ARLINDO LUIS MARCON JUNIOR

GERENCIAMENTO DE POLÍTICAS DE CONTROLE DE ACESSO FRACAMENTE ACOPLADO PARA SERVIÇOS WEB

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

CURITIBA

2008

ARLINDO LUIS MARCON JUNIOR

GERENCIAMENTO DE POLÍTICAS DE CONTROLE DE ACESSO FRACAMENTE ACOPLADO PARA SERVIÇOS WEB

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

Área de Concentração: *Ciência da Computação*

Orientador: Prof. Dr. Altair Olivo Santin

CURITIBA

2008

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central

M321g
2008

Marcon Junior, Arlindo Luis
Gerenciamento de políticas de controle de acesso fracamente acoplado para serviços Web / Arlindo Luis Marcon Junior ; orientador, Altair Olivo Santin. – 2008.
xvii, 128 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2008
Bibliografia: f. 99-103

1. Redes de computação – Medidas de segurança. 2. Serviços da Web. 3. Computadores – Controle de acesso. 4. Redes de computação – Gerência. I. Santin, Altair Olivo. II. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. III. Título.

CDD 20. ed. – 005.8

Pontifícia Universidade Católica do Paraná
Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Informática

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

DEFESA DE DISSERTAÇÃO Nº 10/2008

Aos 28 dias do mês de julho de 2008 realizou-se a sessão pública de Defesa da Dissertação **"Gerenciamento de Políticas de Controle de Acesso Fracamente Acoplado para Serviços Web"**, apresentada pelo aluno **Arlindo Luiz Marcon 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)*	Aprovado *(aprov/reprov.)*
Prof. Dr. Luiz Augusto de Paula Lima Junior PUCPR		aprovado
Prof. Dr. Rafael Rodrigues Obelheiro UDESC	Rafael Obelheiro	aprovado

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.

Prof. Dr. Alessandro Lameiras Koerich
Diretor do Programa de Pós-Graduação em Informática

Programa de Pós-Graduação em Informática
PPGIa
PUCPR

A Deus Pai todo poderoso,
a Jesus Cristo seu único filho, nosso Senhor,
a meus pais, Arlindo e Lidia, pela confiança e apoio.

Agradecimentos

Agradeço a meu orientador, Dr. Altair Olivo Santin, pela paciência, confiança, e persistência.

Aos membros da banca, Dr. Luiz e Dr. Rafael, obrigado pelas contribuições.

As professoras, Dra. Carla e Dra. Michelle, pelo apoio quando as procurei pedindo por informações sobre o mestrado.

Ao professor, Dr. Lau, por ter me auxiliado a entrar no mestrado da PUCPR.

Aos meus professores de graduação, Alexandre, Cirlei e Karen, o apoio deles foi fundamental.

As amizades feitas durante estes anos, Altair, Cheila, Edson, Maicon, Marcelo, Regivaldo, etc., obrigado pelo companheirismo e pelas grandes idéias.

A meus amigos, que mesmo estando longe, me apoiaram e se lembraram de mim.

A minha tia, Hilda, sempre me ligando, e se preocupando comigo.

E principalmente, agradecer a meu pai e minha mãe, Arlindo Luiz Marcon e Lidia Ortigara Marcon, por terem me apoiado nesta aventura do conhecimento, sem eles, nada disso teria sido possível.

Resumo

Atualmente, a troca de informações através da Internet se tornou não somente possível, mas uma necessidade, as empresas precisam interagir com um número cada vez maior de parceiros de negócio e consumidores. Em tais cenários, métodos tradicionais utilizados para gerenciar direitos, identidades, e impor regras de controle de acesso não são mais suficientes. As entidades de controle precisam confiar umas nas outras para trocar informações sensíveis e compartilhar recursos. Interações deste tipo são complexas, visto que as entidades, geralmente, fazem parte de diferentes domínios de segurança ou não possuem relacionamentos confiáveis pré-estabelecidos. Cenários como estes exigem a padronização da troca de informações, gerenciamento de direitos e estabelecimento de confiança para lidar com a heterogeneidade das plataformas. Um dos objetivos do presente trabalho é fazer uso de certificados de autorização para transportar os atributos que serão utilizados em mecanismos de autenticação e autorização distribuídos. Unindo-se os certificados de autorização a uma arquitetura de controle de políticas baseada no modelo de configuração (provisionamento), se diminui a dependência existente entre as entidades que aplicam as políticas de controle de acesso no ambiente distribuído (corporativo, por exemplo). O desenvolvimento do modelo proposto permite a gerência descentralizada das políticas de controle de acesso dispersas pelo ambiente distribuído, possibilitando a geração automatizada de políticas de granularidade fina no ambiente local das filiais da corporação. A abordagem utilizada não permite que as regras de políticas sejam violadas, pois os direitos atribuídos localmente, nas filiais, são derivados dos direitos definidos em nível corporativo. As permissões de acesso e a geração automatizada de políticas locais são derivadas de certificados de autorização providos por uma infra-estrutura de chave pública desprovida de Autoridade Certificadora. Os certificados de autorização fornecem informações suficientes para gerar uma credencial de autorização e uma nova política. A credencial de autorização é utilizada pelo guardião do recurso para liberar, ou bloquear o acesso do sujeito ao recurso oferecido pela filial. Toda nova política é enviada a um repositório de política corporativo, utilizado para preservar a visão administrativa unificada do ambiente distribuído de controle de acesso. Uma cópia da nova política e enviada para a respectiva filial onde está instanciado o mecanismo de controle de acesso, favorecendo assim a autonomia local das filiais. Esta atualização no repositório da

filial mantém o mesmo consistente com o repositório de políticas corporativo. A abordagem proposta concede autonomia às filiais no processo de avaliação de políticas, favorecendo a interoperabilidade, flexibilidade, e escalabilidade. Propriedades estas almejadas pela Arquitetura Orientada a Serviço e pelos Serviços Web. A presente proposta diminui significativamente a intervenção do administrador corporativo, sem que o mesmo perca o controle administrativo sobre as políticas de controle de acesso aplicadas no ambiente distribuído.

Palavras-Chave: Controle de Acesso, Provisionamento de Políticas, Certificados de Autorização, Serviços Web.

Abstract

Nowadays, the information exchange through the Internet has become not only possible, it is a need. The enterprises need to interact with an ever growing number of partners and consumers. In such scenarios, traditional methods for managing rights and identities, and enforcement of access control rules are not enough. The controlling entities need to trust in each other to exchange sensitive information and share resources. Interactions like this are complex, due to the fact that both entities, commonly, belong to different security domains, or don't have previously established trust relationships. Scenarios like this demand a standardization of information exchange, rights management and trust establishment to support platform heterogeneity. One of this work's goals is to make use authorization certificates to convey authentication and authorization attributes that will be used by distributed authentication and authorization mechanisms. By incorporating authorization certificates with an access control architecture based on the provisioning model, we can reduce the dependence present amongst the entities that enforce the access control policies in a distributed environment (e.g.: corporate environment). The implementation of the proposed model allows for the decentralized access control policies management in distributed environment, enabling automated generation of fined grained policies in the local environment of each corporation branch. The employed approach does not allow violation of the policies' rules, as the rights in each branch are derived from the rights defined in the corporate level. Access rights and automated generation of local policies derive from authorization certificates provided by a serverless public key infrastructure. The authorization certificates provide enough data to generate an authorization credential and a new policy. The authorization credential is used by the resource guardian to allow or deny the subject's access to the resource provided by the local branch. Every new policy is sent the corporative policy repository, employed to preserve a keep an unified administrative view of the distributed access control environment. One copy of the new policy is sent to the enterprise branch where the access control mechanism is instantiated, favoring local autonomy on the branches. This updates on the branch's repository keeps same consistent with the corporation's policy. The proposed approach gives autonomy to the branches on the policies evaluation process, favoring interoperability, flexibility, and scalability. Properties which are aimed by Service

Oriented Architectures and Web Services. The present proposal significantly reduces the intervention of the corporative administrator, without him loosing administrative control over the access control policies applied to the distributed environment.

Keywords: Access Control, Policy Provisioning, Authorization Certificates, Web Services.

Lista de Figuras

Lista de Abreviaturas

Abreviatura	Acrônimo por extenso	Significado
ACL	*Access Control List*	Lista de Controle de Acesso
AOS	---	Arquitetura Orientada a Serviço
API	*Application Program Interface*	Interface para Programação de Aplicativos
AS	*Authentication Service*	Serviço de Autenticação
ASCII	*American Standard Code for Information Interchange*	Código Padrão Americano para Intercâmbio de Informações
CA	*Certificate Authority*	Autoridade Certificadora
COPS	*Common Open Policy Service*	Serviço de Política Aberto
COPS-PR	*COPS for Policy Provisioning*	COPS para Provisionamento de Política
CORBA	*Common Object Request Broker Architecture*	Arquitetura Comum para Agente de Requisição de Objetos
CRC	*Certificate Result Certificates*	Certificado Resultado dos Certificados
CRL	*Certificate Revocation Lists*	Lista de Revogação de Certificados
DTD	*Document Type Definition*	Definição do Tipo do Documento
HTTP	*Hypertext Transport Protocol*	Protocolo de Transporte de Hipertexto
ICP	---	Infra-Estrutura de Chave Pública
IdP	*Identity Provider*	Provedor de Identidade
IETF	*Internet Engineering Task Force*	Força Tarefa de Engenharia Internet
IP	*Internet Protocol*	Protocolo Internet
IPSec	*Internet Protocol Security*	Protocolo Internet Seguro
JDK	*Java Development Kit*	Ferramentas para Desenvolvimento de Programas na Linguagem Java

OASIS	*Organization for the Advancement of Structured Information Standards*	Organização para o Avanço de Padrões de Informação Estruturados
PAP	*Policy Administration Point*	Ponto de Administração de Política
PCIM	*Policy Core Information Model*	Modelo Central de Informação de Política
PDP	*Policy Decision Point*	Ponto de Decisão de Políticas
PEP	*Policy Enforcement Point*	Ponto de Aplicação de Políticas
PIB	*Policy Information Base*	Base de Informação de Política
PIP	*Policy Information Point*	Ponto de Informação de Políticas
PKI	*Public Key Infrastructure*	Infra-estrutura de Chave Pública
SAML	*Security Assertion Markup Language*	Linguagem de Marcação para Asserções de Segurança
SDSI	*Simple Distributed Security Infrastructure*	Infra-estrutura de Segurança Distribuída Simples
SMTP	*Simple Mail Transport Protocol*	Protocolo Simples de Transporte de Correspondência
SOA	*Service Oriented Architecture*	Arquitetura Orientada a Serviço
SOAP	*Simple Object Access Protocol*	Protocolo Simples de Acesso a Objetos
SP	*Service Provider*	Provedor de Serviço
SPKI	*Simple Public-Key Infrastructure*	Infra-estrutura de Chave Pública Simples
SPML	*Service Provisioning Markup Language*	Linguagem de Marcação para Configuração de Serviços
SSL	*Secure Sockets Layer*	Camada de Soquetes Segura
SSO	*Single Sign-On*	Inscrição Única
STS	*Security Token Service*	Serviço de Credenciais Seguras
TCP	*Transmission Control Protocol*	Protocolo de Controle de Transmissão
TGS	*Ticket Granting Service*	Serviço de Concessão de Bilhete
TGT	*Ticket Granting Ticket*	Bilhete para Concessão de Bilhete

TLS	*Transport Layer Security*	Segurança na Camada de Transporte
TTP	*Trusted Third Party*	Terceira Parte Confiável
UBR	*UDDI Business Registry*	Registro de Negócios UDDI
UDDI	*Universal Description Discovery and Integration*	Descrição, Descoberta e Integração Universais
URI	*Uniform Resource Identifiers*	Identificador de Recursos Uniforme
URL	*Uniform Resource Locators*	Localizador de Recursos Uniforme
UTC	*Universal Time Coordinated*	Tempo Universal Coordenado
VPN	*Virtual Private Network*	Rede Privada Virtual
W3C	*World Wide Web Consortium*	Consórcio da World Wide Web
WS	*Web Services*	Serviços Web
WSDL	*Web Services Description Language*	Linguagem de Descrição de Serviços Web
WS-I	*Web Services Interoperability Organization*	Organização para Interoperabilidade dos Serviços Web
XACML	*eXtensible Access Control Markup Language*	Linguagem de Marcação Extensível para Controle de Acesso
X-KISS	*XML Key Information Service Specification*	Especificação XML para Serviço de Informação de Chaves
XKMS	*XML Key Management Specification*	Especificação para Gerenciamento de Chaves XML
X-KRSS	*XML Key Registration Service Specification*	Especificação XML para Serviço de Registro de Chaves
XML	*Extensible Markup Language*	Linguagem de Marcação Extensível

Sumário

Capítulo 1

Introdução

O uso da Internet tem se intensificado significativamente, se tornando uma das necessidades nos dias atuais. Usuários têm utilizado cada vez mais a Internet para realizar tarefas cotidianas, como por exemplo, efetuar transações bancárias em Bancos via Internet (*i.e.*: Internet *Banking*), realizar a compra e venda *online* em *sites* que comercializam produtos (comodidade), manter contato com outras pessoas fazendo uso de programas de mensagem instantânea, entre outras atividades diárias.

O aumento da utilidade e da usabilidade da Internet tem sido referenciado como *Web 2.0* [1]. A rede de informações, seja em formato texto (*hipertexto*) ou multimídia (*e.g.*: *YouTube*), revitalizou a Internet em termos de utilidade e usabilidade. A Arquitetura Orientada a Serviço (AOS) é parte integrante da *Web 2.0*, tendo nos Serviços Web (*Web Services* - WS) uma de suas implementações de maior aceitação e que mais tem se difundido.

Neste capítulo explanaremos a contextualização do problema e os principais objetivos que nos motivaram a desenvolver o presente trabalho.

1.1 Contextualização

A ampla cobertura das redes de comunicação está fazendo corporações distribuídas geograficamente vislumbrarem a possibilidade de integrar seus sistemas, configurar os mesmos, ou prestar serviços a clientes na Internet.

Com a globalização, cada organização que fornece serviços pode contratar, ou se associar a outras empresas especializadas em determinadas atividades, realizando assim serviços para os quais, sozinha, não possuiria competência técnica para o provimento. Isto aumenta à abrangência da área de atuação, fornecendo um único serviço composto pela

integração de várias empresas altamente especializadas em determinadas áreas.

A terceirização de serviços de maneira permanente ou temporária causa a interação entre clientes, grupos de trabalho, corporações e suas filiais, e também a necessidade de acesso à recursos localizados em diferentes domínios (*e.g.*: parceiros da corporação). Isto exige um modelo de controle de acesso que aceite opções de configuração de maneira pré-estabelecida e dinâmica.

Em meios de interação como os supracitados, a aplicação de políticas pré-estabelecidas e dinâmicas, sem a intervenção do administrador local (*i.e.*: administrador da filial ou parceira), facilitaria o gerenciamento do ambiente de controle de acesso distribuído. Em condições ideais, o *principal*[1] que deseja acessar certo recurso, mesmo não estando presente nas políticas locais da filial, poderia simplesmente apresentar uma credencial para o guardião do recurso e ter o acesso autorizado.

Realizar a troca de informações de maneira padronizada e segura, e controlar o acesso a recursos expostos na Internet em uma corporação com sistemas distribuídos e heterogêneos, não é uma atividade trivial. Confiar a escrita das políticas de controle de acesso de fina granularidade unicamente a administradores de segurança em nível corporativo, torna o processo demorado, propenso a erros e talvez até inviável. Semelhantemente, confiar a escrita das políticas de controle de acesso para os administradores das filiais não garante que as políticas corporativas serão corretamente aplicadas.

Levando-se em consideração o crescente número de recursos, *principals* e a diversidade de plataformas existentes, o procedimento de escrita e administração de políticas se torna cada vez mais trabalhoso.

Sistemas de controle de acesso tradicionais aplicados na maioria das arquiteturas são centralizados. Esta abordagem, apesar de tornar o sistema mais facilmente gerenciável, concentra os processos de administração de políticas, autenticação e autorização em um único ponto. Com isto, o sistema se torna pouco flexível e escalável, gerando o ponto único de falhas, e restringindo o *principal* de um domínio no acesso a recursos expostos por outros domínios. Todas as credenciais, de todos os *principals*, devem estar devidamente cadastradas

[1] *Principal* é um termo genérico utilizado para designar qualquer entidade (*e.g.*: humano ou programa) que está solicitando acesso a um recurso ou serviço.

e atualizadas nos demais domínios detentores do recurso almejado.

O procedimento de sincronização e atualização de informações interdomínios exige um tempo considerável, sendo propenso a erros manuais. Contas de usuários "órfãos" que deveriam ter sido excluídas ou canceladas podem permanecer no sistema, fornecendo acesso para quem não tem mais direito ao mesmo. Problema comum e facilmente encontrado em sistemas distribuídos de controle de acesso.

Além dos mecanismos de autenticação e autorização inflexíveis e centralizados usados em Arquiteturas Orientadas a Serviço, alguns destes estabelecem forte acoplamento entre os *principals* e os provedores de serviço.

O alto acoplamento pode ser descrito pela excessiva troca de mensagens entre as entidades do sistema, a dependência de entidades externas ao domínio do provedor de serviço, ou a necessidade de se compartilhar um mecanismo de confiança comum (uma *Trusted Third Party* como serviço de autenticação, por exemplo) entre as partes (*e.g.*: clientes e provedores). Gera-se também, muitas vezes, a centralização dos mecanismos que realizam os processos de distribuição de credenciais de autenticação e autorização em um único ponto.

As características supracitadas ajudam a tornar os sistemas mais vulneráveis e susceptíveis a ataques, os quais podem levar a negação de serviço ou a parada do sistema que dependa de tal serviço.

A Arquitetura Orientada a Serviço (AOS) e os Serviços Web (*Web Services* - WS) têm mostrado boa aceitação e rápida padronização, auxiliando na resolução de algumas limitações não abordadas por outros padrões existentes. O uso de interfaces padronizadas utilizadas para descrever as capacidades de um serviço, facilita a localização e a interação com os seus prestadores. A AOS oferece facilidades para conectar os sistemas da matriz com as filiais, ou implementar interações entre parceiros de negócios através da troca de mensagens padronizadas, realizadas entre os sistemas que compõem tais ambientes.

Atualmente há a necessidade de se autenticar, autorizar e confiar no *principal* que está fazendo uso dos serviços expostos na *web* por um determinado provedor. Somente autenticação e autorização não são mais suficientes para prover a segurança necessária, é preciso conhecer o domínio de origem do *principal* e se o mesmo é uma entidade confiável.

Porém, introduzir o conceito confiança para sistemas computacionais não é uma tarefa simples. A maioria das infra-estruturas e modelos de autenticação e autorização propostos na literatura, e usando AOS, abordam de forma genérica o conceito de confiança. Por outro lado,

criam sistemas fortemente acoplados (*i.e.*: exigindo o uso de mecanismos de autenticação e autorização específicos ou fazendo uso de um grande número de troca de mensagens) para tentar suprir o seu uso e a sua complexidade de implementação.

O forte acoplamento está concentrado, principalmente, entre as entidades de segurança do sistema e entre os mecanismos que realizam a avaliação e aplicação das políticas de controle de acesso, como implementado em [2] e [3], por exemplo. O ideal, é que a arquitetura ofereça o máximo de flexibilidade, exigindo o mínimo de dependência de um mecanismo ou serviço específico.

A flexibilidade e a mínima dependência devem ser utilizadas sem afetar a avaliação correta do sistema de controle de acesso. Ou seja, não autorizar *principals* que não possuem direitos de acesso ao recurso. O trabalho apresentado em [4] sugere uma abordagem interessante, porém, se mantém em nível de modelo, carecendo de provas mais concretas para mostrar sua aplicabilidade em ambientes reais.

A grande maioria dos trabalhos disponíveis na literatura científica aborda a AOS de maneira incompleta, necessitando de estudos mais aprofundados sobre aspectos como a aplicação de políticas distribuídas e de granularidade fina, sistemas distribuídos de controle de acesso, estabelecimento dos relacionamentos de confiança, gerenciamento das contas dos *principals*, entre outros. Geralmente, os mesmos estão preocupados com algumas especificidades em um determinado objetivo da AOS, expondo modelos que vão contra os princípios de baixo acoplamento almejados pela Arquitetura Orientada a Serviço.

A adoção de uma infra-estrutura de chave pública flexível auxiliaria na atividade de gestão de políticas para ambientes mais dinâmicos, sem exigir uma infra-estrutura de servidores de autenticação específicos, diminuindo a necessidade de intervenção do administrador. Adicionalmente as políticas de autorização podem ficar dispersas pelo ambiente distribuído. Processos de assinatura e cifragem diminuem as possibilidades de alterações indevidas, garantindo a credibilidade dos certificados de autorização emitidos por tal infra-estrutura.

A administração descentralizada de direitos baseada em uma infra-estrutura de chaves públicas flexível permite que o administrador corporativo delegue direitos ao administrador local (*i.e.*: administrador das filiais). Esta abordagem permite que o administrador corporativo não corra o risco de não perceber o administrador local violar as políticas corporativas. Ou seja, não é possível ao administrador local atribuir à sua discrição algum direito que não seja

derivado daqueles delegados pelo administrador em nível corporativo.

O custo da administração humana de políticas corporativas é bastante significativo, pois é imprescindível que cada regra de política (*i.e.*: política de fina granularidade) seja criada manualmente. É necessário um processo automatizado e que utilize mecanismos padronizados, visando facilitar a administração das políticas.

Através dos certificados de autorização e da concessão de direitos é possível gerar novas políticas com fina granularidade no domínio de cada filial. As novas políticas servem para atualizar o repositório de políticas corporativo, preservando a visão administrativa unificada do ambiente de controle de acesso distribuído. As políticas recebidas pela administração corporativa são usadas para atualizar, automaticamente, os repositórios de políticas dos respectivos provedores de serviços instanciados nas filiais.

A geração automatizada de políticas de fina granularidade é feita de maneira segura, pois a política é extraída de parâmetros codificados nos certificados de autorização. Com esta abordagem, os administradores (em nível corporativo ou nas filiais) estão livres do encargo de escrever novas políticas específicas para todo sujeito e respectivo recurso.

Este esquema ajuda manter a visão unificada das políticas administrativas aplicadas em ambientes distribuídos tal como em Arquiteturas Orientadas a Serviço. Prática esta que seria difícil de obter com as abordagens tradicionais. No esquema proposto neste trabalho, se diminuí consideravelmente a dependência de entidades internas e ou externas ao domínio do provedor, característica esta que seria inevitável com as abordagens tradicionais.

Em ambientes como a Internet, onde a conexão entre as entidades (*e.g.*: monitor de referência e guardião) não pode ser totalmente garantida (*i.e.*: serviço sempre *online*) sem gastos adicionais com infra-estrutura (*i.e.*: garantia fornecida pela prestadora de serviços de telecomunicação), o modelo de avaliação de políticas baseado em pré-configuração torna-se indispensável.

A pré-configuração pode ser descrita pelo fornecimento (*i.e.*: provisionamento) de configurações para um determinado sistema, antes de este executar alguma tarefa.

O modelo de provisionamento pode ser usado para configurar os mecanismos que realizam a aplicação das políticas distribuídas, ajudando a diminuir o acoplamento (*e.g.*: troca de mensagens) entre as entidades de controle presentes no domínio do provedor, e as fornecedoras de informações sobre os *principals* instanciadas no domínio do cliente.

A utilização do modelo de provisionamento reduz o *overhead* de comunicação entre o

monitor de referência corporativo e o guardião no processo de tomada de decisão sobre solicitações de acesso. A implementação deste modelo permite que os domínios dos provedores de serviços (*i.e.*: filiais da corporação) continuem atendendo solicitações de acesso mesmo na indisponibilidade do monitor de referência em nível corporativo, por exemplo.

O maior número de mensagens trocadas entre as entidades que decidem e impõem as políticas, conseqüentemente, gera maior *overhead* no sistema e maior tráfego de informações sobre a rede. Características estas que colaboram para o aumento do tempo de resposta para o *principal* (*i.e.*: cliente) durante a concretização de uma solicitação feita aos provedores de serviço. Estas peculiaridades contribuem para tornar os sistemas pouco flexíveis e escaláveis.

O ganho no tempo de resposta proporcionado pelo modelo de provisionamento diminuiu a espera para que o *principal* obtenha acesso ao recurso, pois o modelo, da maneira como foi implementado, não depende de entidades externas ao domínio do provedor, filial da corporação. A implementação de sistemas de controle de acesso em Serviços Web, baseados no paradigma de provisionamento, sequer foi discutida por algum trabalho da literatura.

O tema é atual e relevante, sendo que isto é facilmente observado quando verificamos as entidades envolvidas na criação de normas relativas ao mesmo: IETF (*Internet Engineering Task Force*), OASIS (*Organization for the Advancement of Structured Information Standards*), WS-I (*Web Services Interoperability Organization*), e W3C (*World Wide Web Consortium*).

Além das entidades supracitadas, grandes empresas como IBM, Intel, Google, Nokia, Oracle, SAP, SUN, e muitas outras, estão diretamente envolvidas com a criação especificações. Algumas destas empresas desenvolvem *frameworks* para serem utilizados na Arquitetura Orientada a Serviço e nos Serviços Web.

Porém, a abordagem adotada pelas empresas e corporações é recorrentemente, baseada na arquitetura centralizada tradicional ou na troca excessiva de mensagens, gerando um alto grau de dependência entre as entidades instanciadas nos diferentes domínios.

A literatura científica apresenta algumas propostas de como implementar sistemas distribuídos de controle de acesso, porém, em geral, estas não conseguem atender satisfatoriamente aos princípios de baixo acoplamento almejados pela AOS. Pois, os modelos propostos são na verdade modelos tradicionais (*i.e.*: centralizados) adaptados para atender as necessidades da AOS e dos WS.

Neste trabalho será desenvolvido o modelo de provisionamento, seguindo uma série

de padrões para manter a compatibilidade e interoperabilidade com qualquer outra Arquitetura Orientada a Serviço ou Serviços Web.

O modelo de avaliação de políticas baseado no paradigma de provisionamento ainda não foi integrado ao contexto dos Serviços Web, visto que a maioria dos padrões e especificações existentes ainda segue a abordagem tradicional, induzindo assim o seu uso e implementação.

A abordagem tradicional facilita a gerência de políticas, pois centraliza a mesma em um único ponto, porém, com esta centralização da administração gera-se o ponto único de falhas no sistema.

O presente trabalho oferece uma alternativa à abordagem tradicional, promovendo a administração descentralizada de políticas usando o modelo de provisionamento e a concessão de direitos através de certificados de autorização.

Como adicional, no caso de uma solicitação de acesso que seria negada na abordagem tradicional, com a integração dos certificados de autorização, o *principal* tem uma alternativa para obter o direito desejado. Ou seja, mesmo se o *principal* não estiver inserido nas políticas de controle de acesso configuradas no provedor de serviços da filial, este pode recorrer ao administrador de seu domínio e obter tal direito.

1.2 Objetivos

O objetivo geral deste trabalho é criar um modelo de controle de políticas fracamente acoplado com o auxílio de uma arquitetura de autenticação e autorização flexível. Com a união dessas abordagens, visa-se o desenvolvimento de um sistema distribuído de controle de acesso de baixo acoplamento, possibilitando a transposição transparente dos domínios de segurança, tanto para *principals* (*i.e.*: clientes) quanto para provedores.

A integração de alguns dos padrões e infra-estruturas existentes facilitará o gerenciamento das políticas corporativas utilizadas nas entidades distribuídas, a aquisição de credenciais de acesso, e a implantação de características que tornarão os sistemas mais independentes no provimento de serviços bem como os *principals* no acesso aos mesmos.

Mais especificamente este trabalho visa:

- Modelar uma Arquitetura Orientada a Serviço fazendo uso de alguns dos vários

padrões existentes visando o baixo acoplamento e a interoperabilidade. Esta abordagem visa prover uma arquitetura padronizada e amplamente aceita.

- Propor um procedimento flexível de autenticação e autorização distribuída, diminuindo a troca de mensagens entre *principals* e provedores ou entre seus respectivos domínios, bem como agilizar o processo de autorização dentro do domínio do provedor.
- Propor um modelo de mensagem apoiado nos padrões existentes, utilizado o mesmo para transportar a credencial contendo a identificação, os atributos e os privilégios de acesso do *principal*. Evitando-se assim a troca de mensagens com outras entidades (*e.g.*: serviços de atributos). Esta abordagem evita a necessidade da instanciação de um serviço para fornecer atributos sobre o *principal*, ou que este necessite de conhecimentos específicos dos mecanismos de controle de acesso e ou autenticação implementados no provedor de serviços.
- Propor um modelo que de autonomia local ao provedor de serviços sem violar as políticas corporativas. Este objetivo será implementado usando o modelo de provisionamento e certificados de autorização.
- Propor um modelo para manter os repositórios de políticas (corporativo e das filiais) sempre atualizados e sincronizados.
- Propor um mecanismo de execução de políticas que aceite mais de um tipo de credencial (*e.g.*: assertivas de autorização, certificados de autorização) para autenticar e autorizar o *principal* no acesso a recursos.
- Propor um modelo que distribua os controles de autorização de maneira segura, visando facilitar o uso da concessão de direitos. Direitos estes transportados pelos certificados de autorização.
- Propor um mecanismo (*parser*) que faça uso dos certificados de autorização para extrair os parâmetros necessários para a geração de políticas de fina granularidade que, posteriormente, irão conceder direitos de acesso a recursos para os respectivos *principals*.
- Propor um mecanismo (*parser*) que faça uso de certificados de autorização para extrair os parâmetros necessários para a geração de uma credencial de acesso, a qual é utilizada para fornecer acesso imediato ao respectivo *principal* que solicitou acesso.
- Utilizar as especificações de segurança disponíveis para descrever as restrições de

acesso exigidas pelos serviços, padronizando o estabelecimento de relações de confiança entre os *principals* e os domínios dos provedores de serviço.

- Propor um esquema para transportar e armazenar as políticas de maneira segura entre os repositórios (*e.g.*: corporativo e da filial) durante a fase de provisionamento.
- Propor um esquema para enviar e armazenar as novas políticas de maneira segura para o repositório da corporação.
- Propor uma infra-estrutura de execução de políticas que possa atuar no modelo de provisionamento e tradicional.
- Implementar as entidades e os mecanismos propostos como Serviços Web.
- Implementar o protótipo do caso de uso, o qual tem a finalidade de avaliar a proposta e testar sua aplicabilidade.

1.3 Organização

A presente dissertação está organizada da seguinte maneira. O **Capítulo 2** apresenta a Arquitetura Orientada a Serviço, o protocolo de transporte de mensagens, bem como a linguagem padronizada utilizada para a troca das informações transportadas pelas mensagens. São também apresentados os Serviços Web. O **Capítulo 3** descreve os principais padrões utilizados para proteger as informações trocadas entre as entidades da AOS, bem como algumas das especificações utilizadas para padronizar e assegurar a troca de informações em ambientes de Serviços Web. O **Capítulo 4** apresenta os principais modelos de segurança e controle de políticas adotados no trabalho. Este capítulo traz algumas especificações adicionais para Serviços Web, sendo estas as mais utilizadas em processos de avaliação de políticas de controle de acesso, e transporte de informações de maneira padronizada entre diferentes domínios de segurança. O **Capítulo 5** apresenta alguns dos principais esforços da literatura em propor modelos de baixo acoplamento, seguros e escaláveis, os quais podem ser utilizados para atender algumas das preposições da AOS. O **Capítulo 6** aborda o modelo proposto. Neste capítulo é apresentado um modelo genérico para AOS, o modelo proposto para Serviços Web, e a dinâmica de um caso de estudo utilizando Serviços Web. O **Capítulo 7** mostra as avaliações feitas no protótipo implementado, expondo as tecnologias que auxiliaram no desenvolvimento do modelo proposto bem como alguns detalhes de

implementação, visando esclarecer e facilitar o entendimento do todo. E finalmente, o **Capítulo 8** apresenta as conclusões e análises feitas com base no capítulo anterior. Neste capítulo são apresentados, também, possíveis trabalhos futuros.

Capítulo 2

Arquitetura Orientada a Serviço e Serviços Web

2.1 Introdução

A expansão das redes de comunicação está fazendo organizações situadas geograficamente distantes vislumbrarem a possibilidade de integrar seus sistemas e, também, oferecer serviços a usuários da Internet.

Com a expansão, surge também a necessidade de diferentes sistemas trocarem informações uns com os outros de forma independente a plataforma, tornando-se necessário uma padronização para favorecer o funcionamento interoperável dos sistemas. Promovendo com isto a troca de informações em ambientes heterogêneos.

Os documentos criados por organizações como OASIS, W3C, WS-I, estabelecem padrões que possibilitam a troca de informações de maneira neutra à plataforma. Estes documentos abordam, também, questões de segurança relacionadas a troca de informações.

Neste capítulo será apresentada a Arquitetura Orientada a Serviço, a qual forma o modelo base para os Serviços Web.

2.2 Arquitetura Orientada a Serviço (AOS)

A Arquitetura Orientada a Serviço – AOS (*Service Oriented Architecture* – SOA) [5] tem como um de seus objetivos propor uma estrutura para disponibilizar serviços que realizem determinada tarefa para atender alguma demanda.

Pode-se dizer, a princípio, que a AOS propõe um sistema distribuído genérico, que não está restrito a uma tecnologia particular (*e.g.*: *Common Object Request Broker Architecture* - CORBA, Serviços Web). A AOS possui agentes (*e.g.*: *hardware*, *software*)

operando em diferentes ambientes, porém trabalhando juntos para um fim comum. Por ser uma arquitetura distribuída, a comunicação entre os agentes pode ser mais lenta e menos confiável se comparada a tecnologias que usam invocação direta de métodos [6].

Os agentes podem ser descritos como sendo de dois tipos básicos: (1) o cliente, caracterizando o *principal* solicitando acesso a um recurso. *Principal* é um termo genérico utilizado para designar qualquer entidade (*e.g.*: humano ou programa) que está solicitando acesso a um serviço; (2) o agente provedor ou provedor de serviço, responsável por implementar e prover o serviço (*e.g.*: armazenamento e pesquisa de dados, processamento de tarefas, etc.). Os termos serviço e recurso serão usados de maneira intercambiável durante a exposição do texto para designar o objetivo almejado pelo *principal*.

A Arquitetura Orientada a Serviço permite realizar a construção, disponibilização e utilização de serviços distribuídos, sendo uma de suas características o fraco acoplamento entre o *principal* e o serviço. Ou seja, o *principal* não necessita conhecer os detalhes internos de funcionamento ou invocação de métodos do provedor que atenderá a solicitação. Desta forma é permitido ao *principal* utilizar outro provedor de serviço acessível em outro endereço da Internet, caso o recurso fornecido atualmente falhe ou o mesmo não atenda mais as necessidades do *principal* [7], por exemplo.

Este modelo de arquitetura de sistemas distribuídos é caracterizado em termos do que o serviço faz e na interação que ocorre durante a troca de mensagens entre o *principal* e o provedor. As mensagens são enviadas em um formato padronizado por meio da interface disponibilizada pelo serviço. A interface expõe somente os detalhes importantes para o uso do serviço. A descrição do modelo conceitual da Arquitetura Orientada a Serviço pode ser encontrada no Apêndice A.1.

A descrição de um serviço exposto, segundo a AOS, pode incluir políticas associadas com o mesmo, fornecendo as informações necessárias para o *principal* avaliar se o serviço opera de maneira consistente com seus requisitos. A política pode descrever restrições ou condições de uso para *principals* e provedores. Estando ambas as partes de acordo com as políticas, estas serão usadas para regrar a interação entre as entidades.

A AOS enfatiza a implementação de seus componentes como módulos de serviço, os quais podem ser descobertos e usados por *principals*. Os serviços podem ter propósitos individuais, ou podem ser integrados para executar um fluxo de trabalho mais complexo, dependendo assim da agregação de outros serviços.

Na seção 2.5 será apresentada uma implementação da AOS baseada na *web* (*i.e.*: Serviços Web) que é fortemente fundamentada em XML (*Extensible Markup Language*) e SOAP (*Simple Object Access Protocol*). Estes assuntos serão brevemente abordados nas Seções 2.3 e 2.4 respectivamente, para facilitar o entendimento dos Serviços Web.

2.3 Linguagem de Marcação Extensível (XML)

A XML (*Extensible Markup Language*) [8] é uma linguagem de marcação extensível muito utilizada para descrever os dados trocados entre os serviços, por exemplo. Sua capacidade de extensão decorre do fato que qualquer estrutura de dados que possa ser descrita com caracteres, pode ser delimitada por *tags* XML.

As limitações da linguagem são impostas pela própria estrutura dos dados. Sendo assim, qualquer indivíduo pode fazer uso da XML e criar sua própria definição de tipos de dados. As *tags* servem para fazer a marcação do conteúdo do documento XML, identificando a sua estrutura e as informações contidas neste, estruturando assim o conteúdo do documento.

Toda estrutura de dados convertida para XML é referenciada como um documento XML, o qual está tipicamente associado a um esquema. O esquema especifica as regras de gramática, as *tags* permitidas, a estrutura das *tags*, o tipo de dados que se espera encontrar, entre outras regras que se queira definir. Se o documento XML estiver de acordo com seu esquema, o mesmo será validado pelo receptor e processado de forma correta.

