Gerenciamento Distribuído de Políticas de Controle de Acesso em Ambiente Corporativo

Arlindo Luis Marcon Jr., Altair Olivo Santin, Maicon Stihler

Programa de Pós Graduação em Informática (PPGIa)
Pontifícia Universidade Católica do Paraná (PUCPR)
Caixa Postal 80215-901 - Curitiba - PR - Brasil

(almjr, santin, stihler)@ppgia.pucpr.br

Abstract. *The unified management of user rights and access control policies in corporations with many branches may seem paradoxical. The environmental heterogeneity and branch particularities require decentralized controls, while policy management aims for control unification. This paper proposes a distributed architecture for access control with unified management of corporate policies. Starting from rights specified on authorization certificates, corporate branches are autonomous to create and apply policies that will update the corporative repository. The provisioning keeps the policy synchronized on the local branch repository. The prototype using SPKI/SDSI and Web Services shows the proposal's feasibility.*

Resumo. *O gerenciamento unificado de direitos de usuários e das políticas de controle de acesso em corporações com muitas filiais pode parecer paradoxal. Pois, a heterogeneidade do ambiente e as particularidades de cada filial exigem a descentralização dos controles enquanto que a gestão das políticas sua unificação. Este artigo propõe uma arquitetura distribuída de controle de acesso com administração unificada das políticas corporativas. A partir de direitos codificados em certificados de autorização, as filiais têm autonomia para aplicar e gerar políticas que alimentarão o repositório corporativo. O provisionamento mantém as políticas sincronizadas no repositório local das filiais. O protótipo usando SPKI/SDSI e Serviços Web mostra a viabilidade da proposta.*

1. Introdução

Atualmente a troca de informações através da Internet se tornou uma necessidade, as empresas precisam interagir com um número cada vez maior de parceiros e consumidores. Em tal cenário, mecanismos tradicionais para gerenciar direitos e aplicar regras de controle de acesso não são mais suficientes. As entidades do sistema precisam confiar umas nas outras sem depender de controles centralizados para trocar informações sensíveis e compartilhar recursos. Interações deste tipo são complexas, visto que ambas as entidades, geralmente, fazem parte de diferentes domínios de segurança ou não possuem relacionamentos confiáveis pré-estabelecidos. Tais cenários exigem padronização na troca de informações, gerenciamento de direitos e estabelecimento de relações de confiança para suportar a heterogeneidade de plataformas.

A Arquitetura Orientada a Serviço (*Service-Oriented Architecture*, SOA) é um modelo que tem demonstrado rápida aceitação em ambientes corporativos. Os Serviços Web (*Web Services*, WS) são uma das implementações padrão para SOA. Porém, os

WS herdam mecanismos de controle de acesso das arquiteturas tradicionais de segurança. Por esta razão existe um forte acoplamento (dependência) entre *principals* (ex: entidades que interagem com os WS), provedores de serviço (servidores), mecanismos de segurança e a administração dos domínios corporativos.

A administração centralizada de políticas em nível corporativo facilita a imposição de regras em ambientes distribuídos. Porém, torna difícil para os administradores da corporação definir regras especificas para todos os recursos presentes nas filiais. Para superar esta limitação, delega-se o gerenciamento dos recursos para os administradores das filiais. Porém, neste caso, não se tem garantias de que as políticas corporativas serão corretamente aplicadas nos domínios locais.

Cooperação entre unidades da corporação para execução de projetos, por exemplo, exigem o acesso a recursos localizados em diferentes domínios (sítios) da mesma. Ou seja, membros de um projeto podem ter a necessidade de trabalhar temporariamente em diferentes filiais da mesma corporação. Cenários assim exigem um modelo de controle de acesso flexível para que cada provedor de serviços, nas filiais, aceite configurações de políticas pré-estabelecidas ou que as mesmas possam ser criadas dinamicamente.

Através de métodos de inicialização faz-se a configuração (provisionamento) das políticas nos repositórios das filiais, diminuindo-se a dependência entre as entidades de segurança em nível corporativo. Porém, tal abordagem precisa de mecanismos de sincronização para manter a consistência entre os repositórios distribuídos de políticas.

Uma maneira de facilitar a administração de sistemas distribuídos heterogêneos é através da adoção de WS, que fornecem padrões visando o baixo acoplamento entre as entidades que exercem o controle de acesso. Contribuindo assim para a administração unificada das políticas e suporte facilitado a transposição dos domínios de segurança.

As abordagens tradicionais de controle de acesso em WS, geralmente, usam trocas adicionais de mensagens para obter baixo acoplamento entre as entidades que exercem a segurança do ambiente. Porém, isto gera alto tráfego de mensagens na rede. Adicionalmente, certificados podem ser aplicados para gerenciar os direitos de acesso, reduzindo a dependência de uma entidade de autorização centralizada e, conseqüentemente o número de mensagens trocadas entre as entidades.

Este artigo propõe a administração unificada das políticas corporativas aplicadas em ambientes baseados em WS. O provisionamento de políticas fornece autonomia para as filiais e descentraliza a arquitetura de controle de acesso. A concessão dos direitos para os *principals* é feita com base em certificados de autorização recebidos da administração corporativa. O procedimento de concessão de direitos nas filiais atualizará o repositório de políticas corporativo, sem exigir qualquer intervenção dos administradores de segurança e sem violar as políticas corporativas. As políticas geradas nos domínios das filiais serão derivadas dos direitos codificados nos certificados de autorização.

O restante deste artigo é organizado da seguinte maneira. A Seção 2 descreve a SOA e padrões para WS. A Seção 3 introduz a arquitetura de controle de políticas baseada em servidor. A Seção 4 descreve o controle de acesso baseado em certificados. A Seção 5 apresenta a arquitetura da proposta. A Seção 6 apresenta o cenário utilizado na avaliação. A Seção 7 discute os trabalhos relacionados, e finalmente, a Seção 8 apresenta as conclusões.

2. Arquiteturas Orientadas a Serviço e Serviços Web

O principal objetivo da Arquitetura Orientada a Serviço (*Service Oriented Architecture*, SOA) é fornecer um modelo arquitetural onde serviços que executam determinada tarefa podem ser disponibilizados de modo padrão [OASIS 2006a].

