto e requisitar acesso a um recurso, serão considerados os papéis ativos em seu domínio de origem. Se os papéis ativos no domínio de origem não forem conflitarem com os papeis do domínio remoto, o RBAC irá importar o papel em questão e o ativará no domínio remoto. Como resultado, um usuário acessa domínios remotos de maneira transparente, sem precisar ativar papeis que estão em uso em seu domino de origem, assim como acontece com SSO para autenticação.

O protótipo mostrou a viabilidade da proposta. Os testes relacionados a multi-domínios com SRA mostraram que o controle de acesso tem um desempenho semelhante a domínio único, com pequeno aumento no tempo de respostas. Considerando as vantagens qualitativas que o SRA fornece ao usuário e o bom desempenho, a proposta é avaliada como viável e promissora.

Agradecimento

Este projeto tem apoio parcial do CNPq, processos 310671/2012-4 e 404963/2013. O estudante Vilmar Abreu Junior gostaria de agradecer ao CNPq pela bolsa DTI, processo 381612/2014-7.

Referências

Ferraiolo, D. F., Kuhn, D. R. e Chandramouli, R. (2003). Role-Based Access Control. 1a.ed. Artech House Publishers.

Ferrini, R. e Bertino, E. (2009). Supporting RBAC with XACML+OWL. In Proceedings

of the 14th ACM symposium on Access control models and technologies. pp. 145-154.

Freudenthal, E., Pesin, T., Port, L., Keenan, E. e Karamcheti, V. (2002). dRBAC: Distributed Role-based Access Control for Dynamic Coalition Environments. Proceedings of 22nd International Conference on Distributed Computing Systems pp. 411-420.

Hardt, D. (2012). The OAuth 2.0 Authorization Framework. Internet Engineering Task Force (IETF), pp. 1–76.

Helil, N. e Rahman, K. (2010). RBAC Constraints Specification and Enforcement in Extended XACML. In Proceedings of 2nd International Conference on Multimedia Information Networking and Security. pp. 546-550.

Hu, V., Ferraiolo, D. e Kuhn, R. (2014). Guide to Attribute Based Access Control (ABAC) Definition and Considerations. NIST Special Publication 800-162. http://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.sp.800-162.pdf. Acesso 13/09/2015.

Joshi, J. B. D., Bhatti, R., Bertino, E. e Ghafoor, A. (2004). Access-Control Language for Multidomain Environments. IEEE Internet Computing, v. 8, n. 6, pp. 40–50.

Li, Q., Zhangt, X., Qing, S. e Xut, M. (2006). Supporting Ad-hoc Collaboration with Group-based RBAC Model. In Proceedings of the 2nd International Conference on Collaborative Computing. pp. 1-8.

Osborn, S., Sandhu, R. e 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. e Lockhart, H. (2013). eXtensible Access Control Markup Language (XACML) Version 3.0. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf. Acesso 13/09/2015.

Power, R. (2000). Tangled Web: Tales of Digital Crime from the Shadows of Cyberspace. Macmillan Press. pp. 1-8.

Sakimura, N., Bradley, J., Jones, M., Medeiros, B. De e Mortimore, C. (2014). OpenID Connect Core 1.0. http://openid.net/specs/openid-connect-core-1_0.html. Acesso 13/09/2015.

Sandhu, R. S. e Samarati, P. (1994). Access Control: Principles and Practice. In IEEE Communications Magazine, v. 32, n. 9, pp. 40-48.

Shafiq, B., Joshi, J. B. D., Bertino, E. e Ghafoor, A. (2005). Secure Interoperation in a Multidomain Environment Employing RBAC Policies. In IEEE Transactions on Knowledge and Data Engineering, v. 17, n. 11, pp. 1557-1577.

Sinnema, R. e Wilde, E. (2013). eXtensible Access Control Markup Language (XACML) XML Media Type. https://tools.ietf.org/html/rfc7061, Acesso em 13/09/2015.

Zhang, H., Li, Z. e Wu, W. (2012). Open Social and XACML based Group Authorization Framework. Proceedings of 2nd International Conference on Cloud and Green Computing and 2nd International Conference on Social Computing and Its Applications. pp. 655–659.