Os dois tipos mais comuns utilizados na validação de documentos XML são o DTD (*Document Type Definition*) e o esquema (XML *Schema*) [9]. Estes descrevem o conteúdo permitido em documentos XML. Assim é possível realizar a validação do documento XML recebido de acordo com as regras especificadas em um dos documentos citados acima.

DTD é um documento de texto que descreve o formato dos dados presentes em um documento XML, porém, não sendo formatado em XML. *Schema* é um tipo de validação de dados formatado em XML, mais atual e também mais utilizado, que também serve para definir a estrutura do documento XML.

O documento XML é dito bem formado ou válido se este atende a todos os requisitos de um ou mais esquemas. Um documento XML válido pode ser interpretado como um documento compartilhado e entendido entre as partes.

O padrão XML faz uso do espaço de nome (*namespace*), permitindo a definição do

domínio no qual o vocabulário definido dentro de um documento XML é válido. Estes domínios são geralmente referenciados por uma URL (*Uniform Resource Locator*).

Freqüentemente a URL resolve para uma página *web* que provê a documentação sobre o espaço de nomes utilizado, assim como informações sobre o tipo de conversão de dados usada. A URL pode ser usada para definir prefixos, usados na identificação de nomes de elementos em documentos XML.

Documentos XML emitidos por diferentes entidades podem possuir elementos com nomes semelhantes. Para resolver possíveis conflitos de nomes, a XML *Namespaces* faz uso do conceito de nomes qualificados (*i.e.*: *QNames*), uma combinação da URL e um nome local, ambos fornecidos por quem emitiu o documento. Assim, independente do contexto, o nome será sempre único.

Uma das principais van*ta*gens do XML para os Serviços Web é a independência dos dados. Os tipos de dados e estruturas dos documentos não são vinculados com as implementações subjacentes dos serviços, como uma linguagem de programação específica, por exemplo. As aplicações convertem os dados para XML e depois fazem o processo inverso, trazendo novamente o dado para o formato nativo de cada plataforma.

O fragmento de um documento XML mostrado na Figura 2.1 expõe os códigos da linguagem XML, mostrando os atributos de um livro. Na hierarquia do documento XML, após o elemento livro, têm-se os elementos que descrevem o livro.

```
<livraria>
      <livro>
            <titulo>Java: Como Programar. 6ª Edição</titulo>
            <autor>Deitel</autor>
            <preco>200,00</preco>
      </livro>
</livraria>
```

Figura 2.1: Fragmento de um documento XML

A XML tem sido aceita como a linguagem para descrever os dados trocados em ambientes de Serviços Web, sendo aplicada na definição e formatação de dados, serialização, e transformação destes em um formato padrão.

A comunicação com os Serviços Web geralmente ocorre através da troca de documentos XML encapsulados em mensagens SOAP (Seção 2.4). Este protocolo carrega solicitações e respostas durante a interação com os Serviços Web.

2.4 Protocolo Simples de Acesso a Objetos (SOAP)

O protocolo SOAP (*Simple Object Access Protocol*) [10] está baseado no padrão XML. Esse é utilizado na troca de informações em ambientes distribuídos e ou heterogêneos. O SOAP provê um formato de mensagem comum para a troca de dados entre agentes solicitantes (*i.e.*: *principals*) e provedores, ou outros componentes da arquitetura.

A mensagem SOAP pode trafegar sobre uma variedade de protocolos subjacentes (*e.g.*: *Hypertext Transport Protocol* – HTTP, *Simple Mail Transport Protocol* – SMTP). Apesar de ser um protocolo para troca de mensagens, o padrão não especifica que toda mensagem SOAP enviada a um destinatário necessita ter uma resposta.

A mensagem SOAP é composta basicamente de três elementos: o envelope, um ou mais cabeçalhos e o corpo (Figura 2.2).

- ***Envelope:*** delimita o início e o fim da mensagem. Esse ainda é composto por mais dois elementos: o espaço de nomes XML, que especifica o nome completamente qualificado (*QName*); e o estilo de codificação, identificando o tipo dos dados usados na mensagem SOAP.

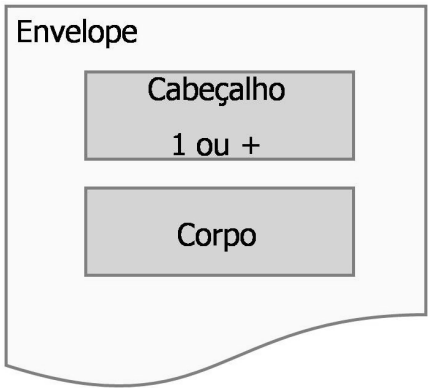

Figura 2.2: Mensagem SOAP

- ***Cabeçalho:*** não é um elemento obrigatório, mas se estiver presente pode ser usado para carregar informações sobre a qualidade do serviço ou informações referentes a segurança (definidas na especificação WS-Security [11]). Se informações adicionais estiverem presentes, um determinado nó intermediário (*i.e.*: processo ou serviço entre

o agente solicitante e provedor) pode executar algum processamento adicional, além do processamento executado pelo destinatário final da mensagem.

- ***Corpo:*** contêm os dados (*i.e.*: *payload*) ou parâmetros da mensagem SOAP, direcionados a um destinatário.

2.5 Serviços Web

Os Serviços Web (*Web Services* – WS) [6] implementam a AOS, tendo como um de seus propósitos promover a interoperabilidade entre diferentes sistemas, instanciados em uma diversidade de plataformas. Sistemas que tipicamente interagem usando mensagens SOAP (Seção 2.4) transportadas por HTTP, tendo a serialização dos dados baseada em XML (Seção 2.3).

Serviços Web é o termo usado para se referir a determinado serviço implementado de maneira concreta em um provedor. A troca de mensagens via HTTP e a orientação a serviço são algumas das características que descrevem o fraco acoplamento entre o *principal* e o provedor [12]. Uma descrição mais concreta da arquitetura geral dos Serviços Web pode ser encontrada no Apêndice A.2.

O serviço por sua vez é implementado em alguma linguagem de programação particular e descrito seguindo uma especificação processável por máquina, a WSDL (*Web Services Description Language*, Seção 2.5.1). A WSDL define os parâmetros aceitos pelo serviço e como ocorrerá a interação com o mesmo. Geralmente, todos os Serviços Web trazem consigo a semântica da operação, refletindo o propósito e a implicação de sua invocação.

Para que a interação entre as entidades possa ocorrer de maneira satisfatória, algumas premissas devem ser satisfeitas: o *principal* e o provedor devem concordar na descrição, na semântica e na política utilizada para manter o serviço seguro. A política deve ser compreendida pelo *principal* e implementada pelo provedor.

Na Figura 2.3 mostram-se as principais entidades da arquitetura dos Serviços Web e a possível interação entre as mesmas no processo de criação, publicação e acesso a um determinado serviço anunciado por um provedor.

Após a criação do serviço no provedor (evento *1*, Figura 2.3) gera-se o documento

WSDL que descreve o serviço e publica-se o mesmo na UDDI (*Universal Description Discovery and Integration*, evento *2*). A UDDI tem a função de repositório, utilizada para armazenar documentos WSDL. Uma descrição mais detalhada sobre a UDDI pode ser encontrada no Apêndice A.3. O *principal* a procura de determinado serviço, entra em contato com o serviço UDDI através de algum mecanismo de pesquisa (evento *3*) para obter o documento WSDL (evento *3.1*).

De acordo com as informações obtidas do documento WSDL, implementa-se o programa cliente (evento *4*). Os dados da solicitação sendo enviada ao provedor de serviço (evento *cinco*) são serializados em XML e encapsulados em mensagens SOAP. Neste caso o protocolo de transporte subjacente é o HTTP. Após o processamento empreendido pelo provedor, o mesmo pode retornar uma mensagem (evento *5.1*) transportando a resposta da solicitação feita anteriormente.

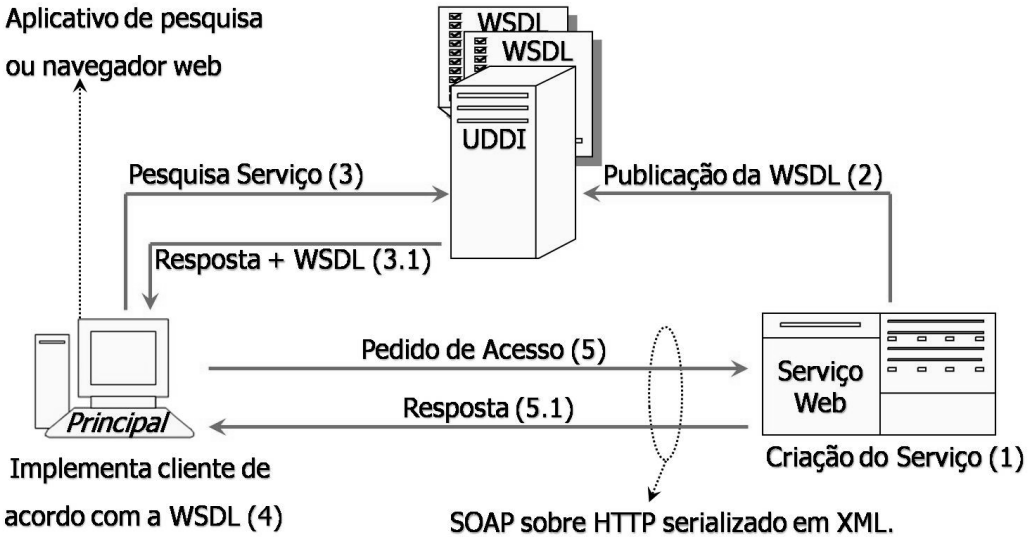

Figura 2.3: Principais entidades que compõem a arquitetura dos Serviços Web

Em Serviços Web, geralmente, as mensagens são trocadas entre *principals* e provedores geograficamente distantes, assim pode ser necessário um grau de certeza que a mensagem foi entregue, e que tanto o emissor quanto o receptor tiveram o mesmo entendimento da mensagem.

O grau de certeza requerido em alguns tipos de Serviços Web pode ser alcançado usando-se especificações adicionais, agregando-se certo nível de garantia na troca de

mensagens entre as entidades (*i.e.*: *principal* e provedor). Mais detalhes sobre algumas especificações que provêem tais garantias podem ser encontrados no Apêndice A.4.

2.5.1 Linguagem para Descrição dos Serviços Web (WSDL)

A WSDL (*Web Services Description Language*) [13] define a estrutura de documentos de acordo com o formato XML, podendo ser usada para descrever os componentes necessários para se realizar alguma interação com os Serviços Web. A WSDL visa assegurar a independência de plataforma e linguagem de programação.

A descrição do serviço divide-se em dois níveis, sendo um deles genérico e um concreto. No nível genérico a WSDL descreve os Serviços Web em termos de mensagens enviadas e ou recebidas. Mensagens tipicamente descritas de acordo com o XML *Schema*. Em nível concreto, a WSDL especifica a ligação (*binding*), ou seja, o tipo de transporte usado na comunicação entre *principals* e provedores, e detalhes do formato de uma ou mais interfaces que compõem o serviço. O documento criado de acordo com o formato da WSDL pode ser processado por programas de computadores específicos e também interpretado manualmente por humanos.

Para se diminuir a quantidade de falhas durante as interações, cada método no provedor de serviços deve ser inequivocamente identificado pelo seu nome qualificado (*i.e.*: *QName*). Isto é, se dois métodos distintos estão no mesmo espaço de nomes do provedor, então esses devem ter *QNames* únicos.

Para haver interação entre o *principal* e o provedor de serviço, estes devem concordar com o formato do documento WSDL. O *principal*, primeiramente, deve localizar (evento *3*, Figura 2.3) a descrição do serviço, a qual contém as informações de como interagir com serviço bem como a sua localização (*e.g.*: URL). Com a WSDL, implementa-se o cliente (evento *4*) e envia-se uma determinada mensagem com os devidos parâmetros para o provedor que está fornecendo o serviço (evento *5*). O provedor é responsável por manipular a mensagem recebida, e caso necessário devolver uma resposta para o endereço determinado na mensagem (evento *5.1*).

2.6 Conclusão

A Arquitetura Orientada a Serviço e os Serviços Web estão sendo vistos como a base para uma nova geração de aplicações distribuídas e ferramentas para gerenciamento de sistemas [12].

A neutralidade a plataforma com a qual os Serviços Web podem ser implementados e acessados, de maneira local e ou remota, é um dos fatores que tem impulsionado a padronização de sua arquitetura. Várias empresas usam e desenvolvem novas e aperfeiçoadas especificações para que este tipo de sistema distribuído ganhe mais adeptos.

O próximo capítulo abordará os principais esforços de organizações como OASIS e W3C em produzir documentos, visando padronizar este meio um tanto quanto conturbado pela evolução constante que os sistemas sempre estão passando.

Capítulo 3

Padronização para XML e Serviços Web

3.1 Introdução

Mecanismos de segurança tradicionais como SSL/TLS (*Secure Sockets Layer/Transport Layer Security*), IPSec (*Internet Protocol Security*) e VPN (*Virtual Private Network*), são tecnologias utilizadas para proteger conexões ponto-a-ponto. Embora tais tecnologias sejam usadas na segurança de Serviços Web, estas não são suficientes para prover segurança fim-a-fim, em nível de mensagem SOAP, como os Serviços Web necessitam [6].

Uma das justificativas para as preocupações com a segurança em Serviços Web é o fato de o protocolo SOAP ser *firewall friendly* [14]. Ou seja, ele passa pelo filtro de pacotes (e pela porta 80) como se fosse trafego HTTP normal. Tendo esta passagem livre, pessoas mal intencionadas podem fazer uso de mensagens SOAP para realizar ataques aos serviços fornecidos pelo provedor.

Entre os vários tipos de ataques que a mensagem SOAP pode sofrer, podem-se destacar aqueles que comprometem a confidencialidade e a integridade, os quais podem ser inibidos pelo uso da cifragem de dados e técnicas de assinatura digital.

A mensagem SOAP pode passar por vários nós intermediários antes de atingir seu destino, então segurança em nível da mensagem é importante ao invés de somente ponto-a-ponto. O contexto de segurança aplicado a mensagem SOAP é fim-a-fim, ou seja, mesmo após o recebimento da mensagem no nó de destino, a mesma continua protegida pela cifragem e assinatura.

Porém, pode haver casos em que um nó intermediário necessita ter acesso a alguma informação transportada pela mensagem SOAP, em uma situação que o nó intermediário está entre o solicitante e o destinatário. A WS-Security [11] estabelece regras e modelos para que

somente partes específicas da mensagem sejam cifradas, deixando as demais partes em texto claro para os nós intermediários terem acesso as informações.

Ataques do tipo repetição, *man-in-the-middle*, *spoofing* e negação de serviço também comprometem a interação segura entre as partes. Alguns mecanismos de autenticação como os propostos pela especificação WS-Security podem ser usados para autenticar os *principals*, reduzindo assim os riscos de fraudes. Dentre os mecanismos podem-se incluir a criptografia, sistemas de autenticação baseados em usuário/senha, *one time pass*, *timestamp*, certificados, etc. Após o *principal* estar autenticado, mecanismos de autorização podem controlar o acesso do mesmo aos recursos do sistema.

O presente capítulo aborda, brevemente, alguns dos principais esforços na criação de padrões para tentar solucionar alguns dos problemas citados acima.

3.2 Padrões de Segurança para XML

Para proteger as informações encapsuladas em documentos XML foram criados padrões que visam suprir tal demanda.

Padrões como o XML *Encryption* e o XML *Signature* são usados para proteger o conteúdo da mensagem SOAP, fornecendo garantias para que as informações trocadas entre os agentes (*i.e.*: *principal* e provedor) não sejam indevidamente alteradas durante seu transporte na rede.

3.2.1 XML Encryption

O padrão XML *Encryption* [15] define um processo para cifrar dados digitais e a forma como o resultado do processo de cifragem deve ser representada em um documento XML. Os dados cifrados podem ser uma seqüência de caracteres, dados binários, um elemento XML, o conteúdo de um elemento ou um documento XML inteiro.

O processo de cifragem provê a confidencialidade do dado cifrado, podendo ser executado por algoritmos de chave simétrica ou assimétrica. O XML *Encryption* também suporta uma super-cifragem de dados, isto é, um dado já cifrado pode ser novamente cifrado.

O resultado do processo de cifragem é colocado em um elemento chamado <*EncryptedData*> dentro do documento XML. Este documento, além de carregar o dado

cifrado, pode fornecer informações sobre o tipo do algoritmo utilizado, tamanho da chave usada para o processo de cifragem, etc. As informações adicionais servem para que o receptor possa decifrar a mensagem. Caso estas informações sejam omitidas, espera-se que o receptor já as conheça previamente.

As Figuras abaixo mostram o resultado de um processo de cifragem. A linha *1* (Figuras 3.1 e 3.2) mostra a versão do XML que está sendo usada. A linha *2* (Figuras 3.1 e 3.2) descreve informações sobre determinado pagamento, juntamente com o espaço de nomes (URI - *Uniform Resource Identifiers*). A linha *3* (Figuras 3.1 e 3.2) informa o nome do sujeito que está efetuando o pagamento.

```
1. <?xml version='1.0'?>
2.          <PaymentInfo xmlns='http://example.org/paymentv2'>
3.                <Name>John Smith</Name>
4.                <CreditCard Limit='5,000' Currency='USD'>
5.                      <Number>4019 2445 0277 5567</Number>
6.                      <Issuer>Example Bank</Issuer>
7.                      <Expiration>04/02</Expiration>
8.                </CreditCard>
9.          </PaymentInfo>
```

Figura 3.1: Documento XML em texto claro

Entre as linhas *4* e *8* (Figura 3.1), juntamente com os elementos aninhados em sua estrutura, estão as demais informações referentes ao pagamento que o sujeito está efetuando (*e.g.*: a moeda em circulação, o nome do banco, data). A linha *9* encerra o documento XML, indicado pela marcação "</>".

```
1. <?xml version='1.0'?>
2.          <PaymentInfo xmlns='http://example.org/paymentv2'>
3.          <Name>John Smith</Name>
4.                <EncryptedData
5.                Type='http://www.w3.org/2001/04/xmlenc#Element'
6.                xmlns='http://www.w3.org/2001/04/xmlenc#'>
7.                      <CipherData>
8.                            <CipherValue>A23B45C56</CipherValue>
9.                      </CipherData>
10.               </EncryptedData>
11.         </PaymentInfo>
```

Figura 3.2: Documento XML cifrado

A julgar pelo fato que as informações referentes ao pagamento (*e.g.*: limite, número de cartão de credito) são consideradas sensíveis e não deveriam ser expostas (linhas *4* a *8*, Figura 3.1), as mesmas necessitam ser protegidas para não serem reveladas.

Entre as linhas *4* e *9* (Figura 3.2) está o elemento que transporta as informações de maneira cifrada, juntamente com o espaço de nomes referente à XML *Encryption*. Entre as linhas *6* e *8* estão os elementos definidos na especificação e o dado cifrado (elementos entre as linhas *4* e *8*, Figura 3.1).

No exemplo citado na Figura 3.2, o emissor e o receptor do documento XML cifrado devem possuir um segredo compartilhado, pois o presente documento não fornece nenhum tipo de informação adicional (*e.g.*: tamanho da chave, algoritmo criptográfico) que auxilie no processo de decifragem do documento XML.

3.2.2 XML Signature

O XML *Signature* [16] define como o dado será assinado e como o resultado da assinatura deve ser representada em um documento XML. Com este padrão, todas ou apenas partes específicas de um documento XML podem ser assinadas. O XML *Signature* pode ser aplicado sobre qualquer conteúdo digital, incluindo o conteúdo de um ou mais serviços.

O uso deste padrão assegura autenticidade, integridade e não repúdio nas interações com algum serviço. O XML *Signature* pode ser usado por aplicações distribuídas para autenticar a identidade do emissor de uma mensagem ou documento, ou aferir que as informações transportadas pela mensagem não foram alteradas enquanto eram conduzidas pela rede. O emissor tendo assinado o documento XML, não pode negar a sua emissão, pois somente esse deveria conhecer a chave para realizar a assinatura, por exemplo.

A assinatura digital é mais complexa de ser implementada do que a cifragem, porque essa é vinculada matematicamente com a representação do dado sendo assinado. O dado pode ser um resumo (*hash*) ou o próprio dado. Deve-se ter cuidado na verificação da consistência do dado, sendo que este procedimento é sensível a ordem de processamento efetuada pelas partes que trocaram alguma informação assinada e ou cifrada.

Mesmo a assinatura sendo válida, em seu período de criação, a mesma pode não ser verificável devido a mudanças ocorridas durante o transporte da mensagem. O elemento <*Signature*> é quem transporta o conteúdo da assinatura, esse pode ser o elemento XML de

nível mais alto na hierarquia da estrutura ou pode estar aninhado em alguma *tag* dentro do documento XML. A padronização dos dados efetuada antes do processo de assinatura é feita pelo padrão XML sob a forma canônica [17] (Seção 3.2.3).

Uma assinatura pode ser associada com um documento de três diferentes modos: (1) *enveloped* e (2) *enveloping*: acompanhando o documento XML ao qual referenciam, e (3) *detached*: o dado assinado está fora do elemento <*Signature*>, sendo apenas referenciado na estrutura da assinatura.

3.2.3 XML Canonical

O padrão XML sob a forma canônica [17] define regras para transformar um documento XML em uma representação padrão. Esse estabelece métodos para determinar se dois documentos são idênticos, e ou não foram trocados durante o transporte, efetuando uma padronização dos elementos do documento antes do processo de resumo e ou assinatura.

O XML *Canonical* utiliza a técnica de resumo (*hash*) intolerante a mínimas mudanças em um documento. Mesmo a introdução de um espaço em branco produz um resumo diferente. Essa intolerância é essencial para a produção de um resumo seguro. Deve ser próximo do impossível de modificar o documento original de tal modo que o documento alterado ainda produza o mesmo resumo.

Para se utilizar esta técnica de maneira satisfatória, cuidados devem ser relevados, visto que documentos XML são freqüentemente analisados ou re-analisados durante o seu transporte do emissor até o receptor. As analises podem fazer alguma mudança significante, (*e.g.*: eliminando um espaço em branco) comprometendo a validação da assinatura por parte do receptor. Sendo que o emissor e o receptor deveriam computar o mesmo resumo, sem levar em consideração os processos (analises) ocorridos durante o transporte na rede, por exemplo.

O XML sob a forma canônica permite que se assine somente o resumo da mensagem ao invés da mensagem inteira. Tendo isto agrupado a algoritmos de resumo adequados, garante-se a autenticidade e a integridade da mensagem.

3.3 Padrões para Serviços Web

Inúmeras são as especificações disponibilizadas para Serviços Web, nesta seção serão

descritas as que mais têm relação com o presente trabalho.

Documentos como o *Basic Profile* [18] e o *Basic Security Profile* [19] fornecidos pela organização WS-I (*Web Services Interoperability Organization*) ajudam a manter um nível maior de interoperabilidade entre os Serviços Web. Estes documentos agregam diversas especificações, todas devidamente testadas, visando fornecer um perfil básico que pode ser seguido durante a implementação de Serviços Web mais seguros.

Algumas das especificações descritas a seguir abordam a troca de mensagem padronizada entre *principals* e provedores, enquanto outras abordam a segurança na troca destas mensagens.

3.3.1 Web Service Security: SOAP Message Security (WS-Security)

A especificação WS-Security [11] forma a base para outros padrões de segurança existentes (Figura 3.3) permitindo à variedade de Serviços Web interoperarem com segurança e de maneira neutra a linguagem e plataforma.

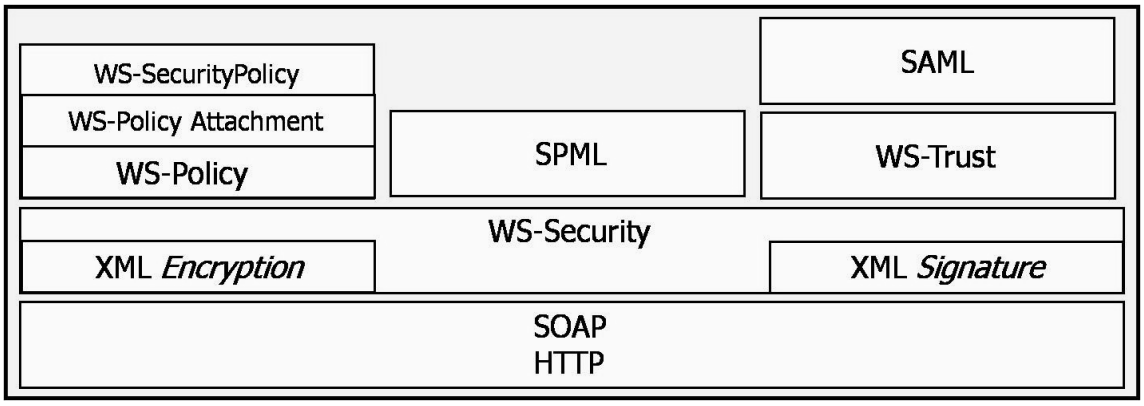

Figura 3.3: Arquitetura de segurança dos Serviços Web

A segurança é alcançada agregando extensões a mensagens SOAP, fornecendo suporte para assinatura, cifragem e à representação de credenciais de segurança, provendo assim interações mais seguras entre diferentes domínios administrativos.

A WS-Security tenta suprir a necessidade de diferentes padrões interagirem em uma linguagem de segurança comum em nível de mensagem, diminuindo assim a barreira imposta por diferentes implementações de segurança e ou domínios administrativos.

Os principais mecanismos providos agregam as capacidades de garantir a integridade, confidencialidade e o mapeamento de credenciais de segurança (*e.g.*: certificados X.509 [20], *tickets Kerberos* [21], asserções SAML – *Security Assertion Markup Language*, etc.) em mensagens SOAP. As credenciais de segurança são informações padronizadas adicionadas às mensagens SOAP. Estas informações podem ser descritas como uma coleção de atributos associados ao *principal* que enviou determinada mensagem para o WS solicitando acesso.

Uma terceira entidade confiável (*e.g.*: *Security Token Service* – STS, *Certificate Authority* – CA) pode aferir o conteúdo de uma credencial de segurança, fazendo uso de sua chave privada para assinar e ou cifrar a credencial. Através destes procedimentos garante-se a veracidade dos atributos, provando que o emissor da mensagem está de fato associado com a credencial de segurança.

A especificação WS-Security: *SOAP Message Security* define três classes de credenciais de segurança que podem ser mapeadas para o formato XML e inseridas em mensagens SOAP. Estas classes são do tipo nome de usuário-senha, binária e XML.

O formato de credenciais com o nome do usuário é a representação do esquema usuário-senha. O formato de credencial binária permite representar informações como, por exemplo, um *ticket Kerberos*, o certificado digital de alguma ICP (Infra-Estrutura de Chave Pública), etc. O formato de credencial XML pode ser usado para representar asserções SAML ou qualquer outro tipo de esquema que possa ser mapeado para o formato XML.

3.3.2 WS-Trust

A WS-Trust [22] é uma especificação que estende a WS-Security, definindo um modelo de mensagens para o transporte de credenciais de maneira segura. Credenciais usadas para se criar relações de confiança intra ou inter-domínios.

A especificação define um conjunto de mensagens pedido/resposta e também o uso do protocolo *challenge-response* [23]. Com a utilização da WS-Trust os *principals*, fazendo uso de mensagens SOAP, podem solicitar a uma autoridade confiável, no caso o STS (*Security Token Service*), que determinado tipo de credencial de segurança seja emitida, renovada ou validada em nome do solicitante. O STS é um serviço capaz de interpretar e emitir os mais variados tipos de credenciais solicitadas pelo *principal*, porém, a especificação não define como a interpretação ocorre.

O processo de emissão da credencial está baseado na identidade do *principal* que é agregada ao pedido enviado ao STS. Sendo estes parâmetros válidos e estando de acordo com as políticas do STS, uma nova credencial será emitida para o *principal*. O processo de renovação está baseado na credencial que segue junto com o pedido para o STS. O STS irá emitir uma nova credencial evidenciando uma nova data de validade. Para uma credencial de segurança ser validada, após a sua avaliação, um resultado deveria ser retornado ao solicitante, este resultado pode ser o estado da credencial, uma nova credencial, ou ambos.

A especificação WS-Security define os modelos base para a criação de mensagens seguras, a WS-Trust usa estes modelos para fazer a emissão e disseminação de credenciais de acordo com as estruturas de segurança dos domínios. A WS-Trust, além de permitir a disseminação de múltiplos formatos de credenciais de segurança, considera, também, a disseminação destas credenciais de maneira confiável e segura.

Para se estabelecer um relacionamento confiável entre as partes, incluem-se processos de autenticação e autorização que são instituídos usando a troca de credenciais de segurança. A confiança pode ser constituída entre duas ou mais partes diretamente, ou indiretamente verificada por uma terceira parte confiável, o STS. Independentemente de como o relacionamento foi estabelecido ou intermediado, cada parte necessita determinar, através de suas próprias políticas de segurança, se a mesma pode confiar na credencial emitida por outra parte.

O modelo fornecido pela WS-Trust está baseado em um procedimento no qual o WS pode exigir que a mensagem recebida forneça e ou prove um conjunto de declarações (*e.g.*: nome, chave, permissão, capacidade). Um modo de provar estas declarações é apresentar uma credencial de segurança assinada pela chave privada do STS.

O solicitante pode obter a credencial de um STS e então encaminhá-la ao provedor dos Serviços Web. O WS pode confiar na credencial emitida pelo STS, verificando a assinatura e ou cifragem empreendida pelo mesmo, ou o WS pode solicitar ao STS que emitiu a credencial para que este realize sua validação. Considerando a existência de um relacionamento confiável previamente estabelecido entre o provedor de serviços e o STS.

O provedor de Serviços Web, quando recebe uma mensagem SOAP transportando uma credencial de acesso, terá de tratar três aspectos relacionados a credencial:

- ***Formato:*** o receptor (*i.e.*: provedor de serviço) pode constatar que o formato ou a

sintaxe da credencial é incompreensível;

- ***Confiança:*** o receptor pode ser incapaz de construir um caminho de confiança desde sua autoridade confiável (*e.g.*: Autoridade Certificadora da ICP) até o portador da credencial;
- ***Espaço de nomes:*** o receptor pode ser incapaz de compreender o conjunto de declarações dentro da credencial. Visto que existem diferenças sintáticas na definição de nomes (*i.e.*: espaço de nomes), impedindo a interpretação correta da credencial.

A WS-Trust considera todos estes aspectos de interoperabilidade através do STS, fazendo uso de um conjunto de mensagens pedido/resposta utilizadas no transporte das credenciais de forma segura.

Tomando como base um exemplo em que o *principal* envia uma mensagem para o STS informando o serviço que fará uso da credencial, e solicitando que uma credencial de segurança seja emitida e ou trocada, o STS responde com uma mensagem contendo a nova credencial.

Os três passos citados anteriormente podem ser abordados da seguinte forma:

- ***Formato:*** a credencial retornada pelo STS está em um formato aceitável pelo serviço;
- ***Confiança:*** a credencial retornada será assinada pelo STS. Supondo que o *principal* e o serviço tenham um relacionamento seguro e confiável estabelecido com o STS, o serviço receptor é capaz de validar e confiar na credencial;
- ***Espaço de nomes:*** o STS assegurará que a credencial retornada usa a sintaxe apropriada para a expectativa do serviço.

Um possível caso de uso utilizando a especificação WS-Trust é mostrado no Apêndice A.5.

A emissão, renovação e validação de credenciais têm semelhanças com tecnologias de segurança existentes, como por exemplo, com o *Kerberos* [21]. O *principal* autentica-se no AS (*Authentication Service*) e recebe um TGT (*Ticket Granting Ticket* – uma emissão) que subseqüentemente utilizará para pedir um novo *ticket* ao TGS (*Ticket Granting Service* – uma troca). O novo *ticket* será encaminhado para um serviço particular.

A especificação WS-Federation [24] tem funcionalidades semelhantes a WS-Trust,

porém essa tem como objetivos padronizar a criação de relações confiáveis interdomínios. Uma descrição mais detalhada da WS-Federation é mostrada no Apêndice A.6.

3.3.3 Web Services Policy (WS-Policy)

Inúmeros são os modelos e formas de interação definidos nas especificações citadas anteriormente, com vários modos de se prover segurança e ou atestar a autenticidade de um *principal*.

A especificação WS-Policy [25], juntamente com a WS-PolicyAttachment [26] e a WS-SecurityPolicy [27], definem alguns esquemas e espaços de nomes para a linguagem XML. Sendo assim, os provedores de serviço podem expor suas exigências à *principals* que desejam realizar alguma interação com o provedor, e acessar algum serviço exposto por este.

O documento WS-Policy pode ser anexado ao arquivo WSDL (a WSDL descreve somente a semântica do serviço), e publicado junto com este na UDDI. Desta maneira, estando em um repositório global, o *principal* pode acessá-lo, e de acordo com o mesmo implementar o serviço cliente atendendo as exigências do provedor.

Este conjunto de especificações descreve, por exemplo: como e onde a mensagem deve ser protegida (*e.g.*: quais elementos da mensagem SOAP devem ser cifrados e ou assinados); qual o tipo de algoritmo que deve ser usado na geração da chave criptográfica bem como o tamanho da chave; tipo da credencial que deve acompanhar o pedido; auxílio do canal de comunicação seguro (*e.g.*: SSL/TLS); entre outros quesitos que podem ser solicitados pelo provedor.

Com as declarações de políticas definidas no documento WS-Policy, organizações podem combinar procedimentos para estabelecer uma interação segura entre os domínios. Estes procedimentos podem estar relacionados à camada de transporte, níveis de segurança e confiabilidade da mensagem, ou contemplar as transações que ocorreram entre as organizações.

Os parâmetros, os procedimentos, e como descrever as políticas (*i.e.*: *policies*) são descritos em XML, que a exemplo da WSDL, são legíveis por maquina, não necessitando de intervenção manual (*i.e.*: leitura ou interpretação executada por um ser humano).

Em um possível caso de uso, o provedor de serviço define suas exigências de segurança com base na especificação WS-Security. As exigências podem ser comunicadas

através de documentos WS-Policy, os quais irão descrever as políticas a serem utilizadas para promover uma interação segura entre as partes. Possíveis negociações ou combinações entre as alternativas de políticas descritas no documento WS-Policy (*e.g.*: tamanhos diferentes de chaves criptográficas para distinguir entre níveis de segurança) podem ser feitas.

O documento WS-Policy encapsula as políticas de segurança dos Serviços Web. A descrição dos mecanismos de segurança a serem utilizados são especificados pelo padrão WS-SecurityPolicy.

Apesar da WS-SecurityPolicy ter sido definida pela W3C, a mesma agrega informações adicionais ao documento WS-Policy original (definido pela OASIS), sem definir um novo documento. Um possível caso de uso da especificação WS-Policy é mostrado no Apêndice A.7.

3.3.4 Especificação para Gerenciamento de Chaves XML (XKMS)

A especificação XKMS (*XML Key Management Specification*) [28] surge da necessidade de um serviço para realizar o gerenciamento de chaves, de maneira simples e de fácil uso para seus clientes.

Cada infra-estrutura de chave pública, por exemplo, possui um modelo particular para representar suas chaves (*e.g.*: certificados X.509 ou SPKI – *Simple Public Key Infrastructure*). O XKMS define um conjunto de mensagens pedido/resposta utilizadas para contactar os serviços responsáveis por realizar a localização, validação, registro de chaves, etc. Uma descrição mais detalhada dos serviços disponibilizados pela especificação XKMS pode ser encontrada no Apêndice A.8.

Com o suporte do XKMS, o *principal* ou provedor não necessitam conhecer as particularidades de todos os tipos de Infra-estruturas de Chaves Públicas (ICP) existentes. Basta delegar a validação e ou recuperação de chaves para os serviços que compõem a arquitetura do XKMS.

3.3.5 Linguagem de Marcação para Configuração de Serviços (SPML)

A SPML (*Service Provisioning Markup Language*) [29] é uma linguagem baseada em XML, sendo basicamente usada no gerenciamento de objetos (*e.g.*: identidades, serviços).

Esta especificação define um conjunto de mensagens com operações para adicionar, remover, atualizar e pesquisar por objetos instanciados em servidores ou serviços, os quais realizam alguma tarefa.

O servidor (recurso alvo) onde os objetos estão, ou serão instanciados é gerenciado por um serviço (*i.e.*: provedor ou serviço de provisionamento) que é responsável por receber, interpretar e executar pedidos de provisionamento no recurso alvo.

O provisionamento, segundo a SPML, pode ser descrito, por exemplo, como o processo para gerenciar a alocação de recursos do sistema para funcionários, sócios e fornecedores, bem como para o gerenciamento de identidades ou políticas.

Programas utilizados para gerenciar objetos distribuídos (*e.g.*: identidades) provêem facilidades para aumentar a variedade de sistemas conectados. A automatização dos processos de gerenciamento de usuários e ou permissões facilita a configuração de dispositivos em grandes organizações. Visto que as organizações progressivamente expandem e consolidam seus sistemas, expondo-se a consumidores e parceiros na Internet, a necessidade de um padrão que permita a configuração de objetos e sistemas dentro e entre organizações é fato.