Os Serviços Web (*Web Services*, WS) [W3C 2004] são um dos meios de implementação de SOA. O serviço pode ser criado em qualquer linguagem de programação, podendo ter a descrição de sua interface disponibilizada para os clientes através da sua publicação na UDDI (*Universal Description Discovery and Integration*) [OASIS 2004]. A descrição da interface deve seguir o formato da WSDL (*Web Services Description Language*) [W3C 2007a]. Em WS as entidades interagem usando mensagens SOAP (*Simple Object Access Protocol*) [W3C 2003] transportadas geralmente via HTTP e com serialização XML (*Extensible Markup Language*) [W3C 2006].

Os Serviços Web possuem várias especificações para diversos fins. Os principais padrões utilizados nesta proposta são brevemente apresentados a seguir.

A WS-Security [OASIS 2006b] prove segurança fim-a-fim (*end-to-end*) a mensagem SOAP, agregando padrões que fornecem suporte para *timestamp*, assinatura e cifragem (XML *Signature/Encryption*) e para a representação de credenciais de segurança (ex: asserções SAML). Uma terceira parte confiável (ex: *Security Token Service*, STS) pode garantir a autenticidade das credenciais ou mesmo fazer a troca das mesmas.

A WS-Trust [OASIS 2006c] faz uso de um modelo de mensagem pedido-resposta utilizado para transportar credenciais de modo seguro. Credenciais emitidas pelo STS podem ser usadas para se criar relacionamentos de confiança intra e inter-domínios.

A WS-Policy [W3C 2007b] descreve um modelo e uma sintaxe em XML para expressar requisitos, capacidades e restrições a WS em forma de políticas. A WS-SecurityPolicy [OASIS 2007a] complementa a WS-Policy definindo um conjunto de mecanismos (ex: algoritmos criptográficos) utilizados nas interações seguras em WS. Ambas definem o contexto de segurança e os requisitos para a interação segura.

O XKMS (*XML Key Management Service*) [W3C 2005] é um serviço que auxilia seus clientes a armazenar e recuperar suas chaves. O XKMS oculta de seus usuários os detalhes de gerenciamento de cada infra-estrutura (ex: X.509, SPKI/SDSI) adotando um esquema neutro de gestão de chaves.

A SPML (*Service Provisioning Markup Language*) [OASIS 2006d] define um conjunto de operações padrão para a configuração de objetos distribuídos (ex: configuração de políticas em um provedor de serviço).

A XACML *(eXtensible Access Control Markup Language)* [OASIS 2005a] define uma linguagem baseada em XML para a escrita de políticas de controle de acesso, e uma arquitetura de avaliação baseada em servidor utilizando as entidades: PEP (*Policy Enforcement Point*), PDP (*Policy Decision Point*), PIP (*Policy Information Point*), e PAP (*Policy Administration Point*). Na XACML, o Manipulador de Contexto intermedia a comunicação entre PEP e PDP fornecendo transparência a heterogeneidade quanto a tecnologia utilizada na implementação de ambos.

A SAML (*Security Assertion Markup Language*) [OASIS 2005b] pode ser usada para transportar informações em um formato padrão, sendo estas tomadas como confiáveis por parte dos outros Serviços Web, considerando-se neste caso a existência de relações pré-estabelecidas. A SAML define três tipos de expressões (autenticação, atributo

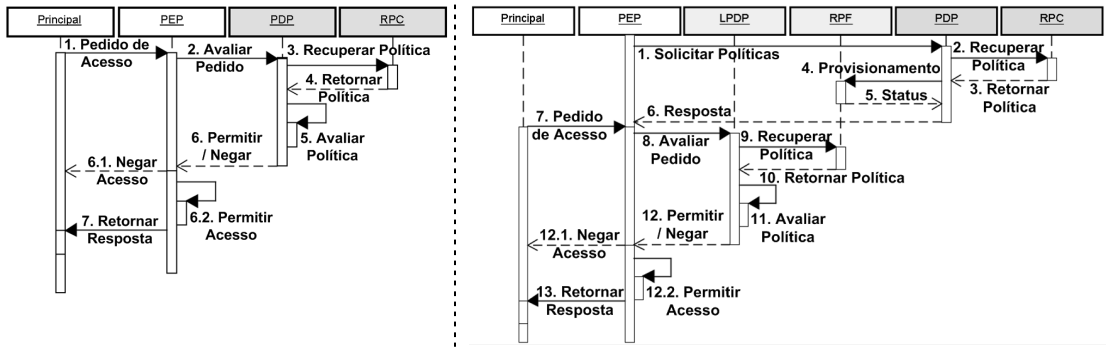
e autorização) que podem ser criadas por uma terceira parte confiável (ex: STS), inseridas em uma asserção e associadas a um *principal*.

3. Arquiteturas de Controle de Políticas Baseadas em Servidor

Arquiteturas deste tipo são compostas de duas entidades, o PDP – responsável pela decisão de autorização (permitir ou negar o acesso), e o PEP – responsável por executar a decisão do PDP (bloqueando ou liberando o acesso aos objetos). Dependendo de como tais entidades interagem, dois modelos distintos de controle de políticas podem ser utilizados, o *outsourcing* e o *provisioning*.

3.1. Modelo *Outsourcing*

No modo de operação *pull* ou modelo *outsourcing* [IETF 2001] toda requisição de um *principal* (evento *1*, Figura 1(a)) tentando acessar um recurso é interceptada pelo PEP e encaminhada para o PDP (evento *2*). O PDP toma a decisão de autorização (evento *5*) baseado na identidade do *principal*, no recurso solicitado e nas políticas recuperadas do Repositório de Políticas Corporativo (RPC, eventos *3* e *4*). Subseqüentemente, a decisão é enviada para o PEP (evento *6*) para que esse libere (eventos *6.2* e *7*) ou bloqueie a solicitação de acesso (evento *6.1*). Neste modelo, o PEP deve sempre consultar o PDP para ter uma política de autorização avaliada. Esta é a arquitetura de controle de acesso tradicional, comumente aplicada nos mecanismos de segurança, e adotada em WS.