O objetivo da especificação SPML é facilitar a interação e a integração de sistemas fornecendo serviços de provisionamento (*i.e.*: configuração). Os serviços permitem que as organizações gerenciem as contas de usuários de forma segura em Serviços Web ou em aplicações internas, por exemplo.

A SPML pode ser usada para atender e executar pedidos de configuração de objetos. A configuração, segundo a SPML, refere-se à preparação de algum determinado sistema antes deste executar uma atividade específica. Inclui-se neste processo a configuração de serviços digitais como, por exemplo, contas de usuários, privilégios de acesso em sistemas, redes e aplicações, e também, possivelmente, a configuração de recursos como celulares.

A seguinte definição tem sido usada pelo PSTC (*OASIS Provisioning Services Technical Committee*) para definir o termo *provisioning*: *"Provisionamento é a automação de todos os passos exigidos para gerenciar (e.g.: configurar, corrigir, revogar) usuários ou direitos de acesso a sistemas, ou dados relativos a serviços publicados eletronicamente"* [30].

Na SPML assume-se a existência de um serviço cujo propósito é o gerenciamento e a execução de pedidos de provisionamento. Uma determinada Autoridade Solicitante (*e.g.*: *principal*) envia ao serviço de provisionamento (*i.e.*: provedor) um pedido na forma de um documento XML, formatado de acordo com a SPML, solicitando a execução de alguma

operação. Baseado em um modelo de execução pré-definido, o serviço de provisionamento coleta as informações de dentro do documento SPML, e verificando-se que o pedido é válido (*e.g.*: analise sintática), executa as ações de provisionamento no objeto alvo. As ações são executadas em objetos (*e.g.*: recursos) que o serviço de provisionamento administra.

As entidades básicas da SPML são: o *principal* (*Requesting Authority* – RA), o provedor de serviço (*Provisioning Service Provider* – PSP), os objetos do serviço (*Provisioning Service Objects* – PSO), e o serviço alvo (*Provisioning Service Target* – PST).

O exemplo da Figura 3.4 expõe o fragmento de uma mensagem SPML solicitando a adição de um novo objeto. A mensagem foi enviada por um *principal*, sendo destinada ao provedor de serviço. A solicitação tem por objetivo adicionar um novo usuário em um sistema de cadastro.

```
1. <addRequest requestID="127" targetID="target2">
2.          <containerID ID="ou=Development, org=Example"/>
3.          <data>
4.                <Person cn="joebob" firstName="joebob"
5.          lastName="Briggs" fullName="JoeBob Briggs">
6.                <email>joebob@example.com</email>
7.                </Person>
8.          </data>
9. </addRequest>
```

Figura 3.4: Pedido SPML enviado ao serviço de provisionamento

A linha *1* (Figura 3.4) transporta o elemento identificando o tipo da mensagem, classificando a mesma como uma solicitação para a adição de um novo objeto. A linha *9* indica o fim do escopo da mensagem. Adicionalmente a linha *1* transporta o elemento *requestID* usado para controlar o pedido efetuado, sendo que a resposta da solicitação deverá retornar o mesmo valor. Não havendo assim a necessidade de confiar no protocolo de transporte. O elemento *targetID*, linha *1*, é usado para identificar o objetivo (PST) dentro do domínio de gerência do serviço de provisionamento.

A linha *2* (Figura 3.4) contém o elemento *containerID* usado para identificar um objeto que existe dentro do espaço de nomes do PST (*i.e.*: algum container supostamente já criado). Os demais elementos da mesma linha são as especificidades do container onde o objeto *Person* (linha *4*) deve ser adicionado. Entre as linhas *4* e *7* estão os elementos que fornecem as informações que caracterizam o objeto *Person*, transportando os dados que

deverão ser adicionadas no container.

A linha *1* (Figura 3.5) carrega o elemento que caracteriza a resposta da solicitação feita na Figura 3.4, bem como o mesmo valor *requestID* enviado na solicitação, e um elemento indicando o sucesso do processo de adição do objeto *Person* ao container. A linha *3* transporta o identificador do novo objeto criado com intuito de facilitar, por exemplo, acessos, atualizações de dados, exclusão, ou outros procedimentos que podem vir a ser executados futuramente sobre o objeto *Person*. Entre as linhas *5* e *8* é retornada a representação XML do novo objeto criado.

Um possível caso de uso da especificação SPML é mostrado no Apêndice A.9, bem como um exemplo de uso da mesma pela empresa Oracle.

```
1.    <addResponse requestID=”127” status=“success">
2.          <pso>
3.          <psoID ID=”2244” targetID=“target2”/>
4.                <data>
5.                      <Person cn=”joebob” firstName=”joebob”
6.                      lastName=”Briggs” fullName=”JoeBob Briggs”>
7.                      <email>joebob@example.com</email>
8.                      </Person>
9.                </data>
10.         </pso>
11.   </addResponse>
```

Figura 3.5: Resposta SPML retornada ao solicitante

3.4 Conclusão

Apesar do grande número de especificações, estas não resolvem completamente todos os problemas de segurança e interoperabilidade existentes, porém, são um passo para diminuir esta lacuna.

Como citado nas muitas especificações, estas formam mais um “bloco de construção” na edificação de Serviços Web mais seguros e interoperáveis.

O capítulo seguinte mostra algumas abordagens em modelos de segurança e controle de políticas usadas em sistemas locais e de larga escala. Modelos estes que visam garantir que as regras e políticas sejam seguidas e cumpridas por quem está sujeito às mesmas.

Capítulo 4

Arquiteturas de Controle de Acesso e Linguagens para Aplicação de Políticas

4.1 Introdução

Com o aumento na adesão de uso da AOS, aumenta-se também a necessidade por padrões que governem um perfil seguro de comunicação e interação entre as entidades da arquitetura. O capítulo anterior mostrou alguns dos principais padrões citados na literatura sendo, também, alguns destes muito utilizados nas implementações dos Serviços Web.

Gerenciar as identidades e direitos de acesso de *principals* que fazem uso de sistemas locais (*e.g.*: mesma rede, mesmo terminal) pode ser fácil e pouco propenso a erro, porém, quando o ambiente é de grande escala – sistemas distribuídos geograficamente – a dificuldade para se gerenciar o ambiente aumenta consideravelmente.

Diferentes ambientes, geralmente, usam diferentes tecnologias para implementar seus sistemas de controle de acesso, característica esta que influencia diretamente para o aumento da complexidade de gerenciamento do ambiente.

Este capítulo mostra alguns modelos de autenticação e autorização que visam oferecer suporte aos sistemas de controle de acesso em sistemas distribuídos. Controles estes que visam garantir que as políticas impostas pela corporação ou provedor de serviço, referente a determinado recurso ou sujeito, sejam seguidas e cumpridas pelas entidades responsáveis por tal aplicação.

Adicionalmente, serão apresentadas neste capítulo uma linguagem padronizada empregada na escrita das políticas de controle de acesso, e uma especificação usada na troca de informações de maneira neutra a plataforma.

4.2 Modelo SPKI/SDSI

O modelo SDSI (*Simple Distributed Security Infrastructure*) [31] projetado no MIT (*Massachusetts Institute of Technology*) pelos professores Ron Rivest e Butler Lampson é uma estrutura de segurança, cuja meta principal é facilitar a construção segura e escalável de sistemas distribuídos.

O modelo SPKI (*Simple Public Key Infrastructure*) [32] proposto por Carl Ellison projeta uma estrutura simples para um modelo de autorização flexível. Os dois projetos foram integrados e os nomes unidos para formar o SPKI/SDSI. Unindo um nome local e uma chave pública gera-se um identificador global único, visando facilitar a administração de nomes e autorizações [33].

O modelo SPKI/SDSI emprega uma Infra-estrutura de Chave Pública igualitária (*i.e.*: não hierárquica), onde os *principals* são chaves públicas, podendo assim emitir certificados como qualquer outro *principal*. Isto facilita a construção de comunidades de maneira distribuída, não necessitando de uma terceira parte confiável (*i.e.*: *Trusted Third Party* – TTP) como a CA (*Certificate Authority*) da ICP X.509.

Os principais tipos de certificados nesta estrutura são [33]: o Certificado de Nome e o Certificado de Autorização.

4.2.1 Certificado de Nomes

Certificados de nomes SPKI/SDSI ligam nomes locais a chaves públicas ou a outros nomes não locais. Esse é composto por quatro campos:

- ***Emissor:*** a chave pública que emite e assina o certificado.
- ***Identificador/Nome:*** define um nome local no espaço de nomes do emissor.
- ***Sujeito:*** chave pública ou um nome não local que será ligado ao identificador.
- ***Especificação de Validade:*** período de tempo durante o qual o certificado é valido.

O SPKI/SDSI adota o espaço de nomes local ao *principal*, ajudando assim a tornar o esquema escalável. Ou seja, um usuário não tem de garantir que os nomes definidos localmente são únicos dentro do espaço global de nomes. A estratégia é associar o nome local

a chave pública do *principal* detentor dos nomes locais. Este esquema permite que a infra-estrutura seja estendida ao espaço global de nomes.

Esta estrutura aceita que o *principal* possua mais de um par de chaves (pública e privada), assim o *principal* pode atuar em mais de um cargo (*e.g.*: papel), tendo em cada cargo um par de chaves diferentes.

O certificado de nome pode ser representado por:

$K_{x\ \text{“nome”}} \rightarrow K_{\text{“a\_definir”}}$

Onde, “$K_{x\ nome}$” significa que “K” é a chave pública do *principal* “x”, e “nome” é referente ao espaço de nome local. E, “$K_{a\_definir}$” representa a chave pública que será associada a “nome”.

Temos então:

$KB_{Mother} \rightarrow K_{MARY\_SMITH}$

Onde “KB” é a chave pública do Bob que emitiu um certificado de nomes para sua mãe, referenciando-a como “Mother” em seu espaço de nome local. Sendo “K_{MARY_SMITH}” a chave pública da mãe do Bob.

4.2.2 Certificado de Autorização

Um certificado de autorização SPKI/SDSI concede uma autorização específica, do emissor do certificado para o sujeito do certificado (*i.e.*: *principal*). Este é composto de cinco campos:

- ***Emissor:*** chave que assina o certificado – *principal* concedendo a autorização.
- ***Sujeito:*** *principal* ou grupo de *principals* recebendo a autorização.
- ***Etiqueta (Tag):*** autorização sendo concedida do emissor para o sujeito.
- ***Bit de Delegação:*** se este *bit* estiver configurado para verdadeiro, o sujeito pode conceder os direitos recebidos na etiqueta de autorização a outros *principals*. Se este *bit* for falso, o emissor não pode delegar (repassar) os direitos recebidos a terceiros.

- ***Especificação de Validade:*** período de tempo durante o qual o certificado é valido.

Um certificado de autorização em que o emissor "K" concede a autorização especificada na etiqueta "T", para o sujeito "S", com o *bit* de delegação "d", e a especificação de validade "V", é representado pela quíntupla "(K, S, T, d, V)".

Considerando que o sujeito "S1" é o receptor de um certificado de autorização delegado por "K1", temos: (K1, S1, T1, d1, V1). O sujeito receptor deseja conceder o certificado para o sujeito "S2", então "S1" agora é o emissor (S1 = K2), temos então (K2, S2, T2, d2, V2). Este procedimento é possível, desde que o *bit* de delegação "d1" esteja com o valor verdadeiro.

A Figura 4.1 mostra o exemplo de um certificado de autorização SPKI/SDSI codificado em *S-Expression* (Apêndice B.1.3). *S-Expression* é uma linguagem para descrever estruturas complexas de dados – uma variação de LISP (*LISt Processing*) – sendo utilizada para representar os dados nos campos dos certificados SPKI/SDSI.

```
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.corporation.com/researcher (* set read)))
14.           (valid
15.                 (not-before "2008-04-21_23:59:59")
16.                 (not-after "2008-07-18_00:00:00"))
17.           (comment This statement applies to this principal at corporation.com))
```

Figura 4.1: Certificado de Autorização SPKI/SDSI codificado em *S-Expression*

As linhas *3* a *9* (Figura 4.1) definem o emissor do certificado (*rights' grantor*), sendo que as linhas *7* a *9* identificam a chave pública que representa o emissor. As linhas *10* e *11* contêm o sujeito (*rights' grantee*) do certificado. A presença da linha *12* indica que o certificado pode ser delegado. A linha *13* contém a autorização concedida pelo certificado, isto é, os direitos que o sujeito terá sobre o objeto. No caso, o sujeito tem o direito de leitura (*read*) em todos os arquivos do diretório *researcher* na URI (servidor) *www.corporation.com*.

As linhas *14* a *16* contêm o período de validade do certificado.

Na Figura 4.1, os campos contendo a assinatura do certificado foram omitidos, sendo mostradas somente as partes principais do certificado de autorização.

4.2.3 Fluxo de Autorização Baseado em Delegação

O fluxo de autorização em SPKI/SDSI ocorre através da delegação de direitos, ou seja, um *principal* devidamente autorizado pode delegar os direitos – ou parte dos mesmos – para outro *principal*, formando assim as cadeias de certificados de autorização SPKI/SDSI. Considerando que o *principal* receptor dos privilégios de acesso é confiável, o mesmo somente repassará os direitos para outro *principal* que o mesmo julgar ser confiável.

A Figura 4.2 mostra o exemplo do fluxo de autorização baseado em delegação utilizado pelo SPKI/SDSI. O Servidor, sendo o detentor do recurso, cria um certificado de autorização, assina o mesmo e o envia a um *principal* (evento *1*). De posse do certificado, e o mesmo sendo delegável, o *principal* receptor pode repassar os direitos recebidos a outros *principals* (evento *2*). O *principal* beneficiário dos direitos (evento *3*) deve enviar todos os certificados ao guardião do recurso (Servidor, evento *4*) para ter seu acesso liberado. Ou seja, apresentar a cadeia de certificados de autorização.

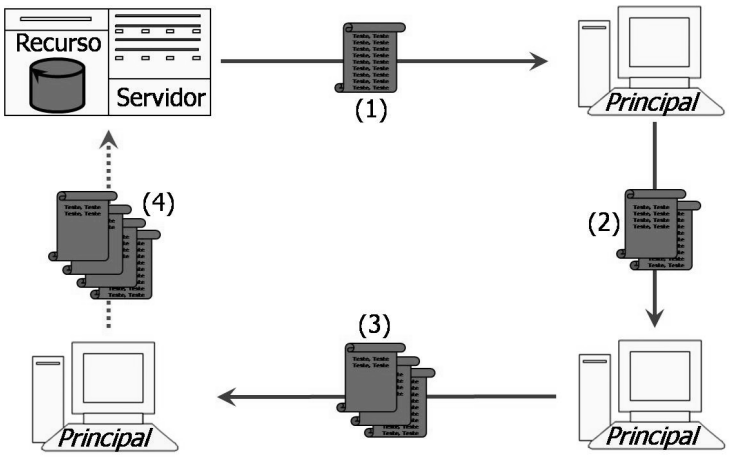

Figura 4.2: Fluxo de Autorização Baseado em Delegação

O Apêndice B.1 mostra como é a dinâmica de acesso em SPKI/SDSI, como a infra-estrutura tolera faltas, como é definido o prazo de validade dos certificados, e como uma lista

de controle de acesso (*Access Control List* – ACL) pode ser implementada em SPKI/SDSI.

4.2.4 Certificado Resultado dos Certificados

Em SPKI/SDSI, após o guardião verificar que o *principal* está autorizado a executar a operação solicitada, através da inspeção da cadeia de certificados, esse pode emitir um certificado de autorização próprio. Este certificado resume o conjunto de direitos de acesso concedidos durante a delegação dos certificados de autorização. O CRC (*Certificate Result Certificates*) é derivando da cadeia de certificados apresentada pelo *principal*.

O CRC apresenta os seguintes campos:

- ***Emissor:*** a chave do guardião.
- ***Sujeito:*** a chave do *principal*.
- ***Bit de Delegação:*** será configurado de acordo com a interseção de todos os *bits* da cadeia de certificados de autorização.
- ***Etiqueta (Tag):*** contém a interseção dos direitos codificados nos certificados da cadeia.
- ***Especificação de Validade:*** contém a interseção dos períodos de validade da cadeia de certificados autorização.

O CRC é assinado com a chave do guardião e retornado para o *principal*, podendo ser utilizado para realizar pedidos futuros. Assim, menos largura de banda da rede é utilizada, como menos certificados são transmitidos entre guardiões e *principals*.

Cadeias de certificado podem consistir em vários certificados, não há limite para o número máximo de certificados que podem estar presentes em uma cadeia de certificados. Porém, quanto maior a cadeia maior a dificuldade de gestão da mesma.

4.3 Modos de operação *Push* e *Pull*

Os modos de operação *push* e *pull* caracterizam os perfis de *principal* ativo e passivo, respectivamente.

No perfil ativo (*push model*, Figura 4.3), o *principal* pode ser o próprio usuário que

busca o arquivo WSDL no diretório UDDI (evento *1*), implementa o serviço cliente, e entra em contato com o IdP (*Identity Provider*, evento *2*). Tendo adquirido a credencial de acesso do IdP, o *principal* monta e envia o pedido para o Provedor de Serviço (PS, evento *3*), sendo capaz de interagir enviando os atributos necessários ao PS, e ou interpretando as respostas retornadas pelo mesmo.

No perfil passivo (*pull model*, Figura 4.3) o *principal* envia uma mensagem para o provedor de serviço solicitando acesso ao recurso (evento *2*) e uma prova de autenticação (*e.g.*: mensagem assinada pelo *principal*, ou uma credencial emitida pelo IdP no processo de autenticação, evento *1*), sendo o PS responsável por entrar com contato com o IdP (evento *3*) ou outros serviços de atributos, para solicitar mais informações sobre o *principal*.

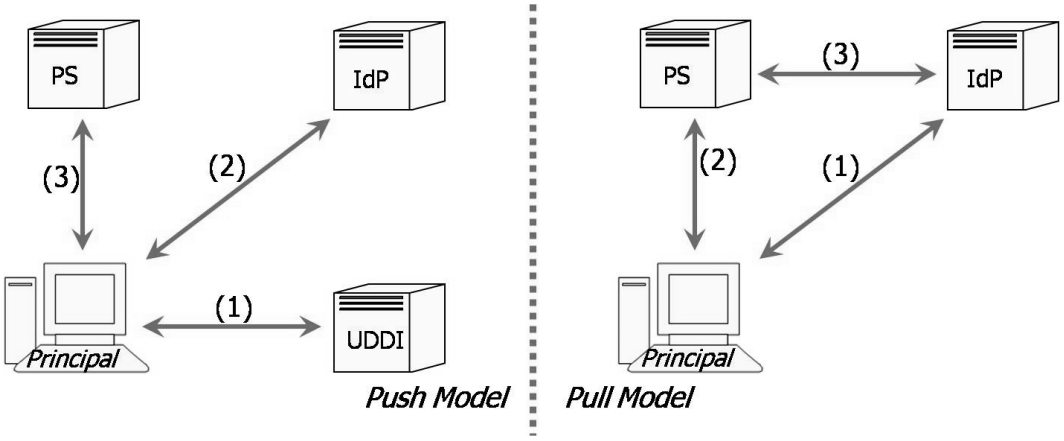

Figura 4.3: Modos de Operação *Push* e *Pull*

4.4 Modelo de Informação de Políticas (PCIM)

O modelo PCIM (*Policy Core Information Model*) [34] define estruturas genéricas para a criação de políticas, fornecendo um modo de descrever condições e ações usadas para subsidiar a aplicação das políticas em determinado ambiente. As definições de classes de políticas neste modelo, bem como as associações entre as mesmas, são suficientemente genéricas para representar políticas relacionadas a qualquer objeto (*e.g.*: dispositivos de rede).

No modelo PCIM são definidas duas hierarquias de classes de objetos: 1) classes estruturais, representando informações e controle de políticas; 2) classes de associação,

indicando quais instâncias das classes estruturais estão relacionadas.

As políticas representam metas (*goals*) e objetivos (*objectives*) de negócios. Pode-se descrever o objetivo do negócio como um acordo em nível de serviço (*Service Level Agreement* – SLA). Uma breve descrição do SLA pode ser encontrada no Apêndice B.2.

Uma rede controlada por políticas pode ser analisada do ponto de vista de máquinas de estado. As políticas são usadas para controlar em qual estado o dispositivo deveria estar ou determinar qual estado é permitido para que o mesmo esteja, por exemplo.

As políticas podem ser usadas independentemente, ou agregadas em grupos de políticas para executar funções mais complexas. Políticas independentes são chamadas regras de políticas podendo avaliar uma simples expressão (*e.g.*: valor booleano). Grupos de políticas são agregações de regras de políticas ou outros grupos de políticas. Grupos podem avaliar interações complexas entre objetos que possuem associações complexas. O objetivo do modelo PCIM é fornecer suporte para políticas independentes e agregadas.

Exemplo de uma agregação de regras pode ser uma política de *logon* ativando o acesso a aplicação de forma segura e configurando as conexões de rede, baseando-se em uma combinação da identidade do usuário, sua localização na rede (*e.g.*: endereço de IP – *Internet Protocol*), tipo de *logon*, hora do dia, etc.

Tipicamente, o PCIM é implementado utilizando dois componentes principais: 1) o servidor de políticas, responsável por passar as informações de política para os dispositivos de rede e aplicações associadas; 2) o cliente, responsável por aplicar as políticas recebidas do servidor sobre um dispositivo ou aplicação.

4.4.1 Modelo Outsourcing (terceirização)

Este modelo de controle de políticas [35] está baseado em duas entidades principais, o ponto de aplicação de política ou guardião (*Policy Enforcement Point* – PEP), e o ponto de decisão de política ou monitor de referência (*Policy Decision Point* – PDP). Estas entidades interagem trocando mensagens a cada solicitação feita por um *principal*.

A interação (Figura 4.4) ocorre no momento que o *principal* envia uma solicitação de acesso ao recurso (evento *1*), esta solicitação é interceptada pelo PEP. O guardião do recurso encaminha um pedido de avaliação para o PDP (evento *2*). Após receber o pedido, o PDP com base em seu repositório de políticas (evento *3*), na identidade do *principal* e no recurso alvo,

toma uma decisão (evento *4*) e retorna ao PEP (evento *5*). O PEP aplica a decisão (evento *6*) de permitir (evento *7*) ou negar o acesso do *principal* de acordo com a resposta recebida do PDP.

Neste modelo, a comunicação entre PEP e PDP deve estar sempre estabelecida, pois havendo uma falha da mesma, o PEP fica incapacitado de tomar decisões sobre os pedidos que chegam dos *principals*. Toda a carga de processamento, análise e interpretação de políticas fica a cargo do PDP. Um caso de uso com o modelo *outsourcing*, de acordo com a RFC 2753 (*Request For Comments*) [36], é apresentado no Apêndice B.3.

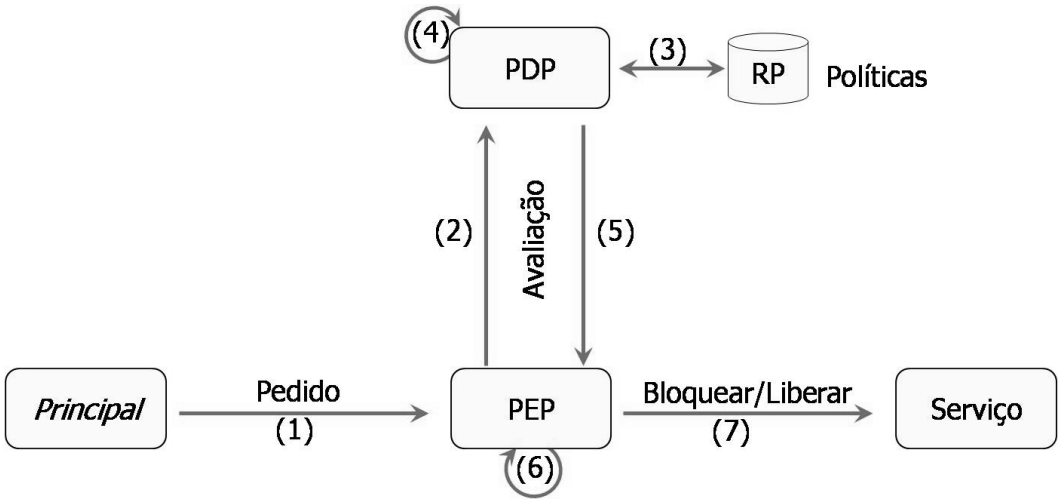

Figura 4.4: Modelo *Outsourcing*

4.4.2 Modelo Provisioning (configuração)

Este modelo de controle de políticas [35] também está baseado em duas entidades principais, o PEP e o PDP, porém nesta abordagem os dois não necessitam estar sempre conectados um ao outro para que as decisões de políticas sejam tomadas.

Quando o PEP é inicializado, este troca mensagens com o PDP, passando suas características e em troca espera receber as políticas correspondentes. Após receber as políticas o PEP as armazena em um repositório local, portanto as decisões e aplicações de políticas podem ser feitas de maneira mais autônoma – sem a necessidade da troca constante de mensagens com o PDP.

O PDP ainda fica responsável por receber e interpretar as novas políticas, sendo responsável por repassar as alterações de seu repositório para o PEP. Interações assíncronas entre PEP e PDP ainda podem ocorrer, visto que novas políticas podem ser criadas ou políticas existentes podem ser atualizadas.

Mesmo após o PEP já estar configurado com todas as políticas, novos recursos podem ser instalados no sistema ou *principals* desconhecidos podem requisitar acesso aos recursos, fazendo o PEP interagir com o PDP para obter uma determinada política, ou uma avaliação da mesma.

A Figura 4.5, de maneira simples, mostra a interação entre as principais entidades da arquitetura, expondo o fluxo de mensagens entre as mesmas. Após o PEP ser inicializado, este envia um pedido (evento *1*) ao PDP solicitando as políticas correspondentes a serem configuradas. O pedido deve, preferencialmente, estar acompanhado de uma identificação, a qual será usada pelo PDP para gerar o conjunto de políticas adequadas para cada PEP. Isto em caso de um PDP atender mais de um tipo de PEP.

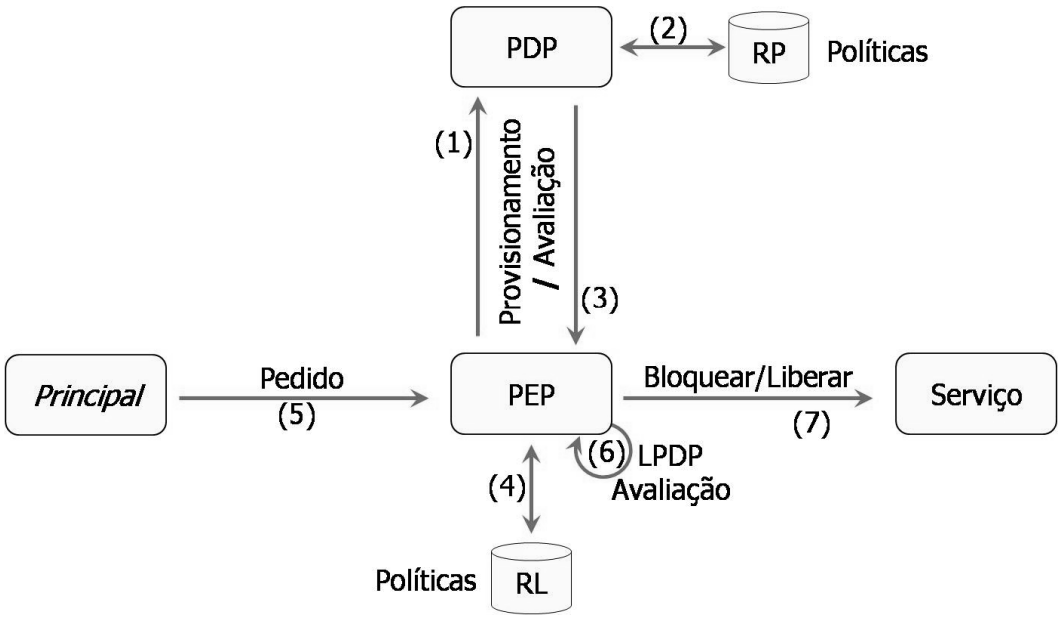

Figura 4.5: Modelo *Provisioning*

Estando o procedimento de verificação correto, e havendo a possibilidade de atender ao pedido do PEP, o PDP recupera o conjunto de políticas correspondentes (evento *2*, Figura 4.5) ao PEP em questão e as envia em uma mensagem de resposta (evento *3*). Ao receber a

mensagem, o PEP armazena as políticas em um Repositório Local (RL, evento *4*).

O *principal* solicita acesso ao serviço enviando uma requisição ao PEP (evento *5*, Figura 4.5). Como neste momento o PEP já está configurado com suas políticas, este toma a decisão referente à autorização localmente (evento *6*), em um LPDP (*Local* PDP). Caso a avaliação tenha sucesso o PEP libera o acesso ao serviço (evento *7*).

O Apêndice B.4 mostra o uso do modelo *provisioning* de acordo com o protocolo COPS-PR (*Common Open Policy Service Usage for Policy Provisioning*) [37], bem como um caso de estudo implantando o mesmo.

4.5 Linguagem de Marcação para Asserções de Segurança (SAML)

A especificação SAML (*Security Assertion Markup Language*) [38] define uma estrutura XML para a troca de informações entre parceiros de negócios, estas informações são referenciadas como asserções de segurança.

As asserções expressam informações sobre *principals* de forma padrão. Informações que outras aplicações, através de relacionamentos pré-estabelecidos, podem confiar. O emissor de tal asserção pode ser descrito como um Provedor de Identidade (*Identity Provider* – IdP). Ou seja, uma terceira parte confiável como o STS.

O IdP é o serviço pertencente a um domínio confiável, certificando que as informações sobre um *principal* são verdadeiras, ou seja, o IdP garante que o *principal* foi autenticado e possui determinados atributos associados através da emissão de uma asserção SAML. As asserções podem ser embutidas no corpo de uma mensagem SOAP e transportadas usando os protocolos de comunicação subjacentes (*e.g.*: HTTP, SMTP).

A entidade do sistema que fará uso da asserção é descrita como Provedor de Serviço (*Service Provider* – SP). O SP é o sistema ou domínio que por relações pré-estabelecidas confia no IdP fornecedor da asserção. Como o nome sugere, o SP é capaz de prover serviços, porém, embora este confie no conteúdo da asserção recebida, as políticas de acesso local ao SP definem quais serviços o *principal* pode acessar.

A SAML define três tipos de expressões que podem ser criadas por um IdP, inseridas em uma asserção e associadas a um *principal*:

- ***Expressões de Autenticação:*** define quem emitiu a asserção, garantindo que o

principal sendo referenciado foi autenticado em um determinado período de tempo.

- ***Expressões de Atributo:*** o *principal* da asserção é associado com os seus atributos. Cada ambiente define atributos específicos de acordo com as necessidades do contexto (*e.g.*: IP de origem, qualificações do *principal*).
- ***Expressões de Decisão de Autorização:*** definem os direitos do *principal*.

Caso o provedor de serviço tenha recebido uma asserção de autenticação SAML de um *principal*, esse pode contatar um serviço de atributos e solicitar uma asserção de atributos do *principal* em questão. Somente após receber a asserção de atributos, o provedor se dirige ao serviço de autorização (*i.e.*: monitor de referência), passando as asserções de autorização e de atributo, indagando se o *principal* tem permissão de executar determinada atividade em certo recurso. Asserções podem conter um identificador temporário, preservando a privacidade de seu portador e ainda assim permitindo um relacionamento persistente com o mesmo.

Um exemplo de um pedido de autenticação SAML, bem como o uso do *Single Sign On/Off* pode ser encontrado no Apêndice B.5.

4.6 Linguagem de Marcação para Controle de Acesso (XACML)

Umas das práticas utilizadas no gerenciamento de políticas de controle de acesso é a configuração de cada ponto de aplicação de políticas separadamente, bem como a escrita de políticas tão específicas quanto possível. Por conseqüência, este procedimento é suscetível a erro, devido a quantidade de recursos, *principals* e direitos a gerenciar.

Sendo assim, manter a visão administrativa consolidada das regras de políticas de todo o sistema distribuído (em uma corporação, por exemplo), na maioria dos casos, é impraticável. Por estes e outros motivos, é que surge a necessidade de uma linguagem comum para expressar políticas de segurança.

A linguagem de política, se aplicada em uma corporação, deve permitir o gerenciamento e aplicação de todos os elementos da política de segurança, em todos os mecanismos do sistema [39], independentemente da plataforma subjacente.

A XACML (*eXtensible Access Control Markup Language*) [40] estabelece uma metodologia para definir políticas de segurança para controle de acesso através da aplicação

do padrão XML. A especificação define um contexto de mensagens pedido/resposta, padronizando a comunicação entre as entidades que efetuam o controle de acesso do sistema (*i.e.*: PEP e PDP).

O contexto de mensagens pedido/resposta descreve como as consultas serão realizadas no serviço de avaliação de políticas e como deverão ser as respostas. As políticas têm a finalidade de controlar o acesso a recursos ou informações protegidas.

A partir do uso da XACML é possível definir políticas singulares aplicáveis a sujeitos, objetos, ações específicas e também definir políticas globais que se aplicam a todos os recursos de um sistema. A linguagem para escrita das políticas de controle de acesso define os direitos que o *principal* tem sobre o recurso.

A especificação faz uso das entidades PEP (*Policy Enforcement Point*), PDP (*Policy Decision Point*), Manipulador de Contexto (*Context Handler*), PAP (*Policy Administration Point*) e PIP (*Policy Information Point*).

O PEP é a entidade que intercepta o pedido vindo de algum *principal*, sendo responsável por repassar o pedido ao Manipulador de Contexto, e aplicar a decisão contida na resposta obtida (*i.e.*: liberando ou bloqueando o acesso ao recurso). O PDP recebe as mensagens vindas do Manipulador de Contexto, e toma uma decisão com base nas informações do *principal* e nas políticas recuperadas do PAP. O PAP é o ponto de administração das políticas de autorização, onde as políticas são escritas e armazenadas. Para fazer as avaliações, o PDP recupera as políticas do PAP.

A decisão tomada pelo PDP pode ser auxiliada por atributos solicitados ao Manipulador de Contexto, tais atributos são recuperados do PIP. O PIP gerencia informações sobre o sujeito, o ambiente e recurso (*e.g.*: *e-mail* do sujeito, informações sobre a carga de trabalho do sistema ou serviço específico), por exemplo.

O Manipulador de Contexto intercepta as mensagens entre PEP e PDP e as traduz para um formato nativo XACML, visto que as mensagens podem chegar de diferentes PEPs transportando diferentes conteúdos, talvez não entendíveis pelo PDP. O Manipulador de Contexto repassa ao PDP, caso necessário, atributos e ou informações extras recuperadas do PIP.

Pedidos sobre avaliação de políticas podem vir de outros domínios, encapsuladas em diferentes linguagens (*e.g.*: certificados, asserções SAML), por isso também é necessária a intermediação do Manipulador de Contexto.

A XACML define três principais elementos de política: regra (<*Rule*>), política (<*Policy*>) e conjunto de política (<*PolicySet*>).

- ***Regra:*** forma uma unidade básica de gerenciamento e pode ser reutilizada em várias políticas. A regra é composta, basicamente, de expressões *booleanas*. A regra não foi concebida com o objetivo ser avaliada sozinha, mas sim para formar a base para as decisões de autorização.
- ***Política:*** contém um conjunto de elementos do tipo regra, e um procedimento especifico para combinar os resultados de sua avaliação. A política é a unidade básica usada pelo PDP, e sendo assim destinada para formar a base das decisões de autorização.
- ***Conjunto de Políticas:*** é composto por um conjunto do elemento política ou outros conjuntos de políticas, e um procedimento específico para combinar os resultados de sua avaliação. Ou seja, fornece um meio de avaliar diferentes conjuntos de políticas e retornar uma única decisão.

Tanto o elemento política como o conjunto de política podem conter diferentes conjuntos de regras e políticas, respectivamente, podendo corresponder a diferentes decisões de controle de acesso. Assim, o uso de mecanismos específicos facilita a avaliação destes conjuntos de regras e políticas. Isto pode ser feito através do uso de algoritmos de combinação que avaliam os conjuntos de regras e políticas, combinando as múltiplas opções em uma única decisão.

Os Algoritmos de Combinação de Políticas definem as estratégias para se chegar a uma decisão de autorização. Os resultados individuais da avaliação de um conjunto de políticas são combinados por um dos seguintes algoritmos: *permit-overrides*, *deny-overrides*, *first-applicable*, *only-one-applicable-policy*.

Os Algoritmos de Combinação de Regras definem um procedimento para se chegar a uma decisão de autorização, dado os resultados individuais da avaliação de um conjunto de regras. Estes podem ser: *permit-overrides*, *deny-overrides*, *first-applicable*.

Para fornecer uma avaliação de autorização, deve ser possível combinar diferentes regras de política e retornar uma única decisão. Adotando o algoritmo de combinação *deny-overrides*, se alguma das verificações retornar *deny* (negar), o resultado final da combinação

será *deny*, por exemplo.

As políticas aplicáveis para um pedido de decisão particular podem ser compostas de uma série de políticas ou regras individuais. Por exemplo, em um sistema que armazene informações pessoais, o proprietário da informação pode definir certas regras de política permitindo a exposição das informações para outras empresas. Enquanto isso, a organização guardiã de tal informação pessoal pode definir outros tipos de regras (mais restritivas, por exemplo).