(a) Modelo *Outsourcing* **(b) Modelo *Provisioning***

Figura 1: Fluxo de Controle de Política

3.2. Modelo *Provisioning*

No modo de operação *push* ou modelo *provisioning* [IETF 2001], durante a inicialização, o PEP envia uma mensagem para o PDP (evento *1*, Figura 1(b)) informando as suas capacidades e requisitando as políticas correspondentes. Todas as políticas recuperadas (eventos *2* e *3*) são armazenadas no Repositório de Políticas da Filial (RPF, eventos *4* e *5*) e disponibilizadas para serem avaliadas localmente pelo LPDP (*Local* PDP, evento *11*). De acordo com tal avaliação, a solicitação de acesso (evento *7*) é permitida (evento *12.2* e *13*) ou negada (evento *12.1*). Neste modelo, não há necessidade da troca de mensagens constante entre as entidades PEP e PDP. Porém, interações assíncronas entre estas podem acontecer a qualquer momento, sendo causadas por: inserção de novos recursos no PEP, atualizações no repositório (RPC do PDP), e ausência da política correspondente para uma avaliação enviada ao LPDP (ex: o *principal* de uma filial está tentando acessar um recurso pertencente ao domínio de outra filial).

4. Arquitetura de Controle de Políticas Baseada em Certificado

O SDSI (*Simple Distributed Security Infrastructure*) [Rivest e Lampson 1996] junto com o SPKI (*Simple Public Key Infrastructure*) [IETF 1999] oferece um modelo de autorização flexível, ideal para construir sistemas distribuídos seguros e escaláveis.

A Figura 2 mostra o exemplo de um certificado de autorização SPKI/SDSI codificado em *S-Expression* (linguagem para descrição de autorizações criada especificamente para este fim). As linhas 3 a 9 definem o emissor do certificado (*rights' grantor*), sendo que as linhas 7 a 9 definem a chave pública que o identifica. As linhas 10 e 11 contêm o *principal* beneficiado com a concessão dos direitos, o sujeito do certificado (*rights' grantee*). A linha 12 indica que o certificado pode ser delegado a terceiros. A linha 13 designa a autorização concedida pelo emissor. No caso, o *principal* tem o direito de leitura (*read*) em todos os arquivos do diretório *researcher* no servidor *www.corporate.com*. Na Figura 2, os campos contendo o período de validade e a assinatura do certificado foram omitidos para simplificar o entendimento.

Em SPKI/SDSI os *principals* são chaves públicas. Ou seja, a identificação de um principal coincide com a sua respectiva chave pública. É também empregado um modelo igualitário, onde qualquer *principal* pode emitir e assinar um certificado. Tal modelo é baseado na autorização ao invés da autenticação. Autorizações são concedidas através da delegação de direitos de um *principal* para outro, formando uma cadeia de certificados de autorização. O *principal* sujeito de um certificado de autorização se torna o *principal* emissor do próximo certificado da cadeia. A assinatura digital dos certificados assegura a autenticidade dos direitos delegados e a cadeia "certifica" sua validade.

```
1.  (cert
2.    (display "Any Readable Display Hint")
3.      (issuer
4.        (public-key
5.          (rsa-pkcs1-md5
6.             (e #11#)
7.                (n |ALNdAXftavTBG2zHV7BEV59gntNlxtJYqfWIi2kTcFIgIPSjKlHleyi9s
8.                5dDcQbVNMzjRjF+z8TrICEn9Msy0vXB00WYRtw/7aH2WAZx+x8erOWR+yn
9.                1CTRLS/68IWB6Wc1x8hiPycMbiICAbSYjHC/ghq2mwCZO7VQXJENzYr45|)))))
10.   (subject
11.     (object-hash (hash md5 |vN6ySKWE9K6T6cP9U5wntA==|)))
12.   (propagate)
13.   (tag (http://www.corporate.com/researcher (* set read)))
```

Figura 2: Certificado de Autorização SPKI/SDSI codificado em *S-Expression*

A cadeia pode ser reduzida a um único certificado, resumindo os direitos concedidos pela mesma. Em SPKI/SDSI, o controle de acesso pode ser aplicado sem o auxilio de um repositório de políticas, visto que a cadeia de certificados já concede os direitos para a última chave (*principal*) da mesma. Para facilitar o gerenciamento da identificação dos *principals*, o SPKI/SDSI emprega certificados de nome. Este associa um nome local a uma chave pública. Nomes locais aplicam-se somente ao espaço de nome de cada *principal*, podendo ser usados em substituição a chave pública para facilitar seu uso.

5. Modelo

O principal objetivo deste trabalho é prover a descentralização da arquitetura de controle de acesso, enquanto se mantém o controle unificado na gestão dos direitos de acesso. A descentralização na avaliação é alcançada com o modelo *provisioning*. O gerenciamento corporativo unificado é obtido usando a delegação de direitos através de certificados de autorização. As novas políticas são geradas localmente nas filiais com base

nos direitos concedidos pela cadeia de certificados de autorização. A política gerada no domínio de cada filial atualiza o repositório de políticas corporativo mantendo o controle unificado. Esta arquitetura de controle de política híbrida forma uma arquitetura de gerenciamento de políticas com baixo acoplamento entre as entidades da arquitetura.

A Figura 3 mostra a arquitetura de controle de política proposta. O PAP (*Policy Administration Point*) armazena todas as políticas da corporação, para todos os recursos disponibilizados pela mesma. O PAP é o único repositório que está sujeito a atualizações administrativas (evento *up1*). O LPAP (*Local PAP*, no domínio da filial) armazena uma copia local (evento *pp*) das políticas aplicadas aos recursos que o PEP controla. A arquitetura proposta suporta ambos os modelos de avaliação (*outsourcing* e *provisioning*). O modelo *outsourcing* é suportado apenas por razões de compatibilidade. O PEP aplica a decisão de autorização (evento *ac*) independentemente se a decisão foi efetuada na filial (LPDP, evento *ev1*) ou no nível corporativo (PDP, eventos *av* e *ev2*, respectivamente).

Quando um *principal* apresenta uma cadeia de certificados de autorização junto com uma solicitação de acesso (mensagem r_1, Figura 3) a um guardião (PEP), este a encaminha a um Gerenciador de Chaves e *Tokens* (GCT_2, evento *ce*) para que a mesma seja reduzida (ex: convertida) a um certificado único. A partir do certificado reduzido é gerada uma credencial de autorização especifica (evento *ce*) para o mecanismo que implementa o PEP. Adicionalmente, as informações atualizam automaticamente o repositório de política corporativo (PAP) como uma atividade administrativa (evento *up2*), gerando políticas específicas para cada *principal* e respectivos recursos a que este tem direito de acesso. O repositório da filial (LPAP) é atualizado sempre que necessário (eventos *rup* e *pp*, respectivamente) para manter sua consistência com o PAP corporativo.

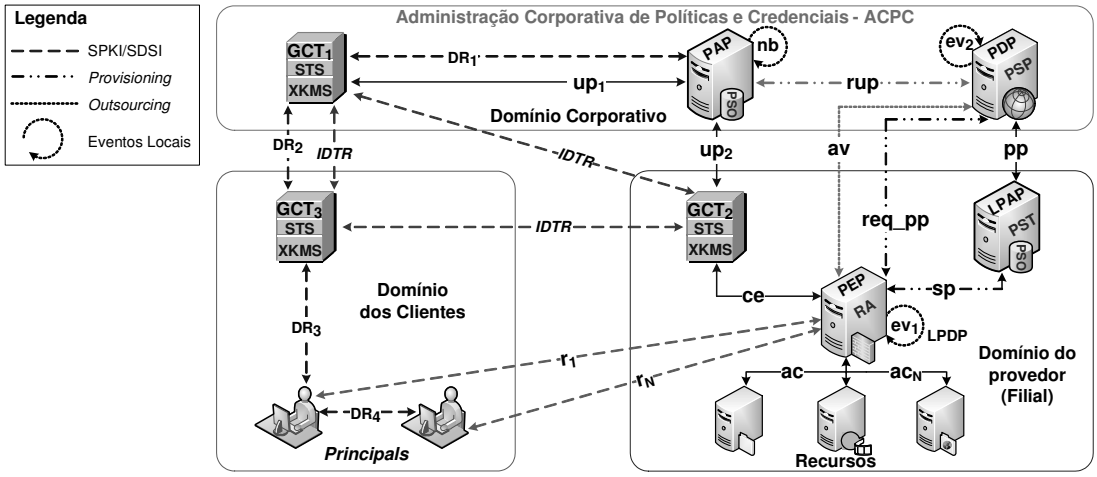


Figura 3: Interação Entre as Entidades do Protótipo

Se por algum motivo o LPAP não puder ser atualizado, gera-se uma pendência, através de uma notificação, no ambiente da Administração Corporativa de Políticas e Credenciais (ACPC, evento *nb*, Figura 3). Este esquema auxilia a descentralização da administração de políticas e ao mesmo tempo mantém as informações de autorização unificadas a partir do repositório ACPC.

Através do uso de certificados SPKI/SDSI é possível criar relações de confiança mútuas entre os GCTs. As relações de confiança inter-domínio (*IDTR*, Figura 3) se baseiam em certificados de nome emitidos mutuamente entre os GCTs. Estas relações de confiança (*IDTR*) têm propósitos administrativos, principalmente para permitir a concessão de direitos entre administradores (ex: GCT_1 para GCT_2 ou GCT_1 para GCT_3).

Para que o administrador do GCT_1 repasse direitos ao GCT_3 é necessário o estabelecimento da relação de confiança, *IDTR*. Esta é uma forma segura de um *principal* conhecer a chave pública do outro, dado que não existe uma autoridade certificadora em SPKI/SDSI. As relações de confiança são consideradas necessidades administrativas, pois raramente o administrador (GCT) faz uso dos direitos que lhe são repassados. Normalmente, estes direitos são parcialmente repassados (concedidos) aos *principals* pertencentes ao domínio do cliente (ex: GCT_3 para *principal*, evento DR_3, Figura 3).

A transposição dos domínios de autorização do cliente é facilmente obtida usando certificados de autorização SPKI/SDSI. Os certificados carregam todos os atributos necessários para a avaliação de uma solicitação de acesso. O provedor de serviço não necessita contatar nenhuma outra entidade (ex: serviço de atributos) para obter atributos adicionais sobre o *principal*. O provedor de serviço pode efetuar a decisão de autorização baseado unicamente na cadeia de certificados. O provisionamento de políticas e as cadeias de certificados favorecem o baixo acoplamento, transposição de domínios, a flexibilidade e a interoperabilidade entre as entidades que controlam as políticas.

6. Cenário

A seguir será considerado o uso da proposta em um ambiente corporativo com funcionários ocupando diferentes cargos em vários departamentos, distribuídos por muitas filiais. Além disto, é considerado que os sistemas de controle de acesso podem ser distintos, pois precisam atender as necessidades específicas de cada filial.

A Figura 4 mostra detalhes da arquitetura proposta, assim como algumas interações entre as entidades na composição do cenário. No cenário são usados Serviços Web, SPKI/SDSI e o modelo *provisioning* para implementar os controles de segurança.

No *bootstrap* do PEP é feito o provisionamento do repositório da filial empregando-se a SPML e o modelo *provisioning* (Seção 3.2). O PEP envia uma mensagem ao PDP (evento *req_pp*, Figura 4) solicitando as respectivas políticas. O PDP recupera as políticas do repositório (eventos *rup* e *sgp*) e as envia ao LPAP da filial (evento *pp*). A partir de então, as políticas estão disponíveis para avaliação no LPDP (evento *ev1*).

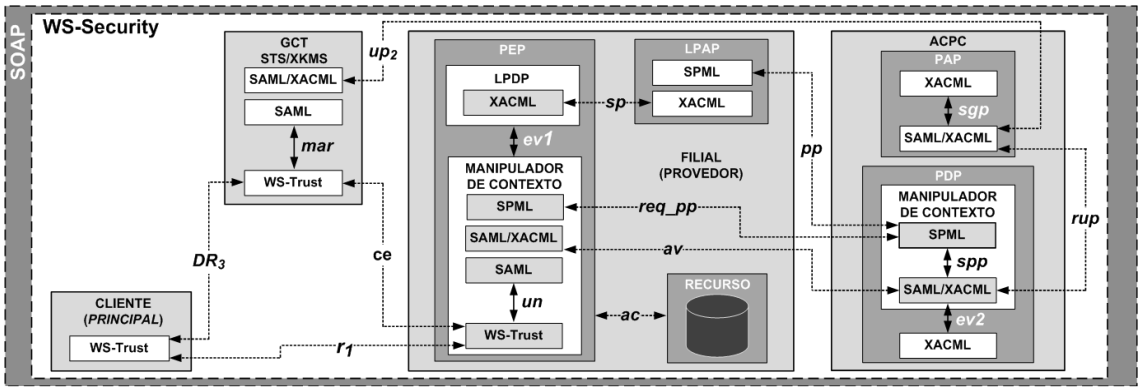


Figura 4: Protocolos e Especificações Aplicadas na Arquitetura do Protótipo

Na Figura 3, de acordo com a arquitetura SPML, o PDP representa o PSP (*Provisioning Service Provider*). O repositório interno ao PAP armazena todas as políticas, ou PSOs (*Provisioning Service Object*). O PEP representa a RA (*Requesting Authority*), e o LPAP representa o PST (*Provisioning Service Target*) que armazena as políticas provisionadas (PSOs). Um PDP (PSP) pode administrar diferentes PEPs (ex: cada PEP sendo responsável por uma filial da corporação).