A XACML (Figura 4.6), baseada no modelo *outsourcing* (Seção 4.4.1), opera de acordo com os seguintes passos:

1: o PAP permite a escrita das políticas, tornando-as disponíveis para o PDP.

2: o *principal* envia um pedido de acesso para o PEP.

3: o PEP envia um pedido de avaliação de acesso para o Manipulador de Contexto em um formato de pedido nativo do PEP (*e.g.*: certificado, asserção SAML), opcionalmente incluindo atributos do sujeito (*e.g.*: CEP – Código de Endereçamento Postal), recurso (*e.g.*: nível de disponibilidade), ação (*e.g.*: leitura, escrita) e ambiente (*e.g.*: carga de trabalho do sistema).

4: o Manipulador de Contexto constrói um pedido XACML e envia este para o PDP.

5: o PDP pode solicitar algum atributo adicional referente ao sujeito, recurso, ou ambiente para o Manipulador de Contexto.

6: o Manipulador de Contexto solicita os atributos do PIP.

7: o PIP obtém os atributos descritos no evento *3*.

8: o PIP retorna os atributos solicitados para o Manipulador de Contexto.

9: opcionalmente, o Manipulador de Contexto adiciona os atributos do recurso.

10: o Manipulador de Contexto envia os atributos para o PDP avaliar a política.

11: o PDP retorna a reposta incluindo a decisão de autorização para o Manipulador de Contexto.

12: o Manipulador de Contexto traduz a mensagem recebida do PDP para o formato de resposta nativo do PEP e retorna esta para o mesmo.

13: o PEP executa a decisão e aplica as obrigações. Ou seja, se o acesso for permitido então o solicitante pode acessar o recurso, do contrario o acesso é negado. Quando o PDP avalia uma política ou conjunto de políticas contendo obrigações, este retorna as

obrigações para o PEP em resposta ao pedido de decisão.

Obrigações (elemento <*Obligations*>) são operações que devem ser executadas pelo PEP em conjunto com uma decisão de autorização [41]. Uma obrigação pode ser associada com uma decisão "*Permit*" ou "*Deny*". Por exemplo, pode ser exigido ao *principal* que obteve acesso a um recurso que este envie um *e-mail* para o proprietário do recurso, ou para que o PEP execute determinada operação.

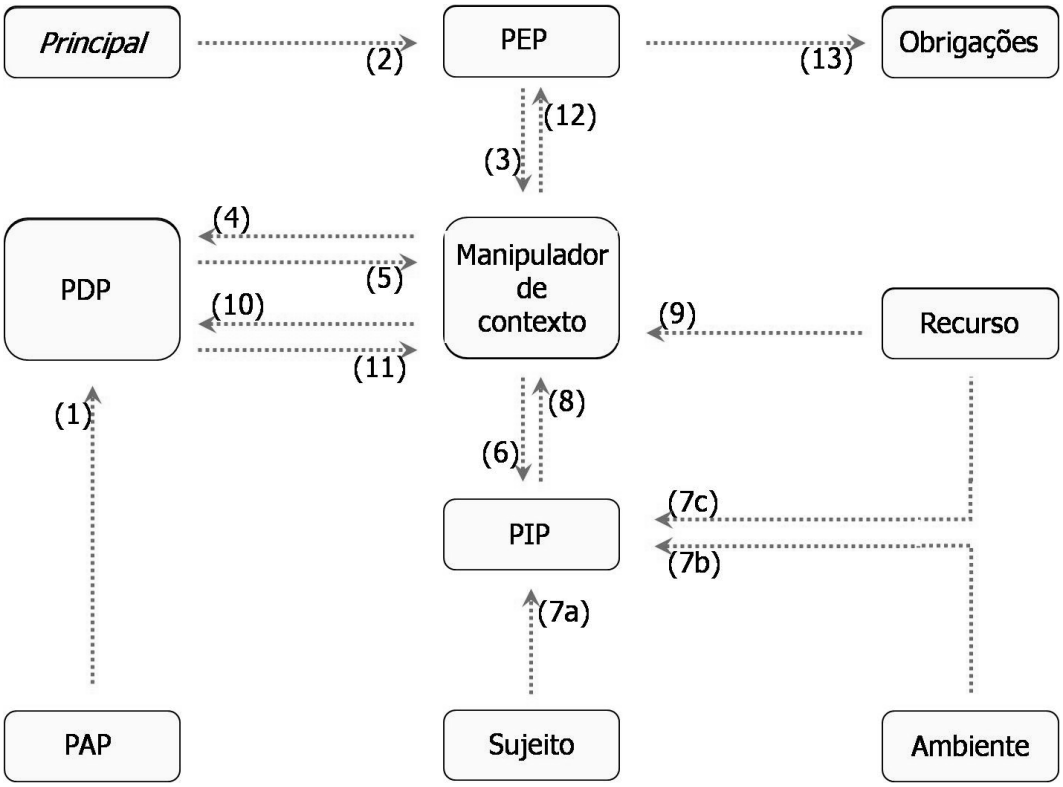

Figura 4.6: Modelo do fluxo de dados com as principais entidades da XACML

As mensagens trocadas entre as entidades PEP e PDP podem ser encapsuladas em asserções SAML. O formato da asserção é definido no documento que estende a especificação SAML, ou seja, o documento SAML 2.0 *profile of* XACML v2.0 [42].

4.7 Conclusão

O controle de acesso tradicional pode ser considerado mais simples de ser implementado, pois o mesmo, geralmente, é centralizado e possui políticas específicas para

cada *principal* e recurso. Porém, executar a mesma tarefa em contextos que fazem uso de atributos de *principals* desconhecidos e ou confiando em terceiros para tomar a decisão de autorização, pode ser considerado uma tarefa delicada.

Em certos ambientes, uma avaliação de políticas errônea, pode causar a revelação de informações sigilosas, ou mesmo permitir o acesso indevido a serviços restritos.

Por mais desconhecidas que sejam as informações ou identidades apresentadas a um guardião, se estas possuírem o endosso de uma entidade confiável, então tais informações podem ser consideradas dignas de credibilidade e honradas.

As especificações visam cada vez mais o compartilhamento de atributos dos sujeitos entre organizações, visando facilitar a administração de usuários, mas sem comprometer a identidade do portador de tal declaração.

O próximo capítulo expõe alguns dos muitos esforços de pesquisa voltados para área da Arquitetura Orientada a Serviço. Trabalhos realizados por pesquisadores, visando colaborar para o enriquecimento das arquiteturas e também da literatura.

Capítulo 5

Controle de Acesso em Arquiteturas Orientadas a Serviço

5.1 Introdução

Na literatura técnica, a segurança para AOS tem sido considerada por vários grupos de pesquisa. Os trabalhos a seguir mostram alguns esforços de pesquisa, bem como suas contribuições, para estimular ainda mais o uso da Arquitetura Orientada a Serviço.

5.2 Mediação de Confiança através de Serviços Web

O modelo sugerido em [2] propõe o uso de um mediador confiável (*i.e.*: um domínio intermediário) capaz de trabalhar com tecnologias de segurança diferentes, que normalmente não interoperam entre si. O modelo permite que ocorra interação entre o cliente de um domínio SPKI/SDSI e um serviço que aceita certificados X.509.

No caso de uso proposto (Figura 5.1), cada domínio (1, 2 e 3) é composto por um administrador que tem o papel de gerenciar a entrada e a saída de *principals*, bem como agrupar os mesmos de acordo com seus atributos de segurança (*e.g.*: credenciais, certificados). O administrador faz uso dos serviços STS e XKMS para realizar a respectiva troca e validação das credenciais. No exemplo (Figura 5.1), existem três domínios com diferentes mecanismos de autorização; o domínio 1 usa SPKI/SDSI; o domínio 3 usa X.509; no domínio 2 (*i.e.*: mediador) o administrador pode trabalhar com SPKI/SDSI e X.509.

Considere que o cliente “C” (domínio 1) solicita acesso a um serviço provido por “R” (domínio 3, evento *1*). Supondo que “R” somente aceite atributos de segurança emitidos pelo

administrador de seu próprio domínio (domínio 3), então este responde com um documento WS-Policy (evento *2*).

O cliente "C" é um *principal* SPKI/SDSI e não trabalha com credenciais do tipo X.509, sendo assim não poderá interagir com o administrador de "R". Deste modo, o cliente "C" entra em contato com administrador do domínio ao qual está vinculado (evento *3*) para tentar encontrar um caminho confiável que lhe conceda a autorização para acessar o serviço "R".

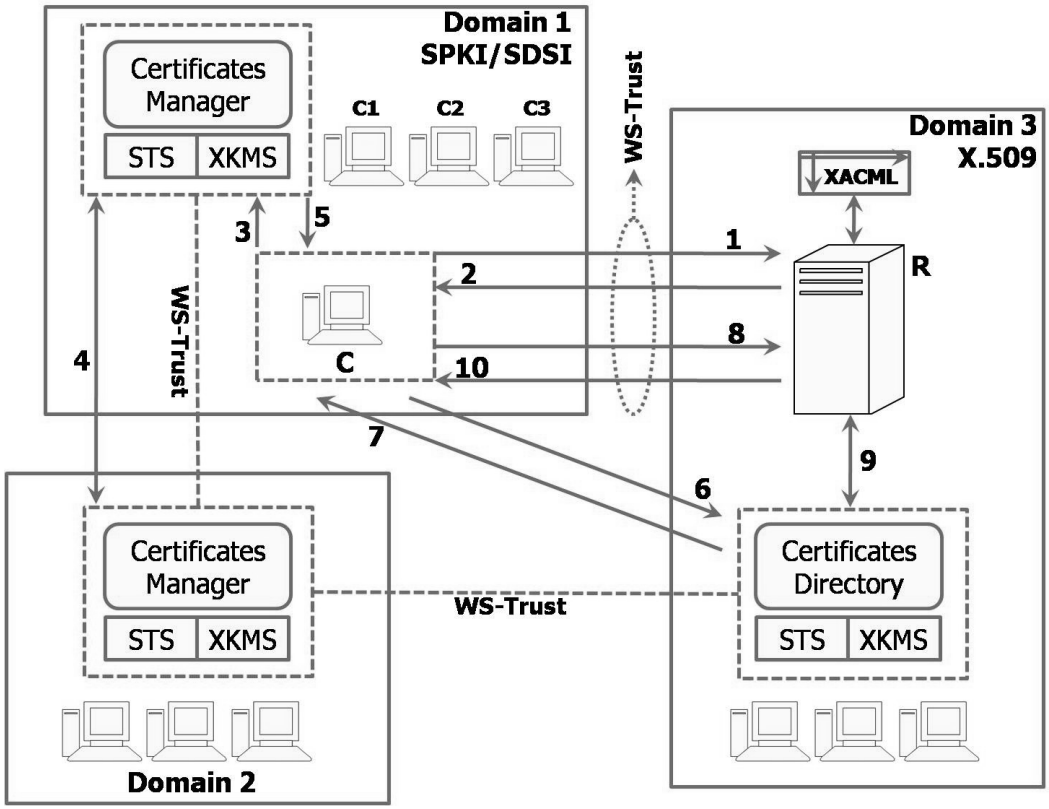

Figura 5.1: Arquitetura do modelo

Os administradores dos domínios de "C" e "R" não possuem relações de confiança entre si (WS-Trust, Figura 5.1). Então, o administrador de "C" deve efetuar uma busca nos domínios com os quais possui relações de confiança (evento *4*) para encontrar um caminho confiável (*e.g.*: cadeia de certificados) que o levará ao administrador de "R" (procedimento intermediado pelo domínio 2).

O administrador de "C" solicita ao STS do domínio 2 (evento *4*) atributos para

estabelecer a comunicação com o administrador de “R”. O administrador do domínio 1, por meio do seu STS, provê ao cliente “C” todas as credenciais necessárias (evento *5*), de forma que o cliente possa se comunicar com o STS do administrador de “R” (evento *6*). O administrador de “R” analisará a credencial informada verificando se a mesma é válida, para então prover as credenciais solicitadas pelo cliente (evento *7*).

De posse da credencial, o cliente envia uma resposta para o domínio 3 (evento *8*). O serviço “R” necessita confrontar a credencial com as políticas aplicadas ao recurso (políticas escritas em XACML), e realizar a validação da credencial através do diretório de certificados (XKMS, evento *9*). Somente então o cliente obtém acesso ao recurso (evento *10*).

O administrador de “C” é responsável por proporcionar a seus clientes todos os meios necessários para este se comunicar diretamente com o proprietário dos direitos, neste caso o administrador de “R” (evento *6*).

As políticas de acesso do sistema estão escritas em XACML, estando o controle de acesso sobre a responsabilidade do PEP e do PDP. A Figura 5.2, mostra como ocorre o processo de controle de acesso ao recurso.

Quando o cliente envia um pedido (evento *0*, Figura 5.2), sem a asserção de autenticação emitida por uma autoridade confiável, este pedido é interceptado pelo PEP que o remete a Autoridade de Autenticação (evento *1*), a qual solicita a validação da assinatura do pedido ao serviço de X-KISS (evento *2*). Se a assinatura for válida, a Autoridade de Autenticação gera uma asserção de autenticação SAML (evento *3*).

De posse da asserção de autenticação SAML, o PEP a repassa a Autoridade de Atributos (evento *4*) para obter uma asserção de atributos SAML do cliente (evento *5*). O PEP remete a asserção de autenticação e a asserção de atributos SAML para o PDP (evento *6*), de forma que este pode determinar se acesso deve ser concedido ou negado.

O PDP solicita ao serviço de X-KISS (evento *7*) a validação das assinaturas nas asserções. Sendo válidas as assinaturas, o PDP confronta as asserções SAML com as políticas do sistema (evento *8*) e retorna uma asserção de autorização SAML para o PEP (evento *9*). O PEP de posse da asserção de autorização SAML, determina qual informação o cliente pode obter do recurso protegido (evento *10*).

A arquitetura do protótipo fez uso de especificações usadas em Serviços Web para manter o mesmo interoperável e extensível, possibilitando que diferentes domínios, com diferentes tecnologias de segurança alcancem um ambiente de interação confiável.

De acordo com a arquitetura proposta, pode ser necessário navegar (*i.e.*: procurar) em mais de um domínio para se obter um "caminho confiável" entre o domínio de origem do cliente, ligando este ao provedor do serviço. Pois, somente assim se obtém a credencial que fornece os direitos de acesso ao serviço.

O uso do modelo *outsourcing* impõe um número significativo de trocas de mensagens no processo de avaliação de políticas. Nesta abordagem o PDP é a entidade responsável por traduzir asserções SAML para o contexto XACML, sendo que o Manipulador de Contextos deveria implementar esta função.

O PEP, entidade que deveria ser responsável por permitir ou negar o acesso ao recurso, também tem sua carga de trabalho ampliada, tendo de recuperar asserções SAML de entidades externas. Funcionalidade esta que deveria estar sobre a responsabilidade do Manipulador de Contexto.

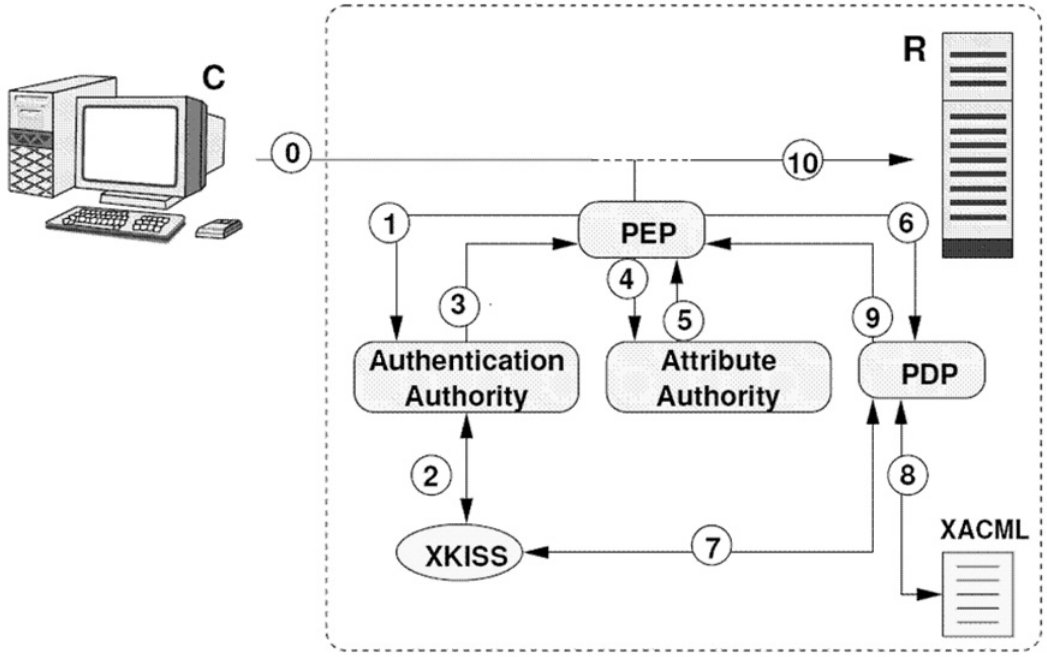

Figura 5.2: Procedimento de controle de acesso efetuado sobre o recurso [2]

A proposta depende de entidades externas aos domínios no processo de autorização, procedimento este que vai de encontro ao princípio de baixo acoplamento, característica esta almejada pelos Serviços Web.

A desvantagem desta abordagem é que o uso de mediadores implica em *overhead*

devido ao aumento do número de mensagens trocadas entre as entidades da arquitetura, conseqüentemente aumentando o tempo para se conseguir a credencial correta (*e.g.*: asserção SAML, certificado).

5.3 Transposição de Autenticação em AOS Através de Identidades Federadas

O trabalho apresentado em [3] faz uso da especificação SAML para transportar asserções de autenticação e atributos, as quais possuem declarações sobre clientes situados em domínios que trabalham com diferentes tecnologias de segurança (*e.g.*: SPKI, X.509). O modelo provê a transposição da autenticação e a conseqüente autorização descentralizada, juntamente com a transposição de atributos do cliente para facilitar as relações interdomínios.

A arquitetura pressupõe o uso de clientes autenticados localmente (*i.e.*: de acordo com a estrutura local, *e.g.*: usuário e senha, certificados) e do conceito de identidades federadas, que permite aos clientes estarem associados a mais de um domínio possuindo assim contas diferenciadas em cada um. Conseqüentemente, por possuírem uma conta em cada domínio, os clientes possuem diferentes atributos espalhados em cada um destes (*e.g.*: *e-mail*, telefone, endereços de casa ou trabalho).

Com a autenticação local, visa-se a autorização de maneira distribuída, processo este baseado na asserção de autenticação apresentada pelo cliente, e auxiliado por atributos recuperados diretamente do domínio de origem do cliente. Assim, para se concretizar a autorização de acesso no provedor de serviços, parte-se da premissa que é necessário transpor o domínio de segurança do cliente, tanto para a autenticação quanto para os atributos.

O propósito desta abordagem é dar suporte à transposição de autenticação em sistemas distribuídos, mantendo para os clientes as propriedades de transparência e de fácil uso nos acessos a serviços. A arquitetura usa a abordagem de cliente passivo, em que este fornece apenas a sua identidade ou pseudônimo ao provedor de serviço, ficando este responsável por procurar nos parceiros da sua federação (*i.e.*: entidades com relações de confiança prévias) os atributos do cliente.

No cenário de avaliação foi considerado um sistema distribuído de larga escala, formando diferentes domínios, baseados em diferentes tecnologias (*e.g.*: *Kerberos*, X.509, SPKI/SDSI) que determinam diferentes controles de segurança no acesso aos seus serviços. A

portabilidade de credenciais entre os diferentes domínios não é garantida por padrão na abordagem proposta. A total transparência das diferenças e a transposição da autenticação diante de diferentes tecnologias de segurança são garantidas por conceitos introduzidos no modelo proposto.

Na abordagem sugerida, ao se fazer uso de identidades federadas, assume-se como premissa as relações de confiança entre domínios. Sendo assim, a Autoridade de Atributos (STS/IdP presente no domínio do provedor de serviço, Figura 5.3) deve conhecer os demais domínios no qual a identidade do cliente está cadastrada. Adicionalmente, sempre que o provedor de serviço necessitar, a Autoridade de Atributos local ao provedor deve fazer requisições e recuperar os atributos do cliente em outras Autoridades de Atributos (evento *7*, Figura 5.3), presentes em outros domínios.

No modelo proposto, cada domínio possui as entidades de autenticação (Serviço de Credenciais Seguras e Provedor de Identidade – STS/IdP) e de atributos (Serviço de Atributos e Pseudônimos – SAP) que respectivamente autenticam o cliente e fornecem atributos sobre o mesmo.

Quando o provedor de serviço recebe um pedido e este é incapaz de interpretar a credencial contida na mensagem (*e.g.*: asserção SAML), esta é repassada para o serviço STS/IdP no domínio do provedor. Este serviço tem a responsabilidade de traduzir a credencial para um formato entendível pelo provedor de serviço (caso o provedor de serviço presente no domínio aceite somente certificados X.509, por exemplo).

A entidade que decide entre permitir ou negar acesso (*i.e.*: PDP), implementada no provedor de serviço, fica responsável por requerer a tradução da credencial de segurança apresentada pelo cliente. Caso o PDP receba do PEP uma asserção de autenticação SAML e a política de proteção do provedor de serviço exija um certificado X.509 ou SPKI/SDSI, o PDP deve entrar em contato com o STS/IdP presente no domínio do provedor para que este realize a tradução da credencial.

O STS/IdP é responsável por procurar e solicitar atributos do cliente nos domínios onde o mesmo está cadastrado. Os atributos podem ser necessários para auxiliar o PDP em uma tomada de decisão, ou para o STC (Serviço Tradutor de Credenciais) criar a nova credencial entendível e aceita pelo provedor de serviços.

O cliente somente obtém uma resposta do serviço perante a apresentação de um pedido assinado o qual contenha a credencial exigida pela estrutura de segurança do domínio

do provedor, e esteja assinada por uma terceira parte confiável. No caso, o STS/IdP do domínio do provedor de serviços. O STS/IdP no domínio de origem do cliente autentica e realiza a emissão, troca e validação de credenciais de segurança para o domínio do provedor de serviços.

Para ilustrar a dinâmica do processo de transposição na solução proposta, uma requisição de serviço, envolvendo dois domínios distintos, é mostrada na Figura 5.3. Neste exemplo, parte-se da premissa que o cliente e o provedor de serviços não se conhecem, porém, há um relacionamento confiável entre os STSs/IdPs de cada domínio. O domínio do cliente emprega o modelo SPKI/SDSI, enquanto o provedor usa a ICP X.509.

No passo *1* (Figura 5.3), o cliente obtém a WSDL do serviço e interpreta a política de proteção anexada a WSDL. O documento WS-Policy anexado a WSDL exige credenciais no formato X.509 emitidas pelo STS/IdP do domínio do provedor de serviços.

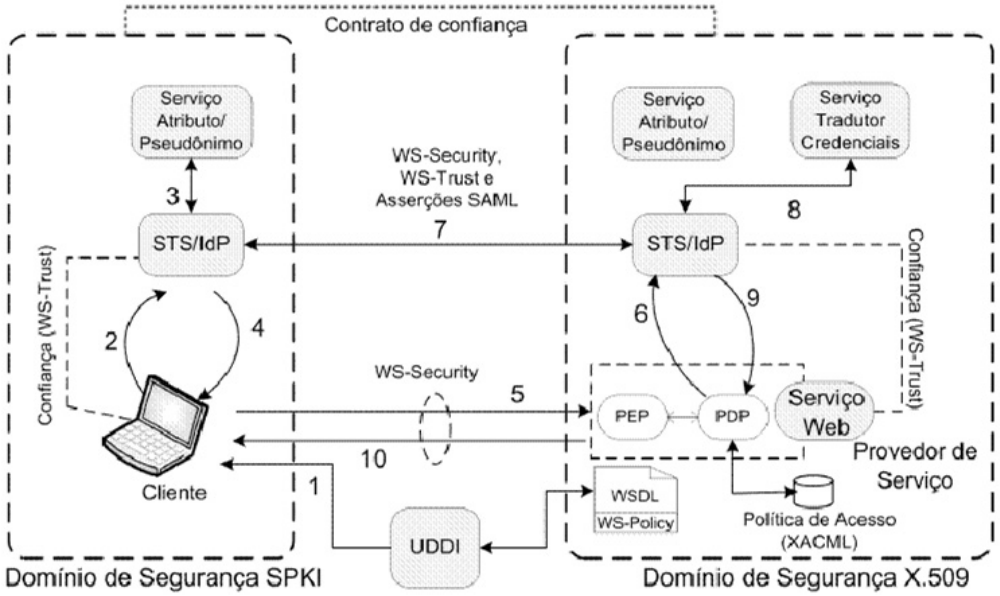

Figura 5.3: Modelo de transposição proposto [3]

Como o cliente não é capaz de interpretar X.509, e desconhece o STS/IdP descrito na política de proteção, o cliente solicita ao STS/IdP de seu domínio que faça a mediação da sua requisição. No passo *2* (Figura 5.3) o cliente efetua a sua autenticação para que o STS/IdP o reconheça como membro do domínio local. No passo *3*, o Serviço de Atributos e Pseudônimos (SAP) gera um pseudônimo para o cliente e o armazena temporariamente. O

pseudônimo é utilizado para atender a possíveis solicitações de atributos referentes ao cliente.

Após receber a asserção de autenticação no passo *4* (Figura 5.3), o cliente apresenta esta ao serviço, conforme o passo *5*, juntamente com a solicitação de acesso ao serviço. Como o provedor de serviços é uma entidade ativa (*i.e.*: o provedor de serviços é quem solicita os atributos do cliente), o cliente recebe de seu domínio de origem somente uma asserção de autenticação atestando a sua identidade. A asserção é enviada junto com a mensagem de solicitação de acesso ao recurso exposto pelo provedor de serviços.

Ao receber a asserção de autenticação SAML, o provedor de serviços deve verificar se a mesma contém os requisitos descritos em sua política de proteção, para fazer isso, o PEP repassa uma solicitação para o PDP requerendo uma avaliação de políticas do mesmo.

Durante o processo de autorização, o PDP realiza solicitações ao STS/IdP de seu domínio para que este traduza a asserção de autenticação SAML para o formato X.509. Se necessário, o PDP irá solicitar atributos (passo *6*) para auxiliar na tomada de decisão referente a solicitação de acesso ao serviço (passo *5*). O STS/IdP do provedor, tendo uma relação de confiança com o STS/IdP que emitiu a asserção, solicita os atributos ao domínio do cliente (passo *7*).

A autoridade de atributos no domínio do cliente recupera os atributos do SAP, e os repassa ao provedor de serviços. Apos receber a asserção de atributos SAML, o STS/IdP no domínio do provedor solicita ao Serviço Tradutor de Credenciais (STC) que um certificado X.509 com as informações do cliente seja construído (passo *8*). Este certificado serve para que o serviço conceda acesso ao cliente. Estando de posse do certificado X.509 e dos atributos, o provedor, através de seu PDP, pode tomar a decisão e efetivar o acesso do cliente ao serviço (passo *10*).

A arquitetura proposta preocupou-se com a transposição da autenticação do cliente, através de asserções de autenticação e de atributos SAML. O foco principal do trabalho foi a tradução de uma credencial no formato SAML para X.509, e a apresentação desta ao provedor de serviço que possui o recurso almejado pelo cliente. Não foi objetivo deste trabalho lidar com a transposição da autorização.

A abordagem utiliza o modelo *outsourcing* para controle e aplicação de políticas, ficando o PDP encarregado de entrar em contato com o serviço de credenciais (STS). O STS é o responsável por recuperar atributos que forneçam mais informações sobre o cliente que está solicitando acesso.

Esta proposta aumenta a carga de trabalho do PDP fazendo este entrar em contato com outra entidade, no caso o STS. Tal carga adicional de trabalho deveria ser encaminhada ao Manipulador de Contexto, pois de acordo com a especificação XACML, esta é a entidade responsável por recuperar outros atributos necessários em uma avaliação de acesso.

O STS é responsável por entrar em contato com os serviços presentes no domínio do cliente (STS/IdP e SAP). Somente após este procedimento, recuperam-se as asserções de atributos e uma nova credencial no formato de um certificado X.509 é gerada, sendo que somente esta será aceita pelo provedor que possui o serviço desejado pelo cliente.

A presente proposta, apesar de fazer uso de especificações amplamente aceitas e difundidas, não descaracteriza a dependência de entidades externas ao domínio do provedor. Quando determinado serviço externo (*i.e.*: serviço de atributos) falhar, o cliente fica impossibilitado de obter acesso ao recurso exposto pelo provedor, pois atributos adicionais solicitados em uma avaliação de acesso ou necessários para a emissão de uma nova credencial não podem ser recuperados do domínio do cliente.

5.4 Modelo de Controle de Acesso Sensível ao Contexto para Serviços Web

O trabalho proposto em [43] sugere extensões aos mecanismos de controle de acesso tradicionais, adicionando-lhes informações retiradas do contexto (*e.g.*: informações temporais, nível de ocupação do sistema). As informações e o próprio contexto são caracterizados de acordo com a implementação e o ambiente onde o mecanismo de controle de acesso será implantado. As informações e os atributos relativos ao ambiente (*e.g.*: de onde o pedido foi efetuado e onde o mesmo será atendido) têm a função de auxiliar na tomada de decisão referente a autorização.

O algoritmo proposto faz uso de atributos retirados dinamicamente do contexto onde o sistema está instanciado. Quando o cliente emite uma solicitação de acesso a um serviço, os atributos podem ser a hora local, localização geográfica baseada no endereço de IP do sistema cliente e o nível de ocupação do sistema onde o serviço se encontra instanciado. Estes atributos são usados para caracterizar um contexto particular. Os atributos do contexto são usados para determinar a confiança que se pode ter no cliente e em sua solicitação, influenciando diretamente na permissão concedida.

O trabalho propõe um esquema para estender o modelo X-RBAC [44] (Controle de Acesso Baseado em Papéis para XML) incorporando-lhe atributos. Assim, a decisão sobre o controle de acesso pode ser avaliada de acordo com os atributos apresentados, e também, de acordo com a política de restrições vigente em uma determinada organização.

Para o funcionamento do sistema de controle de acesso foi proposta uma extensão ao modelo X-RBAC para se obter a noção de sensibilidade do contexto, ou seja, propôs-se o modelo X-GTRBAC (*Generalized Temporal Role Based Access Control for XML*). O GTRBAC [45] é um modelo inicialmente proposto para controlar o acesso em grandes empresas, mas que poderá ser integrado ao controle de acesso de Serviços Web de acordo com as extensões propostas.

O modelo apóia-se nos seguintes principais componentes:

- ***Parameter Name Set (PN):*** representa um conjunto dos possíveis nomes de parâmetros do contexto; PN pode ser *time_of_day*, *location*, *duration*, *system_load*, etc.
- ***Parameter Type Set (PT):*** indica o conjunto dos possíveis tipos de parâmetros do contexto; PT pode ser *Time*, *String*, *Long*, *Integer*, etc.
- ***Parâmetros do Contexto:*** é representado por uma estrutura de dados 'p', com os campos *name* ε *PN*, *type* ε *PT*, e uma função *getValue()* que processa dinamicamente os valores dos parâmetros.
- ***Context:*** é um conjunto 'C' consistindo de 'n' parâmetros de contexto: (p1, … , pn).

Por exemplo, um pedido de acesso é definido pela tripla (*role*, *srv*, *context*), onde *role* é o cargo do sujeito, *srv* é o nome do serviço, e *context* são as informações de contexto. Estes dados são capturados dinamicamente durante a solicitação de acesso (conjuntos PN e PT).

Baseado no pedido, o sistema avalia as políticas aplicáveis à solicitação de acesso. As políticas são baseadas em conjuntos de restrições relativas ao cargo e ao serviço, formando as condições de acesso. As políticas são avaliadas em conjunto com as informações contextuais para aplicar um controle de acesso de fina granularidade.

O acesso aos serviços deve ser fornecido somente após o cliente estar devidamente autenticado. Por exemplo, a seguinte solicitação de acesso pode ser apresentada para avaliação:

```
< role = priv_cust,
      service_name = "review_claim",
      context = (p1 (time, 12PM), p2 (location, "WashDC"),
      p3 (duration, 0), p4 (system_load, "low") >
```

O pedido caracteriza o usuário para o cargo *priv_cust* (cliente privilegiado). O cliente está solicitando a análise de um contrato de seguros em um sistema *online*, através do serviço *review_claim* oferecido pela empresa. Os demais elementos fornecem, respectivamente, as informações de contexto (*context*): tempo, localização, duração do tempo inicial, carga do sistema. Como o pedido ainda não obteve sua autorização, o campo *duration* está com o contador zerado.

O contexto capturado no momento do pedido é fornecido como parâmetro para o algoritmo, sendo usado para auxiliar na tomada de decisão relacionada ao controle de acesso.

A seguinte política de acesso pode ser aplicada à solicitação:

```
CL1: (time > 9AM) AND (time < 5PM)
CL2: (location = "WashDC") OR (location = "NewYork")
CL3: (system_load! = "high")
CL4: (duration <= 600s)
```

Baseado nas informações acima, o sistema retorna uma decisão de autorização para a solicitação de acesso ao serviço. As informações contextuais disponíveis na solicitação indicam que as condições de acesso foram satisfeitas.

Devido as restrições especificadas para as solicitações de serviços, impondo limite à duração do período de acesso (elemento *duration*), e aplicadas pelo mecanismo de restrição temporal do GTRBAC, o tempo de duração do acesso do usuário é monitorado continuamente. Qualquer violação detectada por mecanismos do GTRBAC faz com que alguma providência imediata seja executada para que o tempo não exceda o limite imposto (*e.g.*: cancelar o acesso desvinculando o usuário de seu cargo).

Para se alcançar um bom nível de segurança na autenticação (*e.g.*: proporcionar acesso para clientes desconhecidos), o artigo sugere o uso de certificados emitidos por alguma CA. O certificado, juntamente com as informações contextuais do pedido, serviria para afirmar a identidade do cliente e que o mesmo está associado a um cargo (*role*), sendo assim

considerado confiável.

O artigo está principalmente voltado ao algoritmo usado para a combinação dos parâmetros usados nas tuplas, descrevendo a arquitetura na qual o X-GTRBAC se aplica. O X-GTRBAC pode ser acoplado a tecnologias de Serviços Web e prover um controle de acesso mais fino.

A abordagem proposta usa atributos relacionados a tempo, local, entre outros, que podem ser utilizados de maneira satisfatória para auxiliar na tomada de decisão referente ao controle de acesso. Porém, a proposta carece de um estudo mais profundo com relação a implementação do sistema em redes de larga escala, onde o comportamento do algoritmo e da arquitetura proposta poderiam ser realmente avaliados. A proposta deixa assim uma lacuna entre o modelo proposto e seu real comportamento em sistemas de controle de acesso distribuídos.

Neste trabalho não foram consideradas a utilização de nenhuma das arquiteturas de avaliação de políticas citadas anteriormente (*provisioning* ou *outsourcing*). Considera-se, porém, o uso de certificados como uma das formas de aferir a identidade do *principal* e a confiabilidade que se pode ter sobre o mesmo.

O uso de certificados nesta proposta se restringiu a uma credencial de acesso utilizada no processo de autenticação. O certificado serviu somente para designar um cargo ao *principal*, não sendo explorada todas as possibilidades de uso do mesmo.

A proposta sugere o uso de certificados emitidos por alguma CA, caracterizando assim a dependência de um mecanismo centralizado e pouco flexível por natureza.

5.5 Controle de Acesso Baseado em Autorização para Arquitetura Orientada a Serviço

De acordo com o trabalho proposto em [4], sistemas fundamentados na AOS que baseiam decisões de acesso na identidade do cliente têm sido considerados inflexíveis, com baixa escalabilidade e segurança, difíceis de gerenciar, usar e atualizar.

Este artigo mostra o controle de acesso baseado em identidade (IBAC – *Identity-Based Access Control*) como o *principal* contribuinte para as limitações acima citadas, e propõe basear decisões de controle de acesso em autorizações (ABAC – *Authorization-Based Access Control*) apresentadas pelo cliente.

Existem inúmeras especificações que proporcionam mais segurança aos Serviços Web (*e.g.*: XACML, SAML), sendo relevante para a proposta somente a SAML, cujo objetivo é prover meios para padronizar a troca de informações de segurança entre organizações.

O exemplo usado na especificação SAML descreve uma asserção emitida por um provedor de identidade contendo a seguinte declaração: *"o usuário John possui um e-mail john@company.com, foi autenticado no sistema usando mecanismo de senha"*. Esta declaração presente na especificação, segundo os autores, serve de influência para que as implementações baseadas em AOS, tomem suas decisões de controle de acesso baseadas em informações fornecidas pela identidade do usuário.

O provedor de serviço poderia optar pelo uso da informação fornecida pela identidade, de acordo com as políticas de autorização, para conceder acesso a serviços locais ao domínio. Tipicamente, o provedor de serviço usa a identidade do cliente para localizar as políticas associadas ao mesmo em algum repositório de políticas, e toma a decisão de acesso com as políticas recuperadas do repositório.