Os relacionamentos de confiança entre os GCTs (associações *IDTR*, Figura 3) são baseados em certificados de nome SPKI/SDSI, fazendo-se a inclusão de um GCT no grupo do outro. Ou seja, cada GCT insere o outro GCT em seu grupo local e emite um certificado de nome, denotando que o mesmo é membro de seu grupo. A filiação ao grupo serve de base para assegurar que a entidade que enviou a mensagem é confiável.

Para obter acesso a um recurso o *principal* precisa ser inserido nas políticas do LPAP. Alternativamente, este pode obter os direitos exigidos através de uma cadeia de certificados de autorização. No evento DR_3 (Figura 3), o *principal* no domínio do cliente se autentica no GCT_3, enviando uma solicitação assinada contendo o recurso que o mesmo deseja acessar. O GCT_3 delega o direito exigido para o *principal* acessar o recurso caso já possua a cadeia. Caso contrário, obtém o direito solicitado pelo *principal* através do relacionamento *IDTR* (evento DR_2, Figura 3) do respectivo GCT_1.

Se o *principal* em DR_3 (Figura 3), por exemplo, for o administrador de um departamento o certificado de autorização pode ser delegável (os direitos contidos no certificado podem ser repassados aos funcionários do departamento – evento DR_4, Figura 3). Em caso contrário, o certificado conterá apenas os direitos para que o *principal* acesse o recurso desejado no provedor, ou seja, os direitos não podem ser delegados a terceiros.

6.1. Estudo de caso *

Considere um cenário que em determinado momento um *principal* faz uma solicitação de acesso a um recurso (evento *r1*, Figura 4). O Manipulador de Contexto, junto ao PEP, encaminhará esta solicitação ao LPDP (evento *ev1*) que consultará o seu repositório. Considerando que o LPDP não encontre nenhuma política permitindo o acesso ao recurso (evento *sp*), então este informa ao PEP que não pode avaliar tal pedido.

O PEP encaminha o pedido de avaliação ao PDP (evento *av*, Figura 4), pois o *principal* poderia estar em trânsito (ex: não pertence ao domínio desta filial). Neste caso, a avaliação é feita no modelo *outsourcing*: o PDP na ACPC decide (evento *ev2*) e o PEP no domínio do provedor executa a decisão. A proposta também suporta que a política indisponível seja solicitada ao PDP e avaliação ocorra localmente. As trocas de mensagem entre PDP e PAP (evento *rup*) são feitas de acordo com o *SAML 2.0 Profile of XACML* [OASIS 2007b].

Supondo que o PDP após consultar o PAP (eventos *rup* e *sgp*, Figura 4) não encontre a política para o recurso requisitado. Então, o PEP oferece uma alternativa para o *principal*: usando o protocolo *challenge-response* [NIST 1997] é devolvida ao *principal* uma mensagem informando quais direitos são necessários para acessar o recurso. A mensagem retornada transporta um documento WS-Policy. O *principal* por sua vez solicita ao GCT uma cadeia de certificados SPKI/SDSI que concedam os direitos requeridos. Obtendo a cadeia (evento DR_3) o *principal* envia a solicitação de acesso com a ca-

* Este trabalho é parcialmente financiado pelo CNPq (processo 550962/2007-7)

deia de certificados anexada (evento *r1*). O PEP encaminha a cadeia de certificados ao GCT (evento *ce*) para que seja feita a conversão da mesma em um certificado reduzido.

O GCT encaminha a cadeia ao XKMS, que reduz a cadeia de certificados e devolve um único certificado. Então, o GCT gera uma assertiva SAML (credencial nativa de Serviços Web) baseado nos direitos extraídos do certificado reduzido, e serializa a mesma em uma mensagem de resposta (evento *mar*, Figura 4). O GCT devolve a assertiva SAML (evento *ce*) e o PEP libera o acesso (evento *ac*) conforme as expressões (autorização ou atributo) contidas na assertiva deserializada (evento *un*).

Com base no certificado reduzido o GCT gera, também, uma política XACML encapsulada em SAML de acordo com [OASIS 2007b]. A política é enviada para o PAP (evento *up2*, Figura 4) com intuito de atualizar o repositório corporativo. O PAP armazena uma cópia da política recebida (evento *sgp*) e envia uma mensagem de atualização de política (evento *rup*) para o LPAP usando a infra-estrutura da SPML (eventos *spp* e *pp*, respectivamente). Este procedimento de atualização automática de políticas é equivalente a uma inserção de política efetuada por um administrador humano.

6.2. Tecnologias Utilizadas no Protótipo

O protótipo foi desenvolvido na linguagem de programação JAVA (JDK 1.6, *www.sun.com*) adotando-se a implementação de alguns projetos *opensource*: OpenSPML (*www.openspml.org*), OpenSAML (*www.opensaml.org*), e SUN XACML (*sunxacml.sourceforge.net*). Além destes, módulo Rampart integrado ao Axis2 (*ws.apache.org/axis2*) e servidor de aplicação TomCat (*tomcat.apache.org*). Nos repositórios de política foi utilizado o Banco de Dados Oracle Berkeley (*www.oracle.com*) e as bibliotecas para SPKI/SDSI foram obtidas do desenvolvimento de [Morcos 1998].