Aparentemente, a identidade do cliente não é uma informação crítica, as políticas de autorização parecem ser mais importantes. Se este for realmente o caso, a solicitação pode simplesmente conduzir informações de autorização ao invés da identidade do cliente.

No cenário da Figura 5.4, tem-se o usuário Alice invocando o serviço A6 (evento *1*) no provedor de serviço do Domínio 1 (D1). O serviço A6 acessa o serviço B5 (evento *2*, Domínio 2 – D2); B5 acessa A3 (evento *3*) e B2 (evento *4*); B2 acessa C3 (evento *5*) e C5 (evento *6*); e C5 acessa D5 (evento *7*, Domínio 3 – D3).

O usuário Alice invocou somente o serviço A6 (evento *1*), porém, o serviço A6 invocou outros serviços que são desconhecidos para Alice (*i.e.*: Alice não conhece, ou necessita conhecer a existência dos demais serviços). De fato, Alice talvez não tenha direito de invocação sobre os outros serviços. O problema se torna mais evidente, quando o usuário de um domínio (*e.g.*: Alice, Figura 5.4) necessita usar o serviço de outro domínio (*e.g.*: serviço B2, C3, etc.) tendo as decisões de acesso baseadas no modelo IBAC.

O domínio D2 confia no relacionamento que possui com o domínio D1, assinante da asserção SAML em nome de Alice, mas não em Alice que construiu a solicitação inicial desencadeando o pedido para o serviço B2 (domínio D2). O serviço D5 exige a autenticação do requisitante, mas o domínio D3 não tem um relacionamento confiável com o domínio D1. Neste caso, o domínio D2 deve anexar sua própria asserção SAML ao pedido recebido do

domínio D1 e encaminhar esta para o domínio D3. O Domínio 3 então cria uma entrada em seu repositório de política apoiado na asserção do domínio D2; pois, a asserção do domínio D1 não tem valor para o domínio D3.

Considerando o caso em que Alice muda de cargo dentro de D1, e Bob é designado para executar as tarefas de Alice. D1 não pode simplesmente revogar os certificados de identificação de Alice, Bob precisa deles para executar suas novas tarefas em nome de Alice. Ao invés disto, o repositório de políticas deve ser atualizado, mas não somente no domínio D1, o administrador de D1 deve informar os outros domínios relacionados da mudança, e estes devem atualizar seus repositórios adequadamente.

Raramente há regras que relacionam as responsabilidades impostas a cargos e tarefas quando se atravessam os limites de domínios administrativos de uma organização. Cada domínio possui interpretações e políticas particulares para cada cargo, bem como restrições especificas para a execução de cada tarefa. A exigência que se tem da consistência entre os repositórios de políticas limita a escalabilidade da abordagem IBAC.

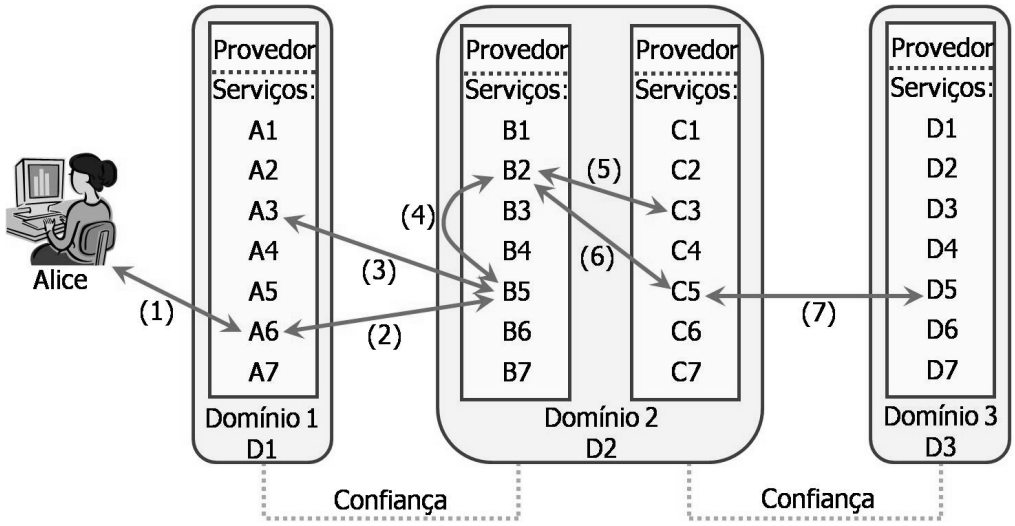

Figura 5.4: Cenário do uso de um serviço que cruza domínios

Outro possível problema com o controle de acesso baseado em identidade é que a identidade do usuário deve estar cadastrada em todos os domínios com o qual este estabelece algum tipo de relacionamento. Assim, informações sobre o usuário e organograma de uma organização acabam sendo expostas a outros domínios.

No modelo IBAC podem ocorrer momentos em que o usuário, Alice, necessita delegar

seus direitos a um subordinado, designando a este a realização de parte de suas tarefas. Para isto, Alice deve solicitar que o administrador de D1 faça uma atualização no repositório de políticas, para permitir que o subordinado, Bob, invoque o serviço A6. Porém, Bob somente será capaz de executar uma tarefa de Alice quando o repositório estiver atualizado, e Bob terá os direitos de Alice enquanto as entradas no repositório não forem removidas. Este exemplo mostra que a exigência de consistência entre repositórios de políticas, situados em diferentes domínios, limita a escalabilidade da abordagem IBAC.

Com o modelo IBAC (Figura 5.5), Alice cria um pedido para o serviço A6 (evento *1*) incluindo a prova de sua identidade (*e.g.*: assinatura). O serviço submete esta informação para o motor de política (evento *2*) que define as autorizações, retornando-as novamente para o serviço. O serviço baseia a decisão de acesso na autorização, e não na identidade de Alice.

Neste caso, fazendo-se uma pequena mudança de procedimento teremos Alice apresentando sua identidade para o motor de política e adquirindo as autorizações (ABAC, evento *1*, Figura 5.5), representando o que a mesma tem permissão de executar. Alice então constrói o pedido para o serviço A6 incluindo seus direitos e a ação que gostaria de executar (evento *2*). O serviço A6 necessita somente se certificar que a autorização para este serviço não foi falsificada (*e.g.*: assinatura do motor de política).

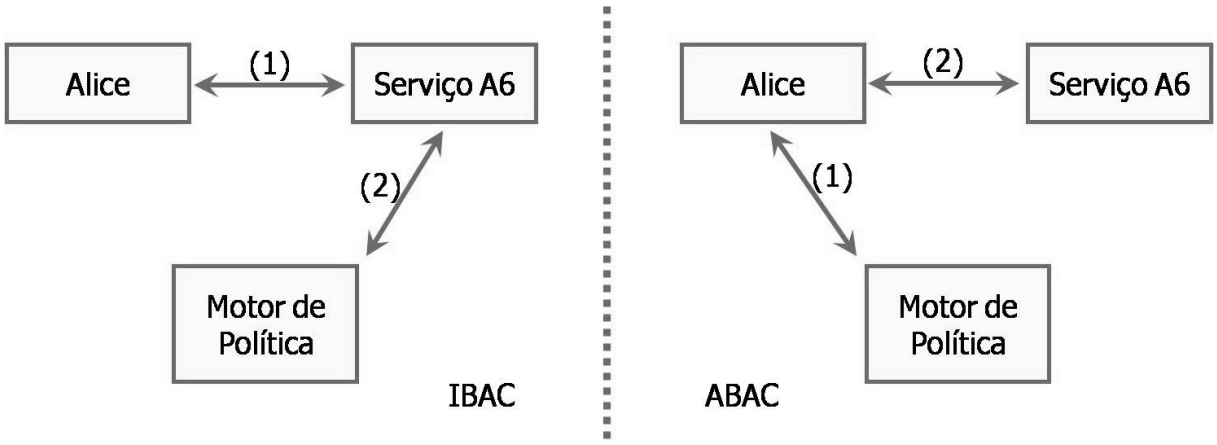

Figura 5.5: Comparação entre IBAC e ABAC

Na abordagem baseada em ABAC, ao invés de Alice prover sua identidade para o serviço e o mesmo adquirir a autorização do motor de política, Alice apresenta sua identidade para o motor de política, adquire suas autorizações e apresenta as apropriadas para o serviço

quando faz uma solicitação.

Enquanto o modelo IBAC tem uma lista de usuários associados com cada serviço, o ABAC tem uma lista de autorizações para cada usuário. Uma vez que os serviços estão registrados nos repositórios de políticas de cada domínio, o motor de política somente ativa a lista de autorização de acordo com a solicitação recebida.

Se Alice ainda não tem as autorizações para usar A6 e D5 (Figura 5.4), ela envia um pedido para seu motor de política. Este retorna a autorização apropriada para Alice que agora pode invocar o serviço A6, incluindo a autorização para a mesma usar D5. A autorização retornada pelo motor de política tem a forma de uma credencial de autorização, determinando que um pedido particular deva ser honrado, sendo a decisão de acesso independente da identidade do cliente.

Quando Alice mudar de cargo e Bob adquirir a responsabilidade pela execução das suas tarefas, o Domínio 1 (Figura 5.4) simplesmente muda as autorizações para a identidade do Bob. Se necessário D1 pode revogar e ou acrescentar as autorizações adequadas. Nenhum outro domínio precisa ser envolvido ou informado sobre as mudanças, não havendo transporte e ou vazamento de informação sobre a estrutura da organização de um domínio para outro.

No presente artigo conclui-se que a AOS é diferente dos sistemas que usam IBAC, pois essa cruza domínios administrativos, possuindo mais usuários e entidades distribuídas, sendo também mais dinâmica, mudando quando os objetos mudam (*e.g.*: cargos de funcionários). Se a AOS é projetada para alcançar estes objetivos, é decisivo reduzir ao máximo o acoplamento entre os domínios.

O modelo IBAC resulta em um acoplamento forte dos sistemas, além de exigir o gerenciamento de identidades distribuídas, podendo levar ao vazamento de informações internas aos domínios. Além disso, manter a consistência nas delegações de direitos pode exigir a atualização de repositórios de políticas multi-dominios.

A facilidade de delegação é uma das vantagens do modelo ABAC. Por exemplo, se Alice não estiver disponível para realizar uma tarefa quando necessário, esta pode enviar para Bob as autorizações necessárias, para que este realize a tarefa em seu lugar. Se Alice é confiável ao motor de política, esta somente entregaria sua autorização para alguém que a mesma julga ser confiável. Alice pode também revogar a autorização quando ela retornar ao cargo.

As atualizações no modelo ABAC ocorrem de maneira mais simples, visto que os

usuários se autenticam em seus domínios de origem. As mudanças no mecanismo de autenticação também são feitas localmente, no domínio de origem do cliente. O formato da credencial de autorização emitida pelo mecanismo de autenticação necessita ser compreendida somente pelo serviço destino e não pelo solicitante da credencial.

O artigo somente expõe e discute o modelo ABAC, não abordando questões como o período de validade nas declarações de autorização, estratégias para revogação de autorização (*e.g.*: onde e como esta informação deve ser divulgada), etc.

Em suma, o artigo somente realiza um comparativo entre as duas abordagens (IBAC e ABAC) enfatizando a flexibilidade do ABAC e ilustrando possíveis tecnologias e casos de uso. ABAC é uma abordagem bem próxima da oferecida pela infra-estrutura SPKI/SDSI.

5.6 Modelo de Controle de Acesso Baseado em Atributos para Serviços Web

Segundo [46] um dos grandes desafios para segurança em Serviços Web é o planejamento de estruturas de controle de acesso com fina granularidade. Estruturas que possam cruzar os domínios de segurança e ser implementadas em sistemas heterogêneos, possibilitando interações entre partes distintas que conhecem pouco uma da outra.

Porém, a maioria dos sistemas de controle de acesso baseia suas decisões de autorização na identidade do sujeito, que muitas vezes é ineficiente em ambientes de larga escala. Assim, um modelo de controle de acesso baseado em atributo (*i.e.*: WS-ABAC que estende o modelo RBAC) é apresentado para abordar este assunto.

Os atributos podem ser descritos como um conjunto de propriedades, podendo ser associadas a determinada entidade, seja esta um usuário (exemplo de atributos: *e-mail*, cargo, idade) ou um recurso (exemplo de atributos: localização, tamanhos de arquivo, parâmetros de acesso).

O WS-ABAC concede acesso para Serviços Web baseando-se em uma credencial digital devidamente assinada, e munida de atributos fornecidos por entidades afins (*i.e.*: alguém com quem se tem relações confiáveis pré-estabelecidas) expressando assim a veracidade dos atributos portados pela credencial.

A política de controle de acesso do WS-ABAC especifica quais atributos devem formar a base para se obter acesso a determinado serviço. Um parâmetro passado para o

serviço é um atributo, por exemplo. O WS-ABAC torna possível dessa forma especificar que um usuário é autorizado a usar um determinado serviço, mas somente para algum valor específico de parâmetros passados para o serviço, provendo assim um controle de acesso de fina granularidade.

Uma vez aceita a solicitação de acesso, o sistema inicia a avaliação desta pelo uso das políticas de controle de acesso correspondentes, e decide se aceita ou não a solicitação. Quando os atributos apresentados não satisfazem as condições de acesso, a solicitação não é simplesmente negada, mas usa-se o mecanismo de ATN (*Automated Trust Negotiation*) para se adquirir os atributos necessários. Somente quando nenhum atributo é encontrado a solicitação de acesso é rejeitada.

No esquema de controle de acesso proposto, as regras são expressas como conjuntos de atributos. A política utilizada no controle de acesso especifica quais atributos e parâmetros de serviço são permitidos para se acessar o serviço solicitado. Um exemplo de restrição poderia ser a definição da seguinte regra: *O serviço pode ser acessado pelo administrador, de dentro do escritório, entre as 10:00 da manhã e as 03:00 da tarde quando a carga do sistema estiver baixa*. A regra seria expressa da seguinte maneira:

```
C = Time >= 10:00 ∩ Time <= 15:00 ∩ System_load = "low" ∩ Location =
"office" ∩ Identity = "manager".
```

Este modelo foi baseado em uma linguagem de tuplas que pode ser mapeada para estruturas XML. Um exemplo de uma política de acesso pode ser definido da seguinte forma: *pol* = < *S, srv, C* >, sendo *pol* a política, *S* o sujeito da política, *srv* o identificador do serviço solicitado e *C* a restrição de atributo necessária para acessar o serviço *srv* de acordo com a política *pol*.

A linguagem XACML fornece meios de padronizar o formato das políticas, bem como o contexto de mensagens utilizadas para solicitar decisões de controle de acesso. Usa-se a XACML como a linguagem para se padronizar as políticas de controle de acesso, estendendo-se a mesma para especificar as restrições das políticas.

No caso de uso da Figura 5.6, o usuário remoto está associado a um domínio de segurança o qual lhe permite acessar Serviços Web divulgados em outros domínios. A aplicação que executa o acesso é referenciada como solicitante. O solicitante, que geralmente

é um portal *web*, autentica o usuário antes do acesso aos recursos ser permitido. O portal *web* também é encarregado de construir os pedidos no formato de mensagens SOAP.

Na arquitetura mostrada na Figura 5.6 o usuário pode selecionar uma determinada opção, mostrada na página de seu navegador *web* (evento *1*). Esta opção dispara a execução de um método em um WS localizado em outro domínio de segurança (evento *7*) em nome do usuário. O solicitante invoca a opção selecionada enviando uma mensagem SOAP para o WS.

Os principais componentes da arquitetura mostrada na Figura 5.6 são:

- ***SOAP Handler:*** encarregado de aceitar mensagens SOAP (evento *2*) e verificar a assinatura digital das informações submetidas no pedido. Se este controle obtiver sucesso, uma nova mensagem transportando o pedido atual é extraída do corpo da mensagem SOAP e enviada para o PEP (evento *3*).
- ***Policy Enforcement Point (PEP):*** após receber a mensagem, este envia um pedido de autorização para o Manipulador de Contexto (evento *4*) que repassará a mensagem para o PDP (evento *5*). O PEP honra a decisão do PDP, a qual será retornada pelo Manipulador de Contexto (*e.g.*: permitir ou negar acesso, evento *7*).
- ***Context Handler:*** implementa duas funções: 1) administra o repositório de atributos e suas implementações. Quando o PDP encontrar um atributo desconhecido, este consulta o Manipulador de Contexto para obter informações sobre o tipo do atributo (evento *5.a*); 2) converte o pedido de decisão (evento *4*) do formato nativo do pedido para o formato XACML (evento *5*), e depois converte a decisão de autorização (evento *5*) da forma XACML para o formato nativo do pedido (evento *4*).
- ***Policy Decision Point (PDP):*** após receber a mensagem (evento *5*) avalia o pedido de acesso baseado nos atributos obtidos da mensagem e nas regras de políticas XACML.
- ***Policy Management Tool:*** ferramenta que permite a definição de políticas e a manutenção das mesmas disponibilizadas ao PDP (evento *6*).
- ***Attribute Authorities:*** define e fornece atributos do sujeito, necessários para uma decisão de autorização (evento *4.a*).
- ***Environment Authorities:*** define e fornece atributos do ambiente computacional, necessários para decisões de autorização (evento *4.b*).
- ***Resource Authorities:*** define e fornece parâmetros de serviço, necessários para acessar um determinado serviço (evento *4.c*).

A proposta foca-se na utilização do modelo WS-ABAC e na descrição da linguagem utilizada para prover um acesso de fina granularidade sobre os recursos.

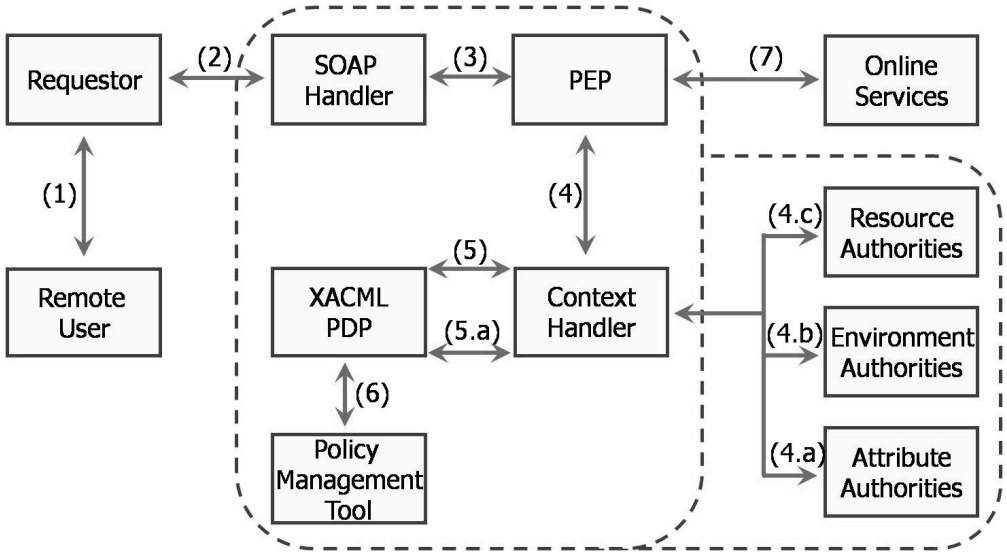

Figura 5.6: Arquitetura do modelo WS-ABAC

A implementação do modelo para Serviços Web focou-se somente na descrição das entidades, bem como a função de cada uma na arquitetura. O modelo de autorização proposto usa a abordagem baseada no paradigma arquitetural *outsourcing*, permitindo que decisões de autorização sejam tomadas e aplicadas baseadas em atributos, parâmetros e políticas. Sendo assim, não descaracteriza o forte acoplamento mencionado nos trabalhos anteriores.

5.7 Conclusão

Neste capítulo mostraram-se algumas abordagens relacionadas a autenticação e autorização, enfatizando que somente o uso de mecanismos tradicionais (*e.g.*: usuário e senha) não são mais suficientes.

Para se prover um bom nível de segurança na autenticação e autorização de *principals* outros métodos foram adotados, acrescentando mais informações (*i.e.*: atributos) no processo de tomada de decisão referente ao controle de acesso.

Algumas das pesquisas apresentadas focaram seus objetivos em propor e assegurar

relacionamentos confiáveis inter-domínios. Sugerindo modelos que visam facilitar o acesso de *principals* a recursos instanciados em diferentes domínios de segurança. Os trabalhos de pesquisa apresentados, em sua maioria, dependem de relacionamentos confiáveis inter-domínios, ou seja, dependem de entidades externas ao domínio do provedor de serviços.

Os trabalhos apresentados neste capítulo fazem uso de especificações amplamente aceitas e difundidas, visando facilitar o estabelecimento de confiança e a interoperabilidade entre os sistemas instanciados em ambientes heterogêneos. O objetivo proposto é alcançado pelos trabalhos, porém, o custo para se alcançar tal objetivo é um grande número de serviços instanciados em cada domínio e uma grande quantidade de mensagens trocadas na rede.

Pode-se perceber em algumas pesquisas, que os modelos implementados ainda estão presos a métodos e mecanismos convencionais, dificultando a evolução dos sistemas de controle de acesso. Em sua maioria, os mecanismos de controle de acesso, autenticação e autorização inter-domínios ainda dependem de entidades centralizadas. Característica esta que colabora para o aumento da vulnerabilidade (*e.g.*: ponto único de falhas) dos sistemas distribuídos.

As pesquisas apresentadas neste capítulo serviram como base para motivar o desenvolvimento da proposta apresentada no capítulo seguinte. Na presente dissertação iremos propor um modelo de controle de acesso e gerencia de políticas descentralizado, tendo como um dos objetivos diminuir o acoplamento (*i.e.*: dependência) de entidades externas ao domínio do provedor de recursos.

Capítulo 6

Gerenciamento de Políticas de Controle de Acesso Fracamente Acoplado para Serviços Web

6.1 Introdução

A Arquitetura Orientada a Serviço (AOS) é uma abordagem genérica e baseada em padrões que visa o baixo acoplamento (independência) entre as entidades que a compõem. Os Serviços Web (*Web Services* – WS) são uma das implementações concretas da AOS.

Serviços Web dispõem de várias especificações, abordando inúmeros aspectos inerentes ao desenvolvimento de aplicações distribuídas seguras, flexíveis, escaláveis, e fracamente acopladas.

Visando atender aos aspectos supracitados, a abordagem de gerenciamento de políticas baseada no modelo de provisionamento pode ser utilizada na arquitetura dos Serviços Web. O modelo de provisionamento ameniza algumas implicações referentes ao gerenciamento das políticas de controle de acesso aplicadas no ambiente distribuído de uma corporação.

Em grandes corporações as quais possuem várias filiais e vários *principals* atuando em diferentes cargos, o ponto de avaliação de políticas (*i.e.*: monitor de referência) e o ponto de aplicação de políticas (*i.e.*: guardião) instanciados nas filiais poderiam ser pré-configurados.

Quando o *principal* de uma das filiais da corporação necessitar fazer uso dos serviços fornecidos em outra divisão da corporação, esse já possuiria seu acesso pré-configurado, ou este seria configurado de maneira dinâmica.

No caso em que o *principal* não possua seu acesso configurado, a solicitação de acesso não seria simplesmente negada, mas sim, o guardião do recurso recorreria ao monitor de referência que lhe proveu as configurações iniciais (*i.e.*: administração da corporação), para

que este tome a decisão de política e retorne ao guardião.

A resposta retornada do monitor de referência, presente na administração da corporação, para o guardião pode ser na forma de uma atualização de política, uma decisão de autorização, ou ambos. Sendo que esta deverá ser aplicada à solicitação efetuada pelo *principal* (*e.g.*: funcionário), permitindo ou negando o acesso do mesmo ao recurso.

No caso de uma solicitação de acesso que seria negada na abordagem tradicional, com a integração do SPKI/SDSI e da WS-Policy, o sujeito será informado como proceder para obter o direito necessário.

As cadeias de certificados de autorização, fornecendo direitos para *principals*, podem ser emitidas com tempos de validade bem definidos. Desta maneira, concederiam ao *principal* um tempo limitado para o uso de seus privilégios de acesso em algum dos recursos presentes nos domínios das filiais.

As informações fornecidas pela cadeia de certificados de autorização SPKI/SDSI permitem que novas políticas de autorização XACML sejam dinamicamente geradas no ambiente de cada provedor de serviço (*i.e.*: filial da corporação). As políticas possuem uma granularidade fina, ou seja, são específicas para cada *principal* e respectivo recurso.

As novas políticas XACML são envidas para o repositório de políticas corporativo, sendo utilizadas para manter a visão administrativa unificada do ambiente distribuído. As políticas recebidas pela administração da corporação são enviadas (*i.e.*: provisionadas) para os respectivos repositórios, nas filiais da corporação.

O SPKI/SDSI, como utilizado na proposta, cria um diferencial bastante significativo em relação à abordagem tradicional aplicada aos Serviços Web.

Na presente proposta, as políticas XACML de fina granularidade são geradas sem a intervenção de um administrador humano, de forma segura e consistente. Ou seja, sem violar as políticas corporativas.

6.2 Modelo para AOS

A arquitetura proposta (Figura 6.1) considera um sistema composto de um repositório centralizado onde está armazenado o conjunto de políticas do ambiente (*i.e.*: corporação), e vários repositórios distribuídos (*i.e.*: filiais da corporação) contendo subconjuntos específicos destas políticas. Podemos visualizar a arquitetura proposta como sendo um ambiente

corporativo com uma matriz e várias filiais, por exemplo, para facilitar o entendimento.

O Repositório de Políticas Corporativo (RPC) armazena todas as políticas da corporação, para todos os recursos disponibilizados pela mesma. O RPC é o único repositório que está sujeito a atualizações administrativas (evento up_1). O RPF (Repositório de Política da Filial) armazena uma cópia local (evento *pp*) das políticas especificamente aplicadas aos recursos que o Guardião (GRD) controla.

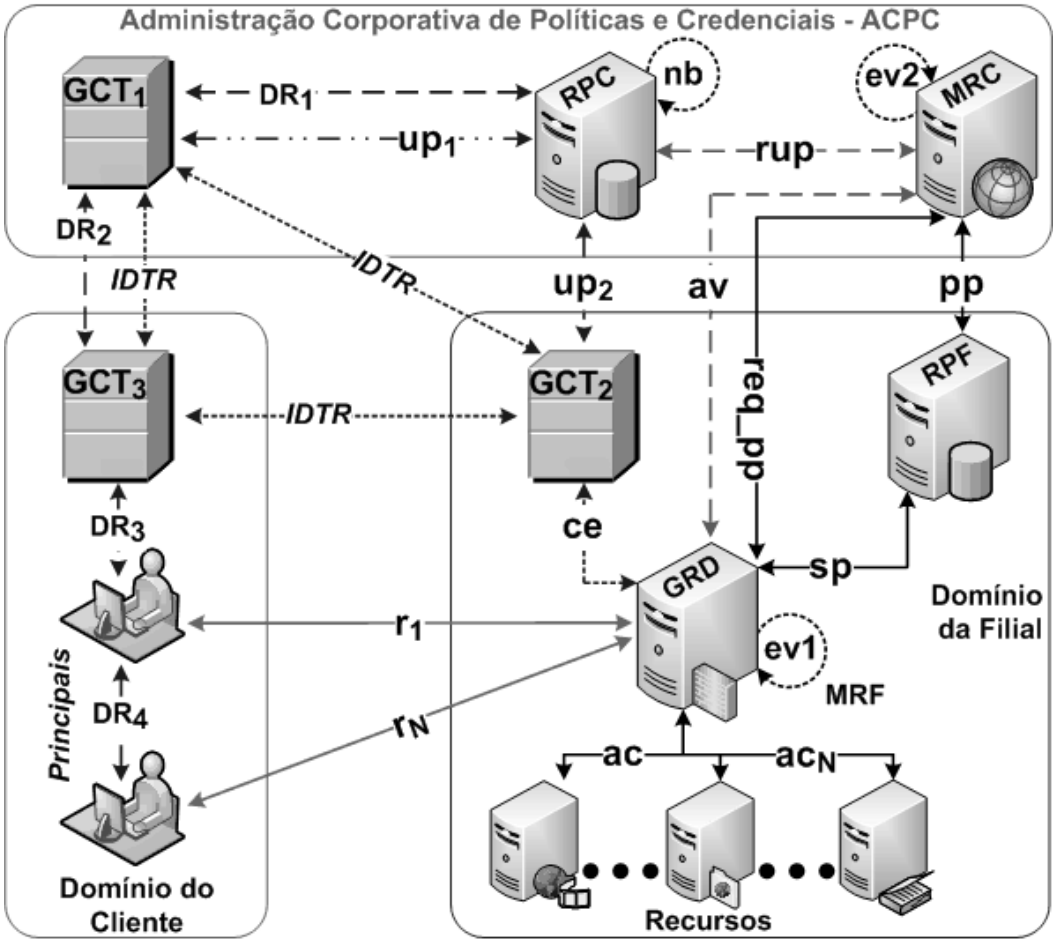

Figura 6.1: Interação entre as entidades do modelo para AOS

A arquitetura de controle de acesso proposta suporta ambos os modelos de avaliação, *outsourcing* e *provisioning*. O Guardião (GRD) impõe a decisão de autorização (evento *ac*) independentemente se avaliação de políticas ocorreu na filial (MRF – Monitor de Referência da Filial, evento ev_1) ou no nível corporativo (MRC – Monitor de Referência Corporativo, eventos *av* e ev_2).

Com o uso de uma infra-estrutura de chave pública flexível e independente da

tecnologia subjacente, pode-se delegar direitos através de certificados (eventos *DR*, Figura 6.1), formando a cadeia de certificados de autorização. Uma das vantagens do uso de tal infra-estrutura é que através da cadeia fica mantida em sistema a informação de quem delegou direitos para quem. Além disto, o administrador corporativo especifica quais direitos podem ser delegados, evitando assim que a concessão de direitos no domínio da filial transgrida as restrições da política corporativa.

Cadeias de certificados de autorização concedem direitos à *principals* sem a necessidade de atividades administrativas efetuadas por humanos. Um *principal* devidamente autorizado, e que tenha direitos em sua cadeia, é capaz repassar estes direitos a outro *principal* (evento DR_4).

A delegação de direitos pode ser total ou parcial, ou seja, pode-se repassar todos os direitos concedidos pela cadeia de certificados ou somente ou subconjunto dos mesmos. O *principal* que possui algum privilégio de acesso em sua cadeia de certificados pode repassar estes a qualquer *principal* que o mesmo julgar ser confiável, desde que a cadeia de certificados permita a delegação.

Quando o *principal* apresenta uma cadeia de certificados de autorização junto com uma solicitação de acesso (mensagem r_1, Figura 6.1) a um Guardião (GRD), este a encaminha a um Gerenciador de Chaves e *Tokens* (GCT_2) para que a mesma seja reduzida (*i.e.*: convertida) em um certificado único.

O procedimento de redução da cadeia tem a finalidade de verificar e validar todos os elementos dos certificados que a compõe. Ou seja, verificar se autorizações indevidas (*i.e.*: que não estão em conformidade com as políticas corporativas) foram inseridas durante os processos de concessão de direitos.

A partir do certificado reduzido é gerada uma credencial de autorização específica (evento *ce*, Figura 6.1) para o mecanismo que implementa o Guardião (GRD). Adicionalmente, as informações extraídas do certificado reduzido são utilizadas para atualizar, automaticamente, o Repositório de Políticas Corporativo (RPC). Este procedimento se assemelha com atividades administrativas humanas (evento up_2), gerando políticas específicas para cada *principal* e respectivo recurso.

O Repositório de Políticas da Filial (RPF) é atualizado sempre que necessário (eventos *rup* e *pp*, respectivamente) para manter sua consistência com o Repositório de Políticas Corporativo (RPC).

Se por algum motivo o RPF 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 6.1). Este esquema auxilia na descentralização da administração de políticas e, ao mesmo tempo, mantém as informações de autorização (*i.e.*: políticas) unificadamente armazenadas no repositório da ACPC.

Fazendo uso de certificados, é possível criar relações de confiança mútuas entre os GCTs. As relações de confiança interdomínios (*IDTR*, Figura 6.1) se baseiam em certificados mutuamente emitidos entre os GCTs. Estas relações de confiança (*IDTR*) têm propósitos administrativos, principalmente para permitir a concessão de direitos entre os administradores dos diferentes domínios (*e.g.*: 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. Esta é a única maneira de um *principal* conhecer seguramente a chave pública do outro – dado que não existe uma autoridade certificadora centralizada na infra-estrutura de chave pública utilizada.

As relações de confiança são consideradas necessidades administrativas, pois, raramente o administrador do domínio (GCT) faz uso dos direitos que lhe são repassados. Normalmente, estes direitos são parcialmente repassados (*i.e.*: concedidos) aos *principals* dentro do domínio do cliente (*e.g.*: GCT_3 para *principal*, evento DR_3).

A transposição dos domínios de autorização é facilmente obtida usando cadeias de certificados de autorização. Os certificados carregam todos os atributos necessários para a avaliação de uma solicitação de acesso. Sendo assim, o provedor de serviço não necessita contatar nenhuma outra entidade (*e.g.*: serviço de atributos) para obter atributos adicionais do *principal*.

O provedor de serviço (*i.e.*: filial da corporação) pode obter a decisão de autorização baseado unicamente nas informações extraídas da 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 no ambiente distribuído.

6.3 Modelo para Serviços Web

A Figura 6.2 mostra detalhes da arquitetura proposta considerando as entidades que

implementam a segurança no ambiente de Serviços Web, assim como algumas interações entre as mesmas na composição do cenário corporativo. Em nosso cenário são usados Serviços Web, SPKI/SDSI e o modelo *provisioning* para implementar os controles de segurança.

No *bootstrap* do PEP é feita pré-configuração das políticas no repositório da filial empregando-se a SPML (Seção 3.3.5), e o modelo *provisioning* (Seção 4.4.2). O PEP envia uma solicitação para o PDP (evento *req_pp*, Figura 6.2) solicitando as respectivas políticas. O PDP recupera as políticas do repositório (evento *rup*) e as envia ao LPAP no domínio da filial (*Local PAP*, evento *pp*). A partir de então, as políticas estão disponíveis para serem avaliadas pelo LPDP (*Local PDP*, evento ev_1).

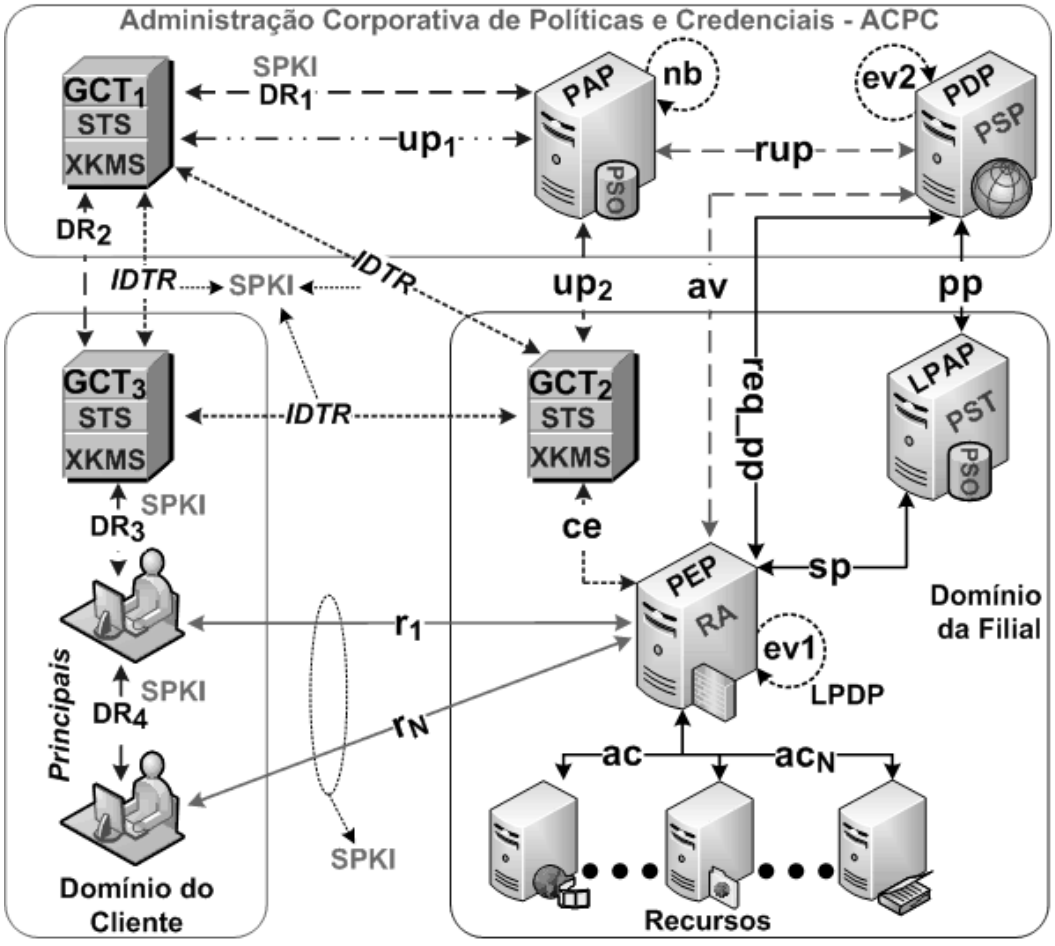

Figura 6.2: Interação entre as entidades do modelo com Serviços Web

De acordo com a arquitetura da especificação SPML, o PDP representa o PSP

(*Provisioning Service Provider*, Figura 6.2). 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 XACML provisionadas, ou PSOs. Um PDP (PSP) pode administrar diferentes PEPs (RAs, *e.g.*: cada PEP sendo responsável por uma filial da corporação). A SPML não é restritiva quanto à localização dos PSOs ou PSTs.

Os relacionamentos de confiança entre os GCTs (associações *IDTR*, Figura 6.2) são baseados em certificados de nome SPKI/SDSI, fazendo-se a inclusão de um GCT (STS) no grupo do outro. Ou seja, o GCT_X insere o GCT_Y em seu grupo local e emite um certificado de nome, denotando que este 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, esse pode obter os direitos exigidos através de uma cadeia de certificados de autorização. No evento DR_3 (Figura 6.2), 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 acessar o recurso caso já possua a cadeia. Caso contrário obtém o direito solicitado pelo *principal* através do relacionamento *IDTR* (evento DR_2) do respectivo GCT_1.

Se o *principal* em DR_3 (Figura 6.2) for o administrador de um departamento, por exemplo, o certificado de autorização pode ser delegável (*i.e.*: os direitos contidos no certificado podem ser repassados aos funcionários do departamento – evento DR_4). Em caso contrário, o certificado de autorização conterá apenas os direitos para que o *principal* acesse o recurso no provedor. Ou seja, os direitos não podem ser repassados pelo *principal* receptor. O parâmetro que define se os privilégios de acesso transportados pelo certificado de autorização podem ser delegados, ou não, é a *tag* "*propagate*" (Figura 4.1).

6.4 Dinâmica do Protótipo em Ambiente de Serviços Web

Considerando que em determinado momento um *principal* envia uma solicitação de acesso a um recurso (evento r_1, Figura 6.3). O Manipulador de Contexto, junto ao PEP, encaminhará esta solicitação ao LPDP (evento ev_1), 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 corporativo (evento *av*, Figura 6.3), pois o *principal* poderia estar em trânsito (*i.e.*: não pertence ao domínio desta filial). Neste caso, a avaliação é feita no modelo *outsourcing*: o PDP na ACPC avalia a política (evento ev_2) e o PEP no domínio do provedor (*i.e.*: filial) executa o resultado retornado pela avaliação.

Todas as trocas de mensagem entre PEP, PDP e PAP (evento *rup*) são feitas de acordo com o *SAML 2.0 Profile of XACML v2.0*. Esta abordagem foi adotada considerando que as entidades são WS.

Supondo que o PDP, após consultar o PAP (eventos *rup* e *sgp*, Figura 6.3) não encontre as políticas para o respectivo *principal* e recurso almejado. Então o PEP oferece uma alternativa ao *principal*, usando o protocolo *challenge-response* [23] é enviada ao *principal* uma mensagem informando quais direitos são necessários para acessar o recurso. A mensagem retornada transporta um documento WS-Policy.

O Apêndice B.1.1. mostra o exemplo de uma aplicação prática do protocolo *challenge-response* em contextos que utilizam a infra-estrutura de chave pública SPKI/SDSI.

O *principal* solicita ao GCT (STS) uma cadeia de certificados SPKI/SDSI que conceda os direitos requeridos de acordo com o documento WS-Policy. Obtendo a cadeia (evento DR_3) o *principal* responde o desafio feito pelo PEP, enviando à solicitação de acesso e a cadeia de certificados (evento r_1). O PEP encaminha a cadeia de certificados ao GCT (evento *ce*) para que seja feita a conversão da mesma em um certificado reduzido.

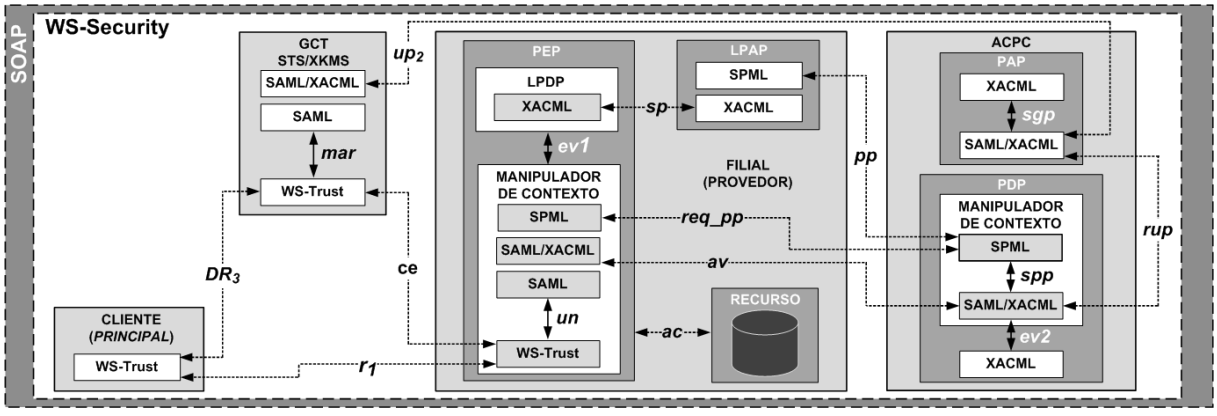

Figura 6.3: Protocolos e especificações utilizadas na arquitetura do protótipo

O GCT (STS) encaminha a cadeia ao XKMS, que reduz a cadeia de certificados e

devolve um único certificado ao GCT. Então, o GCT gera uma asserção 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 6.3). O GCT devolve a asserção SAML (evento *ce*) e o PEP libera o acesso (evento *ac*) conforme as expressões (autorização e atributo) contidas na asserção desserializada (evento *un*).

Além de fornecer a asserção SAML, o GCT gera uma política XACML a partir da cadeia de certificados reduzida. A nova política é armazenada temporariamente em um repositório local. A cada período de tempo pré-definido, um processo executando concorrentemente encapsula as políticas XACML em SAML de acordo com o *SAML 2.0 profile of XACML v2.0* [42] e as envia para o PAP (evento up_2, Figura 6.3). Esta atualização automática do PAP deve ser entendida como uma ação administrativa de geração de política.

Se o PAP retornar uma mensagem afirmando que a política foi persistida em seu repositório, a mesma é removida do repositório temporário do GCT. Este processo tem a finalidade de não interferir no tempo de resposta para o *principal* obter acesso ao recurso. Após receber a política, o PAP envia uma mensagem de atualização de política (eventos *rup*) endereçada ao LPAP, usando a arquitetura da SPML (evento *spp* e *pp*).

6.5 Conclusão

Neste capítulo foi apresentada a proposta e um cenário de estudo, mostrando a interação entre as entidades do modelo de acordo com as especificações disponibilizadas para Serviços Web. A presente proposta provê a descentralização da arquitetura de controle de acesso, enquanto mantém o controle unificado na gestão dos direitos de acesso.

A descentralização na avaliação de políticas é alcançada com o modelo *provisioning*. O gerenciamento corporativo unificado é obtido usando a delegação de direitos através de certificados de autorização SPKI/SDSI.

Novas políticas são geradas localmente nas filiais da corporação 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, colaborando assim para manter o controle unificado das políticas da corporação.

A presente proposta, aplicada ao controle de políticas, forma uma arquitetura de gerenciamento de políticas com baixo acoplamento entre as entidades.

No próximo capítulo serão apresentadas as avaliações que foram feitas no protótipo que implementa arquitetura proposta, mostrando a viabilidade de uso da mesma para ambientes que necessitam de uma gerência de políticas descentralizada.

Capítulo 7

Implementação e Avaliação do Protótipo

7.1 Introdução

Neste capítulo serão apresentados os principais detalhes de implementação e as avaliações feitas no protótipo que implementa a arquitetura proposta. O capítulo mostra um breve comparativo entre os modelos *provisioning* e *outsourcing*, expondo-se as diferenças entre cada um deles para executar a avaliação das políticas de controle de acesso.

O foco principal deste capítulo está na avaliação de acesso feita usando-se a cadeia de certificados de autorização SPKI/SDSI. A cadeia gera políticas dinâmicas e com fina granularidade, e a credencial de acesso usada para autorizar o *principal* na utilização do recurso.

O protótipo foi implementado fazendo uso das contribuições apresentadas nos modelos e infra-estruturas descritas nos capítulos anteriores.

7.2 Tecnologias

As ferramentas e bibliotecas listadas abaixo foram utilizadas no desenvolvimento do protótipo:

- O desenvolvimento dos programas está baseado na linguagem de programação Java (*Java Development Kit* – JDK, http://java.sun.com/);
- Como ambiente de desenvolvimento, foi utilizada a IDE (*Integrated Development Environment*) Eclipse (http://www.eclipse.org/);
- O Apache TomCat (http://tomcat.apache.org/) foi adotado como servidor de aplicação;

- O Apache Axis (http://ws.apache.org/axis2/) foi utilizado para criar as mensagens SOAP e instanciar os interceptadores que aplicam segurança à mensagem SOAP;
- O módulo Apache Rampart (http://ws.apache.org/rampart/index.html) integrado ao Axis, foi utilizado para implementar as especificações WS-Security, WS-Trust;
- O Apache Neethi (http://ws.apache.org/commons/neethi/) foi utilizado para implementar a WS-Policy.
- O suporte a SAML foi obtido a partir do projeto OpenSAML (http://www.opensaml.org);
- O suporte a SPML foi alcançado através do projeto OpenSPML (http://www.openspml.org/);
- A infra-estrutura SPKI/SDSI foi acessada a partir da biblioteca JSDSI (http://jsdsi.sourceforge.net/), inicialmente desenvolvida por Morcos [47];
- O suporte a XACML foi obtido através do projeto SUN XACML (http://sunxacml.sourceforge.net/);
- E o banco de dados com suporte a XML utilizado no protótipo foi o Oracle Berkeley *Data Base* XML (http://www.oracle.com/).

7.3 Implementação

A seguir serão comentados alguns pseudocódigos e suas funcionalidades implementadas no protótipo.

Considerando inicialmente que o *principal* enviou uma solicitação de acesso ao PEP. Após receber a requisição do *principal* (linha *1*, Figura 7.1), o PEP avalia se a requisição transporta uma cadeia de certificados de autorização SPKI/SDSI, e em caso positivo, encaminha a mesma ao GCT_2 (linhas *5* e *6*, respectivamente). Em caso contrário, o PEP solicita a avaliação de políticas no modelo *provisioning* (invocando o LPDP, linha *7*) ou *outsourcing* (invocando o PDP, linha *10*). Independentemente de onde a decisão de autorização foi tomada (LPDP ou PDP), o PEP executa a asserção SAML (linha *14*) ou o resultado da avaliação das políticas XACML (linha *16*).

Caso o *principal* não se enquadre em nenhuma das alternativas anteriores, um documento no formato WS-Policy é retornado ao *principal*, informando-o dos requisitos

necessários para se obter acesso ao recurso (linha *18*).

```
1.      Pep_Module(add_request)
2.      BEGIN
3.              assertion = null;
4.              evaluate = null;
5.              IF(add_request instance_Of spki_cert)
6.                      assertion = Send_To_StsP(add_request);
7.              ELSE IF(add_request instance_Of user_pswd)
8.                      evaluate = Call_Lpdp(add_request);
9.                      IF(evaluate == null)
10.                             evaluate = Call_Pdp(add_request);
11.                     END IF
12.             END IF
13.             IF(assertion != null)
14.                     response = Call_Service(assertion);
15.             ELSE IF(evaluate != null)
16.                     response = Call_Service(evaluate);
17.             ELSE
18.                     return WS_Policy;
19.             END IF
20.             return response;
21.     END
```

Figura 7.1: Pseudocódigo - PEP recebe a solicitação

O GCT_2 ao receber a requisição do PEP (Figura 7.2), extrai a cadeia de certificados e a encaminha para validação. O algoritmo irá validar todos os parâmetros da cadeia, extraindo os campos necessários para a geração da política XACML e asserção SAML (linhas *3* a *10*).

```
1.      Validate_Sequence(spki_Sequence)
2.      BEGIN
3.              valid = Verify_Signatures(spki_Sequence);
4.              propagate = Verify_Propagates(spki_Sequence);
5.              date_Time = Extract_Dates(spki_Sequence);
6.              date = Intersect_Dates(date_Time);
7.              tags = Extract_Tags(spki_Sequence);
8.              right = Intersect_Tags(tags);
9.              principal = Get_Last_Subject(spki_Sequence);
10.             comment = Extract_Comments(spki_Sequence);
11.             IF(all_Extract_Data and Signatures is valid)
12.                     To_Policy (right, principal, comment, date);
13.                     saml = To_Saml(date, right, principal);
14.                     return saml;
15.             ELSE
16.                     return invalid_Sequence;
17.             END IF
18.     END
```

Figura 7.2: Pseudocódigo - Validar a cadeia de certificados SPKI/SDSI no GCT_2

Se a cadeia for válida é realizada uma chamada aos métodos que irão gerar a política XACML (linha *12*, Figura 7.2) e a asserção SAML (linha *13*). A asserção SAML gerada é

retornada ao PEP (linha *14*) para que este libere o acesso de acordo com as expressões transportadas pela asserção.

A partir de uma cadeia de certificados de autorização válida é gerada uma regra de política XACML "*Permit*" (linha *8*, Figura 7.3) que concede o acesso ao *principal*. Após a geração da política, a mesma é armazenada temporariamente em um repositório local ao GCT_2 (linha *12*).

```
1.      To_Policy(right, principal, comment, date)
2.      BEGIN
3.              xacml_PR.set_Id(Some_Id_ + right.get_Action);
4.              xacml_PR.set_Description(comment);
5.              xacml_PR.set_Subject(principal);
6.              xacml_PR.set_Resource(right.get_Resource);
7.              xacml_PR.set_Id(Some_Id_ + right.get_Action);
8.              xacml_PR.set_Effect(Permit);
9.              xacml_PR.set_Action(right.get_Action);
10.             xacml_PR.set_Condition(date.get_Not_Before)
11.             xacml_PR.set_Condition(date.get_Not_After)
12.             Store_Xacml (xacml_PR)
13.     END
```

Figura 7.3: Pseudocódigo - Criar política XACML no GCT_2

Um processo executando concorrentemente monitora o repositório, e caso haja novas políticas, as mesmas são enviadas ao PAP. Se o processo local obtiver uma resposta positiva do PAP, ou seja, a nova política foi realmente persistida, então a mesma pode ser removida do repositório local ao GCT_2. Este procedimento objetiva não interferir no tempo de resposta da requisição.

```
1.      To_Saml(date, right, principal)
2.      BEGIN
3.              assertion.set_Id(Some_Random_Id);
4.              assertion.set_Issue_Instant(new DateTime());
5.              assertion.set_Issuer(Sts_Id);
6.              assertion.set_Subject(principal)
7.              assertion.set_Conditions(date.get_Not_Before);
8.              assertion.set_Conditions(date.get_Not_After);
9.              assertion.set_ Authentication_Context(SPKI);
10.             assertion.set_Attribute(right.get_Action);
11.             ws_trust = Wrap_In_Ws_Trust (assertion);
12.             return ws_trust;
13.     END
```

Figura 7.4: Pseudocódigo - Criar asserção SAML no GCT_2

Após a geração da asserção SAML (Figura 7.4) a partir da autorização transportada

pela cadeia de certificados SPKI/SDSI, a mesma é encapsulada pela WS-Trust (linha *11*) e retornada para o PEP (linha *12*) para que este libere o acesso de acordo com a asserção SAML.

A Figura 7.5 mostra a asserção SAML (transportando as expressões de autenticação e atributo) gerada a partir de um certificado já reduzido (Figura 4.1). A asserção SAML contém o emissor (linha *4*), o *principal* SPKI/SDSI (*i.e.*: sujeito SAML, linha *8*), a interseção das datas de validade (linha *12*), sob qual contexto o *principal* foi autenticado (linha *16*) e qual operação o mesmo está autorizado a 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="2008-04-5T14:14:11.781Z">
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:AttributeValue xmlns: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 7.5: Asserção SAML gerada a partir da cadeia reduzida de certificados SPKI/SDSI

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

Na geração da política XACML a partir do certificado SPKI/SDSI reduzido (Figura 4.1) a linha *2* (Figura 7.6) transporta qualquer comentário que o certificado SPKI/SDSI possa conter. A linha *8* transporta o sujeito da política (*i.e.*: *principal* SPKI/SDSI) e a linha *18* descreve o recurso alvo.

Considerando que a cadeia de certificados é válida, a linha *26* específica por padrão o valor "*Permit*", pois em caso contrário a regra de política XACML não seria gerada. A linha

34 define a operação autorizada. As linhas *47* e *55* transportam o período de tempo durante o qual o *principal* (*i.e.*: sujeito da política XACML) tem direito de acesso ao recurso. Como haverá apenas uma regra de política XACML (linha *26* a *59*) gerada a partir de cada certificado SPKI/SDSI, foi adotado o algoritmo de combinação de regras "*first-applicable*" (linha *1*) para avaliação de políticas.

```
1. <Policy PolicyId="GeneratedReadPolicy" RuleCombiningAlgId="first-applicable">
2.  <Description>This statement applies to this principal at corporation.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.corporation.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.    <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
42.      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than-or-equal">
43.        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-one-and-only">
44.          <EnvironmentAttributeDesignator AttributeId="environment-id" DataType="dateTime"/>
45.        </Apply>
46.        <AttributeValue DataType="dateTime">
47.          2008-04-21T23:59:59.000000000
48.        </AttributeValue>
49.      </Apply>
50.      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than-or-equal">
51.        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-one-and-only">
52.          <EnvironmentAttributeDesignator AttributeId="environment-id" DataType="dateTime"/>
53.        </Apply>
54.        <AttributeValue DataType="dateTime">
55.          2008-07-18T01:00:00.000000000
56.        </AttributeValue>
57.      </Apply>
58.    </Condition>
59.  </Rule>
60.  <Rule RuleId="FinalRule" Effect="Deny"/>
61. </Policy>
```

Figura 7.6: Política XACML gerada a partir da cadeia reduzida de certificados SPKI/SDSI

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 7.6).

Na geração das políticas XACML a partir do certificado SPKI/SDSI reduzido (Figura 4.1) foi assumida a seguinte abordagem: *tudo o que não for explicitamente permitido é por padrão negado* (linha *60*, Figura 7.6). 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*).

7.4 Avaliação do Protótipo

Na avaliação do protótipo foram considerados dois cenários. No primeiro cenário foi avaliado o impacto que entidades importantes da proposta têm em relação ao todo, quando não se aplica segurança em nível de mensagem SOAP. Depois, foram avaliadas as mesmas entidades nas mesmas condições, porém considerando o uso de assinatura digital, cifragem e *timestamp* para prover proteção fim-a-fim às mensagens SOAP, usando a WS-Security.

Na avaliação com os modelos *provisioning* e *outsourcing*, o tamanho da mensagem de requisição enviada do *principal* para o WS PEP (evento r_1, Figura 7.7) é sempre de 247 *Bytes* (Figura 7.8). A mensagem trocada entre WS PEP e WS PDP nas avaliações no modelo *outsourcing* (evento *av*, Figura 7.7) foi sempre de 976 *Bytes*.

Na avaliação baseada em cadeias de certificados SPKI/SDSI, a solicitação de acesso enviada pelo *principal* para o WS PEP (evento r_1, Figura 7.7) é de 4,30 *Kbytes* (Figura 7.8), sendo que cada certificado que compõe a cadeia tem um tamanho de 1,78 *Kbytes*. A requisição transporta uma cadeia de certificados de autorização SPKI/SDSI (2 certificados, 3 chaves) que concederá direitos de acesso ao *principal*.

Os cenários de avaliação descritos a seguir permanecem os mesmos, só se alteram os modelos empregados e o uso ou não de segurança WS.

Foram utilizados três computadores nos testes, todos interligados a uma rede *Ethernet* 10/100. O computador que hospeda o cliente é um Celeron M 1.50 *Gigahertz* com 760 *Megabytes* de RAM (*Random Access Memory*). Para hospedar os Serviços Web (*i.e.*: GCTs, LPAP, PAP, PDP, PEP) foram utilizados dois computadores AMD (*Advanced Micro Devices*) 3.00 *Gigahertz* com 1 *Gigabyte* de RAM.

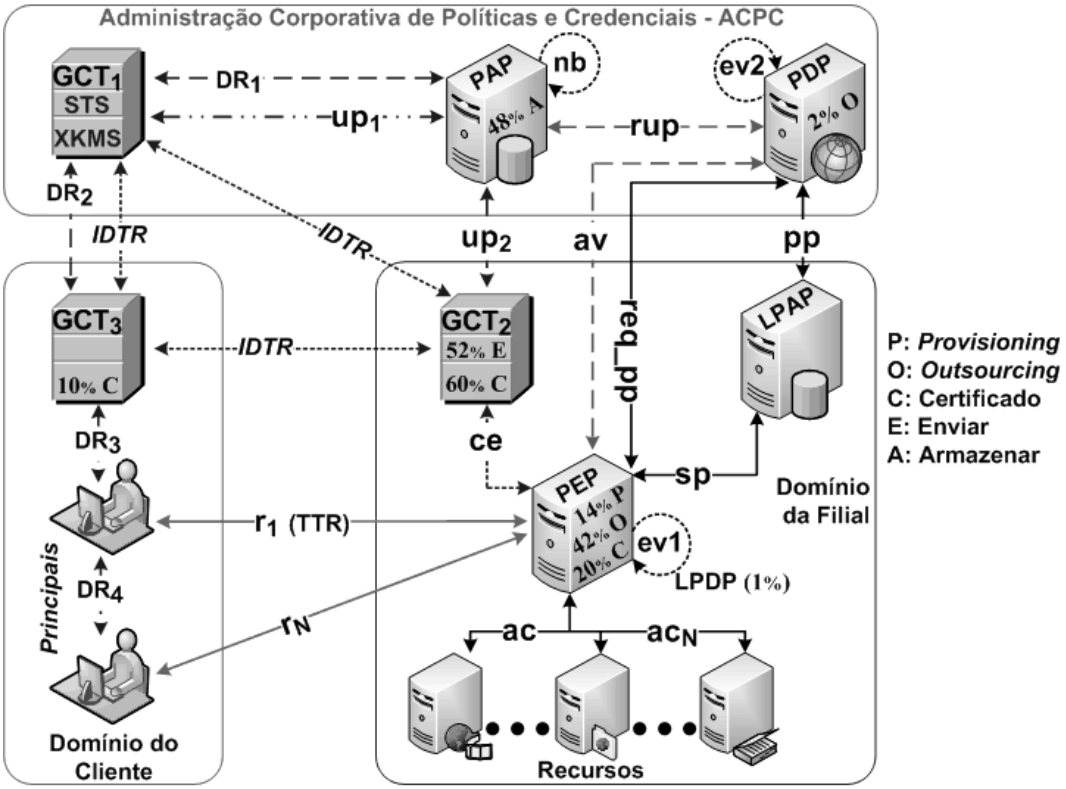

Figura 7.7: Interação entre as entidades do modelo com Serviços Web

Os percentuais citados como resultados da avaliação são sempre considerados em relação ao tempo total de resposta para o cliente (TTR, Figura 7.9). O tempo de resposta foi extraído fazendo-se uma média entre as requisições simultâneas enviadas pelo cliente para os Serviços Web. Foram executadas em média três seções com 400 interações em cada seção.

A quantidade de interações utilizada levou em consideração o aumento médio dos tempos de resposta, e optou-se pela escolha de um valor de interações considerado estável. Ou seja, analisou-se que os tempos de resposta se mantinham similares (*i.e.*: não sofriam grandes variações) mesmo aumentando-se o número de interações (*e.g.*: 800, 1000, 1200) efetuadas pelo cliente.

Na avaliação usando o modelo *provisioning* sem segurança o tempo gasto pelo WS PEP para atender uma requisição do cliente (evento r_1, Figura 7.7) equivale a 15% do TTR (Figura 7.10). Os processos de serialização/desserialização de mensagens e aplicação/imposição da decisão recebida do LPDP representam 14% do TTR (Figura 7.10). O tempo gasto na avaliação de políticas efetuada pelo LPDP (evento ev_1, Figura 7.7)

corresponde a 1% do TTR (Figura 7.10).

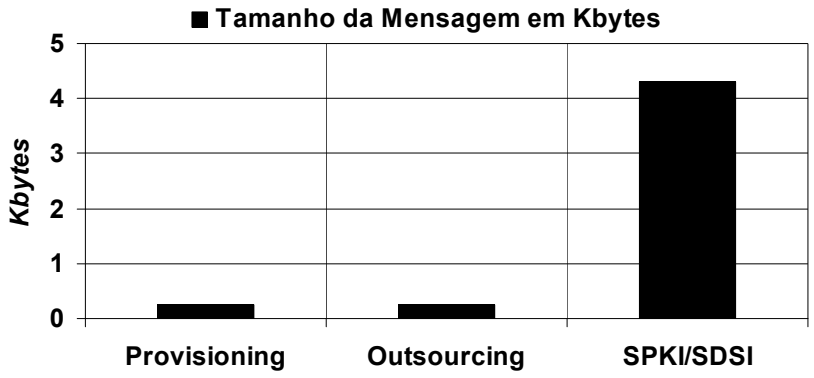

Figura 7.8: Tamanho da mensagem em cada modelo

Fazendo-se a avaliação agora no modelo *provisioning* com segurança WS (*i.e.*: assinatura, cifragem e *timestamp*) e comparando-se os resultados com o mesmo modelo sem segurança, o uso de segurança acresceu 34% no TTR (Figura 7.9) para cada requisição enviada pelo *principal*. Fazendo-se a mesma comparação e usando-se apenas a assinatura digital e *timestamp* o acréscimo no TTR foi de 32% (Figura 7.9).

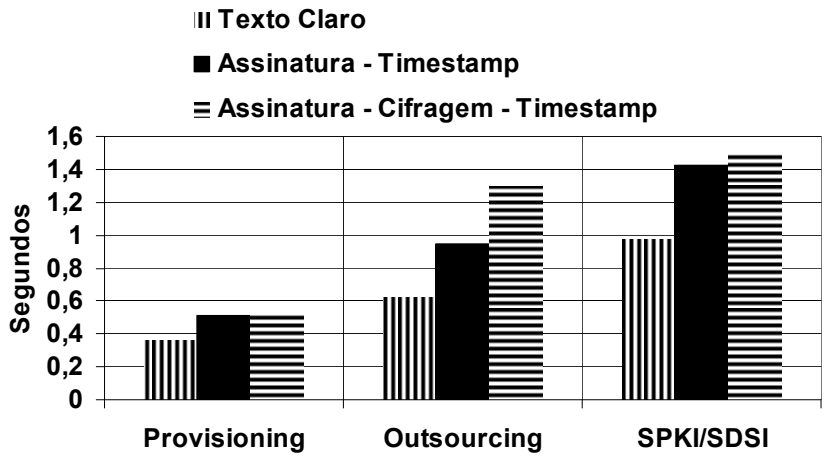

Figura 7.9: Tempo total de resposta para o cliente em cada modelo

Na avaliação no modelo *outsourcing* sem segurança o TTR teve um aumento de 41% em relação ao modelo *provisioning* sem segurança (Figura 7.9). Nesse modelo o tempo gasto

no WS PEP para atender uma requisição do *principal* equivale a 42% do TTR (Figura 7.10). O tempo gasto pelo WS PDP para avaliar uma política XACML correspondeu a 2% do TTR (Figura 7.10).

Na avaliação do modelo *outsourcing* com segurança WS (*i.e.*: assinatura, cifragem e *timestamp*), observou-se um acréscimo de 52% do TTR (Figura 7.9) em relação modelo *outsourcing* sem segurança WS. Usando-se somente a assinatura digital e *timestamp* o acréscimo foi de 34% (Figura 7.9).

Considerando-se a avaliação de acesso no modelo *provisioning*, o WS PEP é a entidade do modelo que gasta mais tempo manipulando mensagens XML e fazendo a aplicação/imposição da decisão retornada pelo LPDP, sendo que esta última é a entidade que menos consome tempo na arquitetura proposta.

Evidentemente, se a quantidade de políticas crescer muito, os percentuais devem sofrer variação. A avaliação foi feita com políticas simples, para que se medisse exatamente o impacto de cada entidade na arquitetura e não a capacidade de avaliação de grandes quantidades de políticas. Aliás, como a ferramenta de avaliação de políticas XACML é *mono-thread*, esta medida não seria conclusiva.

No modelo *outsourcing*, toda requisição enviada pelo *principal* para o Serviço Web PEP (evento r_1, Figura 7.7) é encaminhada e avaliada (eventos *av* e ev_2, respectivamente) pelo WS PDP. Isto faz com que o computador que hospeda o WS PEP fique aguardando a resposta do WS PDP. O *principal* por sua vez acaba tendo que esperar esse tempo adicional gasto na troca de mensagens entre WS PEP e WS PDP.

Na avaliação de acesso usando-se apenas certificados SPKI/SDSI sem segurança em nível de Serviços Web, o WS PEP gastou 20% do TTR (Figura 7.10) para serializar/desserializar as mensagens e fazer a aplicação/imposição do resultado obtido da asserção SAML (evento *ce*, Figura 7.7).

O WS GCT_2 (STS/XKMS) gastou 60% do TTR (Figura 7.10), sendo que 40% do TTR correspondem a validação da cadeia de certificados, criação da política XACML e armazenamento da mesma no repositório local. Somente 5% do TTR correspondem à geração da asserção SAML. O restante do tempo foi gasto em processos de serialização/desserialização das mensagens internamente ao WS GCT_2 (15% do TTR). Neste modo de operação o WS PEP não se tornou um ponto crítico do sistema, sendo que o *principal* e o WS PEP somente herdaram o tempo gasto pelo WS GCT_2 em uma primeira

avaliação de acesso.

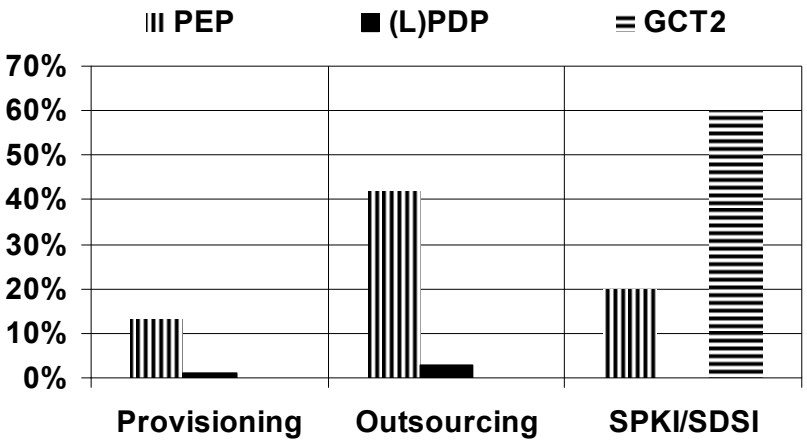

Figura 7.10: Percentagem de tempo gasta em cada entidade dos modelos

O mesmo procedimento de avaliação de acesso usando-se certificados SPKI/SDSI e segurança em nível de Serviços Web (*i.e.*: assinatura, cifragem e *timestamp*) foi feito obtendo-se um aumento de 33% no TTR (Figura 7.9), enquanto que usando-se somente assinatura digital e *timestamp*, o aumento foi de 31% (Figura 7.9) em relação à mesma requisição sem segurança.

Para cada novo certificado inserido na cadeia, observou-se um aumento médio no tempo de resposta de 15% sem segurança, e 11% com segurança. Estes acréscimos são referentes ao aumento no tamanho da mensagem (1,78 Kb para cada novo certificado) e principalmente devido ao aumento de complexidade para validar a cadeia de certificados. Visto que mais parâmetros devem ser verificados (*e.g.*: assinaturas, autorizações e datas de validade).

Este teste simulou a delegação de direitos entre *principals*. Os testes foram efetuados com cadeias compostas por diferentes quantidades de certificados, iniciando com 1 certificado até a quantidade de 6 certificados. O aumento das percentagens se manteve estável.

A quantidade de *principals* atendidos concorrentemente em cada modelo foi obtida realizando-se três seções de testes com diferentes quantidades de *principals* em cada seção (*i.e.*: iniciando com 10 *principals* e aumentando-se este valor até esgotar os recursos do sistema que hospeda os Serviços Web). Os valores expostos nos testes seguintes mostram a

quantidade de *principals* atendidos sem haver perda de requisições.

No modelo *provisioning*, o computador que hospeda o WS PEP atendeu 200 *principals* concorrentemente, sem segurança (Figura 7.11). O mesmo computador conseguiu atender 150 *principals* concorrentemente, quando se aplica segurança ao esquema. Usando-se somente assinatura digital e *timestamp*, o mesmo computador atendeu a 200 *principals* concorrentemente.

No modelo *outsourcing* sem segurança o computador que hospeda o Serviço Web PEP atendeu 150 *principals* concorrentemente (Figura 7.11). O mesmo computador atendeu 50 *principals* concorrentemente na avaliação com segurança. Usando somente assinatura digital e *timestamp*, o mesmo computador atendeu 100 *principals* concorrentemente.

O computador usado para hospedar o Serviço Web GCT_3 suportou atender a 75 *principals* concorrentemente sem segurança, e 50 *principals* concorrentemente com segurança WS (*i.e.*: assinatura, cifragem e *timestamp*).

Na verificação de acesso usando-se cadeias de certificados SPKI/SDSI (2 certificados, 3 chaves), o computador usado para hospedar o WS PEP atendeu 75 *principals* concorrentemente sem segurança, e 50 *principals* concorrentemente aplicando-se segurança em nível de Serviços Web (Figura 7.11).

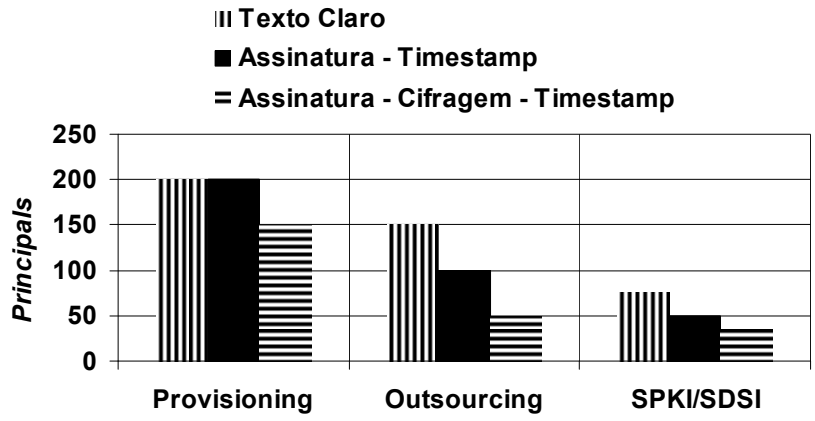

Figura 7.11: Quantidade de *principals* atendidos em cada modelo

O aumento da carga de processamento (*e.g.*: aumento no tempo de resposta e menor quantidade de clientes atendidos) mostrada na avaliação de acesso feita com a cadeia de certificados SPKI/SDSI com segurança se deve, também, ao fato de que cada certificado que

compõe a cadeia já possui sua própria assinatura digital. Além da assinatura digital que vincula a cadeia de certificados a mensagem SOAP – segurança WS.

Considerando o desempenho no caso de uso com segurança é possível observar que se a confidencialidade (cifragem) da mensagem não for importante, se ganha segurança com a assinatura digital (garantia de integridade, autenticidade e não-repúdio) e com o *timestamp* (evitando o ataque de mensagens antigas) sem perder significativamente o desempenho.

O tempo gasto pelo WS GCT_3 (Figura 7.7) para gerar e enviar a cadeia de certificados para o *principal* (evento DR_3) corresponde a 10% do TTR. Localmente, 3% são gastos na serialização/desserialização de mensagens, e 7% são gastos na criação do certificado e na montagem da cadeia de certificados SPKI/SDSI. A mesma requisição com segurança teve um aumento de 55% no TTR.

Para realizar o envio das novas políticas para o WS PAP da corporação (evento up_2, Figura 7.7), o WS GCT_2 gastou 52% do TTR para recuperar a política do repositório, enviá-la e apagar a mesma, caso tenha recebido confirmação positiva do WS PAP. O WS PAP gastou 48% do TTR para realizar o armazenamento da nova política XACML e retornar a mensagem de confirmação ao WS GCT_2. O mesmo procedimento executado com segurança aumentou em 71% o TTR.

A Figura 7.12 mostra a diferença média para se executar o provisionamento de diferentes tamanhos de políticas XACML considerando segurança ou não em nível de mensagem SOAP (*i.e.*: WS-Security) Como era de se esperar, o tempo de envio de cada política XACML aumenta em relação ao tamanho da mesma.

A percentagem média de tempo para se provisionar cada política XACML foi extraída diretamente no WS PDP, ou seja, após o WS PEP efetuar a solicitação de provisionamento (evento *req_pp*, Figura 7.7), o WS PDP recupera as políticas do WS PAP (evento *rup*) e as envia ao WS LPAP (evento *pp*).

A Figura 7.12 mostra a percentagem média de tempo que o WS PDP gastou para realizar 300 interações em cada política XACML. Ao final do processo, o WS PEP recebe uma mensagem indicando se o provisionamento foi efetuado com sucesso ou não.