A Figura 5 mostra uma assertiva SAML (transportando as expressões de autenticação e atributo) gerada a partir de um certificado já reduzido (ex: Figura 2). A assertiva SAML contém o emissor (linha 4), o *principal* (linha 8), o contexto em que o *principal* foi autenticado (linha 16) e a operação que o mesmo pode executar (linha 25).

```
1. <saml:Assertion xmlns:saml="SAML:2.0:assertion" ID="ID_1"
2.  IssueInstant="2008-04-15T14:14:11.562Z" Version="2.0">
3.   <saml:Issuer xmlns:saml="SAML:2.0:assertion">
4.     http://10.32.1.53:9999/axis2/services/Master_Sts_Server_X
5.   </saml:Issuer>
6.   <saml:Subject xmlns:saml="SAML:2.0:assertion">
7.    <saml:NameID xmlns:saml="SAML:2.0:assertion" Format="SAML:2.0:nameid-format:
      SPKIPrincipal">
8.      vN6ySKWE9K6T6cP9U5wntA==
9.    </saml:NameID>
10.  </saml:Subject>
11.  <saml:Conditions xmlns:saml="SAML:2.0:assertion"
12.    NotBefore="2008-04-21T23:59:59.000Z" NotOnOrAfter="2008-07-18T00:00:00.000Z"/>
13.  <saml:AuthnStatement xmlns:saml="SAML:2.0:assertion" AuthnInstant="2008-04-
      15T14:14:11.781Z">
14.   <saml:AuthnContext xmlns:saml="SAML:2.0:assertion">
15.     <saml:AuthnContextDecl xmlns:saml="SAML:2.0:assertion">
16.       urn:oasis:names:tc:SAML:2.0:ac:classes:SPKI
17.     </saml:AuthnContextDecl>
18.   </saml:AuthnContext>
19.  </saml:AuthnStatement>
20.  <saml:AttributeStatement xmlns:saml="SAML:2.0:assertion">
21.   <saml:Attribute xmlns:saml="SAML:2.0:assertion"
      xmlns=http://localhost:8080/axis2/attributes
22.    Name="urn:foo:spki_attribute" NameFormat="SAML:2.0:attrname-format:uri">
23.     <saml:AttributeValuexmlns:saml="SAML:2.0:assertion"xmlns:xs="XMLSchema"
24.      xmlns:xsi="XMLSchema-instance" xsi:type="xs:string">
25.        read
26.     </saml:AttributeValue>
27.    </saml:Attribute>
28.  </saml:AttributeStatement>
29. </saml:Assertion>
```

Figura 5: Assertiva SAML gerada a partir da cadeia de certificados SPKI/SDSI reduzida

As declarações contendo o espaço de nomes referente à WS-Trust (especificação usada no transporte da assertiva SAML) foram omitidas, bem como o elemento da WS-Trust que transporta a URL (*Uniform Resource Locator*) do recurso requerido pelo *principal* (elemento <*AppliesTo*>).

Na geração da política XACML a partir do certificado SPKI/SDSI reduzido (Figura 2), a linha 2 da Figura 6 identifica comentários que o certificado SPKI/SDSI possa conter, no caso, um comentário padrão inserido em todas as novas políticas XACML. A linha 8 define o sujeito da política (*principal*) e a linha 18 descreve o recurso alvo.

Considerando que a cadeia de certificados é valida, a linha 26 explicita por padrão o valor "*Permit*", pois em caso contrário a política não seria gerada. A linha 34 define a operação autorizada. Como haverá apenas uma regra de política XACML (linha 26 a 41) gerada a partir da cadeia de certificados reduzida, foi adotado o algoritmo de combinação de regras *"first-applicable"* (linha 1) para avaliação de políticas.

Certificados de autorização SPKI/SDSI transportando mais de um direito de acesso para o mesmo *principal* e referenciando o mesmo recurso, podem gerar regras de política XACML com mais de uma *tag* <*Action*> (linhas 31 a 38, Figura 6).

Na geração das políticas XACML a partir da cadeia de certificados SPKI/SDSI reduzida (ex: Figura 2) foi assumido que o que não foi explicitamente permitido é por padrão negado (linha 42, Figura 6). Este procedimento evita que o retorno da avaliação seja diferente, explicitamente, de permitido (linha 26) ou negado. Tendo em vista que a política somente será avaliada para o *principal* correspondente (linha 8).

```
1. <Policy PolicyId="GeneratedReadPolicy" RuleCombiningAlgId="first-applicable">
2. <Description>This statement applies to this principal at organization.com
   </Description>
3.  <Target>
4.    <Subjects>
5.      <Subject>
6.        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
7.          <AttributeValue DataType="string">
8.            vN6ySKWE9K6T6cP9U5wntA==
9.          </AttributeValue>
10.         <SubjectAttributeDesignator AttributeId="subject-id" DataType="string"/>
11.       </SubjectMatch>
12.     </Subject>
13.   </Subjects>
14.   <Resources>
15.     <Resource>
16.       <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
17.         <AttributeValue DataType="anyURI">
18.           http://www.organization.com/researcher
19.         </AttributeValue>
20.       <ResourceAttributeDesignator AttributeId="resource-id" DataType="anyURI"/>
21.      </ResourceMatch>
22.     </Resource>
23.   </Resources>
24.   <Actions> <AnyAction/> </Actions>
25.  </Target>
26.  <Rule RuleId="ReadRule" Effect="Permit">
27.    <Target>
28.      <Subjects> <AnySubject/> </Subjects>
29.      <Resources> <AnyResource/> </Resources>
30.      <Actions>
31.        <Action>
32.          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
33.            <AttributeValue DataType="string">
34.              read
35.           </AttributeValue>
36.           <ActionAttributeDesignator AttributeId="action-id" DataType="string"/>
37.         </ActionMatch>
38.       </Action>
39.     </Actions>
40.   </Target>
41.  </Rule>
42.  <Rule RuleId="FinalRule" Effect="Deny"/>
43. </Policy>
```

Figura 6: Política XACML gerada a partir da cadeia de certificados SPKI/SDSI reduzida

6.3. Avaliação do Protótipo

A Figura 7 compara os tempos de resposta para o *principal* obter acesso a um recurso (com e sem as *features* de segurança). A Figura 8 mostra o percentual de tempo que o LPDP e o PDP gastam para avaliar as políticas de controle de acesso, o tempo que o GCT_2 gasta para validar a cadeia de certificados, e o tempo que o PEP gasta para aplicar o resultado retornado pelo LPDP/PDP ou GCT_2, liberando ou bloqueando o acesso de um *principal*.

Os percentuais mostrados na Figura 8 estão baseados nos tempos mostrados na Figura 7, ou seja, em relação ao tempo total de cada requisição obteve-se a percentagem individual que cada entidade gastou no atendimento a uma requisição, de acordo com cada modelo.

Observando a Figura 8 é possível perceber que o uso de certificados exige mais tempo de processamento do que a avaliação baseada nos modelos *provisioning* ou *outsourcing*. Porém, este tempo adicional é gasto somente na primeira solicitação de acesso do *principal*, quando a cadeia precisa ser validada (evento *ce*, Figura 4). Após isso, a política correspondente será criada e provisionada (configurada na filial) e a avaliação ocorrera no modelo *provisioning*. Além disto, as políticas de granularidade fina neste esquema foram criadas automaticamente – sem a intervenção do administrador.

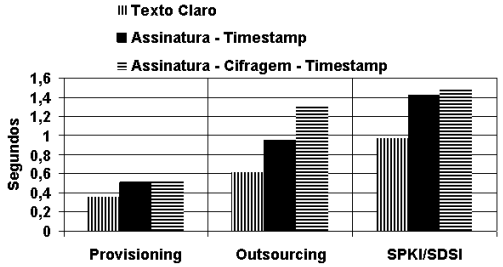


Figura 7: Tempos gastos com e sem segurança em cada modelo

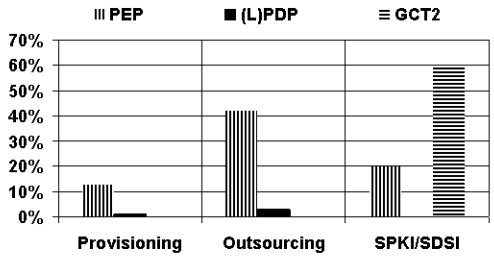


Figura 8: Percentuais de tempo gastos em cada entidade/modelo

Na Figura 8, considerando-se a avaliação de acesso no modelo *provisioning*, o WS PEP é a entidade do modelo que mais gasta tempo manipulando mensagens XML e fazendo a aplicação da decisão do LPDP, sendo que esta é a entidade que menos consome tempo na arquitetura proposta. No modelo *outsourcing* (Figura 8) toda requisição enviada pelo *principal* para o WS PEP (evento *r1*, Figura 4) é encaminhada e avaliada pelo WS PDP (eventos *av* e *ev2*, respectivamente), fazendo com que o computador que hospeda o WS PEP fique aguardando a resposta do PDP. O *principal*, por sua vez, acaba tendo que esperar esse tempo adicional gasto na troca de mensagens entre PEP e PDP.

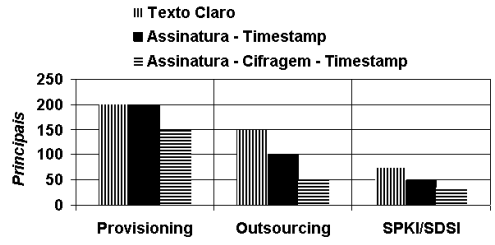


Figura 9: Número de *principals* atendidos simultaneamente

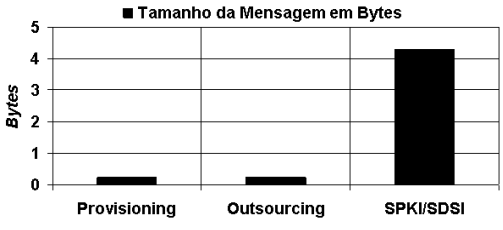


Figura 10: Tamanho das mensagens considerando os três modelos

Com o uso de certificados SPKI/SDSI (Figura 8) o PEP não se tornou um ponto crítico do sistema, sendo que o cliente e o PEP somente herdaram o tempo gasto pelo GCT_2 (STS) em uma primeira avaliação de acesso. O tempo gasto pelo GCT_2 (STS) se deve a complexidade para validar todos os parâmetros de uma cadeia de certificados SPKI/SDSI, gerar a nova política XACML e a credencial de acesso SAML.

O modelo *outsourcing* além de não possuir baixo acoplamento (devido à dependência do PDP) é computacionalmente mais custoso (Figura 8) e dificulta a escalabilidade dos WS. Enquanto que o modelo baseado em certificados SPKI/SDSI e *provisioning* é fracamente acoplado, dado a autonomia de operação e baixa necessidade de troca de mensagens entre as entidades da arquitetura. Além disto, esta combinação mostra que é possível diminuir o custo humano e computacional de gestão das políticas e facilitar a escalabilidade dos WS.

A Figura 9 mostra a quantidade de *principals* requisitando acesso que cada entidade suportou atender simultaneamente, com e sem o uso de segurança. Considerando o desempenho no caso de uso com segurança é possível observar que se a confidencialidade (cifragem) da mensagem não for importante, ganha-se segurança com a assinatura digital (integridade, autenticidade e não-repúdio) e com o *timestamp* (proteção contra ataque de mensagens antigas) sem grande impacto no desempenho do sistema.

A Figura 10 mostra o tamanho da mensagem de requisição enviada do *principal* para o PEP (evento r_1, Figura 4) em cada um dos modelos. A mensagem trocada entre PEP e PDP no modelo *outsourcing* (evento *av*) tem um tamanho de 976 *Bytes*. A requisição com SPKI/SDSI transporta a cadeia de certificados composta de 2 certificados e 3 chaves, cada novo certificado inserido na cadeia adiciona 1,78 *KBytes* a mensagem.

7. Trabalhos Relacionados

A arquitetura sugerida em [Mello e Fraga 2005] propõe o uso de um mediador confiável (um domínio intermediário) responsável por tratar tecnologias de segurança diferentes e que não operam entre si, permitindo a interação entre um cliente num domínio SPKI/SDSI e um servidor que usa certificados X.509. Todo o processo de avaliação necessita que credenciais SPKI/SDSI ou X.509 sejam trocados por asserções SAML para ter um formato comum e um esquema funcional.

O uso do mediador e do modelo *outsourcing* implica em *overhead* devido ao aumento do número de mensagens trocadas entre as entidades e aumento no tempo para se conseguir a credencial de acesso requerida (ex: asserção SAML) em cada avaliação de autorização. A dependência de entidades do domínio do solicitante (ex: serviço de atributos) são contrários aos princípios de baixo acoplamento desejados em Serviços Web.

O trabalho de [Camargo 2006] faz uso de asserções de autenticação e atributos SAML para autorizar os clientes fora de seus domínios de segurança. Os clientes são autenticados localmente, mas visam à autorização de maneira distribuída. Por exemplo, caso o PDP no provedor de serviço receba uma assertiva de autenticação SAML do PEP e a política de proteção do provedor de serviço exija um certificado X.509 ou SPKI/SDSI, o PDP deve interagir com o STS e solicitar a tradução da credencial de segurança apresentada pelo cliente.