O computador usado para hospedar o WS PDP suportou atender a 150 *principals* (WS PEPs) concorrentemente sem segurança, e 75 *principals* (WS PEPs) concorrentemente com segurança.

Não foram efetuados testes com políticas XACML maiores do que 640 *Kbytes* visto

que uma política com este tamanho é relativamente complexa de ser criada. Geralmente uma política XACML é formada por pequenas regras aninhadas no mesmo documento ou fazendo-se referencia a outras regras mais simples armazenadas no repositório (através de identificadores (*i.e.*: IDs) ou URIs, por exemplo).

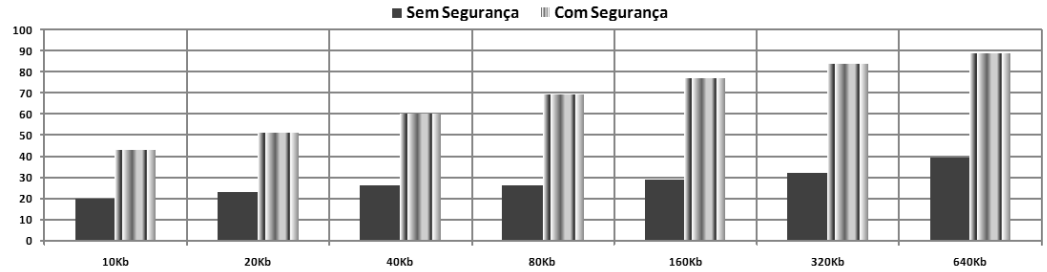

Figura 7.12: Provisionamento de políticas – PDP para LPAP

Como regra geral, a avaliação mostra que o uso de certificados exige um tempo de processamento maior do que a avaliação baseada nos modelos *provisioning* ou *outsourcing*. Porém, este tempo é gasto somente em uma primeira solicitação de acesso a um recurso, quando o *principal* apresenta a cadeia de certificados SPKI/SDSI para o WS PEP.

O WS PEP precisa, então, enviar a cadeia para o WS GCT_2, após isto, a política correspondente será gerada e a próxima avaliação de acesso, referente ao *principal* em questão, acontece de acordo com o modelo *provisioning*. Adicionalmente, as políticas em tal esquema foram criadas automaticamente, sem a intervenção dos administradores (corporativo ou da filial).

Supondo que o administrador humano gaste 2 segundos para gerar uma política XACML para um determinado *principal* e armazenar a mesma no repositório, em nossa proposta esse tempo cai para 0,404 segundos quando se utiliza uma cadeia de certificados composta de 2 certificados e 3 chaves (3 *principals*). Ou seja, supostamente, nossa proposta é 5 vezes mais rápida que o procedimento envolvendo o administrador humano. Admitindo que o administrador humano gaste 2 segundos para realizar a tarefa.

O procedimento descrito acima é executado com garantia de que todas as novas políticas geradas provêem de uma fonte confiável (*i.e.*: a cadeia de certificados). Além disto, as políticas estão menos propensas a introdução de erros manuais durante sua criação, reduzindo o consumo de tempo de administradores humanos para gerenciá-las.

O modelo *outsourcing* além de não ser fracamente acoplado, pela dependência do WS PDP, é computacionalmente mais custoso que o modelo *provisioning*, dificultando a escalabilidade dos Serviços Web. Enquanto, que o modelo baseado em certificados SPKI/SDSI e *provisioning* é realmente fracamente acoplado, dado a autonomia de operação e baixa necessidade de troca de mensagens entre as entidades da arquitetura.

O modelo baseado em certificado e provisionamento é uma ótima combinação para diminuir o custo humano e computacional de gestão das políticas, favorecendo a escalabilidade e autonomia dos sistemas locais sem infringir nenhuma regra imposta pela administração global da corporação. Enquanto isso, o modelo *outsourcing*, além de centralizador, tem um alto custo com troca de mensagens na rede, dificultando a escalabilidade e indo contra o princípio do fraco acoplamento proposto pelos Serviços Web.

7.5 Conclusão

Este capítulo apresentou as avaliações do protótipo, bem como um breve comparativo entre os diferentes modelos utilizados em sistemas de controle de acesso. Os modelos baseados em políticas e o modelo baseado em certificados.

O próximo capítulo apresenta as conclusões da presente dissertação de mestrado.

Capítulo 8

Conclusão

Este trabalho apresentou uma proposta para a administração unificada das políticas, suportando a escrita, avaliação e execução distribuída das mesmas. Observando que a escrita das políticas é feita sob demanda, de maneira automática e segura.

O esquema proposto oferece uma alternativa para o *principal* (cliente) obter os direitos requeridos, quando o mesmo não consta no repositório de políticas da Administração Corporativa de Políticas e Credenciais (ACPC) ou LPAP. Contrastando com a maneira clássica de controle acesso que no caso simplesmente negaria o acesso.

O administrador do domínio cliente ou da filial da corporação pode emitir um certificado de autorização para o *principal* obter os direitos requeridos, e ser incluído automaticamente no repositório corporativo (*i.e.*: PAP) e da filial (*i.e.*: LPAP).

O administrador do domínio cliente ou da filial pode conceder direitos, pois possui relações de confiança baseada na troca de certificados com os provedores que seus *principals* acessam.

O SPKI/SDSI, por ser independente da tecnologia subjacente, dá suporte a um esquema de transporte de direitos e estabelecimento de relações de confiança que pode facilmente transpor domínios de segurança. Da mesma forma, os Serviços Web oferecem uma infra-estrutura com protocolos e serviços de segurança que facilita a interoperabilidade em ambientes heterogêneos.

O modelo proposto pode operar automaticamente no controle de políticas *provisioning* e *outsourcing*. Porém, em geral, a operação é no modelo *provisioning*, pois se diminui a troca de mensagens favorecendo a autonomia local.

As características abordadas neste trabalho não são consideradas em nenhum outro trabalho relacionado, seja a operação nos dois modelos *provisioning* ou *outsourcing*, ou no

uso do SPKI/SDSI como fonte de dados para geração de políticas de forma automática. Esta abordagem libera o administrador da atividade de escrita de políticas de granularidade fina (*i.e.*: específicas para cada domínio ou mecanismo).

A abordagem proposta não depende de infra-estrutura de servidores de autenticação e autorização para transpor domínios de segurança, contribuindo para o baixo acoplamento do sistema. Porém, é compatível com mesma no ambiente de Serviços Web.

O modelo proposto tolera a indisponibilidade da Administração Corporativa de Políticas e Credenciais, pois há uma cópia local das políticas aplicáveis ao domínio de cada filial. Além disto, se as políticas na filial 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 Administração Corporativa de Políticas e Credenciais esteja ativa.

O esquema de segurança proposto para Serviços Web permite baixo acoplamento, pois aplica provisionamento de políticas e certificados de autorização SPKI/SDSI. O protótipo mostra a viabilidade da proposta que pode ser aplicada em outros tipos de cenários.

Outras considerações importantes quando as avaliações do protótipo foram feitas no capítulo anterior.

8.1 Trabalhos Futuros

Aperfeiçoar a implementação do protótipo, inserir o controle de seções no acesso a recursos, explorar cenários mais motivadores, provar formalmente os algoritmos utilizados.

O modelo proposto poderia ser aplicado para configurar dispositivos de controle de acesso que possuem conexão instável e ou limitada, como é o caso de redes *ad hoc*, por exemplo. O que é necessário avaliar neste tipo de ambiente, é se possíveis inconsistências nas políticas, geradas por perda mais prolongada de comunicação entre as entidades de controle, não trarão prejuízos significativos para a segurança do ambiente.

Referências Bibliográficas

[1] J. T. Howerton, "Service-Oriented Architecture and Web 2.0," in CIO Corner - IT Pro, 2007.

[2] E. R. Mello and J. S. Fraga, "Mediation of Trust across Web Services," in ICWS'05. IEEE, 2005.

[3] E. T. Camargo, "Transposição de Autenticação em Arquiteturas Orientadas a Serviço Através de Identidades Federadas.," in PGEEL. Dissertação de Mestrado. Florianópolis. UFSC, 2006.

[4] A. H. Karp, "Authorization-Based Access Control for the Services Oriented Architecture," in C5'06. IEEE, 2006.

[5] OASIS. Reference Model for Service Oriented Architecture v 1.0. Access: Sep. 2007. Available at: http://www.oasis-open.org/specs/index.php#soa-rmv1.0.

[6] W3C. Web Services Architecture. Access: Sep. 2007. Available at: http://www.w3.org/TR/ws-arch/.

[7] E. Ort, "Service-Oriented Architecture and Web Services: Concepts, Technologies, and Tools," v 1. SUN, 2005.

[8] W3C. Extensible Markup Language - XML v 1.1. Access: Sep. 2007. Available at: http://www.w3.org/TR/xml11/.

[9] W3C. XML Schema Part 0: Primer Second Edition v 2.0. Access: Sep. 2007. Available at: http://www.w3.org/TR/xmlschema-0/.

[10] W3C. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). Access: Sep. 2007. Available at: http://www.w3.org/TR/soap12-part1/.

[11] OASIS. Web Services Security: SOAP Message Security 1.1 - WS-Security v 1.1. Access: Sep. 2007. Available at: http://www.oasis-open.org/specs/index.php#wssv1.1.

[12] L. Srinivasan and J. Treadwell, "An Overview of Service-oriented Architecture, Web Services and Grid Computing," v 0.2. HP, 2005.

[13] W3C. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language v 2.0. Access: Sep. 2007. Available at: http://www.w3.org/TR/wsdl20/.

[14] J. Epstein, S. Matsumoto, and G. McGraw, "Software Security and SOA," in IEEE Security & Privacy IEEE, 2006.

[15] W3C. XML Encryption Syntax and Processing Access: Sep. 2007. Available at: http://www.w3.org/TR/xmlenc-core/.

[16] W3C. XML-Signature Syntax and Processing Access: Sep. 2007. Available at: http://www.w3.org/TR/xmldsig-core/.

[17] W3C. Canonical XML v 1.0. Access: Sep. 2007. Available at: http://www.w3.org/TR/xml-c14n.

[18] WS-I. Basic Profile v 1.2. Access: Sep. 2007. Available at: http://www.ws-i.org/Profiles/BasicProfile-1.2.html.

[19] WS-I. Basic Security Profile v 1.0. Access: Sep. 2007. Available at: http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html.

[20] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3280. IETF. Access: August 2008. Available at: http://www.ietf.org/rfc/rfc3280.txt.

[21] C. Neuman, T. Yu, S. Hartman, and K. Raeburn. The Kerberos Network Authentication Service. RFC 4120. IETF. Access: August 2008. Available at: http://www.ietf.org/rfc/rfc4120.txt.

[22] OASIS. WS-Trust 1.3 Access: Sep. 2007. Available at: http://www.oasis-open.org/specs/index.php#wstrustv1.3.

[23] NIST. Entity Authentication Using Public Key Cryptography. FIPS PUB 196. Access: Sep. 2007. Available at: http://csrc.nist.gov/publications/fips/fips196/fips196.pdf.

[24] OASIS. Web Services Federation Language - WS-Federation v 1.0. Access: Sep. 2007. Available at: http://www.oasis-open.org/committees/documents.php?wg_abbrev=wsfed.

[25] W3C. Web Services Policy 1.5 - Primer v 1.5. Access: Sep. 2007. Available at: http://www.w3.org/TR/ws-policy-primer/.

[26] W3C. Web Services Policy 1.5 - Attachment v 1.5. Access: Sep. 2007. Available at: http://www.w3.org/TR/ws-policy-attach/.

[27] OASIS. WS-SecurityPolicy v 1.2. Access: Jan. 2008. Available at: http://www.oasis-open.org/specs/index.php#wssecpolv1.2.

[28] W3C. XML Key Management Specification - XKMS v 2.0. Access: Sep. 2007.

Available at: http://www.w3.org/TR/xkms2/.

[29] OASIS. Service Provisioning Markup Language - SPML v 2. Access: Sep. 2007. Available at: http://www.oasis-open.org/specs/index.php#spmlv2.0.

[30] OASIS. Service Provisioning Markup Language - SPML v 1.0. Access: Sep. 2007. Available at: http://www.oasis-open.org/committees/download.php/4137/os-pstc-spml-core-1.0.pdf.

[31] R. L. Rivest and B. Lampson. SDSI - A Simple Distributed Security Infrastructure. Massachusetts Institute of Technology. Access: Sep. 2007. Available at: http://theory.lcs.mit.edu/~rivest/sdsi10.html.

[32] C. Ellison, B. Frantz, B. Lampson, R. L. Rivest, B. Thomas, and T. Ylonen. SPKI Certificate Theory. RFC 2693. IETF. Access: Sep. 2007. Available at: http://www.ietf.org/rfc/rfc2693.txt.

[33] A. O. Santin, "Teias de Federações: Uma Abordagem Baseada em Cadeias de Confiança para Autenticação, Autorização e Navegação em Sistemas de Larga Escala," in PGEEL. Tese de Doutorado. Florianópolis. Universidade Federal de Santa Catarina - UFSC, 2004.

[34] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen. Policy Core Information Model - PCIM. RFC 3060. IETF. Access: Sep. 2007. Available at: http://www.ietf.org/rfc/rfc3060.txt.

[35] A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, and S. Waldbusser. Terminology for Policy-Based Management. RFC 3198. IETF. Access: Sep. 2007. Available at: http://www.ietf.org/rfc/rfc3198.txt.

[36] R. Yavatkar, D. Pendarakis, and R. Guerin. A Framework for Policy-based Admission Control. Internet Engineering Task Force - IETF. Access: Available at.

[37] R. Yavatkar, K. Chan, J. Seligson, D. Durham, and K. McCloghrie. Common Open Policy Service Usage for Policy Provisioning (COPS-PR). Internet Engineering Task Force - IETF. Access: Available at.

[38] OASIS. Assertions and Protocols for the OASIS Security Assertion Markup Language - SAML v 2.0. Access: Sep. 2007. Available at: http://www.oasis-open.org/specs/index.php#samlv2.0.

[39] M. Lorch, S. Proctor, R. Lepro, D. Kafura, and S. Shah, "First Experiences Using

XACML for Access Control in Distributed Systems," in ACM Workshop on XML Security Fairfax, VA, USA ACM and Sun Microsystems, 2003.

[40] OASIS. eXtensible Access Control Markup Language - XACML v 2.0. Access: Sep. 2007. Available at: http://www.oasis-open.org/specs/index.php#xacmlv2.0.

[41] K. Irwin, T. Yu, and W. H. Winsborough, "On the Modeling and Analysis of Obligations," in CCS'06 Alexandria, Virginia, USA. ACM, 2006.

[42] OASIS. SAML 2.0 profile of XACML v2.0 Access: Sep. 2007. Available at: http://www.oasis-open.org/specs/index.php#samlv2.0.

[43] R. Bhatti, E. Bertino, and A. Ghafoor, "A Trust-based Context-Aware Access Control Model for Web-Services," in International Conference on Web Services (ICWS'04) IEEE, 2004.

[44] R. Bhatti, J. B. D. Joshi, E. Bertino, and A. Ghafoor, "Access Control in Dynamic XML-based Web-Services with X-RBAC," in The First International Conference on Web Services. Las Vegas, 2003.

[45] J. B. D. Joshi, E. Bertino, U. Latif, and A. Ghafoor, "Generalized Temporal Role Based Access Control Model (GTRBAC) (Part I) - Specification and Modeling," in Submitted to IEEE Transaction on Knowledge and Data Engineering. Available as CERIAS technical report 2001-47. IEEE.

[46] S. Hai-bo and H. Fan, "An Attribute-Based Access Control Model for Web Services," in Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'06) IEEE, 2006.

[47] A. Morcos, "A Java Implementation of Simple Distributed Security Infrastructure," in EECS. Master Dissertation. Massachusetts Institute of Technology, 1998, p. 58.

[48] OASIS. Universal Description Discovery & Integration - UDDI v 3.0.2. Access: Sep. 2007. Available at: http://uddi.org/pubs/uddi_v3.htm.

[49] OASIS. Web Services Reliable Messaging TC - WS-Reliability v 1.1. Access: Sep. 2007. Available at: http://docs.oasis-open.org/wsrm/ws-reliability/v1.1/wsrm-ws_reliability-1.1-spec-os.pdf.

[50] OASIS. Web Services Coordination - WS-Coordination v 1.1. Access: Sep. 2007. Available at: http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec-pr-01.pdf.

[51] OASIS. Web Services Atomic Transaction - WS-AtomicTransaction v 1.1. Access: Sep. 2007. Available at: http://docs.oasis-open.org/ws-tx/wstx-wsat-1.1-spec-pr-

01.pdf.

[52] J. D. Edwards, "Oracle SPML Integration," Disponível em http://www.oracle.com/identity. Acesso em 10/03/2006.

[53] D. E. Clarke, "SPKI/SDSI HTTP Server - Certificate Chain Discovery in SPKI/SDSI," in Electrical Engineering and Computer Science. Master Thesis. Massachusetts. Massachusetts Institute of Technology - MIT, 2001.

[54] K. Chan, J. Seligson, D. Durham, S. Gai, K. McCloghrie, S. Herzog, F. Reichmeyer, and R. Yavatkar. Common Open Policy Service Usage for Policy Provisioning - COPS-PR. RFC 3084. IETF. Access: Sep. 2007. Available at: http://www.ietf.org/rfc/rfc3084.txt.

[55] T. E. Squair, "Serviço de Controle de Acesso Utilizando Modelo de Provisionamento," in PPGIa. Dissertação de Mestrado. Curitiba. Pontifícia Universidade Católica do Paraná - PUCPR, 2005.

Apêndice A

Arquitetura Orientada a Serviço e Serviços Web

A.1. Modelo Conceitual da AOS

O modelo conceitual da AOS, de acordo com a especificação, pode ser descrito segundo os três seguintes conceitos [5]:

- ***Visibilidade:*** capacidade do *principal* e provedor poder encontrar um ao outro. Propriedade relacionada ao protocolo de transporte, políticas, semântica do serviço, etc.
- ***Interação:*** atividade que consiste no uso do serviço, tipicamente feita pela troca de mensagens.
- ***Efeito:*** o resultado obtido do uso de um serviço. Pode ser a devolução de uma informação, a mudança no estado dos agentes envolvidos (*e.g.*: *principal* e provedor), etc.

A Figura A.1 mostra as principais entidades inerentes a um ambiente orientado a serviço [12], bem como a possível interação entre as mesmas.

Depois de criada a implementação de um serviço em um provedor, bem como a respectiva descrição de como acessá-lo, o provedor necessita tornar a descrição acessível para o *principal*.

Uma das maneiras seria publicar a descrição da interface em um serviço de registro (evento *1*, Figura A.1) onde ficaria disponível para ser localizada (evento *2*) e adquirida por *principals* (evento *2.1*). Estando munido de todas as informações necessárias (evento *2.1*), o *principal* pode implementar o serviço cliente e realizar a invocação (evento *3*) do serviço

disponível no provedor.

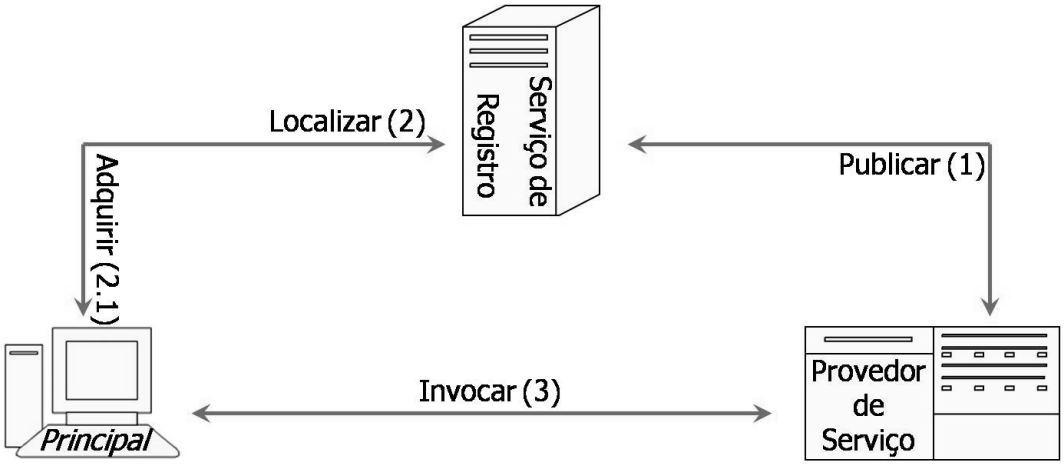

Figura A.1: Interação entre as entidades segundo a Arquitetura Orientada a Serviço

A.2. Arquitetura Geral dos Serviços Web

A arquitetura geral dos Serviços Web [6] divide-se em quatro modelos principais, sendo estes: o modelo orientado a mensagem; o modelo orientado a serviço; o modelo orientado a recurso; e o modelo de política. Todos estes modelos agrupados descrevem a arquitetura completa, pois cada um abrange determinada área, se relacionando aos Serviços Web de diferentes maneiras.

A.2.1. Modelo Orientado a Mensagem (MOM)

O Modelo Orientado a Mensagem (*Message Oriented Model* – MOM) [6] compreende os aspectos da arquitetura tais como: a transmissão da mensagem; a relação entre emissores e receptores; a estrutura da mensagem; e sua relação com os processos.

Para que a mensagem possa ser entregue, esta deve possuir o endereço do destinatário, cujo tipo depende do transporte a ser usado. Essa informação pode ser inserida na própria mensagem, geralmente dentro do envelope (*i.e.*: elemento que delimita o início e o fim da mensagem).

O MOM define políticas relacionadas à entrega da mensagem, sendo uma combinação da mensagem com o mecanismo de transporte. Estas políticas podem originar-se de descrições da própria mensagem, e ou estar intrínsecas no mecanismo de transporte.

A mensagem é uma estrutura de dados transmitida do emissor para o receptor. Geralmente o formato da mensagem está de acordo com protocolo SOAP (Seção 2.4), o qual possui três campos principais, sendo estes: o envelope; o cabeçalho; e o corpo.

Uma mensagem pode estar associada a um contexto, o qual determina se a mensagem recebida é a esperada. Freqüentemente o destinatário final da mensagem é o responsável por completar o seu processamento, e se necessário, devolver uma resposta. Embora toda mensagem possua um emissor, a identidade do mesmo pode estar oculta para os agentes intermediários no caminho da mensagem.

O MOM descreve um padrão genérico para troca de mensagens entre *principals* e provedores. Este é descrito por máquinas de estado as quais definem: o fluxo das mensagens; a correlação entre as mensagens; a manipulação de faltas que podem surgir durante a troca de mensagens; e o término normal e ou anormal de qualquer troca de mensagens.

Os procedimentos acima citados visam reduzir a freqüência de erros, permitindo que uma das entidades que está participando da interação tome decisões, e execute ações para corrigir erros (*e.g.*: confirmações de recebimento, correlação, re-envio, etc.).

A.2.2 Modelo Orientado a Serviço (SOM)

O Modelo Orientado a Serviço (*Service Oriented Model* – SOM) [6] focaliza nos aspectos da arquitetura relacionando o serviço a uma ação, com propósito de interpretar as relações entre o provedor, o serviço que este provê, e a solicitação do *principal*. Uma ação é tipicamente executada para cumprir alguma tarefa, sendo que *principal* e provedor não dependem diretamente um do outro para realizar a tarefa.

O SOM se estrutura no modelo anterior, mas seu foco está na ação que pode ser executada como resultado do recebimento de uma mensagem. O próprio envio de uma mensagem ou a alteração de variáveis de ambiente no domínio do *principal* ou do provedor é caracterizado como um tipo de ação.

O proprietário de um WS pode divulgar quais são as funcionalidades suportadas pelo serviço. Um *principal* necessitando de tal funcionalidade pode selecionar o serviço com base

nesta declaração. É possível, também, que o WS requisite uma capacidade particular para o *principal* que deseja fazer uso do serviço. Ou seja, alguma propriedade que deve ser implementada pelo *principal* (*e.g.*: o uso de determinado mecanismo de segurança).

Os serviços podem ser descritos como conjuntos de tarefas relacionadas, podendo estas ser realizadas por um ou mais agentes. A única parte exposta de um serviço é a sua interface, a qual define os diferentes tipos de mensagens um serviço recebe e envia.

A.2.3 Modelo Orientado a Recurso (ROM)

O Modelo Orientado a Recurso (*Resource Oriented Model* – ROM) [6] visa os aspectos da arquitetura no que diz respeito ao recurso, independentemente da função que o recurso possa ter ou prestar no contexto dos Serviços Web. Segundo este modelo, todo recurso deve ter um nome não ambíguo usado para sua identificação. O recurso também pode ser controlado por políticas de segurança referentes ao seu acesso.

Para se usar um recurso é necessário localizar a sua descrição, a qual contém informações sobre a localização (*e.g.*: URL), como acessar o recurso, quais políticas o regem e o resultado esperado ao se invocar o recurso. A descoberta da descrição de um recurso pode ocorrer de forma estática, durante a fase de desenvolvimento do *software*, ou dinâmica, com o *principal* interagindo diretamente com algum serviço de descoberta (*e.g.*: navegador *web*) para localizar um agente provedor de recurso apropriado.

A.2.4. O Modelo de Política (PM)

O Modelo de Política (*Policy Model* – PM) [6] contempla aspectos da arquitetura no que diz respeito a política, a segurança e a qualidade de serviço. As políticas são restrições no comportamento dos *principals*, provedores e serviços, e como estes executam acessos, ações, ou a prestação de serviços. A segurança aplica restrições para regrar o comportamento das ações, controlando o acesso de *principals* a serviços. Semelhantemente, a qualidade do serviço está relacionada com as restrições aplicadas no serviço.

Uma ação é qualquer ato que pode ser executado por um agente, possivelmente como resultado de receber uma mensagem, a qual pode resultar no envio de outra mensagem, caracterizando assim uma ação.

Principals e provedores de serviço podem estar situados em diferentes domínios, estando deste modo sujeitos as restrições de mais de uma política, definindo assim o alcance e a aplicabilidade das políticas. A política pode ser dividida em políticas de permissões e obrigações. Ou seja, políticas que servem para permitir o acesso ao recurso, e políticas que devem ser executadas em conjunto com o acesso ao recurso, por exemplo.

Associado com as políticas está o guardião de política, que é mecanismo usado para executar as políticas. Este pode estar subdividido em guardião de auditoria e guardião de permissão.

O guardião de auditoria é o mecanismo de aplicação e ou execução usado para monitorar o cumprimento das obrigações definidas na política, monitorando *principals*, provedores de serviços e seus recursos.

O guardião de permissão é o mecanismo usado para executar políticas de permissão, ou seja, assegurar que o uso de qualquer serviço está consistente com as políticas estabelecidas pelo seu proprietário. Tipicamente, o guardião de permissão está entre o *principal* e o serviço.

A.3. Descrição, Descoberta e Integração Universais (UDDI)

A UDDI (*Universal Description, Discovery and Integration*) [48] é um tipo de WS projetado para agir como um informante intermediário entre os consumidores e os provedores de Serviços Web. Ou seja, um repositório onde se publicam serviços fornecidos na *web*, cujo qual dispõe de mecanismos para registro e descoberta de informações sobre serviços (*e.g.*: nome do provedor e os serviços expostos por este).

Depois de criado o serviço e sua descrição (*i.e.*: a WSDL, evento *1*), esta pode ser publicada na UDDI (evento *2*, Figura A.2) onde ficará disponível para os *principals* acessarem (evento *3*), e recuperarem a descrição do serviço desejado (evento *3.1*). Com a WSDL, monta-se a implementação (evento *4*) para se realizar interações com determinado serviço (eventos *5* e *5.1*).

A especificação da UDDI apresenta um modelo de informação composto de instancias de estruturas de dados persistentes chamadas entidades. As entidades são expressas em XML e são persistentemente armazenadas em nodos UDDI.

O modelo de informação da UDDI é composto de instancias de entidades, dentre as

quais podem-se destacar:

- ***businessEntity:*** descreve a organização que prove o serviço (*e.g.*: nome, endereço físico, página da Internet, etc.).
- ***businessService:*** contem a descrição formal (*e.g.*: tipo do negocio, serviços, etc.), descrevendo a coleção de Serviços Web oferecidos por alguma organização descrita na *businessEntity*.
- ***bindingTemplate:*** contem informações técnicas necessárias para acessar o serviço (*e.g.*: URL, porta – endereço para o qual as mensagens são enviadas).
- ***tModel:*** descreve o modelo técnico, assim como a semântica dos Serviços Web, o protocolo usado por este, e o tipo da interface exposta pelo provedor de serviço.
- ***publisherAssertion:*** descreve a relação que a *businessEntity* tem com outra *businessEntity*.

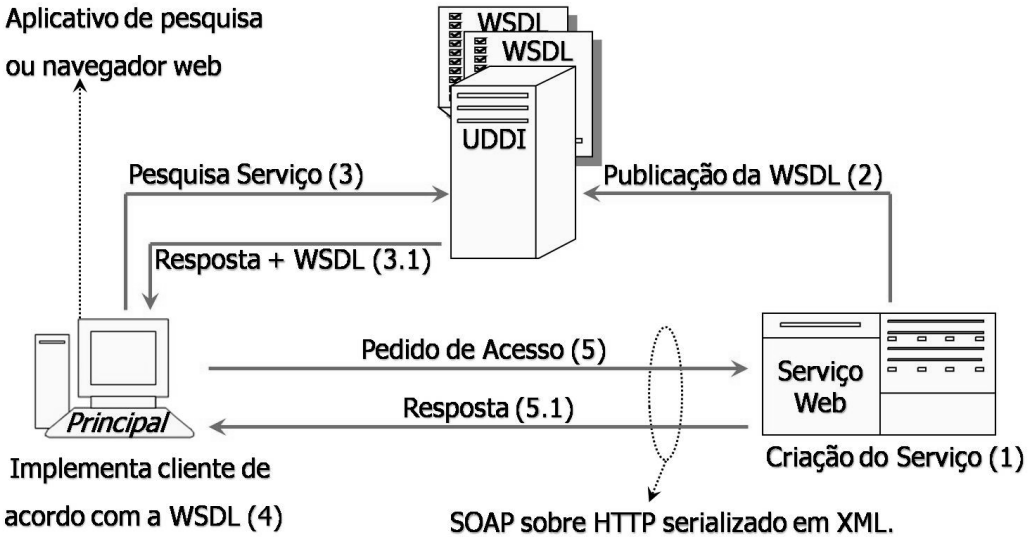

Figura A.2: Interação entre as entidades que compõem a arquitetura dos Serviços Web

Uma *businessEntity* contém um ou mais *businessServices*. Um *businessService* contém um ou mais *bindingTemplates*. Cada *bindingTemplate* refere-se a um *tModel*, que é a especificação de um serviço ligando a estrutura da *businessEntity* e a especificação técnica fornecida na *tModel*.

Podem ser criados outros tipos de diretórios UDDI para atender diferentes paradigmas

de negócios. Por exemplo:

- ***Nodo operador:*** um diretório global público, conhecido como Registro de Negócios UDDI (*UDDI Business Registry* – UBR), que consiste em vários nodos UDDI administrados por companhias como IBM, Microsoft, etc. Quando um provedor de serviço deseja publicar seus serviços, este vai ao UBR, se registra, e publica os documentos WSDL os quais serão replicados para todos os nodos UBR.
- ***Grupo:*** focalizado em um número específico de parceiros comerciais conhecidos, geralmente da mesma indústria. Concentrado em serviços realmente necessários, tentando proporcionar relacionamentos confiáveis entre os provedores e consumidores.
- ***Registros Privados:*** para usuários que desejam manter seus serviços privados, sendo que somente quem registrou o serviço (*e.g.*: indivíduo ou empresa) tem acesso ao conteúdo registrado. Registro privados podem ser implementados dentro dos limites organizacionais, provendo um repositório central para publicação de todos os serviços que a organização desenvolve.

Todo serviço registrado em um diretório UDDI deve receber um identificador único para evitar o conflito de nomes. A comunicação com o diretório UDDI pode ser feita através de mensagens SOAP, sendo possível também à interação manual através do navegador *web* (Figura A.2).

A.4. Especificações Adicionais para Serviços Web

Serviços Web dispõem de varias especificações, abrangendo diversas áreas. Exemplos de especificações utilizadas para prover uma troca mensagens mais confiável entre Serviços Web são:

- ***WS-Reliability*** [49]***:*** garante que as mensagens serão entregues, em ordem, e sem mensagens duplicadas.
- ***WS-Coordination*** [50]***:*** define protocolos de mensagens para coordenar tarefas executadas em aplicações distribuídas.

- ***WS-AtomicTransaction*** [51]***:*** estende a especificação anterior, definindo protocolos de mensagens para a coordenação e a realização de transações atômicas. Geralmente usada em aplicações distribuídas que fazem uso da propriedade *all-or-nothing*. Ou seja, todos os passos de uma transação são finalizados (*i.e.*: *commit*) ou descartados pelos participantes.

Os exemplos de especificações citadas acima foram projetados de forma a prover interoperabilidade, e certo coeficiente de garantia nas trocas de mensagens efetuadas em ambientes distribuídos e heterogêneos. As especificações podem ser estendidas para atender a uma gama de uso além dos necessários em Serviços Web.

A.5. Caso de Uso com a WS-Trust

A Figura A.3 mostra as entidades descritas na especificação WS-Trust, bem como a possível interação entre elas.

Tendo como exemplo o caso quando o *principal* deseja acessar um serviço protegido, este deve primeiramente montar um pedido solicitando uma credencial de acesso (*e.g.*: requisitando uma asserção SAML ou certificado, evento *1*, Figura A.3). A solicitação é enviada para o STS (RST: <*RequestSecurityToken*>, evento *1.1*) que fornecerá ao *principal* a credencial solicitada (RSTR: <*RequestSecurityTokenResponse*>, evento *1.2*) e adequada ao provedor de serviço.

A credencial solicitada pelo serviço pode ter seu formato descrito através da especificação WS-Policy. Um documento que pode ser publicado junto com a WSDL na UDDI. Basta então o *principal* especificar o formato da credencial dentro da mensagem RST (evento *1.1*) que será enviada ao STS. Estando de posse da credencial (evento *1.2*), o *principal* a envia para o provedor de serviço (evento *2*), provando que o mesmo é uma entidade confiável perante o STS.

O provedor, de posse do pedido e da credencial, avalia esta para ter certeza de que a mesma é autentica. A validação pode ser local (evento *3*), verificando-se a assinatura digital do STS. O provedor também pode criar uma solicitação endereçada ao STS emissor da credencial, para que este realize a validação da mesma (evento *3.1*). O STS pode retornar uma mensagem aferindo a validade da credencial para provedor de serviço (evento *3.2*).

A confiança mutua entre o provedor de serviço e o STS está baseada em um relacionamento confiável pré-estabelecido. Se o STS for capaz de validar a credencial corretamente, o provedor responde a solicitação do *principal* (evento *2.1*).

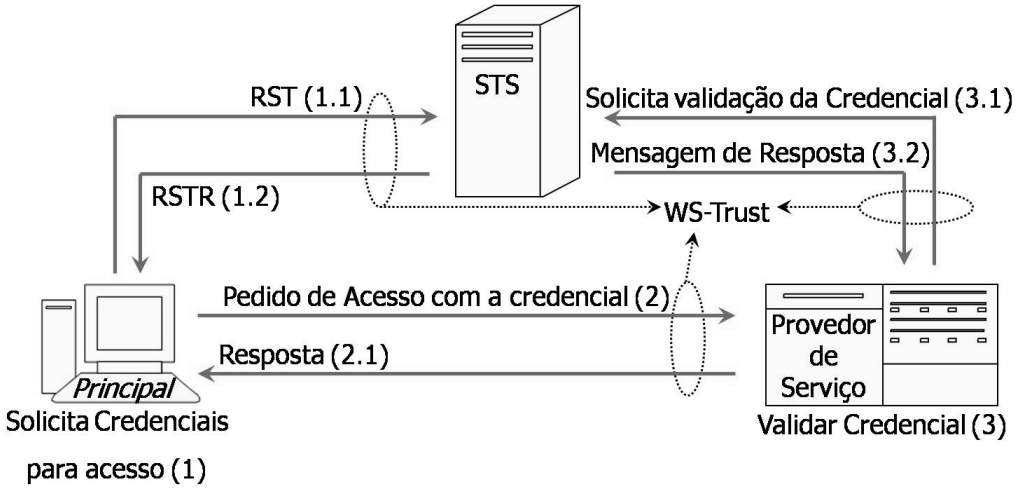

Figura A.3: Arquitetura segundo a WS-Trust

A.6. Web Services Federation Language (WS-Federation)

A especificação WS-Federation [24] tem como um de seus objetivos estabelecer relações confiáveis interdomínios, para permitir que *principals* confiáveis segundo as políticas de um domínio, possam fazer uso deste mesmo benefício mediante as políticas de outros domínios. Este objetivo seria alcançado garantindo ao *principal* que o mesmo não terá sua privacidade (*e.g.*: informação pessoal) revelada, seja esta interceptada durante o transporte na rede (*e.g.*: Internet) ou distribuída de forma não autorizada pelo domínio de origem.

A definição acima é referenciada com o termo identidade federada, ou seja, o *principal* não precisa ter suas informações pessoais registradas em todos os domínios com os quais este estabelece algum tipo de relacionamento. Neste caso, basta o domínio de origem do *principal* emitir uma credencial que seja aceita por outro domínio, atestando a confiabilidade que se pode ter sobre determinado *principal*. Deste modo, o receptor de determinada credencial (*e.g.*: domínio do provedor de serviços) fica livre do encargo de administrar todas as identidades dos *principals* que fazem uso de seus serviços.

Quando dois ou mais domínios estabelecem acordos sobre políticas de negócios e ou acordos técnicos, criando relacionamentos confiáveis entre os mesmos, e proporcionando a os produtores e consumidores maior aproveitamento do que um ou outro domínio tem a oferecer, se vem à tona o termo federação.

A WS-Federation foi desenvolvida por um conjunto de empresas (*e.g.*: BEA, IBM, Microsoft e VeriSign), agregando conceitos e modelos já existentes para Serviços Web, visando formar um domínio que abrange proporções maiores. Contudo, tentando oferecer a mesma segurança e interoperabilidade disposta em redes com tamanhos menores, e por conseqüência, mais fáceis de administrar.

Dentro do domínio de uma mesma organização, geralmente, a infra-estrutura de segurança que protege a empresa de possíveis ameaças externas e ou internas é a mesma, tornando a comunicação mais fácil dentro deste perímetro. Porém, quando se agregam muitos domínios ao grupo formando uma federação a diversidade aumenta. Isto torna o sistema mais heterogêneo e por conseqüência, mais difícil de transpor os domínios de origem sem esbarrar em infra-estruturas de segurança diferentes. Estas infra-estruturas aplicam outros formatos de políticas para garantir a proteção de cada domínio.

A WS-Federation propõe o uso de documentos com metadados, usados para descrever as políticas e outras características inerentes a cada domínio. A especificação define como tornar estes documentos acessíveis de maneira segura, assim, outros domínios podem se integrar a federação.

A especificação focaliza na camada de aplicação e nas mensagens utilizadas por esta para estabelecer um relacionamento confiável entre diferentes domínios. A WS-Federation combina alguns modelos definidos no próprio documento e modelos definidos em especificações como a WS-Security, WS-Trust e WS-Policy, entre outras. Este conjunto de padrões permite realizar o compartilhamento de identidades ou identificadores, atributos, expressões de autenticação e autorização, mantendo a autenticidade e a privacidade das informações nas declarações distribuídas entre diferentes domínios.

Segundo a WS-Federation, um domínio pode fornecer acesso a um recurso que este administra baseado na credibilidade e ou autenticidade de uma identidade declarada em outro domínio. Desde que a entidade declarante faça parte da mesma federação, sendo assim supostamente conhecida e confiável. Estando estas premissas corretas, o domínio detentor do recurso solicitado pelo *principal*, portador de tal identidade, pode tomar uma decisão bem

informada, fornecendo acesso de acordo com suas políticas locais.

A entidade declarante mencionada anteriormente pode ser um Provedor de Identidade (*Identity Provider* - IdP) confiável ao *principal*, ao STS e ao provedor do serviço. Estes dois últimos são os agentes que receberão a identificação aferida pelo IdP. Depois de autenticado, o *principal* pode solicitar que o STS forneça a credencial de segurança necessária para acessar o recurso almejado. O STS pode ser local ao domínio de origem do *principal*, ou pode estar instanciado no domínio detentor do recurso almejado.

Dependendo do tipo da estrutura implementada, a identificação ou autenticação (*e.g.*: asserção SAML) emitida pelo IdP pode ser apresentada diretamente ao STS instanciado no domínio destino (*i.e.*: provedor de serviço) ou apresentada diretamente ao guardião do recurso presente neste domínio. Se apresentada ao STS, este pode devolver uma autorização para ser apresentada diretamente ao guardião do recurso.

A credencial apresentada ao STS está sujeita as mesmas regras de validação descritas na especificação WS-Trust (Seção 3.3.2). A identificação emitida pelo IdP/STS presente no domínio de origem do *principal* pode vir acompanhada por um identificador opaco, fornecido pelo Serviço de Atributos e Pseudônimos (SAP) localizado no domínio do *principal*.

O identificador serve para proteger a identidade real do *principal*, dificultando assim o rastreamento ou revelação de suas informações pessoais. Neste contexto, pode haver também um Serviço de Atributos localizado no domínio de origem do *principal*. Os atributos fornecidos podem ser solicitados pelo provedor de serviço presente no domínio destino.

Os atributos podem fornecer informações sobre as qualificações do *principal*, por exemplo, oferecendo assim, o que foi descrito na especificação como uma melhor experiência ao usuário. Ou seja, oferecendo-lhe outros serviços de acordo com as informações adquiridas.

Os atributos são dependentes da implementação, podendo ser referentes a localização geográfica do usuário, informações registradas sobre conteúdos mais acessados, ou atributos que o *principal* julgou necessário tornar disponíveis para outros domínios. Exemplos de atributos podem ser o CEP, *e-mail*, cidade, *hobbies*, etc.

A Figura A.4 expressa um dos casos de uso possíveis expostos pela especificação. Neste caso em particular, o *principal* autentica-se no IdP/STS de seu domínio (evento *1*, Domínio A) para receber uma identidade, ou mesmo uma credencial que forneça a autorização de acesso ao serviço. A credencial pode vir acompanhada por um identificador opaco fornecido pelo serviço de pseudônimos (evento *1.1*).

Caso tenha recebido uma identidade, este segue para o STS do outro domínio (Domínio B) para obter a credencial necessária para acessar o recurso. A identidade apresentada pelo *principal* deverá ser validada pelo STS (evento *2.a*, Figura A.4) que pode trocar mensagens com o IdP emissor (Domínio A), ou valida-la localmente. Obtendo-se sucesso neste processo, é retornada uma credencial de autorização (evento *2.1*) para que o *principal* realize o acesso ao serviço.

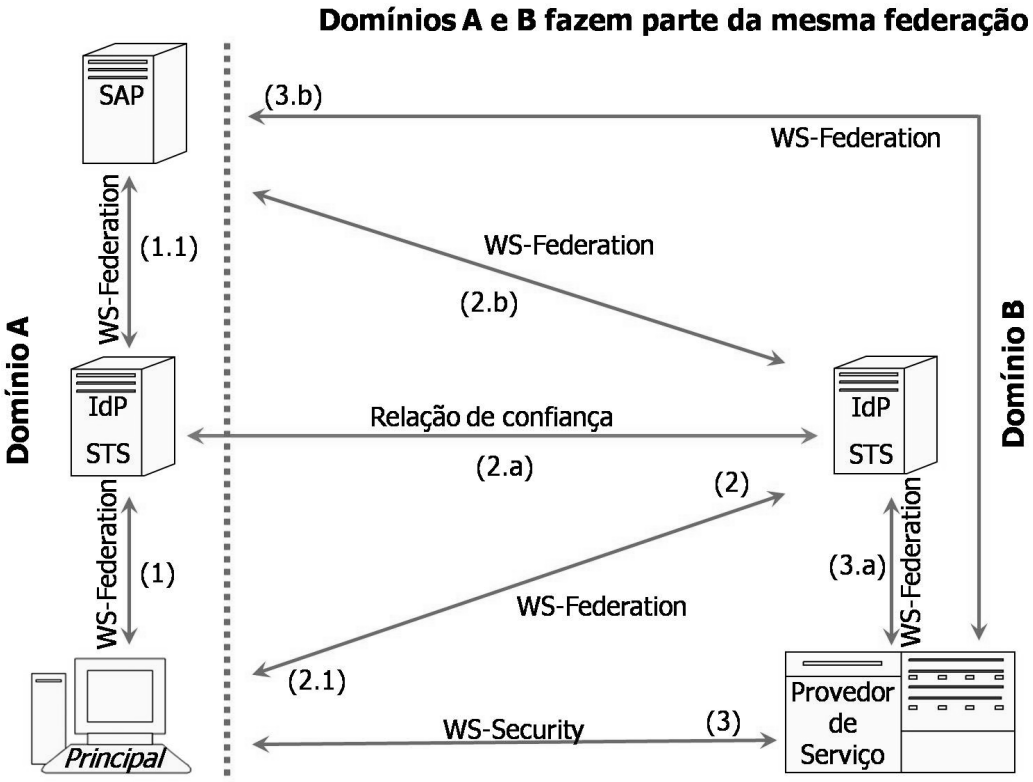

Figura A.4: Arquitetura segundo a WS-Federation

Caso o *principal* já tenha recebido a credencial de autorização (evento *1*, Figura A.4), o evento *2* não será executado. De posse desta credencial, o *principal* troca mensagens diretamente com o provedor do serviço (evento *3*, Domínio B) para concretizar o acesso ao recurso desejado. Evento este também sujeito a algum tipo de validação (evento *3.a*), responsabilidade esta transferida para o STS local (Domínio B).

O provedor de serviço pode recorrer ao Serviço de Atributos e Pseudônimos – SAP (evento *3.b*) presente no domínio de origem do *principal* (Domínio A) para recuperar atributos sobre o mesmo e talvez lhe fornecer um acesso distinto, ou melhor informado.

A especificação descreve como usar a WS-Federation com *principals* ativos, hábeis a emitir pedidos para Serviços Web e reagir a respostas do mesmo. O perfil ativo define que os próprios *principals* são entidades capazes, podendo fornecer as informações exigidas no processo de autenticação. Esta informação é carregada dentro do elemento <*Security*> no cabeçalho da mensagem SOAP. Isto permite a autenticação e autorização da solicitação.

O perfil passivo da WS-Federation descreve como implementar as funcionalidades da federação em um ambiente de *principal* passivo. O mais comum é o navegador *web*, pois este é incapaz de construir seu próprio pedido para um WS. Toda a responsabilidade pela criação de mensagens SOAP e envio de credenciais de maneira segura, ficaria a cargo do provedor de serviço ao qual o *principal* está vinculado.

A.7. Caso de Uso com a WS-Policy

Esta seção descreve um possível caso de uso da especificação WS-Policy e como a mesma pode auxiliar no estabelecimento de relacionamentos seguros entre diferentes entidades.

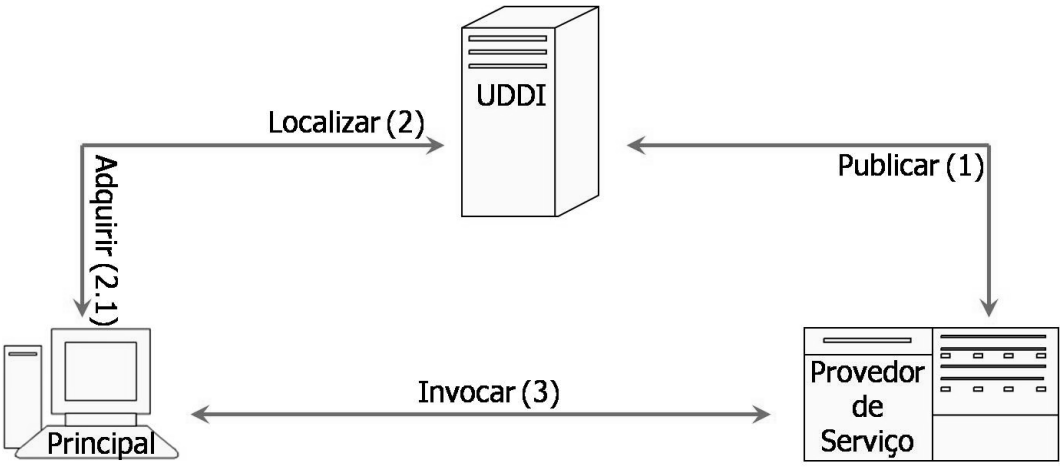

Figura A.5: Exemplo para obtenção do documento WS-Policy

Na Figura A.5, após o provedor de serviço ter criado a implementação do serviço, este cria o arquivo WS-Policy e o anexa ao documento WSDL. Procedimento este descrito na especificação WS-PolicyAttachment. Posteriormente, a WSDL é publicada em um diretório

UDDI (evento *1*).

O *principal* executa o procedimento de localização no diretório UDDI (evento *2*, Figura A.5). Tendo encontrado o serviço que atenda seus requisitos e necessidades, adquire a WSDL para poder implementar seu serviço cliente (evento *2.1*).

Como o arquivo WSDL possui anexado a si o documento WS-Policy, ou uma referencia para onde o mesmo pode ser encontrado, o *principal* é capaz de interpretar como e com quais parâmetros de segurança a interação com o WS irá ocorrer.

O provedor de serviço ao receber a solicitação de acesso (evento *3*, Figura A.5) confere se a mesma está de acordo com as respectivas restrições (*i.e.*: políticas) publicadas. Estando este procedimento correto, o *principal* obtém o acesso desejado.

A.8. Serviços Disponibilizados pela Especificação XKMS

A especificação do XKMS disponibiliza dois serviços distintos para auxiliar no gerenciamento de qualquer tipo de chaves (*e.g.*: simétricas ou assimétricas). Os dois serviços são o X-KISS (*XML Key Information Service Specification*) e o X-KRSS (*XML Key Registration Service Specification*). Estes serviços são abstratos e independentes da infra-estrutura de chaves usada.

O X-KISS permite que o receptor de uma mensagem assinada e ou cifrada delegue parte, ou todas as tarefas de processamento para este serviço. Assim as aplicações se abstraem de conhecer a ICP subjacente (*e.g.*: X.509, PGP – *Pretty Good Privacy*, SPKI/SDSI) deixando está interpretação a cargo de um serviço específico.

O serviço X-KISS suporta duas operações:

- ***Localizar:*** localiza informações relacionadas ao elemento <*ds:KeyInfo*> de um documento XML. As informações recebidas neste elemento são responsáveis por identificar, ou devolver, a chave usada na assinatura e ou cifragem de uma mensagem recebida. Este serviço pode retornar a chave desejada para se criar uma mensagem assinada e ou cifrada, destinada a um usuário particular.
- ***Validar:*** realiza um processo de validação nos dados recebidos para verificar se as informações relacionadas a chave são validas (*e.g.*: se a chave e ou certificado ainda não foi revogado) e retorna uma declaração sobre a situação da chave para o *principal*

que efetuou o pedido. Este serviço pode interagir com o serviço de localização para obter a chave necessária para executar a validação.

O serviço X-KRSS fornece um protocolo de mensagens pedido/resposta utilizado para o registro e gerenciamento de informações referentes a chaves. O *principal* pode requisitar ao serviço o registro e o armazenamento de uma chave pública ou par de chaves. Informações como nomes, identificadores, endereços de *e-mail*, etc., podem ser associadas com as chaves. Neste serviço, quatro operações podem ser feitas:

- ***Registrar:*** realiza o registro e a associação de informações a uma chave. O par de chaves pode ser gerado pela aplicação cliente ou pelo serviço de registro. Caso o par de chaves tenha sido gerado pela aplicação cliente, a mesma deve provar a posse da chave privada. Se a chave for gerada pelo serviço, este envia a chave privada, de alguma maneira considerada segura pelas políticas do serviço, para o cliente, podendo permanecer com uma cópia da mesma.
- ***Re-emitir:*** emitir as informações que estão associadas às chaves públicas, as quais foram registradas na operação de registro. O objetivo desse serviço e permitir gerar novas credenciais na ICP subjacente. Por exemplo, no caso de um certificado expirar.
- ***Revogar:*** revoga uma chave previamente registrada, ou seja, a chave não é mais confiável para o uso, por nenhuma aplicação.
- ***Recuperar:*** recupera a chave privada associada anteriormente. Este é semelhante à operação de registro e só pode ser executado quando a chave privada tiver sido gerada pelo X-KRSS.

O XKMS permite que os serviços que compõem sua estrutura sejam invocados de maneira síncrona, assíncrona ou pedido em duas fazes.

A.9. Caso de Uso com a SPML

A Figura A.6 mostra o relacionamento básico entre as entidades descritas na especificação SPML.

A RA (*Requesting Authority*) é um componente de *software* que emite pedidos SPML

para o PSP (*Provisioning Service Provider*, evento *1*). O PSP é o componente de *software* que atende, processa, e caso necessário, retorna resultados de pedidos SPML para o solicitante. O PST (*Provisioning Service Target*) representa o objetivo ou ponto final (*i.e.*: *endpoint*) que o provedor torna disponível (evento *2*) através da realização do provisionamento, ou configuração.

O objetivo (PST) não é o provedor (PSP), o solicitante (RA) envia requisições ao provedor para que este opere sobre os objetos (PSOs – *Provisioning Service Objects*) que estão e ou serão instanciados no PST.

O objetivo (PST) pode não ser a extremidade final (*i.e.*: *endpoint*), este pode representar uma conta de usuário, uma instância do serviço, ou uma coleção abstrata de pontos finais (PSOs, evento *3*).

O PSO pode representar uma informação referente a um objeto em um objetivo (PST). O provedor poderia representar como um objeto cada conta de usuário que o provedor gerência, com cada objeto possuindo um identificador único (PSO-ID) armazenado em um determinado PST, por exemplo.

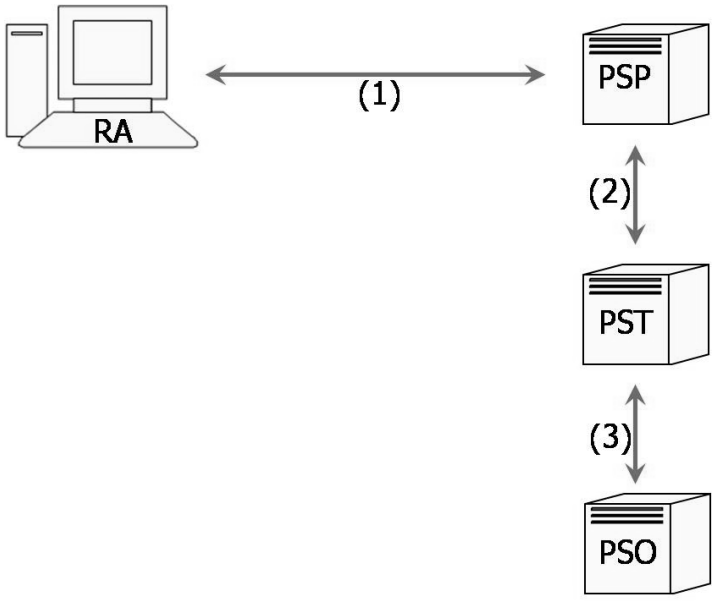

Figura A.6: Relacionamento entre as entidades segundo a SPML [29]

A Figura A.7 mostra o uso que a empresa Oracle [52] faz da linguagem SPML em alguns de seus produtos, visando facilitar o gerenciamento de identidades em ambientes que

cruzam domínios organizacionais.

Supondo que a empresa “A” contrate um novo funcionário (evento *1*, Figura A.7). O departamento de recursos humanos (*Human Resources* – HR), através de um serviço de provisionamento (SP1, evento *2*) gera uma mensagem SPML e a envia, através da Internet, para o serviço de provisionamento da empresa “B”. Neste caso, “B” é filial de “A” (SP2, evento *3*).

A empresa “B” dispõe dos recursos necessários ao novo funcionário da empresa “A”. Ao receber a mensagem SPML, o *software* de provisionamento da empresa “B” configura os direitos de acesso (evento *4*) para o funcionário da empresa “A”. Deste momento em diante, o novo funcionário terá uma conta de usuário em seu nome, podendo usufruir dos recursos expostos pela empresa “B” (evento *5*).

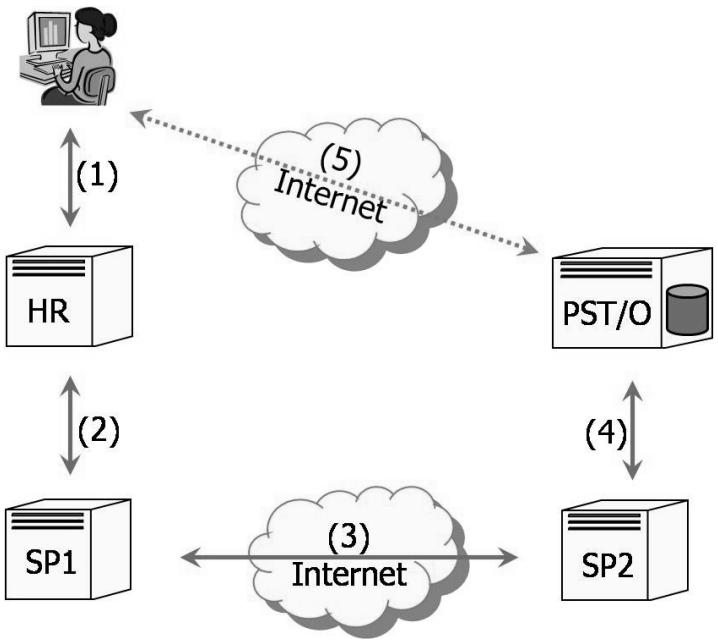

Figura A.7: Caso de uso com a SPML

Apêndice B

Modelos de Segurança e Controle de Políticas

B.1. SPKI/SDSI

B.1.1. Dinâmica de Acesso SPKI/SDSI

A infra-estrutura SPKI/SDSI preocupa-se com autorizar *principals* a executar operações em recursos protegidos. Para aplicar o controle de acesso em um recurso, o guardião do recurso estabelece uma ACL (*Access Control List* – ACL) para proteger este recurso (Seção B.1.5).

Quando o *principal* emite uma solicitação para desempenhar uma operação em um recurso (*e.g.*: ler um arquivo protegido), e para que o guardião possa honrar a solicitação, esta deve estar acompanhada por uma prova de autenticidade e uma prova de autorização.

Tipicamente a prova de autenticidade é o pedido assinado pelo *principal* e a prova de autorização é a cadeia de certificados autorizando o *principal* que assinou o pedido.

Na Figura B.1 mostra-se um exemplo de uso do protocolo *challenge-response* aplicado ao SPKI/SDSI [53]. O *principal* solicita acesso ao recurso protegido (evento *1*, Figura B.1) sem a prova de autorização ou autenticação. O guardião proíbe este pedido inicial e emite um "desafio" para o *principal*: "prove para mim que você tem as credenciais necessárias para executar a ação no objeto protegido pela ACL" (evento *2*).

Usando um algoritmo para descoberta de cadeias de certificado SPKI/SDSI, o *principal* gera uma cadeia de certificados, caso a mesma exista (evento *3*, Figura B.1). Se a cadeia existe, o *principal* assina o pedido e o envia junto com a cadeia de certificados, em um segundo pedido para o guardião (evento *4*). Esta é a resposta do *principal* para o desafio do guardião.

O guardião analisa o segundo pedido para verificar se este permite ao *principal* executar a ação. Verifica-se a assinatura do pedido (*i.e.*: autenticidade) e se a cadeia de certificados prove as permissões necessárias (*i.e.*: autorização, evento *5*), e somente então o guardião honra a solicitação de acesso (evento *6*).

O guardião não necessita conhecer a identidade do *principal* que está fazendo o pedido, esse só precisa saber que o *principal* em particular foi autorizado a executar a ação que está solicitando. O guardião baseia-se nas interações e na cadeia de certificados apresentada pelo *principal*.

Uma cadeia de certificados SPKI/SDSI é relativamente silenciosa sobre a identidade de seu portador.

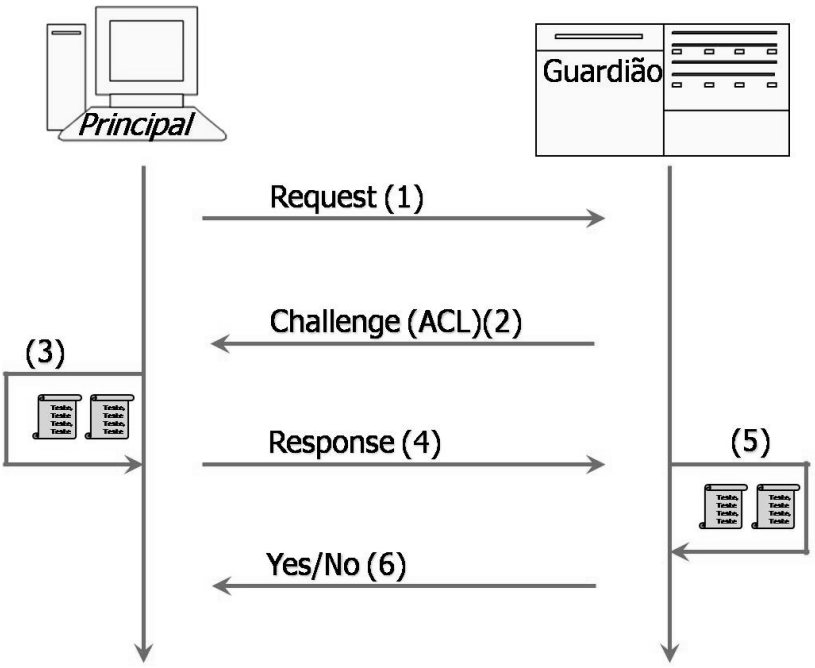

Figura B.1: Protocolo *challenge-response* [23] aplicado ao SPKI/SDSI

B.1.2. Threshold Subjects - TS

O TS prove tolerância à faltas, sendo somente usado em certificados de autorização para especificar um requisito que “k dentre n” chaves devem assinar um pedido antes de este ser honrado. Ou “k dentre n” chaves devem endossar um certificado de autorização para que o

mesmo possa ser delegado.

Sem uma quantidade de “k - 1” chaves for comprometida dentro de um conjunto “n” chaves, o adversário não será capaz de ganhar acesso a recursos protegidos. O adversário precisa persuadir um mínimo de “k” *principals* a assinar o pedido para convencer o guardião.

B.1.3. S-Expressions (Sexps)

A estrutura de dados utilizada na infra-estrutura SPKI/SDSI é chamada de *S-Expressions*. A *Sexps* usa uma representações em ASCII (*American Standard Code for Information Interchange*) que é legível a humanos (*i.e.*: *easy-to-read*), tendo assim um formato simples (*e.g.*: Figura 4.1). A representação ASCII de uma *Sexps* é referenciada como uma forma avançada [53].

B.1.4. Garantia do Certificado

A garantia do certificado SPKI/SDSI diz que: “este certificado é confiável até a data de expiração da validade”. Ou seja, a estrutura não faz uso de listas de revogação de certificados (*Certificate Revocation Lists* – CRL).

Para que um guardião possa proteger um recurso e fornecer acesso ao mesmo, basta o certificado não ter expirado a data de validade, então é garantido ser válido, não sendo necessário conferir se um certificado foi revogado. A responsabilidade sobre o certificado (*i.e.*: proteger a chave privada) é do *principal* que faz uso desta.

B.1.5. Lista de Controle de Acesso em SPKI/SDSI

As listas de controle de acesso (*Access Control List* – ACL) em SPKI/SDSI consistem em uma lista de entradas com os campos: sujeito, etiqueta e o *bit* de delegação. Cada entrada em uma ACL pode ser considerada como um certificado de autorização. O emissor do certificado é o proprietário da ACL, enquanto que o sujeito, a etiqueta e o *bit* de delegação são as especificidades da entrada. As ACLs usam um emissor especial “SELF”, representando o proprietário da ACL, assim, cada entrada na ACL é composta pela quíntupla “(SELF, S, T, p, V)”.

O SPKI/SDSI possui a capacidade de definir grupos, facilitando a administração das ACLs. Se um conjunto de *principals* com as mesmas características deveria ter acesso a certos recursos, cada recurso sendo protegido por uma ACL diferente, um certificado de grupo poderia ser criado, e o nome do grupo colocado em cada uma das ACLs.

Cada grupo possui um nome e uma quantidade de membros (*i.e.*: *principals*), sendo o proprietário do grupo quem decide qual sujeito pode ocupar um determinado cargo, emitindo um certificado de nome para o(s) *principal* apropriado.

B.2. Acordo em Nível de Serviço (SLA)

O SLA (*Service Level Agreement*) pode ser visto como um acordo em nível de serviço. Através do acordo se obtém o documento resultante de uma negociação entre o consumidor e o provedor de serviço. Este documento especifica o nível de disponibilidade, desempenho, operações ou outros atributos do serviço.

O SLA geralmente é escrito em uma terminologia de negócios de alto nível, devendo a mesma ser traduzida para um nível mais baixo, entendível pelos mecanismos que realizarão a avaliação das políticas (*i.e.*: *goals*). As classes definidas no PCIM (Seção 4.4) são projetadas para servir como a base para este esquema.

B.3. Caso de Uso com o Modelo *Outsourcing*

A RFC 2753 (*Request For Comments*) [36] fornece um exemplo de interação entre as entidades PEP e PDP durante uma simples avaliação de política. Essa se inicia quando o PEP recebe uma solicitação de acesso que necessita de uma avaliação de política. O PEP formula um pedido e o envia ao PDP que analisará o pedido e retornará o resultado da avaliação de política para o PEP. O PEP é responsável por aplicar a decisão, liberando ou negando a solicitação do *principal*.

De acordo com a RFC 2753, o PEP pode fazer uso (caso haja disponível) de um ponto de decisão de política local (LPDP – *Local Policy Decision Point*) para obter uma decisão de política local. Esta decisão parcial e o pedido de política original são enviados ao PDP, o qual tomará a decisão final, devolvendo a mesma para que o PEP a execute.

Os passos citados a seguir, com exceção do LPDP, podem ser seguidos na Figura B.2,

a qual mostra um fluxo de mensagens simples entre as entidades citadas. Quando um evento invoca o PEP referente a uma solicitação de acesso (evento *1*, Figura B.2), o PEP cria um pedido incluindo algumas informações retiradas da mensagem, descrevendo assim o pedido recebido.

O PEP, opcionalmente, pode consultar o repositório de configurações local para identificar o conjunto de elementos da política (conjunto A) que podem ser avaliadas localmente. O pedido com o conjunto A é passado para o ponto de decisão de política local (LPDP), obtendo assim um resultado parcial D(A), (eventos não representados).

O PEP encaminha o pedido com todas as informações retiradas do mesmo, e caso haja o D(A), para o PDP (evento *2*, Figura B.2). O PDP tomará a decisão adequada baseado em seu próprio conjunto de políticas (evento *3*) juntamente com as informações recebidas do PEP, obtendo assim uma decisão D(Q) (evento *4*). O resultado D(Q) pode ser combinado com o resultado parcial D(A), usando um algoritmo de combinação para cada decisão final.

O PDP retorna a decisão final para o PEP (evento *5*, Figura B.2) o qual aplicará a decisão (evento *6*) de aceitar ou negar o acesso ao recurso (evento *7*) à solicitação efetuada pelo *principal*. Se o resultado sobre o D(A) for negativo, então o PEP não precisa prosseguir para o PDP, porém, mesmo assim o PDP deve ser informado da falha de avaliação nas políticas locais (evento *2*).

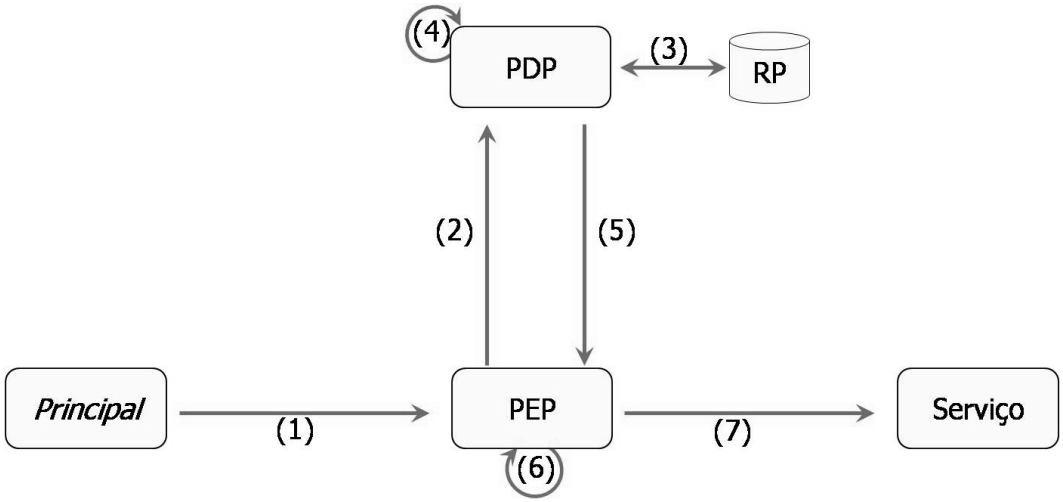

Figura B.2: Modelo *Outsourcing*

O mesmo se aplica quando o processamento da política tiver êxito, mas o controle de

entrada falhe (*e.g.*: devido a indisponibilidade do recurso, evento *7*, Figura B.2), novamente o PDP tem de ser informado da falha. O PDP, a qualquer momento, pode enviar uma notificação assíncrona para o PEP mudar uma decisão anterior ou para gerar uma mensagem de erro ou perigo.

B.4. Common Open Policy Service for Policy Provisioning (COPS-PR)

Diferente do protocolo COPS-PR [54] que usa conexão TCP (*Transmission Control Protocol*) entre as entidades PEP e PDP, na arquitetura de Serviços Web utilizam-se as mensagens SOAP. Caracterizando-se assim o fraco acoplamento entre as entidades PEP e PDP, e também para que estas possam estar dispostas em qualquer lugar de um sistema distribuído (*e.g.*: atrás de filtros de pacote). As mensagens SOAP transportarão como carga útil as avaliações e políticas necessárias para uma ou outra entidade (PEP e PDP).

Figura B.3: Modelo de provisionamento usado em [55]

As mensagens trocas entre as entidades (PEP e PDP, Figura B.3) de acordo com a arquitetura modelada em [55] são: conexão aberta com cliente (OPN), utilizada pelo PEP para informar ao PDP o tipo de cliente a ser configurado; conexão aceita (CAT), é a mensagem de

resposta a OPN tendo como função comunicar o aceite da conexão.

As operações comuns suportadas pelo protocolo COPS-PR correspondem a: Requisição (REQ), é a mensagem enviada do PEP para o PDP solicitando as políticas correspondentes; Decisão (DEC), é a mensagem enviada do PDP para o PEP com as políticas a serem implantadas; Estado (RPT), é a mensagem trocada entre PEP e PDP informando resultados relacionados as aplicações de políticas.

Após ter recebido as políticas, o PEP armazena estas em uma PIB (*Policy Information Base*) local para utilizá-las nas decisões de autorização posteriores.

B.5. Security Assertion Markup Language (SAML)

B.5.1. SAML Single Sign On

A especificação SAML [38] facilita o uso do esquema de *logon/logout* único, também referenciado como *Single Sign On/Off*. No primeiro o *principal* não necessita se autenticar toda vez que deseja executar alguma operação ou acessar algum serviço no domínio pelo qual se está navegando. Enquanto no segundo, todas as seções abertas pelo *principal*, em diferentes domínios, serão fechadas de maneira praticamente imediata quando o *logout* for executado, ou quando o tempo relacionado a tal asserção terminar.

B.5.2. Asserção de Autenticação SAML

A Figura B.4 mostra o exemplo de um pedido de autenticação SAML transportado dentro de uma mensagem SOAP. As linhas *1* a *21* delimitam envelope da mensagem, as linhas *2* a *20* delimitam o corpo da mensagem, e as linhas *3* a *19* descrevem o pedido.

A linha *4* é referente ao espaço de nomes do protocolo SAML, o qual está definido em um esquema XML. A linha *5* é um valor booleano indicando que o provedor de identidade deve autenticar quem está apresentando o pedido, ao invés de confiar em um contexto de segurança prévio.

A linha *6* indica o endereço para onde à mensagem de resposta deve ser retornada, ou seja, para qual serviço consumidor. A linha *7* é usada para declarar identificadores randômicos para asserções SAML, ou seja, o identificador retornado na resposta deve ser

igual ao do pedido. A linha *8* identifica a versão do protocolo SAML. A linha *9* carrega o momento em que o pedido foi emitido, codificado no formato UTC (*Universal Time Coordinated*).

A linha *10* contém uma referencia para o endereço destino para o qual a mensagem foi enviada. Este endereço evita encaminhamentos maliciosos de mensagens para receptores involuntários. Se estiver presente, o receptor deve conferir se a URI possui o endereço do destino correto. A linha *11* é referente ao sujeito sobre o qual a asserção será emitida. A linha *12* é referente ao espaço de nomes da asserção SAML, definida em um esquema XML.

As linhas *14* e *15* são referentes ao sistema de tipos, identificando o formato do atributo usado, no caso o endereço de *e-mail*. A linha *16* identifica o sujeito para qual a asserção de autenticação deverá ser emitida. As demais linhas enceram o escopo da mensagem.

```
1. <env:Envelope xmlns:env=”http://www.w3.org/2003/05/soap/envelope/”>
2.          <env:Body>
3.                <samlp:AuthnRequest
4.                xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
5.                ForceAuthn="true"
6.                AssertionConsumerServiceURL=http://www.example.com/
7.                ID="abe567de6"
8.                Version="2.0"
9.                IssueInstant="2005-01-31T12:00:00Z"
10.               Destination="http://www.example.com/">
11.                     <saml:Subject
12.                     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
13.                           <saml:NameID
14.                           Format="urn:oasis:names:tc:SAML:1.1:nameid-
15.                           format:emailAddress">
16.                           j.doe@company.com
17.                           </saml:NameID>
18.                     </saml:Subject>
19.               </samlp:AuthnRequest>
20.         </env:Body>
21. </env:Envelope>
```

Figura B.4: Pedido de autenticação SAML [38]