A abordagem baseada no modelo *outsourcing* encarrega o PDP de entrar em contato com o serviço tradutor de credenciais (STS) para que este recupere atributos sobre o cliente que esta solicitando acesso. A comunicação inter-domínios adotada nesta pro-

posta aumenta consideravelmente o acoplamento e o tempo de resposta para o cliente obter acesso ao recurso.

Até o presente momento, nenhum outro trabalho disponível na literatura técnica aborda os aspectos de segurança presentes em nossa proposta, principalmente quanto ao uso do SPKI/SDSI como fonte de dados para geração de políticas de autorização de forma automática. Essa abordagem libera o administrador de qualquer ônus relativo à escrita de políticas de granularidade fina e evita violações nas políticas corporativas devido à derivação compulsória de direitos impostas pelo emprego do SPKI/SDSI. Além disto, nenhum trabalho da literatura explora o modelo *provisioning* – mais adequado aos requisitos de baixo acoplamento de SOA/Serviços Web.

8. Conclusão

Este trabalho apresentou uma proposta de administração de políticas que oferece gestão unificada, suportando avaliação, criação, e execução distribuída de políticas. Além disto, a escrita das políticas é feita sob demanda, e de maneira automática e segura.

O esquema proposto oferece uma alternativa para o cliente conseguir os direitos requeridos quando o mesmo não consta no repositório de políticas corporativas. Contrastando com a maneira tradicional de controle acesso que no caso simplesmente negaria o acesso. O administrador do domínio cliente pode emitir um certificado de autorização para o *principal* obter os direitos requeridos e ser incluído automaticamente no repositório do PAP. O administrador local pode facilmente conceder direitos de acesso, pois possui relações mútuas de confiança com os provedores que seus clientes acessam.

O SPKI/SDSI, que é independente de tecnologia subjacente, fornece suporte ao esquema de transporte de direitos, e estabelecimento de relações de confiança que podem facilmente transpor os domínios de segurança das filiais de maneira fácil e segura. Adicionalmente, os Serviços Web oferecem uma infra-estrutura com protocolos e serviços de segurança que facilitam a interoperabilidade em ambientes heterogêneos.

O modelo proposto pode operar automaticamente no controle de políticas no modelo *provisioning* ou *outsourcing* (por compatibilidade com os WS em uso). Esta característica não é encontrada em nenhum outro trabalho relacionado. Porém, em geral, a proposta opera no modelo *provisioning*, pois gera-se muito menos troca de mensagens, o que dá mais autonomia local as filiais e causa menos sobrecarga da rede.

A abordagem proposta é compatível com o modelo baseado em servidores de Serviços Web. Porém, não depende da infra-estrutura baseada em servidores de autenticação e autorização para transpor domínios de segurança, contribuindo para o baixo acoplamento em SOA.

O modelo proposto tolera a indisponibilidade do repositório corporativo de políticas, pois há uma cópia local das políticas aplicáveis ao domínio de cada provedor/filial. Além disto, se as políticas locais não contêm regras para o *principal*, o mesmo pode obter os direitos requeridos para o acesso a partir do administrador do domínio a que está vinculado. Assim, a cadeia de certificados SPKI/SDSI pode conceder direitos a *principals* sem que a ACPC esteja ativa.

O esquema de segurança proposto para SOA (Serviços Web) permite baixo acoplamento, pois aplica provisionamento de políticas e certificados SPKI/SDSI. O protótipo mostra a viabilidade da proposta e o modelo pode ser estendido para cenários mais abrangentes que o mostrado neste trabalho.

Referências

OASIS. (2006a). Reference Model for Service Oriented Architecture. http://www.oasis-open.org/specs/index.php#soa-rmv1.0.

W3C. (2004). Web Services Architecture. http://www.w3.org/TR/ws-arch/.

OASIS. (2004). Universal Description Discovery & Integration - UDDI. http://uddi.org/pubs/uddi_v3.htm.

W3C. (2007a). Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. http://www.w3.org/TR/wsdl20/.

W3C. (2003). SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). http://www.w3.org/TR/soap12-part1/.

W3C. (2006). Extensible Markup Language - XML. http://www.w3.org/TR/xml11/.

OASIS. (2006b). Web Services Security: SOAP Message Security 1.1 - WS-Security. http://www.oasis-open.org/specs/index.php#wssv1.1.

OASIS. (2006c). WS-Trust 1.3. http://www.oasis-open.org/specs/index.php #wstrustv1.3.

W3C. (2007b). Web Services Policy. http://www.w3.org/TR/ws-policy/.

OASIS. (2007a). WS-SecurityPolicy. http://www.oasis-open.org/specs/index.php #wssecpolv1.2.

W3C. (2005). XML Key Management Specification - XKMS. http://www.w3.org/TR/xkms2/.

OASIS. (2006d). Service Provisioning Markup Language - SPML. http://www.oasis-open.org/specs/index.php#spmlv2.0.

OASIS. (2005a). eXtensible Access Control Markup Language - XACML. http://www.oasis-open.org/specs/index.php#xacmlv2.0.

OASIS. (2005b). Assertions and Protocols for the OASIS Security Assertion Markup Language - SAML. http://www.oasis-open.org/specs/index.php#samlv2.0.

IETF. (2001). Terminology for Policy-Based Management. http://www.ietf.org/rfc/rfc3198.txt.

Rivest, R. L. e B. Lampson. (1996). SDSI - A Simple Distributed Security Infrastructure. Massachusetts Institute of Technology.

IETF. (1999). SPKI Certificate Theory. http://www.ietf.org/rfc/rfc2693.txt.

OASIS. (2007b). SAML 2.0 profile of XACML v2.0. http://www.oasis-open.org/specs/index.php#samlv2.0.

NIST. (1997). Entity Authentication Using Public Key Cryptography. FIPS PUB 196. http://csrc.nist.gov/publications/fips/fips196/fips196.pdf.

Morcos, A. (1998). A Java Implementation of Simple Distributed Security Infrastructure. (Master Dissertation). EECS, Massachusetts Institute of Technology.

Mello, E. R. e J. S. Fraga. (2005). Mediation of Trust across Web Services. IEEE ICWS'05.

Camargo, E. T. (2006). Transposição de Autenticação em Arquiteturas Orientadas a Serviço Através de Identidades Federadas. (Dissertação de Mestrado). PGEEL, UFSC.